Supervised Data Mining

Professor Jeff Saltz
Professor Jeff Stanton

ischool.syr.edu

School of Information Studies
SYRACUSE UNIVERSITY

Reminder: Data Science Day

In person and virtually (Today & Tomorrow)

Registration :
https://graduate.admissions.go.syr.edu/register/?id=fe51b3db-7d2f-4df3-a8c0-16c29ea08964

Thursday talks (starting at ~4pm):
- Exploring What is Data Science & Analytics
- What is Visual Analytics?
- Metadata & Analytics
- Learning Analytics
- Graduate Student Panel

Friday talks (starting at ~10am) - Room 347, Hinds Hall
- The Use of Data Science to Better Address COVID-19 - Thibaut Jombart (WHO / UK researcher)
- Applied Deep Learning, Examples in Industry
- Ethics in Data Science: Problems, Approaches, and Future

Break for lunch (then re-start at 1pm)
- Managing Risk in Data Science
- Data Science in the Real World – a discussion of what our adjunct faculty do outside the iSchool
- Data Science Job Market Update

Schedule: https://calendar.syracuse.edu/events/2021-nov-04/data-science-day/

Summary of Previous Learning

What you should know and be able to do at this point :

1. List major skills needed by data scientists and describe the development of a DS project with domain analysis, SMEs, data and modeling
2. Explain and use a data frame in R and various diagnostics for variables and data structures; describe and use multiple strategies for accessing external data from R; use SQL facilities from within R; automate with functions
3. Define and calculate the most common descriptive statistics; describe the effects of randomness on sampling; create and interpret a sampling distribution including defining the law of large numbers and the central limit theorem
4. Use plot and ggplot to visualize data and create maps
5. Create and interpret a multiple regression model using lm()

Objectives

- Create appropriate "training" and "test" dataset environments
- Use the data mining techniques
  - Support vector machines (SVM)
  - Classification and Regression Trees (CART)
- Develop SVM & Cart R code

Using Classification and Regression Trees

ischool.syr.edu

School of Information Studies
SYRACUSE UNIVERSITY

Reviewing The Modeling Process

- Use a substantial number of training cases
  - The machine learning algorithm can use that data to build a model
- Use the results of this process (i.e., the model) on test data set to determine how well algorithm performed
  - Validate the model on new data
- The result is a model that can be used for prediction
  - Predict data that was not used during training
  - Predict future instances of data

*Note: The model is **not always useful** for explaining results to managers.*
In some cases, algorithms produce results that are not easy to interpret or visualize; for some algorithms there is no output that is like a regression coefficient.

An Example: CART
Classification and Regression Trees

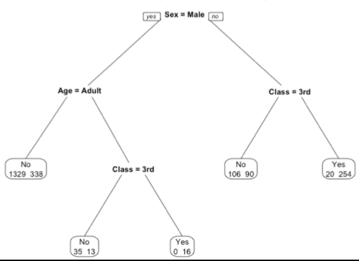
- A family of data mining techniques for developing predictions on **either** continuous outcome variables or class outcome variables
- Can be used either for classification with unordered categories, or pseudo-continuous ordered outcomes
- Uses iterative model building techniques, typically:
  - Develops “splits” in the data where the level of a predictor is used to divide the dataset into two (or more) partitions according to the status of the outcome variable
  - The resulting models can be represented as binary trees

Example of RPART – for titanic

```
library(rpart)
library(rpart.plot)

url <- "https://intro-datascience.s3.us-east-2.amazonaws.com/titanic.raw.Rdata"

download.file(url, "t.raw.Rdata")
load("t.raw.Rdata")

titanic <- titanic.raw

#Explore the dataset
str(titanic)
'data.frame':	2201 obs. of  4 variables:
 $ Class   : Factor w/ 4 levels "1st","2nd","3rd",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ Sex     : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 2 ...
 $ Age     : Factor w/ 2 levels "Adult","Child": 2 2 2 2 2 2 2 2 2 2 ...
 $ Survived: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1
```

Example of RPART – for titanic

```
#Build the model
cartTree <- rpart(Survived ~ ., data = titanic)
prp(cartTree, faclen = 0, cex = 0.8, extra = 1)
```



For left pointing leaves:
- The first number: # of correctly classified cases
- The second number: # incorrectly classified.

For right pointing leaves, vice versa.

