Hw 7

Luca Buchoux

12/3/2024

Recall that in class we showed that for randomized response differential privacy based on a fair coin (that is a coin that lands heads up with probability 0.5), the estimated proportion of incriminating observations $\hat{P}$ [1] was given by $\hat{P} = 2\hat{\pi} - \frac{1}{2}$ where $\hat{\pi}$ is the proportion of people answering affirmative to the incriminating question.

I want you to generalize this result for a potentially biased coin. That is, for a differentially private mechanism that uses a coin landing heads up with probability $0 \leq \theta \leq 1$, find an estimate $\hat{P}$ for the proportion of incriminating observations. This expression should be in terms of θ and $\hat{\pi}$.

Let A be a randomized algorithm. Consider a binary question where people report the truth a with probability p, or they flip their answer to 1-a with probability 1-p. Now consider D, D' to be datasets of responses that differ by one response.
Consider the case when the output is a.
Then $\frac{P[A(1)=1]}{P[A(0)=1]} = \frac{P[Output=1|Input=1]}{P[Output=1|Input=0]} = \frac{p}{1-p}$
Also, $\frac{P[A(0)=1]}{P[A(1)=1]} = \frac{1-p}{p}$
Thus $\frac{P[A(D)\in 1]}{P[A(D')\in 1]} = \frac{p}{1-p} = e^{ln(\frac{p}{1-p})}$

Next, show that this expression reduces to our result from class in the special case where $\theta = \frac{1}{2}$.

$\frac{P[Output=1|Input=1]}{P[Output=1|Input=0]} = \frac{3/4}{1/4} = 3$

Part of having an explainable model is being able to implement the algorithm from scratch. Let's try and do this with `KNN`. Write a function entitled `chebychev` that takes in two vectors and outputs the Chebychev or L^∞ distance between said vectors. I will test your function on two vectors below. Then, write a `nearest_neighbors` function that finds the user specified k nearest neighbors according to a user specified distance function (in this case L^∞) to a user specified data point observation.

[1] in class this was the estimated proportion of students having actually cheated

```r
#student input
#chebychev function
chebychev <- function(x,y) {
  max(abs(x-y))
}
#nearest_neighbors function
nearest_neighbors<-function(x,obs,k,dist){
  dists<-rep(0,nrow(x))
  for(i in 1:nrow(x)){
    dists[i]<-chebychev(x[i,],obs)
  }
  nn<-order(dists)[1:k]
  nn
}

x<- c(3,4,5)
y<-c(7,10,1)
chebychev(x,y)
```

```
## [1] 6
```

Finally create a `knn_classifier` function that takes the nearest neighbors specified from the above functions and assigns a class label based on the mode class label within these nearest neighbors. I will then test your functions by finding the five nearest neighbors to the very last observation in the `iris` dataset according to the `chebychev` distance and classifying this function accordingly.

```r
library(class)
df <- data(iris)
#student input
knn_classifier<-function(nn,class){
  names(sort(table(nn[,class]),decreasing = T))[1]
}

#data less last observation
x = iris[1:(nrow(iris)-1),]
#observation to be classified
obs = iris[nrow(iris),]

#find nearest neighbors
ind = nearest_neighbors(x[,1:4], obs[,1:4],5, chebychev)
as.matrix(x[ind,1:4])
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width
## 128          6.1         3.0          4.9         1.8
## 71           5.9         3.2          4.8         1.8
## 84           6.0         2.7          5.1         1.6
## 102          5.8         2.7          5.1         1.9
## 127          6.2         2.8          4.8         1.8
```

```
obs[,1:4]
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width
## 150          5.9           3          5.1         1.8
```

```
knn_classifier(x[ind,], 'Species')
```

```
## [1] "virginica"
```

```
obs[,'Species']
```

```
## [1] virginica
## Levels: setosa versicolor virginica
```

Interpret this output. Did you get the correct classification? Also, if you specified $K = 5$, why do you have 7 observations included in the output dataframe?

I did get the correct classification. There are only 5 observations in the output data frame.

Earlier in this unit we learned about Google's DeepMind assisting in the management of acute kidney injury. Assistance in the health care sector is always welcome, particularly if it benefits the well-being of the patient. Even so, algorithmic assistance necessitates the acquisition and retention of sensitive health care data. With this in mind, who should be privy to this sensitive information? In particular, is data transfer allowed if the company managing the software is subsumed? Should the data be made available to insurance companies who could use this to better calibrate their actuarial risk but also deny care? Stake a position and defend it using principles discussed from the class.

From a deontology standpoint, I don't believe the data should be made available for other uses. If the information was initially intended to assist in health care, then that is what it should be used for unless explicit consent is granted for other uses. It ties into when we discussed inform consent, the patients have a right to decide who can use or see their information, as well as the right to know how its used.

I have described our responsibility to proper interpretation as an *obligation* or *duty*. How might a Kantian Deontologist defend such a claim?

Proper interpretation is a responsibility because it ensures that models can be properly understood and that any biases present are clear to see. This is essential because muddying up the interpretation can lead to confusion and ambiguity. Essentially, whatever consequences come from some statistical model don't really matter as long as the interpretation is clear and any biases or shortcomings are plain to see. As long as it can be properly interpreted, it is up to users to make their own judgement about it.