Part I

Essential ensemble methods

Chapter 1

Homogeneous parallel ensembles: Bagging and random forests

1.1 Bagging: Bootstrap aggregating

Bagging, short for **bootstrap aggregating**, was introduced by Leo Breiman in 1996. The name refers to how bagging achieves ensemble diversity (through bootstrap sampling) and performs ensemble prediction (through model aggregating).

1.2 Random forests

Random forests use a modified tree learning algorithm, which first randomly samples features before creating a decision node. The resulting tree is a **randomized decision tree**, which is a new type of base estimator.

Thus, random forests contain two types of randomization: (1) bootstrap sampling, similar to bagging; and (2) random feature sampling for learning randomized decision trees.

1.2.1 Feature importances

One benefit of using random forests is that they also provide a natural mechanism for scoring features based on their importance. This means that we can rank features to identify the most important ones and drop less effective features, thus performing feature selection!

1.3 More homogeneous parallel ensembles

We've seen two important parallel homogeneous ensemble methods: bagging and random forests. Let's now explore a few variants that were developed for large data sets (e.g., recommendation systems) or high-dimensional data (e.g., image or text databases). These include bagging variants such as pasting, random subspaces and random patches, and an extreme random forest variant called Extra Trees. All these methods introduce randomization in different ways to ensure ensemble diversity.

1.3.1 Pasting

Bagging uses bootstrap sampling, or sampling with replacement. If, instead, we sample subsets for training **without replacement**, we have a variant of bagging known as **pasting**. **Pasting was designed for very large data sets, where sampling with replacement isn't necessary**. Instead, because training full models on data sets of such scale is difficult, pasting aims to take small pieces of the data by sampling without replacement.

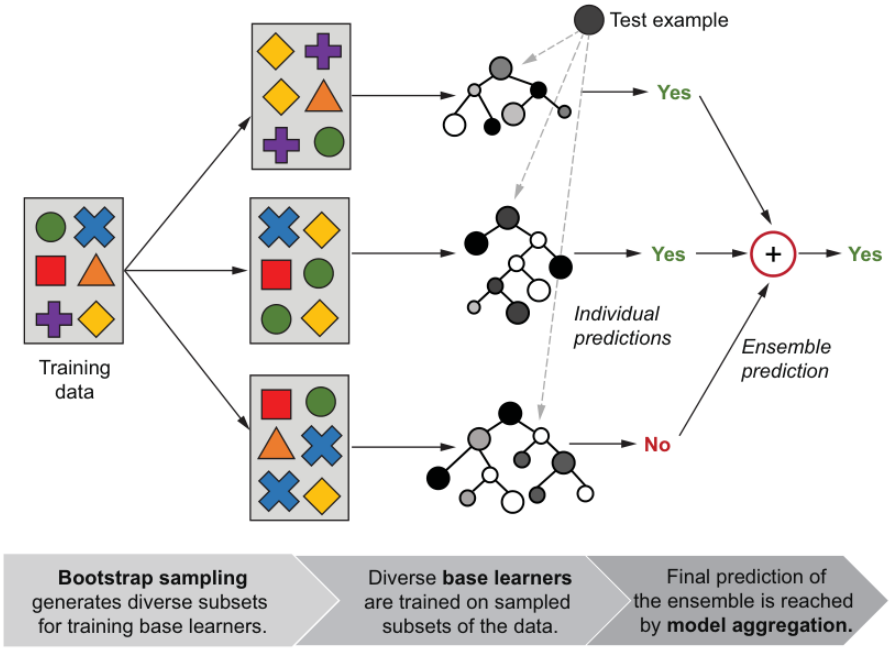


Figure 1.1: Bagging, illustrated. Bagging uses bootstrap sampling to generate similar but not exactly identical subsets (observe the replicates here) from a single data set. Models are trained on each of these subsets, resulting in similar but not exactly identical base estimators. Given a test example, the individual base-estimator predictions are aggregated into a final ensemble prediction. Also observe that training examples may repeat in the replicated subsets; this is a consequence of bootstrap sampling.

