CS300

Artificial Intelligent

Lecturer: Dr. Nguyen Ngoc Thao
Msc. Nguyen Hai Dang

Msc. Nguyen Thanh An

Students: Pham Minh Duy - 1859010
Vuong Quang Huy - 1859021

Quach Hoang Minh - 1859033

Tran Ngoc Phuong My - 1859034

I. Table of Contents

II. Mô tả hướng tiếp cận:

1. Vấn đề được đặt ra:

Nhóm sử dụng tập dữ liệu Fashion MNIST để xây dựng một mô hình nhận dạng, trong đó:

- Input: ảnh (28x28) pixels (ảnh grayscale)
- Output: loại trang phục (0-9)

2. Hướng tiếp cận:

Nhóm sử dụng Scikit-Learn, một library hữu dụng cho Machine Learning trong Python.

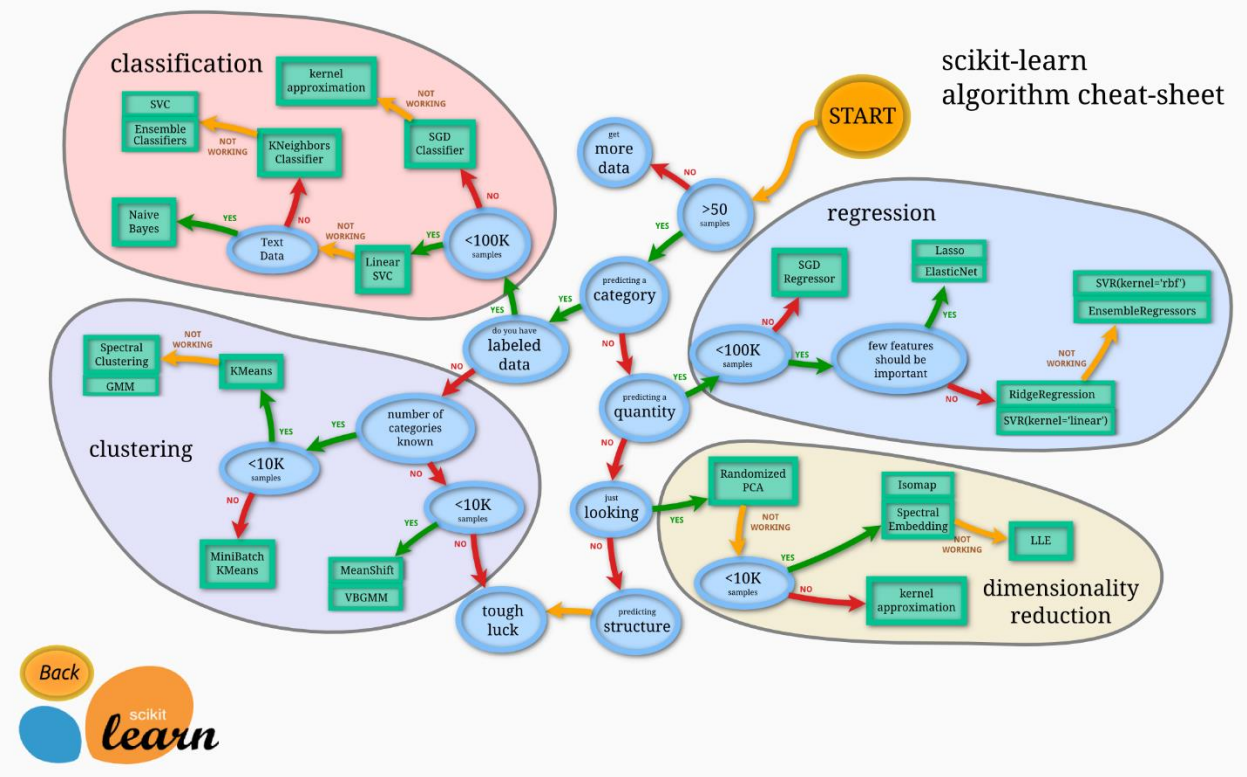


Figure 1https://scikit-learn.org/stable/tutorial/machine_learning_map/index.html

Áp dụng Flowchart trên của Scikit-Learn, tìm kiếm thêm trong tài liệu của Scikit-learn, và đây là vấn đề liên quan đến Classification nên nhóm đã đưa ra quyết định sử dụng hai thuật toán sau :

