Lab 8

Decision Trees

UPC telecom BCN
Escola Tècnica Superior d'Enginyeria de Telecomunicació de Barcelona
UNIVERSITAT POLITÈCNICA DE CATALUNYA

Introduction

Objectives

- Learn how to use decision trees to solve a classification problem
- Learn to prune a decision tree
- Learn to train a random forest classifier
- Solve the Ionosphere classification problem with decision trees and random forests
- Solve the MNIST classification problem with random forests.

Decision Trees in sklearn

Class: `DecisionTreeClassifier`

Scikit-Learn uses the **CART algorithm**, which produces only binary tres. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node.

The main parameters are:

criterion: The function to measure the quality of a split. Supported criteria are "gini" (default) for the Gini impurity and "entropy" for the information gain.

max_depth: The maximum depth of the tree. If None (default), then nodes are expanded until all leaves are pure or until all leaves contain less than **min_samples_split** samples.

min_samples_split: The minimum number of samples required to split an internal node (default=2)

min_samples_leaf The minimum number of samples required to be at a leaf node (default 1). A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

class_weight Weights associated with classes (default None).

Decision Trees in sklearn

```
clf = sklearn.tree.DecisionTreeClassifier(criterion='entropy' )
clf.fit(X, y)
```

#Plot the tree

```
sklearn.tree.plot_tree(clf, ax=fig.subplots());
```

#Export in textual format

```
t = export_text(clf)
```

#Check parameters used

```
clf.get_params()
```

#Feature importances

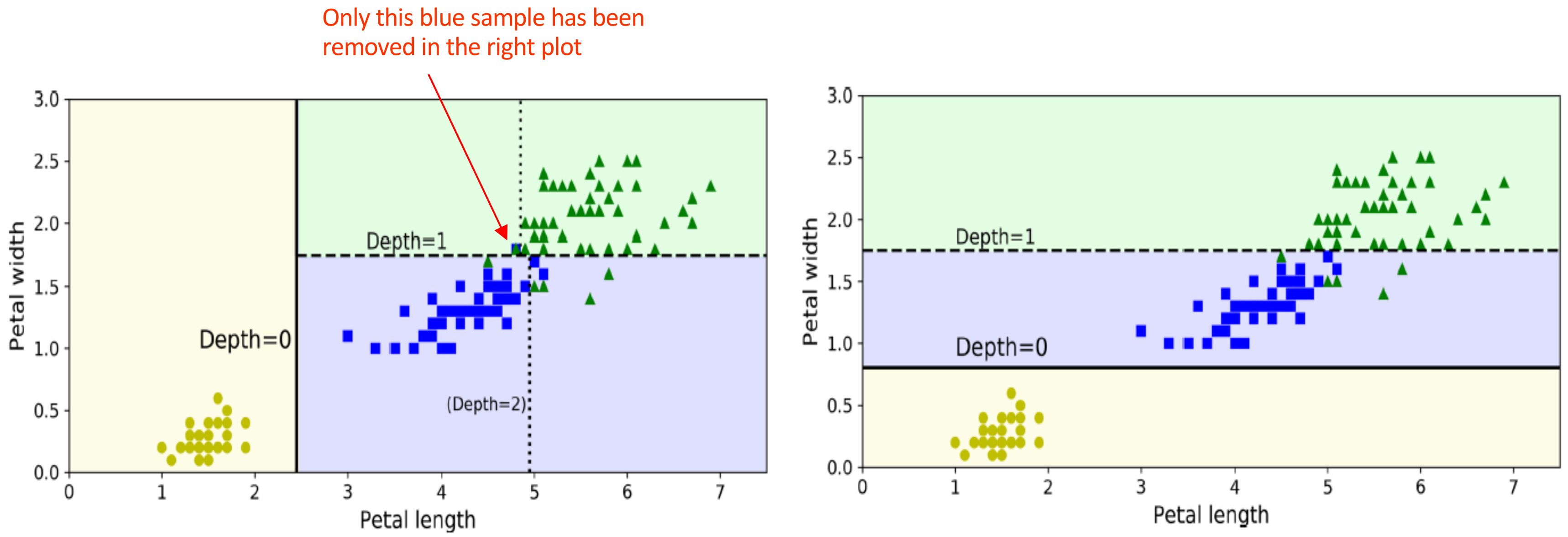
```
clf.feature_importances_
```

#Parameters for regularization

```
max_depth
min_simples_split
min_simples_leaf
min_weight_fraction_leaf
max_leaf_nodes
max_features
```

Bias and variance

Decision trees are large variance predictors: a slight modification in the dataset can dramatically change the decision boundaries.



Random Forests in sklearn

To improve the predictive accuracy and control over-fitting, multiple decision tree classifiers can be trained and later combined.

In random forests, randomness is introduced in the classifier construction by :

1. Each tree is built from a **sample drawn with replacement from the training set**.
2. When splitting each node during the construction of a tree, the best split is found from a **random subset from the input features**.

Random forests achieve a reduced variance by combining diverse trees, sometimes at the cost of a slight increase in bias. In practice the variance reduction is often significant hence yielding an overall better model

Random foeste class: `RandomForestClassifier`

```
clf =
sklearn.ensemble.RandomForestClassifier(criterion='entropy',
min_samples_leaf=5)
clf.fit(X, y)
```

Random forest

One of the most powerful techniques for shallow learning to date.

We generate an ensembles of trees by

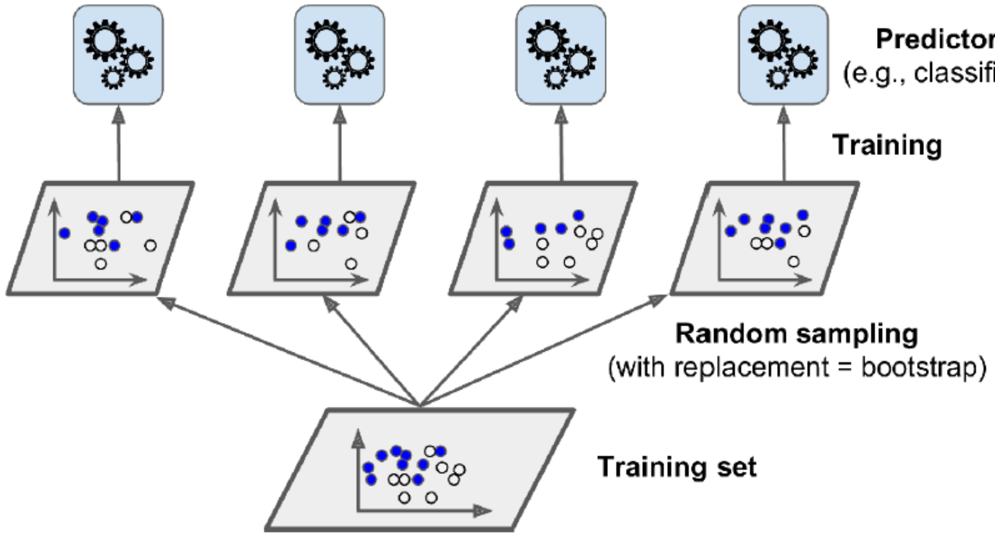
1. Using bagging plus
2. On each node we select a random subset of m features to decide the best. The subset changes randomly from each node to the next.

The value of m and the number of trees M are hyperparameter.

We can keep some of the explainability of decisions trees by checking how many times over all trees some features are selected weighted by the number of samples in the node.

Bagging

The ensemble of classifiers is trained using different subsets of data, possibly with replacement. It reduces variance.


Predictors
(e.g., classifiers)
Training
Random sampling
(with replacement = bootstrap)
Training set