Some notes on the nnc package

A. I. McLeod, M. S. Islam
University of Western Ontario

Abstract

Several matters relating the **nnc** package are discussed. We show that when k, the number of nearest neighbors is even, there is considerable variability in the predictions using `knn` in the package **class**. This variability occurs even with the default setting of the argument `use.all=TRUE`. Cross-validation is discussed as well. The importance of *honest* cross-validation is noted and the delete-d method is recommended. Cross-validation is then used to estimate the prediction errors to compare Fisher linear discriminant classification with kNN for the iris data. We conclude with two graphics that demonstrate why Fisher linear discriminate analysis works so well with the iris data. These graphs should be viewed in color.

Keywords: cluster and multicore computers; cross-validation; delete-d cross-validation; honest cross-validation; k-nearest neighbor classification; repeated-holdout cross validation; Rmpi package.

1. The problem of ties

Holmes and Adams (2003) reported that misclassification rate for Ripley's synthetic test data was 0.82 using the optimal maximum pseudolikelihood estimate for k, $\hat{k} = 66$. The code below verifies that this estimate agrees with our software,

```
R> library(nnc)
R> library(MASS)
R> X <- synth.tr[, 1:2]
R> y <- factor(synth.tr[, 3])
R> KMAX <- 100
R> d <- numeric(KMAX)
R> for (k in 1:KMAX) {
+     z <- nnc(k = k, X = X, Y = y)
+     d[k] <- deviance(glm.fit(x = z, y = y, family = binomial(link = "logit")))
+ }
R> khat <- which.min(d)
R> plot(d, xlab = "k", ylab = "deviance")
R> points(khat, d[khat], col = "red", pch = 16, cex = 1.4)
R> title(sub = bquote(hat(k) == .(khat)))
```

In practice even values of k are often avoided since there is some arbitrariness due to the

