K-NN para regression (ejemplo de algoritmo no paramétrico), K-FCV Y COMPARACIONES

TRABAJO EN EL LABORATORIO

(Rafael Alcalá)

Bibliography:

PAPER para k-NN: Hechenbichler K. and Schliep K.P. (2004) Weighted k-Nearest-Neighbor Techniques and Ordinal Classification, Discussion Paper 399, SFB 386, Ludwig-Maximilians University Munich (http://www.stat.uni-muenchen.de/sfb386/papers/dsp/paper399.ps)

PAPER para k-NN: Samworth, R.J. (2012) Optimal weighted nearest neighbour classifiers. Annals of Statistics, 40, 2733-2763. (http://www.statslab.cam.ac.uk/~rjs57/Research.html)

LIBRO para k-fcv y lm: Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani
An Introduction to Statistical Learning with Applications in R
Springer, 2013
Chapter 05 (sect 5.1.3) and 03

Outline

- k-NN
  - Uso en problemas de regression (variable continua)
  - Visualización de los resultados y Cálculo del RMSE o MSE
  - Influencia de las variables
- K-fold cross-validation
  - Obtención de las medias de entrenamiento y test
- Cómo afrontar un problema dado
- Comparativa general entre distintos algoritmos
- TAREA: Trabajo final a entregar

K-NN – Uso en problemas de regression

(variable continua)

Seguiremos usando el conjunto Boston del paquete MASS, y usaremos la función kknn de la librería kknn:

```
require(MASS)
attach(Boston)
require(kknn)
```

Instalación:
`install.packages("kknn")`

- Usaremos la función kknn

kknn (formula = formula(train), train, test, k = 7, distance = 2, kernel = "optimal", scale=TRUE)

(Aplica k-NN a un conjunto de test dado un conjunto de entrenamiento)

formula – variables usadas al estilo de *lm*; **train y test** – conjuntos de datos de entrenamiento y test; **k** – número de vecinos; **distance** – distancia de Minkowski, es decir con 1 (Manhattan) y con 2 (euclídea); **scale** – escalado de las variables para tener igual sd (desviación estandar); **kernel** – tipo de kernel usado para el uso de pesos según distancias ("rectangular" es k-NN estandar sin pesos). Tipos existentes: "rectangular", "triangular", "epanechnikov", "biweight", "tri-weight", "cos", "inv", "gaussian", "rank" y "optimal“.

K-NN – Uso

- Obtención del modelo para un conjunto de datos

```
fitknn1 <- kknn(medv ~ ., Boston, Boston)
# Por defecto k = 7, distance = 2, kernel = "optimal“
# y scale=TRUE
```

- Información en el objeto

```
names(fitknn1)
```

```
[1] "fitted.values" "CL"          "W"           "D"           "C“
[6] "prob"          "response"    "distance"    "call"        "terms"
```

En ***fitknn1$fitted.values*** se encuentran los valores obtenidos para los ejemplos indicados como test (los mismos usados para training en este caso)

Visualización de los resultados
Cálculo manual del error

- Visualización

```
plot(medv~lstat)
points(lstat,fitknn1$fitted.values,col="blue",pch=20)
```

- Nuevas predicciones y cálculo manual de la raíz del ECM (RMSE)

```
yprime = fitknn1$fitted.values
# o también yprime=predict(fitknn1,Boston)
sqrt(sum((Boston$medv-yprime)^2)/length(yprime)) #RMSE
```

Que da como resultado: **2.467004**

- El mejor modelo con la RL múltiple por la interacción *lstat*rm* (***fit8***) obtiene RMSE=4.68, mientras que un spline de grado 18 (***fit10***) obtiene RMSE=5.06. No obstante se pierde gran parte de la interpretabilidad que tendría ***fit8***.

