ETM538HW5

Jordan Hilton

February 18, 2019

Instance-Based Learning Part A

Question 1

Outlook can be sunny, overcast, or rainy (3); temp can be hot, mild or cool (3); humidity can be high or normal (2), and windy can be false or true (2); so there are $3 * 3 * 2 * 2 = 36$ total possible cases that can be considered.

Question 2

To do this, let me load in our data, assign variable weights, and create a little distance-calculation function. We're also going to generate a list of all possible observations.

```
instancedata<- read_excel("Instance Based Classification v2.xlsx") ## load the spreadsheet
instancedata<- data.frame(instancedata[-c(15:16),-c(6:11)]) ##drop the calculated distance columns and a
instances<-instancedata[-5] ##create a table without the outcome for measuring distances
head(instancedata)
```

```
##    Outlook Temp  Humid Windy Play.
## 1    sunny  hot   high FALSE    no
## 2    sunny  hot   high  TRUE    no
## 3 overcast  hot   high FALSE   yes
## 4    rainy mild   high FALSE   yes
## 5    rainy cool normal FALSE   yes
## 6    rainy cool normal  TRUE    no
```

Here we're just assigning values to the weight for each variable:

```
weights<-c(1,1,1,1) ## make a vector of variable weights
names(weights)<-c("outlook", "temp", "humid", "windy")  ##name them so we don't forget
weights
```

```
## outlook    temp   humid   windy
##       1       1       1       1
```

Here we're creating a function to calculate the Manhattan distance between two observations:

```
distance<-function(x,y){
  matching<-x==y # returns a list of booleans if each element matches
  result<-4-as.numeric(matching) %*% weights #converts our list of matches to numeric instead of boolea
  return(result[1,1]) #the result is a matrix so we just grab the value out of it
}
```

And here we're generating a table of all possible observations and noting that the table has 36 rows:

```
possibleclasses <- expand.grid(levels(as.factor(instances$Outlook)),levels(as.factor(instances$Temp)),le
colnames(possibleclasses)<-c("Outlook", "Temp", "Humid", "Windy") #this fanciness just generates a list
head(possibleclasses)
```

```
##    Outlook Temp Humid Windy
## 1 overcast cool  high FALSE
## 2    rainy cool  high FALSE
```

```
## 3    sunny cool  high FALSE
## 4 overcast  hot  high FALSE
## 5    rainy  hot  high FALSE
## 6    sunny  hot  high FALSE
```

```r
length(possibleclasses[,1])
```

```
## [1] 36
```

Now as an example, let's look at the distance between the first row of the possible classes table and the instances table:

```r
possibleclasses[1,]
```

```
##    Outlook Temp Humid Windy
## 1 overcast cool  high FALSE
```

```r
instances[1,]
```

```
##   Outlook Temp Humid Windy
## 1   sunny  hot  high FALSE
```

```r
distance(possibleclasses[1,], instances[1,])
```

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

We can see that the Manhattan distance is 2, since the rows differ for 2 variables.

Let's get around to finally answering the question. I'm interpreting a "unique" answer to mean a possible observation has exactly 1 row of the data for which its Manhattan distance is minimum, and an "unambiguous" answer to be one for which all of the rows of the data which are at minimum distance have the same value for the outcome, "play". Let's start by generating a table where we calculate all the distances

```r
distancesbyrow<-data.frame() #36 rows iterated by i for possibles, 14 columns iterated by j for instanc

for(i in 1:36){
  for(j in 1:14){
    distancesbyrow[i,j]=distance(possibleclasses[i,],instances[j,])  # calculate the distance between e
  }
}

head(distancesbyrow)
```

```
##   V1 V2 V3 V4 V5 V6 V7 V8 V9 V10 V11 V12 V13 V14
## 1  2  3  1  2  2  3  2  2  2   3   4   2   2   3
## 2  2  3  2  1  1  2  3  2  2   2   4   3   3   2
## 3  1  2  2  2  2  3  3  1  1   3   3   3   3   3
## 4  1  2  0  2  3  4  3  2  3   3   4   2   1   3
## 5  1  2  1  1  2  3  4  2  3   2   4   3   2   2
## 6  0  1  1  2  3  4  4  1  2   3   3   3   2   3
```