- Random Forest Classifier

- Stochastic Gradient Descent

III. Cấu trúc mô hình :

1. Random Forests

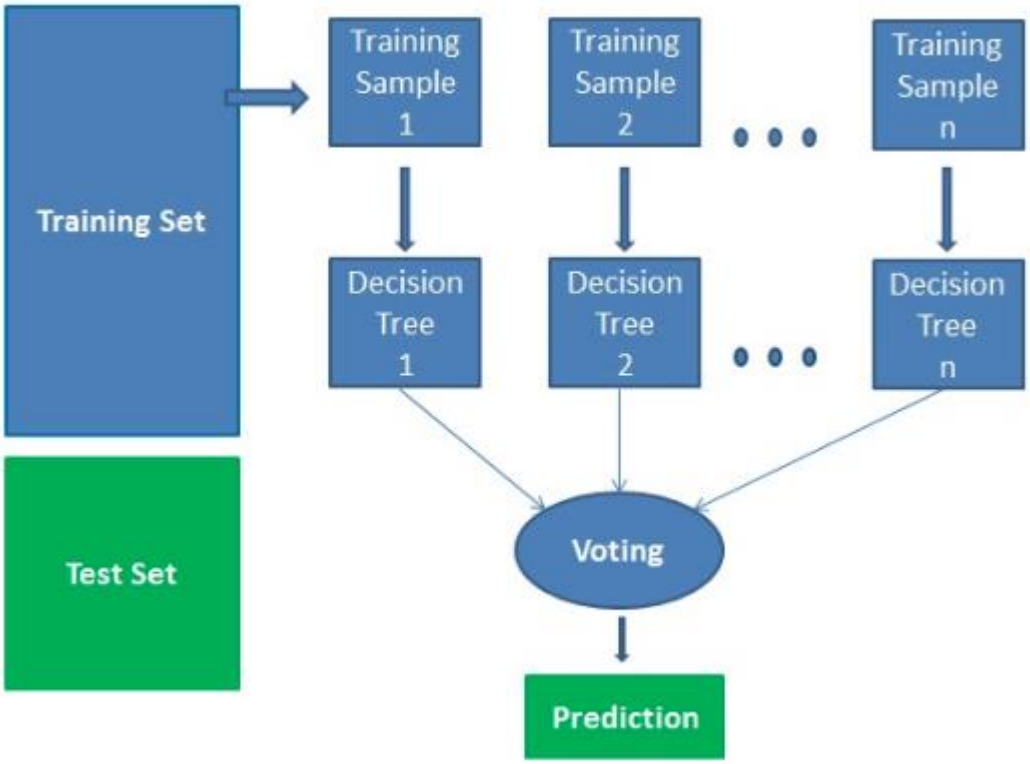
Random Forests là thuật toán học có giám sát (supervised learning). Nó có thể được sử dụng cho cả phân lớp (RandomForestClassifier) và hồi quy (RandomForestRegression).

Random forests tạo ra cây quyết định trên các mẫu dữ liệu được chọn ngẫu nhiên, được dự đoán từ mỗi cây và chọn giải pháp tốt nhất bằng cách bỏ phiếu.

Thuật toán hoạt động theo bốn bước:

1. Chọn các mẫu ngẫu nhiên từ tập dữ liệu đã cho.
2. Thiết lập cây quyết định cho từng mẫu và nhận kết quả dự đoán từ mỗi quyết định cây.
3. Hãy bỏ phiếu cho mỗi kết quả dự đoán.

4. Chọn kết quả được dự đoán nhiều nhất là dự đoán cuối cùng.



Ưu điểm: Random forests được coi là một phương pháp chính xác và mạnh mẽ vì số cây quyết định tham gia vào quá trình này. Nó không bị vấn đề overfitting. Lý do chính là nó mất trung bình của tất cả các dự đoán, trong đó hủy bỏ những thành kiến. Thuật toán có thể được sử dụng trong cả hai vấn đề phân loại và hồi quy.

Nhược điểm: Random forests chậm tạo dự đoán bởi vì nó có nhiều cây quyết định. Bất cứ khi nào nó đưa ra dự đoán, tất cả các cây trong rừng phải đưa ra dự đoán cho cùng một đầu vào cho trước và sau đó thực hiện bỏ phiếu trên đó. Toàn bộ quá trình này tốn thời gian

2. Stochastic Gradient Descent :

Gradient Descent (GD) là 1 trong những thuật toán sử dụng đạo hàm để tìm ra giá trị nhỏ nhất của hàm số. Tuy nhiên vì duyệt toàn bộ phần tử cùng lúc khiến cho GD gặp 1 trở ngại cực lớn đó là bộ nhớ. Stochastic Gradient Descent (SGD) đã được ra đời để giải quyết vấn đề đó.

Nguyên lý hoạt động: Tương tự như GD, SGD cũng được cập nhật lại trọng số dựa sau mỗi lần training data. Tuy nhiên, điểm khác biệt giữa SGD và GD là SGD sẽ chia nhỏ dữ liệu ra làm nhiều phần nhỏ rồi bắt đầu tính toán dựa trên những phần nhỏ đó

Ưu điểm:

- Vì chia nhỏ dữ liệu, SGD có thể tính toán 1 lượng lớn data do các data đã bị chia nhỏ. Do vậy, SGD được ưu tiên rất nhiều đối với lượng data lớn.
- Ngoài ra, SGD cũng giải quyết được 1 vấn đề mà GD gặp phải đó là online learning. Online learning là dữ liệu mới được truyền vào liên tục trong quá trình học. Đối với GD, việc sử dụng toàn bộ dữ liệu khiến cho mỗi lần dữ liệu mới được thêm vào sẽ buộc GD học lại từ đầu. SGD có thể dễ dàng giải quyết vấn đề đó vì có thể tách dữ mới ra thành 1 batch nhỏ riêng và tiến hành học nó sau đó

Nhược điểm:

- Độ chính xác thấp hơn GD do chia nhỏ các phần dữ liệu khiến cho dữ liệu việc tính toán bị thay đổi khá nhiều.
- Tốc độ học (learning rate) là 1 vấn đề tồn tại từ thuật toán GD. Nếu tốc độ học quá thấp sẽ mất rất nhiều thời gian để tìm ra điểm cực tiểu. Tuy nhiên nếu tốc độ học quá cao sẽ khiến cực kỳ khó khăn để tìm ra điểm cực tiểu nếu như không muốn nói là sẽ không thể.