Example of RPART – for titanic

```
cartTree
n= 2201

node), split, n, loss, yval, (yprob)
      * denotes terminal node

 1) root 2201 711 No (0.6769650 0.3230350)
   2) Sex=Male 1731 367 No (0.7879838 0.2120162)
     4) Age=Adult 1667 338 No (0.7972406 0.2027594) *
     5) Age=Child 64  29 No (0.5468750 0.4531250)
      10) Class=3rd 48  13 No (0.7291667 0.2708333) *
      11) Class=1st,2nd 16   0 Yes (0.0000000 1.0000000) *
   3) Sex=Female 470 126 Yes (0.2680851 0.7319149)
     6) Class=3rd 196  90 No (0.5408163 0.4591837) *
     7) Class=1st,2nd,Crew 274  20 Yes (0.0729927 0.9270073) *
```

Predicting values using the Model

```
predictValues <- predict(cartTree, newdata=titanic,
                         type = "class")

titanic[1,]
  Class  Sex   Age   Survived
3  3rd   Male  Child    No

predictValues[1]
1
No
Levels: No Yes
```

What was the Accuracy

```
actualSurvived <- as.factor(titanic$Survived == "Yes")
confMatrix <- table(predictedSurvived,actualSurvived)
confMatrix

                 actualSurvived
predictedSurvived FALSE TRUE
            FALSE  1470  441
            TRUE     20  270

accuracy <- 1 - (sum(confMatrix)  -  sum(diag(confMatrix)))
                                     / sum(confMatrix)
accuracy
0.7905498
```

Questions

Is 79% good?

What is overfitting?

How might that be relevant?

Using Caret to create Training and Test sets

```
library(caret)

# makes the sampling predictable
set.seed(110)

# Randomly sample elements to go into a training data set
trainList <- createDataPartition(y=titanic$Survived,p=.80,list=FALSE)

# Include all of those elements in the training set
trainSet <- titanic[trainList,]

# Construct test set from everything that didn't go into the training
testSet <- titanic[-trainList,]
```

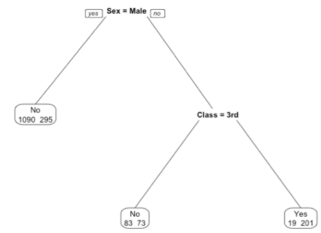
Check trainSet

```
head(trainList)
     Resample1
[1,]         1
[2,]         2
[3,]         4
[4,]         5
[5,]         6
[6,]         7
```

Note that some observations are skipped. . .

Use the trainSet

```
cartTree <- rpart(Survived ~ ., data = trainSet)
prp(cartTree, faclen = 0, cex = 0.8, extra = 1)
```



Evaluate using the testSet

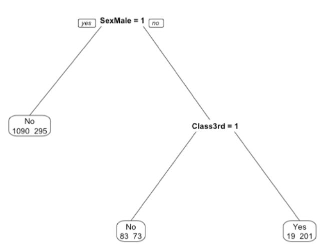
```
#Note use of "class"
predictValues <- predict(cartTree,newdata=testSet,
                         type = "class")

#simpler way to do confusion matrix
confusion <- confusionMatrix(predictValues,
                         testSet$Survived)

confusion$overall[1]
```

Accuracy
0.7954545

Using all the data:
the accuracy was 0.7905498

Using Caret to build a model

```
library(caret)
cartTree1 <- train(Survived ~ ., data=trainSet,  method="rpart")
prp(cartTree1$finalModel, faclen = 0, cex = 0.8, extra = 1)
```



Exploring the model build via Caret

```
cartTree1
CART

1761 samples
  3 predictor
  2 classes: 'No', 'Yes'

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 1761, 1761, 1761, 1761, 1761, 1761, ...
Resampling results across tuning parameters:

  cp          Accuracy   Kappa
  0.009666081  0.7840318  0.4267801
  0.017574692  0.7802065  0.4222871
  0.302284710  0.7206203  0.1868332

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was cp = 0.009666081.
```

```
cartTree1$finalModel
n= 1761

1) root 1761 569 No (0.67688813 0.32311187)
  2) SexMale>=0.5 1385 295 No (0.78700361 0.21299639) *
  3) SexMale< 0.5 376 102 Yes (0.27127660 0.72872340)
    6) Class3rd>=0.5 156  73 No (0.53205128 0.46794872) *
    7) Class3rd< 0.5 220  19 Yes (0.08636364 0.91363636) *
```

complexity parameter (cp): imposes a penalty to the tree for having too many splits

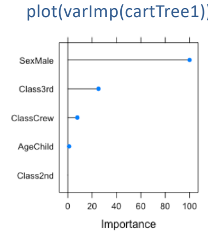
Exploring the model build via Caret

```
predictValues <- predict(cartTree1,newdata=testSet, type = "raw")

confusion <- confusionMatrix(predictValues, testSet$Survived)
confusion
Confusion Matrix and Statistics

          Reference
Prediction FALSE TRUE
     FALSE   297   89
     TRUE      1   53

               Accuracy : 0.7955
                 95% CI : (0.7547, 0.8322)
    No Information Rate : 0.6773
    P-Value [Acc > NIR] : 2.382e-08
```

