Suggestions for speeding up Random Forests

I'm doing some work with the `randomForest` package and while it works well, it can be time-consuming. Any one have any suggestions for speeding things up? I'm using a Windows 7 box w/ a dual core AMD chip. I know about R not being multi- thread/processor, but was curious if any of the parallel packages (`rmpi` , `snow` , `snowfall` , etc.) worked for `randomForest` stuff. Thanks.

EDIT:

I'm using rF for some classification work (0's and 1's). The data has about 8-12 variable columns and the training set is a sample of 10k lines, so it's decent size but not crazy. I'm running 500 trees and an mtry of 2, 3, or 4.

EDIT 2: Here's some output:

```
> head(t22)
  Id Fail     CCUse Age S-TFail         DR MonInc #OpenLines L-TFail RE M-TFail Dep
1  1    1 0.7661266  45       2 0.80298213   9120         13       0  6       0   2
2  2    0 0.9571510  40       0 0.12187620   2600          4       0  0       0   1
3  3    0 0.6581801  38       1 0.08511338   3042          2       1  0       0   0
4  4    0 0.2338098  30       0 0.03604968   3300          5       0  0       0   0
5  5    0 0.9072394  49       1 0.02492570  63588          7       0  1       0   0
6  6    0 0.2131787  74       0 0.37560697   3500          3       0  1       0   1
> ptm <- proc.time()
>
> RF<- randomForest(t22[,-c(1,2,7,12)],t22$Fail
+                   ,sampsize=c(10000),do.trace=F,importance=TRUE,ntree=500,,forest=TRUE)
Warning message:
In randomForest.default(t22[, -c(1, 2, 7, 12)], t22$Fail, sampsize = c(10000),  :
  The response has five or fewer unique values.  Are you sure you want to do regression?
> proc.time() - ptm
   user  system elapsed
 437.30    0.86  450.97
>
```

r random-forest

edited Nov 8 '11 at 20:57
rcs
28.9k ● 8 ● 98 ● 108

asked Oct 20 '11 at 1:46
screechOwl
5,506 ● 15 ● 75 ● 133

2 Can you describe your data and modeling in a bit more detail? That will help a bit in making suggestions on how to address improvements. – Iterator Oct 20 '11 at 1:54

You can often get rf models of equivalent performance only drawing a subsample of your data for each tree (`sampsize`), rather than a resample of size 10k for each tree. – joran Oct 20 '11 at 2:31

3 I can run a rf model on 10k observations and 500 trees with 8 variables *without* using the `sampsize` argument *and* using the formula interface in under 15 seconds on my year old macbook air. I suspect there's something else going on. – joran Oct 20 '11 at 2:47

1 It would really help if you posted code to create sample data and the code you're actually running (i.e. a reproducible example). – Joshua Ulrich Oct 20 '11 at 6:11

2 Are those 12 variables stored as factors in the input data? If they are, are there many levels? As @joran says, don't use the formula interface; that is for convenience not speed. Are you sure you are not maxing out the RAM on your machine during fitting (i.e. are your discs swapping memory into/out of RAM)? 10K rows doesn;t seem like very much to me. – Gavin Simpson Oct 20 '11 at 9:07

4 Answers

The manual of the `foreach` package has a section on Parallel Random Forests (Using The foreach Package, Section 5.1):

```
> library("foreach")
> library("doSNOW")
> registerDoSNOW(makeCluster(4, type="SOCK"))

> x <- matrix(runif(500), 100)
> y <- gl(2, 50)

> rf <- foreach(ntree = rep(250, 4), .combine = combine, .packages = "randomForest")
%dopar%
+    randomForest(x, y, ntree = ntree)
> rf
Call:
randomForest(x = x, y = y, ntree = ntree)
```

```
Type of random forest: classification
Number of trees: 1000
```

If we want want to create a random forest model with a 1000 trees, and our computer has four cores, we can split up the problem into four pieces by executing the `randomForest` function four times, with the `ntree` argument set to 250. Of course, we have to combine the resulting `randomForest` objects, but the `randomForest` package comes with a function called `combine`.

