CITS5508 Machine Learning
Lectures for Semester 1, 2021
Lecture Week 5: Book Chapter 6, Decision Trees

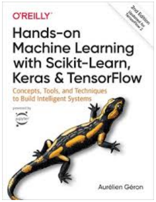
Dr Du Huynh (Unit Coordinator and Lecturer)
UWA

2021

Today

Chapter 6.

Hands-on Machine Learning with Scikit-Learn & TensorFlow


O'REILLY
2nd Edition
Hands-on Machine Learning with Scikit-Learn, Keras & TensorFlow
Concepts, Tools, and Techniques to Build Intelligent Systems
Aurélien Géron

Decision Trees

In this chapter, we will explain the core concepts of Decision Trees (DTs), how to use them, and how they work.

Like SVMs, DTs are versatile ML algorithms that can perform both classification and regression tasks, and even multioutput tasks. They are very powerful algorithms, capable of fitting complex datasets. E.g., recall the housing data example from Ch2.

DTs are also the fundamental components of Random Forests (RF, Ch7), which are among the most powerful ML algorithms available today.

Topics

Here are the main topics we will go cover:

- Training and Visualising a Decision Tree
- Making Predictions
- Estimating Class Probabilities
- The CART Training Algorithm
- Computational Complexity
- Gini Impurity or Entropy
- Regularization Hyperparameters
- Regression
- Instability

Training and Visualising a Decision Tree

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
iris = load_iris()
X = iris.data[:, 2:] # petal length and width
y = iris.target
tree_clf = DecisionTreeClassifier(max_depth=2)
tree_clf.fit(X, y)
```

Then can use `export_graphviz()` method to output a graph definition file and display using graphviz package.

Decision Tree Visualized

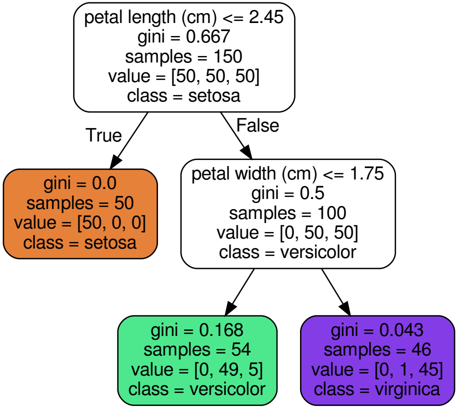


Figure 6-1. Iris Decision Tree

(**Note:** To run this chapter's Python code of the author, you must install an additional package using the command: `conda install python-graphviz`.)

Making Predictions

- Use the tree to make a decision for a single instance by following the sequence of questions and answers down from the node, left or right depending on the answer.
- Class of leaf node is the output class.
- Tree nodes also tell us number of samples allocated to each node, and broken down by class.
- *Gini impurity* (low if most instances from just one class):

$$G_i = 1 - \sum_{k=1}^{n} p_{i,k}^2$$

where $p_{i,k}$ is the ratio of class k instances among the training instances in the i^{th} node.

A "White-Box" Classifier: (i.e. human understandable)

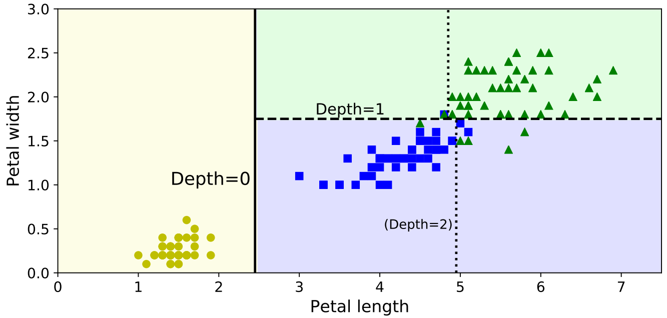


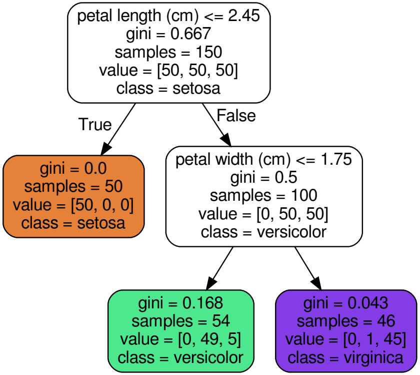
Figure 6-2. Decision Tree decision boundaries

(Refer to Figure 6-1)

- *White box models* – e.g., decision trees
- *Black box models* – e.g., random forests, neural networks

Estimating Class Probabilities

You can use the class breakdowns in each leaf node to get estimates of the probabilities of class membership for each instance that ends up at a certain leaf.



