CURSOS DE VERANO 2014

Aproximación práctica a la ciencia de los datos y Big Data

Resolución de casos prácticos con R

José Manuel Benítez Sánchez

Contenido

- CRAN
- Clasificación
- Clustering

CRAN

CRAN: contenido

- Objetivo
- Contenido
- Task Views
- Mirrors

The Comprehensive R Archive Network

- cran.r-project.org
- CRAN: “network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R”
- CRAN Repository Policy
- Filtro de envíos. Compromiso de mantenimiento

CRAN: Contenido

- Software de R:
  - Entorno principal
  - Paquetes contribuidos, organizados en "Task views"
- The R Journal
- Motor de búsqueda
- Documentación
  - Manuales
  - Vignettes
- Mirrors

CRAN: Task views destacadas

- Bayesian
- Cluster
- ExperimentalDesign
- Graphiucs
- HighPerformanceComputing
- MachineLearning
- Mlutivariate
- NumericalMathematics
- Optimization
- ReproducibleResearch
- Robust
- TimeSeries
- WebTechnologies

Instalación de paquetes

- Con una conexión a Internet disponible, es fácil instalar nuevos paquetes:

```
install.packages("tree")
```

- Realiza una resolución de dependencias
- Descarga, configura, compila e instala
- Uso con library

```
library(tree)
```

Gestión de paquetes

- Usar un paquete: `library(nnet)`
- Listado de paquetes instalados: `library()`
- Trayectoria de búsqueda: `search()`
- Desconectar un paquete: `detach(package:nnet)`
- Fijar un mirror: `chooseCRANmirror()`

Clasificación

Existen paquetes particularizados para muchos otros formatos de datos (propietarios y libres):

- SPSS
- SAS
- Stata
- systat

Problemas de clasificación

- Manejo de datos
- Visualización
- Clasificadores
- Evaluación: medidas de eficacia y procedimientos

Manejo de datos

- Importar datos:
  - csv
  - Excel
  - SQL
  - Otros programas
- Representación:
  - dataframe
  - table
- Operaciones:
  - Selección
  - Proyección
  - Transformación

Importar CSV

- Hay un par de funciones para ello: read.table y read.csv

```
df <- read.table("donut.csv", header=TRUE, sep=",")
```

- Hay diferentes variantes, que fijan determinados argumentos
- Para leer matrices, mejor scan

Importar ficheros de Excel

- Existen distintos paquetes para leerlos: XLConnect, RODBC, xlsx, …

```
wk <- loadWorkbook("eBayAuctions.xls")

data <- readWorksheet(wk, sheet=1)
```

Importar desde bases de datos

- Paquete: RMySQL
- Conexión a la BD
- Consulta
- Procesar datos
- Cerrar conexión

Obtención de datos de MySQL

```
library(RMySQL)

con <- dbConnect(MySQL(), user="username",
password="clave", dbname="database", host="server")

on.exit(dbDisconnect(con))

 rs <- dbSendQuery(con, "select col1, col2, col3 from
table;")

data <- fetch(rs, n=-1)

dbClearResult(rs)

dbDisconnect(con)
```

Importar datos desde la web

- Importar el fichero, si ya está descargado
- HTML
- XML
- JSON
- APIs que producen XML/JSON
- Existen múltiples paquetes para ello: RCurl, XML, RJSNONIO, rOpenSci, …

Algunos ejemplos

```
fpe <-
read.table("http://dicits.ugr.es/datasets/effort.dat")


download.file('http://finance.yahoo.com/q?s=aapl&x=0&y
=0','quote.html')
```

Conjuntos de datos

- iris: `data(iris)`
- pima: `data(PimaIndianDiabetes, package="mlbench")`
- Affairs: `data(Affairs, package="AER")`
- cats: `data(cats, package="MASS")`
- eBayAuctions.xls

Conjuntos de entrenamiento y prueba

```
data(iris)

m <- nrow(iris)

test <- sample(1:m, size=round(m/3),
replace=FALSE)

train.set <- iris[-test]

test.set <- iris[test]
```