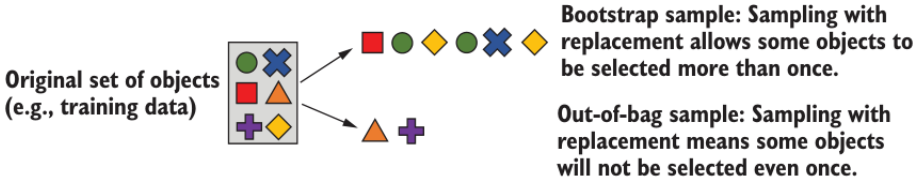


Figure 1.2: Bootstrap sampling illustrated on a data set of six examples. By sampling with replacement, we can get a bootstrap sample size of six, containing only four unique objects but with repeats. Performing bootstrap sampling several times produces several replicates of the original data set—all of them with repeats.

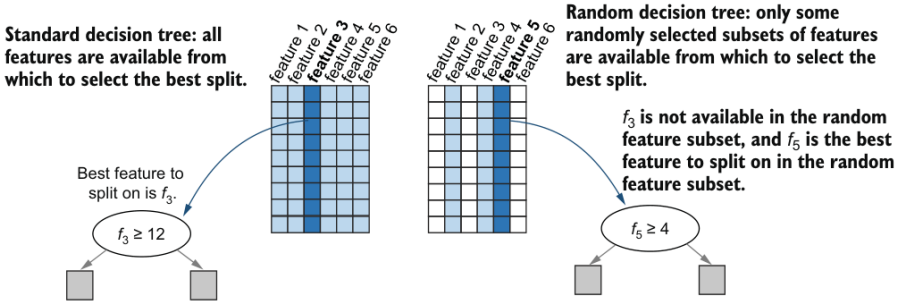


Figure 1.3: Random forests use a modified tree learning algorithm, where a random feature subset is first chosen before the best splitting criterion for each decision node is identified. The unshaded columns represent features that have been left out; the lightly shaded columns represent available features from which the best feature is chosen, shown in the darkly shaded columns.

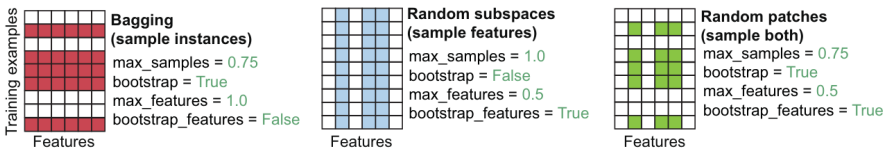


Figure 1.4: Bagging compared to random subspaces and random patches. The unshaded rows and columns represent training examples and features, respectively, that have been left out.

TIP

BaggingClassifier can easily be extended to perform pasting by setting bootstrap=False and making it subsample small subsets for training by setting max_samples to a small fraction, say max_samples=0.05.

1.3.2 Random subspaces and random patches

The key difference between random forests and bagging variants, such as random subspaces and random patches, is where the feature sampling occurs. Random forests exclusively use randomized decision trees as base estimators. Specifically, **they perform feature sampling inside the tree learning algorithm** each time they grow the tree with a decision node.

Random subspaces and random patches, on the other hand, aren't restricted to tree learning and can use any learning algorithm as a base estimator. They **randomly sample features once outside before calling the base-learning algorithm for each base estimator.**

1.3.3 Extra Trees

Extremely randomized trees take the idea of randomized decision trees to the extreme by selecting not just the splitting variable from a random subset of features but also the splitting threshold!

Extremely randomized decision-tree learning also looks at a random subset of features to determine the best f_k. But to be even more efficient, it selects a random splitting threshold. Note that extremely randomized decision trees are yet another type of base learner used for ensembling.

