Data Mining Overview

What is Data Mining?

- Use of algorithms and computers to discover novel and interesting patterns within data
- Diapers and beer
- Machine learning

Data Mining: Four Processes

1. Data preparation
2. Exploratory data analysis
3. Model development
4. Interpretation of results

→Iterative process vs. linear

Data Mining: Four Processes

1. Data preparation
   a) Most time-consuming
   b) Data organization
   c) Data completeness
   d) Data accuracy
   e) Data transformation
2. Exploratory data analysis
3. Model development
4. Interpretation of results

Data Mining: Four Processes

1. Data preparation
2. Exploratory data analysis
   a) Preliminary data context assessment
   b) Visualization analysis
   c) Key values for parameters
3. Model development
4. Interpretation of results

Data Mining: Four Processes

1. Data preparation
2. Exploratory data analysis
3. Model development
   a) Most complex/interesting
   b) Test selection of data mining techniques
   c) Example: “association rules mining”
4. Interpretation of results

Data Mining: Four Processes

1. Data preparation
2. Exploratory data analysis
3. Model development
4. Interpretation of results
   a) Making sense of data mining output
   b) Determine actionable conclusions

Data Mining: Four Processes

1. Data preparation
2. Exploratory data analysis
3. Model development
4. Interpretation of results

Question

→ How is this different from the overall data science process?

Associative Rule Mining

First Look at Data Mining

Data mining is an area of research and practice that is focused on discovering novel patterns in data. As usual, R has lots of possibilities for data mining. In this lecture we will begin experimentation with essential data mining techniques by trying out one of the easiest methods to understand: association rules mining. More beer and diapers, please!

What is Association Rules Mining

Association rules:

- Association rules are if/then statements.
- The rules help uncover relationships between seemingly unrelated data.

→ An example

“If a customer buys a dozen eggs, that person is 80% likely to also purchase milk.”

Example Dataset

- A row is a “basket” of items from one purchase

citrus fruit, semi-finished bread, margarine ,ready soups

tropical fruit, yogurt, coffee

whole milk

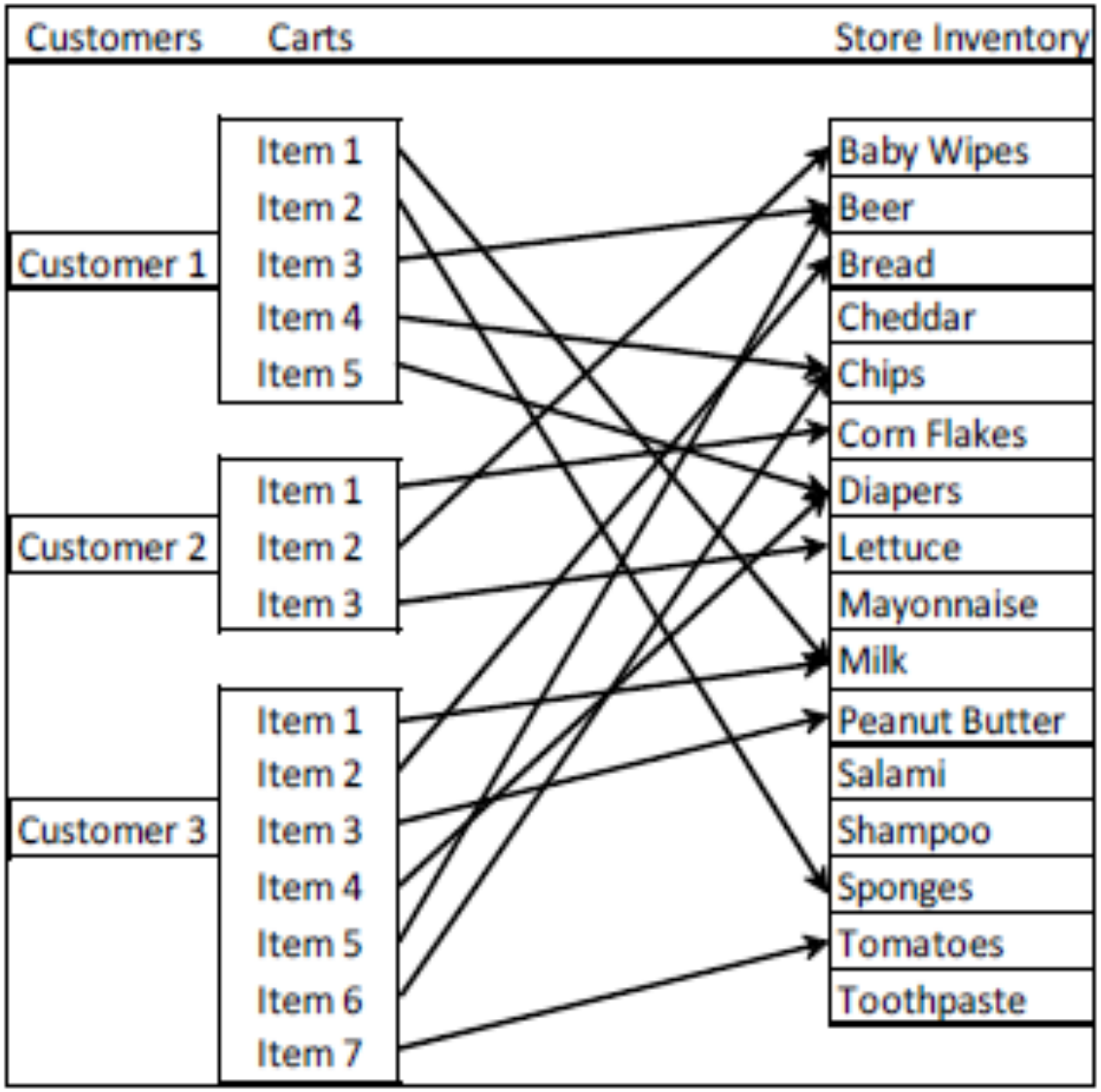
pip fruit, yogurt, cream cheese, meat spreads

other vegetables, whole milk, condensed milk

whole milk, butter, yogurt, rice, abrasive cleaner

rolls/buns

Use case: supermarket grocery cart



Support / Confidence / Lift

- **Support rule**
  - Proportion that a paring occurs across all baskets

  → # of rows having both A AND B / Total # of rows
- **Confidence quantity**
  - How frequently a particular pair occurs among all the items when the first item is present

