Exam 1 Review

Peter Kabai

Review Topics

- Terminology
- Probability
- Common Issues
- More R Coding

Different Kinds of Models

- Classification:
- The target feature is categorical.
- Regression:
- The target feature is numeric.
- Clustering:
- Groups datapoints into clusters.
- Supervised:
- Training data is labeled.
- Un-Supervised:
- The training data is not labled.

More Terminology

- Underfitting:
- When the model is not flexible enough to fit the data well.
- Overfitting:
- When the model fits the training data too well to fit the test data well.
- Model Flexibility:
- How well the model fits the training data, a model can be too flexible or too rigid.
- Which one leads to overfitting?

Even More Terminology

- Vectorized Operation:
- When an operation is performed on every element in a vector.
- What are some examples?
- Aggregate functions:
- When a function takes a vector, and returns one value.
- Is that value a vector?
- Yes, there are no such things as scalers in R.

Probability Tables

 # What does this table signify?

	Parent Vaccinated	Parent Not Vaccinated	
Child Vaccinated	550	210	
Child Not Vaccinated	120	120	

 # Next, lets put in the totals for children.

Probability Tables

	Parent Vaccinated	Parent Not Vaccinated	
Child Vaccinated	550	210	760
Child Not Vaccinated	120	120	240

 # Next, lets put in the totals for parents

Probability Tables

	Parent Vaccinated	Parent Not Vaccinated	
Child Vaccinated	550	210	760
Child Not Vaccinated	120	120	240
	670	330	

```
# Lets put in the total number of datapoints
```

Probability Tables

	Parent Vaccinated	Parent Not Vaccinated	
Child Vaccinated	550	210	760
Child Not Vaccinated	120	120	240
	670	330	1000

 # Next, we can convert to probabilities.

 # What will they be?

Probability Tables

	Parent Vaccinated	Parent Not Vaccinated	
Child Vaccinated	0.55	0.21	
Child Not Vaccinated	0.12	0.12	

 # What about the probabilities for each feature?

Probability Tables

	Parent Vaccinated	Parent Not Vaccinated	
Child Vaccinated	0.55	0.21	0.76
Child Not Vaccinated	0.12	0.12	0.24
	0.62	0.33	1

 # Probability tables are very useful for looking at conditional probability.

Coin Flip Problem

```
# How many possible possible outcomes if you flip 1 coin?
2**1
```

```
## [1] 2
```

```
# H-T, T-H
```

```
# How many possible possible outcomes if you flip 2 coins?
2**2
```

```
## [1] 4
```

```
# H-H, T-T, T-H, H-T
```

Coin Flip Problem

```
# Three coins?
2**3
```

```
## [1] 8
```

```
# H-H-H, H-H-T, H-T-H, H-T-T
# T-H-H, T-H-T, T-T-H, T-T-T

# X coins?
x = 100
2**x
```

```
## [1] 1.267651e+30
```

```
# How would that change if we were flipping 100 die?
```

Coin Flip Problem

```
# If we flip 3 coins what is the probability of exactly two
# heads, given the last flip is heads?
2**3
```

```
## [1] 8
```

```
# What does this code do?
dat = as.data.frame(replicate(1000, replicate(3, sample(0:1, 1))))
space = dat[,dat[3,] == 1]
mean(apply(space, 2, FUN = function(x) { sum(x) == 2 }))
```

```
## [1] 0.4880734
```

Expectation Problem - The Setup

```
# What is the expected value of the sum of a 2 sided and a 4 sided die?

# First, what are the possible sums?
# 1+1, 2+1, 3+1, 4+1, 1+2, 2+2, 3+2, 4+2
# 2,   3,   4,   5,   3,   4,   5,   6
outcome = 2:6
outcome
```

```
## [1] 2 3 4 5 6
```

```
# What is the probability for each?
# 2 - 1/8
# 3 - 2/8
# 4 - 2/8
# 5 - 2/8
# 6 - 1/8
probs = c(1/8,1/4,1/4,1/4,1/8)
```

Expectation Problem - Calculating

```
# How would we calculate the expectation?
exp = sum(outcome*probs)
exp
```

```
## [1] 4
```

```
# Does that seem realistic?
# What's the variance?
var = sum(probs*((outcome-exp)^2))
var
```

```
## [1] 1.5
```

```
# What's the standard deviation?
sqrt(var)
```

```
## [1] 1.224745
```

Expectation Problem - Possible Confusion

```
# Using mean vs using weighted mean and probabilities
# when calculating the variance

out = c(1, 4, 5, 5, 4, 4, 4)
mean((out-mean(out))^2)
```

```
## [1] 1.55102
```

```
uniq = c(1, 4, 5)
probs = c(1/7, 4/7, 2/7)
exp = sum(uniq*probs)
sum(((uniq-exp)^2)*probs)
```

```
## [1] 1.55102
```

```
# These are the same!
```

