Ensemble Methods

Kevin Gillespie

2023-10-12

Section 1

Introduction

Experts

- For important decision we often rely on experts
- Sometimes we seek out second opinions *(or third, etc.)*
- Once acquired we weight the opinions from the multiple various experts

What is an ensemble method

Use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone.

Section 2

Motivating Problem

Common Requests for our Expertise

- Common request: ‘Find the variable(s) that best explain avg. tx outcome’
- **Less common: ‘Construct prediction model that best predicts outcome, unsure which variable(s) are important but have many’**

Common Requests for our Expertise

- In less common case, very often, no one cares what algorithm you use
    - Probably for most of what we do people don't care what algorithm we choose, just that the results are 'right'
- Sometimes difficult to choose 'optimal' algorithm in first place
- Do we just pick our favorite for the task at hand?
- Is there some way to weight the results from multiple 'experts' (algorithms)?

Section 3

Ensemble Learning

How it works

- Ensemble methods build a predictive model by integrating the predictions from multiple **constituent** models
- Many ways to do this: Bagging, Boosting, **Stacking**, etc.

SuperLearner

- R package SuperLearner
- Uses **stacking** method to combine the predictions from multiple **constituent** models trained on the source data
- Geared toward: binary, continuous outcome
  - *Some very smart people at the Hutch have attempted to do the same for time-to-event data*

SuperLearner

- Use data set: $X_i = (Y_i, W_i)), i = 1, ..., n$
  - Y: Outcome of interest
  - W: p-dimensional set of covariates
- Library of algorithm $\mathcal{L}$
  - Algorithm in general is any mapping from the data into a predictor
    - Grand mean assigned to each outcome
    - Linear regression
    - LASSO, ridge, GAM, random forests, etc.
  - Can include screening steps
- Objective is to estimate $\psi_0(W) = E(Y|W)$
  - Expressed as: $\psi_0(W) = \underset{\psi}{\operatorname{argmin}}\, E[L(X, \psi(W))]$

SuperLearner

The learning, denote the library $\mathcal{L}$ and the cardinality of $\mathcal{L}$ as $K(n)$.

1. Fit each algorithm in $\mathcal{L}$ on the entire data set $\mathcal{X} = \{X_i : i = 1, ..., n\}$ to estimate $\hat{\Psi}_k(W), k = 1, ..., K(n)$
2. Training-Testing Split: split into V-equal sized groups, where v-th group is validation and the rest training. Define $T(v)$ to be the v-th training data split and $V(v)$ to be the corresponding validation data split. $T(v) = \mathcal{X} \backslash V(v), v = 1, ..., V$

3. For the v-th fold, fit each algorithm in $\mathcal{L}$ on $T(v)$ and save the predictions on the corresponding validation data, $\hat{\Psi}_{k,T(v)}(W_i), X_i \in V(v)$ for $v = 1, ..., V$
4. Stack the predictions from each algorithm together to create a n by K matrix, $Z = \{\hat{\Psi}_{k,T(v)}(W_{V(v)}, v = 1, ..., V \;\&\; k = 1, ..., K\}$, where we used the notation $W_{V(v)} = (W_i : X_i \in V(v))$ for the covariate-vectors of the $V(v)$-validation sample

5. Propose a family of weighted combinations of the candidate estimators indexed by weight-vector α:

$$m(z|\alpha) = \sum_{k=1}^{K} \alpha_k \hat{\Psi}_{k,T(v)}(W_{V(v)}), \alpha_k \geq 0 \;\forall\; k$$

6. Determine the α that minimizes the cross-validated risk of the candidate estimator $\sum_{k=1}^{K} \alpha_k \hat{\Psi}_k$ over all allowed α-combinations:

$$\hat{\alpha} = \underset{\alpha}{\operatorname{argmin}} \sum_{i=1}^{n} (Y_i - m(z_i|\alpha))^2$$

7. Combine $\hat{\alpha}$ with $\hat{\Psi}_k(W), k = 1, ..., K$ according to the family $m(z|\alpha)$ of weighted combination to create the final super learner fit:

$$\hat{\Psi}_{SL}(W) = \sum_{k=1}^{K} \hat{\alpha}_k \hat{\Psi}_k(W)$$

Section 4

Examples

Examples

- Using a behavior questionnaire to predicit HIV-1 risk in an efficacy trial
- Using microarray, RNAseq, any P (predictors) larger than N (sample)
- Using factors (age, no. of pregnancies, BMI, etc.) to predict diabetes in the Pima Indian Women *(shown)*