  →# of rows having both A AND B / # of rows with A
- **Lift: Confidence / Probability of second item**
  - Confidence / Expected confidence

  Expected confidence = # of rows with B / Total # of rows

An Example: Rule for Diapers → Beer

- Support value: 0.67
  - Diapers and beer occurred in two out of three carts
- Confidence value: 0.5
  - Beer occurred 50% of the time diapers were in the cart
- Lift value: 2.5 (probability of beer in cart: 20%) [0.5/0.2]

Questions:

What are some **examples** of association rules mining "**in the real world**"?

What might be some **possible issues** with using this algorithm?

Associative Rule Mining in R

The DM Process – Data Prep

- **Data Mining Process 1: Data Preparation**

  install.packages("arules")

  library("arules")

  - Groceries data set
  - Ready to be analyzed

The DM Process – Explore

- **Data Mining Process 2: Exploratory Data Analysis**

  data(Groceries)

  summary(Groceries)

The DM Process – Explore/View

```
> summary(Groceries)
transactions as itemMatrix in sparse format with
 9835 rows (elements/itemsets/transactions) and
 169 columns (items) and a density of 0.02609146

most frequent items:
      whole milk other vegetables       rolls/buns             soda           yogurt          (Other)
            2513             1903             1809             1715             1372            34055

element (itemset/transaction) length distribution:
sizes
   1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16   17   18   19   20   21   22   23
2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46   29   14   14    9   11    4    6
  24   26   27   28   29   32
   1    1    1    1    3    1

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  1.000   2.000   3.000   4.409   6.000  32.000
```

The DM Process – Data Summary

- **Process 2: Exploratory Data Analysis**
  - Groceries data set –> context
    - itemMatrix, sparse format
    - 9,835 rows, 169 columns
    - Row = basket/grocery cart
    - Column = grocery item/product
    - Item in grocery basket indication -> 1, 0
    - Items occurring most frequently

The DM Process – itemFrequencyPlot

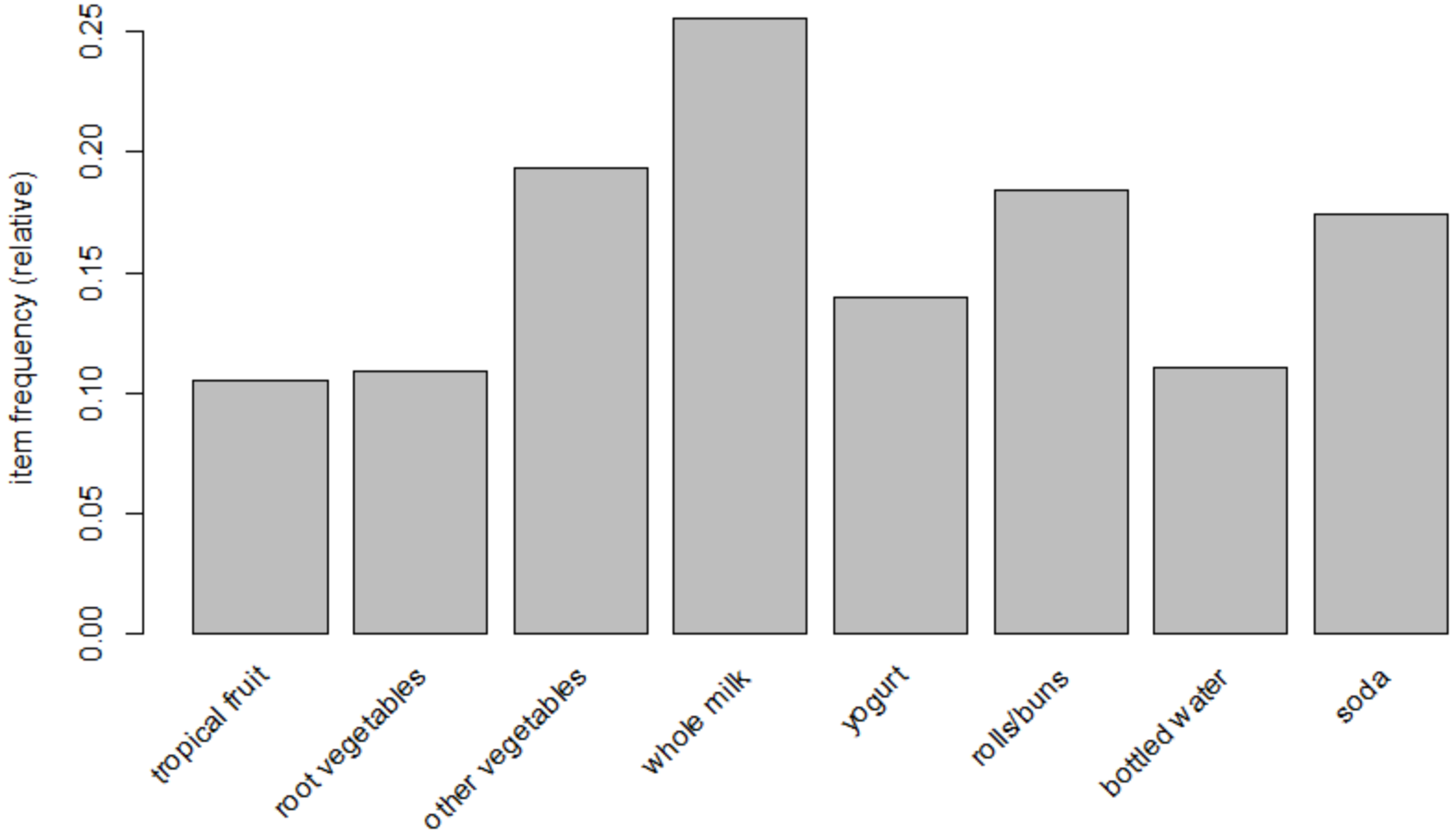
- **Process 2: Exploratory Data Analysis**
  - Groceries data set –> context
    - itemFrequencyPlot() function
    - Need to specify support parameter/ argument