Validación cruzada: cvTools

```
Library(cvTools)
k <- 10
folds <- cvFolds(nrow(data), K=k)
for (i in 1:k) {
  train <- data[folds$subsets[folds$which != i], ]
  test  <- data[folds$subsets[folds$which == i], ]
}
```

Medidas de eficacia

- Porcentaje de aciertos (fallos)
- Matriz de confusión
- ROC
- Clasificación desequilibrada: especificidad, sensibilidad, …

Clasificadores en R: conceptos generales

- Formula:
  - Species ~ .; Species ~ Sepal.Length + Petal.Length; Species ~ .-Petal.Width;
- predict
- fitted
- summary
- print
- plot

Paquetes para clasificación

- caret: Classification and Regression Training
- Rweka: Interfaz de R para Weka
- RSNNS
- Paquetes específicos para técnicas concretas

Clasificadores

- kNN
- Clasificador lineal: Regresión logística
- Naïve Bayes
- Árboles de clasificación
- Random Forests
- Redes neuronales artificiales
- SVM

Paquetes que implementan kNN

- kknn
- DMwR
- caret
- Weka

kknn: preparación

```
library(kknn)

data(iris)

m <- nrow(iris)

val <- sample(1:m, size = round(m/3), replace =
FALSE, prob = rep(1/m, m))

iris.learn <- iris[-val,]

iris.valid <- iris[val,]
```

Selección de conjuntos

kknn: clasificador y evaluación

```
iris.kknn <- kknn(Species~., iris.learn,
iris.valid, distance = 1, kernel =
“rectangular“)

summary(iris.kknn)

fit <- fitted(iris.kknn)

table(iris.valid$Species, fit)
```

Probar con distintos:
- K
- kernel

kNN con DMwR

```
library(DMwR)

m <- nrow(iris)

idx.iris <- sample(1:m, as.integer(0.7*m))

train.iris <- iris[idx.iris,]

test.iris <- iris[-idx.iris,]

nn5 <- kNN(Species ~ ., train.iris, test.iris,
k=5)

table(test.iris$Species, nn5)
```

kNN en caret (1/2)

```
TrainData <- iris[,1:4]

TrainClasses <- iris[,5]


knnFit1 <- train(TrainData, TrainClasses,
                 method = "knn",
                 preProcess = c("center", "scale"),
                 tuneLength = 10,
               trControl = trainControl(method = "cv"))
```

kNN en caret (2/2)

```
knnFit2 <- train(TrainData, TrainClasses,
 method = "knn",
 preProcess = c("center", "scale"),
 tuneLength = 10,
 trControl = trainControl(method = "boot"))
```

kNN con Weka

```
install.packages("RWeka")
library(RWeka)
iris <- read.arff(system.file("arff",
"iris.arff", package = "RWeka"))
classifier <- IBk(class ~ ., data = iris)
summary(classifier)

classifier <- IBk(class ~ ., data = iris,
control=Weka_control(K = 20, X = TRUE))

evaluate_Weka_classifier(classifier,
numFolds=10)
```

Regresión logística en R: preparación

```
data(Affairs, package="AER")

table(Affairs$affairs)

Affairs$ynaffair[Affairs$affairs > 0] <- 1

Affairs$ynaffair[Affairs$affairs == 0] <- 0

Affairs$ynaffair <- factor(Affairs$ynaffair,

  levels=c(0,1), labels=c("No", "Yes"))
```

Uso de factores

Regresión logística en R: clasificador

```
m <- glm(ynaffair~
gender+age+yearsmarried+children+religiousness +
education + occupation + rating, data=Affairs,
family=binomial())

summary(m)
```

```
m <- glm(ynaffair~
age+yearsmarried+religiousness + rating,
data=Affairs, family=binomial())
```

Conjunto de datos pima (1/2)

```
install.packages("mlbench")

library(mlbench)

data(PimaIndiansDiabetes)

pima <- PimaIndiansDiabetes


m <- glm(diabetes ~ . , data=pima,
family=binomial)
```

Conjunto de datos pima (2/2)

```
pred <- predict(m, type="response")

cl.pred <- rep("neg", nrow(pima))

cl.pred[pred > 0.5] <- "pos"

table(cl.pred, pima$diabetes)

mean(cl.pred == pima$diabetes)
```

