Practicals Bioinformatics 2011-2012

Olivier Stern [olivier.stern@ulg.ac.be](mailto:olivier.stern@ulg.ac.be)

Tom Cattaert [tom.cattaert@ulg.ac.be](mailto:tom.cattaert@ulg.ac.be)

22 November 2011: Trees and Forests

Classification and Regression Trees (CART)

* Breiman *et al.* 1993
* Trait **y** = (y1,…,yn) for i=1,…,n where n in number of individuals
* Predictor variables **x**1,…,**x**pwith **x**j = (xj1,…,xjn)
* Relationship between **x**1,…,**x**pand **y** ?
* Suppose for now that all predictors are binary
* Find variable **x**(1)that is most predictive of trait **y**
* Split sample Ω into two groups Ω1 and Ω2 based on x(1)i = 1 or x(1)i = 0
* Find variables **x**(2) and **x**(3) that are most predictive within Ω1 and Ω2
* Further split into Ω1,1, Ω1,2, Ω2,1 and Ω1,2 based on these
* Apply splitting recursively until stopping rule is met
* Procedure gives rise to tree structure

Node impurity: introduction

* Define measure of node impurity I(Ω) of node Ω
* Choose splitting variable as the one that maximizes the reduction in node impurity Φ = I(Ω) - I(ΩL) - I(ΩR) with ΩL and ΩR left and right daughter nodes
* For binary traits, we define I(Ω) = π(Ω)i(Ω) with π(Ω) probability of belonging to node Ω
* Also i(Ω) is commonly referred to as node impurity
* Then Φ = [i(Ω) – πLi(ΩL) - πRi(ΩR)] π(Ω) with conditional probabilities πL/R = p(ΩL/R|Ω) of belonging to ΩL/R given that on belongs to Ω
* Define φ = i(Ω) – πLi(ΩL) - πRi(ΩR) then Φ = φ π(Ω) and maximizing Φ and φ are equivalent

Node impurity: overview

* Binary traits
* Bayes error
* Gini index
* Entropy
* Continuous traits
* Mean square error

Node impurity: Bayes error

* The Bayes error is given by i(Ω) = min(pΩ, 1-pΩ) with pΩ = p(y|Ω) the probability to be a case given that one belongs to Ω
* Example data:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | x1 = 1 | x1 = 0 | x2 = 1 | x2 = 0 | x3 = 1 | x3 = 0 | Total |
| Cases | 60 | 40 | 80 | 20 | 50 | 50 | 100 |
| Controls | 40 | 60 | 20 | 80 | 50 | 50 | 100 |
| Total | 100 | 100 | 100 | 100 | 100 | 100 | 200 |

* We then find φ1 = 0.5 - 0.5\*0.4 - 0.5\*0.4 = 0.1, φ2 = 0.3 and φ3 = 0
* The variable that maximizes the reduction in impurity φ is **x**2
* Practical drawbacks: commonly there is no single best split from a given node; does not favor trees with further growth potential

Node impurity: Gini index

* The Gini index is given by i(Ω) = 2pΩ(1-pΩ), which is twice the variance of Bernouilli random variable with probability pΩ
* For our example data we find φ1 = 2\*O.5\*0.5 - 0.5\*2\*0.4\*0.6 - 0.5\*2\*0.4\*0.6 = 0.02, φ2 = 0.18 and φ3 = 0
* Again, we would begin splitting our data into two groups based on **x**2; this process is then repeated for each of the daughter nodes
* Practical drawback: tends to favor splits that result in two nodes with highly unbalanced sample sizes
* More generally, for a categorical trait y taking on values 1,…,m, the Gini index is i(Ω) = , with p(r|Ω) the probability of being in class r given that one belongs to node Ω

Node impurity: entropy

* The entropy is given by i(Ω) = - pΩlog(pΩ) - (1-pΩ)log(1-pΩ), which is the deviance of a Bernouilli random variable with probability pΩ
* For our example data we find φ1 = - 0.5\*log(0.5) - 0.5\*log(0.5) + 0.4\*log(0.4) + 0.6\*log(0.6) = 0.020136, φ2 = 0.192745 and φ3 = 0
* In practice, the entropy and the Gini index tend to result in very similar trees for binary traits

Introducing Virco data

* The Virco dataset contains protease sequence information on 1066 HIV viral isolates and corresponding fold-resistance measures for 8 protease inhibitors
* Load the data by