This extreme randomization is so effective, in fact, that we can construct an ensem- ble of extremely randomized trees directly from the original data set without bootstrap sampling! This means that we can construct an Extra Trees ensemble very efficiently.

TIP

In practice, Extra Trees ensembles are well suited for high-dimensional data sets with a large number of continuous features.

scikit-learn provides an ExtraTreesClassifier that supports OOB estimation and parallelization, much like BaggingClassifier and RandomForestClassifier. Note that Extra Trees typically do not perform bootstrap sampling (bootstrap=False, by default), as we're able to achieve base-estimator diversity through extreme randomization.

CAUTION

scikit-learn provides two very similarly named classes: sklearn. tree.ExtraTreeClassifier and sklearn.ensemble.ExtraTreesClassifier. The tree.ExtraTreeClassifier class is a base-learning algorithm and should be used for learning individual models or as a base estimator with ensemble methods. ensemble.ExtraTreesClassifier is the ensemble method discussed in this section. The difference is in the singular usage of "Extra Tree" (ExtraTreeClassifier is the base learner) versus

the plural usage "Extra Trees" (ExtraTreesClassifier is the ensemble method).

1.4 Case study: Breast cancer diagnosis

1.4.1 Bagging, random forests, and Extra Trees

ENSEMBLE SIZE VS. ENSEMBLE PERFORMANCE

BASE LEARNER COMPLEXITY VS. ENSEMBLE PERFORMANCE

One key consideration in determining the depth of the base decision trees is computational efficiency. Training deeper and deeper trees will take more and more time without producing a significant improvement in predictive performance.

CAUTION

Note that feature importances will often change between runs owing to randomization during tree construction. Note also that if two features are highly correlated, random forests will often distribute the feature importance between them, leading to their overall weights appearing smaller than they actually are.

1.5 Summary

- Parallel homogeneous ensembles promote ensemble diversity through randomization: random sampling of training examples and of features, or even introducing randomization in the base-learning algorithm.
- Bagging is a simple ensemble method that relies on (1) bootstrap sampling (or sampling with replacement) to generate diverse replicates of the data set and training diverse models, and (2) model aggregation to produce an ensemble prediction from a set of individual base learner predictions.
- Bagging and its variants work best with any unstable estimators (unpruned decision trees, support vector machines [SVMs], deep neural networks, etc.), which are models of higher complexity and/or nonlinearity.
- Random forest refers to a variant of bagging specifically designed to use randomized decision trees as base learners. Increasing randomness increases ensemble diversity considerably, allowing the ensemble to decrease the variability and smooth out predictions.
- Pasting, a variant of bagging, samples training examples without replacement and can be effective on data sets with a very large number of training examples.
- Other variants of bagging, such as random subspaces (sampling features) and random patches (sampling both features and training examples), can be effective on data sets with high dimensionality.
- Extra Trees is another bagging-like ensemble method that is specifically designed to use extremely randomized trees as base learners. However, **Extra Trees doesn't use bootstrap sampling** as additional randomization helps in generating ensemble diversity.
- Random forests provide feature importances to rank the most important features from a predictive standpoint.

Chapter 2

Heterogeneous parallel ensembles: Combining strong learners

In this chapter, we continue exploring parallel ensemble methods, but this time focusing on **heterogeneous ensembles**. Heterogeneous ensemble methods use different base-learning algorithms to directly ensure ensemble diversity.

Heterogeneous ensembles come in two flavors, depending on how they combine individual base-estimator predictions into a final prediction:

- **Weighting** methods—These methods assign individual base-estimator predictions a weight that corresponds to their strength. Better base estimators are assigned higher weights and influence the overall final prediction more. The predictions of individual base estimators are fed into a predetermined combination function, which makes the final predictions.
- **Meta-learning** methods—These methods use a learning algorithm to combine the predictions of base estimators; the predictions of individual base estimators are treated as metadata and passed to a second-level meta-learner, which is trained to make final predictions.

