Supervised - ensemble learning

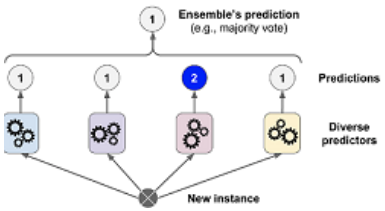
Definition

Definition: aggregating a group of predictors, called **ensemble**, using an algorithm called **ensemble method**. Often used at the end of projects to combine the best estimators and see if there is improvement.

Good ensembles

- work with **weak** learners, provided there are enough
- if classifiers are **independent**, not the case if they are trained on the same set
- when the predictors are **independent**, pick different algorithms to achieve this

Hard voting classifier



majority rules.

Soft voting

Can only be used with classifiers that output probabilities, ensemble predicts the class with the **highest average probability**.

Bagging and pasting

methods to train diverse classifiers for the predictors

- use different algorithms (knn, rf...)
- use the same algorithm but train it on different parts of the training set
  - **b**ootstrap **agg**regat**ing**, sampling with replacement
  - **pasting**, sampling without replacement

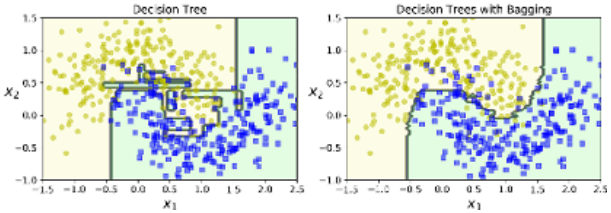
Only bagging allows the same instance to be sampled multiple times for the same predictor. Aggregation method: **most frequent value** for classifiers and the **mean** for regression.

Bagging

Observations on bagging and boosting

- each predictor has higher bias compared to being trained on the full training set, but the aggregation reduces both bias and variance.
- the ensemble generally has similar bias but lower variance compared to a single predictor
- training and prediction can be performed in parallel for scalability.

Decision boundary with bagging



often generalizes better.

Bagging and pasting comparison

The objective of bootstrapping is to create more diverse training sets.

- bagging higher bias
- pasting higher variance

Bagging often results in better models and is more used.
out-of-bag instances: those instances never used in the training of a predictor, can use to evaluate it without a val. split. The ensemble can also be evaluated by averaging out the (oob) evals.

Random patches and subspaces

- A bagging classifier might also perform features sampling ⇒ each predictor will be trained on a random subset of the input features
- Also in this case we specify :
  - the number of features to sample
  - sampling with or without replacement
- Procedure very useful when dealing with high dimentional data (i.e., images)
- **Random Patches method**: sampling both training instances and features.
- **Random Subspaces method**: keeps all training instances, but sampling features.
- Sampling features results in even more predictor diversity, trading a bit more bias for a lower variance.

Random forest

A Random Forest is an ensemble of Decision Trees, generally trained via the bagging method (or sometimes pasting), typically on a sample set whose dimension is equal to the size of the training set.

The Random Forest algorithm introduces extra randomness when growing trees; instead of searching for the very best feature when splitting a node, it searches for the best feature among a random subset of features.

This results in a greater tree diversity, which (once again) trades a higher bias for a lower variance, generally yielding an overall better model.

Extra trees

Avoid wasting time looking for best splitting threshold by providing a random threshold for each feature, much faster to train, trades more bias for lower variance. A priori hard to say which is better, cross validate and parameter tune.

Feature importance

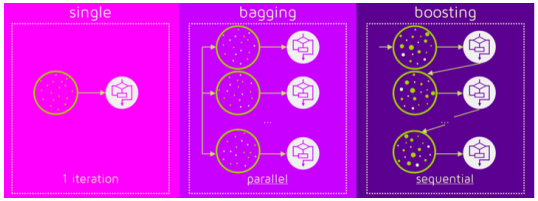
Can be seen by how much, on average (weighted), tree nodes across the forest use the feature to reduce impurity

Boosting types

Most popular boosting metodologies

1. AdaBoost
2. Gradient Boosting

Boosting



Ada booster

Each model tries to correct the previous model's errors, by updating the instance weights. ! See the quiz exercise for the formulae. In case of over fitting you can reduce the number of estimators or regularize on the base estimator.

Ada booster formulae

$$r_j = \frac{\sum_{i=1}^{m} w^{(i)}}{\sum_{i=1}^{m} w^{(i)}} \qquad \alpha_j = \eta log \frac{1 - r_j}{r_j}$$

$$w^{(i)} \leftarrow \begin{cases} w^{(i)} & \text{if } \hat{y}_j^{(i)} = y^{(i)} \\ w^{(i)} exp(\alpha_j) & \text{if } \hat{y}_j^{(i)} \neq y^{(i)} \end{cases}$$

for $i = 1, 2, ..., m$

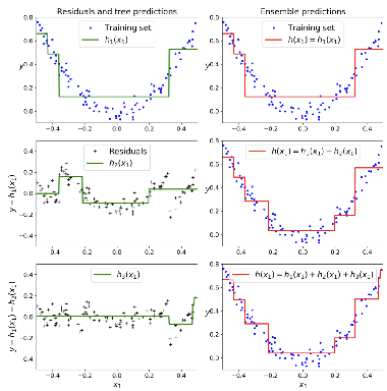
The misclassified instances are boosted

Then all the instance weights are normalized: divided by $\sum_{i=1}^{m} w^{(i)}$

Gradient boosting

Each model tries to correct the previous model's errors, like Ada boosting, but in this case it tries to fit the new prediction to the residuals of the previous prediction

Gradient boosting in action with decision tree



Tuning

learning rate scales the contribution of each tree, lower learning rate means more predictors and slower convergence, but in general better generalization, decreasing it is part of a regularization technique called shrinkage. Early stopping is used to find the number of optimal predictors, either training a large amount and working backwards or stopping when the validation error does not improve after a selected number of epochs.

Stochastic gradient boosting

As usual higher bias for lower variance by selecting only a fraction of the training instances for each tree.