The Development of Model Ensembles Using Cross Validation for Open Source Data Challenges

Agustin Calatroni, Rebecca Z. Krouse, Petra LeBeau
Rho Federal Systems Division

Abstract

As data collection grows in size and complexity across a variety of industries, open source data challenges are becoming more widespread. We present our experience developing prediction models within the context of data challenges. With the goal of maximizing predictive performance, we explored ensemble learning methods to train our models. We demonstrate the use of these methods using R packages such as h2o[1] and h2oEnsemble[2] and cloud computing platforms. In order to obtain an approximation of our predictive ability prior to challenge submission, we developed wrapper code to perform cross validation on the H2O ensembles. We display our process for determining the expected level of performance of the trained model on external data sources.

1: Spencer Aiello, Tom Kraljevic, Petr Maj and with contributions from the H2O.ai team (2015). h2o: R Interface for H2O. R package version 3.8.1.3. https://CRAN.R-project.org/package=h2o

2: Erin LeDell (2016). h2oEnsemble: H2O Ensemble Learning. R package version 0.1.6. https://github.com/h2oai/h2o-3/tree/master/h2o-r/ensemble/h2oEnsemble-package

Background

Data competitions are a popular way for companies to incentivize analysts to interpret their large databases. Many data competitions are oriented toward predicting different targets of interest. In the typical predictive competition, a company divides their database into “training” and “test” datasets. The “training” dataset that contains full information on the target variable as well as many potential predictors. The test dataset is similar to the training set, except that the target is hidden from contest participants.

Competitors in the data challenge develop their models on the training dataset and submit predicted values for the test data target variable. Scoring is based on how closely the predictions match the true outcomes

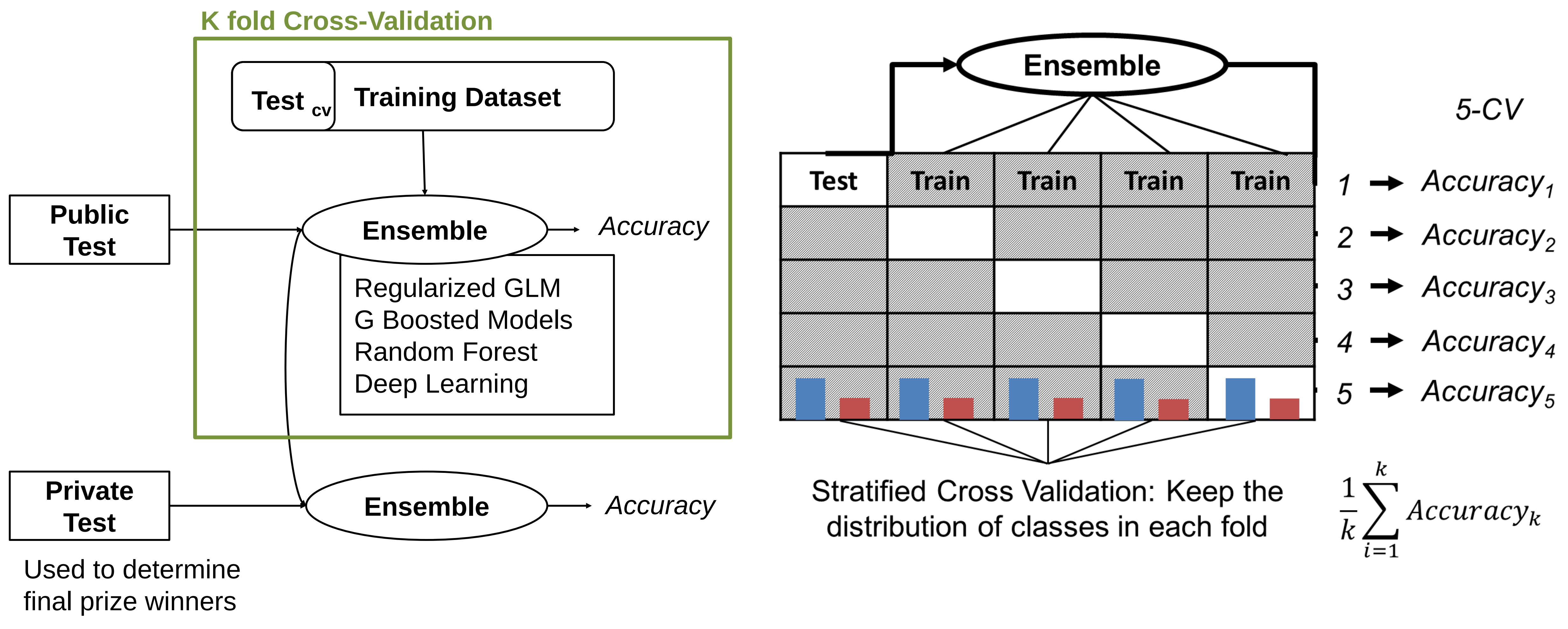
Ensemble Methods

No one model is perfect; different models see different aspects of the problem space and are wrong in different ways. Ensemble methods combine multiple prediction models with a weighted average. The result is lower variance, less chance of overfitting, and stronger predictive value.

