Comments on bnclassify package runtimes

Bojan Mihaljevic, Concha Bielza, Pedro Larranaga

2017-09-06

Contents

1 Prediction

bnclassify implements fast prediction for augmented naive Bayes models with complete data. On the car evaluation data set (see `vignette("introduction", package="bnclassify")`) it is roughly 100 times faster than prediction with the gRain (Højsgaard 2012) package.

```
library(bnclassify)
data(car)
nb <- lp(nb('class', car), car, smooth = 0)
gr <- as_grain(nb)
library(microbenchmark)
microbenchmark(bnclassify = predict(nb, car),
               gRain  = gRain::predict.grain(gr, 'class', newdata = car),
               times = 1)
#> Unit: milliseconds
#>        expr         min          lq        mean      median          uq
#>  bnclassify    3.123856    3.123856    3.123856    3.123856    3.123856
#>       gRain 1059.465100 1059.465100 1059.465100 1059.465100 1059.465100
#>          max neval
#>     3.123856     1
#>  1059.465100     1
```

2 Wrapper algorithms

The wrapper algorithms can be computationally intensive. The following are runtimes for `tan_hc` on a Windows 7, 2.80 GHz, 16 GB RAM machine.

```
microbenchmark(
  tan_hc = {set.seed(0); t <- b <- tan_hc('class', car, k = 10,
                                          epsilon = 0)},
```

```r
  tan_hc5 = {set.seed(0); t <- b <- tan_hc('class', car, k = 5,
                                             epsilon = 0)},
  times = 1)
#> Unit: milliseconds
#>     expr     min      lq    mean  median      uq     max neval
#>   tan_hc 623.358 623.358 623.358 623.358 623.358 623.358     1
#>  tan_hc5 392.908 392.908 392.908 392.908 392.908 392.908     1
```

5-fold cross-validation should take roughly 5 times more than learning.

```r
tan_hc5 <- tan_hc('class', car, k = 5, epsilon = 0)
tan_hc5 <- lp(tan_hc5, car, smooth = 1)
microbenchmark(tan_hc = {set.seed(0); cv(tan_hc5, car, k = 5)},
               times = 1)
#> Unit: seconds
#>    expr      min       lq     mean   median       uq      max neval
#>  tan_hc 2.002771 2.002771 2.002771 2.002771 2.002771 2.002771     1
```

With the Soybean data set, which has 36 features, and 562 instances after removing the incomplete ones, `tan_hc` takes about 80 seconds on the above mentioned Windows 7 machine.

```r
library(mlbench)
data(Soybean)
soy_complete <- na.omit(Soybean)
dim(soy_complete)
microbenchmark(
  tan_hc = {set.seed(0); tan_hc('Class', soy_complete, k = 5,
                                 epsilon = 0)},
  times = 1)
```

3 Incomplete data

`bnclassify` uses `gRain` to compute the class posterior of instances with missing values (`NA`s). Even with a single `NA` in a dataset, runtime degrades significantly.

```r
nb <- bnc('nb', 'class', car, smooth = 1)
car_na <- car
car_na[1, 4] <- NA
microbenchmark(predict(nb, car),
               predict(nb, car_na),
               times = 1)
#> Unit: milliseconds
#>                 expr       min        lq      mean    median        uq
#>     predict(nb, car)  3.004384  3.004384  3.004384  3.004384  3.004384
#>  predict(nb, car_na) 18.052776 18.052776 18.052776 18.052776 18.052776
#>        max neval
#>   3.004384     1
#>  18.052776     1
```

This is especially relevant for wrapper learners, which call prediction during learning. It is therefore probably not a bad idea to use wrappers with incomplete data sets, unless these are rather small.

References

Højsgaard, Søren. 2012. "Graphical Independence Networks with the `gRain` Package for `R`." *Journal of Statistical Software* 46 (10): 1–26.