Regresión logística multinomial

```
library(nnet)

model <- multinom(Species ~., data=iris)

summary(model)

p <- predict(model)

table(p, iris$Species)
```

Naïve Bayes

```
library(e1071)

model <- naiveBayes(Species ~., iris)

pred <- predict(model, iris[, 1:4])
```

Árboles de clasificación: paquetes

- tree
- rpart
- caret
- RWeka

Árboles con rpart

```
library(rpart)
data(iris)
tree <- rpart(Species ~ ., data= iris,
 method="class")
summary(tree)
printcp(tree)
plot(tree)
text(tree)
print(tree)
```

Árboles con tree

```
library(tree)
data(Carseats, package”ISLR”)
High <-  ifelse(Sales <= 8, “No”, “Yes”)
Carseats <- data.frame(Carseats,High)
tm <- tree(High ~ .-Sales, Carseats)
plot(tm)
text(tm)
```

Poda y mejora de árboles

```
cv.tree(tm, FUN=prune.misclass)

par(mfrow=c(1,2))

plot(cvt$size, cvt$dev, type="b")

plot(cvt$k, cvt$dev, type="b")

pruned.t <- prune.misclass(tm, best=9)
```

Árboles con RWeka

```
library(RWeka)

J48(Species ~ ., data=iris)
```

Random Forest (1/2)

```
install.packages("randomForest")

library(randomForest)

data(Boston)

train <- sample(1:nrow(Boston),
      as.integer(0.8*nrow(Boston)))


m <- randomForest(medv ~  ., data=Boston,
      subset=train, mtry=13, importance=TRUE)
```

Random Forests (2/2)

```
boston.test <- Boston[-train,]

test.result <- predict(m, newdata = boston.test)

mean((test.result - boston.test)^2)


m <- randomForest(medv ~ ., data=Boston,
      subset=train, mtry=13, ntree=25)
```

Otros paquetes para RandomForest

- `randomForestSRC`: Implementación unificada del modelo de Breiman para problemas de regresión, clasificación y "survival".
- `varSelRF`: Selección de variables
- `Boruta`: Selección de bariables
- `bigrf`: Para conjuntos de datos grandes

Boosting: AdaBoost

```
library(ada)
çdata("soldat")
n <- nrow(soldat)
set.seed(100)
ind <- sample(1:n)
trainval <- ceiling(n * .5)
testval <- ceiling(n * .3)
train <- soldat[ind[1:trainval],]
test <- soldat[ind[(trainval + 1):(trainval +
 testval)],]
valid <- soldat[ind[(trainval + testval
+ 1):n],]
```

Adaboost: llamada

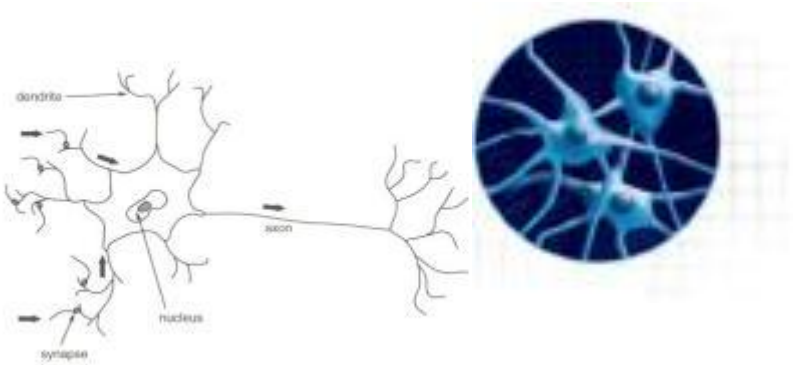
```
control <- rpart.control(cp = -1, maxdepth = 14,
 maxcompete = 1, xval = 0)

gen1 <- ada(y~., data = train, test.x =
 test[,-73], test.y = test[,73], type =
"gentle", control = control, iter = 70)

gen1 <- addtest(gen1, valid[,-73], valid[,73])
summary(gen1)
varplot(gen1)
```

