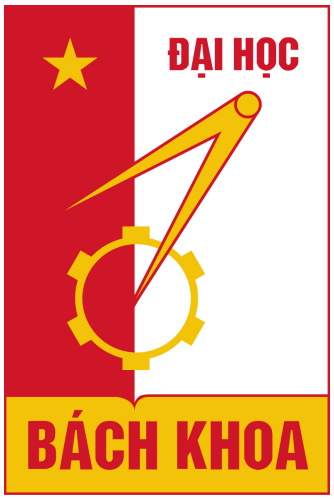
ĐẠI HỌC BÁCH KHOA HÀ NỘI

o0o



BÁO CÁO LAB AITHINGS

CHỦ ĐỀ: MACHINE LEARNING BASIC

Người thực hiện: Vũ Văn Huy - 20216931

Lớp: Hệ thống thông tin quản lý - K66

Hà Nội, 10/2023.

Mục lục

Mở đầu

Xin chào tất cả mọi người,

Mình xin gửi tới mọi người lời chào trân trọng và xin phép tự giới thiệu mình là một thành viên mới của lab. Đây là bài báo cáo đầu tiên của tôi và đồng thời là sự khởi đầu cho hành trình học hỏi và cống hiến trong lĩnh vực trí tuệ nhân tạo (AI).

Sự đam mê của mình dành cho lĩnh vực này đã thúc đẩy mình tham gia vào lab, và bài báo cáo này là cơ hội tuyệt vời để chia sẻ với mọi người những kiến thức và tìm hiểu của tôi về 6 vấn đề quan trọng trong AI.

Bài báo cáo này gồm 6 phần chính, mỗi phần sẽ trình bày một vấn đề cụ thể. Qua việc tìm hiểu về cac thuật toán, tool, K-Nearest Neighbors (KNN), Decision Trees, Git Flow, Adaptive Boosting (AdaBoost), Support Vector Machines (SVM), và Random Forests. Từ đó mỗi phần sẽ cung cấp một cái nhìn tổng quan về vấn đề đó và mô tả cách chúng hoạt động cùng với các ứng dụng thực tế.

Rất mong muốn nhận được sự góp ý và đóng góp từ mọi người trong các buổi tới. Cùng nhau chia sẻ kiến thức và kinh nghiệm để cùng nhau phát triển và nắm bắt cơ hội trong lĩnh vực AI đầy tiềm năng.

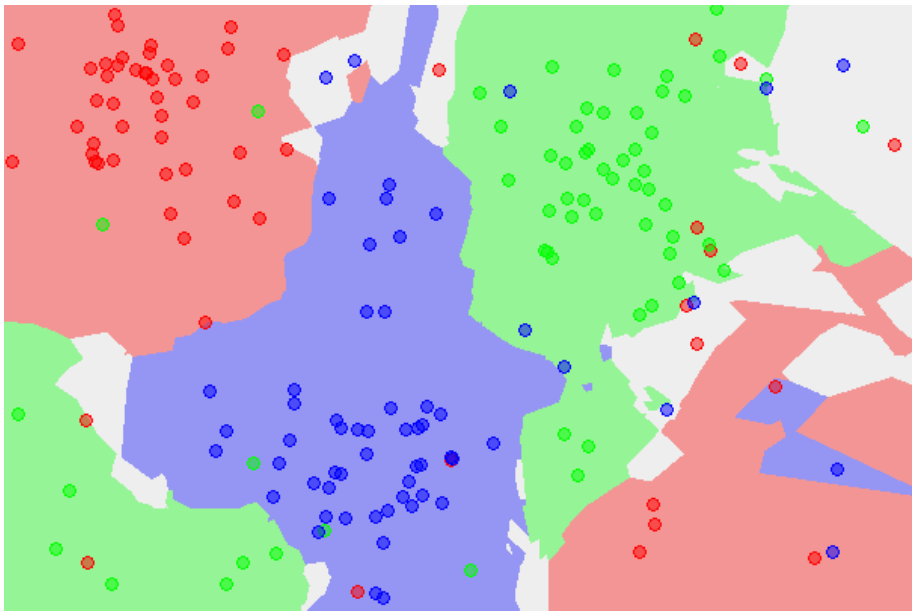
Trân trọng cảm ơn

Người báo cáo

Vũ Văn Huy - 20216931

1 Tìm hiểu KNN

1.1 KNN là gì?



Hình 1: **Minh họa KNN**

KNN(K - Nearest Neighbors) là một thuật toán học có giám sát, đơn giản và dễ triển khai. Thường dùng cho các bài toán liên quan đến phân loại và dự đoán, hồi quy.(xếp loại lazy machine do không học một điều gì từ dữ liệu training).

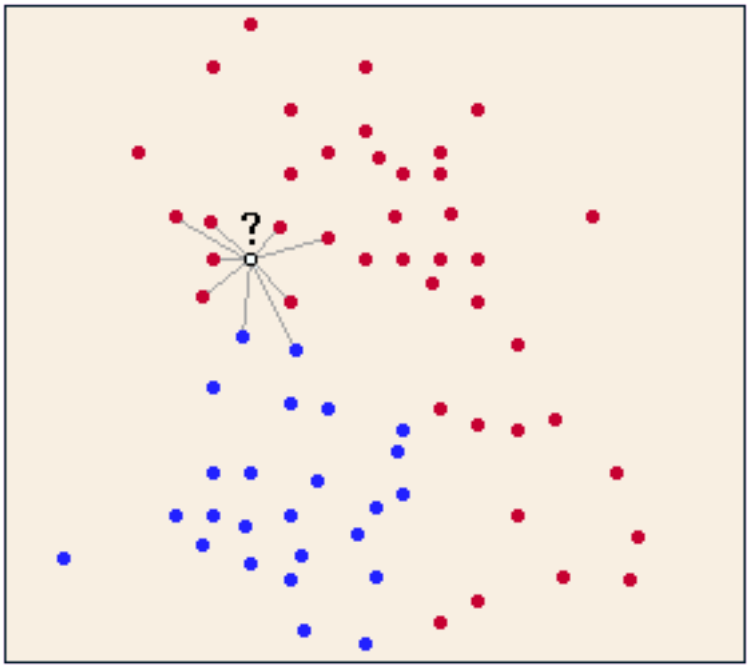
Ý tưởng: KNN cho rằng những điểm tương tự nhau sẽ tồn tại **gần nhau** trong 1 không gian, từ đó ta chỉ cần kiểm tra k điểm gần với dữ liệu cần kiểm tra nhất

Input: Dataset, Số K(số neighbors), điểm dữ liệu mới.

Output: là lớp của bản ghi cần phân loại(bài toán phân loại). Hoặc giá trị của hàm mất mát tại bản ghi được xem xét(bài toán hồi quy). Giá trị dự đoán này là trung bình nhãn của k mẫu gần nhất.

1.2 Các bước của thuật toán

1. Tiền xử lý dữ liệu và chuẩn bị dữ liệu
2. Xác định giá trị K(không phải là giá trị tùy ý như KMeans), có thể cho k chạy từ 1 - n xem cái nào tốt nhất.
3. Tính khoảng cách giữa các điểm dữ liệu.



Hình 2: **Minh họa đối với 1 point mới được thêm và cần dự đoán**

4. Lựa chọn K neighbors gần nhất(thường chọn k là $\sqrt{n}$).
5. Đưa ra dự đoán về nhãn của 1 điểm dữ liệu

1.3 Các hàm tính khoảng cách áp dụng

- Khoảng cách Manhattan:

$$\boxed{d = \sum_{i=1}^{n} ||x_i - y_i||}$$

- Khoảng cách Euclid:

$$\boxed{d = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}}$$

- Khoảng cách Cosine:

$$\boxed{\cos\theta = \frac{\vec{a}.\vec{b}}{||\vec{a}||\vec{b}||}}$$

- Khoảng cách minkowski:

$$\boxed{d = (\sum_{i=1}^{k} (|x_i - y_i|)^q)^{\frac{1}{q}}}$$

1.4 Bài toán điển hình

Xem tại bản note tay.

