Data Mining

Exercise: Business Intelligence (Part 5)
Summer Term 2014
Stefan Feuerriegel

Today's Lecture

Objectives

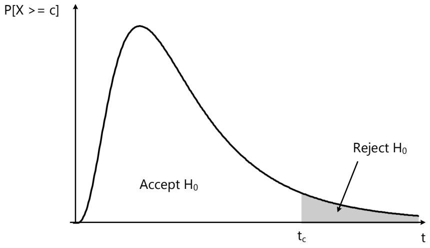
1. Recapitulating common concepts of machine learning
2. Understanding the k-nearest neighbor classification
3. Creating and pruning decision trees
4. Learning how k-means clustering works

Outline

Outline

Hypothesis Testing

- Results are called statistically significant if it has been predicted as unlikely to have occurred by chance alone, according to a pre-determined threshold probability, the significance level
- H_0: null hypothesis associated with a contradiction to a theory
- H_A: alternative hypothesis associated with a theory to prove
- *P*-value gives probability, assuming the null hypothesis is true, of observing a result at least as extreme as the test statistic t



Linear Models

- Linear Model: $\boldsymbol{y} = \alpha + \beta_1 \boldsymbol{x}_1 + \ldots + \beta_k \boldsymbol{x}_k + \boldsymbol{\varepsilon}$
  - Given $\boldsymbol{y}$ named observations, response or dependent variable
  - Given $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k$ named regressors, exogenous or independent variables
- Estimate intercept α and the coefficients $\beta_1, \ldots, \beta_k$ by minimizing error terms $\boldsymbol{\varepsilon}$, e. g. via ordinary least squares (OLS) estimator

$$\min_{\alpha,\beta_1,\ldots,\beta_k} \|\boldsymbol{\varepsilon}\| = \min_{\alpha,\beta_1,\ldots,\beta_k} \|\boldsymbol{y} - (\alpha + \beta_1 \boldsymbol{x}_1 + \ldots + \beta_k \boldsymbol{x}_k)\|$$

$\rightarrow$ important to test assumptions to avoid confounded results

Linear Regression Models

```
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      15.912      2.586    6.15  1.4e-05 ***
## d$PlayerValue     2.323      0.674    3.44   0.0033 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Estimate gives the least squares estimates of α and coefficients
- Std. Error shows standard errors $\hat{\sigma}_i$ of each coefficient estimate
- *t*-value and *P*-value columns test whether any of the coefficients might be equal to zero
  - *t*-statistic is calculated as $t = \beta_i / \hat{\sigma}_i$, if errors $\boldsymbol{\varepsilon}$ follow a normal distribution
    $\rightarrow$ large values of t indicate that the null hypothesis can be rejected and that the corresponding coefficient is not zero
  - *P*-value expresses the results of the hypothesis test as a significance level; conventionally, *P*-values smaller than 0.05 are taken as evidence that the coefficient is non-zero

Process: OLS Estimation

- The OLS technique imposes several assumptions in order for the method to give meaningful results
  1. Homoscedasticity means that the error term has the same variance σ^2 in each observation
  2. Non-Autocorrelation requires that the errors are uncorrelated between observations
  3. No Linear Dependence prerequisites regressors to all be linearly independent
- After verifying assumption, identify parameters with significant influence on outcome $\rightarrow$ t-value and P-value
- Look at overall model fit in terms of R^2, adjusted R^2 and F-test
- Select model that competes best in terms of information criterion
- Interpret magnitude and sign of coefficients, as well as significance level

Prediction with Linear Models

- An already estimated linear model $\boldsymbol{y} = \alpha + \beta_1 \boldsymbol{x}_1 + \ldots + \beta_k \boldsymbol{x}_k + \boldsymbol{\varepsilon}$ can be used to evaluate with new values $x'_1, \ldots, x'_k$ giving

$$y' = \alpha + \beta_1 x'_1 + \ldots + \beta_k x'_k$$

- Use the command `predict(m, newdata=d)` for a model `m` and new data `d`
- Example

```
m <- lm(Goals ~ PlayerValue, data = d)
nd <- data.frame(PlayerValue = 5)
predict(m, newdata = nd)
```

