B529: Homework 4

Nathan Byers

Wednesday, April 21, 2015

Question 1

For 2-dimensional data points, we define a kernel function $K(\mathbf{x}, \mathbf{x}') = (1 + \mathbf{x}^{\mathrm{T}} x')^3$, what is its corresponding transformation function $z = \Phi(x)$. (15 points)

Answer 1

$$K(\mathbf{x}', \mathbf{x}) = (1 + \mathbf{x}^{\mathrm{T}}\mathbf{x}')^3 = \left(1 + \begin{bmatrix} x_1 & x_2 \end{bmatrix} \begin{bmatrix} x'_1 \\ x'_2 \end{bmatrix}\right)^3$$

$$= (1 + x_1 x'_1 + x_2 x'_2)(1 + x_1 x'_1 + x_2 x'_2)(1 + x_1 x'_1 + x_2 x'_2)$$

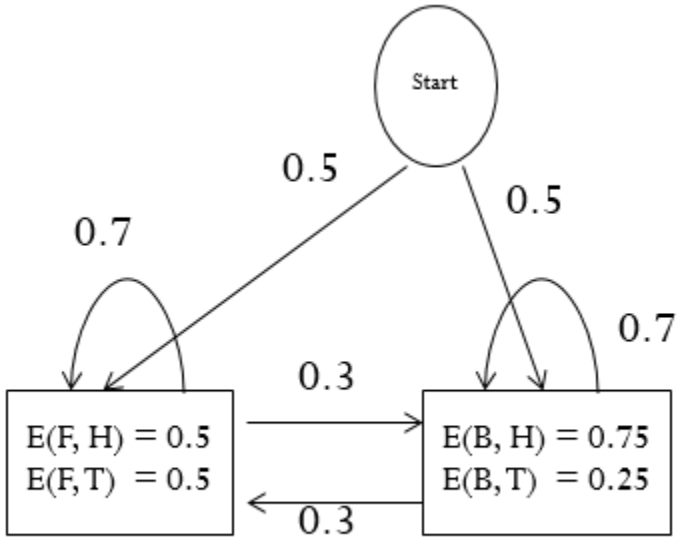
$$= (1 + 2x_1 x'_1 + 2x_2 x'_2 + x_1^2 x'^2_1 + 2x_1 x'_1 x_2 x'_2 + x_2^2 x'^2_2)(1 + x_1 x'_1 + x_2 x'_2)$$

$$= 1 + 3x_1 x'_1 + 3x_2 x'_2 + 3x_1^2 x'^2_1 + 6x_1 x'_1 x_2 x'_2 + 3x_2^2 x'^2_2 + x_1^3 x'^3_1 + 3x_1^2 x'^2_1 x_2 x'_2 + 3x_1 x'_1 x_2^2 x'^2_2 + x_2^3 x'^3_2$$

$$\Phi(\mathbf{x})^{\mathrm{T}}\Phi(\mathbf{x}) = \begin{bmatrix} 1 & \sqrt{3}x_1 & \sqrt{3}x_2 & \sqrt{3}x_1^2 & \sqrt{6}x_1 x_2 & \sqrt{3}x_2^2 & x_1^3 & \sqrt{3}x_1^2 x_2 & \sqrt{3}x_1 x_2^2 & x_2^3 \end{bmatrix} \begin{bmatrix} 1 \\ \sqrt{3}x'_1 \\ \sqrt{3}x'_2 \\ \sqrt{3}x'^2_1 \\ \sqrt{6}x'_1 x'_2 \\ \sqrt{3}x'^2_2 \\ x'^3_1 \\ \sqrt{3}x'^2_1 x'_2 \\ \sqrt{3}x'_1 x'^2_2 \\ x'^3_2 \end{bmatrix}$$

Question 2

In the following Hidden Markov model, the observation sequence is TTHH. If the hidden path is BBFF, what is the probability that the sequence is generated by the path BBFF (10 points). If the hidden path is unknown, generate the graph for the decoding problem (15 points). What is the heaviest path in the graph? (10 points)



Answer 2

The probabilities of getting a heads or tails, given that it's a fair or biased coin, are:

$$P(H|F) = P(T|F) = 0.5, P(H|B) = 0.75, P(T|B) = 0.25$$

The transition probabilities are:

$$a_{\mathrm{FF}} = a_{\mathrm{BB}} = 0.7, a_{\mathrm{BF}} = a_{\mathrm{FB}} = 0.3$$

