Practical Machine Learning - Week 3 - Bagging

Jeff Leek, notes + revised graphs by Bruno Fischer Colonimos

22 juin 2017

Contents

1 Required packages (install before running) 1

2 Bootstrap aggregating (bagging) 1

3 Bagging Example: Ozone 1

3.1 data . . . 1
3.2 Bagged loess . . . 2
3.2.1 Resample and predict with loess . . . 2
3.3 Compute bagged loess . . . 2

4 Bagging in `caret` 3

4.1 More bagging in `caret`, Custom Bagging Example . . . 3
4.2 Parts of bagging . . . 5
4.2.1 `fit` . . . 5
4.2.2 `predict` . . . 5
4.2.3 `aggregate` . . . 6

5 Notes and further resources 6

---

1 Required packages (install before running)

“caret”, “party”, “ElemStatLearn”

2 Bootstrap aggregating (bagging)

Basic idea:

1. Resample cases and recalculate predictions
2. Average or majority vote

Notes:

- Similar bias
- Reduced variance
- More useful for non-linear functions

More: http://en.wikipedia.org/wiki/Bootstrap_aggregating

3 Bagging Example: Ozone

3.1 data

```r
library(ElemStatLearn); data(ozone,package="ElemStatLearn")
ozone <- ozone[order(ozone$ozone),]

kable(head(ozone))
```

	ozone	radiation	temperature	wind
17	1	8	59	9.7
19	4	25	61	9.7
14	6	78	57	18.4
45	7	48	80	14.3
106	7	49	69	10.3
7	8	19	61	20.1

I'm going to try to predict temperature as a function of ozone. The first thing that we can do is just show you an example of how this works

3.2 Bagged loess

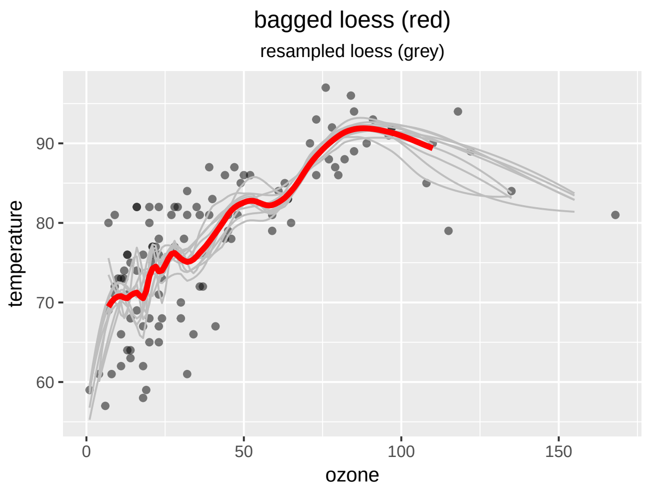
3.2.1 Resample and predict with loess

The basic idea is, I'm going to create a matrix here and it's going to have 10 rows and a 155 columns. Then I'm going to resample the data set ten different times, so then a loop over ten different samples of the data set. Each time I'm going to sample width replacement from the entire data set.

```r
set.seed(1357)

# VO0 == prediction for ozone in 1:155 (original jleek)
ll <- matrix(NA,nrow=10,ncol=155) # the matrix
# loop
for(i in 1:10){
  ss <- sample(1:dim(ozone)[1], replace=T) # resample
  ozone0 <- ozone[ss,] # resampled dataset
  ozone0 <- ozone0[order(ozone0$ozone),] # reordered by ozone
  loess0 <- loess(temperature ~ ozone,data=ozone0,span= 0.3)
  ll[i,] <- predict(loess0,newdata=data.frame(ozone=1:155))
}
```

3.3 Compute bagged loess

```r
# VOriginale (jleek)
# plot(ozone$ozone,ozone$temperature,pch=19,cex=0.5)
# for(i in 1:10){
#         lines(1:155,ll[i,],col="grey",lwd=2)
# }
# lines(1:155,apply(ll,2,mean),col="red",lwd=2)


# VO0 BFC (prediction pour Ozone = 1:155)
# graph basis layer
g <- ggplot(data = ozone, aes(ozone, temperature)) +
        geom_point(alpha = 0.5)

# loop for individual resamples
for (i in 1:10) {
        g <- g + geom_line(data = data.frame(ozone = 1:155,
                                             rpred=ll[i,]),
                           aes(ozone, rpred), color = "grey")
}
# last layer (bagged prediction)
g <- g + geom_line(data = data.frame(ozone = 1:155,
                                     final = apply(ll,2,mean)),
                   aes(ozone, final), color="red", size = 1.5)
# final touch
g + labs(title = "bagged loess (red)",
         subtitle ="resampled loess (grey)") +
        theme(plot.title = element_text(hjust = 0.5),
              plot.subtitle = element_text(hjust = 0.5))
```