    Example:

    itemFrequencyPlot(Groceries, support=0.1)

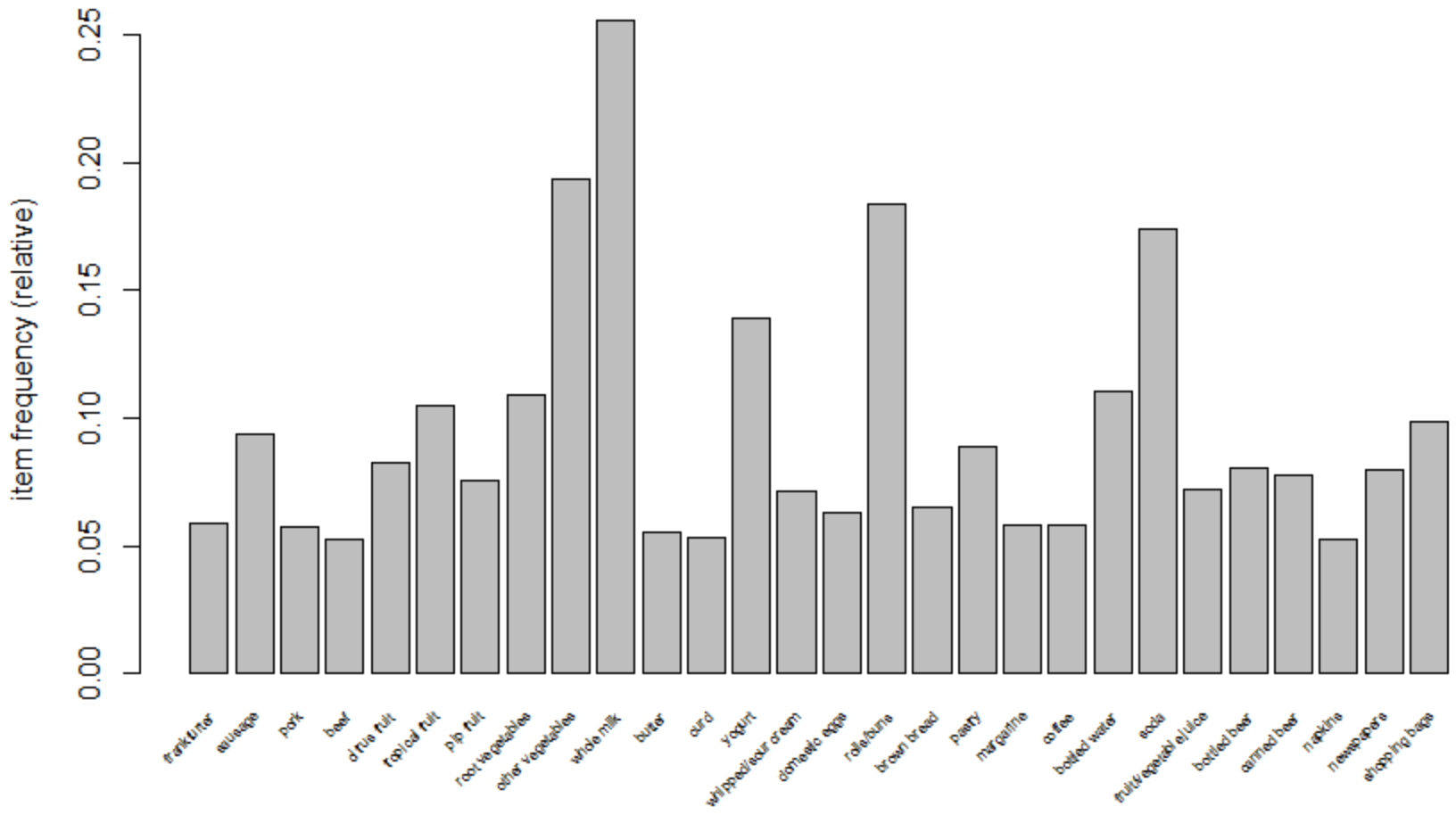
The DM Process – itemFrequencyPlot

```
itemFrequencyPlot(Groceries,support=0.1)
```



The DM Process – itemFrequencyPlot

```
itemFrequencyPlot(Groceries,support=0.05,cex.names=0.5)
```



The DM Process – Model Development

- Generate rules
  - Values of support
  - Focus on items with meaningful frequency
  - apriori() command
    - Finding rules in transaction data
    - Rules format: Equation context: LHS (lefthand side), RHS (righthand side)
  - RHS/LHS
    - RHS: one item
    - LHS: multiple items

The DM Process – Support/Lift/Confidence

- Support for a rule
  - Frequency of co-occurrence: LHS & RHS together
  - Ex: milk & butter occur together in 10% of carts
- Confidence of a rule
  - Proportion of time that LHS & RHS occur together vs. the total # of appearances of LHS
  - If milk by itself, occurs in 25% of the carts, then milk/butter confidence is .40 (.10/.25)
- Lift
  - Confidence / the probability of the RHS occurring

Using Apriori

```
apriori(Groceries,parameter=list(support=0.005,confidence=0.5))
```

```
parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen target   ext
        0.5    0.1    1 none FALSE            TRUE   0.005      1     10  rules FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [120 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [120 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
set of 120 rules
```

Iterating Using Apriori

- Change support value.
- Rerun apriori.
- Store resulting rules.
- Review output via summary function.