1. Bài toán phân loại các loài hoa với bộ dữ liệu nổi tiếng **Iris,**

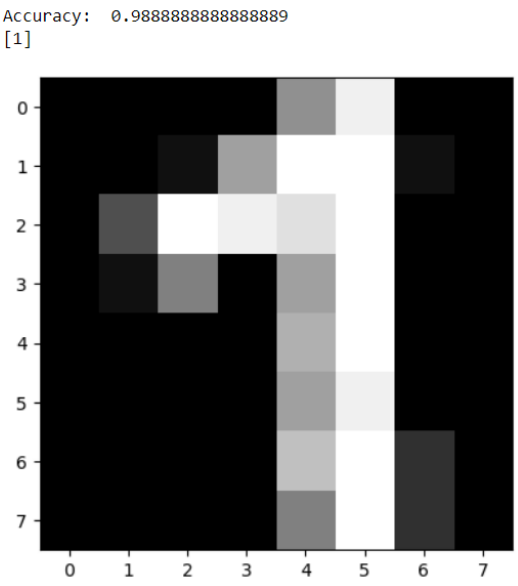
```
from sklearn import datasets, neighbors
import numpy as np
# split data
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

iris = datasets.load_iris()
iris_X = iris.data # data
iris_y = iris.target # label

randIndex = np.arange(iris_X.shape[0])
np.random.shuffle(randIndex)

iris_X = iris_X[randIndex]
iris_y = iris_y[randIndex]

X_train, X_test, y_train, y_test = train_test_split(iris_X, iris_y,
    test_size=50)
# get data point format np array numpy array
print(len(y_test))
'''
input trainset, test set, train label, test label.
- Test_size = 50 which mean it dived 50 to train and 100 to test
'''

# module algo
knn = neighbors.KNeighborsClassifier(n_neighbors = 5)

# train model
knn.fit(X_train, y_train) # fit is train it, input is datapoints(100),
    y_train is lable
y_predict = knn.predict(X_test)
print(y_predict)
print(y_test)

accuracy = accuracy_score(y_predict, y_test)
print(accuracy)
```

```
50
[1 2 1 0 1 0 2 1 1 2 0 2 1 1 1 0 1 2 2 2 0 1 1 2 1 0 2 1 2 0 1 0 0 2 2 2 1
 2 2 1 1 0 2 2 0 2 2 1 1 0]
[1 2 1 0 1 0 2 1 1 1 0 2 1 1 1 0 1 2 1 2 0 1 1 2 1 0 2 1 2 0 1 0 0 2 2 2 1
 2 2 1 1 0 2 2 0 2 2 1 1 0]
0.96
```

Hình 3: Kết quả Iris

2. Dự đoán chữ với bộ dữ liệu **chữ viết tay**.

```
from sklearn import datasets, neighbors
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

digit = datasets.load_digits()
digit_X = digit.data # data
digit_y = digit.target # label

randIndex = np.arange(digit_X.shape[0])
np.random.shuffle(randIndex)

digit_X = digit_X[randIndex]
digit_y = digit_y[randIndex]

X_train, X_test, y_train, y_test = train_test_split(digit_X, digit_y,
    test_size=360)

knn = neighbors.KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train, y_train)
y_predict = knn.predict(X_test)

accuracy = accuracy_score(y_predict, y_test)
print(accuracy)

# Test 1 image
plt.gray()
plt.imshow(X_test[0].reshape(8,8))
print(knn.predict(X_test[0].reshape(1, -1)))
plt.show()
```

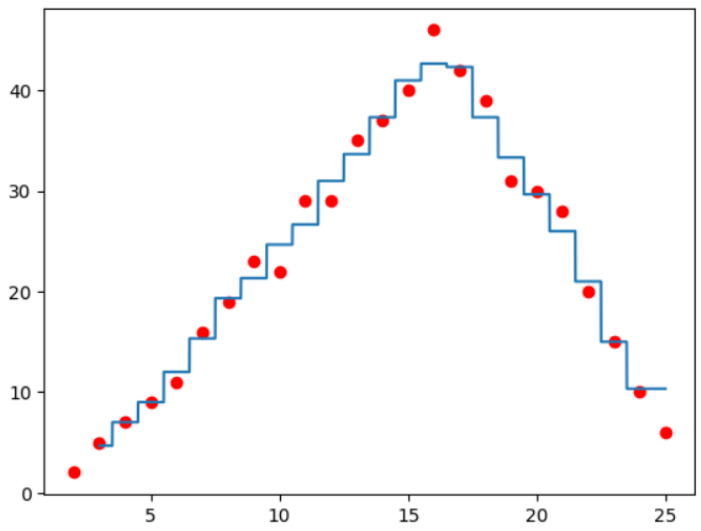


Hình 4: Kết quả digit

3. Dự đoán **điểm trung bình** của 1 học sinh khi biết thời gian học.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import neighbors

X = [2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25]
y = [2,5,7,9,11,16,19,23,22,29,29,35,37,40,46,42,39,31,30,28,20,15,10,6]
plt.plot(X, y, 'ro')

X = np.array(X).reshape(-1, 1)
y = np.array(y)

x0 = np.linspace(3,25,10000).reshape(-1,1) # get in 3 number create a np
    array with insight is 10000 number from 3-25
y0 = []

test = [[15.6]]
predict_test = knn.predict(test)
print(predict_test)

knn = neighbors.KNeighborsRegressor(n_neighbors = 3)  # n = 3 find 3
   point arround it and nearest
knn = knn.fit(X,y)
y0 = knn.predict(x0)

plt.plot(x0, y0)

