Decision Tree (2022), **103**, 1, *pp*. 1–7
Bài báo cáo CS115.N11.KHTN

Bài báo cáo về Thuật toán decision tree

BY LÊ THANH MINH
University of Information Technology
Khu phố 6, P.Linh Trung, Tp.Thủ Đức, Tp.Hồ Chí Minh
21520063@gm.uit.edu.vn

SƠ LƯỢC

Bài cáo cáo này sẽ thực hiện đối với yêu cầu của môn học CS115.N11.KHTN:

- Nghiên cứu về Decision Tree trong sklearn (Python). Nghiên cứu về công dụng và cách sử dụng của nó
- Xây dựng được một thuật toán Decision Tree dựa trên các tiêu chuẩn split như gini,entropy, ...
- Nộp 01 file pdf báo cáo (khuyến khích dùng LaTeX để soạn thảo), đặt tên dạng BT1_MSSV.pdf với MSSV là mã số sinh viên của mỗi bạn.
- Viết báo cáo bằng tiếng Việt, trình bày gọn gàng sạch sẽ.
- Trong báo cáo có link tới Google Colab chứa source code của demo (chế độ public)

1. DECISION TREE

1·1. *Công dụng*

Cây quyết định (DT) là một phương pháp **học có giám sát phi tham số** (supervised learning with non-parametric) được sử dụng trong bài toán phân loại và hồi quy. Mục tiêu là tạo ra một **mô hình dự đoán giá trị của mục tiêu** bằng cách đưa ra quy tắc quyết định rút ra từ các đặc trưng của dữ liệu. Một cái cây có thể được coi là một xấp xỉ không đổi từng phần.

Cây quyết định có những ưu điểm sau:

- **Đơn giản** để hiểu và giải thích. Cây cối có thể được mô hình hóa.
- Yêu cầu **chuẩn bị ít dữ liệu**. Các kỹ thuật khác thường yêu cầu chuẩn hóa dữ liệu, các biến giả cần được tạo và các giá trị trống được loại bỏ. Tuy nhiên, lưu ý rằng module sklearn này không hỗ trợ các giá trị bị thiếu.
- **Chi phí sử dụng cây thấp** (tức là dữ liệu dự đoán) vì là logarit trong số lượng tập dữ liệu được sử dụng để huấn luyện cây.
- Có thể xử lý cả dữ liệu số và phân loại . Các kỹ thuật khác thường chuyên phân tích các bộ dữ liệu chỉ có một loại biến. Xem thuật toán[1] để biết thêm thông tin.
- Có thể xử lý các vấn đề đa đầu ra.
- Sử dụng mô hình hộp màu trắng. Nếu một trường hợp quan sát được trong một mô hình, có thể dễ dàng được giải thích bằng logic boolean. Ngược lại, trong mô hình hộp đen (ví dụ: trong mạng nơ-ron nhân tạo), kết quả có thể khó diễn giải hơn.

Cây cũng có một số nhược điểm sau:

- Decision tree có thể tạo ra các cây phức tạp không có tính tổng quát hóa (**overfitting**).

[1] https://scikit-learn.org/stable/modules/tree.htmltree-algorithms-id3-c4-5-c5-0-and-cart