Now let's create a table that stores the minimum distance for each possible observation, counts the number of occurrences for that minimum distance, and finally determines whether or not all of those instances of the minimum distance have matching values for "play":

```r
distanceresults<-data.frame() #36 rows, one for each possible class. first column is minimum distances,

for(i in 1:36){ #traversing across the 36 possible cases
  distanceresults[i,1]<-min(distancesbyrow[i,]) #the minimum distance for each row of distancesbyrow
  distanceresults[i,2]<-sum(distanceresults[i,1]==distancesbyrow[i,]) #the number of occurences of this
```

```r
  distanceresults[i,3]<-instancedata[which.min(distancesbyrow[i,]),5] ## pulls the "play" value for the
  for(j in 1:14){ ## traversing across the 14 distances for one possible case
    if(distancesbyrow[i,j]==distanceresults[i,1] & instancedata[j,5]!=distanceresults[i,3]){
      distanceresults[i,3]<-"amb"
    } ## if the distance for this case is minimum AND the playvalue is not equal to the first playvalue
  }
}

colnames(distanceresults)<-c("minimumdistance","occurrences", "playvalue")
head(distanceresults)
```

```
##   minimumdistance occurrences playvalue
## 1               1           1       yes
## 2               1           2       yes
## 3               1           3       amb
## 4               0           1       yes
## 5               1           3       amb
## 6               0           1        no
```

```r
sum(distanceresults$occurrences==1) # the number of possible classes that have a unique closest neighbo
```

```
## [1] 17
```

```r
sum(distanceresults$playvalue!="amb") # the number of unambiguous results
```

```
## [1] 26
```

So to finally answer the question, there are 17 possible cases with a unique closest neighbor in the data set, and 26 with an unambiguous result in the playvalues of its closest neighbors.

Question 3

Well, the maximum number of answers for "playvalue" is 2, since playvalue can be either "yes" or "no", and all of 10 of the ambiguous cases identified above will have both "yes" and "no" for playvalue in their nearest neighbors. Let me instead hand you the possible case with the highest number of nearest neighbors and show you that it's ambiguous:

```r
which.max(distanceresults$occurrences) #which possible case has the highest number of nearest neighbors
```

```
## [1] 27
```

```r
possibleclasses[27,] #let's look at what the case looks like
```

```
##    Outlook Temp Humid Windy
## 27   sunny mild  high  TRUE
```

```r
distanceresults$minimumdistance[27] #what is the minimum distance?
```

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

```r
distanceresults$occurrences[27] #how many nearest neighbors does it have?
```

```
## [1] 5
```

```r
instancedata[c(2,8,11,12,14),] #here are the five rows that are distance 1 from this possible case, wit
```

```
##     Outlook Temp  Humid Windy Play.
## 2     sunny  hot   high  TRUE    no
## 8     sunny mild   high FALSE    no
## 11    sunny mild normal  TRUE   yes
```

```
## 12 overcast mild   high  TRUE   yes
## 14    rainy mild   high  TRUE    no
```

Question 4

The distances in the spreadsheet are Manhattan distances - they're the linear sum of the weighted distances of the case in question against the dataset. The Euclidean distance would be the sum of the squares of the weighted distances. These happen to be equivalent when the weight is 1.

Question 5

The weights change the importance of each variable in our distance calculation; a higher weight will cause a non-matching variable to have a higher "distance" between the observation and the data.

Question 6

We did this by just rerunning our code from part 3, changing the weights to see what we could get for the highest number of unambiguous possible classes. We did find a set of weights that made all possible classes give an unambiguous result. Our initial instinct was to assign higher weights to the variables with a smaller number of possible values, but that turned out to be the wrong way to go. Here's a table of the results:

```r
result1<-c(1,1,2,2,30)
result2<-c(1,1,2,4,30)
result3<-c(2,2,1,1,31)
result4<-c(4,3,1,1,33)
result5<-c(6,4,2,1,36)

resultstable<-rbind(result1, result2, result3, result4, result5)
colnames(resultstable)<-c("Outlookweight", "Tempweight", "Humidweight", "Windyweight", "Unambiguous")
pander(resultstable)
```

	Outlookweight	Tempweight	Humidweight	Windyweight	Unambiguous
result1	1	1	2	2	30
result2	1	1	2	4	30
result3	2	2	1	1	31
result4	4	3	1	1	33
result5	6	4	2	1	36