Exámen: Minería de datos

José Antonio García Ramírez

12/06/2016

```
## Loading required package: Matrix
##
## Attaching package: 'arules'
##
## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

1. Reglas de Asociación

a)Partimos del conjunto de datos dado en I, para encontrar patrones frecuentes y reglas de asociación realizaría una transformación para cambiar las variables de $Temperature$ y $Humedity$ en variables ordinales de la siguiente manera (cuidando de no crear demasiados niveles en cada variable para que la generación de patrones, que es una construcción de conjuntos, no sea tan amplia pues tenemos pocos registros):

Si $Temperature < 70$ entonces $Temperature = Frio$

Si $70 \leq Temperature < 80$ entonces $Temperature = Templado$

Si $Temperature \geq 80$ entonces $Temperature = Caliente$

Si $Humidity \leq 80$ entonces $Humidity = Seco$

Si $80 \leq Humidity < 90$ entonces $Humidity = Medio$

Si $Humidity \geq 90$ entonces $Humidity = Mojado$

Entonces los datos toman la siguiente forma:

	Outlook	Temperature	Humedity	wind	play
1	sunny	Caliente	Medio	FALSE	no
2	sunny	Caliente	Mojado	TRUE	no
3	overcast	Caliente	Seco	FALSE	yes
4	rain	Templado	Mojado	FALSE	yes
5	rain	Frio	Medio	FALSE	yes
6	rain	Frio	Seco	TRUE	no
7	overcast	Frio	Seco	TRUE	yes
8	sunny	Templado	Mojado	FALSE	no
9	sunny	Frio	Seco	FALSE	yes
10	rain	Templado	Medio	FALSE	yes
11	sunny	Templado	Seco	TRUE	yes
12	overcast	Templado	Mojado	TRUE	yes
13	overcast	Caliente	Seco	FALSE	yes
14	rain	Templado	Medio	TRUE	no

b)Encuentre las reglas de asociación con una cobertura mínima de 0.25 y una confianza mínima de 0.9 para un consecuente en la regla de Play=no y Play=yes.

Sólo existe una regla con las características especificadas

```
rules<-apriori(datos, parameter=list(support=0.25,confidence=0.9),
               appearance= list(rhs=c("play=no","play=yes")))
```

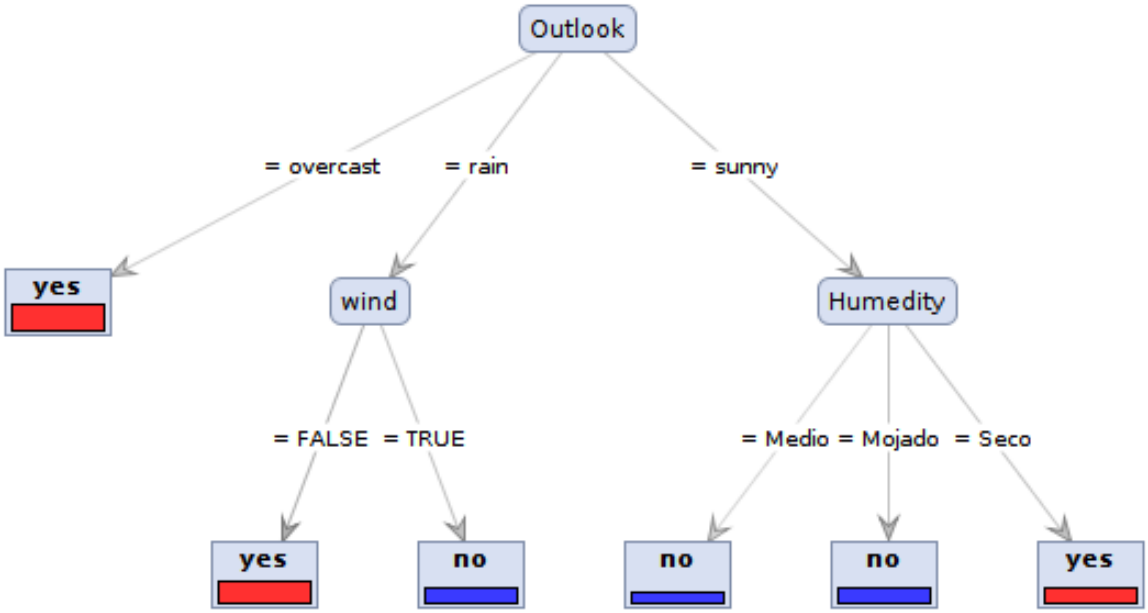
```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.9    0.1    1 none FALSE            TRUE    0.25      1     10
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 3
##
## set item appearances ...[2 item(s)] done [0.00s].
## set transactions ...[12 item(s), 14 transaction(s)] done [0.00s].
## sorting and recoding items ... [12 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [1 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(rules)
```