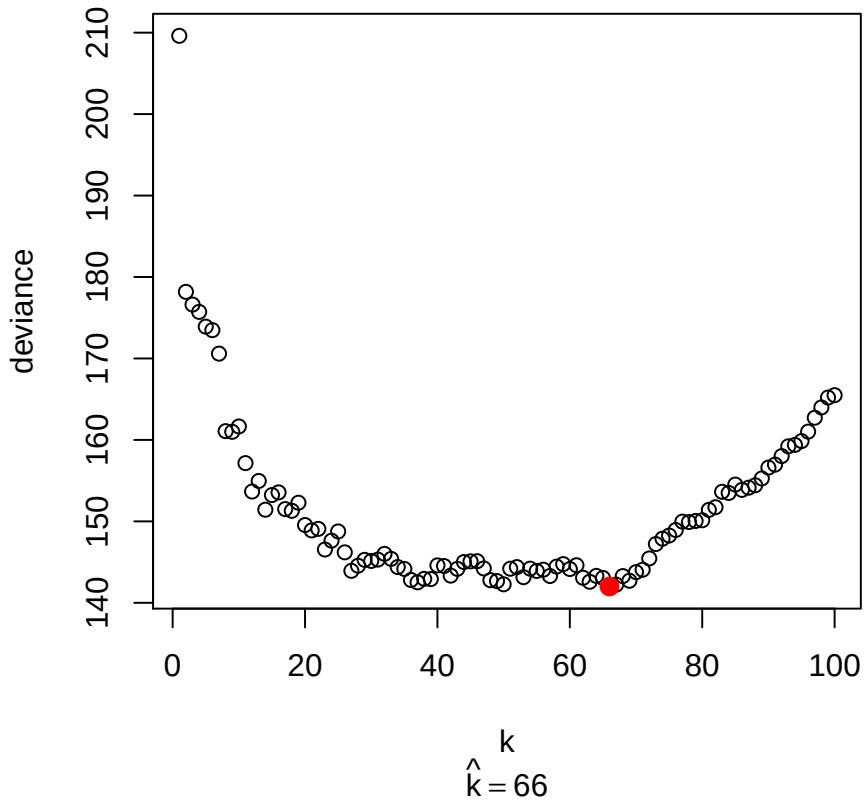

Figure 1: Optimal k for kNN for Ripley's synthetic data

likelihood of ties values. In the `knn` function, ties are still a problem even when the default option, `use.all=TRUE`, is used.

```
R> XTest <- synth.te[, 1:2]
R> yTest <- factor(synth.te[, 3])
R> REP <- 100
R> K <- 66
R> e <- numeric(REP)
R> for (i in 1:REP) {
+     yhat <- knn(train = X, test = XTest, cl = y, k = K, use.all = TRUE)
+     e[i] <- mean(yhat != yTest)
+ }
R> hist(e, main = "Distribution of error rate", xlab = "Error rate")
```

As can be seen a misclassification rate of 0.82 is on the low side of what can be expected when $k = 66$ is used. Trying $k = 67$, we find, as expected, the misclassification rate remains constant.

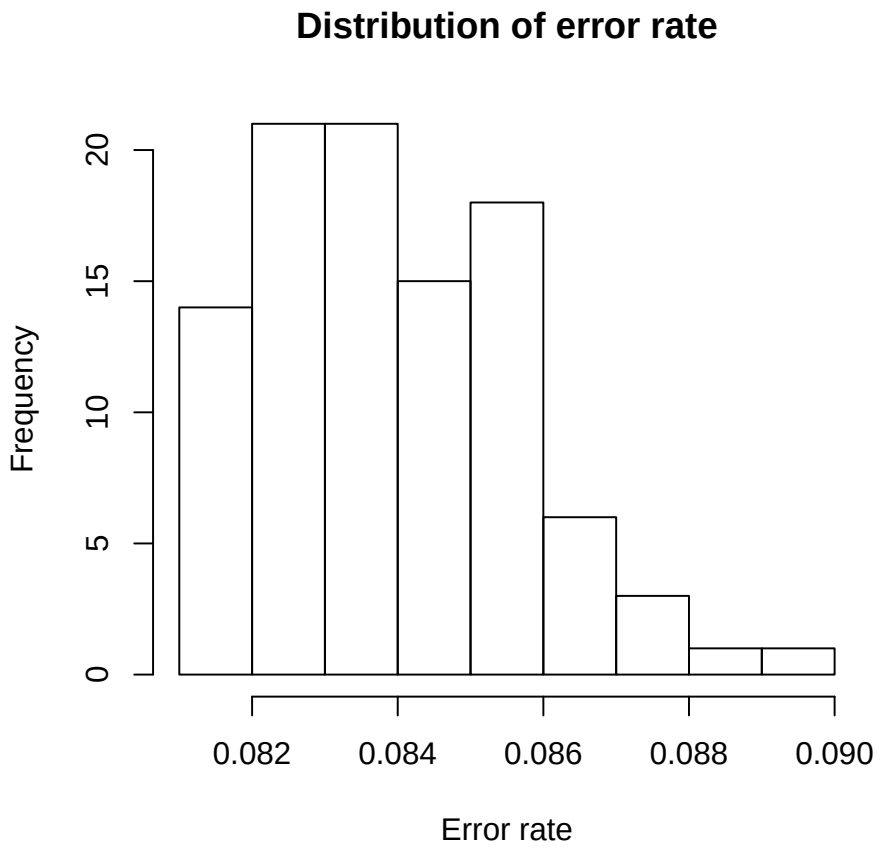

Figure 2: Histogram of misclassification rates for the test data when $k = 66$ is used for the training data.

```
R> K <- 67
R> for (i in 1:REP) {
+     yhat <- knn(train = X, test = XTest, cl = y, k = 67)
+     e[i] <- mean(yhat != yTest)
+ }
R> summary(e)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  0.085   0.085   0.085   0.085   0.085   0.085
```

2. Fisher's iris data

We use the built-in R dataset `iris` to illustrate how our algorithms work for the case when there are $Q = 3$ classes. With the iris data there are 3 classes, according to iris species, setosa,

versicolor, and virginica. The inputs are measurements of sepal length and width and petal length and width, respectively. There are 50 observations on each species.

