Metode ansamblu
Ensemble learning

Ruxandra Stoean
rstoean@inf.ucv.ro
http://inf.ucv.ro/~rstoean

Bibliografie

- Breiman, L., Bagging Predictors, Machine Learning , vol. 24, issue 2, pp. 123-140, 1996
- http://www.cs.utsa.edu/~bylander/cs6243/bagging-boosting.pdf
- Freund, Y. and Schapire, R.E., Experiments with a new boosting algorithm, Proceedings of the Thirteenth International Conference on Machine Learning, 148–156, Morgan Kaufmann, 1996
- Breiman, L., Random Forests, Machine Learning 45(1), 5-32, 2001.
- Dianne Cook, Deborah F. Swayne, Graphics for Data Analysis. Interactive and Dynamic With R and Ggobi, Springer, 2007

Invatare ansamblu

- Se antreneaza mai multi clasificatori de baza si se combina predictiile acestora.

- Modul de aplicatie al clasificatorilor poate avea loc la mai multe niveluri.

- Abordari clasice:
  - Bagging
  - Boosting
  - Random forests

Bagging

- Este prescurtarea de la Bootstrap AGGregatING.
- Se imparte multimea de *m* date de antrenament in *b* parti (bags).
  - Fiecare submultime are tot *m* elemente.
  - Datele din fiecare submultime sunt selectate aleator cu inlocuire.
- Clasificatorul de baza ales se antreneaza pe fiecare submultime.
- Cele *b* modele construite voteaza iesirea pentru datele noi de test:
  - Media iesirilor celor *b* modele - pentru regresie
  - Clasa cu cele mai multe aparitii in predictiile celor *b* modele - pentru clasificare
- Exemplele neselectate (out-of-bag) se utilizeaza pentru a estima eroarea de generalizare.

Pachetul R ipred

- Clasificatorul de baza este un arbore de decizie (bagged tree).
  - Este folosit in acest sens pachetul rpart.
- Daca iesirea este vazuta drept factor, se are in vedere clasificarea.
  - Daca este numerica, se trateaza regresia.
- Parametrul nbagg specifica numarul de replicatii dorit – default 25.
- Parametrul coob=true specifica dorinta de a calcula o estimare a erorii de generalizare (out-of-bag estimation)

Exemplu 1/2

```
library(ipred) # pachetul pentru functia bagging
library(mlbench) # pachetul pentru Breast Cancer
library(e1071) # pachetul pentru matricea de confuzie

data(BreastCancer)
dat <- BreastCancer

classColumn <- 11
# se imparte multimea de date o data in training si test
testindex <- sample(index, trunc(length(index)/4))
testset <- dat[testindex, ]
trainset <- dat[-testindex, ]
```

Exemplu 2/2

```
# se aplica bagging pe multimea de antrenament
# se doreste si estimarea erorii de predictie
bg <- bagging(Class ~., data = trainset, coob=TRUE)
print(bg)

bg.pred <- predict(bg, testset[, -classColumn])
contab <- table(pred = bg.pred, true = testset[, classColumn])
accuracy <- classAgreement(contab)$diag

print(accuracy)
```

Rezultat

```
> bg <- bagging(Class ~., data = trainset, coob=TRUE)
> print(bg)

Bagging classification trees with 25 bootstrap replications

Call: bagging.data.frame(formula = Class ~ ., data = trainset, coob = TRUE)

Out-of-bag estimate of misclassification error:  0.0458

>
> bg.pred <- predict(bg, testset[, -classColumn])
> contab <- table(pred = bg.pred, true = testset[, classColumn])
> accuracy <- classAgreement(contab)$diag
>
> print(accuracy)
[1] 0.96
```

Boosting: Adaboost – ADAptive BOOSTing

- Are ca scop marirea acuratetii pentru un clasificator.
- Se aplica clasificatorul de baza in mod repetat.
- Fiecare data de antrenament (din cele m) are atasata o pondere – initial toate egale cu $1/m$.
- La fiecare pas, datele clasificate gresit primesc pondere mai mare.
- Modelul rezultat la fiecare pas primeste un vot ponderat dupa acuratetea sa de predictie
  - Ponderea e data de masura clasificarilor sale gresite pe multimea de antrenament.

