Machine Learning Introduction

Supervisor
PD Dr. Friedhelm Schwenker
Institute of Neural Information Processing
Ulm University
Students
Abdelrahman Mahmoud And Mohamed Ashry

April 1, 2020

Decision Trees

- Like SVMs, Decision Trees are versatile Machine Learning algorithms that can per- form both classification and regression tasks, and even multioutput tasks. They are very powerful algorithms, capable of fitting complex datasets.
- In this chapter we will start by discussing how to train, visualize, and make predic- tions with Decision Trees. Then we will go through the CART training algorithm used by Scikit-Learn, and we will discuss how to regularize trees and use them for regression tasks. Finally, we will discuss some of the limitations of Decision Trees.

Training and Visualizing a Decision Tree

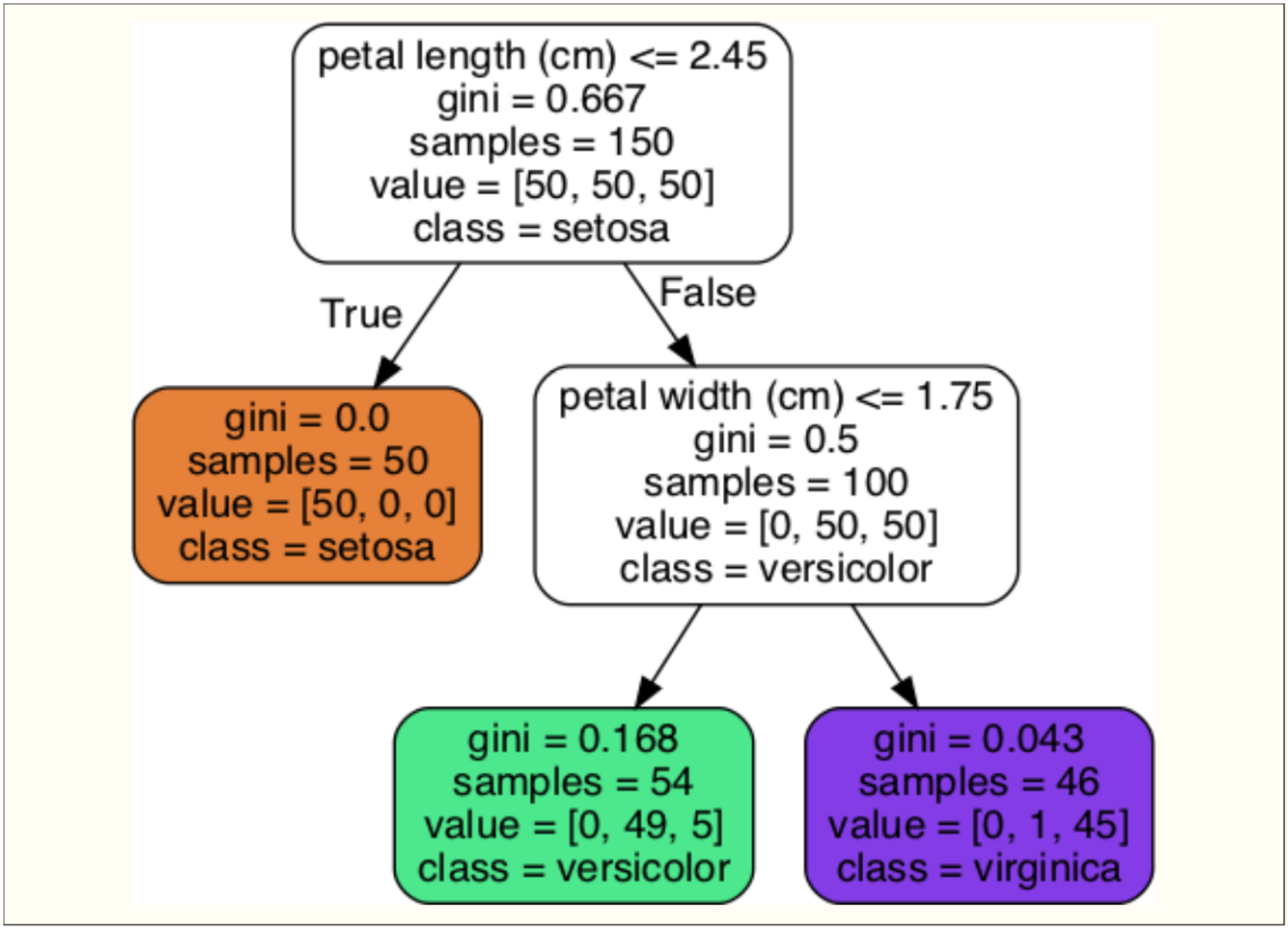
```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier
```

```
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)
```