2.1. Optimal k

Here we find the optimal k,

```
R> khat(iris[, 1:4], iris[, 5])
```

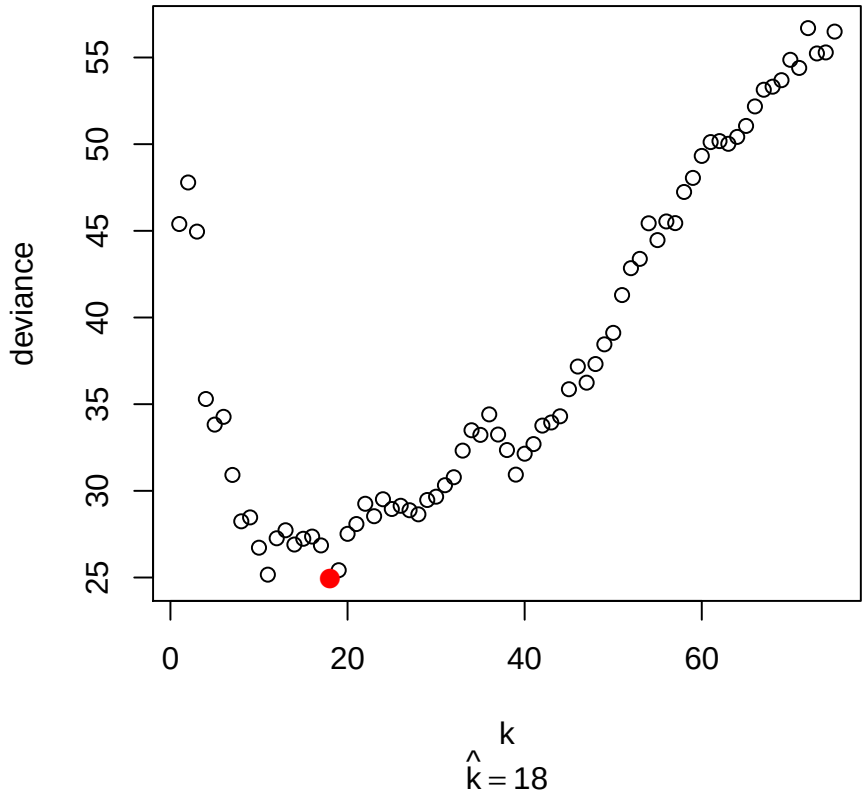

Figure 3: Profile deviance plot for estimate k with iris data

2.2. Iris data visualization

Wikipedia has a nice article about this dataset with pictures of the three iris species: setosa, virginica and versicolor.

See http://en.wikipedia.org/wiki/Iris_flower_data_set

The scatterplot in the figure below, obtained from the Wikipedia article cited above, shows how well separated the three species are.

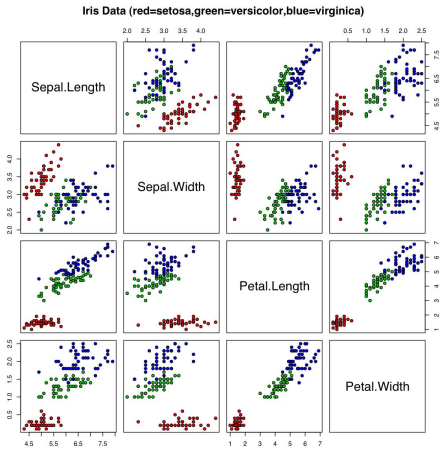

Figure 4: Scatterplot matrix for iris data

Fisher (1936)[1] used this iris data[2] to illustrate the technique we know today as Fisher linear discriminant analysis. It is no surprise that the method is probably the best method for this data.