2.1 Fitting base estimators

Unlike homogeneous ensembles, we can use any number of different learning algorithms and parameter settings to train base estimators. The key is to ensure that we choose learning algorithms that are different enough to produce a diverse collection of estimators. The more diverse our set of base estimators, the better the resulting ensemble will be.

2.1.1 Individual predictions of base estimators

> **Note**
>
> Most classifiers in scikit-learn can return the probability of a label rather than the label directly. Some of them, such as SVC, should be explicitly told to do so (notice that we set probability=True when initializing SVC), while others are natural probabilistic classifiers and can represent and reason over class probabilities. These probabilities represent each base estimator's confidence in its prediction.

Prediction probabilities are often called soft predictions. Soft predictions can be converted to hard (0–1) predictions by simply picking the class label with the highest probability.

Of course, the most important step is how we combine these individual predictions: by weighting or by meta-learning.

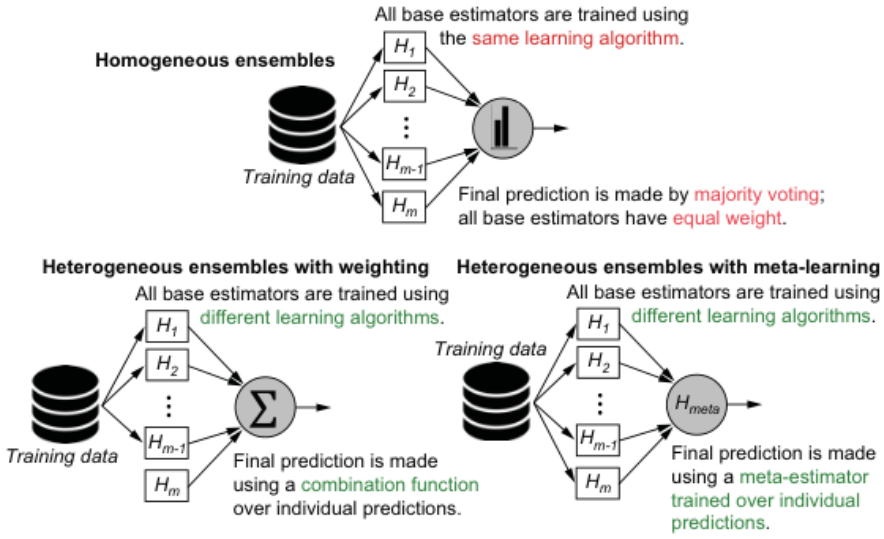


Figure 2.1: Homogeneous ensembles (Chapter 1), such as bagging and random forests, use the same learning algorithm to train base estimators, and they achieve ensemble diversity through random sampling. Heterogeneous ensembles (this chapter) use different learning algorithms to achieve ensemble diversity.

CAUTION

The prediction function just discussed is specifically written for two- class, that is, binary classification, problems. It can be extended to multiclass problems if care is taken to store the prediction probabilities for each class. That is, for multiclass problems, you'll need to store the individual prediction probabilities in an array of size `n_samples * n_estimators * n_classes`.

2.2 Combining predictions by weighting

We should do in a manner consistent with the intuition, such that the final prediction is influenced more by the stronger classifiers and less by the weaker classifiers.

2.2.1 Majority vote

Majority voting can be viewed as a weighted combination scheme in which each base estimator is assigned an equal weight; that is, if we have m base estimators, each base estimator has a weight of $w_{clf} = \frac{1}{m}$. The (weighted) predictions of the individual base estimators are combined using the majority vote.