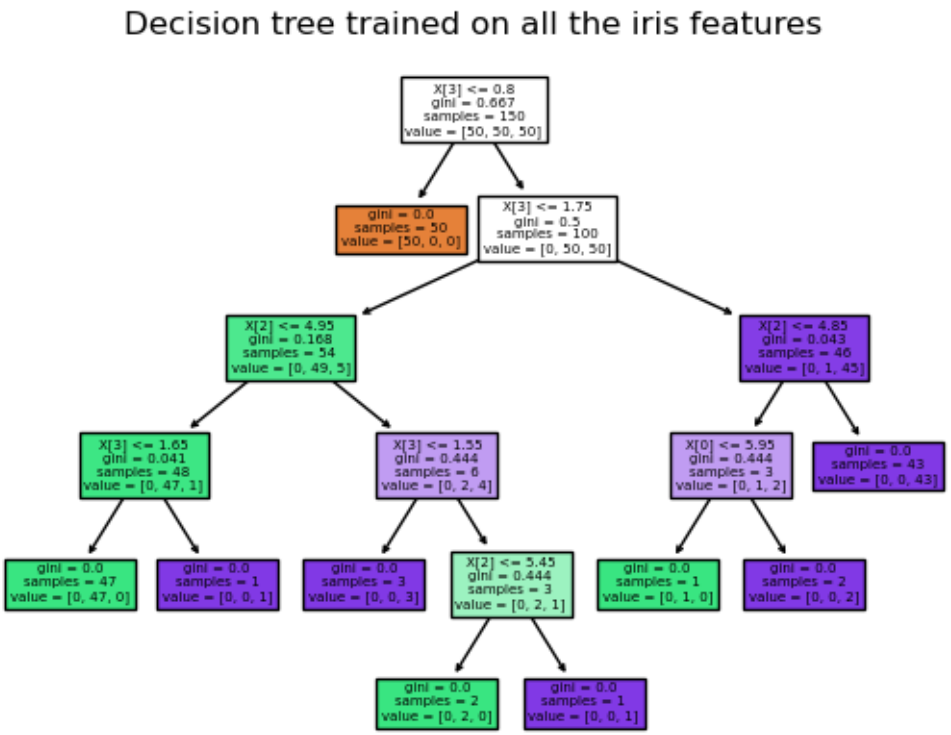
2

- Dùng thuật toán tham lam (Greedy) nên giá trị đúng với data cục bộ nhưng lại dễ sai với unseen data (Generalization), chính vì thế sẽ có phương pháp cắt tỉa (pruning) để tối ưu hóa kết quả

1·2. *Cách sử dụng*

Chúng ta sử dụng hàm DecisionTreeClassifier() trong sklearn để thực hiện bài toán phân loại

```
import numpy as np

from sklearn.datasets import load_iris
from sklearn import tree
# Get dataset from sklearn module
iris = load_iris()
# x is an array of samples, including features, y is a resulted label
# of samples
X, y = iris.data, iris.target
clf = tree.DecisionTreeClassifier()
# Put data to classify the sample
clf = clf.fit(X, y)
# Visualize the decision tree
tree.plot_tree(clf)
```