Influencia de las variables

- Considerando las interacciones aprendidas con regressión lineal

```
fitknn2 <- kknn(medv ~ lstat*rm+I(lstat^2)+age+crim+dis, Boston, Boston)
yprime = fitknn2$fitted.values; sqrt(sum((Boston$medv-yprime)^2)/length(yprime)) #RMSE
#[1] 2.421144

fitknn3 <- kknn(medv ~ lstat*rm+I(lstat^2)+age+crim+dis+black+nox, Boston, Boston)
yprime = fitknn3$fitted.values; sqrt(sum((Boston$medv-yprime)^2)/length(yprime)) #RMSE
#[1] 2.259452

fitknn4 <- kknn(medv ~ . + lstat*rm+I(lstat^2) - chas, Boston, Boston)
yprime = fitknn4$fitted.values; sqrt(sum((Boston$medv-yprime)^2)/length(yprime)) #RMSE
#[1] 2.029163
```

- En realidad, no tiene por qué coincidir con lo mejor para regresión lineal

```
fitknn5 <- kknn(medv ~ . - chas, Boston, Boston)
yprime = fitknn5$fitted.values; sqrt(sum((Boston$medv-yprime)^2)/length(yprime)) #RMSE
#[1] 2.009211

fitknn6 <- kknn(medv ~ . - chas - ptratio -zn, Boston, Boston)
yprime = fitknn6$fitted.values; sqrt(sum((Boston$medv-yprime)^2)/length(yprime)) #RMSE
#[1] 1.947602

plot(medv~lstat)
points(lstat,fitknn1$fitted.values,col="blue",pch=20)
points(lstat,fitknn5$fitted.values,col="red",pch=20)
points(lstat,fitknn6$fitted.values,col="green",pch=20)
```

K-fold cross-validation – Obtención de las medias del error sobre las mismas particiones

- Es necesario utilizar las mismas particiones para probar distintos algoritmos.
- Lo recomendable es tenerlas preparadas en ficheros para usarlas en tantas pruebas como se quiera
- Usaremos las particiones para California en KEEL:
  - Keel.ugr.es -> Keel-dataset -> Regression datasets (32)
  - Descargar el el fichero 5-fcv del conjunto “California”

K-fold cross-validation – Obtención de las medias del error sobre las mismas particiones

- Como ejemplo usamos el modelo LM con todas las variables (para MSE)

```
nombre <- "california"
run_lm_fold <- function(i, x, tt = "test") {
        file <- paste(x, "-5-", i, "tra.dat", sep="")
        x_tra <- read.csv(file, comment.char="@", header=FALSE)
        file <- paste(x, "-5-", i, "tst.dat", sep="")
        x_tst <- read.csv(file, comment.char="@", header=FALSE)
        In <- length(names(x_tra)) - 1
        names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
        names(x_tra)[In+1] <- "Y"
        names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
        names(x_tst)[In+1] <- "Y"
        if (tt == "train") {
                test <- x_tra
        }
        else {
                test <- x_tst
        }
        fitMulti=lm(Y~.,x_tra)
        yprime=predict(fitMulti,test)
        sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}
lmMSEtrain<-mean(sapply(1:5,run_lm_fold,nombre,"train"))
lmMSEtest<-mean(sapply(1:5,run_lm_fold,nombre,"test"))
```

```
lmMSEtrain = 4830423550
lmMSEtest  = 4848069879
```

K-fold cross-validation – Obtención de las medias del error sobre las mismas particiones

- Con k-NN (para MSE)

```
nombre <- "california"
run_knn_fold <- function(i, x, tt = "test") {
        file <- paste(x, "-5-", i, "tra.dat", sep="")
        x_tra <- read.csv(file, comment.char="@", header=FALSE)
        file <- paste(x, "-5-", i, "tst.dat", sep="")
        x_tst <- read.csv(file, comment.char="@", header=FALSE)
        In <- length(names(x_tra)) - 1
        names(x_tra)[1:In] <- paste ("X", 1:In, sep="")
        names(x_tra)[In+1] <- "Y"
        names(x_tst)[1:In] <- paste ("X", 1:In, sep="")
        names(x_tst)[In+1] <- "Y"
        if (tt == "train") {
                test <- x_tra
        }
        else {
                test <- x_tst
        }
        fitMulti=kknn(Y~.,x_tra,test)
        yprime=fitMulti$fitted.values
        sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}
knnMSEtrain<-mean(sapply(1:5,run_knn_fold,nombre,"train"))
knnMSEtest<-mean(sapply(1:5,run_knn_fold,nombre,"test"))
```

```
lmMSEtrain  = 4830423550
lmMSEtest   = 4848069879
knnMSEtrain = 1562164927
knnMSEtest  = 3847242396
```