```
##   lhs                   rhs        support   confidence lift
## 1 {Outlook=overcast} => {play=yes} 0.2857143 1          1.555556
```

2. Árboles de Decisión

a) Utilizando la codificación que definí en el punto 1 sobre los atributos *Temperature* y *Humidity* el árbol de entropía es el siguiente:



b) La predicción para la tupla $(Outlook = rain, temperatura = 73, Humedity = 82, wind = TRUE)$ es: $play = NO$

3. Clasificador ingenuo de Bayes

a) ¿Cuál es su predicción para las siguientes tuplas?

$(Outlook = rain, Temperature = 73, Humedity = 82, wind = TRUE)$

$(Outlook = sunny, Temperature = 72, Humedity = 65, wind = TRUE)$

En esta sección utilizaré los datos originales dados en I, en vista de que disponemos de pocos datos para entrenar el modelo utilizo los 14 registros en la etapa de *training* y en la etapa de *test* considerare la estimación de las dos tuplas que nos interesan :

	Outlook	Temperature	Humidity	Wind	Play
1	sunny	85	85	FALSE	no
2	sunny	80	90	TRUE	no
3	overcast	83	78	FALSE	yes
4	rain	70	96	FALSE	yes
5	rain	68	80	FALSE	yes
6	rain	65	70	TRUE	no
7	overcast	64	65	TRUE	yes
8	sunny	72	95	FALSE	no
9	sunny	69	70	FALSE	yes
10	rain	75	80	FALSE	yes
11	sunny	75	70	TRUE	yes
12	overcast	72	90	TRUE	yes
13	overcast	81	75	FALSE	yes
14	rain	71	80	TRUE	no
15	rain	73	82	TRUE	?
16	sunny	72	65	TRUE	?

```
library(e1071)
DatosBayes<-read.csv("/home/fou/Desktop/bayes.csv")
ModeloBayes<-naiveBayes(Play ~ ., data=DatosBayes[1:14,])
test<-DatosBayes[15:16,]
predict(ModeloBayes,test)
```

```
## [1] yes yes
## Levels:  no yes
```

Para ambas tuplas la predicción es $Play = yes$

b) Una de las ventajas de los árboles de decisión es que su interpretación es sencilla pues no realizan transformaciones sobre las variables iniciales. Permiten analizar varias opciones comparando costos en función de probabilidades de ocurrencia de eventos. En desventaja es difícil elegir un árbol en vista de la variedad de criterios para construirlos (gini o entropía) aunado de la especificación de su tamaño (en ocasiones puede ser necesario podarlos) además requiere de un preprocesamiento de datos pues se define sobre variables no continuas.

El modelo de naive Bayes también es sencillo de transmitir y de interpretar pero posee desventajas teóricas fuertes en el planteamiento del modelo pues en la práctica es difícil encontrarse con datos cuyas variables se distribuyan normalmente y además independientemente unas de otras.

4. Knn

Consideremos los siguientes datos:

	a1	a2	a3	clase
1	+	20	0.10	0
2	+	70	0.30	1
3	+	44	0.08	1
4	+	32	0.41	1
5	-	81	0.11	1
6	-	29	0.06	0
7	-	43	0.21	0
8	-	83	0.38	1
9	+	19	0.10	?

a) Elijo fijar $k = 1$ con lo cual los métodos de reagrupamiento general y por clase coinciden.

Considerando sólo los atributos continuos $a2$ y $a3$

La predicción para la novena tupla tomando como *training* los demás datos es 0