```
##      1
## 27.52
```

$\rightarrow$ the expected number of goals is 27.52

Outline

Machine Learning

- Principles, methods and algorithms for learning and prediction on the basis of past evidence
- Examples:
  - Speech recognition (e. g. Siri, speed-dialing)
  - Hand-writing recognition (e. g. letter delivery)
  - Fraud detection (e. g. credit cards)
  - Text filtering (e. g. spam filters)
  - Image processing (e. g. object tracking, Kinect)
  - Robotics (e. g. Google driverless car)

Machine Learning

- Goal: Learning to perform a task from experience
- Learning
  - We do not want to encode the knowledge ourselves
  - Machine should learn the relevant criteria automatically from past observations and adapt to the given situation
  - Tools: Statistics, probability theory, optimization
- Task
  - Often expressed as mathematical function

$$y = f(x, w)$$

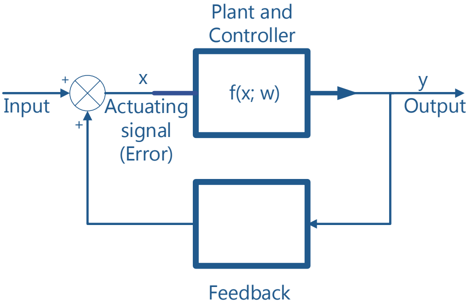
  - Input x, output y, parameter w (this is what is "learned")
  - Output y is either discrete or continuous
  - Selection of the "right" features w is crucial
  - Curse of dimensionality: Complexity increases exponentially with number of dimensions

→ from Schiele & Leibe (2010).

Task Learning: Examples

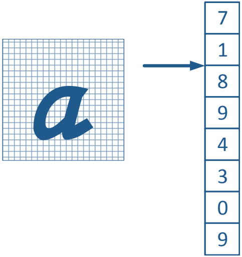
Regression with continuous output

- Automatic control of a vehicle



Classification with discrete output

- Email filtering
  $x \in [a-z]^+$
  $\mapsto y \in \{\text{important}, \text{spam}\}$
- Character recognition
  $x \in \mathbb{R}^n \mapsto y \in \{a, \ldots, z\}$



→ from Schiele & Leibe (2010).

Machine Learning

- Goal: Machines that learn to perform a task from experience
- Performance
  - Measured typically as one number
    $\rightarrow$ e. g. % correctly classified letters, % games won
  - "99 % correct classification"
    $\rightarrow$ Of what? Characters, words or sentences? Speaker/writer independent? Over what data set?
  - Example: "The car drives without human intervention 99 % of the time on country roads"
- Experience $\rightarrow$ what data is available?
  - Supervised learning ($\rightarrow$ data with labels)
  - Unsupervised learning ($\rightarrow$ data without labels)
  - Reinforcement learning ($\rightarrow$ with feedback/rewards)
- Most often learning = optimization
  - Search hypothesis space for the "best" function and model parameter w
  - Maximize $y = f(x, w)$ with respect to the performance measure

$\rightarrow$ from Schiele & Leibe (2010).

Supervised vs. Unsupervised Learning

Supervised learning

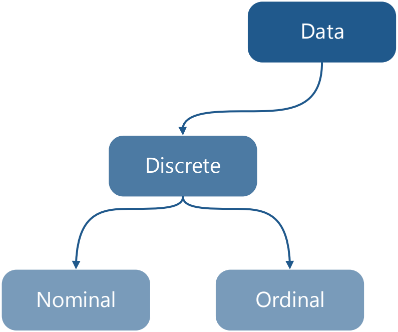
- Machine learning task of inferring a function from labeled training data
- Training data includes both the input and the desired results
  $\rightarrow$ correct results (target values) are given

Unsupervised learning

- Methods try to find hidden structure in unlabeled data
- The model is not provided with the correct results during the training
- No error or reward signal to evaluate a potential solution
- Examples:
  - Hidden Markov models