> virco <- read.csv(file='Virco\_data.csv', header=TRUE, sep=',')

> attach(virco)

* Fold resistance to drug IDV is given by variable IDV.Fold

> IDV.Fold[1:10]

[1] 14.2 13.5 16.7 3.0 7.0 21.0 8.0 100.0 18.0 15.0

* A higher value indicates the isolate is more resistant to IDV

Introducing Virco data

* Genotype information corresponding to amino acid position 71 within the protease region of the viral sequence is given in variable P71

> P71[1:10]

[1] - V V T - V - V - V

Levels: - AT AV F G I L T TA TI V VA VAIT VATI VI VL X

* A value ‘–‘ indicates presence of the population consensus amino acid, while a letter indicates a mutation in the form of the amino acid corresponding to this letter

Introducing Virco data

* We want genotype information in the form of binary variables indicating presence or not of a mutation at each of the 99 sites of the protease region

> VircoGenoBin <- data.frame(virco[,substr(names(virco),1,1)=='P']!='-')

> VircoGenoBin [1:10,71]

[1] FALSE TRUE TRUE TRUE FALSE TRUE FALSE TRUE FALSE TRUE

* We want to study the binary trait ‘fold resistance to IDV bigger than that to NFV’

> Trait[1:10]

[1] TRUE FALSE FALSE <NA> <NA> <NA> <NA> <NA> <NA> <NA>

Levels: FALSE TRUE

Classification tree for Virco data

* Classification trees can be obtained in R using the rpart package obtainable from CRAN

> install.packages('rpart')

> library(rpart)

> ClassTree <- rpart(Trait~., method='class', data=VircoGenoBin)

* We specify method=’class’ although this is the default when the trait is a factor variable

Classification tree for Virco data

* Textual output can be generated

> ClassTree

n=976 (90 observations deleted due to missingness)

node), split, n, loss, yval, (yprob)

\* denotes terminal node

1) root 976 399 FALSE (0.5911885 0.4088115)

2) P54< 0.5 480 130 FALSE (0.7291667 0.2708333)

4) P76< 0.5 466 116 FALSE (0.7510730 0.2489270) \*

5) P76>=0.5 14 0 TRUE (0.0000000 1.0000000) \*

3) P54>=0.5 496 227 TRUE (0.4576613 0.5423387)

…

Classification tree for Virco data

* Looking in detail at node 2

2) P54< 0.5 480 130 FALSE (0.7291667 0.2708333)

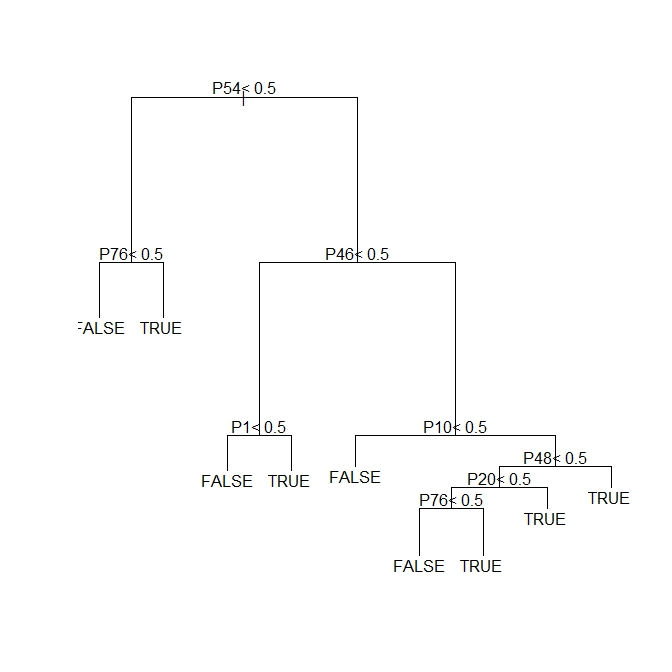
* Viral isolates that are wildtype at site P54 (i.e. P54=0) are assigned to this node, and this feature is inherited by daughter nodes
* There are 480 individuals in this node
* The majority value for the trait in this group is FALSE
* If adopting this majority value as predicted value, 130 individuals are classified incorrectly
* The proportion of sequences for which the trait is FALSE is 0.73, whereas the proportion for which it is TRUE is 0.27
* This is not a terminal node (no \*)

Classification tree for Virco data

* Visual output can also be obtained

> plot(ClassTree)

> text(ClassTree)