Looking at the Rules

```
ruleset <- apriori(Groceries,
parameter=list(support=0.01,confidence=0.5))
summary(ruleset)
```

```
set of 15 rules

rule length distribution (lhs + rhs):sizes
 3
15

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      3       3       3       3       3       3

summary of quality measures:
    support         confidence          lift
 Min.   :0.01007   Min.   :0.5000   Min.   :1.984
 1st Qu.:0.01174   1st Qu.:0.5151   1st Qu.:2.036
 Median :0.01230   Median :0.5245   Median :2.203
 Mean   :0.01316   Mean   :0.5411   Mean   :2.299
 3rd Qu.:0.01403   3rd Qu.:0.5718   3rd Qu.:2.432
 Max.   :0.02227   Max.   :0.5862   Max.   :3.030

mining info:
      data ntransactions support confidence
 Groceries          9835    0.01        0.5
```

Looking at the Rules

```
> inspect(ruleset)
    lhs                          rhs                    support confidence     lift
1   {curd,
     yogurt}                  => {whole milk}       0.01006609  0.5823529 2.279125
2   {other vegetables,
     butter}                  => {whole milk}       0.01148958  0.5736041 2.244885
3   {other vegetables,
     domestic eggs}           => {whole milk}       0.01230300  0.5525114 2.162336
4   {yogurt,
     whipped/sour cream}      => {whole milk}       0.01087951  0.5245098 2.052747
5   {other vegetables,
     whipped/sour cream}      => {whole milk}       0.01464159  0.5070423 1.984385
6   {pip fruit,
     other vegetables}        => {whole milk}       0.01352313  0.5175097 2.025351
7   {citrus fruit,
     root vegetables}         => {other vegetables} 0.01037112  0.5862069 3.029608
8   {tropical fruit,
     root vegetables}         => {other vegetables} 0.01230300  0.5845411 3.020999
9   {tropical fruit,
     root vegetables}         => {whole milk}       0.01199797  0.5700483 2.230969
```

Question:

Why do you think there are so many rules focused on milk?

Associative Rule Mining in R

Data Science

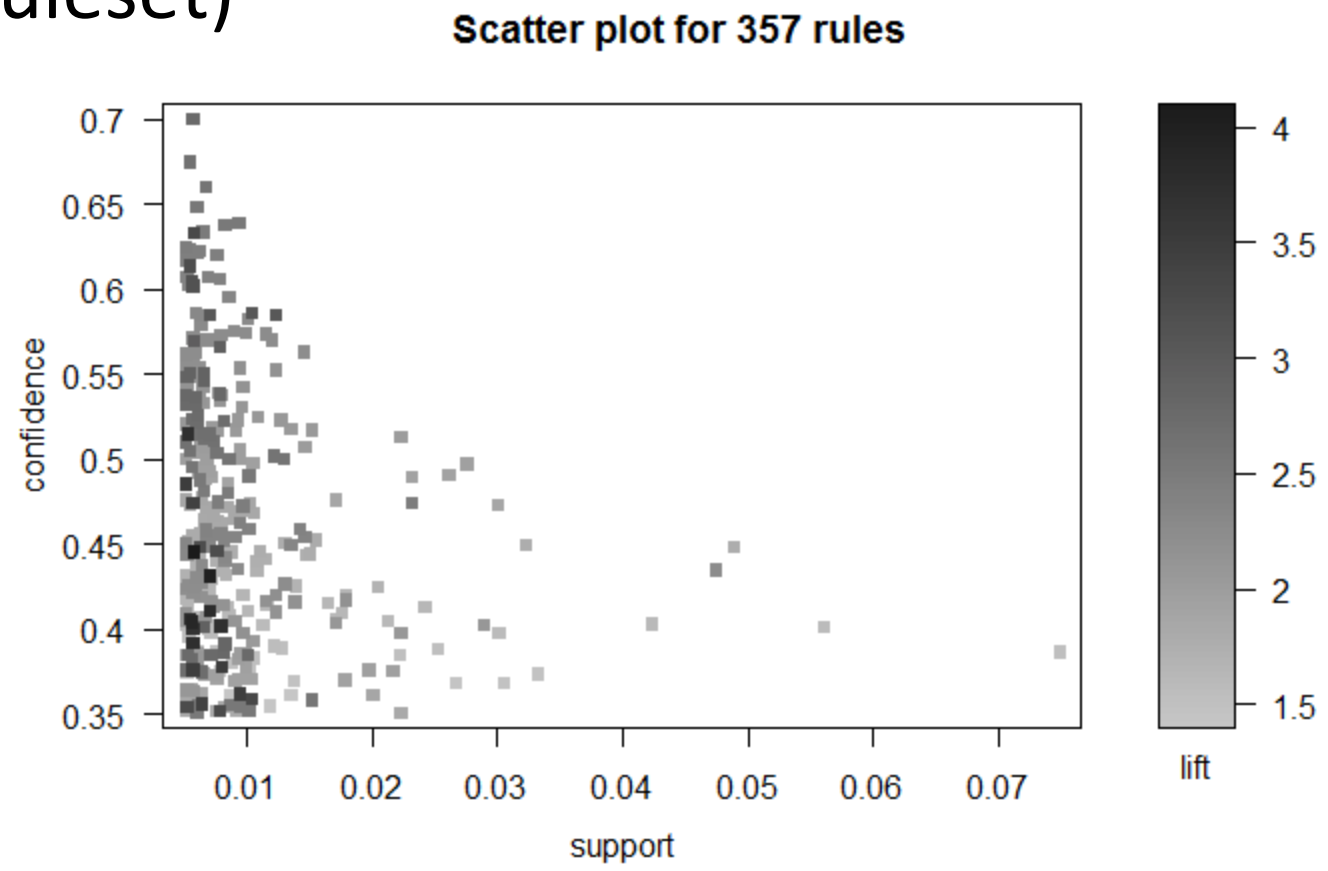
```
#
>install.packages("arulesViz")
>library(arulesViz)
#
>ruleset <-
apriori(Groceries,parameter=list(support=0.005,confidence=0.35))
>plot(ruleset)
#
```