Redes Neuronales Artificiales

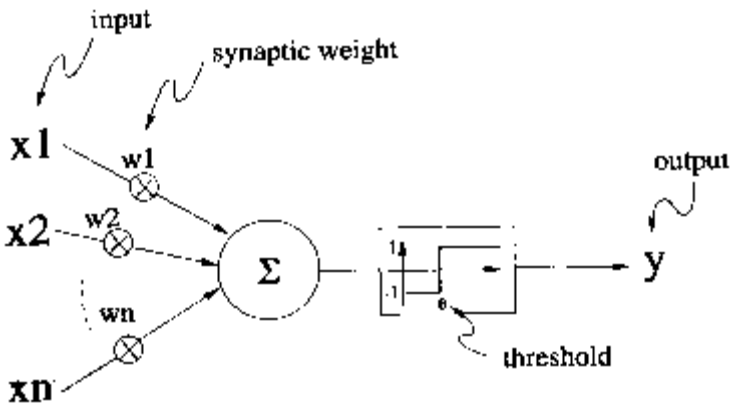
- Constituyen modelos de inteligencia computacional inspirados originalmente en el funcionamiento del cerebro humano (después han derivado a múltiples pardigmas distintos)
- Su propiedad más distintativa es las adaptabilidad, capacidad de aprendizaje, y por tanto, generalización, frente a nuevas entradas
- Pueden aprender (ser entrenadas) medinate métodos supervisados y no supervisados