```
conglomerados<-read.csv("/home/fou/Desktop/class.csv")
a<-(knn(conglomerados[,3:4],data.frame(a2=19,a3=.10),cl=conglomerados[,"clase"],k=1))
as.character(a)
```

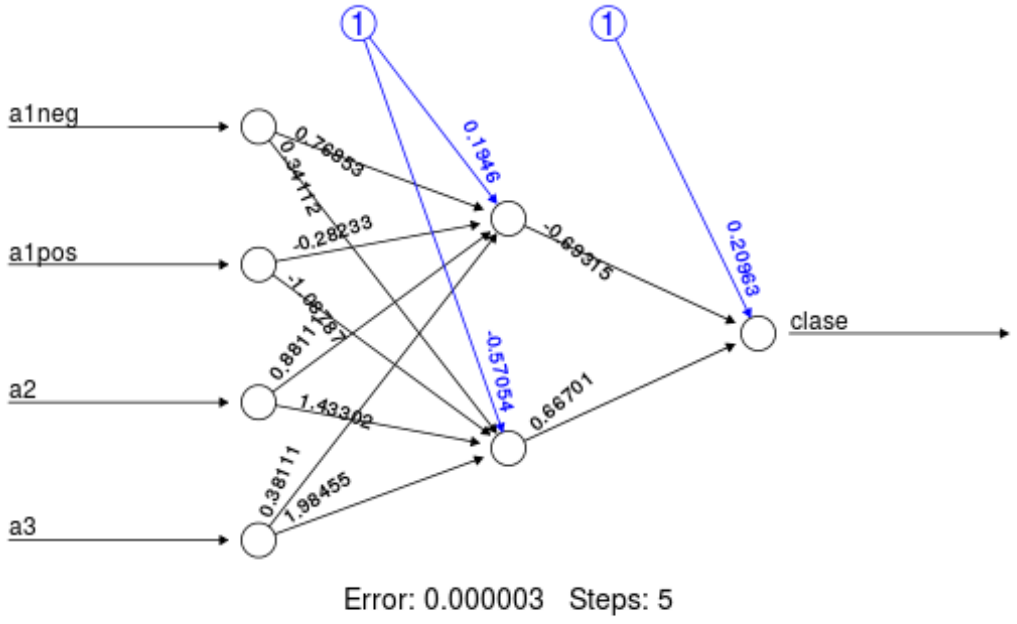
```
## [1] "0"
```

b) Es la misma predicción pues tome $k = 1$

5. BPNN

```
## Loading required package: grid
## Loading required package: MASS
```

a) Proponga la siguiente topología (ignorar coeficientes), con solo una capa oculta con dos nodos los cuales son suficientes para la estimación no lineal de la red (a diferencia de tener una sola capa con un solo nodo que es el caso de la regresión lineal). Siguiendo a [1] pág. 400 comienzo con pesos iniciales aleatorios en $[-0.7, 0.7]$ y una tasa de aprendizaje de 0.9



Con los siguientes pesos iniciales

```
pesos
```

```
## [[1]]
##                [,1]
##  [1,] -0.3282878716
##  [2,] -0.1790265405
```

```
##  [3,]  0.1019947087
##  [4,]  0.5714909060
##  [5,] -0.4176452965
##  [6,]  0.5577455590
##  [7,]  0.6225453760
##  [8,]  0.2251169095
##  [9,]  0.1807596615
## [10,] -0.6134992213
## [11,] -0.4116355951
## [12,] -0.4528205465
## [13,]  0.2618319853
```

Escogiendo la primer tupla y alimentando a la red los pesos finales son:

```
red$weights
```

```
## [[1]]
## [[1]][[1]]
##                 [,1]          [,2]
## [1,] -0.36396613365  0.5626706795
## [2,] -0.17902654051  0.6225453760
## [3,]  0.06631644665  0.2300420300
## [4,]  0.57149090599  0.1807596615
## [5,] -0.42121312275 -0.6130067093
##
## [[1]][[2]]
##                [,1]
## [1,] -0.1567220096
## [2,] -0.3389222281
## [3,]  0.4312912261
```

b) El atributo $a2$ requiere ser normalizado para tomar valores en el intervalo $[0, 1]$ por otro lado $a1$ se puede codificar como dos variables *dummies* que llame $a1neg$ y $a2pos$ Los datos de entrada de la red neuronal queda de la siguiente manera:

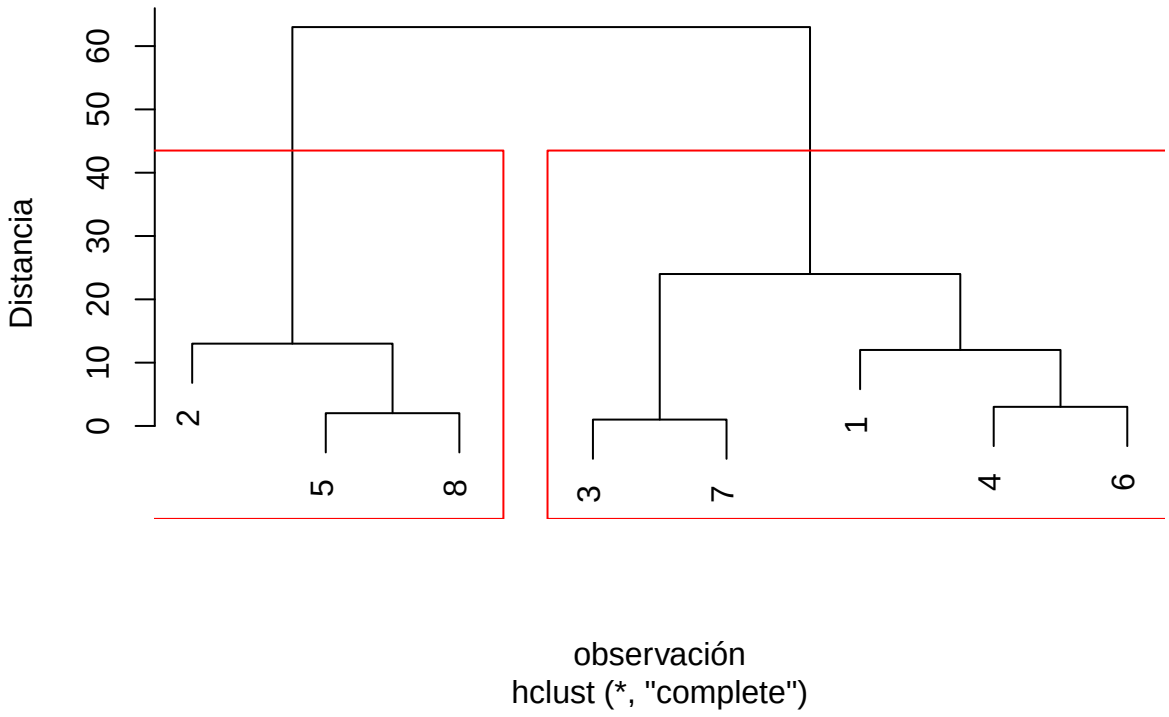
	a1neg	a1pos	a2	a3	clase
1	0.00	1.00	0.00	0.10	0
2	0.00	1.00	0.79	0.30	1
3	0.00	1.00	0.38	0.08	1
4	0.00	1.00	0.19	0.41	1
5	1.00	0.00	0.97	0.11	1
6	1.00	0.00	0.14	0.06	0
7	1.00	0.00	0.37	0.21	0
8	1.00	0.00	1.00	0.38	1

6. Formación de conglomerados.

a)

Utilizando los datos en bruto, sin ninguna transformación, la clasificación que obtenemos usando clusterización jerárquica es que las observaciones $\{2, 5, 8\}$ pertenecen al mismo conjunto y las otras a otro conjunto.

```r
conglomerados<-read.csv("/home/fou/Desktop/class.csv")
Datos<- conglomerados[,3:4]
Distancias <- dist(Datos)
Hcluster <- hclust(Distancias)
plot(Hcluster,main="",ylab="Distancia",xlab="observación")
rect.hclust(Hcluster,2)
```




```r
conglomerados$PrediccionJeraquica<-cutree(Hcluster,2)
```

b)

Utilizando el mismo conjunto de datos y con $k = 2$ encontramos la siguiente clasificación:

```r
conglomerados$PrediccionKmeans<-kmeans(Datos,centers=2)$cluster
```

	Instancia	a1	a2	a3	clase	PrediccionJeraquica	PrediccionKmeans
1	1	+	20	0.10	0	1	1
2	2	+	70	0.30	1	2	2
3	3	+	44	0.08	1	1	1
4	4	+	32	0.41	1	1	1
5	5	-	81	0.11	1	2	2
6	6	-	29	0.06	0	1	1
7	7	-	43	0.21	0	1	1
8	8	-	83	0.38	1	2	2

c)

De la matriz de confusión podemos afirmar que en vista de que ambos algoritmos clasifican igual, la precisión de ambos es de 75%

```r
library(caret)
```

```
## Loading required package: lattice
## Loading required package: ggplot2
```

```r
conglomerados$clase<-conglomerados$clase+1
confusionMatrix(conglomerados$clase, conglomerados$PrediccionKmeans)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 1 2
##          1 0 3
##          2 3 2
##
##                Accuracy : 0.25
##                  95% CI : (0.031854, 0.6508558)
##     No Information Rate : 0.625
##     P-Value [Acc > NIR] : 0.9943947
##
##                   Kappa : -0.6
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.000
##             Specificity : 0.400
##          Pos Pred Value : 0.000
##          Neg Pred Value : 0.400
##              Prevalence : 0.375
##          Detection Rate : 0.000
##    Detection Prevalence : 0.375
##       Balanced Accuracy : 0.200
##
##        'Positive' Class : 1
##
```

References

[1] Hastie, T., Tibshirani, R.,, Friedman, J. (2001). *The Elements of Statistical Learning*. New York, NY, USA: Springer New York Inc..