plt.show()
```

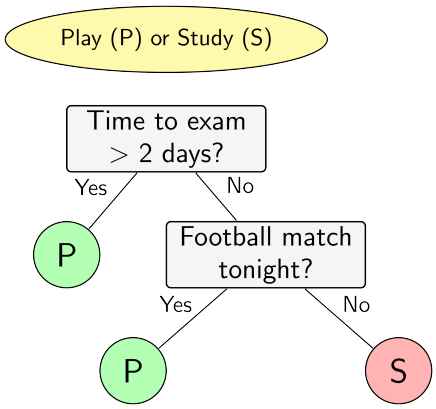
```
[42.66666667]
```



Hình 5: Kết quả predict_point_student

2 Decision tree

2.1 Decision tree



Cây quyết định là một thuật toán mô phỏng con người để đưa ra quyết định. Thực chất giống như một chuỗi if else.

Mô hình được sử dụng khá phổ biến và hiệu quả trong cả hai lớp bài toán phân loại và dự báo của học có giám sát.

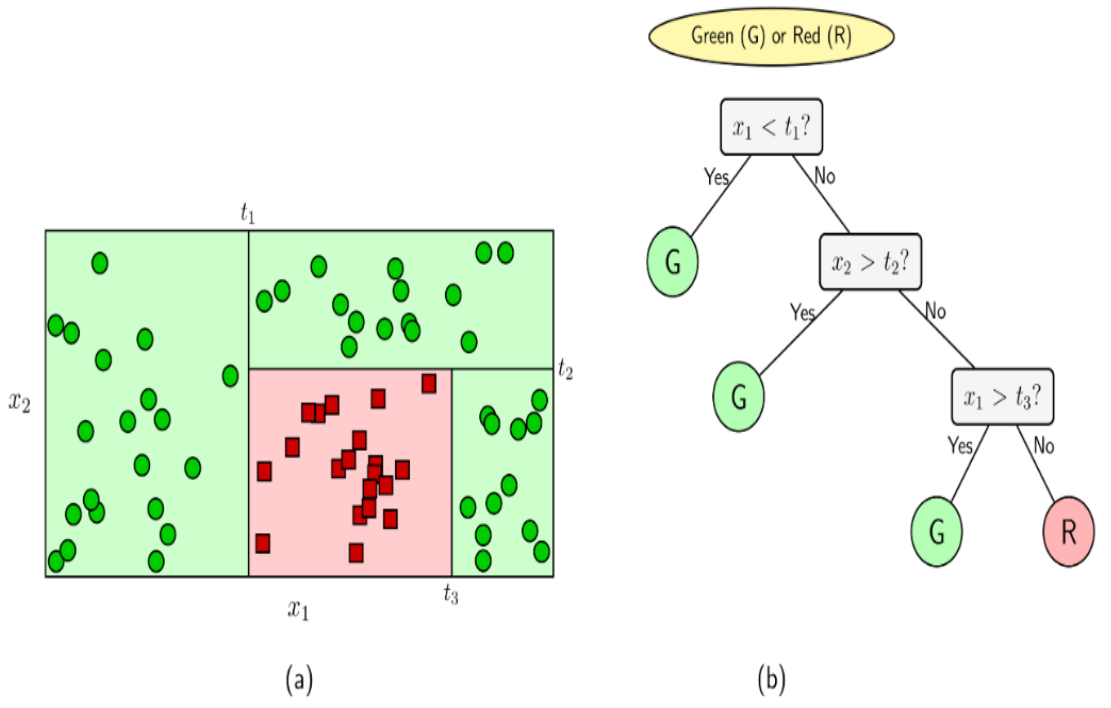
Cây quyết định có tiềm năng lớn, hiệu suất ổn định khi thực hiện học trên các tập dữ liệu phức tạp.

Cây quyết định là một trong các mô hình học máy nổi tiếng do tính đơn giản, dễ hiểu.

Gặp bài toán gì trước hết cứ phải nghĩ đến sử dụng cây

→ Kinh nghiệm của các ông anh khóa trước đã gia nhập thị trường lao động. ☺

Ví dụ:



Hình 2: Ví dụ về bài toán phân lớp sử dụng decision tree.

Chú thích: Trích lý thuyết đồ thị bài toán cây từ học phần Toán Rời Rạc.

- Các hoa văn màu đỏ, xám và xanh lục trên cây được gọi là **node**
- Node màu đỏ và màu xanh lá cây được gọi là **node lá hoặc node cuối.**
- Node màu xám chứa câu hỏi và được gọi là **node không có lá**.
- Cái màu vàng được gọi là **gốc.**
- Node không phải lá có thể chứa **một hoặc nhiều node con.**
- Node con có thể là node không phải **lá hoặc node lá.**
- Tất cả các node con từ một node không phải là lá được gọi là **node anh em.**
- Cây có tất cả các node không phải lá chỉ có hai node con được gọi là **cây quyết định nhị phân.**
- Các câu hỏi trong **các node** không phải lá của cây quyết định nhị phân có thể được chuyển đổi thành **câu hỏi có/không**.
- Cây có các **node không phải lá** có nhiều hơn hai node con có thể được chuyển đổi thành **cây quyết định nhị phân** vì hầu hết tất cả các câu hỏi đều có thể được chuyển đổi thành **câu hỏi có/không.**

Dữ liệu thường được tổ chức dưới dạng:

$$\boxed{(x, Y) = (x_1, x_2, x_3, ..., x_k, Y)}$$

Trong đó:

- Y là nhãn, đầu ra mong muốn phải học
- x tạo bởi các đặc trưng $x_1, x_2, x_3, ...$

Các thuật toán nổi tiếng liên quan đến cây quyết định:

- ID3
- C4.5
- CART
- Chi-square automatic iteraction detection
- MARS
- Cây quy nạp điều kiện

Thứ tự lựa chọn câu hỏi:

Đối với những bộ dữ liệu có số lượng biến đầu vào d lớn, xác suất đúng là $\frac{1}{d}$ và khả năng chọn sai là rất cao.

→ Cần phải có một tiêu chí nào đó để lựa chọn biến phù hợp

→ Hình thành nên các độ đo như entropy, Gini đo lường mức độ tinh khiến (purity) và vẩn đục (impurity) của một biến

2.2 Tinh khiết, vẩn đục

Các phép đo để đánh giá cây quyết định:

Ước lượng tham số kết quả dương chính xác (Estimate of Positve Correctness):

$$\boxed{E_p = TP - FP}$$

Độ thuần khiết Gini (Gini impurity , khác với hệ số Gini - Gini index)

- Khái niệm: Là độ đo tần suất một phần tử được chọn ngẫu nhiên từ tập hợp bị dãn nhãn sai nếu nó được dán ngẫu nhiên thoe phân phối nhãn trong tập dữ liệu con.Tính toán xem cây nên rẽ theo hướng nào.

Tinh khiết (purity). Trái ngược lại với nó là vẩn đục (impurity), tức phân phối của các nhãn tại node lá còn khá mập mờ, không có xu hướng thiên về một nhãn nào cụ thể.

Công thức:

$$\boxed{\sum_{k\#i} p_k = 1 - p_i}$$

2.3 Entropy function

Một vấn đề của mô hình cây quyết định là làm thế nào để chọn thuộc tính để phân chia mẫu dữ liệu của mỗi node?

Một cách đơn giản là chọn thuộc tính tốt nhất dựa trên một tiêu chí cụ thể cho từng node.

☆ Tuy nhiên, tiêu chí là gì?

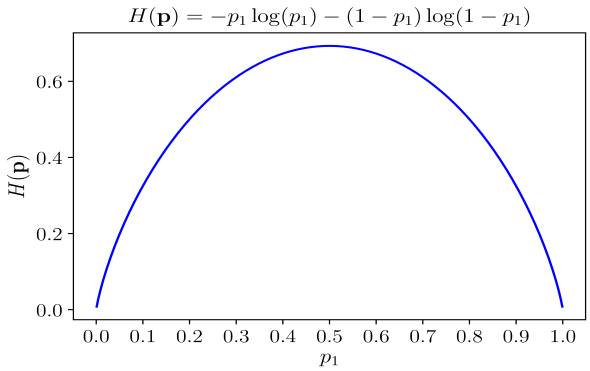
Thuộc tính tốt để phân tách mẫu dữ liệu của chúng ta là thuộc tính tạo node lá (hoặc gần như node lá).

Thuộc tính xấu để phân chia mẫu dữ liệu của chúng tôi là thuộc tính không thể phân chia dữ liệu của chúng tôi một cách rõ ràng.

Vậy nên sử dụng hàm entropy để đánh giá độ tinh khiết của phương pháp tách.

- Giá trị của hàm entropy càng cao thì phương pháp phân tách càng có nhiều tạp chất.
- Giá trị của hàm entropy càng thấp thì phương pháp phân tách càng tinh khiết.

If n = 2 (means x can be one of 2 values):



Generally, if n > 2, $H(p)$ sẽ nhỏ nhất nếu $p_i = 1$ và $H(p)$ sẽ lớn nhất nếu mọi p_i tương tự($p_0 = p_1 = 0.5$).

2.4 Thuật toán ID3 (Iterative Dichtomiser 3)

ID3 là thuật toán kinh điển , đầu tiên để giúp ta học cây. ID3 algorithm cũng gọi entropy bởi vì nó sử dụng hàm emtropy làm hàm mất mát.

ID3 model cố gắng chọn thuộc tính cho mỗi node đến tổng nhỏ nhất của tất cả mỗi giá trị emtropy

Sử dụng lại ví dụ ban đầu:

Chúng ta gặp vấn đề về phân loại với C các lớp học. Đối với một nút không có lá, chúng ta có S là một tập hợp các điểm dữ liệu và S chứa N điểm dữ liệu.

Đối với mỗi lớp c trong C, chúng ta có N_c mẫu dữ liệu thuộc lớp c và chúng ta có xác suất của một mẫu ngẫu nhiên thuộc về lớp c là $\frac{N_C}{n}$.

Chúng ta có entropy của node này,

$$\boxed{H(s) = -\sum_{c=1}^{C} \frac{N_c}{N} \log \frac{N_c}{N}}$$

Giả sử chọn thuộc tính x cho node này và x có thể tạo K Node con $S_1, S_2, ..., S_K$ với số lượng mỗi node con là $i_1, i_2, ..., i_k$

$$\boxed{H(x, S) = \sum_{k=1}^{K} \frac{i_k}{N} H(S_k)}$$

Thông tin thu được trên x tính bằng:

$$\boxed{G(x, S) = H(S) - H(x, S)}$$

Sau mỗi thuộc tính của node, chúng ta muốn **thu được nhiều thông tin** hoặc **giảm entropy của nút gốc** .

$$\boxed{x^* = argmax G(x, S) = argmin H(x, S)}$$

2.5 Ví dụ sử dụng ID3

Dataset:

Sử dụng thuật ID3 để giải quyết:

Bước 1 : Tính entropy của node gốc.

$$H(S) = -\frac{5}{14} \log\left(\frac{5}{14}\right) - \frac{9}{14} \log\left(\frac{9}{14}\right) \approx 0.65$$

Bước 2 : Tính toán entropy của từng thuộc tính để chọn thuộc tính đầu tiên trong cây.

Cho thuộc tính outlook:

$$H(S_s) = -\frac{2}{5} \log\left(\frac{2}{5}\right) - \frac{3}{5} \log\left(\frac{3}{5}\right) \approx 0.673$$
$$H(S_o) = -\frac{0}{4} \log\left(\frac{0}{4}\right) - \frac{4}{4} \log\left(\frac{4}{4}\right) = 0$$
$$H(S_r) = -\frac{3}{5} \log\left(\frac{2}{5}\right) - \frac{3}{5} \log\left(\frac{3}{5}\right) \approx 0.673$$
$$H(outlook, S) = \frac{5}{14} H(S_s) + \frac{4}{14} H(S_o) + \frac{5}{14} H(S_r) \approx 0.48$$

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rainy	mild	high	strong	no

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
11	sunny	mild	normal	strong	yes

id	outlook	temperature	humidity	wind	play
3	overcast	hot	high	weak	yes
7	overcast	cool	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes

id	outlook	temperature	humidity	wind	play
4	rainy	mild	high	weak	yes
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
10	rainy	mild	normal	weak	yes
14	rainy	mild	high	strong	no

Cho thuộc tính temperature:

$$H(S_h) = -\frac{2}{4}\log\left(\frac{2}{4}\right) - \frac{2}{4}\log\left(\frac{2}{4}\right) \approx 0.693$$
$$H(S_m) = -\frac{4}{6}\log\left(\frac{4}{6}\right) - \frac{2}{6}\log\left(\frac{2}{6}\right) \approx 0.637$$
$$H(S_c) = -\frac{3}{4}\log\left(\frac{3}{4}\right) - \frac{1}{4}\log\left(\frac{1}{4}\right) \approx 0.562$$
$$H(temperature, S) = \frac{4}{14}H(S_h) + \frac{6}{14}H(S_m) + \frac{4}{14}H(S_c) \approx 0.631$$

Tương tự, đối với thuộc tính **humidity and wind**, ta có:

$$H(outlook, S) \approx 0.48$$
$$H(temperature, S) \approx 0.631$$
$$H(humidity, S) \approx 0.547$$
$$H(wind, S) \approx 0.618$$

Do $H(outlook, S)$ nhỏ nhất, nên chọn **outlook** là thuộc tính đầu trong ID3.

Bước 3 : Tiếp tục tính entropy của từng thuộc tính để chọn thuộc tính cho từng nút con của outlook.

For **outlook = sunny**, we use **humidity** to have the zero entropy (humidity = high ⇒ play = no and humidity = normal => **play = yes**).

For outlook = rainy, we use **wind** to have the zero entropy (wind = weak play = yes and wind = strong ⇒ **play = no**).

id	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
13	overcast	hot	normal	weak	yes

id	outlook	temperature	humidity	wind	play
4	rainy	mild	high	weak	yes
8	sunny	mild	high	weak	no
10	rainy	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
14	rainy	mild	high	strong	no

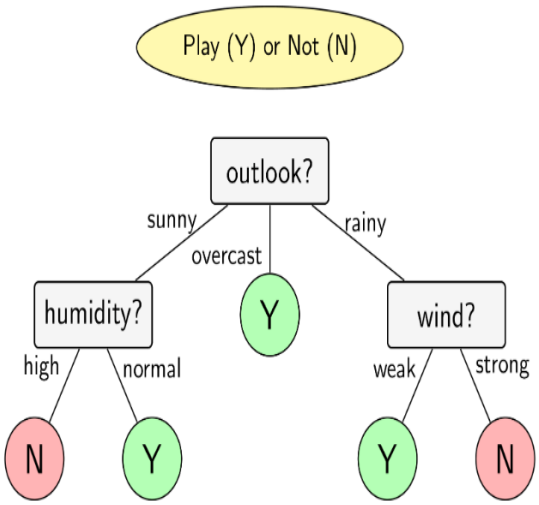
id	outlook	temperature	humidity	wind	play
5	rainy	cool	normal	weak	yes
6	rainy	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
9	sunny	cool	normal	weak	yes

2.6 Điểm yếu, các vấn đề của ID3

Vấn đề lớn nhất của ID3 hay Decision tree là **overfiting**

Một số điều kiện để ngừng tạo node mới hoặc ngừng chia tách các điểm dữ liệu:

- Entropy = 0 có nghĩa là tất cả các điểm dữ liệu của nút này thuộc về một lớp ⇒ Stop.
- Số lượng điểm dữ liệu < ngưỡng khiến một số điểm dữ liệu trong nút này sẽ bị phân loại sai.



⇒ Stop .

- Khoảng cách giữa nút và gốc > một ngưỡng làm cho cây trở nên phức tạp hơn⇒ Stop.
- Số nút lá > một ngưỡng làm cho cây trở nên phức tạp hơn ⇒ Stop.
- Mức tăng thông tin < ngưỡng không làm giảm entropy nhiều ⇒ Stop.

Giải pháp:

- Đầu tiên, cần huấn luyện ID3 cho đến khi tất cả các điểm dữ liệu trong tập huấn luyện được phân loại chính xác.
- Thứ hai, tỉa một số nút lá và biến các nút không phải lá của chúng trở thành nút lá.
- Thứ ba, có thể căn cứ một số tiêu chí để đánh giá quá trình cắt tỉa:
  + Sử dụng một bộ xác nhận để đánh giá
  + Thêm một thuật ngữ chính quy vào hàm mất
  + Tất cả các điểm dữ liệu trong tập huấn luyện được phân loại chính xác nên giá trị loss = 0.
  + Thêm một thuật ngữ λK đến hàm mất mát và bắt đầu quá trình cắt tỉa.
  + Cân bằng số hạng entropy và λK tối ưu hóa mô hình.

3 Github flow

3.1 Git là gì

Hình 6: Git và GitHub (https://git-scm.com/)

- Git là 1 công cụ quản lý mã nguồn hiệu quả, cho phép việc dễ dàng quản lý sự thay đổi của dự án cá nhân. Đồng thời kết hợp với remote là Github để hỗ trợ việc lưu trữ tài source code.

- Github là một mạng xã hội cho lập trình viên đẩy các dự án các nhân lên cho mọi người cùng xem(open source). Dễ dàng theo dõi code, làm việc nhóm, theo dõi tiến độ, phân nhanh code, hosting website tĩnh,... Kết hợp với git là 1 công cụ hoàn hảo để quản lý source code.

3.2 Các trạng thái của git

Để hiểu được cách thức hoạt động của git thì phải nắm chắc được các trạng thái tại một thời điểm bất kỳ. Tất cả những lỗi sinh ra đều có thể sửa và khắc phục khi repo đang ở trạng thái nào.

1. Staging: các file chưa sẵn sàng

2. Commit: các file đã được xác minh và tạo log

3. Pushed: file đã được đẩy lên và lưu tại remote

3.3 Các lệnh git cơ bản

Thông thường trong dự án chỉ cần sử dụng một số lệnh cơ bản của git. Người mới học hoặc không có hiểu biết về git thì chỉ biết thực hiện các câu lệnh theo 1 trình tự nhất định(Đi làm người ta hay bảo thế 😂).

- Tải repo của người khác qua link