Common Issues - Creating Vectors

```
# These two expressions create same thing...
x = 1:10
y = c(1:10)

identical(x, y)
```

```
## [1] TRUE
```

```
# ...but the first way of doing it is much better!

# Why did I check them using identical() rather than == ?
y == x
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
```

Common Issues - Using replicate()

```
# What do these lines do?
x = mean(sample(1:10, 5))
replicate(10, x)
```

```
##  [1] 5.6 5.6 5.6 5.6 5.6 5.6 5.6 5.6 5.6 5.6
```

```
# How is this different?
replicate(10, mean(sample(1:10, 5)))
```

```
##  [1] 5.4 3.0 6.0 5.0 4.4 5.4 7.6 6.4 5.6 6.4
```

Common Issues - Using replicate()

```
# Here's a function
test = function(x) {
  return(sample(x, 1))
}

# Can functions be used in replicate?
replicate(10, test(1:10))
```

```
##  [1] 10 10  2  8  1  3 10  2  4  1
```

Common Issues - Using ifelse()

```r
# What is the result of this expression?
x = 1:10
ifelse(x <= 5, 0, 1)
```

```
##  [1] 0 0 0 0 0 1 1 1 1 1
```

```r
# Writing the same with a boolean vector
bool_vel = c(TRUE, TRUE, TRUE, TRUE, TRUE, FALSE, FALSE, FALSE, FALSE, FALSE)
ifelse(bool_vel, 0, 1)
```

```
##  [1] 0 0 0 0 0 1 1 1 1 1
```

Common Issues - Using runif()

```
# What's the value of r?
r = runif(1000)
head(r)
```

```
## [1] 0.73022919 0.36763590 0.06289644 0.32930885 0.43056628 0.41330400
```

```
# runif(n) generates n random numbers between 0 and 1

# What is the value of this expression?
mean(r)
```

```
## [1] 0.5043439
```

Common Issues - Combining Functions

```
# What's the value of r?
r = runif(200)+0.9
head(r)
```

```
## [1] 1.8150784 0.9001173 1.0141490 1.1119170 1.6257523 1.4905443
```

```
# What is the result of this expression?
head(floor(runif(200)+0.9), 40)
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1
```

```
# Now, let's replicate it!
prob = replicate(100, mean(floor(runif(200)+0.9)))
head(prob, 10)
```

```
##  [1] 0.890 0.920 0.900 0.880 0.905 0.915 0.880 0.905 0.900 0.895
```

Common Issues - Working with Strings

```
# Does this concatenate?
c("Two", "Strings")
```

```
## [1] "Two"     "Strings"
```

```
# Nope! Althogh c() can stand for "combine", it just creates a vector
```

```
# So how would I do that?
```

Common Issues - Working with Strings

```
# Does this concatenate?
c("Two", "Strings")
```

```
## [1] "Two"     "Strings"
```

```
# Nope! Althogh c() can stand for "combine", it just creates a vector
```

```
# So how would I do that?
paste("With", "spacing")
```

```
## [1] "With spacing"
```

```
paste0("No", "spacing")
```

```
## [1] "Nospacing"
```

Common Issues - Working with Strings

```
# Finding the length of a string

# How would I find the number of characters?
nchar("The quick brown fox jumps over the lazy dog")
```

```
## [1] 43
```

```
length("The quick brown fox jumps over the lazy dog")
```

```
## [1] 1
```

```
# length() is used to count the number of elements in a vector
# nchar() is used to count the number of characters in a string

# ncol() and nrow() can be used on data frames
```

Common Issues - Working with Strings

```
# Finding the length of a string

# How would I find the number of characters?
nchar("The quick brown fox jumps over the lazy dog")
```

```
## [1] 43
```

```
length("The quick brown fox jumps over the lazy dog")
```

```
## [1] 1
```

```
# length() is used to count the number of elements in a vector
# nchar() is used to count the number of characters in a string

# ncol() and nrow() can be used on data frames
```

Sequences

```
# What is the output of this sequence?
seq(0, 500, 100)
```

```
## [1]   0 100 200 300 400 500
```

Data Frames

```
# Create a data frame with a column of numbers 1 to 10
# and a column of letters a to j
```

Data Frames

```
# Create a data frame with a column of numbers 1 to 10
# and a column of letters a to j

dat = data.frame(1:10, c('a','b','c','d','e','f','g','h','i','j'))
names(dat) = c("Number","Letter")
dat
```

```
##    Number Letter
## 1       1      a
## 2       2      b
## 3       3      c
## 4       4      d
## 5       5      e
## 6       6      f
## 7       7      g
## 8       8      h
## 9       9      i
## 10     10      j
```

Data Frames

```
# Give me a vector of letters, where the matching number is > 5
```

Data Frames

```
# Give me a vector of letters, where the matching number is > 5

as.vector(dat$Letter[dat$Number > 5])

## [1] "f" "g" "h" "i" "j"

as.vector(dat[dat$Number > 5,]$Letter)

## [1] "f" "g" "h" "i" "j"
```

Questions?