4 Bagging in caret

- Some models perform bagging for you, in `train` function consider `method` options
  - `bagEarth`
  - `treebag`
  - `bagFDA`
- Alternatively you can bag any model you choose using the `bag` function. (You can actually build your own bagging function in caret. This is a bit of an advanced use and so I recommend that you read the documentation carefully if you're going to be trying to do that yourself)

4.1 More bagging in caret, Custom Bagging Example

You basically are going to take your predictor variable and put it into one data frame. (So I'm going to make the predictors be a data frame that contains the ozone data). Then you have your outcome variable (Here's it's going to be just the temperature variable from the data set). And I pass this to the bag function in caret package. So I tell it:

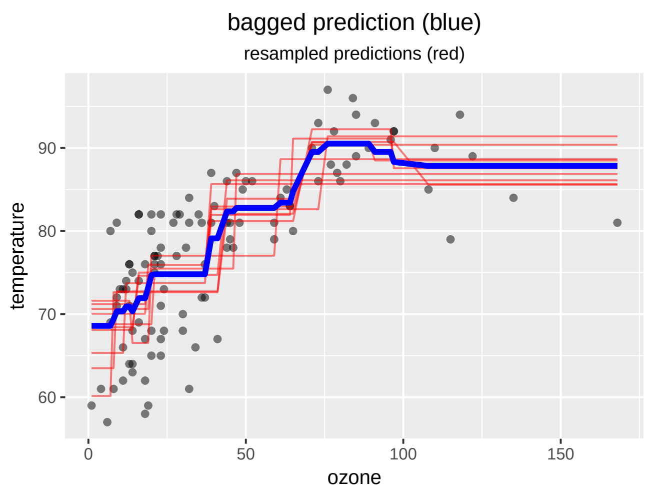
- I want to use the predictors from that data frame,
- this is my outcome,
- this is the number of replications with the number of sub samples I'd like to take from the data set.
- And then bagControl tells me something about how I'm going to fit the model.
  - So fit is the function that's going to be applied to fit the model every time. This could be a call to the `train` function in the caret package.
  - Predict is a the way that given a particular model fit, that we'll be able to predict new values. So this could be, for example, a call to the predict function from a trained model.

- And then aggregate is the way that we'll put the predictions together. So for example it could average the predictions across all the different replicated samples.

```r
set.seed(1357)

predictors = data.frame(ozone=ozone$ozone)
temperature = ozone$temperature
treebag <- bag(predictors, temperature, B = 10,
                bagControl = bagControl(fit = ctreeBag$fit,
                                        predict = ctreeBag$pred,
                                        aggregate = ctreeBag$aggregate))
```

We plot the resulting predictions:

```r
# # VOriginale (jleek)
# plot(ozone$ozone,temperature,col='lightgrey',pch=19)
# # erreur probable
# points(ozone$ozone,predict(treebag$fits[[1]]$fit,predictors),pch=19,col="red")
# points(ozone$ozone,predict(treebag,predictors),pch=19,col="blue")

# # BFC with ggplot2
# # graph basis layer
g <- ggplot() +
        geom_point(data = ozone, aes(ozone, temperature), alpha = 0.5)

# individual resamples
for (i in 1:10){
df2 <- data.frame(ozone = ozone$ozone,
                  y = predict(treebag$fits[[i]]$fit, # !!le nom "y" est imposé !!
                              predictors))
g <- g + geom_line(data = df2,
                   aes(ozone, y), color = "red", alpha = 0.5)

}
# # last layer (bagged prediction)
g <- g + geom_line(data = data.frame(ozone = ozone$ozone,
                                     final = predict(treebag,predictors)),
                   aes(ozone, final), color="blue", size = 1.5)
# # final touch
g <- g + labs(title = "bagged prediction (blue)",
              subtitle ="resampled predictions (red)") +
        theme(plot.title = element_text(hjust = 0.5),
              plot.subtitle = element_text(hjust = 0.5))

g
```