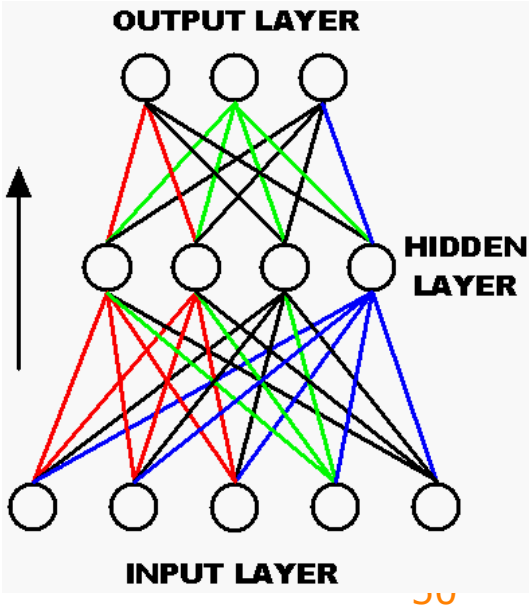
Redes Neuronales Artificiales

Basados en la simulación del comportamiento del Sistema Nervioso

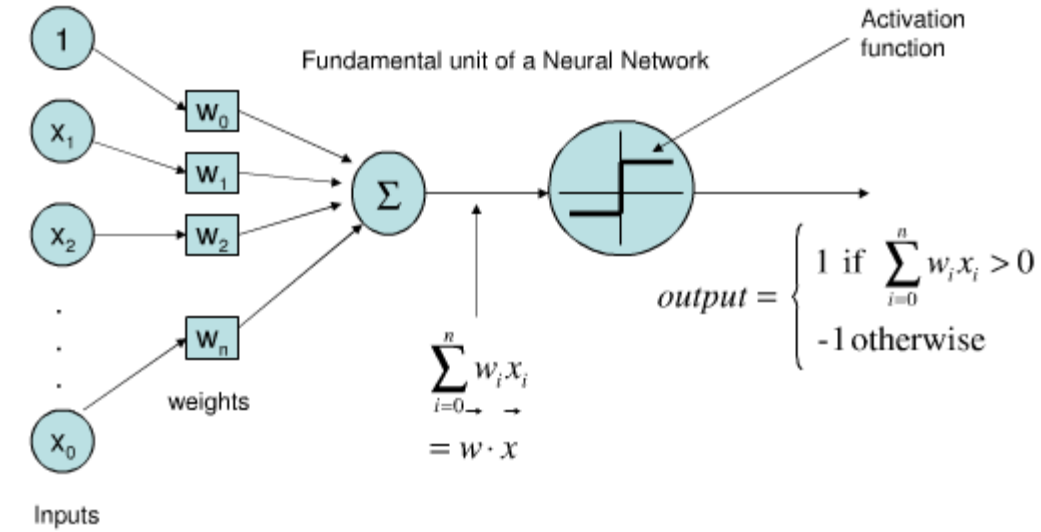


Paradigma de Aprendizaje Automático

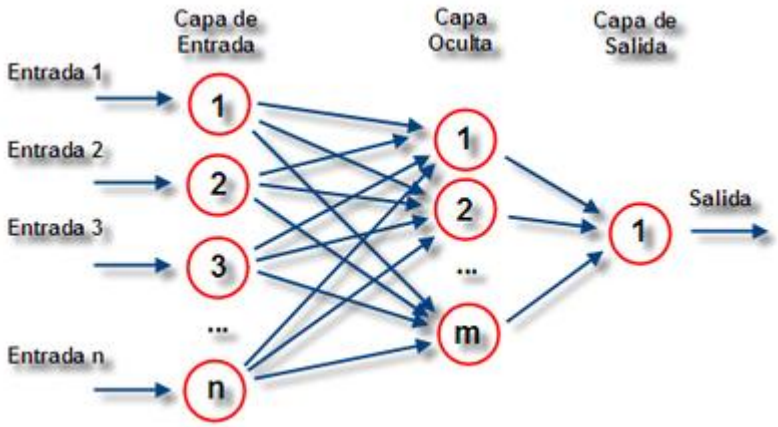




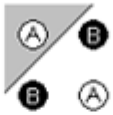
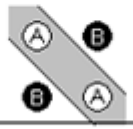
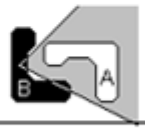
Perceptron


1
x_1
x_2
x_0
w_0
w_1
w_2
w_n
weights
Inputs
Fundamental unit of a Neural Network
Σ
Activation function
$\sum_{i=0}^{n} w_i x_i = w \cdot x$
$output = \begin{cases} 1 \text{ if } \sum_{i=0}^{n} w_i x_i > 0 \\ -1 \text{ otherwise} \end{cases}$

Perceptron multicapa



Perceptron multicapa

Estructura	Regiones de Desición	Problema de la XOR	Clases con Regiones Mezcladas	Formas de Regiones más Generales
1 Capa 	Medio Plano Limitado por un Hiperplano			
2 Capas 	Regiones Cerradas o Convexas			
3 Capas 	Complejidad Arbitraria Limitada por el Número de Neuronas			

Elementos de una RNA

- Tipos de unidades:
  - Activación lineal
  - Activiación sigmoide (logística, tanh, …)
- Arquitectura:
  - Organización (capas)
  - Conexiones: feedforward, feedback
- Algoritmo de entrenamiento
- Datos

Retropropagación de errores

- Extensión del método de aprendizaje del perceptron para redes con una o más capas ocultas
- Se basa en realizar un descenso en gradiente de la función de error
- Ajustes incrementales de error aplicando cambios sobre los parámetros (pesos sinápticos)
- Regulado por varios parámetros: tasa de aprendizaje, momento, …
- Multitud de variantes: QuickProp, Rprop, …

Tipos de RNAs

- Perceptron
- Perceptron multicapa (MLP)
- Redes de función de base Radial (RBF)
- Memorias asociativas (modelos Hopfield)
- Redes recurrentes
- Autoorganizativas (SOM, Kohonen)
- …

Redes neuronales artificiales: paquetes

- nnet
- neuralnet
- RSNNS
- caret
- Rweka

RNA con nnet

```
library(nnet)

model <- nnet(Species ~ ., data = iris, size=3)

predict(model, type="class")
```

```
model <- nnet(Species ~ ., data = iris, size=4,
maxit=500)
```

Probar a variar los parámetros:
- Size
- Rang
- Decay

RNA con neuralnet (1/2)

```
m <- nrow(iris)
iris <-  iris[sample(1:m,length(1:m)),
1:ncol(iris)]
irisValues <-  iris[,1:4]
irisTargets <- iris[,5]
irisDecTargets <-
decodeClassLabels(irisTargets)
iris <-  splitForTrainingAndTest(irisValues,
irisDecTargets, ratio = 0.15)
iris <-  normTrainingAndTestSet(iris)
```