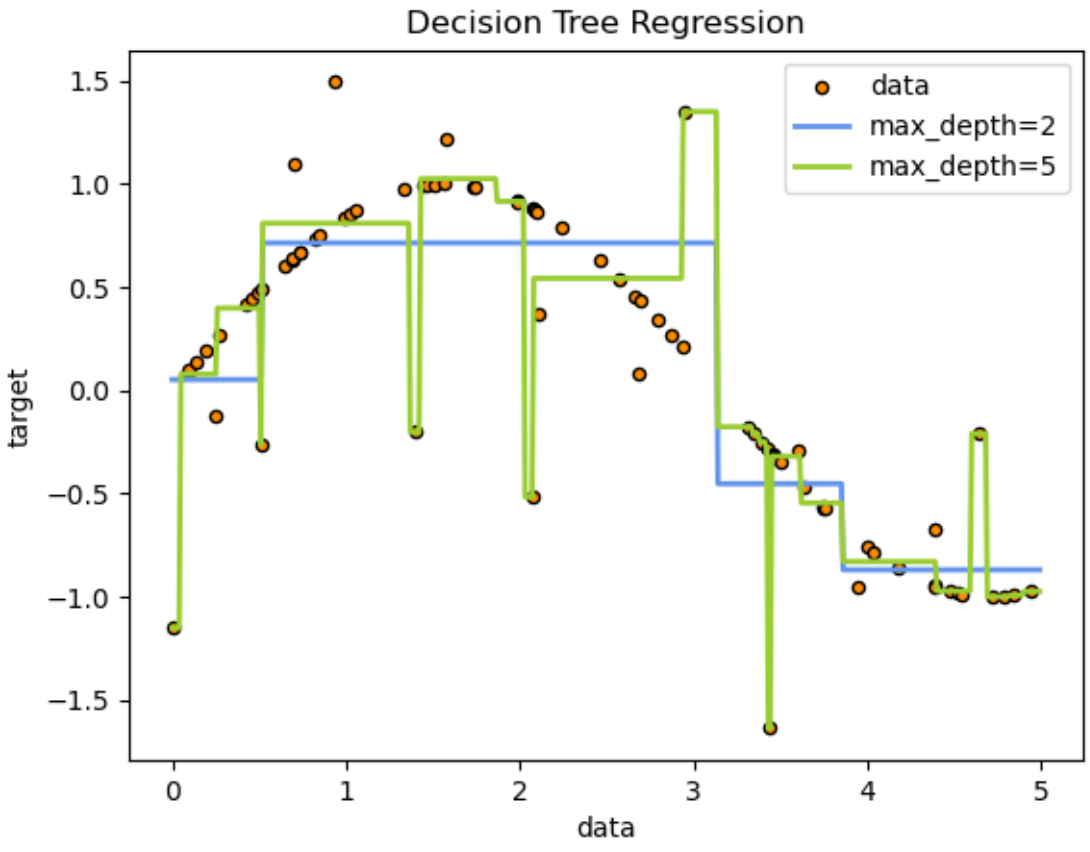


Chúng ta sử dụng hàm DecisionTreeRegressor() trong sklearn để thực hiện bài toán hồi quy

```
# Import the necessary modules and libraries
import numpy as np
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt

# Create a random dataset
rng = np.random.RandomState(1)
X = np.sort(5 * rng.rand(80, 1), axis=0)
y = np.sin(X).ravel()
y[::5] += 3 * (0.5 - rng.rand(16))

# Fit regression model
regr_1 = DecisionTreeRegressor(max_depth=2)
regr_2 = DecisionTreeRegressor(max_depth=5)
regr_1.fit(X, y)
regr_2.fit(X, y)

# Predict
X_test = np.arange(0.0, 5.0, 0.01)[:, np.newaxis]
```

4

```
y_1 = regr_1.predict(X_test)
y_2 = regr_2.predict(X_test)

# Plot the results
plt.figure()
plt.scatter(X, y, s=20, edgecolor="black", c="darkorange", label="data")
plt.plot(X_test, y_1, color="cornflowerblue", label="max_depth=2", linewidth
plt.plot(X_test, y_2, color="yellowgreen", label="max_depth=5", linewidth=2
plt.xlabel("data")
plt.ylabel("target")
plt.title("Decision Tree Regression")
plt.legend()
plt.show()
```



1·3. *Các tham số*

Decision tree hỗ trợ rất nhiều tham số mà bạn có thể truyền vào để tối ưu hóa cây của bạn. Một vài các tham số cần lưu ý như sau:

- **criterion**: Tiêu chuẩn tính, bao gồm: {'gini', 'entropy', 'log_loss'}.Mặc định là 'gini'. Dùng để xác định tiêu chuẩn chia để tạo decision tree.

- **max_depth**: Nhận vào giá trị là một số nguyên dương, mặc định là None. Dùng để giới hạn độ sâu của cây. Nếu max_depth = None cây sẽ mở rộng đến khi tất cả node lá chứa ít mẫu hơn tham số min_samples_split.
- **min_samples_split**: Nhận vào giá trị > 0, mặc định là 2. Là số lượng mẫu tối thiểu để tiếp tục tách một node.
- **min_samples_leaf**: Nhận vào giá trị > 0, mặc định là 1. Là số lượng mẫu tối thiểu cần thiết ở một nút lá. Việc phân chia một điểm ở bất cứ độ sâu nào chỉ được thực hiện khi số lượng mẫu phải lớn hơn min_samples_leaf ở cả nhánh trái và phải
- **max_leaf_nodes**: Nhận vào giá trị là một số nguyên dương, mặc định là None. Trồng cây có số lượng node tối đa theo cách tốt nhất

2. TRÌNH BÀY THUẬT TOÁN

Trước khi trình bày thuật toán ta sẽ đến với hai tiêu chuẩn để chia mục trong Decision Tree đó là Entropy (Shannon entropy) và Gini impurity. Và sau đó là thuật toán ID3. Từ nền tảng toán học đó ta sẽ tiến hành implement Decision Tree

2·1. *Entropy và Information gain*

Entropy là một khái niệm khá thông dụng được dùng trong nhiều lĩnh vực vật lý, toán học v..v.. Trong lĩnh vực xử lý thông tin, nhà toán học Claude Shannon đã đưa ra khái niệm information entropy (Shannon entropy). Shannon entropy thể hiện mức độ hỗn loạn hay độ nhiễu của data. Với một hệ thống với N trạng thái có thể xảy ra thì công thức của Shannon entropy sẽ như ở dưới:

$$S = -\sum_{i=1}^{N} p_i \log_2 p_i$$

Trong đó là xác suất trạng thái thứ xuất hiện ở trong hệ thống. Giá trị entropy càng cao thì độ hỗn loạn của hệ thống càng cao, còn nếu càng thấp thì hệ thống càng có trật tự. Và một hệ thống có trật tự cũng tương đương với việc data được phân nhánh một cách chuẩn chỉ. Vì vậy khi càng có nhiều data, entropy ngày càng có hệ thống hơn và khả năng phân loại ổn định hơn. Do vậy, độ giảm của entropy được gọi là information gain và có công thức như sau:

$$IG(Q) = S_0 - \sum_{i=1}^{q} \frac{N_i}{N} S_i$$

- IG : giá trị information gain
- Q : điều kiện để chia data
- q : số nhóm sau khi chia
- N_i : số dữ liệu trong tập