Pachetul R ada

- Implementeaza din nou un arbore de decizie ca si clasificator de baza (boosted tree) – folosind tot pachetul rpart.
- Parametrul iter specifica numarul de iteratii dorit.
- Functia plot ofera posibilitatea vizualizarii:
  - Erorii de clasificare la fiecare iteratie a algoritmului de boosting
    - Eroarea poate fi calculata atat pentru multimea de training cat si pentru cea de test
  - Masurii de concordanta (agreement) kappa dintre clasificarea prognozata si cea reala la fiecare iteratie de boosting pentru ambele multimi

Exemplu 1/2

```
library(ada)
library(mlbench)
library(e1071)

data(BreastCancer)
dat <- BreastCancer

classColumn <- 11

testindex <- sample(index, trunc(length(index)/4))
testset <- dat[testindex, ]
trainset <- dat[-testindex, ]
```

Exemplu 2/2

```
boost <- ada(Class~., data = trainset, iter=20)
summary(boost)

# adaugam multimea de test
boost=addtest(boost, testset[, -classColumn], testset[, classColumn])

# graficul erorii de clasificare per iteratie
plot(boost,TRUE,TRUE)

boost.pred <- predict(boost, testset[, -classColumn])
contab <- table(pred = boost.pred, true = testset[, classColumn])
accuracy <- classAgreement(contab)$diag
print(accuracy)
```

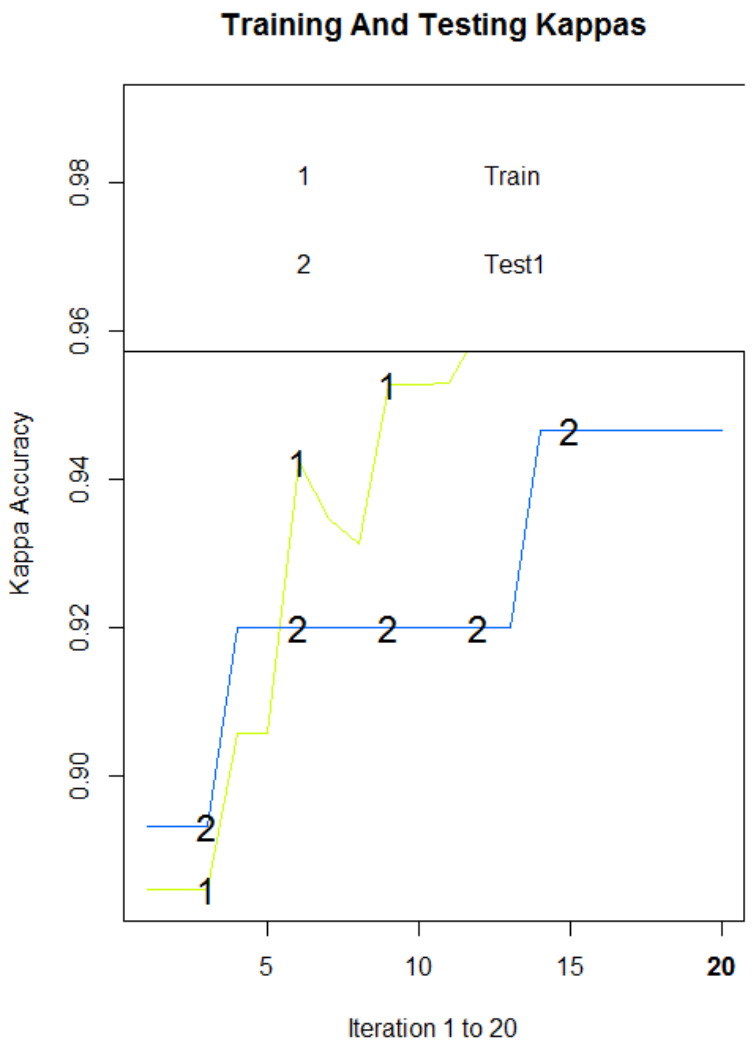
Rezultate

```
> library(ada)
> library(mlbench)
> library(e1071)
>
> data(BreastCancer)
> dat <- BreastCancer
>
> classColumn <- 11
>
> testindex <- sample(index, trunc(length(index)/4))
> testset <- dat[testindex, ]
> trainset <- dat[-testindex, ]
>
> boost <- ada(Class~., data = trainset,iter=20,nu=1,type="discrete")
> summary(boost)
Call:
ada(Class ~ ., data = trainset, iter = 20, nu = 1, type = "discrete")

Loss: exponential Method: discrete   Iteration: 20

Training Results

Accuracy: 0.995 Kappa: 0.989
```

```
> ##add testing data set
> boost=addtest(boost, testset[, -classColumn], testset[, classColumn])
> ##plot boost
> plot(boost,TRUE,TRUE)
>
> boost.pred <- predict(boost, testset[, -classColumn])
> contab <- table(pred = boost.pred, true = testset[, classColumn])
> accuracy <- classAgreement(contab)$diag
>
> print(accuracy)
[1] 0.9733333
```

