MA8701 Advanced methods in statistical inference and learning

Part 3: Ensembles. L15: Stacked ensembles

Mette Langaas

2/28/23

Course homepage:
https://wiki.math.ntnu.no/ma8701/2023v/start

Before we start

Literature

- Erin Le Dell (2015): Scalable Ensemble Learning and Computationally Efficient Variance Estimation. PhD Thesis, University of California, Berkeley. or https://github.com/ledell/phd-thesis

Supporting literature

- Mark J. van der Laan, Eric C. Polley, Alan E. Hubbard (2007): Super Learner, Statistical Applications in Genetics and Molecular Biology, 6, 1, 25
- Eric C. Polley, Sherri Rose, Mark J. van der Laan (2011): Super Learning. Chapter 3 of M. J. van der Laan and S. Rose. Targeted Learning, Springer.

Ensembles - overview

(ELS Ch 16.1)

With ensembles we want to build *one prediction model* which combines the strength of *a collection of models*.

These models may be simple base models - or more elaborate models.

We have studied bagging - where we take a simple average of the prediction from many models (or majority vote), and the base models can be trees - or other type of models.

Random forest is a version of bagging, with trees made to be different (decorrelated).

We have studied boosting, where the models are trained on sequentially different data - from residuals or gradients of loss functions - and the ensemble members cast weighted votes. We have in particular looked into the xgboost variant of boosting, and also evaluated possible parameters to tune to optimize performance.

In L7 we also look into an ensemble built of elaborate models with the Super Learner and study some possible methods for tuning hyperparameters.

Super Learner

Why do we want to study the Super Learner?

The Super Learner or *generalized stacking* or "stacked ensemble" is an algorithm that combines

- multiple, (typically) diverse prediction methods (learning algorithms) called *base learners* into a
- a *metalearner* - which can be seen as a *single* method.

Development:

- 1992: stacking introduce for neural nets by Wolpert
- 1996: adapted to regression problems by Breiman - but only for one type of methods at once (cart with different number of terminal nodes, glms with subset selection, ridge regression with different ridge penalty parameters)
- 2006: proven to have asymptotic theoretical oracle property by van der Laan, Polley and Hubbard.

Ingredients:

- *Training data* (level-zero data) $O_i = (X_i, Y_i)$ of N i.i.d observations.
- A total of L *base learning algorithms* Ψ^l for $l = 1, \ldots, L$, each from some algorithmic class and each with a specific set of model parameters.
- A *metalearner* is used to find an *optimal combination* of the L base learners.

Algorithm

Step 1: Produce level-one data $\mathbf{Z}$

a) Divide the training data $\mathbf{X}$ randomly into V roughly-equally sized validation folds $\mathbf{X}_{(1)}, \ldots, \mathbf{X}_{(V)}$. V is often 5 or 10. (The responses $\mathbf{Y}$ are also needed.)

b) For each base learner Ψ^l perform V-fold cross-validation to produce prediction.

This gives the level-one data set $\mathbf{Z}$ consisting prediction of all the level-zero data - that is a matrix with N rows and L columns.

What could the base learners be?

"Any" method that produces a prediction - "all" types of problems.

- linear regression
- lasso
- cart
- random forest with mtry=value 1
- random forest with mtry=value 2
- xgboost with hyperparameter set 1
- xgboost with hyperparameter set 2
- neural net with hyperparameter set 1

Step 2: Fit the metalearner

a) The starting point is the level-one prediction data $\mathbf{Z}$ together with the responses $(Y_1, \ldots, Y_N)$.
b) The metalearner is used to estimate the weights given to each base learner: $\hat{\eta}_i = \alpha_1 z_{1i} + \cdots + \alpha_L z_{Li}$.

What could the metalearner be?

- the mean (bagging)
- constructed by minimizing the
  - squared loss (ordinary least squares) or
  - non-negative least squares
- ridge or lasso regression
- constructed by minimizing 1-ROC-AUC

(Class notes: Study Figure 3.2 from Polley et al)

Step 3: Re-estimate base learners and combine into superlearner on full training data

a) Fit each of the L base learners to the full training set.
b) The *ensemble fit* consists the L base learner fits together with the metalearner fit.