2·2. *Gini impurity*

Đơn giản hơn so với Entropy và information gain, Gini Impurity là chỉ số thể hiện mức độ phân loại sai khi ta chọn ngẫu nhiên một phần tử từ tập data. Gini Impurity có công thức như sau:

$$G = \sum_{k} p_k * (1 - p_k)^2 = 1 - \sum_{k} (p_k)^2$$

- G: giá trị Gini Impurity
- K: số các lớp có trong tập data

- p_k :xác suất mà một phần tử ngẫu nhiên thuộc lớp

Ta thấy, giá trị Gini Impurity sau khi chia của mỗi nhóm đều nhỏ hơn so với giá trị ban đầu => Sau khi chia nhóm mức độ phân loại sai của hệ thống đã giảm. Độ giảm của Gini Impurity được gọi là Gini Gain và có công thức tính tương tự như information gain, chỉ khác là ta sẽ sử dụng giá trị Gini Impurity thay vì Entropy:

$$GG(Q) = G_0 - \sum_{i=1}^{q} \frac{N_i}{N} G_i$$

2·3. *Thuật toán ID3*

ID3 (Iterative Dichotomiser 3) được phát triển vào năm 1986 bởi Ross Quinlan. Thuật toán tạo ra một cây đa đường, tìm cho mỗi nút (Greedy) phân loại nhiều mục,từ đó phân loại tối đa các đặc trưng của mẫu.Cây luôn đạt kích thước tối đa và sau đó bước cắt tỉa (pruning) được áp dụng để khái quát hóa dữ liệu sẽ được dự đoán.Thuật toán sử dụng đệ quy (recursive), tham lam (greedy) và chia để trị (Divide and conquer). Ở đây chúng ta sử dụng tiêu chuẩn Gini hay Entropy đều không thay đổi kết quả chọn nhiều lắm

Các bước tiến hành thuật toán :

- Thuật toán ID3 bắt đầu với tập hợp ban đầu S là nút gốc (root node)
- Sau đó mỗi lần đệ quy tính ra những thành phần không sử dụng của tập S, tính ra entropy hoặc là Information Gain.
- Sau khi tính được, chọn Entropy có giá trị nhỏ nhất (hoặc Information gain là lớn nhất)
- Từ việc chọn ta chia cây ra các nhánh nhỏ hơn quyết định dựa trên entropy, sau đó ta tiếp tục đệ quy trên các nhánh con để xác định tiếp, xác định dựa trên nhiều thuộc tính (attributes) khác của cây

Chính vì đệ quy hết mức nên đối với tập data test ta luôn đạt được độ chính xác 100% nhưng chỉ là chính xác đối với tập test nhưng rất dễ lệch khi xác định trên tập thật. Để khắc phục điều này ta có thể sử dụng cắt tỉa và điều kiện để tổng quát hóa cây, nhằm tăng độ chính xác với dữ liệu chưa biết

Các điều kiện dừng cây :

- Entropy hoặc Information Gain bằng 0 hoặc nhỏ hơn một giá trị chấp nhận được
- Số lượng node hoặc số lượng nhánh cây đạt tới giới hạn
- Sau khi đạt số lượng node hoặc nhánh cây thì tối ưu lại node của cây để tăng độ chính xác trên tập test

3. KẾT LUẬN

Decision Tree là một phương pháp để chọn dữ liệu đơn giản và dễ hiểu, tuy nhiên để có độ chính xác cao thì cần phải tối ưu và điều chỉnh cho phù hợp.

Tài liệu tham khảo:

- *https://machinelearningcoban.com/2018/01/14/id3/*
- https://en.wikipedia.org/wiki/ID3_algorithm
- *https://www.oreilly.com/library/view/hands-on-machine-learning/9781491962282/*
- https://scikit-learn.org/stable/modules/classes.htmlmodule-sklearn.tree
- *https://scikit-learn.org/stable/modules/tree.html*

Link colab: *https://colab.research.google.com/drive/1rfR7k_xr1hFMTDriWaNFvhotHdTd_pji#scrollToJ̄jTJ5D-QphlW*