Classification tree for Virco data

* The default Gini index criterion can be changed into entropy by the split=’information’ parameter

> rpart(Trait~., method='class', parms=list(split='information'), data=VircoGenoBin)

* The first splits are identical, but further down there are differences
* However, we have not assessed whether the full tree is statistically significant or rather a chance finding
* Therefore, a pruning strategy will be typically adopted
* The difference between Gini index and entropy criteria might then become irrelevant

Classification tree for Virco data

* The control parameter minsplit allows to specify the minimum number of individuals in a node for considering additional splits
* The control parameter minbucket indicates the minimum number of individuals in a terminal node

> rpart(Trait~., method='class', parms=list(split='gini'), control=rpart.control(minsplit=150, minbucket=50), data=VircoGenoBin)

* Many of the original splits are not taking place now because one of the resulting daughter nodes would contain less than 50 individuals

Node impurity: mean square error

* For quantitative traits, the most common measure of node impurity is the mean square error (MSE), i.e. the average sum of squared deviations from the mean
* More concretely the MSE is given by I(Ω) = , where is the mean trait
* This corresponds to the least squares criterion in linear regression
* The aim is again to maximize the reduction in node impurity given by Φ = I(Ω) - I(ΩL) - I(ΩR) where ΩL and ΩR are the left and right daughter nodes

Regression tree for Virco data

* Define a quantitative trait as the difference in fold resistance between NFV and IDV

> Trait <- NFV.Fold-IDV.Fold

* Regression trees can also be obtained from the rpart package

> RegTree <- rpart(Trait~., method='anova', data=VircoGenoBin)

* We specified method=’anova’ although it is the default for a trait variable of type numeric

Regression tree for Virco data

* Textual output

> RegTree

n=976 (90 observations deleted due to missingness)

node), split, n, deviance, yval

\* denotes terminal node

1) root 976 6437933.00 4.288320

2) P54>=0.5 496 1247111.00 -3.916935

4) P46>=0.5 338 343395.20 -10.567160 \*

5) P46< 0.5 158 856789.90 10.309490

10) P58< 0.5 144 110944.10 2.570139 \*

11) P58>=0.5 14 648503.60 89.914290 \*

3) P54< 0.5 480 5122921.00 12.767080

…

Regression tree for Virco data

* Looking in detail at node 2

2) P54>=0.5 496 1247111.00 -3.916935

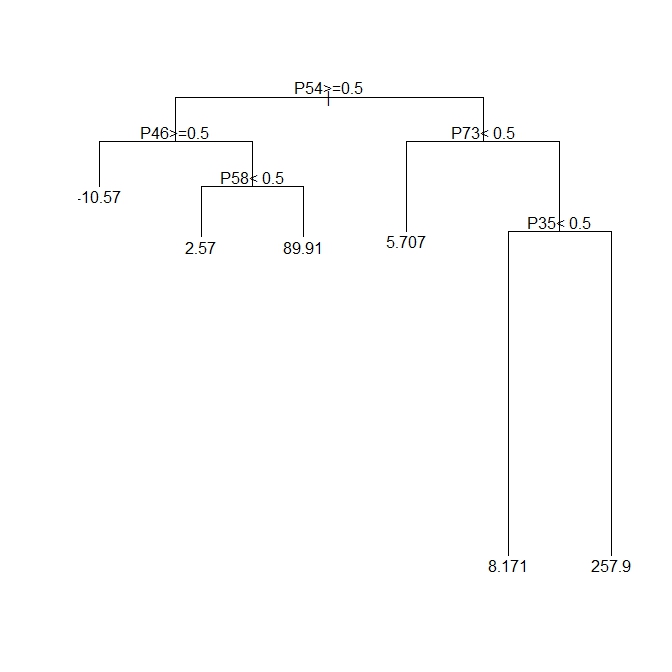
* Viral isolates that have a variant at site P54 (i.e. P54=1) are assigned to this node, and this feature is inherited by daughter nodes
* There are 496 individuals in this node
* The mean trait value in this node is -3.916935
* The deviance, defined as the sum of squared differences between observed and predicted trait, is 1247111.00
* From this we can obtain the MSE as 1247111.00/496=2514.34
* This is not a terminal node (no \*)

Regression tree for Virco data

* Visual output

> plot(RegTree)

> text(RegTree)



