Ensemble learning

Lluís Talavera, 2022

UNIVERSITAT POLITÈCNICA DE CATALUNYA BARCELONATECH

ETSEIB

Ensemble learning

Seeks better predictive performance by combining predictions from multiple models called base learners or base models.

Can be used for decreasing variance error or bias error.

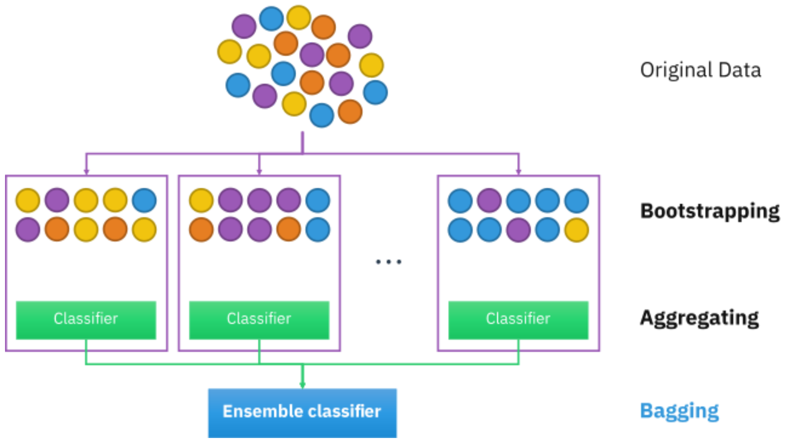
Divided into two grups:

> Parallel learners: Base learners are generated in parallel and their predictions aggregated.
>
> Sequential learners: Base learners are generated sequentially and the mistakes of previous models are learned by their successors.

Bagging (Bootstrap Aggregating)

Bootstrap sample: A random sample of the data taken with replacement, i.e., data can be selected several times for inclusion. It is the same size as the original dataset.

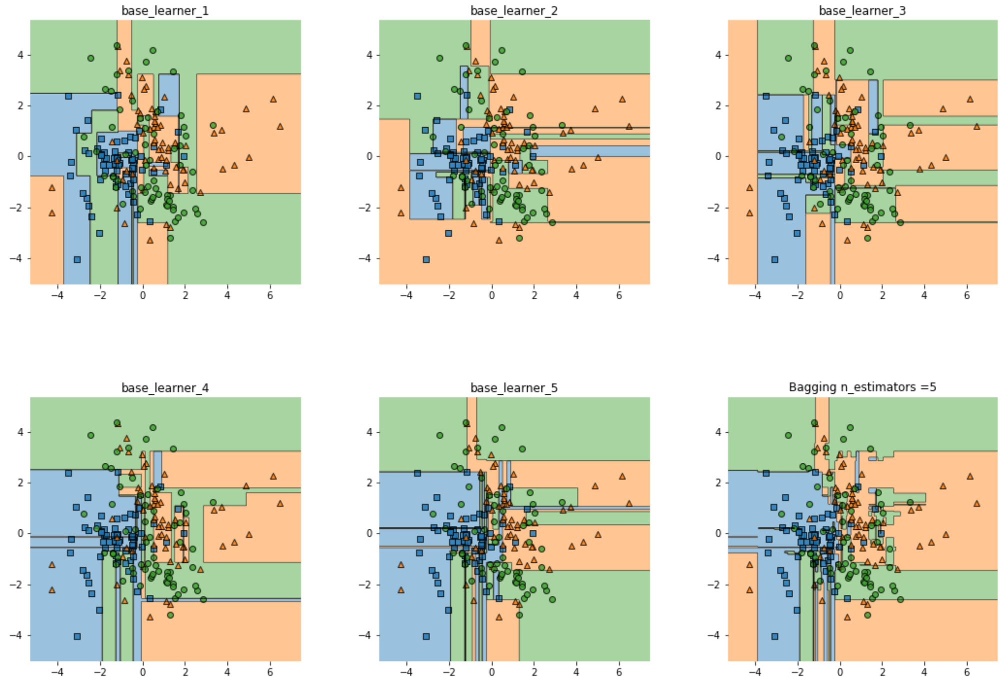
1. Given a training set X generate m bootstrap samples $X_1,X_2, \dots , X_m$.
2. Train a full decision (or regression) tree with each X_i sample.
3. Make predictions for new data by majority vote (classification) or averaging (regression).



Some remarks about Bagging

- By model averaging, Bagging helps to reduce variance and minimize overfitting.
- Works especially well for **unstable**, **high variance base learners**, i.e., algorithms whose predicted output changes in response to small changes in the training data. Usually applied to decision trees but other models can be used, e.g., k-NN with small k.
- For algorithms that are more stable or exhibit high bias, Bagging may not improve the performance.
- Tree correlation may limit the effect of variance reduction.
- Bagging improves performance at the expense of interpretability (if base learners are interpretable).
- We can compute *feature importance* by adding up the total amount of impurity (or residual sum of squares) decreased by splits over a given predictor, averaged over all trees (not directly implemented in `sklearn`).
- Hyperparameters: the number of trees, the size of the bootstrap sample.