  - Dimension reduction (e. g. by principal component analysis)
  - Clustering (e. g. by k-means algorithm)
    $\rightarrow$ group into classes only on the basis of their statistical properties

Statistical Data Types

- Data type specifies semantic content of the variable
- Controls which probability distribution can be used
- Determines the type of regression analysis



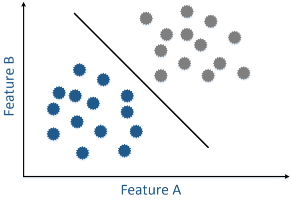
- Discrete variable can take on one of a limited (and usually fixed) number of possible values
- Ordinal: With natural ordering, e. g. grades (A, . . . , F)
- Nominal: Without this ordering, e. g. blood type (A, B, AB, 0)

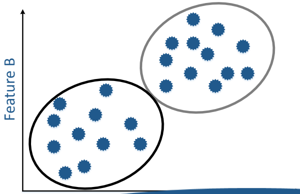
- If data cannot be described by a single number, called multivariate
  $\rightarrow$ e. g. vectors, matrices, sequences, networks

Taxonomy of Machine Learning

- Machine learning estimates function and parameter in $y = f(x, w)$
- Type of method varies depending on the nature of what is predicted

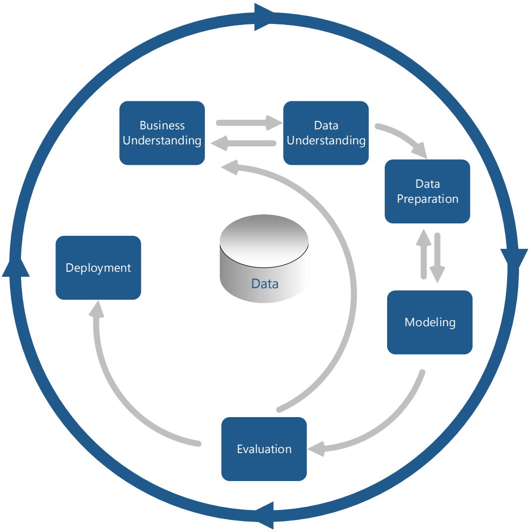
- Regression
  - Predicted value refers to a real number
  - Continuous y
- Classification
  - Predicted value refers to a class label
  - Discrete y (e. g. class membership)
- Clustering
  - Group points into clusters based on how "near" they are to one another
  - Identify structure in data





CRISP-DM

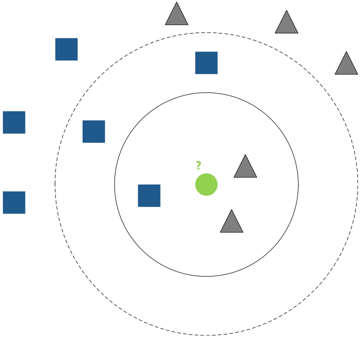
- Cross Industry Standard Process for Data Mining
- Data mining process model that describes commonly used approaches in practice
- Data understanding and data preparation require most time



Outline

k-Nearest Neighbor (k-NN) Classification

- Input: training examples as vectors in a multidimensional feature space, each with a class label
- No training phase to calculate internal parameters
- Testing: Assign to class according to k-nearest neighbors
- Classification as majority vote
- Problematic
  - Skewed data
  - Uneven frequency of classes



$\rightarrow$ What label to assign to the circle?

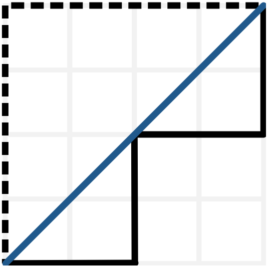
Distance Metrics

- Distance metrics measure distance between two points
- Given points $\boldsymbol{p} = [p_1, \ldots, p_N] \in \mathbb{R}^N$ and $\boldsymbol{q} = [q_1, \ldots, q_N] \in \mathbb{R}^N$
- Arbitrary distance metric $d(\boldsymbol{p}, \boldsymbol{q})$
- Euclidean distance

$$d_2(\boldsymbol{p}, \boldsymbol{q}) = \|\boldsymbol{q} - \boldsymbol{q}\|_2 = \sqrt{\sum_{i=1}^{N} (q_i - p_i)^2}$$

- Manhattan distance

$$d_1(\boldsymbol{p}, \boldsymbol{q}) = \|\boldsymbol{q} - \boldsymbol{q}\|_1 = \sum_{i=1}^{N} |q_i - p_i|$$

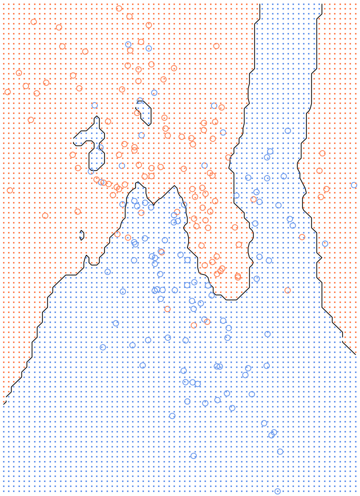


Blue $\rightarrow$ Euclidean

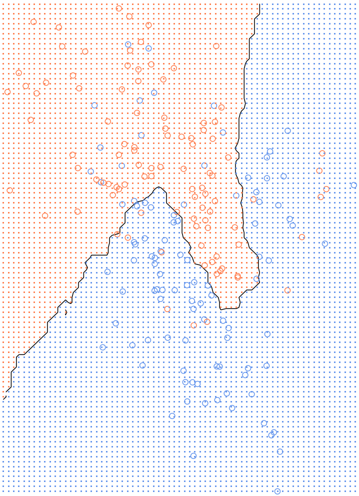
Black $\rightarrow$ Manhattan

Choosing Number of Nearest Neighbors *k*

5-Nearest Neighbor



15-Nearest Neighbor



k-Nearest Neighbor Classification

BI Case Study

Question: Can we predict the credit scoring of consumers based on past applications?

Training Data: Past Applications

- Age in years, income in €1000, number of credit cards