RNA con neuralnet (2/2)

```
model <-  mlp(iris$inputsTrain, iris$targetsTrain, size =
5, learnFuncParams = c(0.1), maxit = 60, inputsTest =
iris$inputsTest,  targetsTest = iris$targetsTest)

predictions <-  predict(model, iris$inputsTest)

plotIterativeError(model)
plotRegressionError(predictions[,2],
iris$targetsTest[,2], pch = 3)
plotROC(fitted.values(model)[,2], iris$targetsTrain[,2])
plotROC(predictions[,2], iris$targetsTest[,2])
```

RSNNS

- El paquete más completo y flexible para el modelado de RNAs es SNNS
- RSNNS: Versión de SNNS para R
- Multitud de topologías y algoritmos de aprendizaje
- Topologías flexibles
- Diversas herramientas de análisis y visualización

Support Vector Machines

"Una **SVM** es un modelo de aprendizaje que se fundamenta en la *Teoría de Aprendizaje Estadístico*. La idea básica es encontrar un hiperplano canónico que maximice el margen del conjunto de datos de entrenamiento, esto nos garantiza una buena capacidad de generalización."

Representación dual de un problema

+

Funciones Kernel

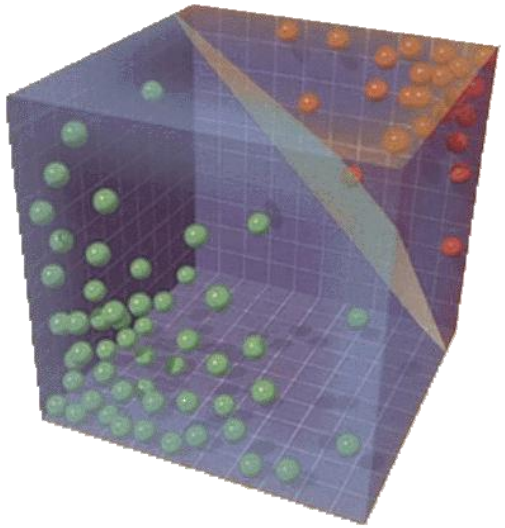
+

Teoría de Aprendizaje Estadística

+

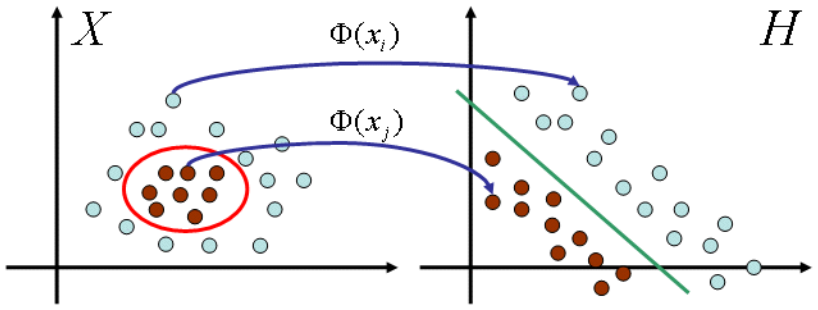
Teoría Optimización Lagrange

=



Métodos kernel (1)

- Una **SVM** es un **máquina de aprendizaje lineal** (requiere que los datos sean linealmente separables).
- Estos métodos explotan la información que proporciona el **producto interno (escalar)** entre los datos disponibles.
- La idea básica es:



Métodos kernel (2)

- **Problemas de esta aproximación:**
  - ¿Cómo encontrar la función ϕ?
  - El espacio de características inducido **H** es de **alta dimensión.**
    - **Problemas de cómputo y memoria.**

Métodos kernel (3)

- **Solución:**
  - Uso de funciones kernel.
  - **Función kernel =** producto interno de dos elementos en algún espacio de características inducido (**potencialmente de gran dimensionalidad**).
  - Si usamos una función kernel no hay necesidad de especificar la función ϕ**.**

Ejemplos de funciones kernel

- Polinomial: $K(x,y) = \langle x,y \rangle^d$
- Gausiano: $K(x,y) = e^{-\|x-y\|^2/2\sigma}$
- Sigmoide: $K(x,y) = \tanh(\alpha\langle x,y \rangle + \beta)$