Exploring Variable Importance

```
varImp(cartTree)
        Overall
Age    8.178044
Class  98.019060
Sex   157.306989

varImp(cartTree1)
rpart variable importance

          Overall
SexMale   100.000
Class3rd   25.233
ClassCrew   7.825
AgeChild    1.136
Class2nd    0.000
```

plot(varImp(cartTree1))



Advanced Example: Using Classification and Regression Trees on Credit Decisioning

The German Credit Database

- From the UCI Machine Learning Repository
- Built-in dataset stored in the caret package
- 1000 cases and 62 variables
- "Class" is the outcome variable:
  binary creditworthiness ("good" or "bad")

Duration	Age	NumberExistingCredits	NumberPeopleMaintenance	Telephone	ForeignWorker	Class
4	67	2	1	0	1	Good
2	22	1	1	1	1	Bad
3	49	1	2	1	1	Good
4	45	1	2	1	1	Good
4	53	2	2	1	1	Bad
4	35	1	2	0	1	Good
4	53	1	1	1	1	Good
2	35	1	1	0	1	Good

Showing 1 to 9 of 1,000 entries

The German Credit Database

- Example predictors:
  - Checking account status, duration, credit history, purpose of the loan, amount of the loan, savings accounts or bonds, installment rate in percentage of disposable income, other installment plans, number of existing credits
  - Employment duration, personal information, other debtors/guarantors, residence duration, property, age, housing, job information, number of people being liable to provide maintenance for, telephone, and foreign worker status
- For convenience, we will only use the first 10 columns, including Class
- We will try the "treebag" algorithm, a bagged CART; there are more than 200 fitting algorithms
- See http://topepo.github.io/caret/modelList.html

Create Training and Test sets

```
# Just grab a subset of the data for the demo
data("GermanCredit")
subCredit <- GermanCredit[,1:10]

# makes the sampling predictable
set.seed(111)

# Randomly sample elements to go into a training data set
trainList <-
    createDataPartition(y=subCredit$Class,p=.80,list=FALSE)

# create test and train datasets
trainSet <- subCredit[trainList,]
testSet <- subCredit[-trainList,]
```

Train Using a treebag model

```
fit1 <- train(Class ~ ., data=trainSet, method="treebag",
                                        preProc=c("center","scale"))
```

- The **train()** command trains the specified model (here it is "**treebag**" – look it up)
- **Class ~ .** This is the standard model specification syntax; the dot after the tilde includes all of the other variables as predictors; otherwise spell them out separated with plus signs
- **preProc=** allows pre-processing of the input data, in this cases taking the precaution of centering and scaling to put every input variable on the same scale

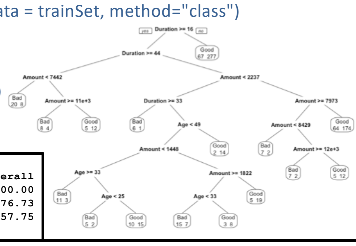
Interpret Results: Variable Importance

```
varImp(fit1)
treebag variable importance

                          Overall
Amount                     100.00
Age                         76.73
Duration                    57.75
ResidenceDuration           35.73
InstallmentRatePercentage   34.33
NumberExistingCredits       20.84
Telephone                   13.91
NumberPeopleMaintenance     11.77
ForeignWorker                0.00
```

Interpret Results: Use rpart

- Use rpart to display a simple CART tree with key variables.
- This tree gives a general idea of how variables are working, but is different from the treebag model (which has 25 trees!)

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

prp(cartTree, faclen = 0,
    cex = 0.8, extra = 1)
```




```
treebag variable importance
                 Overall
Amount            100.00
Age                76.73
Duration           57.75
. . .
```

Step 4: Assess Fit with New (test) Data

```
predOut <- predict(fit1, newdata=testSet)
confusion <- confusionMatrix(predOut, testSet$Class)
confusion
Confusion Matrix and Statistics

          Reference
Prediction Bad Good
      Bad   20   18
      Good  40  122

         Accuracy : 0.71
```

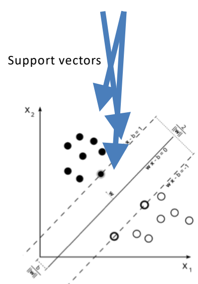
Is this a good model?