answered Oct 20 '11 at 6:17

rcs
28.9k ● 8 ● 98 ● 108

2 Unfortunately we loose some outputs when using parallel computations, such as the R² – Stéphane Laurent Nov 17 '14 at 13:13

Stephane, Why does this happen? – robertevansanders Jun 11 at 15:03

I run parallel RF and when I call print(rf) it doesn't show confusion matrix anymore. is there way to get confusion matrix? – hmi2015 Jul 20 at 20:35

There are two 'out of the box' options that address this problem. First, the caret package contains a method 'parRF' that handles this elegantly. I commonly use this with 16 cores to great effect. The randomShrubbery package also takes advantages of multiple cores for RF on Revolution R.

answered Jul 2 '12 at 22:23

Brent
111 ● 1 ● 3

Hi Brent, in Revolution R does randomShrubbery take .xdf as input dataset or does have to be a dataframe. I have a 10GB .xdf training file. – user1509107 Sep 10 '12 at 22:10

It's part of the train function though, so you won't find it with ?parRF, instead do ?train and read up. This answer is the better way forward really, cheers Brent – N. McA. Dec 26 '12 at 16:36

Why don't you use an already parallelized and optimized implementation of Random Forest? Have a look to SPRINT using MPI. http://www.r-sprint.org/

answered Apr 15 '13 at 13:14

Manolete
1,021 ● 2 ● 19 ● 46

Is there any particular reason why you're not using Python (namely the scikit-learn and multiprocessing modules) to implement this? Using joblib, I've trained random forests on datasets of similar size in a fraction of the time it takes in R. Even without multiprocessing, random forests are significantly faster in Python. Here's a quick example of training a RF classifier and cross validating in Python. You can also easily extract feature importances and visualize the trees.

```
import numpy as np
from sklearn.metrics import *
from sklearn.cross_validation import StratifiedKFold
from sklearn.ensemble import RandomForestClassifier

#assuming that you have read in data with headers
#first column corresponds to response variable
y = data[1:, 0].astype(np.float)
X = data[1:, 1:].astype(np.float)

cm = np.array([[0, 0], [0, 0]])
precision = np.array([])
accuracy = np.array([])
sensitivity = np.array([])
f1 = np.array([])
matthews = np.array([])

rf = RandomForestClassifier(n_estimators=100, max_features = 5, n_jobs = 2)

#divide dataset into 5 "folds", where classes are equally balanced in each fold
cv = StratifiedKFold(y, n_folds = 5)
for i, (train, test) in enumerate(cv):
        classes = rf.fit(X[train], y[train]).predict(X[test])
        precision = np.append(precision, (precision_score(y[test], classes)))
        accuracy = np.append(accuracy, (accuracy_score(y[test], classes)))
        sensitivity = np.append(sensitivity, (recall_score(y[test], classes)))
```

```
        f1 = np.append(f1, (f1_score(y[test], classes)))
        matthews = np.append(matthews, (matthews_corrcoef(y[test], classes)))
        cm = np.add(cm, (confusion_matrix(y[test], classes)))

print("Accuracy: %0.2f (+/- %0.2f)" % (accuracy.mean(), accuracy.std() * 2))
print("Precision: %0.2f (+/- %0.2f)" % (precision.mean(), precision.std() * 2))
print("Sensitivity: %0.2f (+/- %0.2f)" % (sensitivity.mean(), sensitivity.std() * 2))
print("F1: %0.2f (+/- %0.2f)" % (f1.mean(), f1.std() * 2))
print("Matthews: %0.2f (+/- %0.2f)" % (matthews.mean(), matthews.std() * 2))
print(cm)
```

edited Sep 24 '13 at 8:12

answered Sep 24 '13 at 7:39

eagle34

531 ● 2 ● 13

1 +1 Not every screwdriver fits the screw. – Brandon Bertelsen Sep 24 '13 at 7:56