Machine Learning

Bagging

Sebastian Engelke

Master in Business Analytics

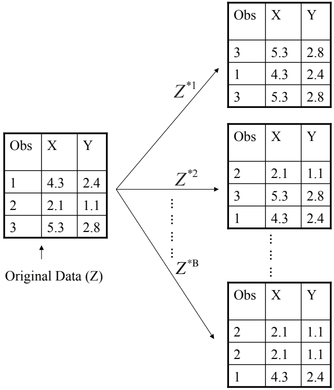
UNIVERSITÉ DE GENÈVE

Bagging: main idea

- Bootstrap aggregation, or bagging is a general-purpose procedure for reducing the variance of a statistical learning method.
- Recall that given a set of B independent observations $z_1, \ldots, z_B$ each with variance σ^2, the variance of the mean
$$\overline{z} = \frac{1}{B} \sum_{i=1}^{B} z_i$$
of the observations is given by σ^2/B.
- We can thus reduce the variance by averaging a set of predictions. Of course, this is not practical because we generally do not have access to multiple training sets.
- Instead we use bootstrap by taking repeated samples from the training data $Z = \{(x_i, y_i), i = 1, \ldots, n\}$.

The bootstrap

- Let B be the number of desired bootstrap data sets $Z^{*1}, \ldots, Z^{*B}$.
- For $b = 1, \ldots, B$:
- sample n oberservations from Z `with replacement`;
- this is the bth `bootstrap data set` denoted by Z^{*b}.
- Each bootstrap data set contains `on average` 63.2% of the original samples.



Bagging

- Let $Z^{*1}, \dots, Z^{*B}$ be B bootstrap data sets from the training data.
- For all $b = 1, \dots, B$ we can fit our predictive regression model $\widehat{f}^{*b}$ on the bth bootstrap data set Z^{*b}.
- We then average all the predictions to obtain a single, bagged model

$$\widehat{f}_{\text{bag}}(x) = \frac{1}{B} \sum_{b=1}^{B} \widehat{f}^{*b}(x),$$

  which is called bagging.
- Especially if the single functions $\widehat{f}^{*b}$ have a `low bias but a high variance` the `combined estimator` $\widehat{f}_{\text{bag}}$ has a similar bias but `lower variance`!
- In practice we use a value of B sufficiently large that the `error has settled down`. Using a value of B between 100 and 1000 is often sufficient.

Out-of-bag error estimate

- There is a very straightforward way to estimate the test error of a bagged model, called the `out-of-bag (OOB)` error estimate.
- The key idea of bagging is that the model is repeatedly fitted to bootstrapped subsets of the observations, which contain on average around `two-thirds of the observations` ($\sim 63\%$).
- The `remaining one-third` ($\sim 37\%$) of the observations not used to fit a given bagged model are referred to as the `out-of-bag` observations.
- We can predict the response for the ith observation using all models $\widehat{f}^{*b}$ with $b \in I_{x_i} = \{b = 1, \ldots, B : (x_i, y_i) \notin Z^{*b}\}$, in which that `observation was OOB`. This will yield around $B/3$ predictions for the ith observation, which we average:

$$\mathrm{Err}_{\widehat{f}_{\mathrm{bag}}}(x_i) = \frac{1}{|I_{x_i}|} \sum_{b \in I_{x_i}} (y_i - \widehat{f}^{*b}(x_i))^2.$$

- The average over all n observations is called `out-of-bag (OOB) error` estimate:

$$\mathrm{Err}_{\widehat{f}_{\mathrm{bag}}} = \frac{1}{n} \sum_{i=1}^{n} \mathrm{Err}_{\widehat{f}_{\mathrm{bag}}}(x_i).$$

- For large B this estimator can be shown to be `equivalent to the Leave-One-Out` CV estimator. But in bagging it comes for free!

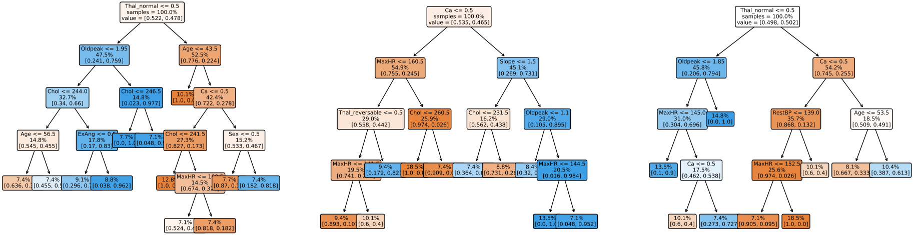
Bagging for trees

- Classification and regression trees are `perfectly suited` for bagging, since their poor predictive accuracy comes from their high variance.
- For a regression problem, for all B bootsrap training sets we grow a large tree `without pruning`, to obtain a `high-variance` predictive model.
- We then average all these (possibly overfitted) trees to obtain a `bagged` model with reduced variance.
- This has shown to give impressive improvements in `prediction accuracy`, at the price of intepretability.
- Important: here, B is `not` a classical `tuning parameter`, since large B will not overfit the data; the error will just stabilize.
- For `classifiction`, we fit B classification trees $\widehat{G}^{*b}$ and then choose the predicted class by `majority vote`, that is

$$\widehat{G}_{\text{bag}}(x) = \text{argmax}_{g=1,\dots,q} \frac{1}{B} \sum_{b=1}^{B} \mathbf{1}\{g = \widehat{G}^{*b}(x)\}.$$

Bootstraped trees in python

```
np.random.seed(1)
fig, axs = plt.subplots(1,3,figsize=(24,7))
for i in range(1,4):
    boot_indexes = np.random.choice(X.index, size=X.shape[0], replace=True)
    tree = DecisionTreeClassifier(criterion='entropy', min_samples_leaf=21, ccp_alpha=0)
    tree.fit(X.loc[boot_indexes], y.loc[boot_indexes])
    plot_tree(tree, feature_names=X.columns.tolist(), impurity=False, label="root",
           filled=True, proportion=True, rounded=True, fontsize=8, ax=axs[i-1])
plt.show()
```



Heart data: single tree

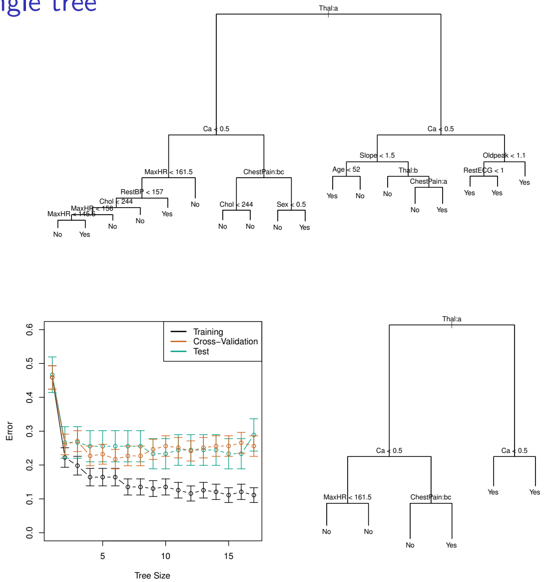


Figure: `Heart` data. Top: The unpruned tree. Bottom left: CV error, training, and test error, for different sizes of the pruned tree. Bottom right: The pruned tree corresponding to the minimal CV error.

Heart data: bagging