Pediatric IBI 2022

This contains the software and data used to create a classifier for invasive bacterial infections based on clinical characteristics. There are three main parts of this process:

- Training the classifier
- Validating the classifier
- Creating honest confidence intervals

Training the classifier

Our classifier is a super learner ensemble of learners, based on the `SuperLearner` package in `R`. The super learner uses a `library` of learners that can be trained on the data. We train each learner in the library and combine them into an ensemble, so that a prediction using all of the learners in the ensemble is better than any individual one. This makes the training method more robust to modeling assumptions and improves the generalizability of the classifier.

The invasive bacterial infection (IBI) outcome is nested under the bacterial infection (BI) outcome, in the sense that all IBI-positive infants are BI-positive, but not all BI-positive infants are IBI-positive. This provides a challenge to training the classifier, since there are very few IBI-positive infants in general. To improve the training process, we used BI outcomes as a surrogate for the IBI outcomes, since there are many more examples of BI-positive infants in the data to work with. To ensure that the classifier actually targeted the IBI outcome rather than the BI outcome, we used observation weights to provide more importance to the

Mathematical description

Let Y be the IBI outcome we want to predict, X be the covariates to classify with, and Z be the BI surrogate outcome. We used observation weights $w = w(Z, Y)$ which were created to up-weight infants with $Y = 1$. If a classifier f would usually minimize an empirical risk of the form

$$R^{\dagger}(f) = \frac{1}{n}\sum_{i=1}^{n} L\big\{f(X_i); Z_i\big\},$$

then our observation weights targeted the slightly different function

$$R(f) = \frac{1}{n}\sum_{i=1}^{n} w(Z_i, Y_i) L\big\{f(X_i); Z_i\big\}.$$

The weights were created to up-weight the importance of $Z_i = Y_i = 1$ and downweight the occurrence of $Z_i = 1, Y_i = 0$. This allows the weighted classifier to target the occurrence $Y_i = 1$ rather than $Z_i = 1$, even though Z_i is used in the training process.

Validating the Classifier

We used cross-validation to estimate the generalization performance of the super learner classifier. Since super learning itself utilizes a layer of cross-validation, this means that we used, in effect, a 2-layer nested cross-validation procedure: the outer layer (examining the performance of the super learner) used 10 folds for cross-validation while the inner layer (used to find an optimal combination of the learner library) used 5 folds. Each layer optimized the weighted criteria described above. We picked the weights from among 7 different possibilities on the basis of the cross-validated AUC for predicting IBI. I.e., we trained 7 different versions of the `CV.SuperLearner` with different weights and picked the one offering the best AUC performance when applied to the IBI outcome. All of these super learners used the same cross-validation folds and library of learners. The weight we chose in practice ended up being the weights that took a value of 8 when $Y_i = Z_i = 1$ and 1 otherwise. Effectively, this upweighted IBI-positive infants 8-fold in importance compared to other infants. This provided the best AUC performance, which is a balanced measure of performance among the IBI-positive and IBI-negative infants. ^20knbg2676io

Since we didn't use another layer of cross-validation to encapsulate the selection of the weights, we used the boostrap bias-corrected cross-validation procedure to provide inference.

Creating honest confidence intervals

Nested cross-validation provided protection from overfitting the classifier. However, we did not wrap up the weight selection in yet another layer of cross-validation. This would have been very troublesome for the analysis, as there were already 5×10 different cross-validation folds being created and very few outcomes that could be used for training in each fold. To get around this, we used the bootstrap bias-corrected CV procedure. Here is a rough outline of how this process works:

- Start with the cross-validated super learner predictions corresponding to one of the seven weight configurations. Call the predictions $\hat{Y}_w$ for $w = 1, \ldots, 7$.
- Use the bootstrap to resample the row indices $\{1, \ldots, n\}$. Let I_b be the b^{th} resample of the indices.
- For $b = 1, \ldots, \ 50,000$:
  - The resampled outcomes and predictions $Y[I_b], \hat{Y}_w[I_b]$ can be used to select the AUC-optimal weight configuration $\hat{w}_b$.
  - Use the out-of-bootstrap sample to estimate performance on the selected configuration: $\hat{Y}_{\hat{w}_b}[I_b^c]$ for I_b^c the set complement of I_b. We provide another layer of bootstrap here to resample among the I_b^c to provide confidence intervals. Call the resampled I_b^c indices J_b.
  - Let $T(\hat{Y}, Y)$ be one of the test characteristics we want to provide inference for, e.g. the Sensitivity or Specificity for some predictions $\hat{Y}$

and true class values Y. In the b^{th} iteration, return $T\left(\hat{Y}_{\hat{w}_b}[J_b], Y[J_b]\right)$

- Use the percentile bootstrap method to create confidence intervals for the test statistic T. I.e. use quantiles of the 50,000 values of $T\left(\hat{Y}_{\hat{w}_b}[J_b], Y[J_b]\right)$ to build confidence intervals.