Scikit-learn ゼミ

1.8. Decision Trees
1.8.1. Classification

新納浩幸

Decision Tree

決定木

学習アルゴリズムの1つ、
識別と回帰に使える

ノンパラメトリックな手法、
概略、if-then 規則を学習する

Decision Tree の長所

- 学習できた規則が理解しやすい
- 学習の前処理（正規化など）が必要ない
- 識別の処理時間が木のノード数の対数に比例
- 数値データでもカテゴリカルデータでも扱える
- 多値の出力も可能
- white box model である（中身がわかるモデル）
- 統計的検定でモデルの有効性を測れる
- 仮定が真のモデルに反している場合でも動く

Decision Tree の欠点

- 過学習の問題がある
- 安定していないかも、アンサンブル学習が効果あり
- 最適な木を作るのは NP-困難な問題
- XOR など学習が困難なものがある
- あるクラスのデータが多大だと、バイアスの
  かかった木ができる

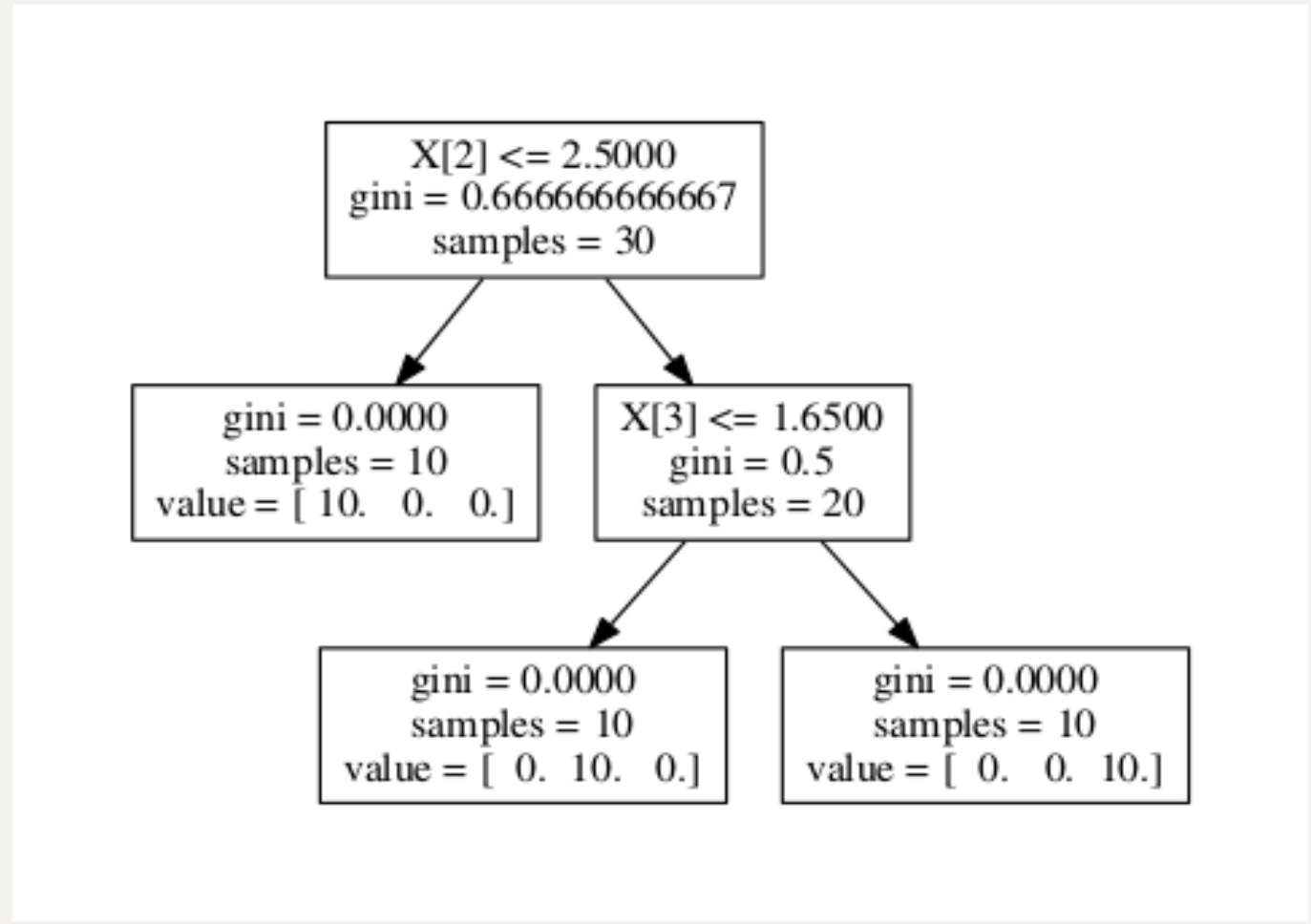
実行例

```
>>> from sklearn.datasets import load_iris
>>> from sklearn import tree
>>> iris = load_iris()
>>> clf = tree.DecisionTreeClassifier()
>>> a = iris.data
>>> b = iris.target
>>> a1 = a[range(0,10)+range(50,60)+range(100,110)]
>>> b1 = b[range(0,10)+range(50,60)+range(100,110)]
>>> clf = clf.fit(a1, b1)
>>> clf.predict(a)
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 2, 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 1, 2, 2, 2, 1, 1, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])
```

木の表示(1)

```
>>> from sklearn.externals.six import StringIO
>>> import pydot
>>> dot_data = StringIO()
>>> tree.export_graphviz(clf, out_file=dot_data)
<StringIO.StringIO instance at 0x7ff5842a9e18>
>>> graph = pydot.graph_from_dot_data(dot_data.getvalue())
>>> graph.write_pdf("iris.pdf")
True
>>>
```

木の表示(2)


X[2] <= 2.5000
gini = 0.666666666667
samples = 30
gini = 0.0000
samples = 10
value = [10. 0. 0.]
X[3] <= 1.6500
gini = 0.5
samples = 20
gini = 0.0000
samples = 10
value = [0. 10. 0.]
gini = 0.0000
samples = 10
value = [0. 0. 10.]