Meta Bagging Algorithm

Michael S. Kim

Meta Modeling

Tradeshift Text Classification's winning entry used a two stage model where random forest predictions are fed into XGBoost.

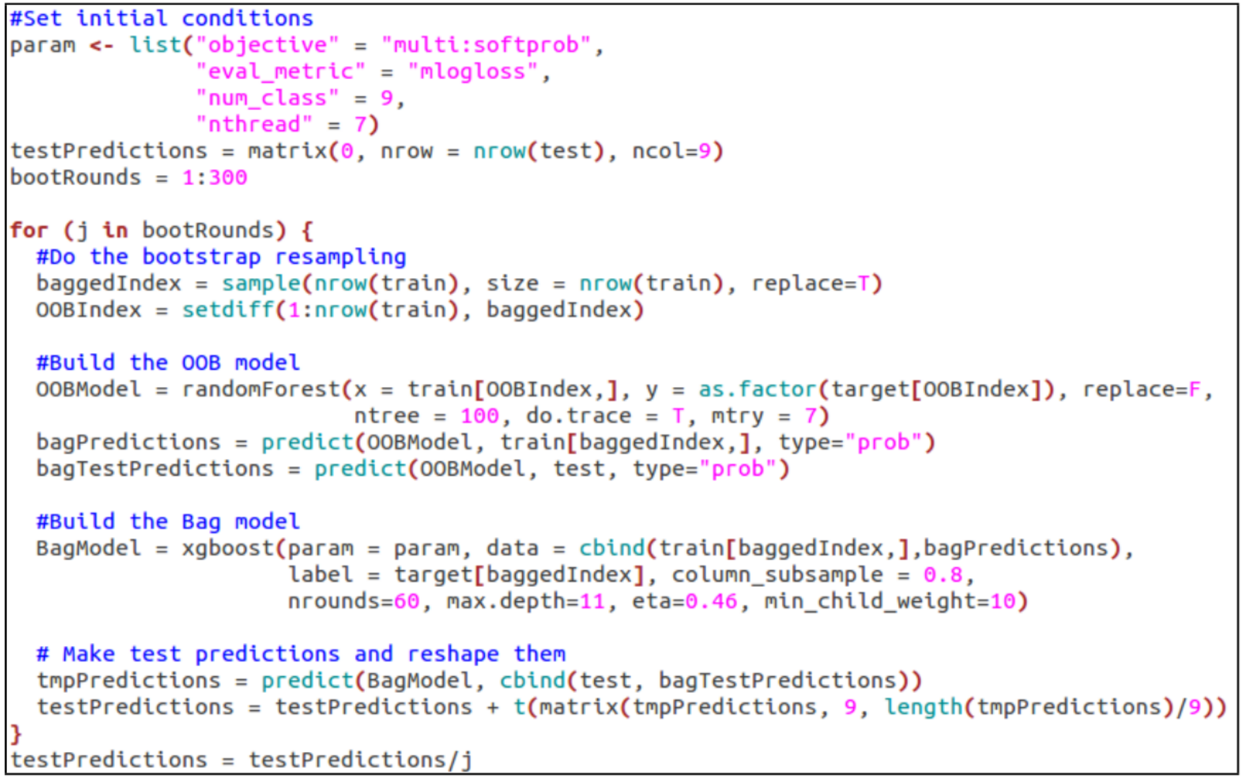
However, there was no explicit bagging. The ensembling of different meta models allowed some blending.

Explicit Bagging of Meta Models

When data is large as in the case of Tradeshift, bagging might be too computationally intensive.

However, when data is small like in the case of Otto or Microsoft, bagging is computationally feasible and gives performance gains.

R Code



```r
#Set initial conditions
param <- list("objective" = "multi:softprob",
              "eval_metric" = "mlogloss",
              "num_class" = 9,
              "nthread" = 7)
testPredictions = matrix(0, nrow = nrow(test), ncol=9)
bootRounds = 1:300

for (j in bootRounds) {
  #Do the bootstrap resampling
  baggedIndex = sample(nrow(train), size = nrow(train), replace=T)
  OOBIndex = setdiff(1:nrow(train), baggedIndex)

  #Build the OOB model
  OOBModel = randomForest(x = train[OOBIndex,], y = as.factor(target[OOBIndex]), replace=F,
                          ntree = 100, do.trace = T, mtry = 7)
  bagPredictions = predict(OOBModel, train[baggedIndex,], type="prob")
  bagTestPredictions = predict(OOBModel, test, type="prob")

  #Build the Bag model
  BagModel = xgboost(param = param, data = cbind(train[baggedIndex,],bagPredictions),
                     label = target[baggedIndex], column_subsample = 0.8,
                     nrounds=60, max.depth=11, eta=0.46, min_child_weight=10)

  # Make test predictions and reshape them
  tmpPredictions = predict(BagModel, cbind(test, bagTestPredictions))
  testPredictions = testPredictions + t(matrix(tmpPredictions, 9, length(tmpPredictions)/9))
}
testPredictions = testPredictions/j
```

Bagging 2 Stage Models

OOB data is used to train one or more base classifiers, and these predict into the bag data.

The bag data with predictions are then boosted.

Model Tuning

Hyper parameter tuning is not intuitive:

1. The best base models using OOB data are usually under trained (for instance a random forest with less than 500 trees).
2. The best meta model appears to be either XGBoost or a Neural Network (Lasagne in Otto). It should directly optimize the loss function.
3. Sometimes ensembling base models leads to gains (SVM, GBC, RF).

Used by Many Kagglers

This method has been used by many Kagglers either directly or indirectly and has been rediscovered independently as a strong approach.

The second place solution to Microsoft used the meta bagging with extra trees and XGBoost.

The method seems much stronger when bagging takes place relative to when there is no bagging.

Why does it work?

It seems to work since all data is used in every iteration of a bag. And more data is almost always better.

The loss function is always directly minimized when the meta model is XGBoost or a neural network.

The meta classifier does automated ensembling for you. The bagging may prevent overfitting.

Future Work

There is potential for theoretical work on why this approach seems to work better than almost all alternative approaches when features are held constant.

What are the limitations of this approach besides the computational cost (which can be parallelized)? Are there specific failure modes?