Other types of predictors

* For nominal categorical predictors, taking on the values 1,…,m, there are m(m-1)/2 possible ways of splitting based on x
* CART considers all these possible splits for all possible predictors
* For ordinal categorical predictors, which have an ordering from low to high, only m-1 splits are considered
* The decision to include a variable as ordinal or nominal depends on prior knowledge, e.g. for SNPs there may be prior knowledge suggesting that heterozygous individuals cannot be more extreme than variant homozygous individuals w.r.t. influencing the trait
* For continuous predictors, they are first ordered and then the approach for ordinal categorical predictors is followed

Other types of predictors

* Define a genotype data frame containing the full nominal categorical information of sites in the protease region

> VircoGenoCat <-data.frame(virco[,substr(names(virco),1,1)=='P'])

> VircoGenoCat[1:10,71]

[1] - V V T - V - V - V

Levels: - AT AV F G I L T TA TI V VA VAIT VATI VI VL X

* On can turn the nominal into an ordinal variables

> VircoGenoOrd <- sapply(VircoGenoCat, as.numeric)

> VircoGenoOrd[1:10,71]

[1] 1 11 11 8 1 11 1 11 1 11

* The ordering is alphabetical, but is not really meaningful here

Other types of predictors

* Obtain regression tree for APV.Fold using categorical predictors

> rpart(APV.Fold~., method='anova', data=VircoGenoCat)

> Tree

n=939 (127 observations deleted due to missingness)

node), split, n, deviance, yval

\* denotes terminal node

1) root 939 356632.300 12.946540

2) P54=-,A,L,MI,S,T,TI,TS,V,VA,VI,VIM,VL,X 889 237601.200 10.726550

4) P46=-,ILM,IM,LM,LMI,MI,MIL,ML,V,VIM,X 481 44960.940 4.506653

8) P54=-,T,TI,TS,VI,X 342 4475.893 1.980702 \*

9) P54=A,L,MI,S,V,VA 139 32934.020 10.721580

18) P89=-,M,ML 132 24074.510 9.125000

…

Strategies for dealing with covariates

* Ignore covariates in analysis
* Include covariates as potential predictors
* Stratify analysis based on levels of covariates
* Residualize the trait based on prior model fitting of the trait with the covariates as predictors; for linear regression residuals are defined by **r** = – **y**, where are the fitted values
* Which approach is most appropriate varies from case to case and the choice may depend on prior knowledge

Tree impurity

* Each node has an associated error r()
* For categorical traits, this is the misclassification rate R() = 1 – p(s|), with s the majority vote category within , whereas for quantitative traits, it is the MSE R() =
* The tree impurity associated with a tree T is given by

R(T) = , where is the set of all terminal nodes in T, and provides a measure of how well T predicts the observed data

* The resubstitution estimate of this quantity is obtained by resubstituting the orginal data in the tree and calculating the corresponding quantities

Overfitting problem

* Inclusion of additional splits will always improve this resubstitution estimate of the tree impurity
* However, we want to avoid chance findings specific to the data sample under consideration, similar to overfitting in regression
* Therefore, we want to determine the optimal subtree, through pruning our original tree to a portion of it that best fits the general population of interest
* For this, we need an honest estimate of the tree impurity that is applicable to any sample from the population of interest, which can be obtained through ten-fold cross-validation
* Random forests are a different solution to the overfitting problem

Ten-fold cross-validation

* Divide the individuals in the sample L into ten approximately equal parts Li for i=1,…,10
* For each i construct tree T-ibased on the learning sample L-i = L \ Li
* For each i obtain the error Rts(T-i) from running the test sample Li through the tree T-i
* The cross-validation estimate of the tree impurity is then given by RCV(T) = (T-i)
* This is an honest estimate of the tree impurity associated with the tree T constructed based on the entire sample L

Cost-complexity pruning

* Identifying the right-sized tree should take into account both tree impurity R(T) and tree size |T|, i.e. the number of terminal nodes Therefore, define cost-complexity as Rα(T) = R(T) + α|T| with complexity parameter α0
* In fact we add a penalty for the number of nodes in the tree, since additional nodes make the tree more difficult to interpret
* Pruning will search through subtrees, i.e. a portion of a tree that excludes a given node and all its offspring, in order to identify the one that minimizes cost-complexity
* Breiman et al. 1993 proved that for given α and T, there exists a unique smallest subtree of T that minimizes Rα(T)

