Data Analytics
EEE 4774 & 6777

Module 4 - Classification

Ensemble Methods - Boosting - Bagging - Random Forest

Spring 2022

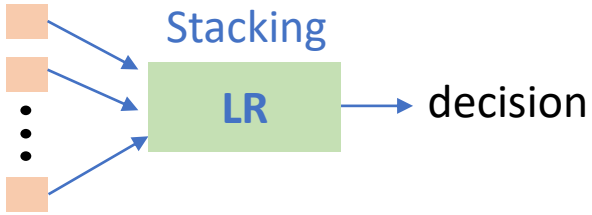
Ensemble Methods

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

Most data challenges are won by ensemble methods

- **Bayesian model averaging** (BMA) makes predictions using an average over several models with weights given by the posterior probability of each model given the data.

- **Stacking** involves training a learning algorithm to combine the predictions of several other learning algorithms. First, all of the other algorithms are trained using the available data, then a combiner algorithm is trained to make a final prediction using all the predictions of the other algorithms as additional inputs. In practice, a logistic regression model is often used as the combiner.



Boosting and Bagging

- **Boosting** involves incrementally building an ensemble by training each new model instance to emphasize the training instances that previous models mis-classified.

- **Bootstrap aggregating (*bagging)***, involves having each model in the ensemble vote with equal weight. In order to promote model variance, bagging trains each model in the ensemble using a randomly drawn subset of the training set. For example, the random forest algorithm combines random decision trees with bagging to achieve very high classification accuracy in many problems.

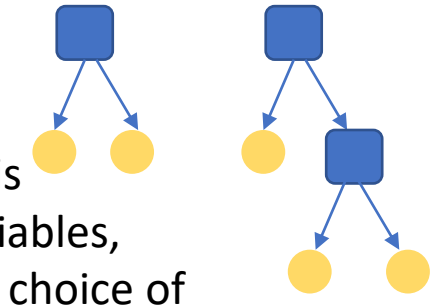
Boosting

- Produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees, for both regression and classification problems.
- Gradient boosting is typically used with decision trees (especially CART trees) of a fixed size as base learners.
- Generic gradient boosting: at the *m*-th step fit a decision tree $h_m(x)$ to pseudo-residuals, i.e., correctly classified vs. misclassified instances

$$h_m(x) = \sum_{j=1}^{J_m} b_{jm} \mathbf{1}_{R_{jm}}(x),$$

$$F_m(x) = F_{m-1}(x) + \sum_{j=1}^{J_m} \gamma_{jm} \mathbf{1}_{R_{jm}}(x), \quad \gamma_{jm} = \arg\min_{\gamma} \sum_{x_i \in R_{jm}} L(y_i, F_{m-1}(x_i) + \gamma).$$

- Size of trees: the number of terminal nodes in trees controls the maximum allowed level of interaction between variables in the model. With $J = 2$ no interaction between variables is allowed. With $J = 3$ the model may include effects of the interaction between up to two variables, and so on. Typically, $4 \leq J \leq 8$ work well for boosting and results are fairly insensitive to the choice of J in this range.



- Ex: **AdaBoost, XGBoost**

Python Exercise: AdaBoost in scikit-learn

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=1000, n_features=4, n_informative=2, n_redundant=0, random_state=0,
shuffle=False)
clf = AdaBoostClassifier(n_estimators=100, random_state=0)
clf.fit(X, y)
clf.predict([[0, 0, 0, 0]])
clf.score(X, y)
```

Bagging

- Bootstrap aggregating (bagging) for tree learners
  - First obtain B bootstrap sets from the original training set
  - For b = 1, ..., B:
    - Sample, with replacement, n training examples from X, Y; call these X_b, Y_b.
    - Train a classification or regression tree on X_b, Y_b.
  - After training, predictions for unseen samples can be made by taking the majority vote in the case of classification trees
- By sampling with replacement, some observations may be repeated in each bootstrap set. For large n, each bootstrap set is expected to have the fraction (1 - 1/e ≈ 63.2%) of the unique examples of original dataset, the rest being duplicates.
- While the predictions of a single tree are highly sensitive to noise in its training set, the average of many trees is not, as long as the trees are not correlated
- Simply training many trees on a single training set would give strongly correlated trees (or even the same tree many times, if the training algorithm is deterministic); bootstrap sampling is a way of de-correlating the trees by showing them different training sets.
- The number of samples/trees, *B*, is a free parameter. Typically, a few hundred to several thousand trees are used, depending on the size and nature of the training set. An optimal number of trees *B* can be found using cross-validation, or by observing the *out-of-bag error*: the mean prediction error on each training sample x_i, using only the trees that did not have x_i in their bootstrap sample. The training and test error tend to level off after some number of trees have been fit.

Random Forest

- A bagging algorithm for averaging multiple deep decision trees, trained on different parts of the same training set, with the goal of reducing the variance
- At the expense of a small increase in the bias and some loss of interpretability, but generally greatly boosts the performance in the final model
- Random forests differ in only one way from the original bagging algorithm for trees:
  - they use a modified tree learning algorithm that selects, at each candidate split in the learning process, a random subset of the features.
  - This process is sometimes called "feature bagging". The reason for doing this is the correlation of the trees in an ordinary bootstrap sample: if one or a few features are very strong predictors for the response variable (target output), these features will be selected in many of the *B* trees, causing them to become correlated.
- Typically, for a classification problem with p features, $\sqrt{p}$ (rounded down) features are used in each split.
- In practice the best values for these parameters will depend on the problem, and they should be treated as tuning parameters