Problema de optimización convexo

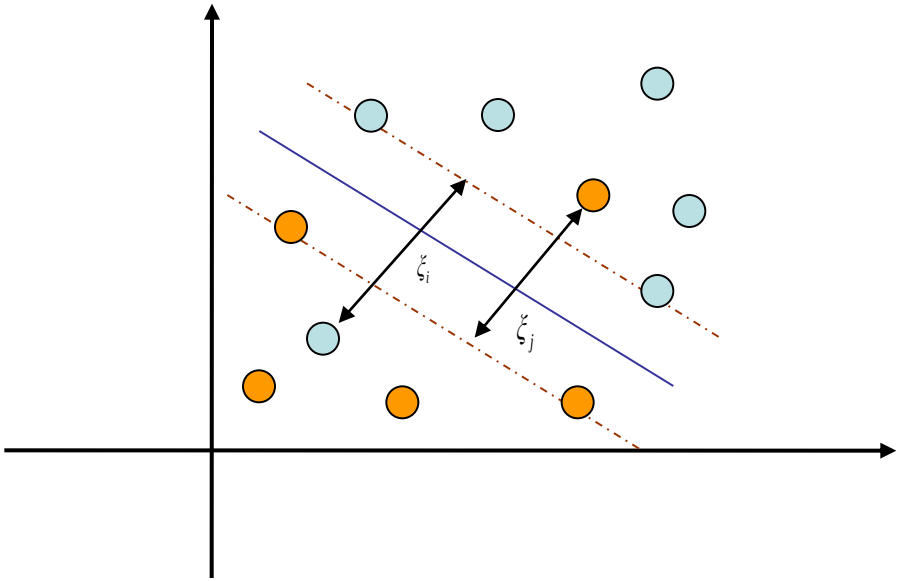
- Para resolver el problema de optimización planteado se usa la **teoría de Lagrange**.

Minimizar $$L(w,b)=\frac{1}{2}\langle w,w\rangle-\sum_{i=1}^{l}\alpha_i\left[y_i\left(\langle w,x_i\rangle+b\right)-1\right]$$

Condicionado a $$\alpha_i \geq 0$$

- Cada una de las variables α_i es un multiplicador de Lagrange y existe una variable por cada uno de los datos de entrada.

Clasificador de margen blando



$$y_i\left[\langle w, x_i\rangle + b\right] \geq 1 - \xi_i$$

Clasificación multicategoría

- Originariamente el **modelo de aprendizaje basado en SVMs** fue diseñado para **problemas de clasificación binaria**.
- De aquí en adelante consideramos problemas con datos pertenecientes a K clases diferentes.
- La generalización a la clasificación multicategoría ha seguido dos vertientes:
  - **Combinación de SVMs binarias.**
  - Resolución de un problema de optimización considerando todos los datos a la vez.
- Tres métodos principales:
  - SVMs 1 vs R
  - SVMs 1vs1
  - Grafo Acíclico Dirigido de Decisión (GADD)
  - FPWC

Support Vector Machines

```
library(e1071)

data(cats, package="MASS")

model <- svm(Sex ~. data= cats)

plot(model, cats)
```

SVM

```
m <- nrow(cats)
testindex <- sample(index, round(m/3))
testset <- cats[testindex,]
trainset <- cats[-testindex,]
model <- svm(Sex~., data = trainset)
p <- predict(model, newdata=testset[-1]))
tab <- table(pred = p, true = testset[,1])
```

Mejora del modelo

```
classAgreement(tab)

 tuned <- tune.svm(Sex~., data = trainset, gamma
= 10^(-6:-1), cost = 10^(1:2))

summary(tuned)
```

Otros clasificadores: Task view MachineLearning

- Regurarlized and Shrinkage Metods: lasso2, lars, glmnet
- Boosting: gbm, GAMBoost
- Bayesian methods: tgp
- Fuzzy rule-based Systems: frbs
- RoughSets
- …

Caso de estudio: eBayAuctions

```
library(XLConnect)

wk <- loadWorkbook("eBayAuctions.xls")

data <- loadWorksheet(wk, sheet=1)

summary(data)

data$currency <- factor(data$currency)

data$Category <- factor(data$Category)

df$Competitive <- factor(df$Competitive,
labels=c("no", "yes"))
```

