FastML

Machine learning made easy

Impute missing values with Amelia

2014-05-08 16:14

One of the ways to deal with missing values in data is to impute them. We use Amelia R package on The Analytics Edge competition data. Since one typically gets many imputed sets, we bag them with good results. So good that it seems we would have won the contest if not for a bug in our code.

The competition

Much to our surprise, we ranked 17th out of almost 1700 competitors - from the public leaderboard score we expected to be in top 10%, barely. The contest turned out to be one with huge overfitting possibilities, and people overfitted badly - some preliminary leaders ended up down the middle of the pack, while we soared up the ranks.

But wait! There's more. When preparing this article, we discovered a bug - apparently we used only 1980 points for training:

```
points_in_test = 1980
train = data.iloc[:points_in_test,]     # should be [:-points_in_test,]
test = data.iloc[-points_in_test:,]
```

If not for this little bug, we would have won, apparently.

#	Δ5d	Team Name	Score	Entries	Last Submission UTC (Best - Last Submission)	
-		**Onward To Victory**	**0.78467**	-	**Mon, 12 May 2014 12:36:29**	Post-Deadline

Post-Deadline Entry
If you would have submitted this entry during the competition, you would have been around here on the leaderboard.

1	↑809	Ed53	0.78225	24	Mon, 05 May 2014 19:06:44 (-11.7h)

Imputing data

A common strategy found in the forums, besides using Support Vector Machines as a classifier, was to impute missing values with mice, as described in the class.

```
imputed_data = complete( mice( data ))
```

Imputing with *mice*, while straightforward, seemed very slow - no end in sight - so we turned to another R package: Amelia.

Amelia

Amelia is a program from a few Harvard folks. The package is named after Amelia Earhart, a famous American woman aviator who went missing over the ocean. Amelia is also a name of a birth defect of lacking one or more limbs.

The good thing about the software is that it works much faster than *mice*, partly due to employing multiple cores. It also has a nice PDF manual.

The bad is that it sometimes crashes with a segmentation fault. It doesn't like collinear columns, so you have to remove them. However, this wasn't an issue with this particular data set.

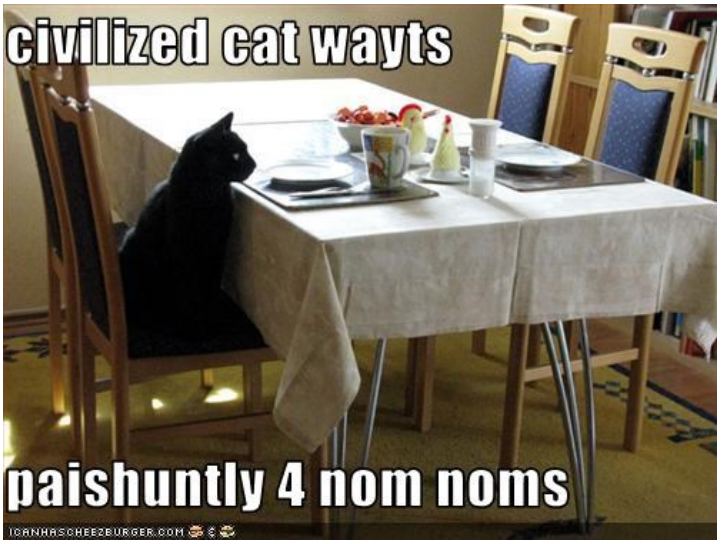
Column types

The main thing to do when running the algorithm is to specify types of columns:

```
noms = c( 'some', 'nominal', 'columns' )
ords = c( 'and', 'ordinal', 'if', 'you', 'need', 'them' )
idvars = c( 'these', 'will', 'be', 'ignored' )

a.out = amelia( data, noms = noms, ords = ords, idvars = idvars )
```

Nominal is another word for categorical. Ordinals you can skip, unless you specifically want them imputed as integers. And *idvars* will be preserved in the output but otherwise ignored - put your target variable among them. The software will treat all the other columns as real numbers.



Parallelism

By default, Amelia performs five runs, so you end up with five imputed sets. The reason for this is that for given data there are many possible imputations.

It makes sense to set the number of imputed datasets to be equal to [multiple of] the number of cores in your CPU, so that all cores work in parallel.

```
ncpus = 8
m = ncpus * 10      # 80 sets, see Bagging

a.out = amelia( data, m = m, ..., parallel = 'multicore', ncpus = ncpus )
```

Output

The object returned by the function holds a list of imputed sets. This is how to access the *i*-th:

```
a.out$imputations[[i]]
```

You can save the results like this:

```
a.out = amelia( ... )
write.amelia( a.out, file.stem = 'train_and_test_imp' )
```

The function above will write each set in a separate file. See the manual for other options.

Bagging

Having many similiar sets naturally leads to bagging: you train a model on each set and average the results. This strategy produced the best score for us, although SVM trained on data with missing values filled in a particular way was also quite good (final AUC = 0.77761).

The 0.785 result pictured above comes from bagging 96 imputed sets. The code is available at GitHub.

Post scriptum: an experiment with imputing *y*

In principle, it's possible to use imputation software to fill missing values for *y* in the test set. We tried that and scored around 0.6 AUC in validation without bagging - better than random, worse than normal supervised methods.

Posted by Zygmunt Z. 2014-05-08 16:14 Kaggle, code, software

« Converting categorical data into numbers with Pandas and Scikit-learn

Comments

Recent Posts

Twitter

Follow @fastml for notifications about new posts.

- Status updating...

Follow @fastml 946 followers

Also check out @fastml_extra for things related to machine learning and data science in general.

GitHub

Most articles come with some code. We push it to Github.

https://github.com/zygmuntz