Boehringer Ingelheim

Completed • $20,000 • 699 teams

Predicting a Biological Response

Fri 16 Mar 2012 – Fri 15 Jun 2012 (3 years ago)

How to do blending

Start Watching

1 vote

From results of different competitions, blending seems like an obvious way to go. I would like to learn how to blend. My idea comes from KDD Cup 2010 writeup by Toscher und Jahrer, and is roughly as follows:

- you have a bunch of classifiers (models)
- you take each of them and perform cross-validation on a training set
- for each classifier, collect predictions from each fold of CV. These predictions will be one column in a blender training set, B_train
- train each classifier on a full training set and get predictions for a test set. These predictions will be one column in a blender test set, B_test
- train a blender on B_train
- get predictions for B_test. Those are the end product

Here come the questions:

Is this correct?

How many classifiers do you need for blending?

Do you put some other data into B_train, or just CV predictions?

What classifier do you use as a blender (linear, NN...)?

#1 | Posted 3 years ago

Foxtrot

Permalink

0 votes

just FYI, at a datamining class I'm taking this semester, we called this tecnique 'classifier *stacking*'.

HTH,

#2 | Posted 3 years ago

kernc

1 vote

That all seems correct and reasonable. And indeed, it is often called stacking. You often get better search hits off "ensemble" instead of "blending".

As for how many classifiers? The sky is the limit. You can do the same algorithm with different hyperparameters to improve their performance. However, you generally get better ensembling by doing completely different algorithms (tree based, linear, svm, knn etc).

As for including other stuff into B_Train? Well that's the question isn't it. Well, that and what algorithm works best to do the stacking?

#3 | Posted 3 years ago


Shea Parkes

2 votes

I'm relatively new to Ensemble Methods as well, but here is my current take:

- How to do blending?

Stacking seems like a reasonable option, but simple things like voting or averaging classifier outputs can be useful as well.

However, take care when applying averaging that the scores will eventually gravitate towards 0.5 (Central Limit Theorum is the criminal here) which might have peculiar effects on this competition's metric (e.g. it might deteriorate LogLoss despite enhancing AUC)

(Some literature suggests that stacking only works when done using linear regression as the meta-model, I've seen no evidence to support that personally.)

- When to do blending?

When you have several competitive models that are fundamentally diverse, either because they come from different algorithms, different samples, different feature subsets, or any other form of diversity. This is what makes merging teams in predictive modeling competition such a good idea. And is probably the best way to divide work in teams. (i.e. by blending results at the very end without sharing methods or insights before then, to insure diversity of the models)

#4 | Posted 3 years ago | Edited 3 years ago


D33B

1 vote

Of minor note, I have seen empirical evidence that linear based model tend to work better in the meta-model role. There are still a large variety of types of linear models though.

And yes, diversity is key. I often struggle with making myself do the pre-processing in multiple ways. I can wrap my head around the algorithms needing to be different, but the data scrubbing is just so boring to do it multiple times in multiple ways...

#5 | Posted 3 years ago


Shea Parkes

1 vote

My apologies for bumping the topic, but the problem I am facing is that blending or stacking is easier on regression but complex in case of classification models especially when considering the prediction probabilities instead of straight predictions. In case of probabilities, we get a kind of 3D vector **#ofTrainSamples x probabilites_of_each_class x #ofClassifiers.** Now how do you train the metalevel classifier on this?

#6 | Posted 2 years ago


saurabh sharma

0 votes

saurabh sharma wrote:

> My apologies for bumping the topic, but the problem I am facing is that blending or stacking is easier on regression but complex in case of classification models especially when considering the prediction probabilities instead of straight predictions. In case of probabilities, we get a kind of 3D vector **#ofTrainSamples x probabilites_of_each_class x #ofClassifiers.** Now how do you train the metalevel classifier on this?

Same question here

#7 | Posted 2 months ago


SecondPlan

Reply You must be logged in to reply to this topic. **Log in »**

Start Watching « Back to forum