(A) Cómo afrontar un problema dado

- **Sobre el conjunto total: (1)** Se analizan los datos para comprenderlos y se plantean los modelos lineales simples a modo de estudio
  - California has 20640 rows and 9 columns (ver en Keel.ugr.es -> Keel-dataset -> Regression datasets (32))
    - **Longitude, Latitude, HousingMedianAge, TotalRooms, TotalBedrooms, Population, Households, MedianIncome, MedianHouseValue**
    - **Objective** - Predict MedianHouseValue from others.
    - **MedianHouseValue** - The dependent variable is **ln(**median house value**)**.

```
California <- read.csv("california.dat", comment.char="@", header=FALSE)
n<-length(names(California)) -1;
names(California)[1:n] <- paste ("X", 1:n, sep="");  names(California)[n+1] <- "Y"
temp <- California
plotY <- function (x,y) {
      plot(temp[,y]~temp[,x], xlab=names(temp)[x], ylab=names(temp)[y])
}
par(mfrow=c(2,4)); x<-sapply(1:(dim(temp)[2]-1), plotY, dim(temp)[2]); par(mfrow=c(1,1))
fitX8 <- lm(Y~X8, California)
fitX7 <- lm(Y~X7, California)
fitX4 <- lm(Y~X4, California)
#.... etc
```

(A) Cómo afrontar un problema dado

- **(2)** Se construye el modelo multivariable con todas y luego intentamos mejorarlo con interacciones, no linealidad, etc. Es un proceso manual y tedioso, pero ayuda a comprender el problema.

```
fit1=lm(Y~X8+I(X8^2)+I(X8^3)+I(log(X3))+I(log(X4/X6))
    +I(log(X5/X6))+I(log(X6/X7))+I(log(X7)),California)
#Modelo que usa el logaritmo por ser Y=ln(median house value)
#recomendado en el website que describe el problema

#Haciendo pruebas y analizando los datos se plantean posibles
#modelos hasta llegar a uno satisfactorio
fit2=lm(Y~., California)
fit3=lm(Y~.+X4*X7*X8, California)
fit4=lm(Y~.+I(X1^2)+I(X6^2)+I(X8^2)+I(X8^3)+I(X8^4)
    +X7*X8*X4*X5*X6, California)

summary(fit1)$adj.r.squared
summary(fit2)$adj.r.squared
summary(fit3)$adj.r.squared
summary(fit4)$adj.r.squared
```

4154461552

(A) Cómo afrontar un problema dado

- **(3)** Se utiliza k-fcv para ver la capacidad de generalización del modelo.
- **(4)** Se prueba con otras técnicas y se **comparan** según el **MSE de test** medio, sacando las conclusiones oportunas que sirvan para escoger el modelo más adecuado atendiendo a los criterios de predición vs inferencia

```
nombre <- "california"
run_lm_fold <- function(i, x, tt = "test") {
      file <- paste(x, "-5-", i, "tra.dat", sep=""); x_tra <- read.csv(file, comment.char="@", header=FALSE)
      file <- paste(x, "-5-", i, "tst.dat", sep=""); x_tst <- read.csv(file, comment.char="@", header=FALSE)
      In <- length(names(x_tra)) - 1
      names(x_tra)[1:In] <- paste ("X", 1:In, sep=""); names(x_tra)[In+1] <- "Y"
      names(x_tst)[1:In] <- paste ("X", 1:In, sep=""); names(x_tst)[In+1] <- "Y"
      if (tt == "train") { test <- x_tra }
      else { test <- x_tst }
      fitMulti=lm(Y~.+I(X1^2)+I(X6^2)+I(X8^2)+I(X8^3)
                  +I(X8^4)+X7*X8*X4*X5*X6,x_tra)
      yprime=predict(fitMulti,test)
      sum(abs(test$Y-yprime)^2)/length(yprime) ##MSE
}
lmMSEtrain<-mean(sapply(1:5,run_lm_fold,nombre,"train"))
lmMSEtest<-mean(sapply(1:5,run_lm_fold,nombre,"test"))

#Se obtendría también para k-NN tal cual se hizo antes
```

```
Antes (todas las variables)
  lmMSEtrain  = 4830423550
  lmMSEtest   = 4848069879
Ahora (interac. + no-lin.)
  lmMSEtrain  = 4154461552
  lmMSEtest   = 5318954701
k-NN (obtenido antes)
  knnMSEtrain = 1562164927
  knnMSEtest  = 3847242396
```