Diabetes Risk in Pima Indian Women

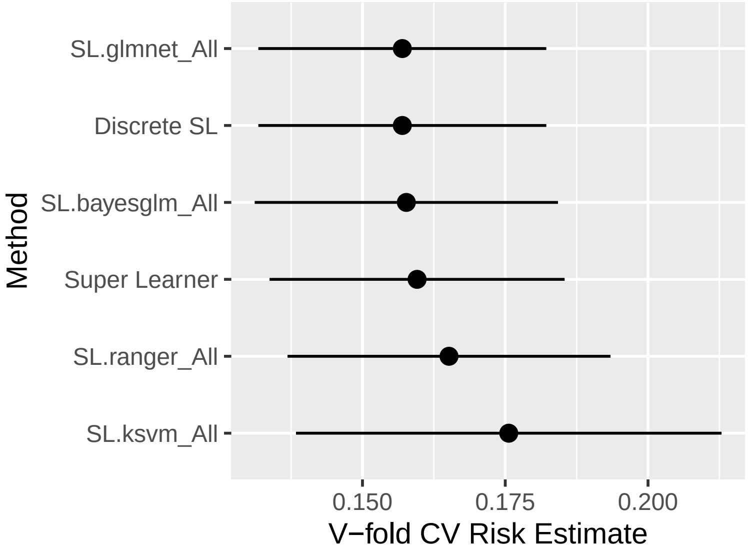
- `MASS` package dataset
- A population of women who were at least 21 years old, of Pima Indian heritage and living near Phoenix, Arizona, was tested for diabetes according to World Health Organization criteria. The data were collected by the US National Institute of Diabetes and Digestive and Kidney Diseases. We used the 532 complete records after dropping the (mainly missing) data on serum insulin.
- Variables:
  - npreg: No. of pregnancies
  - glu: plasma glucose concentration in an oral glucose tolerance test
  - bp: diastolic blood pressure (mm Hg)
  - skin: triceps skin fold thickness (mm)
  - bmi: body mass index (weight in $kg/(m)^2$)
  - ped: diabetes pedigree function
  - age: age in years
  - type: Yes or No, for diabetic according to WHO criteria

Diabetes Risk in Pima Indian Women

Table 1: Summary of Risk Estimates using 5-fold CV

Algorithm	Ave	se	Min	Max
Super Learner	0.160	0.013	0.129	0.185
Discrete SL	0.157	0.013	0.123	0.185
SL.ranger_All	0.165	0.014	0.143	0.183
SL.ksvm_All	0.176	0.019	0.157	0.193
SL.glmnet_All	0.157	0.013	0.123	0.185
SL.bayesglm_All	0.158	0.014	0.120	0.186

Diabetes Risk in Pima Indian Women



Diabetes Risk in Pima Indian Women

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0 201  44
##          1  22  65
##
##                Accuracy : 0.8012
##                  95% CI : (0.7542, 0.8428)
##     No Information Rate : 0.6717
##     P-Value [Acc > NIR] : 1.116e-07
##
##                   Kappa : 0.5247
##
##  Mcnemar's Test P-Value : 0.00974
##
##             Sensitivity : 0.9013
##             Specificity : 0.5963
##          Pos Pred Value : 0.8204
##          Neg Pred Value : 0.7471
##              Prevalence : 0.6717
##          Detection Rate : 0.6054
##    Detection Prevalence : 0.7380
##       Balanced Accuracy : 0.7488
##
##        'Positive' Class : 0
##
```

Questions?

Section 5

Extra slides

SuperLearner wrappers

- Uses these wrappers for constituent learners *(LASSO presented below)*

```
## function (Y, X, newX, family, obsWeights, id, alpha = 1, nfolds = 10,
##     nlambda = 100, useMin = TRUE, loss = "deviance", ...)
## {
##     .SL.require("glmnet")
##     if (!is.matrix(X)) {
##         X <- model.matrix(~-1 + ., X)
##         newX <- model.matrix(~-1 + ., newX)
##     }
##     fitCV <- glmnet::cv.glmnet(x = X, y = Y, weights = obsWeights,
##         lambda = NULL, type.measure = loss, nfolds = nfolds,
##         family = family$family, alpha = alpha, nlambda = nlambda,
##         ...)
##     pred <- predict(fitCV, newx = newX, type = "response", s = ifelse(useMin,
##         "lambda.min", "lambda.1se"))
##     fit <- list(object = fitCV, useMin = useMin)
##     class(fit) <- "SL.glmnet"
##     out <- list(pred = pred, fit = fit)
##     return(out)
## }
## <bytecode: 0x000000002de9bce8>
## <environment: namespace:SuperLearner>
```