Proyecto Final

Fernando Hernández

7 de Julio de 2016

Introducción

Este conjunto de datos corresponde a un análisis de perfiles de expresión en macrófagos de individuos infectados con enfermedad de Chagas. Estas células fueron aisladas y estimuladas con el parásito *Trypanosoma cruzi*. Se tienen 32 chips que corresponden al conjunto de pacientes y 10 que corresponden al conjunto de controles. El objetivo de este análisis es intentar identificar patrones partículares de expresión tanto en pacientes como en controles.

Se cargan los datos en **R**, debido a costo computacional se limita el conjunto de información a sólo 5000 columnas (mayor número de columnas requiere mas memoria **RAM**)

```
Data <- read.table("data_2.csv", header = TRUE, sep = ",", row.names = 1)
tData <- t(Data)
tData <- as.data.frame(tData, header = T )
tData <- tData[,1:5000]
```

Se cargan las librerías para el análisis de los datos.

```
library(nnet)
library(neuralnet)
```

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

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

```
library(C50)
library(kernlab)
```

Se procede a definir los conjuntos de entrenamiento y de prueba para la realización de los distintos algoritmos.

```
training <- tData[1:30,]
test <- tData[31:42,]
preds <- names(tData[, names(tData) != "Enfermo"])
fm <- as.formula(paste("Enfermo~", sep= "",paste(preds, collapse = "+", sep = "")))
```

Algoritmo de Support Vector Machines

Se aplica el algoritmo sobre el conjunto de entrenamiento, construyendo primero el clasificador empleando el kernel *vanilladot*

```
#m <- neuralnet(fm, data = training)
data_Clasifier <- ksvm(Enfermo ~ ., data = training, kernel = "vanilladot")
```

```
##  Setting default kernel parameters
```

```
data_Clasifier
```

```
## Support Vector Machine object of class "ksvm"
##
## SV type: eps-svr  (regression)
##  parameter : epsilon = 0.1  cost C = 1
##
## Linear (vanilla) kernel function.
##
## Number of Support Vectors : 29
##
## Objective Function Value : -0.0036
## Training error : 0.009678
```

Ahora se procede a evaluar el rendimiento del modelo empleando la función *predict*.

```
data_Prediction <- predict(data_Clasifier, test)
data_Prediction
```

```
##                 [,1]
## P4.ME.CEL  0.9876788
## P4.MNE.CEL 0.7966016
## P5.ME.CEL  0.7270765
## P5.MNE.CEL 0.6827424
## P6.ME.CEL  0.7745444
## P7.ME.CEL  0.8224533
## P8.ME.CEL  0.6460120
## P9.ME.CEL  0.8867285
## C4.ME.CEL  0.2352994
## C4.MNE.CEL 0.1560890
## C5.ME.CEL  0.6641021
## C5.MNE.CEL 0.4451158
```

Luego con el siguiente comando se verifica que la información obtenida concuerde con el conjunto de datos de prueba, adicionalmente se realiza una tabla para verificación.

```
agreement <- data_Prediction == test$Enfermo
table(agreement)
```

```
## agreement
## FALSE
##    12
```

Ahora se procede a intentar mejorar el rendimiento del modelo empleando para esto el kernel **rbfdot**

```
data_classifier_rbf <- ksvm(Enfermo ~ ., data = training,
                            kernel = "rbfdot")
data_prediction_rbf <- predict(data_classifier_rbf, test)
agreement_rbf <- data_prediction_rbf == test$Enfermo
table(agreement_rbf)
```

```
## agreement_rbf
## FALSE
##    12
```

Algoritmo C50

```
#training <- as.factor(training$Enfermo)
#data_model_c50 <- C5.0(training[2:5000], training_50$Enfermo)
```