Random Forest: To cross-validate or Not to cross-validate

RESOLVED

crossvalidation × randomForest × +

Add Tag

Jennifer Li Signature Track · 3 days ago

I was under the impression from the lectures that it's very important to perform cross-validation when using random forests due to the possibility of overfitting. At the very least, one should have an untouched test set to get a sense of the out-of-sample error.

However, I found the following statement in this link: (http://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm#ooberr)-- which incidentally was part of Quiz 3:

"In random forests, there is **no need for cross-validation or a separate test set** to get an unbiased estimate of the test set error. It is estimated internally, during the run, as follows:

Each tree is constructed using a different bootstrap sample from the original data. About one-third of the cases are left out of the bootstrap sample and not used in the construction of the kth tree.

Put each case left out in the construction of the kth tree down the kth tree to get a classification. In this way, a test set classification is obtained for each case in about one-third of the trees. At the end of the run, take j to be the class that got most of the votes every time case n was oob. The proportion of times that j is not equal to the true class of n averaged over all cases is the oob error estimate. This has proven to be unbiased in many tests."

So which one is it? Should we cross-validate or not cross-validate?

1 · flag

Brett Ryan Condron Signature Track · 3 days ago

I think you'll find that using "oob" in your trControl call will give you a more accurate model much more quickly than using the default (which is bootstrapping) or changing to cv, at least assuming you're using some variant of a Random Forest.

1 · flag

Jennifer Li Signature Track · 3 days ago

Thanks, Brett!

0 · flag

+ Comment

Anonymous · a day ago

During the last few days I read several discussions about using OOB error estimates instead of CV for random forests... people seem to have different opinions on that matter. I use OOB for training & model selection because it's just so much faster than CV. But I do not report the OOB error estimate obtained during training as out of sample error... I calculate the out of sample error using predictions on a test set.

I think since caret does parameter tuning (of the mtry parameter) we have to calculate the out of sample error on a test set anyway. The mtry parameter is selected to best fit the training data... so the error we get from training might be slightly biased / too optimistic, no matter if we use CV or OOB.

0 · flag

+ Comment

Jennifer Li Signature Track · a day ago

Hi guys, I have a new problem. I tried using both OOB and cross-validation to determine my error estimates.

USING OOB

```
set.seed(32343) modFit = train(as.factor(trainingdata$classe[inTrain])~ . , data = Train, method = 'rf', trControl = trainControl(method='oob'))
modFit$results
```

```
Accuracy     Kappa mtry

1 0.9789619 0.9733745    2

2 0.9759773 0.9695967   19

3 0.9692800 0.9611198   37
```

Here, the estimated accuracy is about 97.8%. As expected, this is slightly larger (optimistic bias) than the accuracy for the untouched dataset, shown below (97.6%).

```
# making predictions

predictions = predict(modFit, Test)

table(predictions,trainingdata$classe[-inTrain])

accuracy_true = sum(predictions ==trainingdata$classe[-inTrain])/length(predictions)
```

```
accuracy_true

[1] 0.9760408
```

USING CV

```
# creating cross-validation object

ctrl <- trainControl(method = "cv", number  = 2) # 2-fold CV  (to make RF run a bit faster)
```

```
# building RF model

modFit = train(as.factor(trainingdata$classe[inTrain])~ . , data = Train, method = 'rf',
trControl = ctrl)
modFit$results

  mtry  Accuracy     Kappa    AccuracySD     KappaSD

1    2 0.9576326 0.9463668 0.0020633463 0.002605778

2   19 0.9509354 0.9378804 0.0010345435 0.001300253

3   37 0.9438742 0.9289315 0.0001087273 0.000131724
```

As you can see, with CV, the accuracy estimate has gotten noticeably worse, to about 95%. In fact, it is now WORSE than the accuracy for the untouched dataset (97.5%):

```
# making predictions

predictions = predict(modFit, Test)

table(predictions,trainingdata$classe[-inTrain])

accuracy_true = sum(predictions ==trainingdata$classe[-inTrain])/length(predictions)
accuracy_true

[1] 0.975531
```

How can this be? What am I doing wrong?? P.S. I tried this a few times with different random seeds, to no avail.

0 · flag

Anonymous · 13 hours ago

Hi Jennifer,

could you try to increase the number of CV folds to maybe 10? With 2 folds it gets to train on only 50% of the data... with 10 folds it'd get to train on 90% of the data. I guess training on less data might result in parameter tuning / model selection not working as well?

0 · flag

Jennifer Li Signature Track · 13 hours ago

You are absolutely right-- changing the CV folds to 3 (10 would break my poor computer, I think) changed the accuracy estimate to 96.7%. What I don't understand is why the accuracy estimate has a pessimistic bias. It seems that it should have an optimistic bias, seeing as how caret chooses the best model based on the CV accuracy estimate. Any thoughts?

0 · flag

Anonymous · 12 hours ago

I think the reason is that the final/best model is trained **after** the accuracies for different model parameters have been estimated.

Here's a nice graphic that explains how the train function works:

```
1  Define sets of model parameter values to evaluate
2  for each parameter set do
3  |  for each resampling iteration do
4  |  |  Hold–out specific samples
5  |  |  [Optional] Pre–process the data
6  |  |  Fit the model on the remainder
7  |  |  Predict the hold–out samples
8  |  end
9  |  Calculate the average performance across hold–out predictions
10 end
11 Determine the optimal parameter set
12 Fit the final model to all the training data using the optimal parameter set
```

(source: http://caret.r-forge.r-project.org/training.html)

The accuracy estimates that we get to see when looking at a model are calculated in line 9. With 2-fold CV they come from models that were trained on only 50% of the training data (line 6).

The final model is trained in line 12 using the best parameters and on 100% of the training data. Note that the train function cannot estimate the final model's accuracy... because it already spent 100% of the data for training of that model.

So with 2 fold CV there's quite a difference - we get to see accuracy estimates from models that were trained on 50% of the data, but we end up with a final model that was trained using 100% of the data. => A model trained with twice as much data will often perform better than one trained with less data.

0 · flag

Jennifer Li Signature Track · 10 hours ago

Ok. I get it now. Thanks!

0 · flag

+ Comment

New post

To ensure a positive and productive discussion, please read our forum posting policies before posting.

B | I | Link | <code> | Pic | Math | Edit: Rich | Preview

This thread is marked as resolved. Staff are no longer monitoring this thread. If the problem is not fixed, please start a new thread here to let staff know that there is still a problem.

Make this post anonymous to other students

Subscribe to this thread at the same time

Add post