Visualization of the data with suitable graphics reveals that the species are well separated with in fact no overlap between setosa and the other species.

The lattice graphic strongly suggests linear decision boundaries and so it is not surprising the Fisher linear discriminate classification works well with this data. It is a little suprising that kNN works as well as it does!

The parallel coordinates plot below also shows how well separated the three classes are and that there is no suggestion of a nonlinear decision boundary.

```
R> library(lattice)
R> parallel(~iris[, 1:4], groups = iris[, 5], auto.key = list(space = "top",
+     columns = 3))
```

2.3. Cross-validation comparison with FLDA

Hastie, Tibshirani, and Friedman (2009, §7.10.2 and 7.10.3) have pointed out that frequently cross-validation methods have been misapplied when a parameter is not re-estimated each time on the training data. The term *honest* cross-validation may be used to emphasize that this is done. The consequence of not doing so is that a much more optimistic result is obtained. The true test error is likely to be much larger.

Hastie *et al.* (2009, §7.10) recommend k-fold cross-validation using the one-standard-deviation rule. An important drawback to k-fold cross-validation is that the results are quite variable since there is not really enough replication. Kim (2009) recommended that the method given in Hastie *et al.* (2009) not be used for classification problems and instead recommended

[1] http://digital.library.adelaide.edu.au/coll/special//fisher/138.pdf

[2] There was a typo in the data listed in Fisher (1936). See http://archive.ics.uci.edu/ml/datasets/Iris

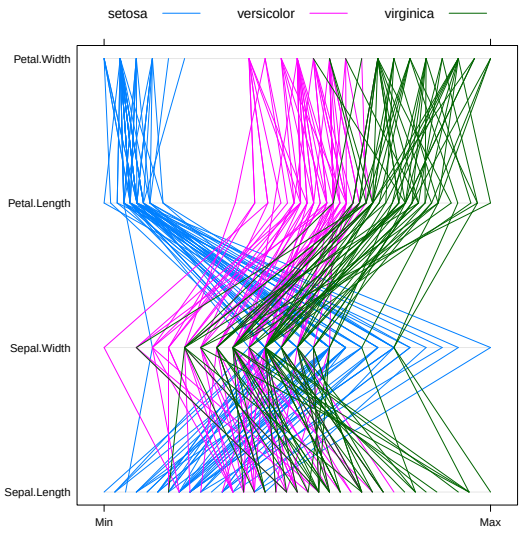

Figure 5: Scatterplot matrix for iris data

repeated-holdout cross validation. This is essentially equivalent to the delete-d method recommended by Shao (1997) for linear model selection. With this algorithm, a randomly sampling algorithm is used to select the test data and the validation data is then formed by removing this test data. Many replications are required, at least 1000. This cross-validation method is also very easy to implement using an R package for cluster or parallel computing.

The R function, `OneItn`, shown below computes a single-cross-validation estimate. In the code we use the parameter `fold` to specify that the fraction of data being used for the test sample will be $1/fold$. This function can then be independently evaluated on each thread or node in a multicore or cluster computer.