```
>>> tree_clf.predict_proba([[5, 1.5]])
array([[ 0.  , 0.90740741, 0.09259259]])
>>> tree_clf.predict([[5, 1.5]])
array([1])
```

The CART Training Algorithm

Scikit-Learn uses the Classification And Regression Tree (CART) algorithm to train Decision Trees (also called "growing" trees). The idea is really quite simple: the algorithm first splits the training set in two subsets using a single feature k and a threshold t_k (e.g., "petal length" ≤ 2.45 cm?). How does it choose k and t_k? It searches for the pair (k, t_k) that produces the purest subsets (weighted by their size). The cost function that the algorithm tries to minimize is given by

$$J(k, t_k) = G_{\text{left}} + G_{\text{right}}$$

Note: we use a *Greedy Algorithm* to get a reasonable solution, not necessarily optimal (which would take too long to find).

Computational Complexity

Making predictions requires traversing the Decision Tree from the root to a leaf. Decision Trees are generally approximately balanced, so traversing the Decision Tree requires going through roughly $O(\log_2(m))$ nodes. Since each node only requires checking the value of one feature, the overall prediction complexity is just $O(\log_2(m))$, independent of the number of features. So predictions are very fast, even when dealing with large training sets. However, the training algorithm compares all features (or less if `max_features` is set) on all samples at each node. This results in a training complexity of $O(nm \log_2(m))$. For small training sets (less than a few thousand instances), Scikit-Learn can speed up training by presorting the data (set `presort=True`), but this slows down training considerably for larger training sets.

(m = number of leaf nodes; n = number of features)

Gini Impurity or Entropy

- By default, the Gini impurity measure is used
- *entropy* is an alternative (by setting the `criterion` hyperparameter to "`entropy`")
- From thermodynamics: *entropy* is a measure of molecular disorder
- Later more widespread use, e.g., in *Shannon's information theory* it measures the average information content of a message
- frequently used as an impurity measure in ML: a set's entropy is zero when it contains instances of only one class.

Gini Impurity or Entropy

- Definition of Entropy H_i of node i:

$$H_i = -\sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \log_2(p_{i,k})$$

- Most of the time it does not make a big difference which one you use: they lead to similar trees.
- Gini impurity is slightly faster to compute, so good default.
- But when they differ, entropy tends to produce slightly more balanced trees.

Regularization Hyperparameters

- DTs make few assumptions about the training data: they are thus referred to as *nonparametric models*, very (maybe too) adaptive.
- To avoid overfitting, we need to restrict DTs' freedom during training, e.g., using regularization or set the `max_depth` hyperparameter smaller.
- `DecisionTreeClassifier` has other similar parameters: `min_samples_split`, `min_samples_leaf`, `min_weight_fraction_leaf`, `max_leaf_nodes`.
- Increasing `min_*` hyperparameters or reducing `max_*` hyperparameters regularizes the model.

Example Regularization of a Decision Tree

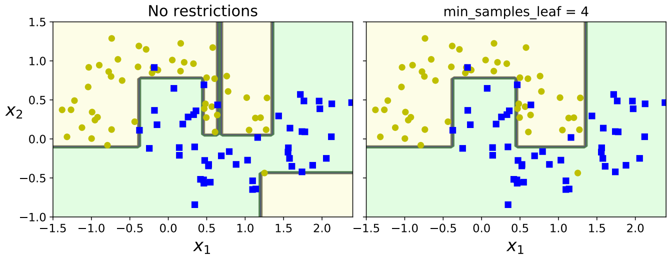


Figure 6-3. Regularization using min_samples_leaf

Two Decision Trees trained on the moons dataset are shown above.

- Left: Decision tree trained with the default hyperparameters (i.e., no restrictions)
- Right: Decision tree trained with `min_samples_leaf=4`.

It is quite obvious that the model on the left is overfitting, and the model on the right will probably generalize better.

Regression

Example tree trained on a noisy quadratic (`max_depth=2`).

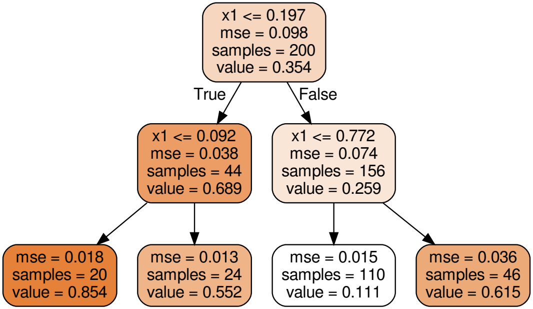


Figure 6-4. A Decision Tree for regression

