

A VISUAL GUIDE TO SUPERLEARNING



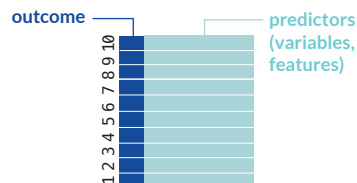
Katherine Hoffman, MS
@rkatlady

Superlearning, or stacking, weights the results of many individual machine learning algorithms to create an optimal overall prediction algorithm. Superlearner predictions are guaranteed to perform at least as well as any of the individual learners in large sample sizes.

There can be minor variations in how the learners are combined, but the algorithm described here is the same as in Chapter 3 of *Targeted Learning* by Eric Polley, Sherri Rose, and Mark van der Laan.




STEP 1

Split data into 10 blocks in preparation for 10-fold cross validation.



STEP 2







Train multiple base learners, or regressions, on 9 of the 10 blocks of data. Note that you are not limited to a specific number of learners.

```
fit_1a <- lnrn_a( ~  )  
fit_1b <- lnrn_b( ~  )  
fit_1c <- lnrn_c( ~  )
```

An example of three base learners for a binary variable could be random forest, gradient boosting, and logistic regression.

STEP 3

Obtain predictions from each base learner for the held-out block of data.

```
 <- predict(fit_1a,  
  newdata = )  
 <- predict(fit_1b,  
  newdata = )  
 <- predict(fit_1c,  
  newdata = )
```

STEP 4

Repeat until each of the 10 blocks have served as the hold-out data and you have three sets of cross-validated predictions spanning the full data set.



STEP 5




Using a new learner, predict the outcome using those three sets of cross-validated predictions.

```
SL_fit <- meta_lnrn( ~  +  +  )
```

An example of the metalearner could be as simple as a generalized linear model. As with any statistical learning algorithm, the choice reflects the loss function we want to minimize for our cross-validated predictions.




STEP 6

Fit the base learners on the entire data set.

```
fit_a <- lnrn_a( ~  )  
fit_b <- lnrn_b( ~  )  
fit_c <- lnrn_c( ~  )
```

STEP 7

Obtain predictions from the full data set for each learner.

```
 <- predict(fit_a)  
 <- predict(fit_b)  
 <- predict(fit_c)
```

STEP 8

Use the coefficients from Step 5 to weight the full data predictions from Step 7. These are the final superlearner predictions.

```
 <- predict(SL_fit,  
  newdata = )
```

This makes the final superlearner predictions a weighted combination of the base learners' predictions.

EVALUATION

To test the prediction capability of the superlearner and to prevent overfitting, the entire algorithm (Steps 1-8) could be cross-validated.



TARGETED LEARNING

Like most algorithms designed for prediction, the predictions from superlearning do not have standard errors, making statistical inference such as confidence intervals and p-values impossible.

When prediction is not the end goal, superlearning can be easily combined with Targeted Learning methods (EX: Targeted Maximum Likelihood Estimation (TMLE)) for statistical inference. This allows us to utilize flexible models and place minimal assumptions on the distribution of our

APPLICATION IN R

An example of using the Superlearner package to predict a binary outcome with three learners: gradient boosting (xgboost), random forest (ranger), and logistic regression (glm):

```
SL_fit <- Superlearner( , ,  
  family=binomial(),  
  SL.library = c("SL.xgboost",  
    "SL.ranger",  
    "SL.glm"))
```

```
 <- predict(SL_fit)
```

A tutorial with R code, references, and more detailed explanations can be found at: www.khstats.com/blog/sl/superlearning.