Cost-complexity pruning

* For a given internal node τ, we define R(Tτ) = , where is the set of terminal nodes that are offspring of τ
* There is a single value α for which the cost-complexities Rα(τ) and Rα(Tτ) are equal, given by ατ =
* For α ατ, the tree with terminal node τ is preferred to the tree including all the offspring of τ
* Define α(1) as the minimal ατ over all internal nodes τ; this node is the weakest link in the tree; multiple nodes may meet this criterion
* Define the first pruned subtree as the tree that converts all nodes τ for which ατ = α(1) to terminal nodes

Cost-complexity pruning

* Then repeat the procedure on to find α(2) and
* Ultimately this leads to nested set of optimal subtrees given by … with corresponding sequence of complexity parameters …
* The best tree is the one that minimizes the true impurity associated with the tree
* An honest estimate of this tree impurity can be found by ten-fold cross-validation
* Therefore, we will now describe cross-validation in the context of cost-complexity pruning

Cost-complexity pruning

* Divide the individuals in the sample L into ten approximately equal parts Li for i = 1,…,10
* For each learning sample L-i we determine the nested set of optimal subtrees based on the original complexity parameters α(k) that were derived using the entire dataset L
* For each i obtain the tree impurities Rts() from running the test sample Li through the trees
* The cross-validation estimates are then RCV() = ()
* These are honest estimates of the tree impurities associated with the nested subset of trees based on the entire sample L

Cost-complexity pruning

* Finally, we account for the variability in the estimated tree impurity
* Denote by sek the standard error of RCV(), which is calculated from (sek)2 = () - RCV()]2
* Suppose RCV() is minimized for k = k\*
* Then we select the best tree as the smallest tree such that RCV() is within the interval [RCV() - sek\* , RCV() + sek\*]

Pruning regression tree for Virco data

* Consider regression tree Tree constructed previously to predict APV.Fold using categorical predictors

> Tree

n=939 (127 observations deleted due to missingness)

node), split, n, deviance, yval

\* denotes terminal node

1) root 939 356632.300 12.946540

2) P54=-,A,L,MI,S,T,TI,TS,V,VA,VI,VIM,VL,X 889 237601.200 10.726550

4) P46=-,ILM,IM,LM,LMI,MI,MIL,ML,V,VIM,X 481 44960.940 4.506653

8) P54=-,T,TI,TS,VI,X 342 4475.893 1.980702 \*

9) P54=A,L,MI,S,V,VA 139 32934.020 10.721580

18) P89=-,M,ML 132 24074.510 9.125000

…

Pruning regression tree for Virco data

* Textual overview

> printcp(Tree)

…

Root node error: 356632/939 = 379.8

n=939 (127 observations deleted due to missingness)

CP nsplit rel error xerror xstd

1 0.230717 0 1.00000 1.00200 0.080864

2 0.113693 1 0.76928 0.80533 0.066752

3 0.042528 2 0.65559 0.69127 0.058336

4 0.024727 3 0.61306 0.67654 0.059109

5 0.024016 5 0.56361 0.69247 0.061537

6 0.022684 6 0.53959 0.69393 0.061909

…

Pruning regression tree for Virco data

* Looking in detail at second optimal subtree

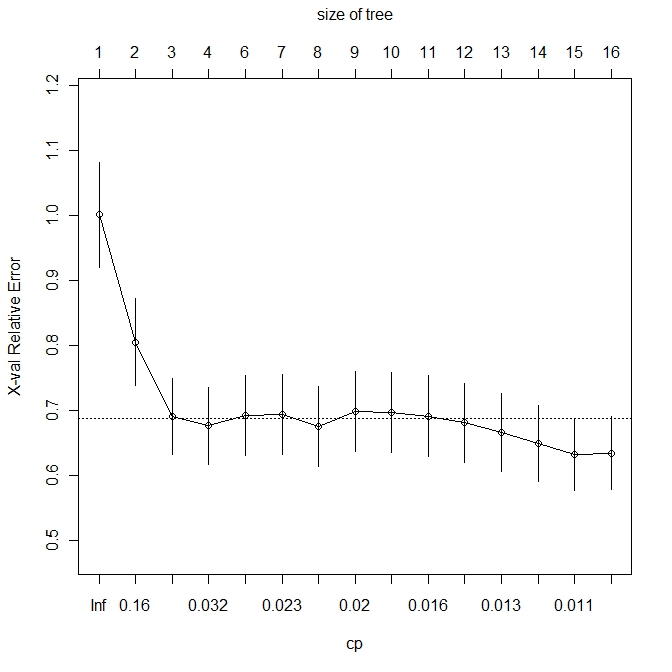
2 0.113693 1 0.76928 0.80533 0.066752

* The complexity parameter at this stage is 0.1137
* The number of splits is 1
* The resubstitution estimate of the relative tree impurity (relative to the root) is 0.7693; this keeps decreasing when adding splits
* The cross-validation estimate of the relative tree impurity is 0.8053; this does not keep decreasing when adding splits
* The standard deviation on the CV’ed relative tree impurity is 0.0668
* In order to make the CV estimates reproducible, a random seed needs to be specified, e.g. set.seed(1980), before running rpart