```
$ git clone <link>
```

- Khởi tạo 1 repository:

```
$ git init
```

- Thêm file vào trạng thái sẵn sàng

```
$ git add <file_name>
```

- Kiểm tra trạng thái: hiển thị các file đã sẵn sàng và chưa.

```
$ git status
```

- Chuyển code sang trạng thái commit

```
$ git commit -m "content"
```

  Mở đoạn văn bản mô tả sửa nó, đoạn này sẽ đc ghi đè lên commit trước đó

```
$ git commit --amend
```

  có thể alias các lệnh git cho ngắn lại bằng lệnh config.

- Xóa file

```
$ git rm file_name
```

- Xem các log(và các nhật ký tạo ra mỗi khi commit)

```
$ git log
```

- liên kết thư mục hiện tại với remote(chú ý phải đổi tên nhánh thành main vì nhánh mặc định master là nhánh yếu).

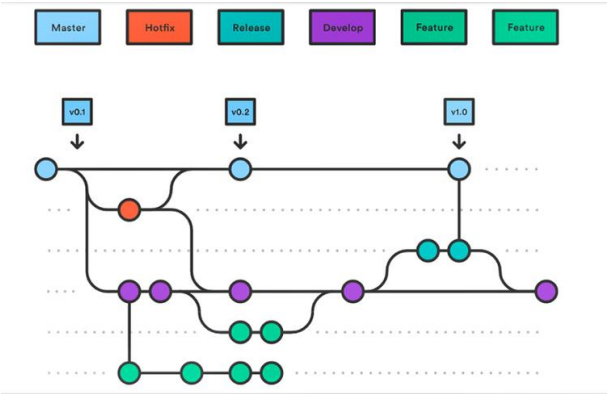
```
$ git remote add origin
```

- Đẩy code lên remote(nếu đã push 1 lần rồi).

```
$ git push
```

- Bỏ qua các file với đuối cố định bằng cách tạo file .ignore. Ví dụ: *.log: bỏ qua các file.log
- Ngoài ra các lệnh khác trong git đều tương tự Comand Line

3.4 Làm việc với nhánh



Hình 7: Các nhánh trong git

Làm việc với nhánh là việc thường xuyên đối với người dùng git.

- Kiểm tra nhánh hiện tại

```
$ git branch
```

- Đổi tên nhánh hiện tại(thường đổi là main cho giống với nhánh trên github

```
$ git branch -M main
```

- Tạo 1 nhánh mới

```
$ git checkout -b ten_nhanh_moi
```

- Chuyển đổi giữa các nhánh:

```
$ git checkout ten_nhanh
```

- Xem danh sách nhánh

```
$ git branch
```

- Xóa nhánh:

```
$ git branch -d ten_nhanh
```

- Hợp một nhánh vào 1 nhánh khác. Chú ý khi merge sẽ xảy ra xung đột, để merge thành công cần phải sử lý xung đột bằng cách vào nhánh hiện tại. Xóa thủ công bằng tay những đoạn không cần thiết. VsCode hỗ trợ rất nhiều extension xử lý điều này.