IV. Thuyết giải về lý thuyết :

1. RandomForestClassifier:

Đầu tiên, lấy dữ liệu của MNIST

Sau đó, phân chia thanh các training và test set.

```
from mnist import fashion_mnist
x, y, x_test, y_test = fashion_mnist('FASHION_MNIST')
```

Nhóm xử lý dữ liệu đầu vào, chuyển từng hình ảnh về mảng 1 chiều và chuyển từng pixel 0-255 về 0-1

```
# convert each image to 1 dimensional array
x = x.reshape(len(x),-1)
```

```
x_test = x_test.reshape(len(x_test),-1)

# normalize the data to 0 - 1
x = x.astype(float) / 255.
x_test = x_test.astype(float) / 255.

print(x.shape)
print(y.shape)
print(x_test.shape)
print(y_test.shape)

output :
(60000, 784)
(60000,)
(10000, 784)
(10000,)
```

Sau đó, nhóm sử dụng Scikit-Learn và import RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100)
```

n_estimators là số lượng cây ở trong “’rừng’

Kế đến, nhóm huấn luyện mô hình trên bằng training set (x, y).

Sử dụng hàm fit : xây dựng một rừng của các cây từ tập training (x, y)

```
rf.fit(x, y)
```

Sử dụng cross_val_score() để đánh giá điểm bằng phương pháp cross-validation. Cross validation là một phương pháp thống kê được sử dụng để ước lượng hiệu quả của các mô hình học máy.

cv : Xác định chiến lược phân tách xác thực chéo. 'cv=5’ : tính toán điểm số 5 lần liên tiếp (với các mức phân chia khác nhau mỗi lần)

Sử dụng np.mean để tính trung bình kết quả.

```
import numpy as np
from sklearn.model_selection import cross_val_score

scores = cross_val_score(rf, x, y, cv=5)
np.mean(scores)

Output: 0.8818333333333335
```

Cuối cùng, nhóm tính toán độ chính xác trên test_set (x_test, y_test):

Sử dụng hàm score: hàm trả về độ chính xác trung bình (mean accuracy) trên các nhãn và dữ liệu thử nghiệm được cho.

```
rf.score(x_test, y_test)
output: 0.8776
```

2. Stochastic Gradient Descent:

Tương tự như Random Forest, các bước lấy dữ liệu và tách dữ liệu ra thành 2 cụm: training và testing. Cụm data training 60000 phần tử dùng để training và được test lại bằng cụm testing với 10000 phần tử.

Tiếp theo, ta import Stochastic Gradient Descent và tạo model mới với loss function là “hinge”

```
from sklearn.linear_model import SGDClassifier

sgd_clf = SGDClassifier(loss='hinge')
```

Sau đó, ta sẽ đưa dữ liệu đầu vào và tiến hành training model bằng hàm fit() và:

```
sgd_clf.fit(x, y)
```

Tiếp đến, ta dùng hàm cross_value_score() với cv=5 để ước lượng hiệu quả thuật toán

```
import numpy as np
from sklearn.model_selection import cross_val_score

scores = cross_val_score(sgd_clf, x, y, cv=5)
print("Training Accuracy", np.mean(scores))


output: 0.8386166666666666
```

Cuối cùng, ta tính toán độ chính xác dựa trên tập dữ liệu test

```
predicted = sgd_clf.predict(x_test)

print("Testing Accuracy", accuracy_score(y_test, predicted))

-->Testing Accuracy 0.8294
```

V. Kết luận và đánh giá :

So với SGD, RF chậm hơn do phải tính toán nhiều tree và dự đoán dựa trên nó. Tuy nhiên, dự đoán của RF sẽ nhanh hơn và đưa ra giải pháp nhanh và chính xác hơn so với SGD nếu sử dụng trên dữ liệu nhỏ. Còn đối với dữ liệu lớn, việc chia data thành nhiều nhóm nhỏ khiến cho việc xử lý của SGD nhanh hơn RF 1 khoảng tương đối

VI. Tham khảo :

https://www.codingame.com/playgrounds/37409/handwritten-digit-recognition-using-scikit-learn

https://viblo.asia/p/phan-lop-bang-random-forests-trong-python-djeZ1D2QKWz

https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.htmls

https://scikit-learn.org/stable/modules/cross_validation.html

https://www.dathoangblog.com/2018/07/gradient-descent.html

https://viblo.asia/p/optimizer-hieu-sau-ve-cac-thuat-toan-toi-uu-gdsgdadam-Qbq5QQ9E5D8

THE END