2.2.2 Accuracy weighting

ACCURACY WEIGHTS USING A VALIDATION SET

Once we've trained each base classifier (clf), we evaluate its performance on a validation set. Let α_t be the validation accuracy of the t-th classifier, H_t. The weight of each base classifier is then computed as follows:

$$w_t = \frac{\alpha_t}{\sum_{t=1}^{m} \alpha_t} \tag{2.1}$$

Given a new example to predict $\boldsymbol{x}$, we can get the predictions of the individual classifiers, y_t. Now,

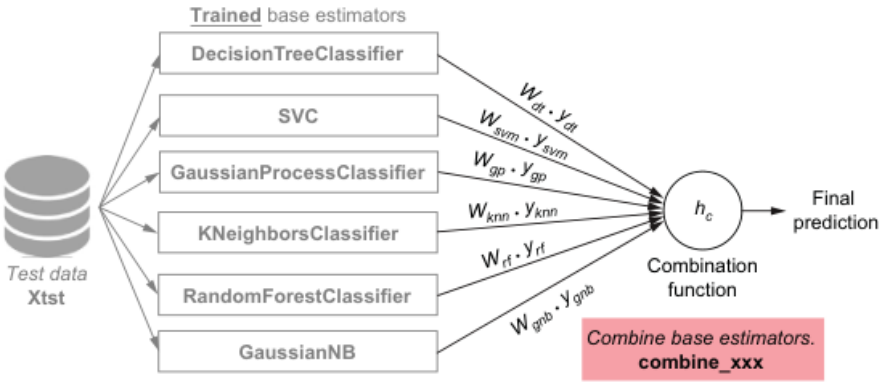


Figure 2.2: Each base classifier is assigned an importance weight that reflects how much its opinion contributes to the final decision. Weighted decisions of each base classifier are combined using a combination function.

the final prediction can be computed as a weighted sum of the individual predictions:

$$y_{final} = w_1 y_1 + w_2 y_2 + \cdots + w_m * y_m = \sum_{t=1}^{m} w_t y_t \tag{2.2}$$

2.2.3 Entropy weighting

The entropy weighting approach is another performance-based weighting approach, except that it uses entropy as the evaluation metric to judge the value of each base estimator. Entropy is a measure of uncertainty or impurity in a set; a more disorderly set will have higher entropy.

Entropy

Entropy, or information entropy to be precise, was originally devised by Claude Shan- non to quantify the "amount of information" conveyed by a variable. This is deter- mined by two factors: (1) the number of distinct values the variable can take, and (2) the uncertainty associated with each value.
Entropy quantifies this notion of uncertainty across various outcomes. Entropy-based measures are commonly used during decision-tree learning to greedily identify the best variables to split on and are used as loss functions in deep neural networks.

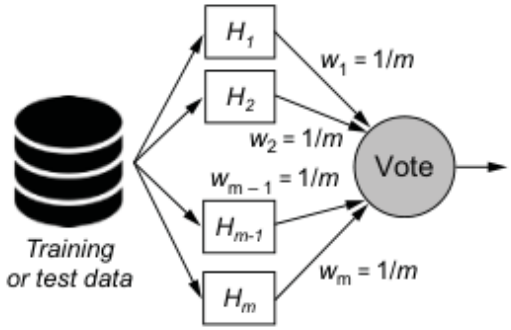
ENTROPY WEIGHTING WITH A VALIDATION SET

Let E_t be the validation entropy of the t-th classifier, H_t. The weight of each base classifier is

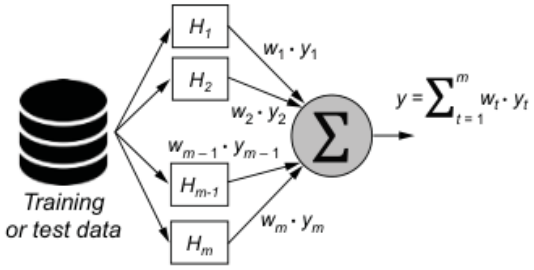
$$w_t = \frac{\frac{1}{E_t}}{\sum_{t=1}^{m} \frac{1}{E_t}} \tag{2.3}$$

There are two key differences between entropy weighting and accuracy weighting:

- The accuracy of a base classifier is computed using both the true labels ytrue and the predicted labels ypred. In this manner, the accuracy metric measures how well a classifier performs. A classifier with high accuracy is better.
- The entropy of a base classifier is computed using only the predicted labels ypred, and the entropy metric measures how uncertain a classifier is about its predictions. A classifier with low entropy (uncertainty) is better. Thus, individual base classifier weights are inversely proportional to their corresponding entropies.