Pruning regression tree for Virco data

* Visual overview

> plotcp(Tree)



Pruning regression tree for Virco data

* Determine the cutoff value RCV() + sek\*

> cutoff <- (Tree$cptable[,'xerror']+Tree$cptable[,'xstd'])[which.min(Tree$cptable[,'xerror'])]

> cutoff

14

0.6883446

* Determine the smallest such that RCV() satisfies cutoff

> cpvalue <- Tree$cptable[which(Tree$cptable[,'xerror']<cutoff)[1],'CP']

> cpvalue

[1] 0.02472712

* In this example, also first increase in CV’ed tree impurity is here

Pruning regression tree for Virco data

* Prune the tree using the selected complexity parameter

> PrunedTree <-prune(Tree,cp=cpvalue)

> PrunedTree

n=939 (127 observations deleted due to missingness)

node), split, n, deviance, yval

\* denotes terminal node

1) root 939 356632.30 12.946540

2) P54=-,A,L,MI,S,T,TI,TS,V,VA,VI,VIM,VL,X 889 237601.20 10.726550

4) P46=-,ILM,IM,LM,LMI,MI,MIL,ML,V,VIM,X 481 44960.94 4.506653 \*

5) P46=I,L,LI,LIM,VL 408 152093.90 18.059310

10) P47=- 340 107235.30 15.332650 \*

11) P47=A,V,VI 68 29691.79 31.692650 \*

3) P54=LI,M,MIL,VM 50 36749.95 52.418000 \*

Pruning regression tree for Virco data

* Compare initial and pruned tree visually

> plot(Tree)

> plot(PrunedTree)

|  |  |
| --- | --- |
|  |  |

Random Forests (RF)

* Breiman et al. 2001
* Extension of CART
* Generates an ensemble of classification or regression trees
* Does not yield a clear structure for the model of association, i.e. a final tree like in CART is not obtained
* Instead, we get a measure of variable importance for each of the potential predictors
* Hence, there is a paradigm shift from characterizing structure to quantifying importance

Random forest algorithm

* Randomly sample with replacement n individuals (bootstrapping)
* This learning sample (LS) contains n10.632 n unique individuals
* The other n2 = n - n1 individuals are called out-of-bag (OOB) data
* Generate unpruned tree in the LS data only, randomly selecting a subset of the p predictors at each node to be considered as the potential splitting variables
* Typically approximately predictors are selected in classification and approximately p/3 in regression problems
* Repeated sampling of sets of predictors at the tree-splitting stage offers a natural approach to deal with collinearity among predictors, e.g. high LD between SNPs

Random forest algorithm

* Measure overall tree impurity () by running the OOB data through the tree
* For each j = 1,…,p permute **x**j and reassess overall tree impurity, now denoted by (), using the permuted data
* Define variable importance for the j-th predictor as the increase in tree impurity in permuted data, i.e. δbj = () - ()
* The idea is that permuting an unimportant variable will lead to a small change in the overall tree impurity, while permuting an influential variable will lead to a considerable increase
* In fact, using OOB data of (repeated) bootstrapping replaces pruning to ensure applicability of the results to alternative samples

Random forest algorithm

* Repeat the whole procedure B times for b = 1,…,B
* The final variable importance scores are averaged over these multiple runs, i.e.
* Often a standardized measure of variable importance is used, i.e. , where the standard error is obtained from (sej)2 = - )2; this is called the mean decrease in accuracy
* Alternatively, the average (over the B trees) total node impurity explained by splits on the corresponding variable is considered, calculated based on the OOB data but not involving permutations; this is called the mean decrease in node impurity

Random forest for Virco data

* Random forests can be obtained in R using the randomForest package obtainable from CRAN

> install.packages('randomForest')

> library(randomForest)