Bagging decision boundaries

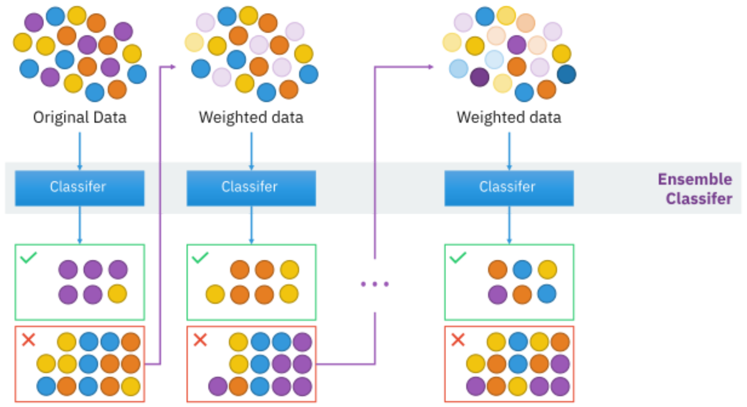


Random Forests

- Improve Bagging by de-correlating the trees.
- Inject more randomness into the tree building process: use only a **random subset of the original set of variables** as candidates for the splits. A fresh sample of variables is taken at each split.
- Hyperparameters:
    - The number of trees.
    - The number of features to consider when looking for the best split.
    - Size of the bootstrap sample.
    - Hyperparameters of the individual decision trees.
- Bagging would be a special case of Random Forests where instead of selecting a random subset of variables the full set is used.
- As in Bagging, we can compute *feature importance* (provided by `sklearn` as a property of the model).

Boosting

- Weak learner: A model that performs only slightly better than random prediction.
- Boosting tries to build a strong learner (model) from the errors of several weaker models.
- Models are added sequentially until training data is predicted perfectly or the maximum numer of models have been included. Each model in the sequence slightly improves upon the performance of the previous one by focusing on the examples of where the previous model had the largest errors.



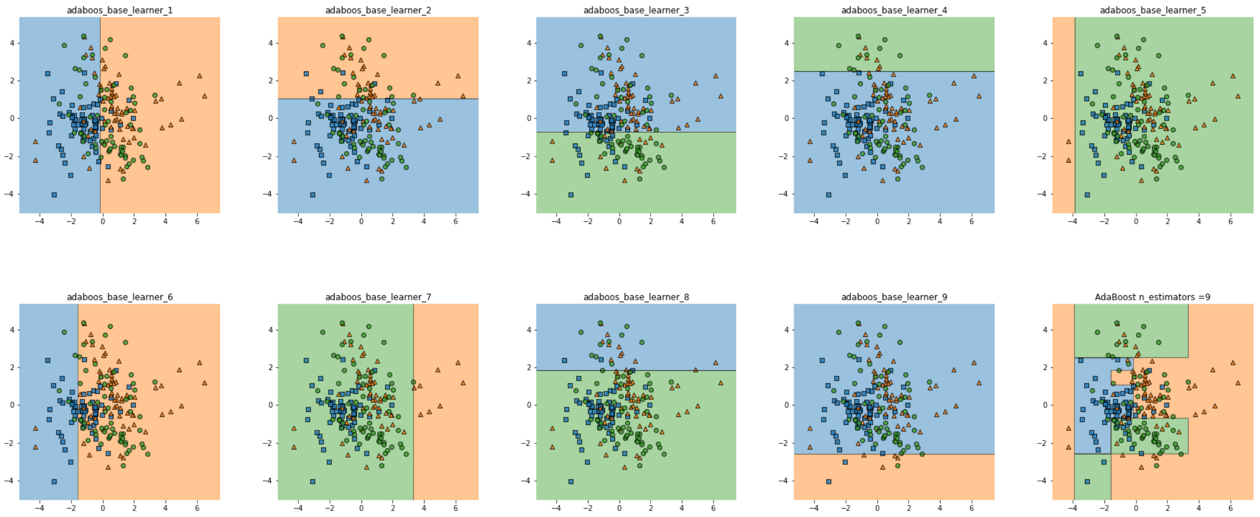
AdaBoost (**Adaptive Boosting**)

1. Assign equal weight to all the examples.
2. Create a **decision stump** with favors the splitting of sets of examples with large weights.
3. Assign a weight to the model. Re-assign the weights of the examples (higher weights to incorrectly classified examples).
4. Repeat 2 to 4 until all the data points have been correctly classified or the maximum iteration level has been reached.
5. Given a new example, return a prediction of the label with the higher sum of weights.

Hyperparameters:

- Number of trees
- Learning rate: weight applied to each classifier at each boosting iteration.

AdaBoost decision boundaries



Gradient Boosting

- Re-defines boosting as a numerical optimisation problem where the objective is to minimize the loss function of the model by adding weak learners using **gradient descent** (finds a local minimum of a differentiable function).
- It does not modify the training data distribution as base learners are trained to predict the residuals (errors) of the previous learner.
- Base learners can be deeper trees although still constrained in order to remain weak learners.
- All the learners have equal weight.
- Hyperparameters:
    - Number of trees.
    - Learning rate: determines the contribution of each tree on the final outcome and controls how quickly the algorithm proceeds down the gradient descent.
    - Hyperparameters of the individual decision trees.

Some remarks about Boosting

- Tries to reduce the bias error which arises when models are not able to identify relevant trends in the data.
- We can compute *feature importance* (provided by `sklearn` as a property of the model).
- Computationally expensive (not as easy to parallelize as Bagging methods).
- GB (or variants such as XGBoost) is one of the most competitive algorithms.
- Can overemphasize outliers and be prone to overfitting.