```
$ git merge ten_nhanh_khac
```

- Kéo thông tin từ nhánh khác về

```
$ git pull
$ git fecth
```

Ngoài ra git còn vô vàn câu lệnh hữu ích. Hầu như nó có thể đáp ứng tất cả nhu cầu của người dùng. Khi code bất kỳ dự án nào nên sử dụng git nếu không dự án có thể **bay màu** bất cứ lúc nào ☺

3.5 Thao tác bổ sung

- Quay lại commit trước
- Đặt tên cho nhánh cho phù hợp
- Git log –graph –oneline: để xem các thay đổi dưới dạng đồ thị giúp hiểu rõ hơn các trường hợp commit
- Git remote -v : kiểm tra repository hiện tại đang làm việc

- Git rebase -i: thì nó hiện ra cái bảng tất cả commit trước đó và mình có thể xem 1 cách chi tiết nó.
- Alias lệnh git khi phải commit quá nhiều.
- Có các tool, nền tảng nào có công dụng tương tự như github: bitbuckit, gitlab. Khi nào nên dùng github. Sử dụng tool nào thì rẻ nhất nếu project để private và muốn share. **Bài toán tối ưu chi phí.**

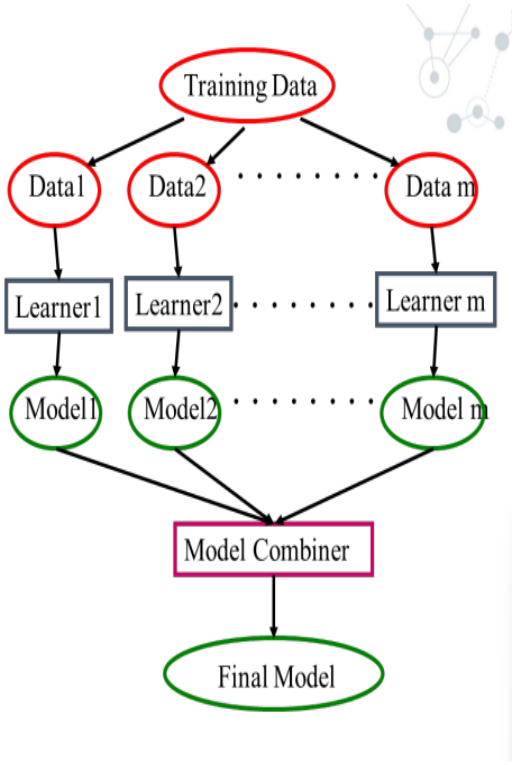
  ⟶ 8 Tip hữu dụng với github

4 Adaboost

4.1 3 hiện tượng thường gặp khi xây dựng 1 mô hình học máy

- **Underfiting**: học kém, tệ trên tập train và tập test. Là do mô hình đang đơn giản quá hoặc do perform quá kém (Trên train và test/value).
- **Overfiting:** Mô hình học quá tốt vào tập huấn luyện nhưng tệ trên tập test. Gặp phải tình trạng này là do huấn luyện quá nhiều, dataset chưa tốt, mô hình quá phức tạp.
- **Goodfit:** mô hình chạy tốt ổn định trên các tập.

4.2 Ensemble learning



Hình 8: Thay vì học một chế độ, học nhiều và kết hợp chúng

Đơn giản nó như team work, 1 mình thì chả làm được cái gì còn nhiều ông mà lại có team lead giỏi thì sẽ làm được việc

Ý tưởng: thay vì huấn luyện 1 mô hình 1 dataset thì ensemble nó bằng nhiều mô hình. Mỗi mô hình ra 1 kết quả và combine lại bằng quá trình voting. Đánh trọng số theo model tốt nhất.

Ví dụ phân biệt nhãn 1, 0. Đến cuối thì theo

Tại sao nói là Emsemble learning làm việc tốt:

Ví dụ:

- Khi có 25 mô hình tỷ lệ sai của 1 mô hình là $\varepsilon = 0.35$(tạm coi).
- Để xác xuất mô hình kết hợp tạo ra 1 cái wrong predict là

$$\boxed{\sum_{i=13}^{25} \binom{25}{i} \varepsilon^i (1-\varepsilon)^{25-i} = 0.06}$$

  Từ 13 ông bị sai trở lên, các biến cố đều độc lập, theo thức bernulli ta có công thức trên. Tính coi chung trong trường hợp xấu nhất. Đôi khi nó còn thấp hơn.

Trong machine learning tồn tại định lý không có bữa trưa miến phí (No free luch theorem), tức là không tồn tại thuật toán tốt cho mọi ứng dụng và tập dữ liệu, vì các thuật toán ML thường dựa trên một tập các thâm số hoặc giả thiết nhất định về phân phối dữ liệu. Để tìm thuật toán phù hợp cần test nhiều lần, sau đó hiệu chỉnh để đạt hiệu quả cao.

Một cách khác có thể tăng độ chính xác trên tập dataset là kết hợp các mô hình với nhau. Phương pháp này gọi là esemble learning.

Các phương pháp Ensemble Learning được chia thành 3 loại sau đây:

1. Bagging (đóng bao): Voting nhiều ông chọn 1 thì theo 1, chọn nhiều 0 thì theo 0. Tất cả đều có cùng trọng số. Ví dụ 1 ông nhà nghèo với 1 ông nhà giàu đi bầu cử hoặc gì đấy