Step 4: Using the ensemble for prediction

For a new observaton $\mathbf{x}^*$

a) Use each of the L base learners to produce a prediction $\mathbf{z}^*$, and
b) feed this to the metalearner-fit to produce the final prediction y^*.

The metalearning

Some observations

- The term *discrete super learner* is used if the base learner with the lowest risk (i.e. CV-error) is selected.
- Since the predictions from multiple base learners may be highly correlated - the chosen method should perform well in that case (i.e. ridge and lasso).
- When minimizing the squared loss it has been found that adding a non-negativity constraint $\alpha_l \leq 0$ works well,
- and also the additivity constraint $\sum_{l=1}^{L} \alpha_l = 1$ - the ensemble is a *convex combination* of the base learners.
- Non-linear optimization methods may be employed for the metalearner if no existing algorithm is available
- Historically a regularized linear model has "mostly" been used
- For classification the logistic response function can be used on the linear combination of base learners (Figure 3.2 Polley).

Examples

Simulations examples

(Class notes: Study Figure 3.3 and Table 3.2 from Polley et al)

Real data

(Class notes: Study Figure 3.4 and Table 3.3 from Polley et al. RE=MSE relative to the linear model OLS.)

Theoretical result

(Le Dell 2015, page 6)

- Oracle selector: the estimator among all possible weighted combinations of the base prediction function that minimizes the risk under the *true data generating distribution*.
- The *oracle result* was established for the Super Learner by van der Laan et al (2006).
- If the *true prediction function* cannot be represented by a combination of the base learners (available), then "optimal" will be the closest linear combination that would be optimal if the true data-generating function was known.
- The oracle result require an *uniformly bounded loss function*. Using the convex restriction (sum alphas =1) implies that if each based learner is bounded so is the convex combination. In practice: truncation of the predicted values to the range of the outcome in the training set is sufficient to allow for unbounded loss functions

Uncertainty in the ensemble

(Class notes: Study "Road map" 2 from Polley et al)

- Add an outer (external) cross validation loop (where the super learner loop is inside). Suggestion: use 20-fold, especially when small sample size.
- Overfitting? Check if the super learner does as well or better than any of the base learners in the ensemble.
- Results using *influence functions* for estimation of the variance for the Super Learner are based on asymptotic variances in the use of V-fold cross-validation (see Ch 5.3 of Le Dell, 2015)

Other issues

- Many different implementations available, and much work on parallell processing and speed and memory efficient execution.
- Super Learner implicitly can handle hyperparameter tuning by including the same base learner with different model parameter sets in the ensemble.
- Speed and memory improvements for large data sets involves subsampling, and the R `subsemble` package is one solution, the H2O Ensemble project another.

R example from Superlearner package

Code is copied from Guide to SuperLearner and the presentation follows this guide. The data used is the Boston housing dataset from `MASS`, but with the median value of a house dichotomized into a classification problem.

Observe that only 150 of the 560 observations is used (to speed up things, but of cause that gives less accurate results).

```
data(Boston, package = "MASS")
#colSums(is.na(Boston)) # no missing values
outcome = Boston$medv
# Create a dataframe to contain our explanatory variables.
data = subset(Boston, select = -medv)
#Set a seed for reproducibility in this random sampling.
set.seed(1)
# Reduce to a dataset of 150 observations to speed up model
train_obs = sample(nrow(data), 150)
# X is our training sample.
x_train = data[train_obs, ]
```

R example from H2O-package

References

Efron, Bradley, and Trevor Hastie. 2016. *Computer Age Statistical Inference - Algorithms, Evidence, and Data Science*. Cambridge University Press. https://hastie.su.domains/CASI/.

Hastie, Trevor, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Vol. 2. Springer series in statistics New York. hastie.su.domains/ElemStatLearn.

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Vol. 112. Springer.