(a) Combining by majority voting. Bagging can be viewed as a simple weighting method applied to a homogeneous ensemble. All classifiers have equal weights, and the combination function is the majority vote. We can adopt the majority voting strategy for heterogeneous ensembles as well.



(b) Combining by performance weighting. Each classifier is assigned a weight proportional to its accuracy. The final prediction is computed as a weighted combination of the individual predictions.

Figure 2.3: Combining predictions by weighting

2.2.4 Dempster-Shafer combination

DST FOR LABEL FUSION

ST uses a number between 0 and 1 to indicate belief in a proposition, such as "the test example x belongs to Class 1." This number is known as a basic probability assignment (BPA) and expresses the certainty that the text example $\boldsymbol{x}$ belongs to Class 1. BPA values closer to 1 characterize decisions made with more certainty. The BPA allows us to translate an estimator's confidence into a belief about the true label.

We've seen four methods of combining predictions into one final prediction. Three use the predictions directly, while one use prediction probabilities. We can visualize the decision boundaries produced by these weighting methods, as shown in Figure 2.4.

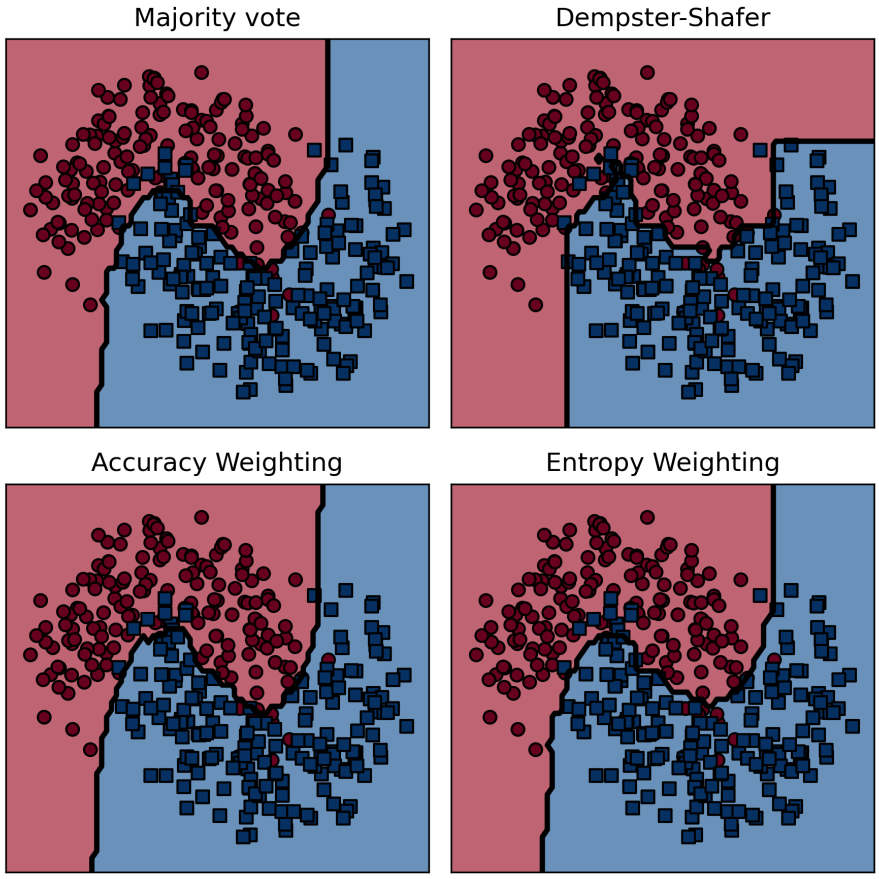


Figure 2.4: Decision boundaries of different weighting methods