Machine Learning - 1100-ML0ENG (Ćwiczenia informatyczne Z-23/24)

Gradient boosting - gbm

Gradient boosting gbm

```
install.packages("gbm")
library(gbm)
```

Gradient boosting ensembles weak learners and creates a new base learner that maximally correlates with the negative gradient of the loss function.

The set used for modelling, must contain only numerical values.

We transform the set **gc** to numerical form

```
gc<-read.csv("germancredit.csv", stringsAsFactors = T)
gc.dv <- dummyVars("~ .",gc[-21], fullRank = F)
gc.d = as.data.frame(predict(gc.dv, newdata=gc[-21]))
gc.d=cbind(gc.d,gc[21])
str(gc.d)
summary(gc.d)
```

```
library(caTools)
set.seed(12345)
split = sample.split(gc.d$credit_risk, SplitRatio = 0.7)
gc.d.Train <- subset(gc.d, split == TRUE)
gc.d.Test <- subset(gc.d, split == FALSE)
```

```
str(gc.d.Train)
```

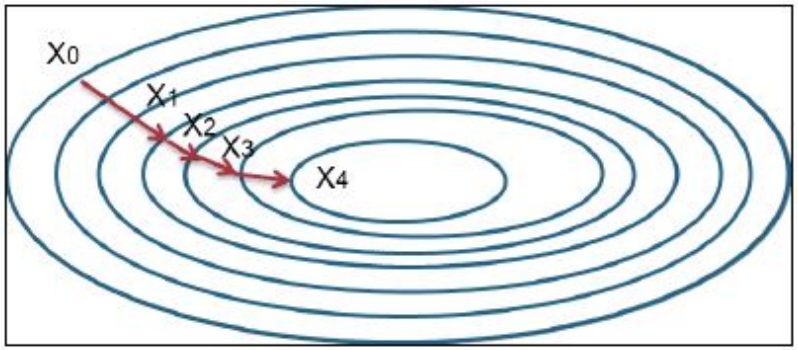
```

prop.table(table(gc.d$credit_risk))
prop.table(table(gc.d.Train$credit_risk))
prop.table(table(gc.d.Test$credit_risk))
```

The gbm function

Parameters used:

1. **distribution**
   - bernoulli - for binary classifications 0-1
   - gaussian - (regresion) root mean square errors
   - adaboost - for binary classifications 0-1
   - poisson - counting data
2. **shrinkage <-> learning rate**
3. **bag.fraction** - coefficient controlling the selection of observations in the training set - part of the observations from the training set, is used to build the next tree. The default is 0.5. For small collections it is worth increasing this parameter.
4. **interaction.depth** - the default depth of each tree is 1, which means that we create a set of stumps
5. n.minobsinnode - minimum number of observations per leaf
6. n.trees - the number of iterations




```
gc.d.gbm <- gbm(credit_risk~.,distribution="bernoulli",data= gc.d.Train,
 n.trees=500,shrinkage = 0.05)
```

```
gc.d.gbm
summary(gc.d.gbm)
```

We use the **gbm.perf** function to find the optimum iteration. The error measurement here is a **bernoulli distribution**, which we have defined earlier in the training stage. The blue dash line on the plot shows where the optimum iteration is.

```
n.trees=gbm.perf(gc.d.gbm)
```

Testing the built model

```
gc.d.gbm.pred=predict(gc.d.gbm, gc.d.Test, n.trees = n.trees)
```

Then, we use the predict function to obtain the odd value of a log in each testing case returned from the Bernoulli loss function. In order to get the best prediction result, one can set the n.trees argument to an optimum iteration number.

However, as the returned value is an odd value log, we still have to determine the best cut off to determine the label. Therefore, we use the roc function to generate an ROC curve and get the cut off with the maximum

accuracy.

.

So

```
#install.packages("pROC")
library(pROC)
gbm.roc = roc(gc.d.Test$credit_risk, gc.d.gbm.pred)
x=plot(gbm.roc)
x
coords(gbm.roc, "best")
```

```
> coords(gbm.roc, "best")
  threshold specificity sensitivity
1 0.6883459   0.7777778         0.7
```

```
gc.d.gbm.pred.class = ifelse(gc.d.gbm.pred > 0.6883459, 1, 0)
```

```
table(gc.d.gbm.pred.class, gc.d.Test$credit_risk)
```

```
acc(gc.d.gbm.pred.class, gc.d.Test$credit_risk)
```

Note that the **shrinkage paramete**r cannot be too low. This is because we have obtained a model without significant variables on the target variable.

```
gc.d.gbm2 <- gbm(credit_risk~.,distribution="bernoulli",data= gc.d.Train,
 n.trees=500,shrinkage = 0.00001)
gc.d.gbm2
```

```
gbm.perf(gc.d.gbm,plot.it=TRUE)
gbm.perf(gc.d.gbm2,plot.it=TRUE)
```

Dataset: sna

```
sna<-read.csv("sna.csv", stringsAsFactors = T)
summary(sna)
sna=sna[-1]
sn.dv <- dummyVars("~.",sna[-2:-4], fullRank = F)
sn.d = as.data.frame(predict(sn.dv, newdata=sna[-2:-4]))
sn.d=cbind(sn.d,sna[2:4])
str(sn.d)
summary(sn.d)
```

```
library(caTools)
set.seed(12345)
split = sample.split(sn.d$Purchased, SplitRatio = 0.7)
sn.d.Train <- subset(sn.d, split == TRUE)
sn.d.Test <- subset(sn.d, split == FALSE)
str(sn.d.Train)
```

```
prop.table(table(sn.d$Purchased))
prop.table(table(sn.d.Train$Purchased))
prop.table(table(sn.d.Test$Purchased))
```

```

sn.d.gbm <- gbm(Purchased~.,distribution="bernoulli",data= sn.d.Train,
 n.trees=500,shrinkage = 0.05)
sn.d.gbm
```

```
n.trees.1=gbm.perf(sn.d.gbm)
```

```

summary(sn.d.gbm)
summary(sn.d.gbm,n.trees=1) # for the first tree
summary(sn.d.gbm,n.trees=500)
```

```
sn.d.gbm.pred=predict(sn.d.gbm, sn.d.Test, n.trees = 500,type = "response")
```

```
pp=predict.gbm(sn.d.gbm, sn.d.Test,type = "response")
klasa = ifelse(pp > 0.5, 1, 0)
table(klasa)
```

```
table(klasa, sn.d.Test$Purchased)
acc(klasa, sn.d.Test$Purchased)
roc.function(klasa, sn.d.Test$Purchased)
```

Last modified: środa, 17 stycznia 2024, 9:33

Uniwersytetu Łódzkiego obowiązuje nazwa użytkownika i hasło wykorzystywane podczas logowania się do systemu USOSweb.