```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X, y)
```

… Or with depth 3 …

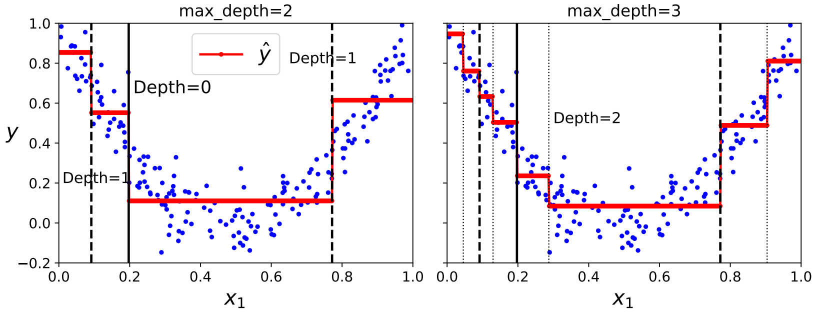


Figure 6-5. Predictions of two Decision Tree regression models

Training consists of trying to minimize the MSE.

Prone to overfitting so use regularization

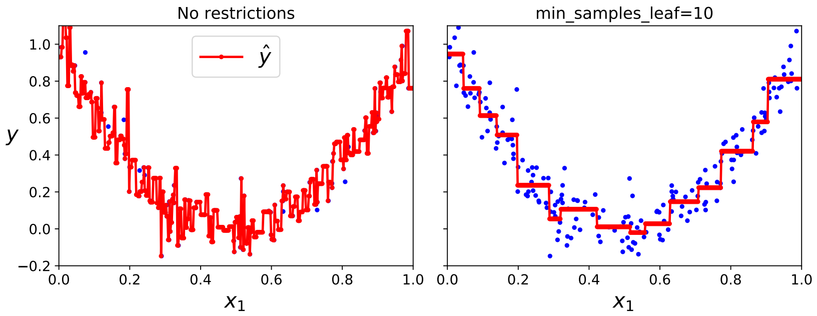


Figure 6-6. Regularizing a Decision Tree regressor

- Left: without any regularization (i.e., using the default hyperparameters). Overfitting the training data is obvious.
- Right: minimum number of samples per leaf node is set to 10, resulting in a much more reasonable model.

Instability: e.g., sensitivity to rotation

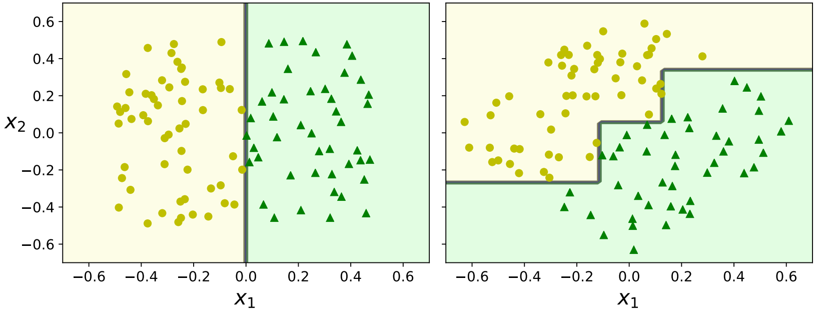


Figure 6-7. Sensitivity to training set rotation

(later in the unit we see PCA which can help)

Instability: DTs also sensitive to small changes in data

E.g., retrain DT with just one instance ($petal\_length$=4.8cm,$petal\_width$=1.8cm) removed.

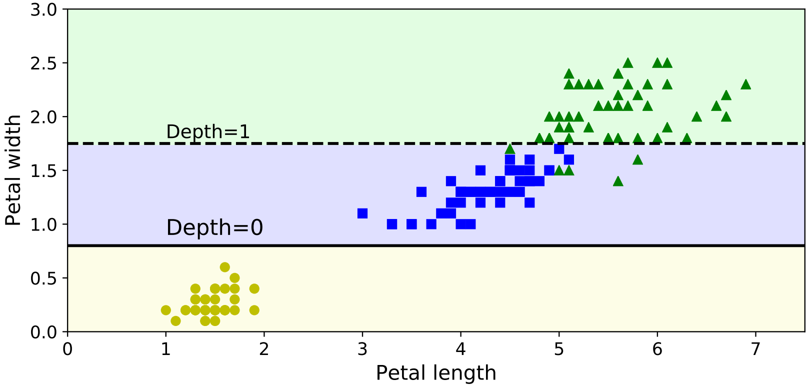


Figure 6-8. Sensitivity to training set details

(see random forests (RF) to overcome this)

Summary for Chapter 6

- Training and Visualising a Decision Tree
- Making Predictions
- Estimating Class Probabilities
- The CART Training Algorithm
- Computational Complexity
- Gini Impurity or Entropy
- Regularization Hyperparameters
- Regression
- Instability

For next week

Work through the lab sheet and attend the supervised lab. The Unit Coordinator or a casual Teaching Assistant will be there to help.

Read Chapter 7 on Ensemble Learning and Random Forests.

And that's all for the fifth lecture.

Have a good week.