(B) Comparativa general entre distintos algoritmos

- Las comparativas deben realizarse en las mismas condiciones para todos los algoritmos
- Parámetros genéricos en los algoritmos que los tengan
- Nada de consideraciones específicas para cada problema (LM se ejecuta de forma genérica con todas las variables para cada problema – Y ~ .)
- Se obtienen los errores de test medios mediante K-fcv y se comparan mediante tests estadísticos en un buen número de datasets (más de 15… cuantos más mejor)
- Las particiones K-fcv deben de ser las mismas para todos los algoritmos (usar particiones ya disponibles facilita futuras comparaciones)

(B) Comparativa general entre distintos algoritmos

- Dadas las tablas disponibles en csv con resultados de LM, k-NN y el modelo de regresión M5' (obtenidas con las particiones disponibles en KEEL), cargaríamos los datos para aplicar tests estadísticos así:

```
#leemos la tabla con los errores medios de test
resultados <- read.csv("regr_test_alumnos.csv")
tablatst <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatst) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatst) <- resultados[,1]

#leemos la tabla con los errores medios de entrenamiento
resultados <- read.csv("regr_train_alumnos.csv")
tablatra <- cbind(resultados[,2:dim(resultados)[2]])
colnames(tablatra) <- names(resultados)[2:dim(resultados)[2]]
rownames(tablatra) <- resultados[,1]
```

(B) Comparativa general entre distintos algoritmos

- Comparativa por pares de LM y k-NN (Wilcoxon's test)
  - Wilcoxon ordena por las diferencias para los rankings. En clasificación los errores están en el mismo rango. Sin embargo en regresión hay que normalizar el error.
  - Se puede usar la normalización propuesta en

M.J. Gacto, R. Alcalá, F. Herrera, Integration of an Index to Preserve the Semantic Interpretability in the Multi-Objective Evolutionary Rule Selection and Tuning of Linguistic Fuzzy Systems. IEEE Transactions on Fuzzy Systems 18:3 (2010) 515-531

$$\text{DIFF} = \frac{\text{Mean}(\text{Other}) - \text{Mean}(\text{Reference Algorithm})}{\text{Mean}(\text{Other})}$$

  - Donde tomamos como algoritmo de referencia aquel que estemos proponiendo como candidato a ser mejor (en nuestro caso k-NN)
  - Se monta una nueva tabla donde para cada conjunto de datos, si la diferencia es positiva se pone 0 para *Other* y abs(DIFF) para *Ref.Alg.,* y a la inversa si es negativa

(B) Comparativa general entre distintos algoritmos

- Comparativa por pares de LM y KNN (Wilcoxon's test)

```
##lm (other) vs knn (ref)
# + 0.1 porque wilcox R falla para valores == 0 en la tabla

difs <- (tablatst[,1] - tablatst[,2]) / tablatst[,1]
wilc_1_2 <- cbind(ifelse (difs<0, abs(difs)+0.1, 0+0.1), ifelse (difs>0, abs(difs)+0.1, 0+0.1))
colnames(wilc_1_2) <- c(colnames(tablatst)[1], colnames(tablatst)[2])
head(wilc_1_2)
```

- Se aplica el test y se interpretan los resultados

```
LMvsKNNtst <- wilcox.test(wilc_1_2[,1], wilc_1_2[,2], alternative = "two.sided", paired=TRUE)
Rmas <- LMvsKNNtst$statistic
pvalue <- LMvsKNNtst$p.value
LMvsKNNtst <- wilcox.test(wilc_1_2[,2], wilc_1_2[,1], alternative = "two.sided", paired=TRUE)
Rmenos <- LMvsKNNtst$statistic
Rmas
Rmenos
pvalue
```

	R+	R-	p-value
LM(R+) vs KNN(R-)	78	93	0.7660

No existen diferencias significativas entre ambos
Sólo hay un (1-0.7660) *100 = 23.4% de confianza en que sean distintos