```
age <- c(35, 22, 63, 59, 25, 37)
income <- c(35, 50, 200, 170, 40, 50)
creditcards <- c(3, 2, 1, 1, 4, 6)
train <- as.data.frame(cbind(age, income, creditcards))
```

- Corresponding credit scoring

```
scoring <- c("Bad", "Good", "Bad", "Bad", "Good", "Good")
```

k-NN Classification in R

- Loading required library `class`

```
library(class)
```

- Predictions via `knn(train, test, labels, k)` for `test` data using historic observations `train` with corresponding `labels`
- Predict scoring for person (age 37, €50000 income, 2 credit cards)

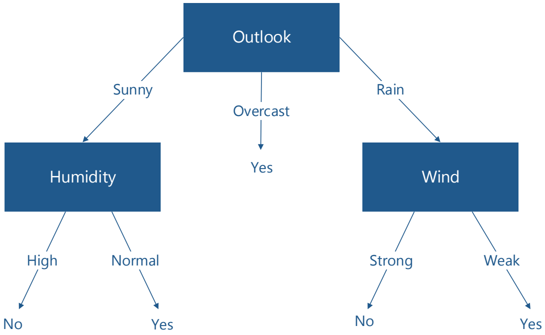
```
# With 1-nearest neighbor
knn(train, c(37, 50, 2), scoring, 1)

## [1] Good
## Levels: Bad Good

# With 3-nearest neighbor
knn(train, c(37, 50, 2), scoring, 3)

## [1] Good
## Levels: Bad Good
```

- Output: predicted label in 1st row out of all possible labels (2nd row)

Outline

Decision Trees

- Flowchart-like structure in which nodes represent tests on attributes
- End nodes (leaves) of each branch represent class labels
- Example: Decision tree for playing tennis



Decision Trees

- Issues
  - How deep to grow?
  - How to handle continuous attributes?
  - How to choose an appropriate attributes selection measure?
  - How to handle data with missing attributes values?
- Advantages
  - Simple to understand and interpret
  - Requires only few observations
  - Words, best and expected values can be determined for different scenarios
- Disadvantages
  - Information Gain criterion is biased in favor of attributes with more levels
  - Calculations become complex if values are uncertain and/or outcomes are linked

Decision Trees in R

- Loading required libraries `rpart`, `party` and `partykit`

```
library(rpart)
library(party)
library(partykit)
```

- Accessing credit scores

```
library(caret)
data(GermanCredit)
```

- Split into training and testing data

```
samps <- runif(nrow(GermanCredit))
train <- GermanCredit[-(1:10), ]
test <- GermanCredit[1:10, ]
```

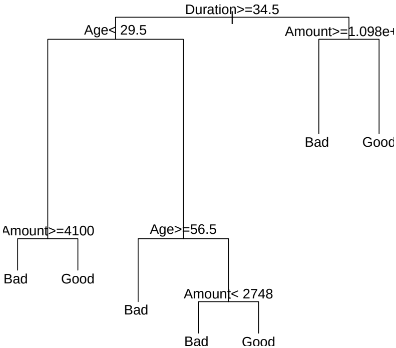
- Building a decision tree with
  `rpart(formula, method="class", data=d)`