Data Science

```
> ruleset <- apriori(Groceries,parameter=list(support=0.005,confidence=0.35))

parameter specification:
 confidence minval smax arem  aval originalSupport support minlen maxlen
       0.35    0.1    1 none FALSE            TRUE   0.005      1     10
 target   ext
  rules FALSE

algorithmic control:
 filter tree heap memopt load sort verbose
    0.1 TRUE TRUE  FALSE TRUE    2    TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
set item appearances ...[0 item(s)] done [0.00s].
set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
sorting and recoding items ... [120 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [357 rule(s)] done [0.00s].
creating S4 object  ... done [0.00s].
```

Data Science

plot(ruleset)



Data Science

Data Mining Process 4: Interpreting Results

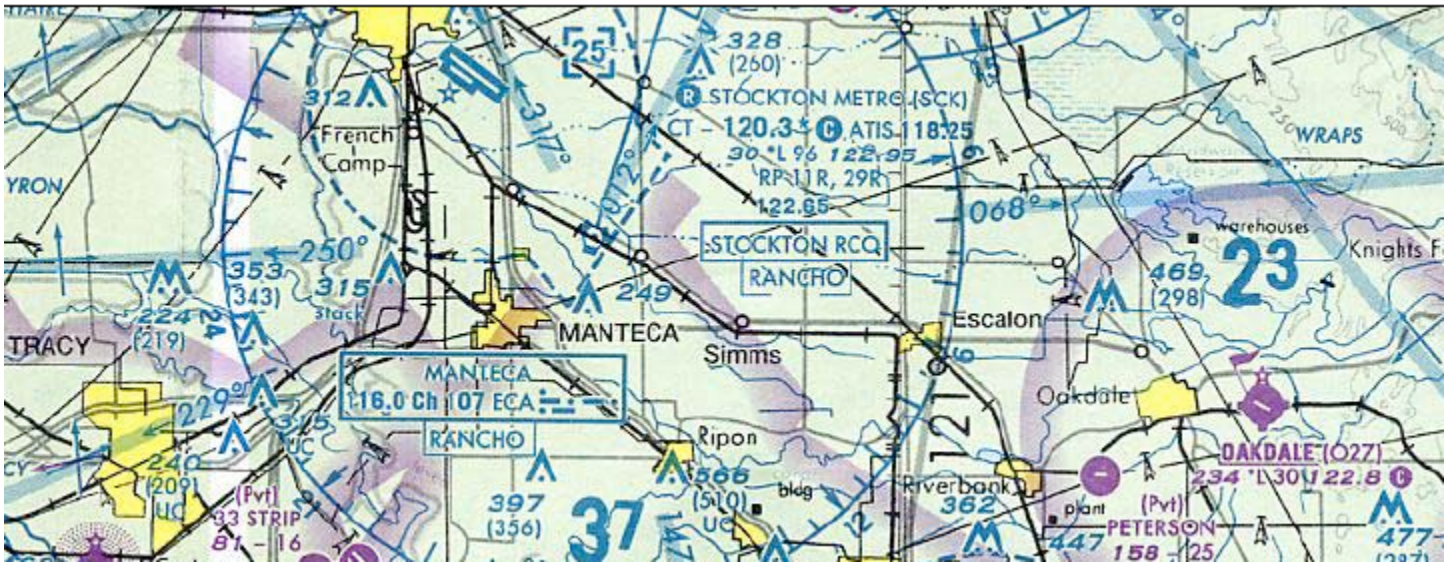
```
> goodrules <- ruleset[quality(ruleset)$lift > 3.5]
> inspect(goodrules)
  lhs                                rhs                    support confidence     lift
1 {herbs}                         => {root vegetables}  0.007015760  0.4312500 3.956477
2 {onions,
   other vegetables}              => {root vegetables}  0.005693950  0.4000000 3.669776
3 {beef,
   other vegetables}              => {root vegetables}  0.007930859  0.4020619 3.688692
4 {tropical fruit,
   curd}                          => {yogurt}           0.005287239  0.5148515 3.690645
5 {citrus fruit,
   pip fruit}                     => {tropical fruit}   0.005592272  0.4044118 3.854060
6 {pip fruit,
   other vegetables,
   whole milk}                    => {root vegetables}  0.005490595  0.4060150 3.724961
7 {citrus fruit,
   other vegetables,
   whole milk}                    => {root vegetables}  0.005795628  0.4453125 4.085493
8 {root vegetables,
   whole milk,
   yogurt}                        => {tropical fruit}   0.005693950  0.3916084 3.732043
9 {tropical fruit,
   other vegetables,
   whole milk}                    => {root vegetables}  0.007015760  0.4107143 3.768074
```

Data Science

- **Data Mining Process 4: Interpreting Results**
  - Shoppers purchasing in particular combinations that might be recipe oriented
    - Soup
    - Fruit platter with dip
  - Actionable recommendations to management might include:
    - Publish recipes along with coupons.
    - Publish visuals of the finished product along with the appropriate marketing verbiage.
    - Place recipes and visuals coincident with item/product locations.

SVM Overview

Intro to SVM



In this lecture, we will examine a set of supervised learning approaches known as support vector machines (SVMs). SVMs are flexible algorithms that excel at addressing classification problems.

Supervised Learning

- "Supervised learning" Data Mining
  - Train algorithm on an initial set of data.
  - Test algorithm on a new set of data.
  - Validate "trained" algorithm predicted the right outcome.