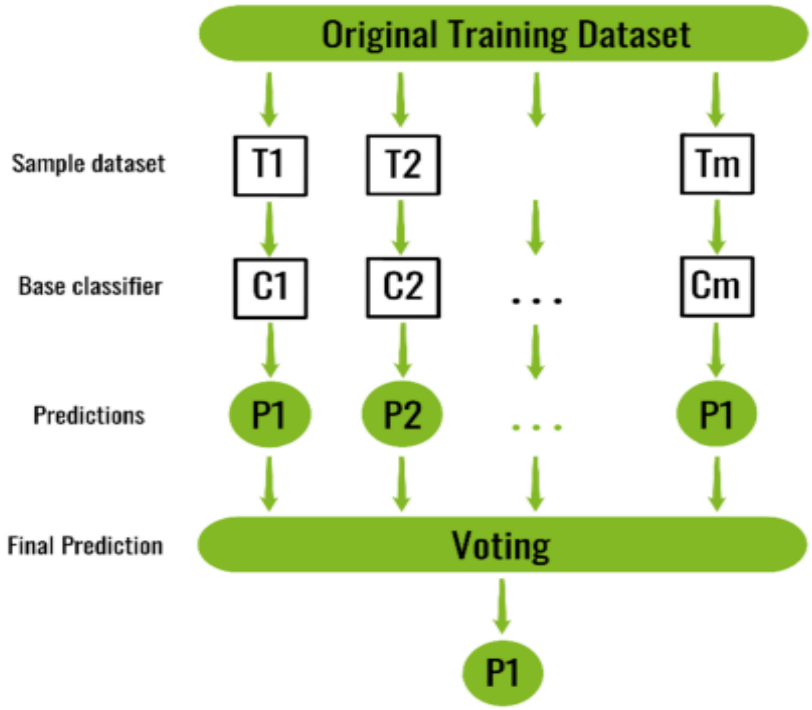
Nhược điểm chí mạng là bagging đang coi trọng số của tất cả mô hình là như nhau. Tức là ông dự báo tốt cũng ngang với ông dự báo vừa vừa.

→ Cho nhiều data liên quan hơn. Tăng trọng số của model còn lại. Ví dụ ông nhà giàu cho ý kiến và ông nhà nghèo cho ý kiến thì ý kiến của ông giàu tốt hơn.

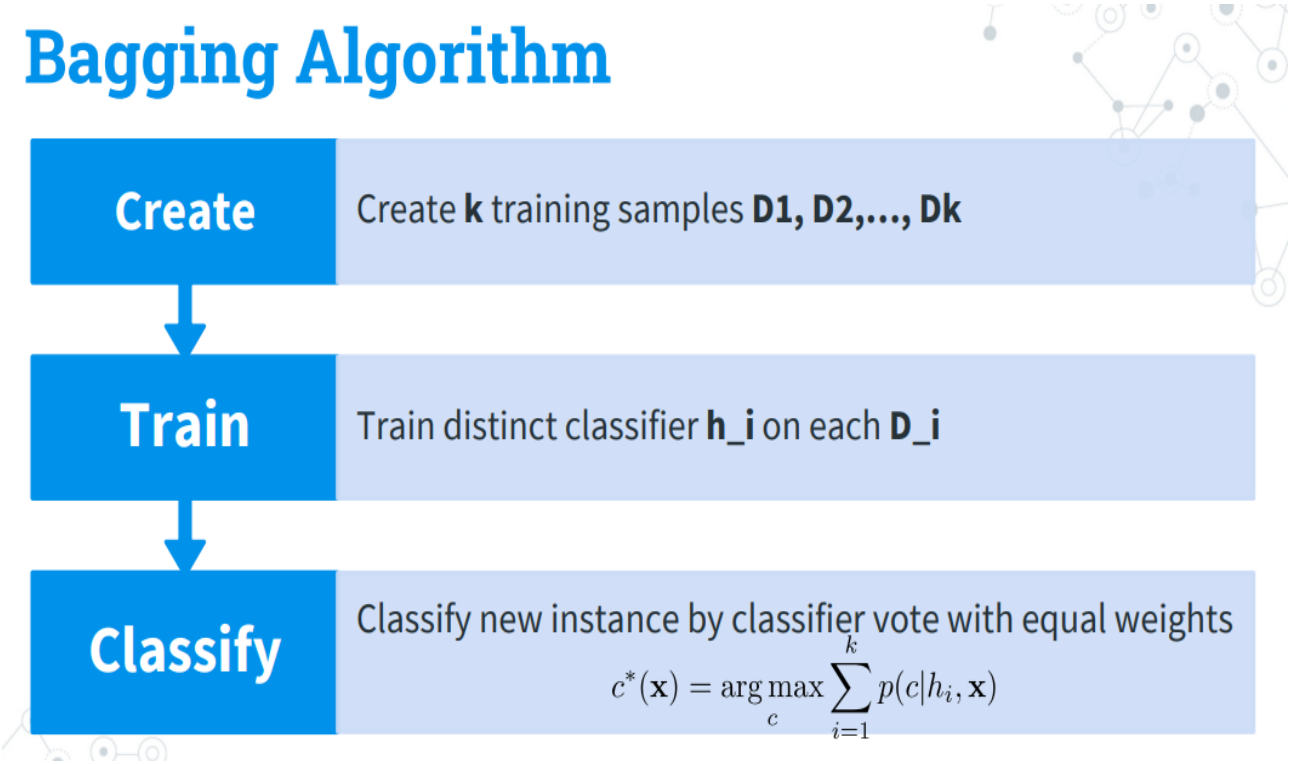
2. Boosting (tăng cường): Ban đầu huấn luyện. Sau 1 thời gian thấy 1 số trường hợp nó dự báo sai(con mèo đội mũ nhận thành con hổ).

Phải cho mô hình huấn luyện thêm vào datapoint khác và cộng vào mistake cho đến khi nào mô hình học tốt thì thôi. Về cuối mô hình không có quá nhiều sai số.

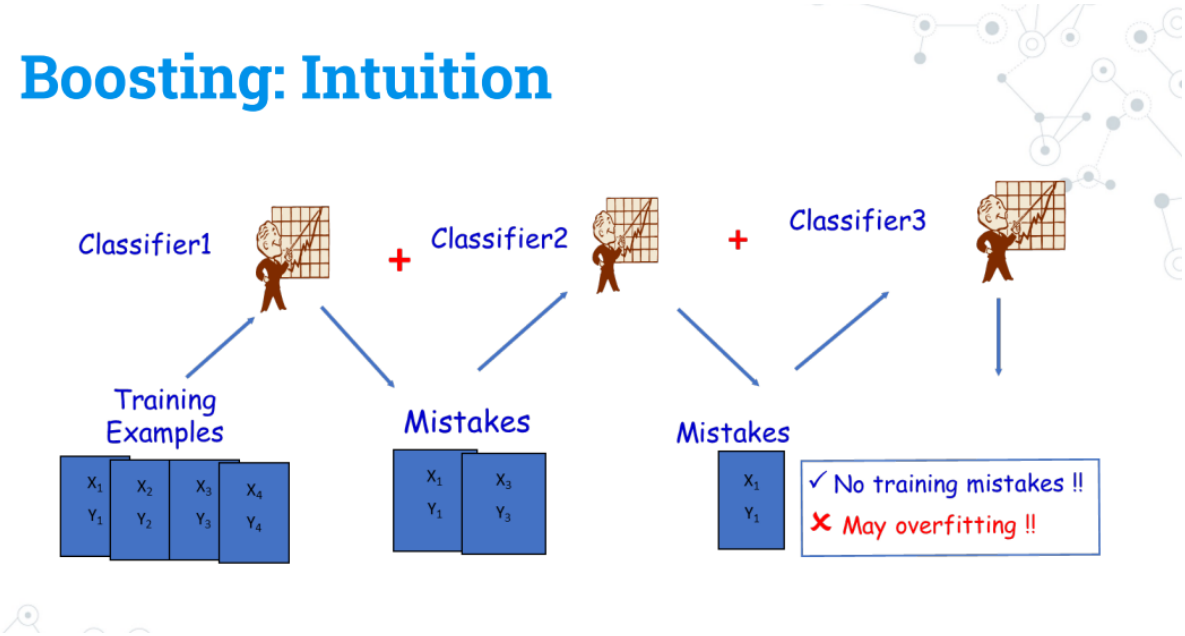
Trong quá trình huấn luyện ko nên để chỉ số boosting cao quá. Để nó học các datapoint khác.



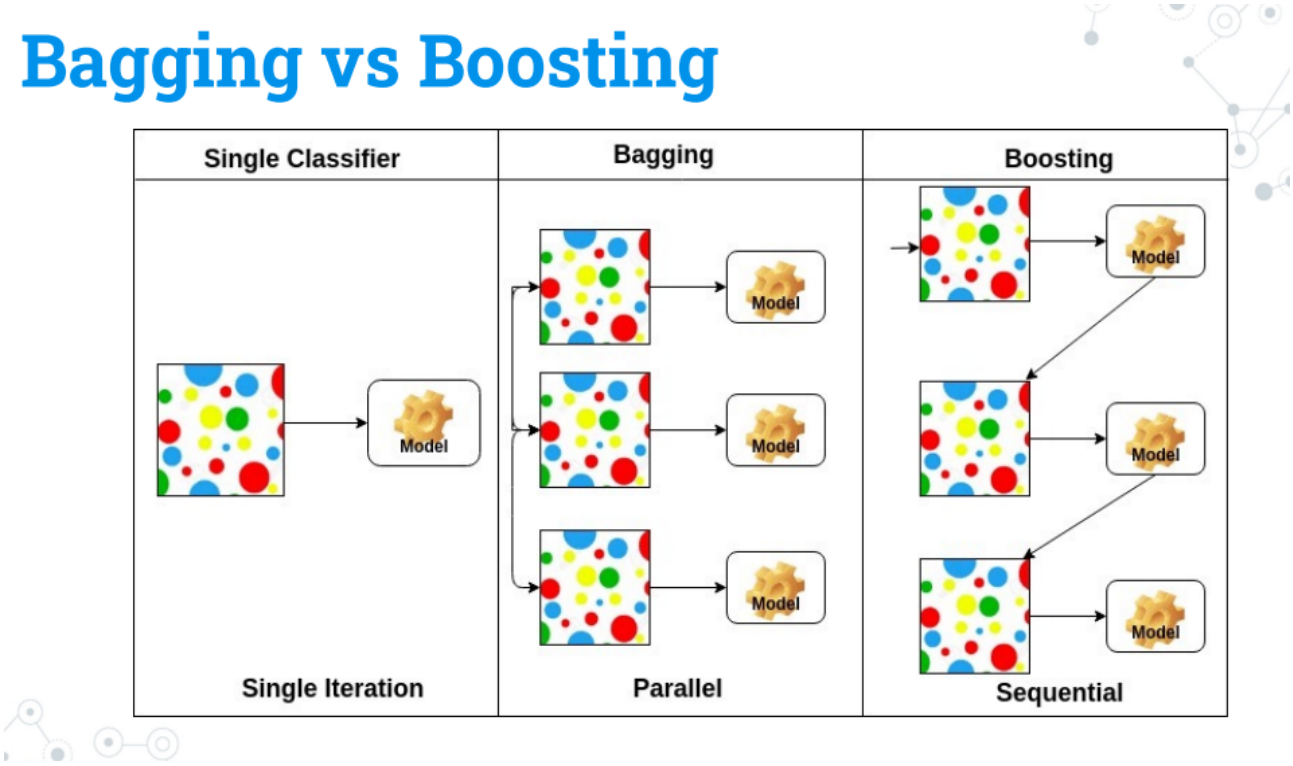
Hình 9: Bagging



Hình 10: Bagging algorithm



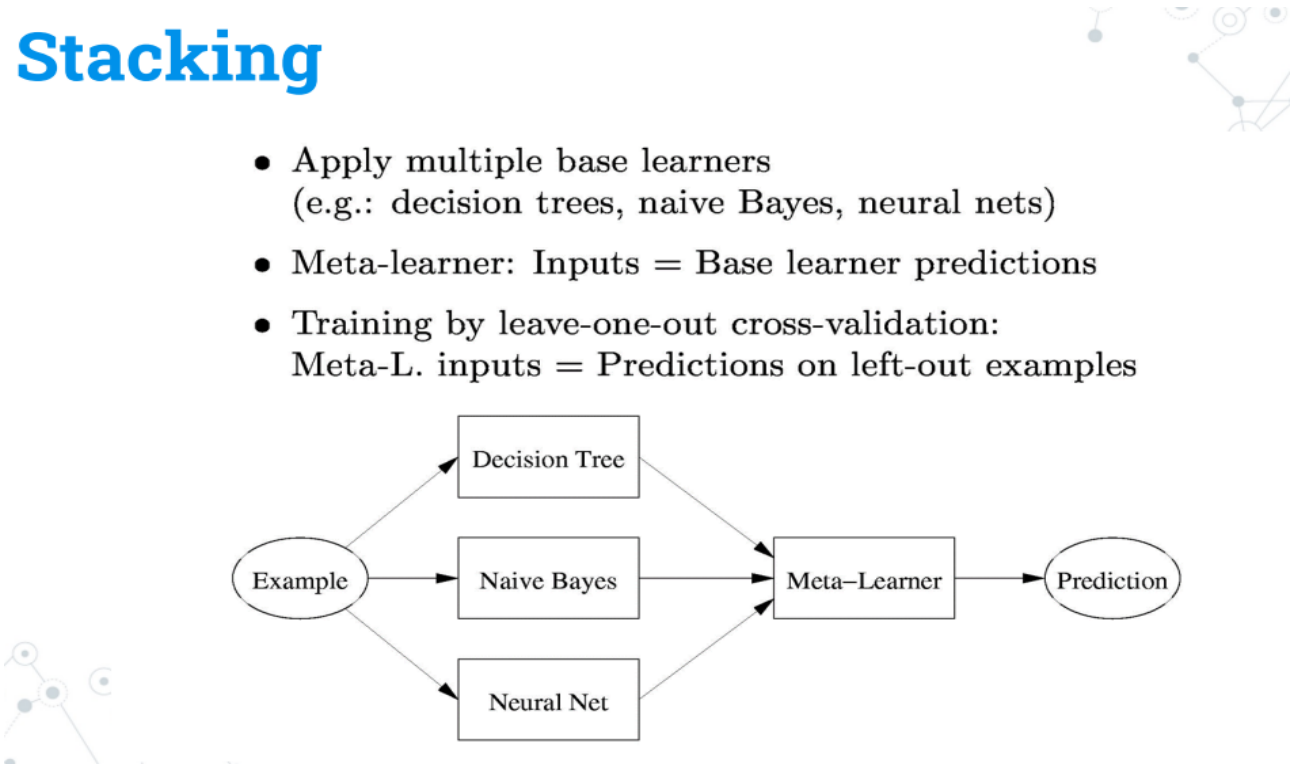
Hình 11: Boosting algori



Hình 12: So sánh bagging và boosting

- **Bagging:** Huấn luyện song song các mô hình và voting
- **Boosting:** Các sai số của mô hình đằng trước sẽ được bổ sung vào mô hình tiếp theo để mô hình học tốt hơn. Trong chuỗi này mỗi model sau sẽ học cách sửa những errors của model trước .

3. Stacking (xếp chồng): nâng cao hơn bagging. Thay vì combine bằng voting thì lấy đầu ra đằng trước làm input đằng sau. Ví dụ huấn luyện model nhận được con chó rồi con mèo thì chưa

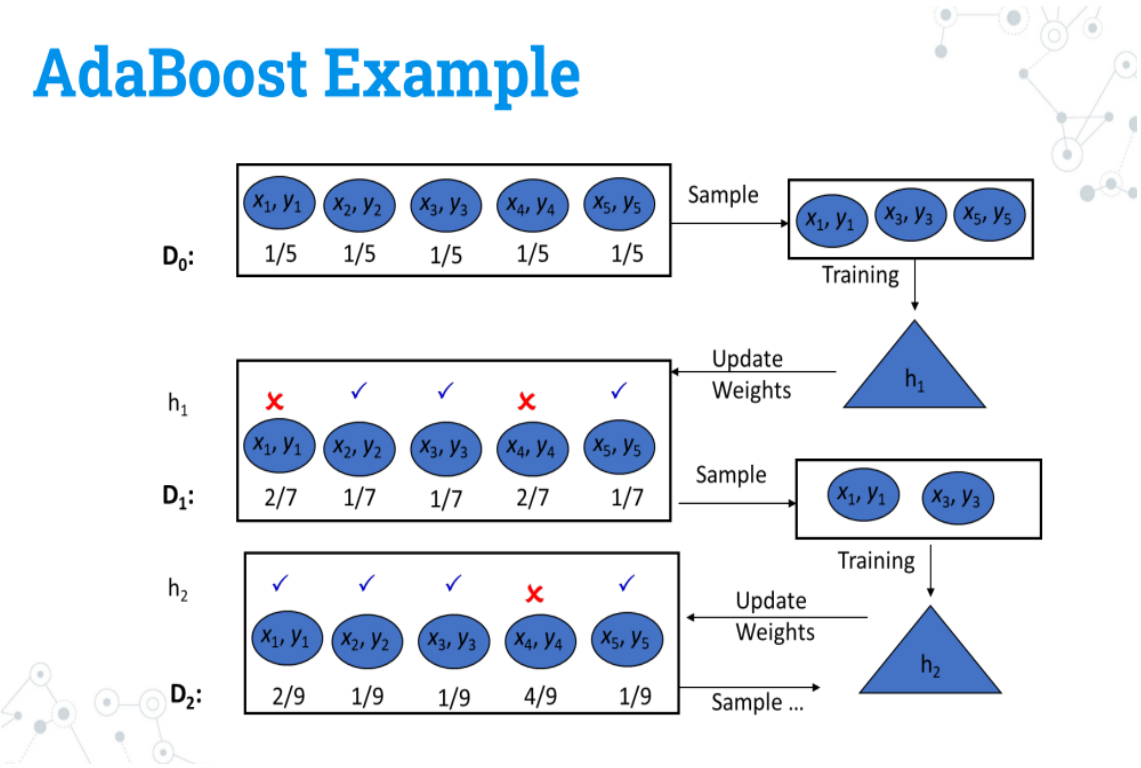


Hình 13: Stacking

4.3 Adboost là cái gì và nó làm được gì?

AdaBoost, viết tắt của **"Adaptive Boosting"** là một thuật toán máy học dùng để cải thiện hiệu suất của các mô hình phân loại yếu (weak classifiers) bằng cách kết hợp chúng lại để tạo thành một mô hình phân loại mạnh (strong classifier).

Thường được sử dụng trong các bài toán phân loại như phát hiện khuôn mặt, phân loại email spam, hay các ứng dụng khác trong lĩnh vực máy học.



Hình 14: ADaboost Example

Là một trong những thuật toán treebase đời đầu trong boosting

Ban đầu trọng số của 5 datapoint, quá trình huấn luyện là như nhau giữa các datapoint. Sau đó sample được 30% dữ liệu

Bổ sung thêm những thằng mới vào x_1 học sai, x_4 chưa đc học

Nên phải tăng tỉ lệ học của nó lên. Sau đó tiếp tục sample. x1 học được rồi, tiếp tục tăng weights lên.

Tăng tỷ lệ sample học sai, chưa học, giúp mô hình huấn luyện được với nhiều case sample, phân phối khác nhau.

4.4 Các bước thuật toán Adaboost

1.- Khởi tạo trọng số quan sát $w_i = \frac{1}{N}, \forall i = \overline{1, N}$.

2.- Lặp lại quá trình huấn luyện chuỗi mô hình ở mỗi bước b, $b = 1, 2, \ldots, B$ gồm các bước con:

a. Khớp mô hình $\hat{f}^b$ cho tập huấn luyện sử dụng trọng số w_i cho mỗi quan sát $(\mathbf{x}_i, y_i)$.

b. Tính sai số huấn luyện:

$$r_b = \frac{\sum_{i=1}^{N} w_i \mathbf{1}(y_i \neq \hat{f}^b(\mathbf{x}_i))}{\sum_{i=1}^{N} w_i}$$

Ở đây $\mathbf{1}(y_i \neq \hat{f}^b(\mathbf{x}_i))$ chính là những quan sát bị dự báo sai ở mô hình thứ b. Giá trị $r_b \in [0, 1]$.