```
confusion
Confusion Matrix and Statistics

          Reference
Prediction Bad Good
      Bad   20   18
      Good  40  122

               Accuracy : 0.71
                 95% CI : (0.6418, 0.7718)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.412322
```

Using Support Vector Machines

School of Information Studies
SYRACUSE UNIVERSITY
ischool.syr.edu

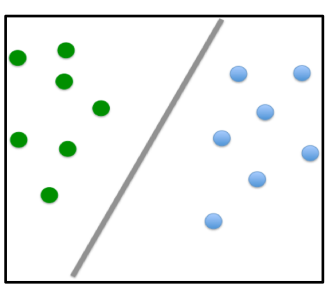
SVM: What are support vectors?



- Very **different approach to prediction** compared with the linear methods and tree methods
- Based on **statistical learning theory**
- Use the **cases that are difficult to predict** to help define a robust prediction model (because any model can handle "easy" cases!)
- Support vectors ***are* the harder-to-predict cases** on a defined border + margin that separate the cases
- The **profiles of predictor scores** define the **support** or border + margin that serves to classify future cases

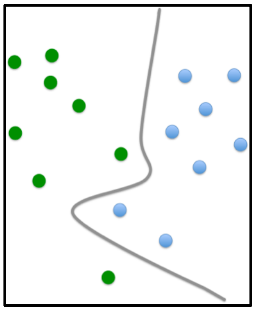
Visualizing the Algorithm

This is Easy



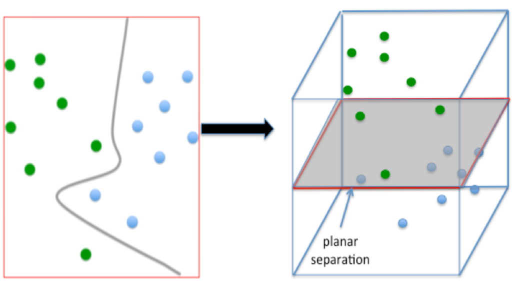
Visualizing the Algorithm

What About This?



Visualizing the Algorithm

2D to 3D Mapping



Use SVM to Explore the Credit Data

```
#Step 1: load the required packages was already done

#Step 2: Create Training and Test sets
data("GermanCredit")
subCredit <- GermanCredit[,1:10]

# makes the sampling predictable
set.seed(111)

# Randomly sample elements to go into a training data set
trainList <- createDataPartition(y=subCredit$Class,p=.80,list=FALSE)
trainSet <- subCredit[trainList,]
testSet <- subCredit[-trainList,]
```

Use SVM to Explore the Credit Data

```
#Train the SVM model
fit2 <- train(Class ~ ., data=trainSet, method="svmRadial",
                                   preProc=c("center","scale"))

#Assess Fit with new data
predOut <- predict(fit2, newdata=testSet)
```

Compare the Results

TreeBag

```
confMatrix <-
       table(predOut,testSet$Class)
confMatrix

predOut Bad Good
   Bad   20   18
   Good  40  122

prop.table(confMatrix)

predOut  Bad Good
   Bad  0.10 0.09
   Good 0.20 0.61

errorRate <- (sum(confMatrix) -
          sum(diag(confMatrix)))
                 /sum(confMatrix)
errorRate
[1] 0.29
```

SVM

```
confMatrix <-
       table(predOut,testSet$Class)
confMatrix

predOut Bad Good
   Bad   10    3
   Good  50  137

prop.table(confMatrix)

predOut  Bad Good
   Bad  0.05 0.015
   Good 0.25 0.685

errorRate <- (sum(confMatrix) -
          sum(diag(confMatrix)))
                 /sum(confMatrix)
errorRate
[1] 0.265
```

39

Compare the Results (varImp)

TreeBag

	Importance
Amount	100.00
Age	76.73
Duration	57.75
ResidenceDuration	35.73
InstallmentRatePerc	34.33
NumberExistingCredits	20.84
Telephone	13.91
NumberPeopleMaint.	11.77
ForeignWorker	0.00

SVM

	Importance
Duration	100.00
Age	51.92
Amount	43.20
InstallmentRatePerc	31.54
NumberExistingCredits	25.19
ForeignWorker	9.85
NumberPeopleMaint	7.01
ResidenceDuration	0.27
Telephone	0.00

Explore the confusion matrix

```
# Review the error – use the built in 'confusionMatrix'
confusion <- confusionMatrix(predOut, testSet$Class)
confusion
Confusion Matrix and Statistics

          Reference
Prediction Bad Good
      Bad   10    3
      Good  50  137

               Accuracy : 0.735
                 95% CI : (0.6681, 0.7948)
    No Information Rate : 0.7
    P-Value [Acc > NIR] : 0.1579
```

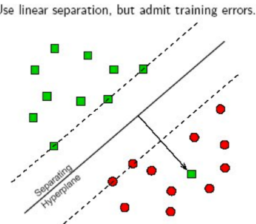
Explore the "C" parameter

```
fit2
Support Vector Machines with Radial Basis Function Kernel

800 samples
  9 predictor
  2 classes: 'Bad', 'Good'

Pre-processing: centered (9), scaled (9)
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 800, 800, 800, 800, 800, 800, ...
Resampling results across tuning parameters:

  C     Accuracy
  0.25  0.6969592
  0.50  0.6974949
  1.00  0.6989207
```

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.

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**

Question

If SVM is such a ”black box” – where it is hard to know how the model actually works and what variables mater the most – why would we ever bother to use it?