Supervised Learning Example

- Train an algorithm to predict the weather
- Collect weather data over a period of time
  - Sunny, cloudy
  - Temperature
  - Barometer
  - Wind speed and direction
- Train an algorithm with these variables
- Collect more weather data and then:
  predict the weather via our trained algorithm,
  and validate the prediction

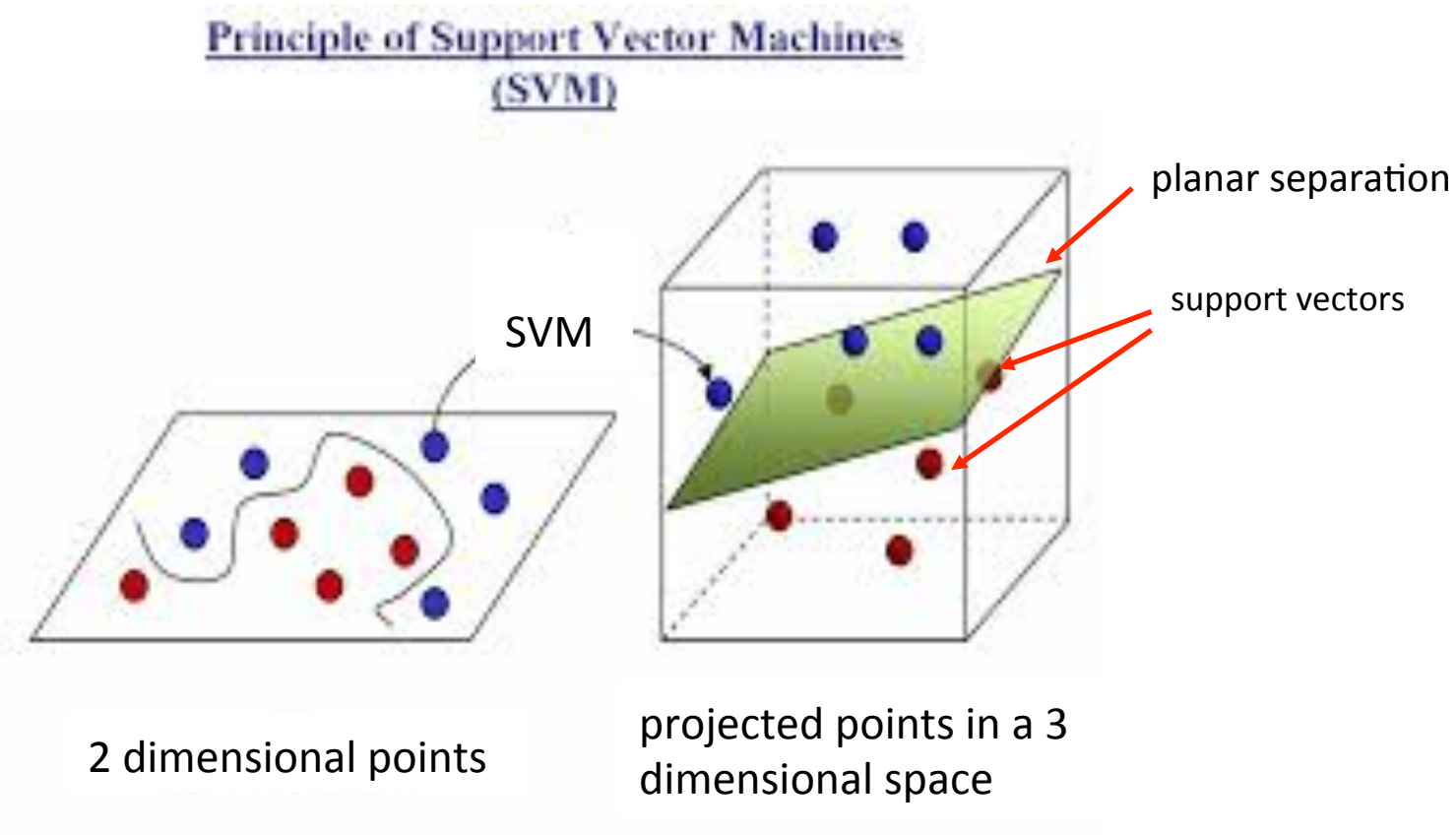
Supervised Learning Strategy

- **Substantial number of training cases** that the algorithm can use to discover and mimic the underlying pattern
- Use the results of this process on a test data set to **determine how well algorithm performed**, i.e., "cross validate"
- **Cross validate**: the process of verifying that the trained algorithm can carry out its prediction or classification task accurately on novel data

Chapter Use Case

- Kernel: mapping algorithm
  - Input data
    (independent variables from a given case)
  - Kernel:
    Formula run on each case's input data
  - Output data:
    Position of that case in a multidimensional space