* We again consider the difference in NFV and IDV fold resistance as the trait and the indicators for mutations as the predictors
* As the randomForest() function does not allow missing data in the trait, we first restrict to those individuals with complete trait data

> Trait.c <- Trait[!is.na(Trait)]

> VircoGenoBin.c <- VircoGenoBin[!is.na(Trait),]

* A seed needs to be specified for reproducibility due to randomness

Random forest for Virco data

* Fit the random forest

> set.seed(1980)

> RegRF <- randomForest(VircoGenoBin.c, Trait.c, importance=TRUE)

> RegRF

Call:

randomForest(x = VircoGenoBin.c, y = Trait.c, importance = TRUE)

Type of random forest: regression

Number of trees: 500

No. of variables tried at each split: 33

Mean of squared residuals: 5584.832

% Var explained: 15.33

Random forest for Virco data

* Obtain variable importance measures ordered according to mean decrease in accuracy

> RegRF$'importance'[order(RegRF$'importance'[,1], decreasing=TRUE),]

%IncMSE IncNodePurity

P36 4.972631e+03 4.478201e+05

P35 3.386103e+03 5.212716e+05

P54 2.405841e+03 2.399119e+05

P20 2.197378e+03 4.133476e+05

P73 2.174494e+03 4.251285e+05

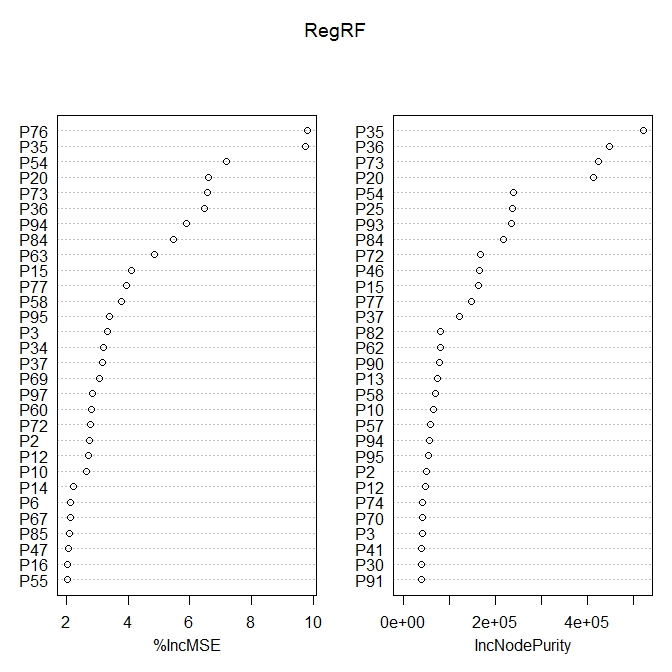
P84 9.781583e+02 2.173470e+05

…

Random forest for Virco data

* Plot ordered variable importance measures

> varImpPlot(RegRF)



Random forest for Virco data

* The five most important variables according to mean decrease in accuracy (left on figure) are P76, P35, P54, P20 and P73
* Interestingly, P76 and P20 were not our original unpruned tree
* Top-ranked variables are most predictive of the trait among those considered, but are not always statistically significant predictors of the trait; nevertheless they are worthy of further investigation
* The higher B, specified by the parameter ntree, the more accurate and less dependent on random seed the results become
* One can use the function getTree() to investigate the structure of particular trees within the forest, but the results can vary dramatically from tree to tree and interpretation is difficult

RF with missing predictors

* If missingness is random, one can simply remove individuals with any missing data; this is called a complete case analysis
* However, when there are many predictors, the majority of the individuals might be incomplete w.r.t. at least one of these
* Simple imputation replaces each missing value with a single imputed value based on the observed data for this variable
* For categorical predictors, missing values are typically replaced by the most frequently observed category for this variable
* For quantitative predictors, missing values are typically replaced by the median of the observed data for this variable
* Breiman 2003 describes a more advanced approach to missing data

Introducing missing genotypes in Virco data

* First we randomly generate 5% missing genotypes

> VircoGenoCat.c <- VircoGenoCat[!is.na(Trait),]

> mean(is.na(VircoGenoCat.c))

[1] 0

> VircoGenoCat.m <- VircoGenoCat.c

> set.seed(1980)

> makeNA <- matrix(sample(c(FALSE,TRUE), nrow(VircoGenoCat.c)\*ncol(VircoGenoCat.c), replace=TRUE, prob=c(0.95, 0.05)), nrow=nrow(VircoGenoCat.c), ncol=ncol(VircoGenoCat.c))