```
dt <- rpart(Class ~ Duration + Amount + Age,
            method="class", data=train)
```

Decision Trees in R

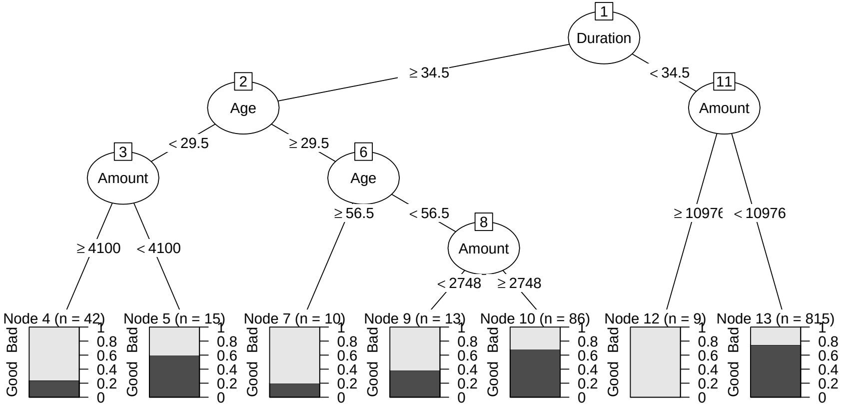
- Plot decision tree using `plot(dt)` and `text(dt)`

```
plot(dt)
text(dt)
```



Drawing Decision Trees Nicely

```r
plot(as.party(dt))
```



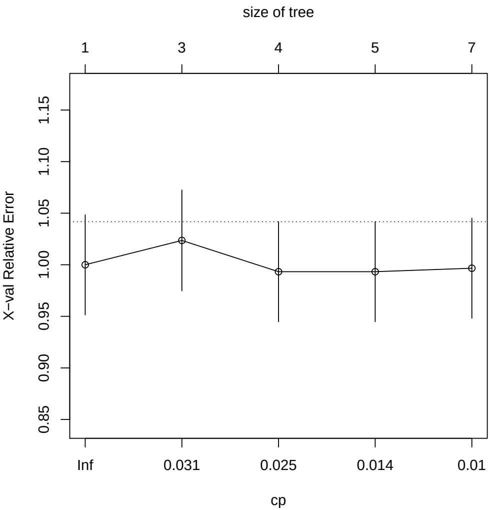
Complexity Parameter

```
printcp(dt)

##
## Classification tree:
## rpart(formula = Class ~ Duration + Amount + Age, data = train,
##     method = "class")
##
## Variables actually used in tree construction:
## [1] Age      Amount   Duration
##
## Root node error: 297/990 = 0.3
##
## n= 990
##
##      CP nsplit rel error xerror  xstd
## 1 0.032      0      1.00   1.00 0.049
## 2 0.030      2      0.94   1.02 0.049
## 3 0.020      3      0.91   0.99 0.048
## 4 0.010      4      0.89   0.99 0.048
## 5 0.010      6      0.87   1.00 0.049
```

- Rows show results for trees with different numbers of nodes
- Cross-validation error in column `xerror`
- Complexity parameter in column `CP`, similar to number of nodes

Complexity Parameter

```r
plotcp(dt)
```



Pruning Decision Trees

- Reduce tree size by removing nodes with little predictive power
- Aim: Minimize cross-validation error in column `xerror`

```
m <- which.min(dt$cptable[, "xerror"])
```

- Optimal size of nodes

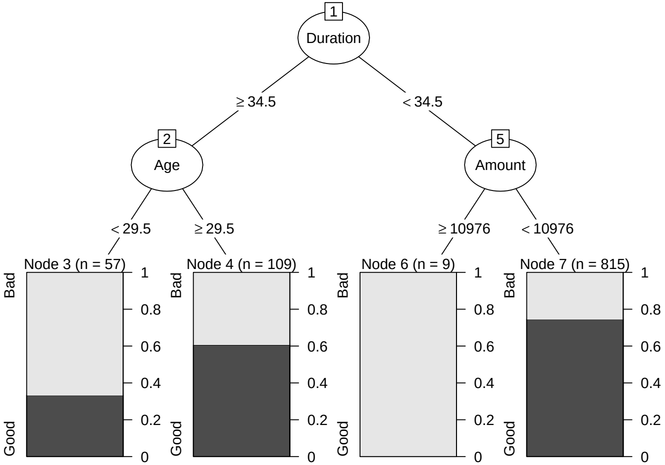
```
m

## 3
## 3
```

- Choose corresponding complexity parameter

```
dt$cptable[m, "CP"]

## [1] 0.0202
```

Pruning Decision Trees

```
p <- prune(dt, cp = dt$cptable[which.min(dt$cptable[, "xerror"]), "CP"])
plot(as.party(p))
```



Prediction with Decision Trees

- `predict(dt, test, type="class")` predicts classes on new data `test`

