setting values for ntree and mtry for random forest regression model

Asked 7 years, 2 months ago Active 1 year, 7 months ago Viewed 71k times

38

17

I'm using R package `randomForest` to do a regression on some biological data. My training data size is `38772 X 201`.

I just wondered---what would be a good value for the number of trees `ntree` and the number of variable per level `mtry`? Is there an approximate formula to find such parameter values?

Each row in my input data is a 200 character representing the amino acid sequence, and I want to build a regression model to use such sequence in order to predict the distances between the proteins.

r | statistics | machine-learning | regression | random-forest

edited Jul 7 '18 at 1:08

Kim

2,226 ● 2 ● 16 ● 37

asked Dec 19 '12 at 16:09

DOSMarter

1,157 ● 3 ● 14 ● 24

6 This sounds more like a job for stats.stackexchange.com – MattLBeck Dec 19 '12 at 16:20

1 I agree, while a fine question, it does not belong here. Also, maybe try to make it more readable. – PascalVKooten Dec 19 '12 at 16:21

In the reality of building random forests from large datasets, ntrees is often a compromise between runtime and precision. – blmoore Dec 19 '12 at 16:29

37

The default for mtry is quite sensible so there is not really a need to muck with it. There is a function `tuneRF` for optimizing this parameter. However, be aware that it may cause bias.

There is no optimization for the number of bootstrap replicates. I often start with `ntree=501` and then plot the random forest object. This will show you the error convergence based on the OOB error. You want enough trees to stabilize the error but not so many that you over correlate the ensemble, which leads to overfit.

Here is the caveat: variable interactions stabilize at a slower rate than error so, if you have a large number of independent variables you need more replicates. I would keep the ntree an odd number so ties can be broken.

For the dimensions of you problem I would start `ntree=1501`. I would also recommended looking onto one of the published variable selection approaches to reduce the number of your independent variables.

edited Dec 19 '12 at 16:31

joran
146k ● 24 ● 363 ● 411

answered Dec 19 '12 at 16:29

Jeffrey Evans
2,087 ● 9 ● 16

Hope you don't mind I cleaned this up a tiny bit just to make it more readable. – joran Dec 19 '12 at 16:31

Regarding the last point of @Jeffrey Evans answer, I would suggest the use of the `rfcv` (explained also here stats.stackexchange.com/questions/112556/…). I found it helpful for removing the least important independent variables. – Nemesi Jan 24 '17 at 11:23

18

The short answer is no.

The `randomForest` function of course has default values for both `ntree` and `mtry`. The default for `mtry` is often (but not always) sensible, while generally people will want to increase `ntree` from it's default of 500 quite a

generally people will want to increase `ntree` from it's default of 500 quite a bit.

The "correct" value for `ntree` generally isn't much of a concern, as it will be quite apparent with a little tinkering that the predictions from the model won't change much after a certain number of trees.

You can spend (read: waste) a lot of time tinkering with things like `mtry` (and `sampsize` and `maxnodes` and `nodesize` etc.), probably to some benefit, but in my experience not a lot. However, every data set will be different. Sometimes you may see a big difference, sometimes none at all.

The **caret** package has a very general function `train` that allows you to do a simple grid search over parameter values like `mtry` for a wide variety of models. My only caution would be that doing this with fairly large data sets is likely to get time consuming fairly quickly, so watch out for that.

Also, somehow I forgot that the **ranfomForest** package itself has a `tuneRF` function that is specifically for searching for the "optimal" value for `mtry`.

answered Dec 19 '12 at 16:24

joran
146k ● 24 ● 363 ● 411

1 FYI, I have talked with Adele Cutler regarding optimization of RF parameters and she indicated that the stepwise procedures that "tuneRF" and "train" use leads to bias. Also, as indicated in my post, it is possible to overfit RF by over correlating the ensemble. So, there is a balance in the number of bootstrap replicates between error convergence, variable interaction and avoiding overfit. – Jeffrey Evans Dec 21 '12 at 17:16

Could this paper help ? Limiting the Number of Trees in Random Forests

5

> Abstract. The aim of this paper is to propose a simple procedure that a priori determines a minimum number of classifiers to combine in order to obtain a prediction accuracy level similar to the one obtained with the
>
> combination of larger ensembles. The procedure is based on the McNemar non-parametric test of significance. Knowing a priori the minimum size of the classifier ensemble giving the best prediction

minimum size of the classifier ensemble giving the best prediction accuracy, constitutes a gain for time and memory costs especially for huge data bases and real-time applications. Here we applied this procedure to four multiple classifier systems with C4.5 decision tree (Breiman's Bagging, Ho's Random subspaces, their combination we labeled 'Bagfs', and Breiman's Random forests) and five large benchmark data bases. It is worth noticing that the proposed procedure may easily be extended to other base learning algorithms than a decision tree as well. The experimental results showed that it is possible to limit significantly the number of trees. We also showed that the minimum number of trees required for obtaining the best prediction accuracy may vary from one classifier combination method to another

They never use more than 200 trees.