You can visualize the trained Decision Tree by first using the `export_graphviz()` method to output a graph definition file called *iris_tree.dot*:

```
from sklearn.tree import export_graphviz

export_graphviz(
        tree_clf,
        out_file=image_path("iris_tree.dot"),
        feature_names=iris.feature_names[2:],
        class_names=iris.target_names,
        rounded=True,
        filled=True
    )
```

Your first decision tree looks like Figure 6-1.



Estimating Class Probabilities

- A Decision Tree can also estimate the probability that an instance belongs to a partic- ular class k: first it traverses the tree to find the leaf node for this instance, and then it returns the ratio of training instances of class k in this node.

The CART Training Algorithm

- The CART Training Algorithm Scikit-Learn uses the Classification And Regression Tree (CART) algorithm to train Decision Trees (also called "growing" trees).
- he idea is really quite simple: the algo- rithm first splits the training set in two subsets using a single feature k and a thres- hold tk (e.g., "petal length 2.45 cm"). How does it choose k and tk? It searches for the pair (k, tk) that produces the purest subsets (weighted by their size). The cost function that the algorithm tries to minimize is given by Equation 6-2.

Equation 6-2. CART cost function for classification

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

where $\begin{cases} G_{\text{left/right}} \text{ measures the impurity of the left/right subset,} \\ m_{\text{left/right}} \text{ is the number of instances in the left/right subset.} \end{cases}$

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(log2(m)) nodes.
- Since each node only requires checking the value of one feature, the overall prediction complexity is just O(log2(m)), independent of the number of features. So predictions are very fast, even when deal- ing with large training sets.
- However, the training algorithm compares all features (or less if maxfeatures is set) on all samples at each node. This results in a training complexity of O(n m log(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.

Regularization Hyperparameters

- Decision Trees make very few assumptions about the training data (as opposed to lin- ear models, which obviously assume that the data is linear, for example). If left unconstrained, the tree structure will adapt itself to the training data, fitting it very closely, and most likely overfitting it.
- Such a model is often called a nonparametric model, not because it does not have any parameters (it often has a lot) but because the number of parameters is not determined prior to training, so the model structure is free to stick closely to the data.
- In contrast, a parametric model such as a linear model has a predetermined number of parameters, so its degree of freedom is limited, reducing the risk of overfitting (but increasing the risk of underfitting).

- To avoid overfitting the training data, you need to restrict the Decision Tree's freedom during training. As you know by now, this is called regularization. The regularization hyperparameters depend on the algorithm used, but generally you can at least restrict the maximum depth of the Decision Tree. In Scikit-Learn, this is controlled by the maxdepth hyperparameter (the default value is None, which means unlimited). Reducing maxdepth will regularize the model and thus reduce the risk of overfitting

Regression

```
from sklearn.tree import DecisionTreeRegressor
tree-reg = DecisionTreeRegressor(max-depth=2)
tree-reg.fit(X, Y)
```

The resulting tree is represented on Figure 6-4.