If you look at this custom bag version of the conditional regression trees, you can see that it gets some of the benefit that I was showing you in the previous slide with bag loess. So the idea here is I'm plotting ozone again on the x-axis versus temperature on the y-axis. The little grey dots represent actual observed values. The red dots represent the fit from a single conditional regression tree. And so you can see that for example, it doesn't capture the trend that's going on down here very well, the red line is just flat, even though there appears to be a trend upward in the data points here. But when I average over ten different bagged model model fits with these conditional regression trees. I see that there's an increase here in the values in the blue fit, which is the fit from the bagged regression.

http://www.inside-r.org/packages/cran/caret/docs/nbBag

4.2 Parts of bagging

We're going to look a little bit at those different parts of the bagging function. In this particular case I'm using the ctreeBag function, which you can look at in, if you've loaded the caret package in R.

4.2.1 `fit`

The fit part takes the data frame and the outcome that we've passed at, and it basically uses the `ctree` function to train a tree, (conditional regression tree) on the data set. This is the last command (the `ctree` command). So it returns this model fit from the `ctree` function.

```
ctreeBag$fit
```

```
## function (x, y, ...)
## {
##     loadNamespace("party")
##     data <- as.data.frame(x)
##     data$y <- y
##     party::ctree(y ~ ., data = data)
## }
## <environment: namespace:caret>
```

4.2.2 `predict`

The prediction takes in the object from the `ctree` model fit, and a new data set x, and it's going to get a new prediction. So what you can see here is it basically calculates each time the *tree response* or the outcome from the object and the new data. It then calculates this probability matrix and returns either the actually the observed levels that it predicts or it actually re, just returns the response, the predicted response from the variable.

```
ctreeBag$pred
```

```
## function (object, x)
## {
##     if (!is.data.frame(x))
##         x <- as.data.frame(x)
##     obsLevels <- levels(object@data@get("response")[, 1])
##     if (!is.null(obsLevels)) {
##         rawProbs <- party::treeresponse(object, x)
##         probMatrix <- matrix(unlist(rawProbs), ncol = length(obsLevels),
##             byrow = TRUE)
##         out <- data.frame(probMatrix)
##         colnames(out) <- obsLevels
##         rownames(out) <- NULL
##     }
##     else out <- unlist(party::treeresponse(object, x))
##     out
## }
## <environment: namespace:caret>
```

4.2.3 aggregate

The aggregation then takes those values and averages them together, or puts them together in some way.

So here what this is doing: is it's basically getting the prediction from every single one of these model fits, so that' s across a large number of observations. And then it binds them together into one data matrix by with each row being equal to the prediction from one of the model predictions. And then it takes the median at every value. So in other words it takes the median prediction from each of the different model fits across all the bootstrap samples

```
ctreeBag$aggregate
```

```
## function (x, type = "class")
## {
##     if (is.matrix(x[[1]]) | is.data.frame(x[[1]])) {
##         pooled <- x[[1]] & NA
##         classes <- colnames(pooled)
##         for (i in 1:ncol(pooled)) {
##             tmp <- lapply(x, function(y, col) y[, col], col = i)
##             tmp <- do.call("rbind", tmp)
##             pooled[, i] <- apply(tmp, 2, median)
##         }
##         if (type == "class") {
##             out <- factor(classes[apply(pooled, 1, which.max)],
##                 levels = classes)
##         }
##         else out <- as.data.frame(pooled)
##     }
##     else {
##         x <- matrix(unlist(x), ncol = length(x))
##         out <- apply(x, 1, median)
##     }
##     out
## }
## <environment: namespace:caret>
```

5 Notes and further resources

Notes:

- remember that the basic idea is to basically resample your data, refit your nonlinear model, then average those model fits together over resamples to get a smoother model fit than you would've got from any individual fit on its own
- Bagging is most useful for nonlinear models
- Often used with trees - an extension is random forests
- Several models use bagging in caret's *train* function

Further resources:

- Bagging
- Bagging and boosting
- Elements of Statistical Learning