c. Tính trọng số quyết định cho từng mô hình:

$$\alpha_b = \log(\frac{(1 - r_b)}{r_b})$$

d. Cập nhật trọng số cho từng quan sát:

$$w_i := w_i \exp[\alpha_b \mathbf{1}(y_i \neq \hat{f}^b(\mathbf{x}_i))]$$

với $\forall i = \overline{1, N}$. Như vậy ta có thể nhận thấy rằng:

$$w_i := \begin{cases} w_i & \text{if } y_i = \hat{f}^b(\mathbf{x}_i) \\ w_i \exp(\alpha_b) & \text{if } y_i \neq \hat{f}^b(\mathbf{x}_i) \end{cases}$$

Sau khi tính xong các trọng số w_i thì giá trị của chúng sẽ được chuẩn hoá bằng cách chia cho tổng $\sum_{i=1}^{N} w_i$.

3.- Cập nhật dự báo cuối cùng:

$$\hat{f}(\mathbf{x}) = \text{sign}(\sum_{i=1}^{p} \alpha_i \hat{f}^i(\mathbf{x})) \quad (1)$$

Trọng số α_i được tính ở bước thứ 2 thể hiện vai trò quan trọng trong việc ra quyết định của mô hình thứ i. Giá trị này được tính theo một hàm nghịch biến với sai số của mô hình. Chúng

4.5 Ví dụ

5 Support Vector Machine - SVM

5.1 Nhắc lại về công thức khoảng cách trong không gian n chiều

Trong không gian 3 chiều, để tính khoảng cách giữa một điểm $X^* = (x_1^*, x_2^*, x_3^*)$ và một mặt phẳng $w_1x_1 + w_2x_2 + w_3x_3 + b = 0$.

$$D = \frac{|w_1x_1^* + w_2x_2^* + w_3x_3^* + b|}{\sqrt{w_1^2 + w_2^2 + w_3^2}}$$

Thường, trong không gian n chiều, để tính khoảng cách giữa một điểm $X^* = (x_1^*, x_2^*, \ldots, x_n^*)$ và một siêu mặt phẳng $w_1x_1 + w_2x_2 + \cdots + w_nx_n + b = W^TX + b = 0$.

$$D = \frac{|W^TX^* + b|}{\sqrt{\sum_{i=1}^{n} w_i^2}}$$

5.2 Introduction about SVM

Hình 15: SVM

Là 1 thuật toán học có giám sát, sử dụng để phân loại các đối tượng thuộc các lớp khác nhau.

Thường SVM được sử dụng cho những bài toán phân loại có dataset phức tạp, kích thước nhỏ đến vừa.

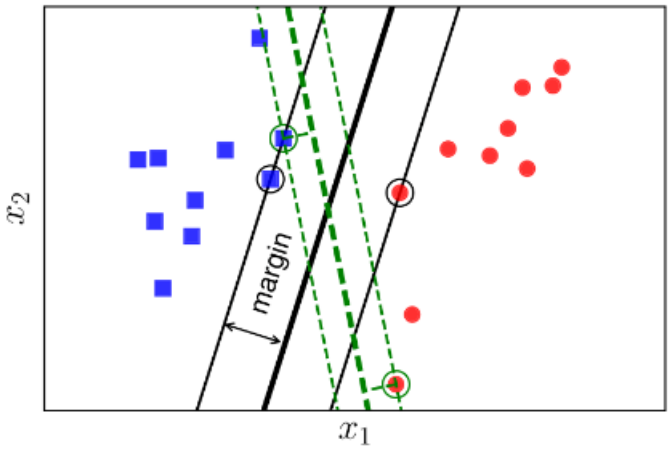
SVM sẽ sinh ra một hoặc nhiều siêu mặt phẳng (tức một mặt phẳng trong không gian

nhiều chiều) trong không gian cao chiều hoặc vô hạn chiều, để phục vụ mục đích phân loại, hồi quy,... Hiểu trực quan hơn, ta có sự phân tách tốt khi khoảng cách từ các điểm dữ liệu gần nhất tới biên từ cả 2 class là xa nhất có thể, vì thường biên càng lớn, điểm loss của mô hình càng thấp

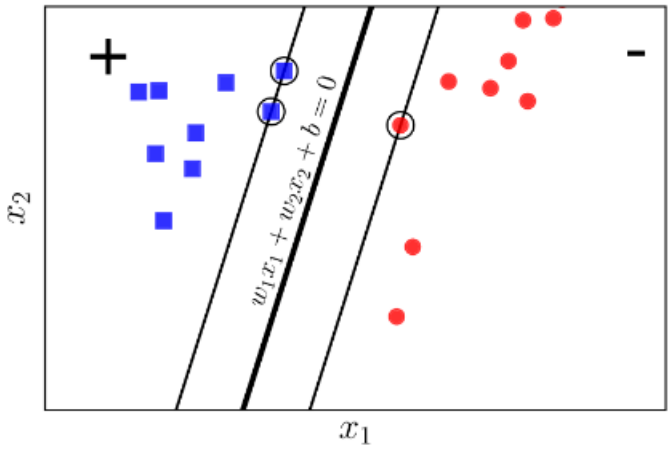
5.3 Optimization

Tập huấn luyện $(X_1, y_1), (X_2, y_2), \ldots, (X_N, y_N)$ khi X_i là vector mẫu dữ liệu có n chiều và y_i là nhãn ($y_i = 1$ hoặc $y_i = -1$)

Để đơn giản hóa, chúng ta xem xét không gian 2 chiều,



Chúng ta giả sử rằng $W^T X + b = w_1 x_1 + w_2 x_2 + b = 0$ là biên phân loại. Chúng ta có khoảng cách giữa một điểm dữ liệu (X_i, y_i) và biên phân loại



$$D = \frac{y_i(W^T X_i + b)}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

và độ lớn nhất của khoảng cách này chính là độ rộng biên,

$$margin = \min_i \frac{y_i(W^T X_i + b)}{\sqrt{\sum_{i=1}^{d} w_i^2}}$$

Chúng ta cần tối ưu hóa giá trị của độ rộng biên hoặc nói cách khác, chúng ta cần tìm W và b để làm cho độ rộng biên là lớn nhất.

$$(W, b) = \arg\max_{W,b} \left\{ \min_i \frac{y_i(W^T X_i + b)}{\sqrt{\sum_{i=1}^{d} w_i^2}} \right\}$$

