

Advanced Microarray Data Analysis

Practicals

Vlad Popovici
vlad.popovici@isb-sib.ch

Swiss Institute of Bioinformatics

November 4th, 2009



Bibliography:

Venables, Ripley, *Modern Applied Statistics with S*

```
load("data.rdata")
```

```
ls()  
dim(X)  
dim(Xvd)
```

```
sum(rownames(X) == names(y.erpos))
```

- load data and check the variables
- check if the labels are in the same order as the samples
- check the label values
- endpoints: ER status, pCR
- 2 data sets: modeling and validation

Have a look at `kcv.r` file.

```
cv = kcv.splits(c(1,2,1,2,1,2,1,2,1,2), k=5, stratified=TRUE)
cv$cv.train
cv$cv.valid
```

What is `stratified` parameter for?

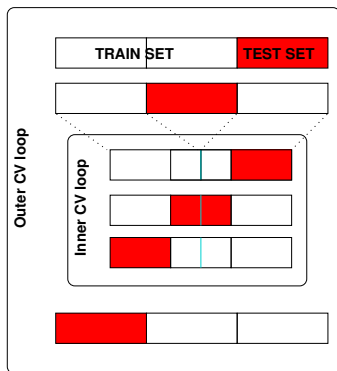
- have a look at `perf.r` file
- how is the AUC computed? what are the values needed?

Have a look at `clsf-DLDA.r` file.

- the main CV loop is in the `cvestim.dlda()` function
- what are its parameters? its output?
- what is `nf = tune.dlda(Xtr, ytr, ntop, kfold=5, rep=5)` doing?
- what is `idx = fr.wilcoxon(Xtr, ytr)` doing?

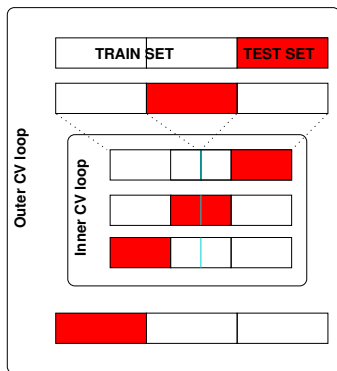
Optimizing meta-parameters

Two-external (nested) CV:



Optimizing meta-parameters

Two-external (nested) CV:



Identify the structure above in the `clsf-DLDA.r` file.


```

tune.dlda = function(X, y, ntop=2:nrow(X), kfold=5, rep=10)
{
...
    for (k in 1:kfold) {
        itr = cv$cv.train[[k]]          # index of elements for training
        ivd = cv$cv.valid[[k]]          # index of elements for validation
        Xtr = X[,itr]; ytr = y[itr]
        Xvd = X[,ivd]; yvd = y[ivd]

        # Rank the features
        idx = fr.wilcoxon(Xtr, ytr)

...
        for (nf in ntop) {                # for each number of feature, estimate the AUC
            Xtr1 = Xtr[idx[1:nf],]; Xvd1 = Xvd[idx[1:nf],]
...
        }
    }
}

return (ntop[which.max(A)])
}

cvestim.dlda = function(X, y, ntop=2:nrow(X), kfold=5, rep=10)
{
...
    for (k in 1:kfold) {
...
        nf = tune.dlda(Xtr, ytr, ntop, kfold=5, rep=5)
...
        # Rank the features
        idx = fr.wilcoxon(Xtr, ytr)
        Xtr = Xtr[idx[1:nf],]
        Xvd = Xvd[idx[1:nf],]
...
    }
}
}

```

Putting it all together...

WARNING: feature selection must be done inside CV!

```
source("clsf-DLDA.r")  
s = apply(X, 1, sd, na.rm=TRUE)  
i = which(s> 1.25)           # normally you would use all features !!!  
r = cvestim.dlda(X[i,], y.erpos, ntop=c(2,5,7), rep=3)
```

- have a look at `clsf-LREG.r`.
- which function is used for building the models?
- which package is needed? read the manual page for the new function
- replace the current feature ranking function `fr.wilcoxon` with a new one based on fold change
- estimate the performance on pCR endpoint

```
while ( i <= iterations ) {  
  w = exp(-ytr*f)  
  w = w/sum(w)  
  
  stump = rpart(y ~. , data.frame(Xtr , y=ytr) , w, method="class",  
    control=rpart.control( minsplitlevel=0,minbucket=0,cp=-1,maxcompete=0,  
      maxsurrogate=0, usesurrogate=0, xval=0, maxdepth=1))  
  ...  
  gmin.tr = 2 * (p1 < p2) - 1  
  ...  
  gmin.ts = 2 * (p1 < p2) - 1  
  e = sum( w*(1 - (ytr == gmin.tr)) )  
  alpha = 0.5*log ( (1-e) / e )  
  f      = f + alpha*gmin.tr  
  f.ts   = f.ts + alpha*gmin.ts  
  tr.err[i] = sum(f*ytr < 0) / n.tr  
  ts.err[i] = sum(f.ts*yts < 0) / n.ts  
  i = i + 1  
}
```

```
source("adaboost.r")
r = simple.adaboost(t(X[i,]), 2*y.pcr-1,
  t(Xvd[i,]), 2*yvd.pcr-1, iterations=200)

plot(r[[1]], type="l", ylim=c(0,0.4), xlab="Iterations",
  ylab="Error", lwd=2)
lines(r[[2]], col="red", lwd=2)
legend("topright", legend=c("Train data", "Validation data"),
  col=c("black", "red"), lwd=2)
```