```
pred <- predict(p, test, type = "class")
pred

##    1    2    3    4    5    6    7    8    9   10
## Good  Bad Good Good Good Good Good Good Good Good
## Levels: Bad Good
```

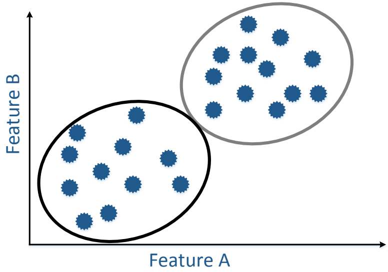
- Output: predicted label in 1st row out of all possible labels (2nd row)
- Confusion matrix via `table(pred=pred_classes, true=true_classes)`

```
# horizontal: true class; vertical: predicted class
table(pred = pred, true = test$Class)

##       true
## pred   Bad Good
##   Bad    1    0
##   Good   2    7
```

Outline

k-Means Clustering

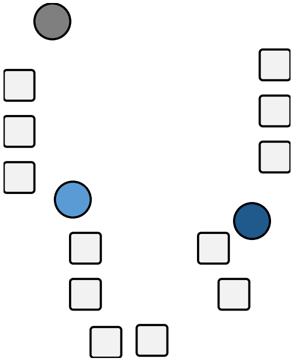
- Partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype for the cluster



- Computationally expensive; instead, we use efficient heuristics
- Default: Euclidean distance as metric and variance as a measure of cluster scatter

Lloyd's Algorithm: Outline

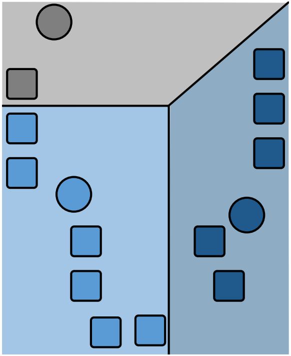
1. Randomly generated k initial "means" (here: $k = 3$)



2. Create k clusters by associating every observation with the nearest mean (colored partitions)
3. Centroid of each of the k clusters becomes the new mean
4. Repeat steps 2 and 3 until convergence

Lloyd’s Algorithm: Outline

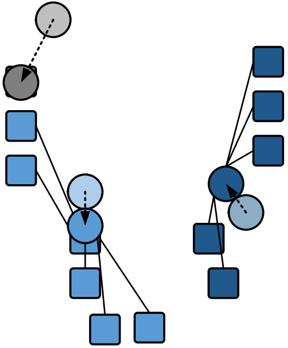
1. Randomly generated k initial "means" (here: $k = 3$)
2. Create k clusters by associating every observation with the nearest mean (colored partitions)



3. Centroid of each of the k clusters becomes the new mean
4. Repeat steps 2 and 3 until convergence

Lloyd's Algorithm: Outline

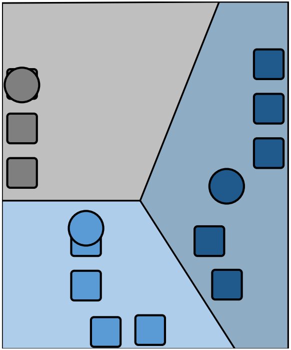
1. Randomly generated k initial "means" (here: $k = 3$)
2. Create k clusters by associating every observation with the nearest mean (colored partitions)
3. Centroid of each of the k clusters becomes the new mean



4. Repeat steps 2 and 3 until convergence

Lloyd's Algorithm: Outline

1. Randomly generated k initial "means" (here: $k = 3$)
2. Create k clusters by associating every observation with the nearest mean (colored partitions)
3. Centroid of each of the k clusters becomes the new mean
4. Repeat steps 2 and 3 until convergence



Lloyd's Algorithm: Pseudocode

1. **Initialization**
   Choose a set of k means $\mathbf{m}_1^{(1)}, \ldots, \mathbf{m}_k^{(1)}$ randomly
2. **Assignment Step**
   Assign each observation to the cluster whose mean is closest to it, i.e.

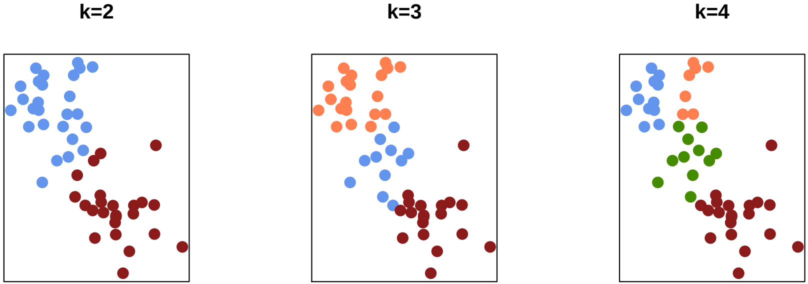
$$S_i^{(t)} = \left\{ \mathbf{x}_p : \left\| \mathbf{x}_p - \mathbf{m}_i^{(t)} \right\| \leq \left\| \mathbf{x}_p - \mathbf{m}_j^{(t)} \right\| \; \forall\, 1 \leq j \leq k \right\}$$

   where each observation is assigned to exactly one cluster, even if it could be is assigned to two or more of them
3. **Update Step**
   Calculate the new means to be the centroids of the observations in the new clusters

$$\mathbf{m}_i^{(t+1)} = \frac{1}{\left| S_i^{(t)} \right|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j$$

Optimal Choice of k

Example: Plots show the results of applying k-means clustering with different values of k



Note: Final results can vary according to random initial means!
$\rightarrow$ In practice, k-means clustering will be performed using multiple random assignments and only the best result is reported

Optimal Choice of k

- Optimal choice of k searches for a balance between maximum compression ($k = 1$) and maximum accuracy ($k = n$)
- Diagnostic checks to determine the number of clusters, such as
  1. Simple rule of thumb sets $k \approx \sqrt{n/2}$
  2. Elbow Method: Plot percent of explained variance vs. number of clusters
  3. Usage of information criteria
  4. ...
- k-means minimizes the within-cluster sum of squares (WCSS)

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{\boldsymbol{x}_j \in S_i} \|\boldsymbol{x}_j - \boldsymbol{\mu}_i\|^2$$

with clusters $S = \{ S_1, \ldots, S_k \}$ and mean points $\boldsymbol{\mu}_i$ in S_i

k-Means Clustering in R

- Prepare 2-dimensional sample data