Table 2. Experimental Results. Performance in terms of the prediction accuracy (%), minimum recommended number of trees with respect to McNemar in bold and in brackets.

	C4.5	Bag	MFS	Bagfs	Bagrf
ringnorm	89.3	94.0	97.7	98.4	96.2
		(10)	**(30)**	**(50)**	**(60)**
satimage	84.0	88.2	89.8	89.6	89.1
		(20)	**(70)**	**(50)**	**(50)**
image	93.6	96.1	96.2	96.0	94.5
		(10)	**(50)**	**(10)**	**(40)**
DNA	86.5	89.5	90.0	91.5	89.8
		(20)	**(20)**	**(30)**	**(130)**
letter	81.4	88.6	90.4	91.6	89.0
		(90)	**(50)**	**(110)**	**(200)**

edited Mar 1 '16 at 11:15

answered Mar 1 '16 at 10:27

bastaPasta

669 ● 6 ● 13

One nice trick that I use is to initially start with first taking square root of the number of predictors and plug that value for "mtry". It is usually around the

2

same value that tunerf funtion in random forest would pick.

answered Jul 7 '17 at 16:09

KHAN irfan
243 ● 1 ● 8

1

I use the code below to check for accuracy as I play around with ntree and mtry (change the parameters):

```
results_df <- data.frame(matrix(ncol = 8))
colnames(results_df)[1]="No. of trees"
colnames(results_df)[2]="No. of variables"
colnames(results_df)[3]="Dev_AUC"
colnames(results_df)[4]="Dev_Hit_rate"
colnames(results_df)[5]="Dev_Coverage_rate"
colnames(results_df)[6]="Val_AUC"
colnames(results_df)[7]="Val_Hit_rate"
colnames(results_df)[8]="Val_Coverage_rate"


trees = c(50,100,150,250)
variables = c(8,10,15,20)

for(i in 1:length(trees))
{
  ntree = trees[i]
  for(j in 1:length(variables))
  {
    mtry = variables[j]
    rf<-randomForest(x,y,ntree=ntree,mtry=mtry)
    pred<-as.data.frame(predict(rf,type="class"))
    class_rf<-cbind(dev$Target,pred)

    colnames(class_rf)[1]<-"actual_values"
    colnames(class_rf)[2]<-"predicted_values"
    dev_hit_rate = nrow(subset(class_rf, actual_values
==1&predicted_values==1))/nrow(subset(class_rf, predicted_values ==1))
    dev_coverage_rate = nrow(subset(class_rf, actual_values
==1&predicted_values==1))/nrow(subset(class_rf, actual_values ==1))


    pred_prob<-as.data.frame(predict(rf,type="prob"))
    prob_rf<-cbind(dev$Target,pred_prob)
    colnames(prob_rf)[1]<-"target"
    colnames(prob_rf)[2]<-"prob_0"
    colnames(prob_rf)[3]<-"prob_1"
```

```
    colnames(prob_rf)[3]<-"prob_1"
    pred<-prediction(prob_rf$prob_1,prob_rf$target)
    auc <- performance(pred,"auc")
    dev_auc<-as.numeric(auc@y.values)

    pred<-as.data.frame(predict(rf,val,type="class"))
    class_rf<-cbind(val$Target,pred)

    colnames(class_rf)[1]<-"actual_values"
    colnames(class_rf)[2]<-"predicted_values"
    val_hit_rate = nrow(subset(class_rf, actual_values
==1&predicted_values==1))/nrow(subset(class_rf, predicted_values ==1))
    val_coverage_rate = nrow(subset(class_rf, actual_values
==1&predicted_values==1))/nrow(subset(class_rf, actual_values ==1))

    pred_prob<-as.data.frame(predict(rf,val,type="prob"))
    prob_rf<-cbind(val$Target,pred_prob)
    colnames(prob_rf)[1]<-"target"
    colnames(prob_rf)[2]<-"prob_0"
    colnames(prob_rf)[3]<-"prob_1"
    pred<-prediction(prob_rf$prob_1,prob_rf$target)
    auc <- performance(pred,"auc")
    val_auc<-as.numeric(auc@y.values)
    results_df =
rbind(results_df,c(ntree,mtry,dev_auc,dev_hit_rate,dev_coverage_rate,val_auc

  }
}
```

answered Apr 27 '17 at 14:48

Hillary Murefu
21 ● 1