adaboost615

Dylan Sun and Nina Zhou

2016-12-13

Contents

Introduction

For our project, we have implemented a specific case of ADABOOST, an ensemble classification alogorithm. The paper we followed is:

`RATSCH, G. "Soft Margins for AdaBoost." Machine Learning, 42, 287-320, 2001.`

Our function takes in a binary outcome variable and any number of continuous predictors (categorical variables must be binary or ordinal).

Specifically, we implemented a simplified, heavily modified version of the algorithm that uses weighted univariate logistic regression as the weak classifier. The algorithm loops once for each predictor, recalculating the weak classifier while emphasizing data points that were misclassified in the previous iteration.

A description of the algorithm is as follows:

1. Input the training data and choose initial weights for each data pair; for initializing, just weight each data pair the same (must add to 1)
2. Train the weak classifier with respect to the weighted sample set and obtain hypothesis classifications; in our case, we run univariate logistic regression on a single column and save the fitted beta values.
3. Calculate the training error.
4. Update the weights for each data pair, using the log odds of the training error. If the original pair was classified incorrectly, its weight goes up; if the pair was classified correctly, the weight goes down.
5. Based on the training error, save the overall weight of this weak classifier. Weak classifiers that have poor training errors will have lower weights.
6. Repeat until there are no more predictors.
7. The saved betas and weights can be used for prediction.

Installation Instructions

Our package can be downloaded directly from our github repository within R.

Instructions:

1. install.packages(“devtools”)
2. library(devtools)
3. install_github(repo = “dylanfsun/adaboost_615”)
4. library(adaboost615)
5. ?adaboost615

6. ?test.data
7. ?univLogReg
8. ?predictAda615
9. ?errorRate

Loading in the test data

Our test data is based on the publicly available Titanic dataset.

The first two columns of the test data, Age and Sex, and the outcome variable, Survival, were directly subsetted from the Titanic dataset. See `?test.data` for more details on the original dataset and how to download it.

```
library(adaboost615)
data(test.data) # see ?test.data
```

Age	Sex	1	2	3	4	5	6	7	Survival
29.00	0	38.00576	42.63594	85.13355	58.20710	16.67390	81.43056	29.07983	1
2.00	0	65.24753	31.65398	63.17236	33.15056	52.80509	33.21826	131.90626	0
30.00	1	67.26989	52.38348	85.28449	29.05324	61.44706	58.50666	54.24469	0
25.00	0	82.08835	78.77719	35.63550	34.49266	59.81800	30.26527	50.99907	0
0.92	1	43.53146	46.51378	49.75485	43.85330	46.94532	61.58333	62.39378	1
47.00	1	41.30894	43.37413	39.19790	56.90094	73.92038	34.29750	91.09393	1
63.00	0	74.37572	56.29026	34.74858	54.30321	53.13687	74.05210	18.01960	1
39.00	1	25.59125	47.36002	52.46127	50.34745	24.15067	52.51977	50.15696	0
58.00	0	35.58119	44.54205	38.27009	40.79579	50.14332	43.70558	36.79103	1
71.00	1	30.65308	35.46754	32.77753	27.20887	25.77023	34.34763	38.30459	0

Example 1: two main predictors

This example uses the first 656 rows as the training dataset and the last 100 as the test dataset. It builds the model using only two preidctors: Sex and Age, the two continuous predictors originally included with the dataset.

Error rate

```
beta_weights1 <- adaboost(as.matrix(test.data[1:656, c(1:2,1003)]))
pred1 <- predictAda615(as.matrix(test.data[657:756,1:2]), beta_weights1)
errorRate(pred1, test.data[657:756,1003])
```

```
## [1] 0.32
```

Using these two predictors, our error rate on the test dataset is 0.32. This is the same error rate produced by fitting a logistic regression model using the glm package in R with both covariates.

```
logmodel <- glm(Survival ~ Age + Sex, data = test.data[1:656, c(1:2,1003)],
                family = binomial(link = "logit"))
pred1 <- predict(logmodel, test.data[657:756,1:2], type = "response") >= 0.5
errorRate(pred1, test.data[657:756, 1003])
```

```
## [1] 0.32
```