> VircoGenoCat.m[makeNA] <- NA

> mean(is.na(VircoGenoCat.m))

[1] 0.0505568

RF with missing predictors for Virco data

* Perform single imputation using na.roughfix()

> VircoGenoCat.r <- na.roughfix(VircoGenoCat.m)

> mean(is.na(VircoGenoCat.r))

[1] 0

* As missingness is simulated, we can evaluate quality of imputation

> 1-mean(VircoGenoCat.r!=VircoGenoCat.c)/mean(is.na(VircoGenoCat.m))

[1] 0.8808598

RF with missing predictors for Virco data

* Study in detail what happens

> table(VircoGenoCat.c$P71)

- AT AV F G I L T TA TI V VA VAIT VATI VI VL X

404 1 2 1 1 27 8 96 12 2 386 23 1 2 7 1 2

> table(VircoGenoCat.m$P71)

- AT AV F G I L T TA TI V VA VAIT VATI VI VL X

380 1 2 1 1 26 8 91 11 2 369 22 1 2 7 1 2

> table(VircoGenoCat.r$P71)

- AT AV F G I L T TA TI V VA VAIT VATI VI VL X

429 1 2 1 1 26 8 91 11 2 369 22 1 2 7 1 2

RF with missing predictors for Virco data

* Perform random forest as before

> set.seed(1980)

> RF.r <- randomForest(VircoGenoCat.r, Trait.c, importance=TRUE)

> RF.r

Call:

randomForest(x = VircoGenoCat.r, y = Trait.c, importance = TRUE)

Type of random forest: regression

Number of trees: 500

No. of variables tried at each split: 33

Mean of squared residuals: 6625.574

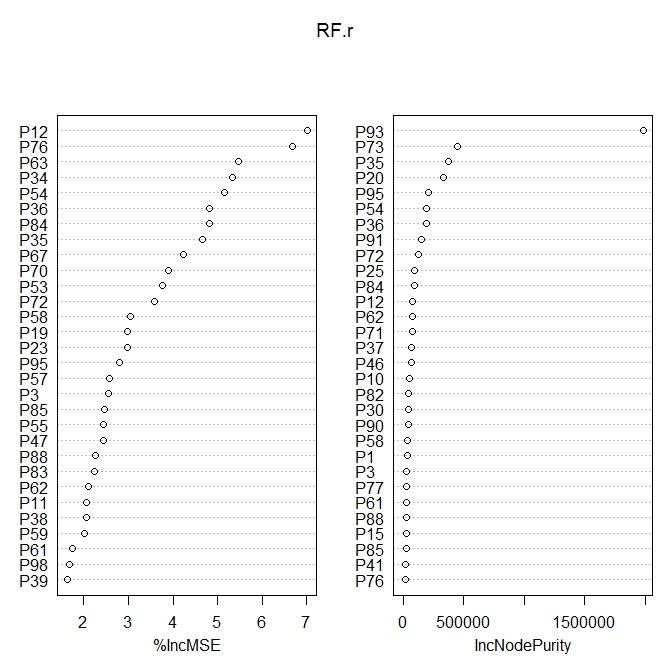
% Var explained: -0.44

* Bad performance not due to missingness but categorical variables

Random forest for Virco data

* Plot ordered variable importance measures again

> varImpPlot(RF.r)



More on trees and forests

* The ctree() and cforest() functions in package party (available from CRAN) provide a well-defined conditional inference procedure with splitting rules based on statistical testing that overcome selection bias towards covariates with many possible splits or missing values
* The rfImpute() function in package randomForest provides a more sophisticated approach to missing predictor variables using multiple imputation and trait information to reconstruct missing values
* Package mirf (see CRAN) can be used with unknown allelic phase
* Random Jungle provides an efficient implementation of random forest methodology that can also run on multiple CPUs: <http://www.imbs-luebeck.de/imbs/de/node/227>

Exercises

* For each of the exercises use the binary indicators for mutations rather than the categorical variables
* Construct and prune a classification tree to determine whether any mutations within the protease region of the viral genome are associated with greater APV than IDV fold resistance
* Construct and prune a regression tree to determine whether any mutations within the protease region of the viral genome are associated with a difference in APV and IDV fold resistance
* Grow a random forest to determine which protease mutations are most highly associated with SQV fold resistance
* Repeat after introducing 10% missingness in the genotypes