Random forest computing time in R

I am using the party package in R with 10,000 rows and 34 features, and some factor features have more than 300 levels. The computing time is too long. (It has taken 3 hours so far and it hasn't finished yet.)

I want to know what elements have a big effect on the computing time of a random forest. Is it having factors with too many levels? Are there any optimized methods to improve the RF computing time?

r random-forest

edited Sep 16 '12 at 14:36
gung
56.7k 15 123 240

asked Sep 16 '12 at 6:18
Chenghao Liu
131 1 5

4 Answers

The overall complexity of RF is something like $\text{ntree} \cdot \text{mtry} \cdot (\#\text{ objects}) \log(\#\text{ objects})$; if you want to speed your computations up, you can try the following:

1. Use `randomForest` instead of `party`, or, even better, `ranger` or `Rborist` (although both are not yet battle-tested).
2. Don't use formula, i.e. call `randomForest(predictors,decision)` instead of `randomForest(decision~.,data=input)`.
3. Use `do.trace` argument to see the OOB error in real-time; this way you may detect that you can lower `ntree`.
4. About factors; RF (and all tree methods) try to find an optimal subset of levels thus scanning $2^{(\#\text{ of levels}-1)}$ possibilities; to this end it is rather naive this factor can give you so much information -- not to mention that randomForest won't eat factors with more than 32 levels. Maybe you can simply treat it as an ordered one (and thus equivalent to a normal, numeric variable for RF) or cluster it in some groups, splitting this one attribute into several?
5. Check if your computer haven't run out of RAM and it is using swap space. If so, buy a bigger computer.
6. Finally, you can extract some random subset of objects and make some initial experiments on this.

edited Sep 8 at 13:48
user777
8,658 3 17 42

answered Sep 16 '12 at 11:54
mbq ♦
15.8k 7 44 93

1 Great answer @mbq! +1. – Michael Chernick Sep 16 '12 at 14:47

1 Thank you,I learn a lot from your answer and did a test as you said,besides, why the second suggestion works? – Chenghao Liu Sep 24 '12 at 17:55

2 @ChenghaoLiu Formulas were designed for small yet complex liner model frames, and thus they are inefficient when copying the set becomes expensive. – mbq ♦ Sep 24 '12 at 21:47

Why does calling randomForest(predictors, decision) reducing running time? – Andy Blankertz Sep 27 '14 at 1:00

What's mtry? – NaN Sep 8 at 11:53

I would suggest a couple of links:

1) Shrink number of levels of a factor variable is a link to a question on `stackoverflow` to deal with a similar issue while using the `randomForest` package. Specifically it deals with using only the most frequently occurring levels and assigning a new level to all other, less frequently occurring levels.

The idea for it came from here: 2009 KDD Cup Slow Challenge. The data for this competition had

lots of factors with lots of levels and it discusses some of the methods they used to pare the data down from 50,000 rows by 15,000 columns to run on a 2-core/2GB RAM laptop.

My last suggestion would be to look at running the problem, as suggested above, in parallel on a hi-CPU Amazon EC2 instance.

answered Sep 28 '12 at 19:39

screechOwl

674 9 18

There's no **2)**. You should provide the important part of the page instead of relying entirely on the link. – A.L Jan 20 at 22:36

I love how those EC instances run. Wow are they nice. I think the virtualized hardware is better than the real-thing. – EngrStudent Sep 9 at 14:19

Because randomForest is a collection of independent carts trained upon a random subset of features and records it lends itself to parallelization. The combine() function in the randomForest package will stitch together independently trained forests. Here is a toy example. As @mpq 's answer above states you should not use the formula notation but pass in a dataframe/matrix of variables and a vector of outcomes. I shameless lifted these from the docs.

```
library("doMC")
library("randomForest")
data(iris)

registerDoMC(4) #number of cores on the machine
darkAndScaryForest <- foreach(y=seq(10), .combine=combine ) %dopar% {
   set.seed(y) # not really needed
   rf <- randomForest(Species ~ ., iris, ntree=50, norm.votes=FALSE)
}
```

I passed the randomForest combine function to the similarly named .combine parameter(which controls the function on the output of the loop. The down side is you get no OOB error rate or more tragically variable importance.

Hope it helps.

Edit:

After rereading the post i realize that i talk nothing about the 34+ factor issue. A wholey un-thought out answer could be to represent them as binary variables. that is each factor a column that is encoded 0/1 -level factor about its presence/non-presence. By doing some variable selection on unimportant factors and removing them you could keep you feature space from growing too too large. Just a thought.

edited Sep 17 '12 at 17:04

answered Sep 17 '12 at 1:14

jdennison

161 3

Welcome to the site, @jdennison. This looks like a really nice contribution (although I really don't know too much about RFs & nothing about parallel computing). One note, the ordering of the answers can fluctuate over time, so it's best not to refer to "the answer above", but rather 'the answer by \@so-and-so' instead. – gung Sep 17 '12 at 16:35

good point. thanks – jdennison Sep 17 '12 at 17:02

Sorry for answer you late.I read your blog ,great work – Chenghao Liu Sep 24 '12 at 18:08

I can't speak to the speed of specific algorithms in R but it should be obvious what is causing long computing time. For each tree at each branch CART is looking form the best binary split. So for each of the 34 features it most look at the splits given by each of the levels of the variables. Multiply the run time for each split in a tree by the number of branches in the tree and then multiple that by the number of trees in the forest and you have a long running time. Who knows? Maybe even with a fast computer this could take years to finish?

The best way to speed things up I think would be to lump some of the levels together so that each variable is down to maybe 3 to 5 levels instead of as many as 300. Of course this depends on being able to do this without losing important information in your data.

After that maybe you could look to see if there is some clever algorithm that can speed up the search time for splitting at each node of the individual trees. it could be that at a particular tree the

split search is a repeat of a search already done for a previous tree. So if you can save the solutions of the previous split decisions and identify when you are repeating maybe that strategy could save a little on computing time.

answered Sep 16 '12 at 11:39

Michael Chernick
24k 2 27 64

Thank you again ,i totally agree with you.And I try to reduce the levels number with a fake dummy method.For example,I replace a predictor with 600 levels with 4 predictors(as 600<5^4)After this transformation,I can run random forest algorithm.However,the RMSE result is strange,I will open two other question about how to reduce the level of factor feature and what's the relationship between 10-fold CV RMSE and test set RMSE score? – Chenghao Liu Sep 24 '12 at 16:33