Benchmarking

```
library(microbenchmark)
microbenchmark(beta_weights1<-adaboost(as.matrix(test.data[1:656,c(1:2,1003)])),
               logmodel <- glm(Survival ~ Age + Sex,
                               data = test.data[1:656, c(1:2,1003)],
                               family = binomial(link = "logit"))
  )
```

```
## Unit: milliseconds
##
##                                                  beta_weights1 <- adaboost(as.matrix(test.data[1:656
##  logmodel <- glm(Survival ~ Age + Sex, data = test.data[1:656,      c(1:2, 1003)], family = binomial
##       min       lq     mean   median       uq      max neval
##  2.038901 2.077398 2.183951 2.133781 2.221842 3.097388   100
##  2.466077 2.582631 3.185898 2.852886 3.432372 6.639241   100
```

Based on microbenchmark with 100 evaluations, our adaboost function model-fitting step runs faster than the R glm function when only using these two predictors, while retaining the same error rate.

Example 2: Two main predictors with nine columns of random noise

In this example, nine columns of randomly generated noise are added to the model. These columns were pre-generated and have already been included in the test.data dataset.

Error rate

```
## After adding 9 columns of noise
beta_weights2 <- adaboost(as.matrix(test.data[1:656, c(1:11,1003)]))
pred2 <- predictAda615(as.matrix(test.data[657:756,1:11]), beta_weights2)
errorRate(pred2, test.data[657:756,1003])
```

```
## [1] 0.32
```

```
logmodel <- glm(Survival ~ ., data = test.data[1:656, c(1:11,1003)],
                family = binomial(link = "logit"))
pred2 <- predict(logmodel, test.data[657:756,1:11], type = "response") >= 0.5
errorRate(pred2, test.data[657:756, 1003])
```

```
## [1] 0.32
```

Benchmarking

```
microbenchmark(beta_weights2 <- adaboost(as.matrix(test.data[1:656, c(1:11,1003)])),
               logmodel <- glm(Survival ~ .,
                               data = test.data[1:656, c(1:11,1003)],
                               family = binomial(link = "logit"))
  )
```

```
## Unit: milliseconds
##
##                                    beta_weights2 <- adaboost(as.matrix(test.data[1:656, c(1:11,
```

```
##  logmodel <- glm(Survival ~ ., data = test.data[1:656, c(1:11,      1003)], family = binomial(link =
##       min       lq     mean    median        uq      max neval
##  9.835609 10.07818 10.86791 10.413539 11.204822 15.16426   100
##  4.154422  4.47262  5.51576  4.840459  6.275086 11.14605   100
```

On 11 predictors, the error rate is still the same for both functions. The adaboost function is slower since it fits a separate logistic regression for each predictor, while glm is fitting only once.

Example 3: Two main predictors with 1000 columns of random noise

Similar to example 2, the model is rerun, but now with 1000 columns of random noise added to the dataset, rather than just 9.

Error rate

```
## Adding 1000 columns of noise
beta_weights3 <- adaboost(as.matrix(test.data[1:656,]))
pred3 <- predictAda615(as.matrix(test.data[657:756,-1003]),beta_weights3)
errorRate(pred3,test.data[657:756,1003])
```

```
## [1] 0.24
```

```
logmodel <- glm(Survival ~ ., data = test.data[1:656,],
                family = binomial(link = "logit"))
pred3 <- predict(logmodel, test.data[657:756,], type = "response") >= 0.5
errorRate(pred3, test.data[657:756, 1003])
```

```
## [1] 0.5
```

Benchmarking

```
microbenchmark(beta_weights3 <- adaboost(as.matrix(test.data[1:656,])),
               logmodel <- glm(Survival ~ ., data = test.data[1:656,],
                               family = binomial(link = "logit")),
               times = 2
  )
```

```
## Unit: milliseconds
##                                                                                  expr
##                                     beta_weights3 <- adaboost(as.matrix(test.data[1:656, ]))
##  logmodel <- glm(Survival ~ ., data = test.data[1:656, ], family = binomial(link = "logit"))
##        min        lq      mean    median        uq       max neval
##   887.1912  887.1912  913.7009  913.7009  940.2106  940.2106     2
##  8390.3667 8390.3667 8652.1601 8652.1601 8913.9536 8913.9536     2
```

With 1002 predictors, the glm function fails to converge and is extremely slow. The adaboost function decreases in test error after incorporating the random noise. Note that 0.24 is the proportion of survivors in the dataset overall.