```r
d <- cbind(c(1, 2, 4, 5), c(1, 1, 3, 4))
```

- Call *k*-means via `kmeans(d, k, nstart=n)` with `n` initializations to get cluster means

```r
km <- kmeans(d, 2, nstart = 10)
km

## K-means clustering with 2 clusters of sizes 2, 2
##
## Cluster means:
##   [,1] [,2]
## 1  4.5  3.5
## 2  1.5  1.0
##
## Clustering vector:
## [1] 2 2 1 1
##
## Within cluster sum of squares by cluster:
## [1] 1.0 0.5
##  (between_SS / total_SS =  91.0 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"
```

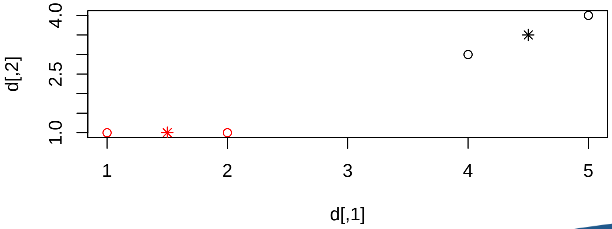
k-Means Clustering in R

- Calculate within-cluster sum of squares (WCSS) via

```
sum(km$tot.withinss)

## [1] 1.5
```

- Plot dataset as circles colored (`col=`) according to calculated cluster
- Add cluster centers `km$centers` as stars (`pch=8`)

```
plot(d, col = km$cluster)
points(km$centers, col = 1:nrow(km$centers), pch = 8)
```



Clustering

Research Question

Group countries based on income, literacy, infant mortality and life expectancy (file: `countries.csv`) into three groups accounting for developed, emerging and undeveloped countries.