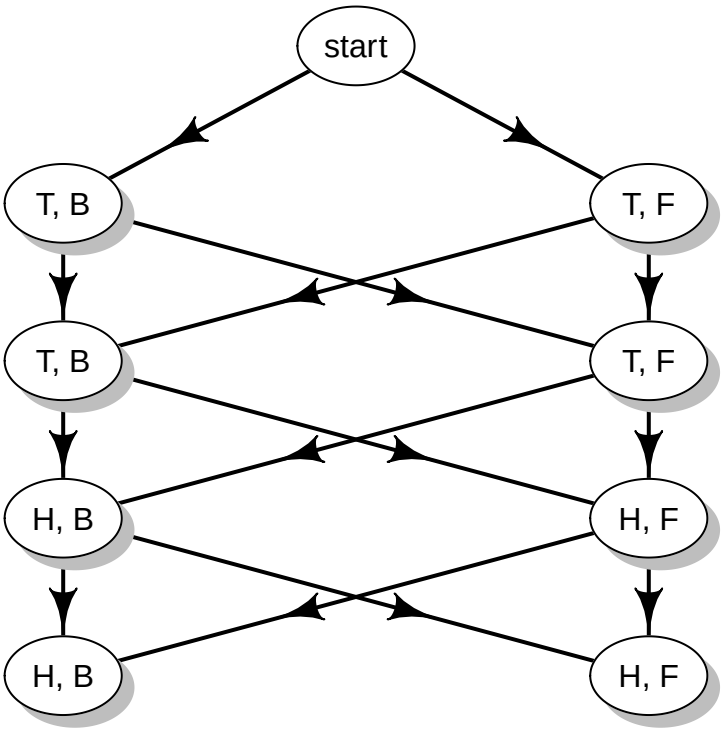
The probability that the fair/biased sequence BBFF would give a coin flip sequence TTHH is:

$$P(\mathbf{x} = TTHH|\pi = BBFF) = (0.5 \times 0.25)(0.7 \times 0.25)(0.3 \times 0.5)(0.7 \times 0.5) = 0.00115$$

I use the `diagram` package to draw the graph in R.

```
library(diagram)
# set margins
par(mar = c(1, 1, 1, 1))
# open a plotting region
openplotmat()
# first argument of coordinates() specifies the number of elements in each row
elpos <- coordinates (c(1, rep(2, 4)))
# create data.frame say from what position to where
from <- rep(1:7, each = 2)
to <- c(2:3, rep(4:5, 2), rep(6:7, 2), rep(8:9, 2))
arrpos <- matrix(ncol = 2, nrow = length(from))

# draw arrows
for(i in 1:length(from)){
  arrpos[i, ] <- straightarrow(to = elpos[to[i], ], from = elpos[from[i], ],
                lwd = 2, arr.pos = 0.6, arr.length = 0.5, endhead = FALSE)
}
```

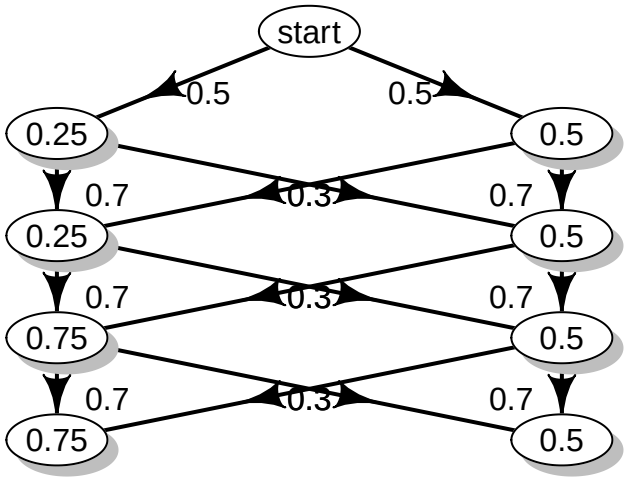
```
# draw circles
flip <- rep(c("T", "H"), each = 4)
pi <- rep(c("B", "F"), 4)
textellipse(elpos[1,], .05, lab = "start", shadow.size = 0)
for(i in 1:(length(flip))) {
  textellipse(elpos[i + 1,], 0.05, lab = paste(flip[i], pi[i], sep = ", "))
}
```



Here I redraw with the probabilities at each state, and add the transition probabilities for the arrows.

```
openplotmat()

# draw arrows
for(i in 1:length(from)){
  straightarrow(to = elpos[to[i], ], from = elpos[from[i], ],
                                lwd = 2, arr.pos = 0.6, arr.length = 0.5, endhead = FALSE)
}

# draw circles
prob <- sapply(1:length(flip), function(i){
  state <- paste(flip[i], pi[i])
  if(state == "T B"){
    .25
  }else if(state == "H B"){
    .75
```