With the iris data, it is more efficient to use stratified sampling, taking an equal number from each class for the training samples. This makes the code below slightly more complicated and lengthy.

```
R> OneItn <- function() {
+     Y <- iris[, 5]
+     X <- iris[, 1:4]
+     Xset <- iris[iris$Species == "setosa", 1:4]
+     Xver <- iris[iris$Species == "versicolor", 1:4]
+     Xvir <- iris[iris$Species == "virginica", 1:4]
+     Yset <- iris[iris$Species == "setosa", 5]
+     Yver <- iris[iris$Species == "versicolor", 5]
+     Yvir <- iris[iris$Species == "virginica", 5]
+     n <- nrow(Xset)
+     fold <- 5
+     nTest <- floor(n/fold)
+     NREP <- 100
+     kmax <- 80
```

```
+     jset <- sample(1:n, size = nTest)
+     jver <- sample(1:n, size = nTest)
+     jvir <- sample(1:n, size = nTest)
+     indsetTest <- rep(FALSE, n)
+     indsetTest[jset] <- TRUE
+     indverTest <- rep(FALSE, n)
+     indverTest[jver] <- TRUE
+     indvirTest <- rep(FALSE, n)
+     indvirTest[jvir] <- TRUE
+     YsetTrain <- Yset[!indsetTest]
+     YsetTest <- Yset[indsetTest]
+     XsetTrain <- Xset[!indsetTest, ]
+     XsetTest <- Xset[indsetTest, ]
+     YverTrain <- Yver[!indverTest]
+     YverTest <- Yver[indverTest]
+     XverTrain <- Xver[!indverTest, ]
+     XverTest <- Xver[indverTest, ]
+     YvirTrain <- Yvir[!indvirTest]
+     YvirTest <- Yvir[indvirTest]
+     XvirTrain <- Xvir[!indvirTest, ]
+     XvirTest <- Xvir[indvirTest, ]
+     XTrain <- rbind(XsetTrain, XverTrain, XvirTrain)
+     YTrain <- c(YsetTrain, YverTrain, YvirTrain)
+     XTest <- rbind(XsetTest, XverTest, XvirTest)
+     YTest <- c(YsetTest, YverTest, YvirTest)
+     train.LDA <- lda(XTrain, YTrain)
+     yfitLDA <- predict(train.LDA, newdata = XTest, dimen = 1)$class
+     etaFLDA <- mean(as.numeric(yfitLDA != YTest))
+     kopt <- khat(XTrain, YTrain, kmax, plot = FALSE)
+     ans <- knn(train = XTrain, test = XTest, k = kopt, cl = YTrain)
+     etaKNN <- mean(as.numeric(ans != YTest))
+     eta <- c(etaFLDA, etaKNN)
+     names(eta) <- c("LDA", "kNN")
+     eta
+ }
```

Using the script below we evaluate `OneItn` five times to estimate the time required.

```
R> library(nnc)
R> StartTime <- proc.time()[1]
R> NREP <- 5
R> eta <- matrix(numeric(NREP * 2), ncol = 2)
R> for (i in 1:NREP) eta[i, ] <- OneItn()
R> EndTime <- proc.time()[1]
R> TotTime <- EndTime - StartTime
R> e <- apply(eta, MARGIN = 2, FUN = mean)
R> e
```

```
[1] 0.02666667 0.21333333

R> TotTime

user.self
      8.4
```

The script used for running using **Rmpi** is given below.

```
##Source: nncRmpi.R
#Delete d-cross validation to compare FLDA and kNN for iris data.
#Note the function OneItn() is defined in the workspace and is also given
#  in the vignette for the 'nnc' package.
#TO RUN THIS SCRIPT
#LOAD 'nnc/Rmpi/.Rdata' or load script for 'OneItn'
#Timing: NumRep<-10^4 takes about 27 minutes with 8 threads
#
library(Rmpi)
Start <- proc.time()[3]
StartDate <- date()
#start slave nodes
mpi.spawn.Rslaves(nslaves=8)
#compare FLDA and kNN using cross-validation
mpi.bcast.cmd(library(nnc))
mpi.bcast.cmd(library(MASS))
NumRep <- 10^4
ISEED <- 19100437
#setup parallel RNG. seed can be specified.
mpi.setup.rngstream(ISEED)
#export function
mpi.bcast.Robj2slave(OneItn)
#Use parallel replication
ans<-mpi.iparReplicate(n=NumRep, expr=OneItn())
End <- proc.time()[3]
EndDate<-date()
TotalTime <- End-Start
write(TotalTime, file="TotalTime.txt")
write(StartDate, file="TotalTime.txt", append=TRUE)
write(EndDate, file="TotalTime.txt", append=TRUE)
ans<-t(ans) #so columns are FLDA, kNN
#save data for analysis later
save(ans, file="ans.Rdata")
#print out
cat("Total time in seconds =", TotalTime, fill=TRUE)
cat("Total time in minutes =", TotalTime/60, fill=TRUE)
apply(ans, 2, mean)
apply(ans, 2, median)
```

```
apply(ans, 2, function(x), mean(x, trim=0.1))
boxplot(ans)
boxplot(jitter(ans))
tb<-apply(ans, 2, table)
#frequency plot
graphics.off()
library(lattice)
tbLDA<-as.data.frame(tb$LDA)
tbLDA$Var1 <- round(as.numeric(as.character(tbLDA$Var1)), 3)
xyplot(Freq/NumRep ~ Var1, data=tbLDA, type="h", lwd=4, col="red", xlab="value", ylab="fre
#
tbkNN<-as.data.frame(tb$kNN)
tbkNN$Var1 <- round(as.numeric(as.character(tbkNN$Var1)), 3)
xyplot(Freq/NumRep ~ Var1, data=tbkNN, type="h", lwd=4, col="red", xlab="value", ylab="fre
#close and quit
mpi.close.Rslaves()
mpi.quit()
```