Algunos clasificadores

```
nn <- nnet(Competitive~., data=df, size=5,
maxit=500)

 table(predict(nn, type="c"), df$Competitive)
```

Caso de estudio: Accidentes

- Accidentes.xls
- Manejo de datos:
  - Selección de instancias
  - Selección de columnas (feature selection)
- Manejo de NA
- Transformación de datos

Clustering

Tipos de Clustering

- ***Hierarchical*** – Nested set of clusters created.
- ***Partitional*** – One set of clusters created.
- ***Incremental*** – Each element handled one at a time.
- ***Simultaneous*** – All elements handled together.
- ***Overlapping/Non-overlapping***

CRAN cluster Task View

- Hierarchical Clustering:
  - cluster
  - fastcluster
  - dynamicTreeCut
- Partitioning Clustering:
  - apcluster
  - bayseclust
  - Kernlab
- Model-based Clustering:
  - EMCluster

K-means (sobre iris)

```
data <- iris[,1:4]

fit <- kmeans(data, 5)

summary(fit)

fit$centers

fit$clusters

fit$size
```

K-means: visualización

```
library(cluster)

clusplot(data, fit$cluster, color=TRUE, shade =
TRUE, labels=2, lines=0)

library(fpc)

plotcluster(data, fit$cluster)
```

K-means: wine data

```
data(wine, package="rattle")

fit.km <- kmeans(df, 3, nstart=25)

df <- scale(wine[-1])

fit.km <- kmeans(df, 3, nstart=25)

aggregate(wine[-1],
by=list(cluster=fit.km$cluster), mean)
```

K-means: número de clusters

```
library(NbClust)
set.seed(1234)
nc <- NbClust(df, min.nc=2, max.nc=15, method="kmeans")
table(nc$Best.n[1,])
barplot(table(nc$Best.n[1,]),
   xlab="Numer of Clusters", ylab="Number of Criteria",
   main="Number of Clusters Chosen by 26 Criteria")
```

Partitioning Around Medoids

```
f <- pam(data, 4)

names(f)

f$clustering

plot(f)
```

Clustering jerárquico “Ward”

```
d <- dist(data, method=“euclid”)

fit <- hclust(d, method=“ward”)

plot(fit)

groups <- cutree(fit, k=4)

rect.hclust(fit, k=5, border=“red”)
```

Clustering jerárquico basado en boosting

```
library(pvclust)

fit <- pvclust(data, method.hclust="ward",
method.dist="euclid")

plot(fit)

pvrect(fit, alpha=.95)
```

Fuzzy C-Means

```
library(e1071)

fcm <- cmeans(data, 4, iter.max=50,
method="cmeans")

plot(iris, col=fcm$cluster)
```

Clustering basado en modelos

```
library(mclust)

fit <- Mclust(data)

plot(fit)

summary(fit)
```

Caso de estudio: prestamo.xls

```
wkp <- loadWorkbook("prestamo.xls")

dp <- readWorksheet(wk2, sheet=2)

dpf$Education <- factor(dpf$Education,
labels=c("Undergrad", "Grad", "Advanced"))

dpf$Personal.Loan <- factor(dpf$PersonalLoan,
labels=c("no", "yes"))
```

Adaptación de datos

```
dpf$Securities.Account <-
factor(dpf$Securities.Account, labels=c("no",
"yes"))

dpf$CD.Account <- factor(dpf$CD.Account,
labels=c("no", "yes"))

dpf$Online <- factor(dpf$Online, labels=c("no",
"yes"))

dpf$CreditCard <- factor(dpf$CreditCard,
labels=c("no", "yes"))
```

Referencias

- M. Kuhn, K. Johnson, "Applied Predictive Modeling", Springer, 2013
- J. Adler, "R in a Nutshell", O'Reilly, 2010
- R.I. Kabacoff, "R in Action", Manning, 2011
- P. Teetor, "R Cookbook", O'Reilly, 2011
- G. James, D. Witten, T. Hastie, R. Tibshirani, "An Introduction to Statistical Learning," Springer, 2013