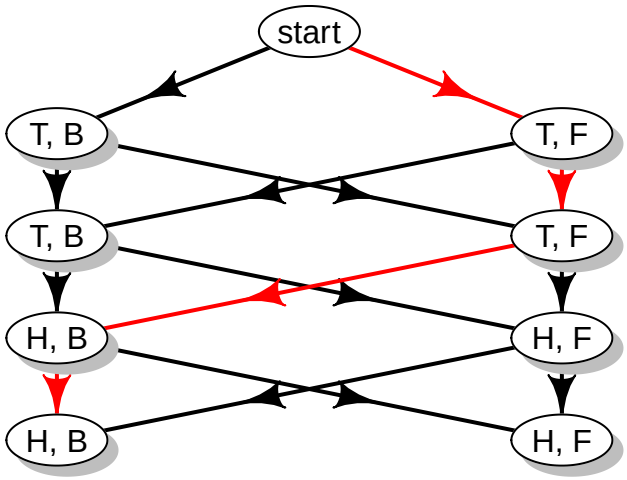
```
  }else{
    .5
  }
})
textellipse(elpos[1,], .05, lab = "start", shadow.size = 0)
for(i in 1:(length(flip))) {
  textellipse(elpos[i + 1,], 0.05, lab = prob[i])
}

pi_prob <- c(.5, .5, rep(c(.7, .3, .3, .7), 3))

# draw transition probabilities
for(i in 1:14){
  if((i %% 2) == 0){
    text(arrpos[i, 1] - 0.05, arrpos[i, 2], pi_prob[i])
  }else{
    text(arrpos[i, 1] + 0.05, arrpos[i, 2], pi_prob[i])
  }

}
```



Below is the heaviest edge.

```
openplotmat()
edge <- c(2, 6, 9, 11)
# draw arrows
```

```
for(i in 1:length(from)){
  if(i %in% edge){col <- "red"}else{col <- "black"}
  straightarrow(to = elpos[to[i], ], from = elpos[from[i], ], lcol = col,
                lwd = 2, arr.pos = 0.6, arr.length = 0.5, endhead = FALSE)
}

textellipse(elpos[1,], .05, lab = "start", shadow.size = 0)
for(i in 1:(length(flip))) {
  textellipse(elpos[i + 1,], 0.05, lab = paste(flip[i], pi[i], sep = ", "))
}
```



So the most likely hidden state is FFBB.

Question 3

Using the following data set for credit card application, compute the mutual information for each feature (15 points), compute the chi-square value for each feature (15 points)

Input			Output
Age	**Income**	**Gender**	**Risk**
<25	>50K	M	High
<25	>50K	F	High
≥25	<50K	F	High
≥25	>50K	F	Low
≥25	>50K	M	Low
<25	<50K	M	High

Answer 3

First I calculate the mutual information for each input.

```
age <- factor(c(0, 0, 1, 1, 1, 0), labels = c("<25", ">25"))
income <- factor(c(0, 0, 1, 0, 0, 1), labels = c(">50k", "<50k"))
gender <- factor(c(0, 1, 1, 1, 0, 0), labels = c("M", "F"))
risk <- factor(c(0, 0, 0, 1, 1, 0), labels = c("High", "Low"))

df <- data.frame(age, income, gender, risk)

df
```

```
##   age income gender risk
## 1 <25   >50k      M High
## 2 <25   >50k      F High
## 3 >25   <50k      F High
## 4 >25   >50k      F  Low
## 5 >25   >50k      M  Low
## 6 <25   <50k      M High
```

```
library(dplyr)

calcMutualInform <- function(input, outcome){
  # get priors
  input.prior = sapply(split(input, input),
                       function(level, total){length(level)/total},
                       total = length(input))
  outcome.prior = sapply(split(outcome, outcome),
                       function(level, total){length(level)/total},
                       total = length(input))
  # put in data frame, group by factor and outcome,
```

```
  # get total counts
  df = data.frame(input, outcome, I = 1)
  df = group_by(df, input, outcome)
  df.sum = summarize(df, count = sum(I))
  # calculate mutual information
  sum(unlist(apply(df.sum, 1, function(row, n = length(input)){
    union <- as.numeric(row["count"])/n
    prior1 <- as.numeric(input.prior[row["input"]])
    prior2 <- as.numeric(outcome.prior[row["outcome"]])
    union*log(union/(prior1*prior2))
  })))
}

lapply(df[, 1:3], calcMutualInform, outcome = df$risk)
```

```
## $age
## [1] 0.3182571
##
## $income
## [1] 0.174416
##
## $gender
## [1] 0
```