```r
# Use first column as row names for each observation
d <- read.csv("countries.csv", header = TRUE, sep = ",", row.names = 1)
head(d)
```

```
##            Per.capita.income Literacy Infant.mortality Life.expectancy
## Brazil                 10326     90.0            23.60            75.4
## Germany                39650     99.0             4.08            79.4
## Mozambique               830     38.7            95.90            42.1
## Australia              43163     99.0             4.57            81.2
## China                   5300     90.9            23.00            73.0
## Argentina              13308     97.2            13.40            75.3
```

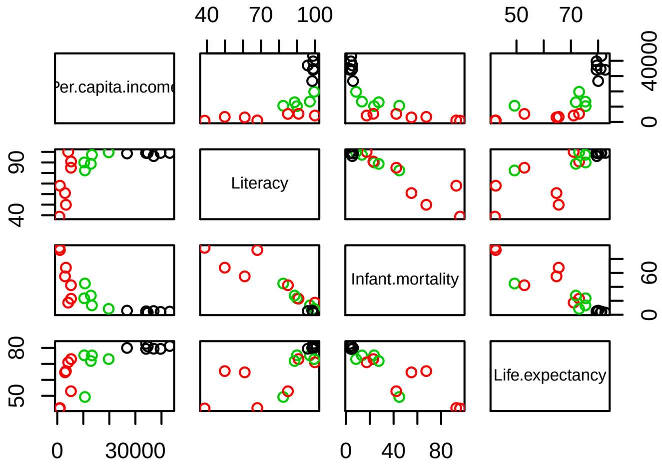
Clustering

```r
km <- kmeans(d, 3, nstart = 10)
km
```

```
## K-means clustering with 3 clusters of sizes 7, 7, 5
##
## Cluster means:
##   Per.capita.income Literacy Infant.mortality Life.expectancy
## 1             35642    98.50            4.477           80.43
## 2              3267    70.50           56.251           58.80
## 3             13370    91.58           23.560           68.96
##
## Clustering vector:
##         Brazil        Germany     Mozambique      Australia          China
##              3              1              2              1              2
##      Argentina United Kingdom   South Africa         Zambia        Namibia
##              3              1              3              2              2
##        Georgia       Pakistan          India         Turkey         Sweden
##              2              2              2              3              1
##      Lithuania         Greece          Italy          Japan
##              3              1              1              1
##
## Within cluster sum of squares by cluster:
## [1] 158883600  20109876  57626083
##  (between_SS / total_SS =  94.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"
## [5] "tot.withinss" "betweenss"    "size"
```

Visualizing Results of Clustering

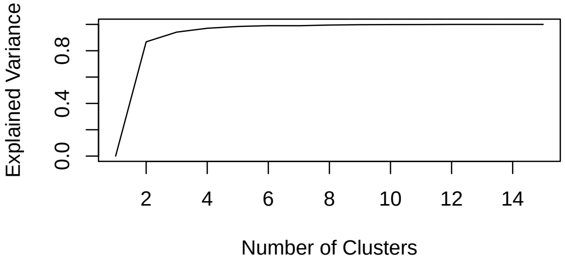
```
plot(d, col = km$cluster)
```



Elbow Plot to Choose k

Choose k (here: $k = 3$) so that adding another cluster doesn't result in much better modeling of the data

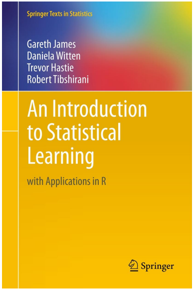
```r
ev <- c()
for (i in 1:15) {
    km <- kmeans(d, i, nstart = 10)
    ev[i] <- sum(km$betweenss)/km$totss
}
plot(1:15, ev, type = "l", xlab = "Number of Clusters", ylab = "Explained Variance")
```

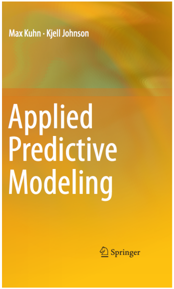


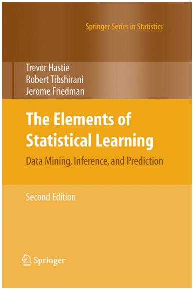
Outline

Books on Machine Learning with R

- James, Witten, Hastie & Tibshirani. An Introduction to Statistical Learning: with Applications in R. Springer, 2013.
- Kuhn & Johnson. Applied Predictive Modeling. Springer, 2013.
- Hastie, Tibshirani & Friedman. The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, 2013.







Summary: Data Mining

Classification, Regression & Clustering

Classification	Predict discrete values / class labels
Regression	Predict continuous values / real numbers
Clustering	Group nearby data into clusters

Supervised vs. Unsupervised

Supervised	Learning function from labeled training data
Unsupervised	Find structure in unlabeled data

Commands

`knn(train, test, labels, k)`	*k*-nearest neighbor classification
`rpart()`	Creating decision tree
`prune()`	Pruning decision tree
`kmeans(d, k, nstart=n)`	*k*-means clustering: partition data into similar groups
Elbow plot	Determines optimal number of clusters *k*