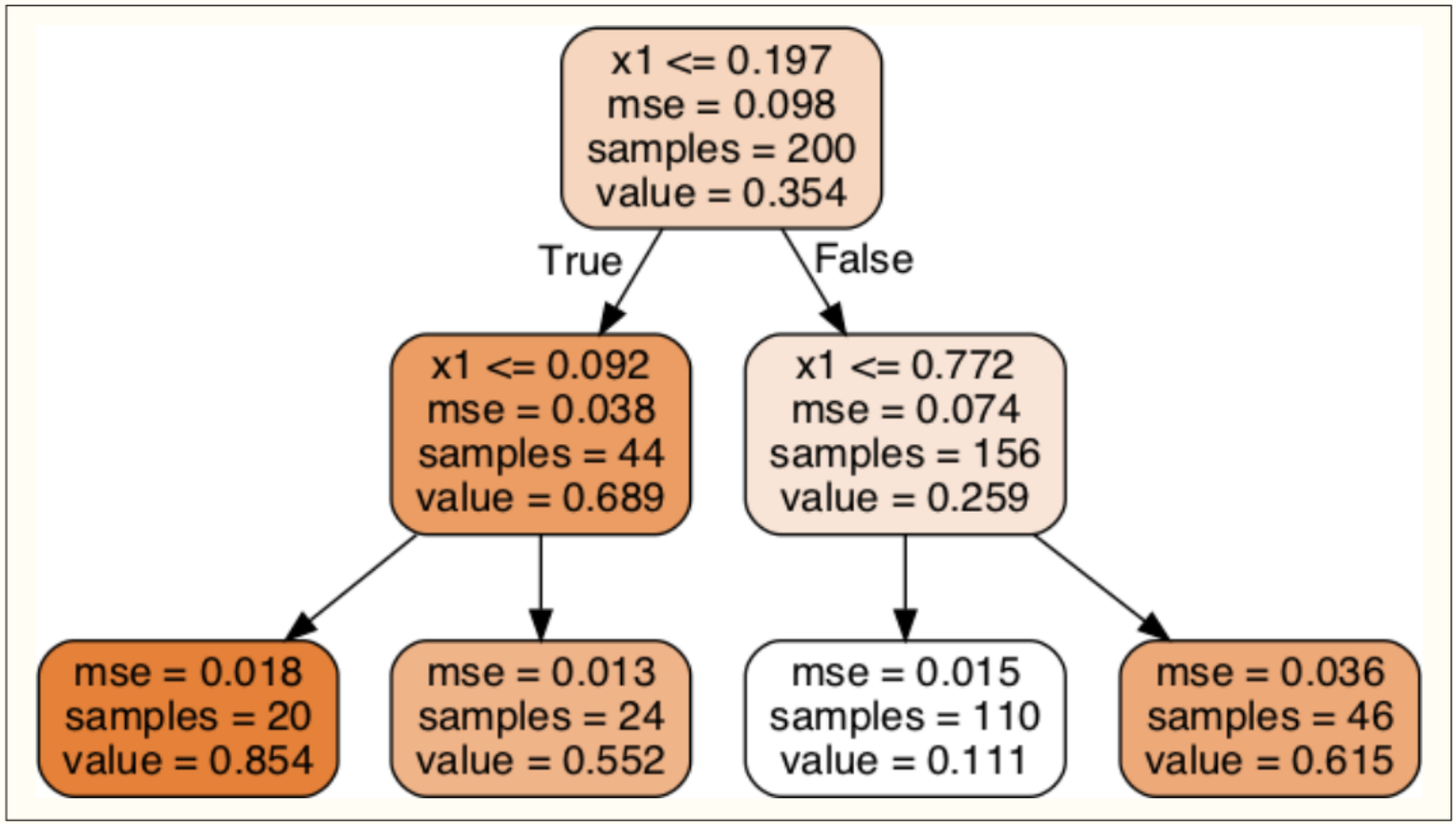


Figure 6-4. A Decision Tree for regression

Thank You !