We ran the R script on a Mac Pro with 8 virtual cpu's with `NumRep<-10000` and it took about 27 minutes. The mis-classification rates are summarized in the table below.

	LDA	kNN
mean	0.0198	0.1809
sd	0.0224	0.0845
median	0.0000	0.1667
10 % trim mean	0.0164	0.1778

Table 1: Summary of 10000 cross-validation replications

As expected FLDA performed much better than kNN for this data. This is confirmed in the boxplot shown below.

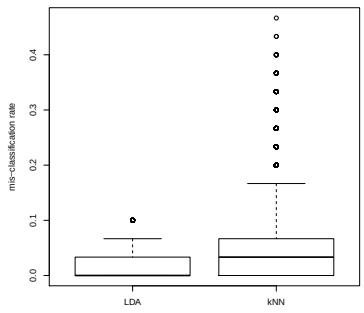

Figure 6: Boxplot of the mis-classification rates in 10,000 replications for FDLA and kNN.

There is extreme discretness in the cross-validated mis-classification rates. The lattice style line plots show that in fact the FDLA test errors take on only 4 distinct values in all 10,000

replications.

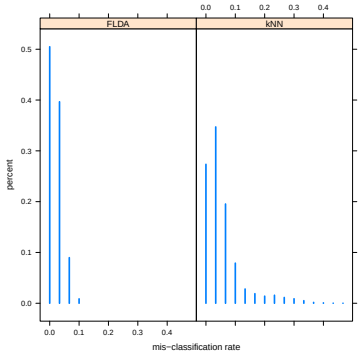

Figure 7: Lattice plot showing the distribution of mis-classification in 10,000 replications for FDLA and kNN.

References

Fisher RA (1936). "The Use of Multiple Measurements in Taxonomic Problems." *Annals of Eugenics*, **7**, 179–188.

Hastie T, Tibshirani R, Friedman JH (2009). *The Elements of Statistical Learning: Data Mining, Inference and Prediction*. Springer-Verlag, New York, 2nd edition.

Holmes CC, Adams NM (2003). "Likelihood Inference in Nearest-Neighbour Classification Models." *Biometrika*, **90**(1), 99–112.

Kim JH (2009). "Estimating Classification Error Rate: Repeated Cross-validation, Repeated Hold-out and Bootstrap." *Computational Statistics and Data Analysis*, **53**, 3735–3745.

Shao J (1997). "An Asymptotic Theory for Linear Model Selection." *Statistica Sinica*, **7**, 221–264.

Affiliation:

A. Ian McLeod
Department of Statistical and Actuarial Sciences
University of Western Ontario
E-mail: aim@stats.uwo.ca
URL: http://www.stats.uwo.ca/faculty/aim