Some competitions allow many submissions to be made, allowing participants to follow a trial and error strategy. However, it is also common for the test data to be released in stages. An intermediary test set is posted for the bulk of the challenge, only to be swapped out for the other portion of the test data for the final submission. With this type of challenge, the trial and error strategy will be inefficient and misleading.

We recently participated in a data challenge that only allowed several submissions to be made. Therefore, we needed to gain confidence in a given ensemble prior to submission. In order to compare the generalizability of various ensembles as we built them, we developed code to perform a repeated cross validation. Code can be found at *https://github.com/agstn/useR16_ensemble*.

Developing Ensembles Using K-Fold Cross Validation



H2O Cross Validation Functions

In order to estimate the prediction error of our ensembles on the test data, we developed code to perform a repeated k-fold cross validation. We present a series of functions built on top of existing H2O functionality to facilitate this process:

h2o.ensemble_cv()

This primary cross validation function takes an object of class h2o.ensemble and a training dataset. The training set is divided into k folds, each maintaining the distribution of the target variable. Based on a user-specified object of class h2o.ensemble, an ensemble is trained on each combination of k-1 folds.

h2o.ensemble_performance_cv()

This scoring function takes an object of class h2o.ensemble_cv and returns performance metrics based on the unused portion of the data for each fold.

h2o.metalearn_cv()

This function takes an object of class h2o.ensemble_cv and retrains it using all the same parameters, except this time using a different user-specified metalearner.

Existing H2O

```
fit <- h2o.ensemble(
         x = x, y = y,
         training_frame = train,
         family = family,
         learner = learner,
         metalearner = "h2o.glm_nn“)

perf <- h2o.ensemble_performance(
           fit,
           newdata = test)

nfit <- h2o.metalearn(
           fit,
           metalearner =
           "h2o.deeplearning.wrapper")

nperf <- h2o.ensemble_performance(
            nfit,
            newdata = test)
```

H2O Cross Validation

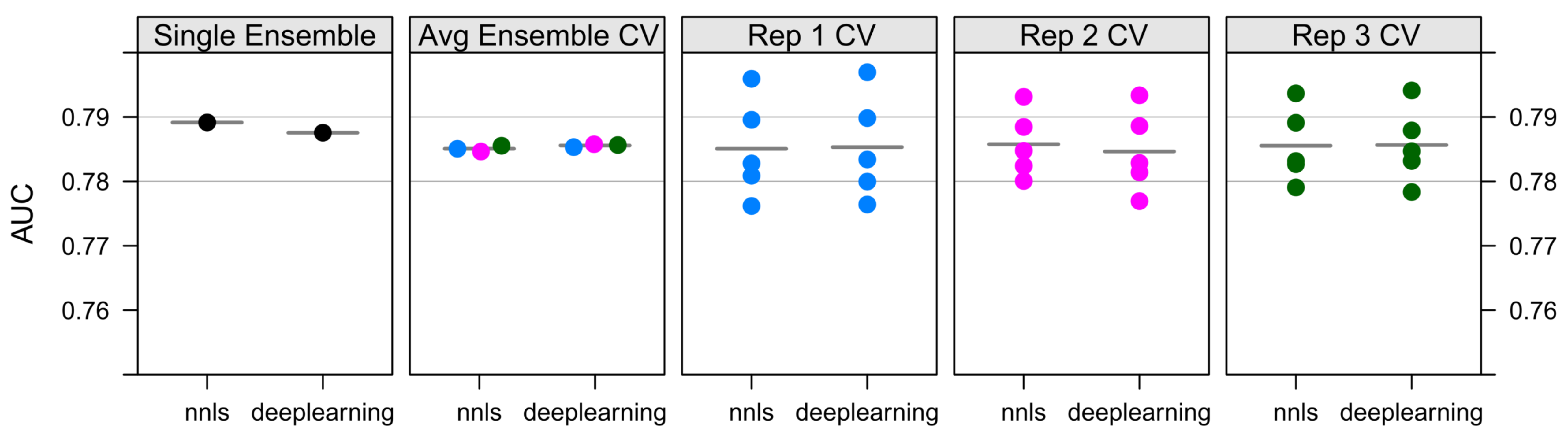
```
fit_cv <- h2o.ensemble_cv(
             fit,
             training_frame = train,
             K = 5, times = 3)


perf <- h2o.ensemble_performance_cv(
           fit_cv,
           training_frame=train)

nfit_cv <- h2o.metalearn_cv(
             fit_cv,
             metalearner=
             "h2o.deeplearning.wrapper")

nperf <- h2o.ensemble_performance_cv(
            nfit_cv,
            training_frame=train)
```

Example: Single Ensemble vs. Ensemble CV


Single Ensemble
Avg Ensemble CV
Rep 1 CV
Rep 2 CV
Rep 3 CV
AUC
nnls
deeplearning