Now I get the χ^2 values.

```
calcChiSq <- function(input, outcome){

  Obs <- table(input, outcome)

  colsum <- colSums(Obs)
  rowsum <- rowSums(Obs)

  colmatr <- matrix(rep(colsum, 2), nrow = 2, byrow = TRUE)
  rowmatr <- matrix(rep(rowsum, 2), nrow = 2)

  E <- colmatr*rowmatr/sum(c(colsum, rowsum))

  Chi <- (Obs - E)^2/E

  sum(colSums(Chi))

}

lapply(df[, 1:3], calcChiSq, outcome = df$risk)
```

```
## $age
## [1] 9
##
## $income
## [1] 6
##
## $gender
## [1] 3
```

Question 4

Given a 2-dimensional data set (0,1), (2,3), (3,4), (2,4), (4,6). Please use R to find two resulting vectors in PCA and plot the data points and the vectors (20 points).

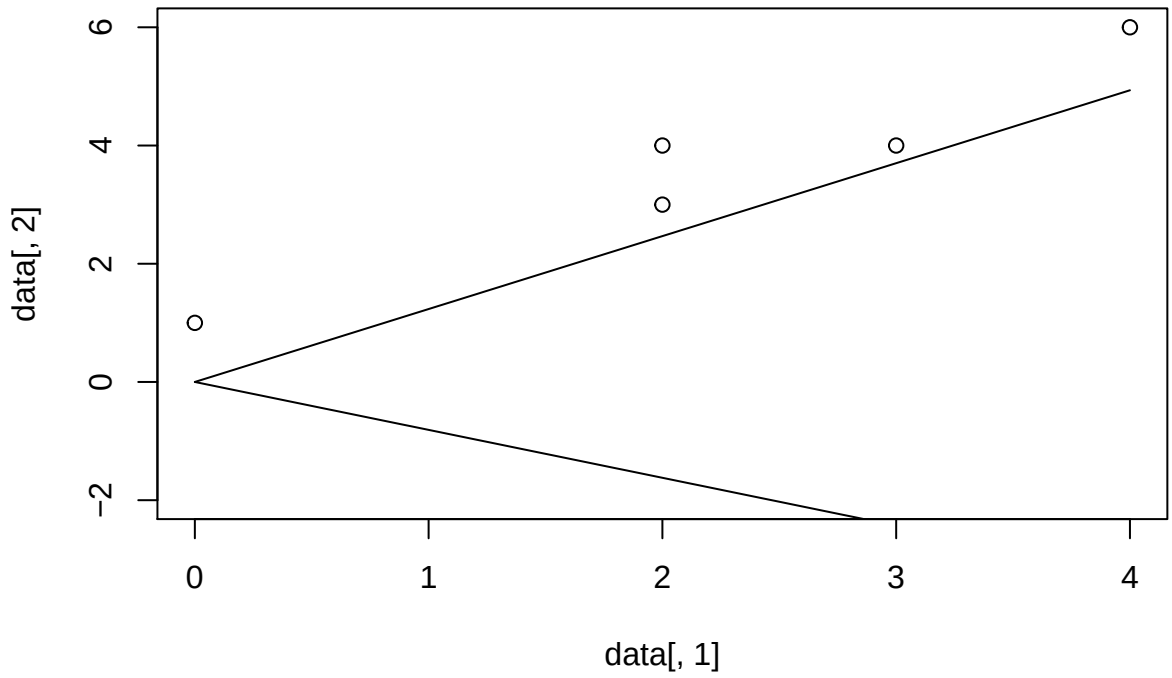
Answer 4

Here I find the covariance of the two vectors, use the `eigen()` function to get the eigen vectors, and plot vectors along with the data.

```
data <- matrix(c(0, 1, 2, 3, 3, 4, 2, 4, 4, 6), ncol=2, byrow=TRUE)
cova = cov(data)
ei = eigen(cova)
ei$vectors
```

```
##           [,1]       [,2]
## [1,] 0.6296989 -0.7768393
## [2,] 0.7768393  0.6296989
```

```
plot (data[,1], data[,2], ylim = c(-2, 6))
curve(x*ei$vectors[2, 1]/ei$vectors[1, 1], add=TRUE)
curve(x*ei$vectors[2, 2]/ei$vectors[1, 2], add = TRUE)
```



The vectors show the direction of the largest variance in the data space.