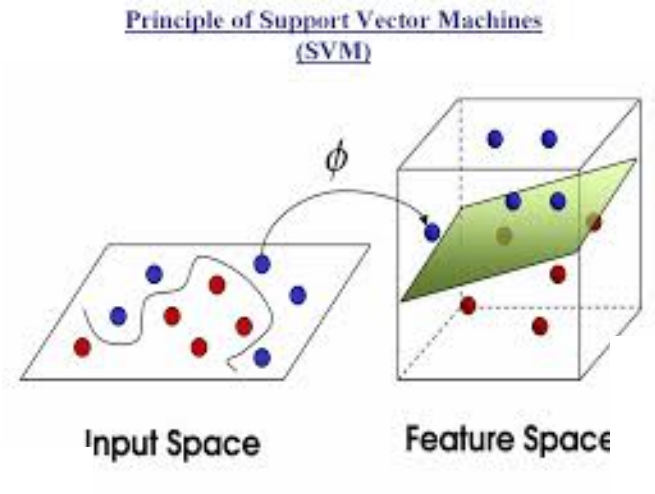
SVM Illustration

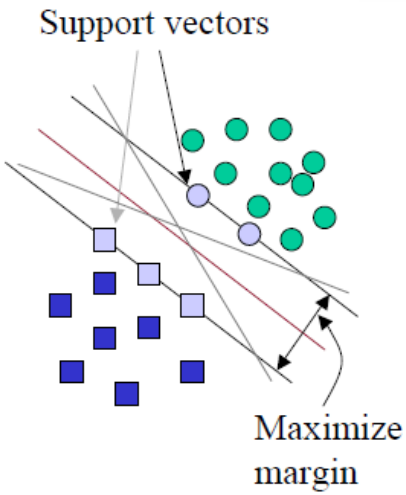


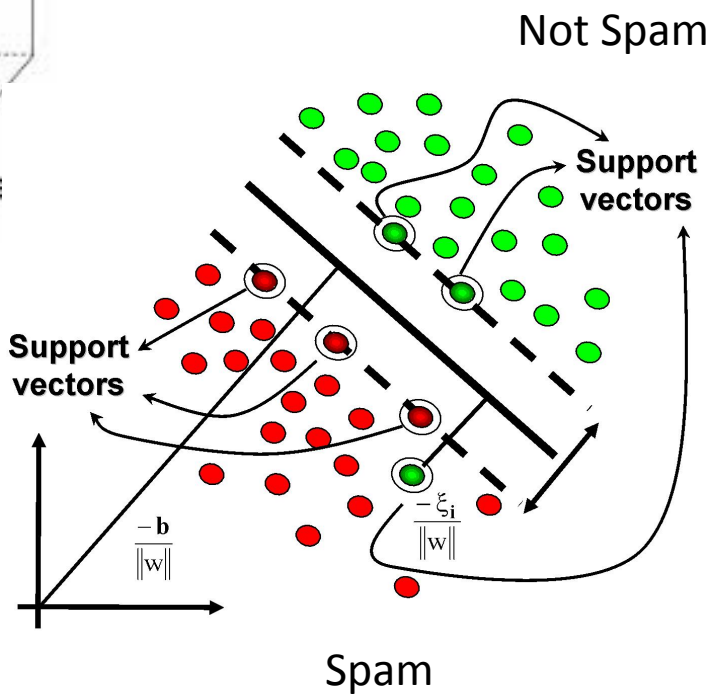
Another Metaphor

- Snow-capped mountain
- 2D/3D perspective
- Linear (planar) separation: snow cap/tree line
- Support vector machine analysis
  - Mathematical description of the position and orientation of the planar separation
- Novel data point to be mapped via a SVM into higher-dimensional space and indicate whether the point was above or below the planar separation

SVM Illustration (continued)







Question

What are some examples where something like SVM could be used?

SVM Example

SVM Setup

- R kernalab package
- Spam data set
  http://archive.ics.uci.edu/ml/datasets/Spambase

An Example – Spam Email

- Develop a support vector machine (SVM) to classify e-mails as **spam or not spam**

- SVM:
  Map a low-dimensional problem into a higher-dimensional space

- Goal:
  Being able to describe geometric boundaries between different regions

Looking at Spam in R

```
install.packages("kernlab")
library(kernlab)

# load R supplied data set
data(spam)

# structure, Inspect contents
str(spam)

# dimension, overview
dim(spam)

# delineates spam, nonspam counts
table(spam$type)
```

Looking at the Structure of Spam Data

```
> str(spam)
'data.frame': 4601 obs. of  58 variables:
 $ make             : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06 ...
 $ address          : num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
 $ all              : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.77 ...
 $ num3d            : num  0 0 0 0 0 0 0 0 0 0 ...
 $ our              : num  0.32 0.14 1.23 0.63 0.63 1.85 1.92 1.88 0.61 0.19 ...
 $ over             : num  0 0.28 0.19 0 0 0 0 0 0 0.32 ...
 $ remove           : num  0 0.21 0.19 0.31 0.31 0 0 0 0.3 0.38 ...
 $ internet         : num  0 0.07 0.12 0.63 0.63 1.85 0 1.88 0 0 ...


 $ capitalAve       : num  3.76 5.11 9.82 3.54 3.54 ...
 $ capitalLong      : num  61 101 485 40 40 15 4 11 445 43 ...
 $ capitalTotal     : num  278 1028 2259 191 191 ...
 $ type             : Factor w/ 2 levels "nonspam","spam": 2 2 2 2 2 2 2 2 2 2
```

```
> dim(spam)
[1] 4601   58
> table(spam$type)

nonspam    spam
   2788    1813
>
```

SVM Setup

Divide into a “training” and “test” data set:

- Suggested delineation
  - 2/3 -> Training
  - 1/3 -> Test
- Create a data set of 4601 random indexes utilizing sample function
- Create 2/3 cut point
- Create **trainData** (using 2/3 cut point) and **testData** (using the rest of the data) and the previously created random indexes

R Code for SVM Setup

```
> randIndex <- sample(1:dim(spam)[1])      # Create list/vector variable-random index
> summary(randIndex)                       # verify indicies in randIndex
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
      1    1151    2301    2301    3451    4601
> length(randIndex)                        # verify indicies in randIndex
[1] 4601
> head(randIndex)                          # look at first few cases
[1] 2272 2974 4153 3142 2280 2491
> cutPoint2_3 <- floor(2 * dim(spam)[1]/3) # create 2/3 cutpoint
> cutPoint2_3                              # verify 2/3 cutpoint
[1] 3067
> trainData <-spam[randIndex[1:cutPoint2_3],]   #create training data set
> testData <-spam[randIndex[(cutPoint2_3+1):dim(spam)[1]],]  # create test data set
```

trainData & TestData

trainData 3067 obs. of 58 variables

	row.names	make	address	all	num3d	our	over	remove	internet	order	mail	receive	will	people	report
1	2272	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	2974	0.10	0.00	0.21	0.00	0.31	0.00	0.00	0.10	0.00	0.00	0.00	0.63	0.21	0.00
3	4153	0.00	0.78	1.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.78	0.00	0.00
4	3142	0.00	14.28	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	2280	0.00	0.00	1.58	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.79	0.00	0.00
6	2491	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	587	0.00	0.00	0.36	0.00	0.36	0.00	0.00	0.00	0.36	0.36	0.00	0.36	0.00	1.09
8	2751	0.00	0.00	1.16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.32	0.00	0.00
9	1967	0.00	0.00	0.00	0.00	2.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

testData 1534 obs. of 58 variables

	row.names	make	address	all	num3d	our	over	remove	internet	order	mail	receive	will	people	report
1	2272	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
2	2974	0.10	0.00	0.21	0.00	0.31	0.00	0.00	0.10	0.00	0.00	0.00	0.63	0.21	0.00
3	4153	0.00	0.78	1.56	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.78	0.00	0.00
4	3142	0.00	14.28	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	2280	0.00	0.00	1.58	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.79	0.00	0.00
6	2491	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	587	0.00	0.00	0.36	0.00	0.36	0.00	0.00	0.00	0.36	0.36	0.00	0.36	0.00	1.09
8	2751	0.00	0.00	1.16	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	2.32	0.00	0.00
9	1967	0.00	0.00	0.00	0.00	2.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Creating the SVM Model