Plot


Training And Testing Error
1 Train
2 Test1
Error
0.01
0.02
0.03
0.04
0.05
0.06
5
10
15
20
Iteration 1 to 20


Training And Testing Kappas
1 Train
2 Test1
Kappa Accuracy
0.90
0.92
0.94
0.96
0.98
5
10
15
20
Iteration 1 to 20

Random forests

- Este o metoda ansamblu ce combina mai multi arbori de decizie.
- Fiecare arbore este generat prin extragerea aleatoare (cu inlocuire) a
  - Atributelor
  - Inregistrarilor (bagging)
- Pentru fiecare arbore exista automat
  - O multime de antrenare (in-bag)
  - O multime de masurare a erorii de generalizare (out-of-bag)
- Clasa unui exemplu nou se stabileste prin vot pentru clasificare si prin medie pentru regresie.

Pachetul randomForest

- Parametri:
  - ntree – numarul de arbori; implicit egal cu 500
  - mtry – numarul de atribute alese aleator
- Metoda nu poate manevra datele lipsa.
  - In apelarea metodei se poate folosi na.action = na.omit pentru a omite datele lipsa.
- Output:
  - Estimarea erorii de generalizare – out-of-bag (OOB) estimate
  - Matricea de confuzie
  - Importanta atributelor

Exemplu 1/2

```
library(randomForest)
library(mlbench)
library(e1071)

data(BreastCancer)
dat <- BreastCancer

classColumn <- 11
noClass <- 2

testindex <- sample(index, trunc(length(index)/4))
testset <- dat[testindex, ]
trainset <- dat[-testindex, ]
```

Exemplu 2/2

```
rf <- randomForest(Class ~ ., data = trainset, importance=TRUE, mtry = 4,
na.action = na.omit)
print(rf)

rf.pred <- predict(rf, testset[, -classColumn])
contab <- table(pred = rf.pred, true = testset[, classColumn])
accuracy <- classAgreement(contab)$diag

print(accuracy)

#importanta variabilelor, bazata pe masura Gini
noColumn <- noClass+2
order(rf$importance[,noColumn], decreasing=T)
```

```
> rf <- randomForest(Class ~ ., data = trainset, importance=TRUE, mtry = 4, na.
action = na.omit)
> print(rf)

Call:
 randomForest(formula = Class ~ ., data = trainset, importance = TRUE,      mtr
y = 4, na.action = na.omit)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 3.11%
Confusion matrix:
          benign malignant class.error
benign       396        12  0.02941176
malignant      7       196  0.03448276
>
> rf.pred <- predict(rf, testset[, -classColumn])
> contab <- table(pred = rf.pred, true = testset[, classColumn])
> accuracy <- classAgreement(contab)$diag
>
> print(accuracy)
[1] 0.9861111
>
> #importanta variabilelor, bazata pe masura Gini
> noColumn <- noClass+2
> order(rf$importance[,noColumn], decreasing=T)
 [1]  3  4  8  7  6  9  2  5  1 10
```

Exercitii

- Aplicati bagging, boosting si random forests din R pentru problema Pima Indians Diabetes [1].

[1] https://archive.ics.uci.edu/ml/datasets/Pima+Indians+Diabetes