(B) Comparativa general entre distintos algoritmos

- Comparativas Múltiples – Usaremos Friedman y como post-hoc Holm (los rankings se calculan por posiciones de los algoritmos para cada problema y **no** hace falta normalización)

```
test_friedman <- friedman.test(as.matrix(tablatst))
test_friedman
```

Friedman chi-squared = 8.4444
p-value = **0.01467**

Existen diferencias significativas al menos entre un par de algoritmos

- Se aplica post-hoc Holm

```
tam <- dim(tablatst)
groups <- rep(1:tam[2], each=tam[1])
pairwise.wilcox.test(as.matrix(tablatst), groups, p.adjust = "holm", paired = TRUE)
```

	1	**2**
2	0.580	-
3	0.081	0.108

Existen diferencias significativas a favor de M5' (**3**vs**1** 0.081 y **3**vs**2** 0.108, con aprox. 90% de confianza) mientras que los otros dos pueden ser considerados equivalentes

- Como información adicional se puede (incluso debe) ver lo que pasa en training (sobreaprendizaje?, etc)

TAREA (para evaluación continua): Reproduzca también éste estudio desde la primera transparencia para el conjunto de datos California y suba el/los fichero/s .R a PRADO en la actividad correspondiente

Keel.ugr.es -> Keel-dataset -> Regression datasets (32)

TRABAJO TEÓRICO-PRÁCTICO: Descripción

- El estudiante debe utilizar el dataset$_R$ asignado para realizar lo siguiente:
  1. Utilizar el algoritmo de regresión lineal simple sobre cada regresor (variable de entrada) para obtener los modelos correspondientes. Si el dataset$_R$ asignado incluye más de 5 regresores, seleccione a su criterio los 5 que considere más relevantes. Una vez obtenidos los modelos, elegir el que considere más adecuado para su conjunto de datos según las medidas de calidad conocidas.

Consiste en utilizar el **conjunto de datos completo** para analizar los datos y realizar un máximo de 5 regresiones sobre las 5 variables más interesantes de su problema tal cual se ha visto en el trabajo de laboratorio. Ver según los indicadores de calidad cual sería la variable más interesante y coméntelo.

Vea la información relativa a su dataset en Keel.ugr.es -> Keel-dataset -> Regression datasets (32) para ello.

TRABAJO TEÓRICO-PRÁCTICO: Descripción

2. Utilizar el algoritmo para regresión lineal múltiple. Justificar adecuadamente si el modelo obtenido aporta mejoras respecto al modelo elegido en el paso anterior.

Consiste en utilizar el **conjunto de datos completo** para aplicar la regresión lineal múltiple sobre todas las variables, y posteriormente considerar posibles interacciones y no linealidad tal cual se ha visto en el laboratorio. Justificar según los indicadores de calidad los pasos dados y sacar conclusiones sobre las relaciones existentes.

3. Aplicar el algoritmo k-NN para regresión no paramétrica.

Consiste en utilizar las **particiones 5-fcv** (ya disponibles para su dataset) para aplicar k-NN y ver si mejora con las mismas particiones al mejor modelo obtenido en el paso anterior con la regresión lineal múltiple tal cual se ha visto para la resolución de un problema concreto en el laboratorio. Comentar los resultados.

TRABAJO TEÓRICO-PRÁCTICO: Descripción

4. Comparar los resultados de los dos algoritmos de regresión múltiple

Consiste en utilizar las **particiones 5-fcv** para su dataset con k-NN (ya lo tiene del punto anterior) y con la regresión lineal múltiple con todas las variables (para comparación general de algoritmos). Reemplazar los resultados de las tablas disponibles en regr_test_alumnos.csv y en regr_train_alumnos.csv por los suyos para el dataset que le toca (en ambos ficheros los valores aparecen redondeados y se pretende que los de su dataset estén sin redondear).

Comparar LM y k-NN por Wilcoxon y los tres algoritmos (incluido el algoritmo M5') por Friedman y Holms. Comentar los resultados.

En todos los casos se deben incluir las instrucciones utilizadas (mejor como apéndice) y los resultados obtenidos incluidos los gráficos utilizados para los distintos análisis