- Train support vector model
  - svmOutput <- ksvm(type ~ ., data=trainData,kernel="rbfdot",kpar="automatic", C=5,cross=3,prob.model=TRUE)
- type ~ specifies the model we want to test, i.e., want to have the **type** variable (spam, nonspam) as the output variable to predict.
- "." uses all other variables in the data frame to predict **type**.
- **Data** specifies the data frame to use in the analysis, i.e., trainData.

Creating the SVM Model (continued)

- Train support vector model
  - svmOutput <- ksvm(type ~ ., data=trainData,kernel="rbfdot",kpar= "automatic",C=5,cross=3,prob.model=TRUE)
- rbfdot—kernel function that projects the low-dimensional problem into higher-dimensional space, i.e., getting the maximum separation of distance between spam and nonspam cases
- **kpar** refers to a variety of parameters that can be used to control the radial bias function kernel (rbfdot)

Creating the SVM Model (continued)

- Train support vector model
  - svmOutput <- ksvm(type ~ ., data=trainData,kernel="rbfdot",kpar="automatic",C=5,cross=3, prob.model=TRUE)
- **C** argument refers to "cost of constraints"
  - High C value impact (See Next Slide)
  - Low C value impact (See Next Slide)
- **cross** refers to the cross-validation model that the algorithm uses -- "overfitting"

Cost Parameter ('C')

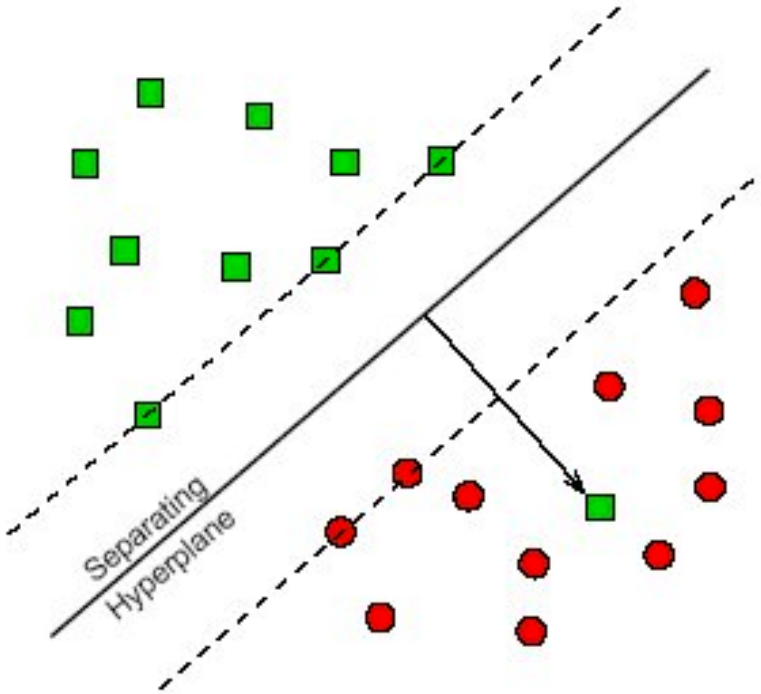
ksvm Cost Parameter Impact Summary

Higher Cost 'C' Value	**Fewer Classification Mistakes** **Fewer Problem Points**	**Smaller Margin of Separation**	**Specialized Model**	**Higher Cross-Validation Error**	**Lower Training Error**
Lower Cost 'C' Value	**More Classification Mistakes** **More Problem Points**	**Higher Margin of Separation**	**Generalized Model**	**Lower Cross-Validation Error**	**Higher Training Error**

Why Error Might Not Be Bad

Non-separable training sets

Use linear separation, but admit training errors.



Penalty of error: distance to hyperplane multiplied by *error cost* C.

Looking at the SVM Output

```
> svmOutput
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 5

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.0292958021260521

Number of Support Vectors : 948

Objective Function Value : -1739.444
Training error : 0.031953
Cross validation error : 0.074994
Probability model included.
>
```

Changing the "Cost" Parameter

```
> svmOutput <- ksvm(type ~ .,
data=trainData,kernel="rbfdot",kpar="automatic",C=50,cross=3,prob.model=TRUE)
Using automatic sigma estimation (sigest) for RBF or laplace kernel
> svmOutput
Support Vector Machine object of class "ksvm"

SV type: C-svc  (classification)
 parameter : cost C = 50

Gaussian Radial Basis kernel function.
 Hyperparameter : sigma =  0.0290816646010172

Number of Support Vectors : 847

Objective Function Value : -7525.442
Training error : 0.011412
Cross validation error : 0.091296
Probability model included.
```

C=5

```
Objective Function Value : -1739.444
Training error : 0.031953
Cross validation error : 0.074994
Probability model included.
>
```

Predicting "testData" Results

- Predict "testData" where svmOutput had C=5

```
> svmPred <- predict(svmOutput, testData,type="votes")
> compTable <- data.frame(testData[,58],svmPred[1,])
> table(compTable)
               svmPred.1...
testData...58.    0   1
       nonspam   33 884
       spam     551  66
```

Interpretation

0 – spam 1 – not spam

33 cases not spam but classified as spam
884 cases not spam and classified as not spam
551 cases spam and classified as spam
66 cases classified as spam but not spam

33 + 66 = 99 error cases
99/1534 = .05867 total error rate %
Cross Validation Error : .0749

Accuracy rate: 94.2%


SYRACUSE UNIVERSITY
SUOS
CULTORES
SCIENTIA
CORONAT