Việc tối ưu hóa công thức trên là phức tạp, vì vậy chúng ta xem xét biên phân loại

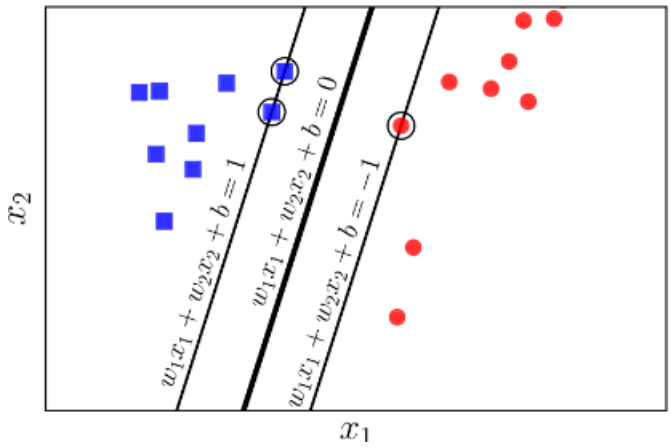
$$W^T X + b = w_1 x_1 + w_2 x_2 + b = 0$$

$$kW^T X + kb = kw_1 x_1 + kw_2 x_2 + kb = 0$$

Nếu nhân cả hai bên với $k > 0$, biên phân loại không thay đổi. Do đó, chúng ta có được tử số $y_i(W^T X_i + b)$ từ công thức độ rộng biên và giả định rằng

$$y_i(W^T X_i + b) = v$$

$$y_i(kW^T X_i + kb) = 1$$



Với giả định đó, chúng ta có

$$\forall i, y_i(kW^T X_i + kb) \geq 1$$

và tối ưu

$$(W, b) = \arg\max_{W,b} \left\{ \min_i \frac{y_i(W^T X_i + b)}{\sqrt{\sum_{i=1}^n w_i^2}} \right\}$$

$$= \arg\max_{W,b} \frac{1}{\sqrt{\sum_{i=1}^n w_i^2}}$$

$$\forall n, y_i(kW^T X_i + kb) \geq 1$$

Chúng ta sửa đổi để có được công thức tối ưu hóa đơn giản hơn

$$(W, b) = \arg\min_{W,b} \sqrt{\sum_{i=1}^n w_i^2}$$

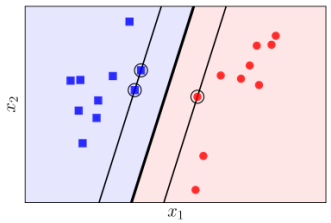
$$= \arg\min_{W,b} \sum_{i=1}^n w_i^2$$

$$\forall n, y_i(kW^T X_i + kb) \geq 1$$

Để tối ưu hóa công thức này, chúng ta sử dụng các thuật toán tối ưu hóa lồi.

5.4 Soft-margin SVM

Một vấn đề của SVM là mô hình này chỉ hoạt động với các tập dữ liệu có thể phân tách tuyến tính.

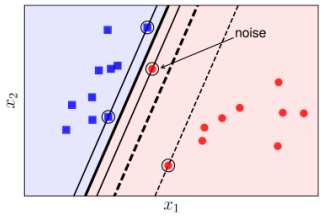


Còn đối với tập dữ liệu có thể phân tách tuyến tính nhưng chứa nhiễu thì sao?

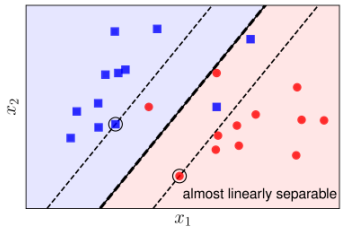
SVM có thể tìm ra một giải pháp trong trường hợp này, nhưng giải pháp không tốt lắm do độ rộng của đường biên quá hẹp.

Đó là lý do tại sao chúng ta gọi SVM là mô hình nhạy cảm đối với nhiễu.

Và bây giờ, đối với tập dữ liệu gần như có thể phân tách tuyến tính?

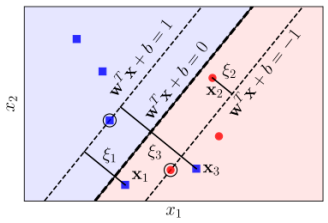


SVM không thể tìm ra một giải pháp trong trường hợp này!



Tuy nhiên, trong cả hai trường hợp trên, nếu chúng ta bỏ qua một số điểm dữ liệu (cho phép chúng rơi vào trong đường biên), SVM có thể đề xuất một giải pháp thực sự tốt với một đường biên lớn (đường chấm).

Đó là lý do tại sao chúng ta cần một phiên bản khác của SVM - SVM có đường biên mềm, và phiên bản ban đầu của SVM trở thành SVM có đường biên cứng.



Trong SVM có đường biên mềm, chúng ta có thể bỏ qua một số điểm dữ liệu, nhưng chúng ta phải tối thiểu hóa số điểm dữ liệu bị bỏ qua.

Vì vậy, bài toán tối ưu hóa của SVM có đường biên mềm là sự kết hợp giữa việc tối đa hóa độ rộng của đường biên và tối thiểu hóa số điểm dữ liệu bị bỏ qua.

Để tối đa hóa độ rộng của đường biên, tương tự như SVM có đường biên cứng, chúng ta tối ưu hóa

$$(W, b) = \arg\min_{W,b} \sum_{i=1}^{d} w_i^2.$$

Để tối thiểu hóa số điểm dữ liệu bị bỏ qua, đối với mỗi điểm dữ liệu x_i trong tập dữ liệu, chúng ta đề xuất một biến gọi là ξ để kiểm tra vị trí của chúng đối với đường biên.

Đối với mỗi điểm dữ liệu x_i trong tập dữ liệu:

Nếu x_i được phân loại đúng và x_i nằm bên ngoài đường biên, thì $\xi_i = 0$. Nếu x_i được phân loại đúng và x_i nằm bên trong đường biên, thì $0 < \xi_i < 1$. Nếu x_i không được phân loại, thì $\xi_i > 1$. Chúng ta có bài toán tối ưu hóa sau cho SVM có đường biên cứng:

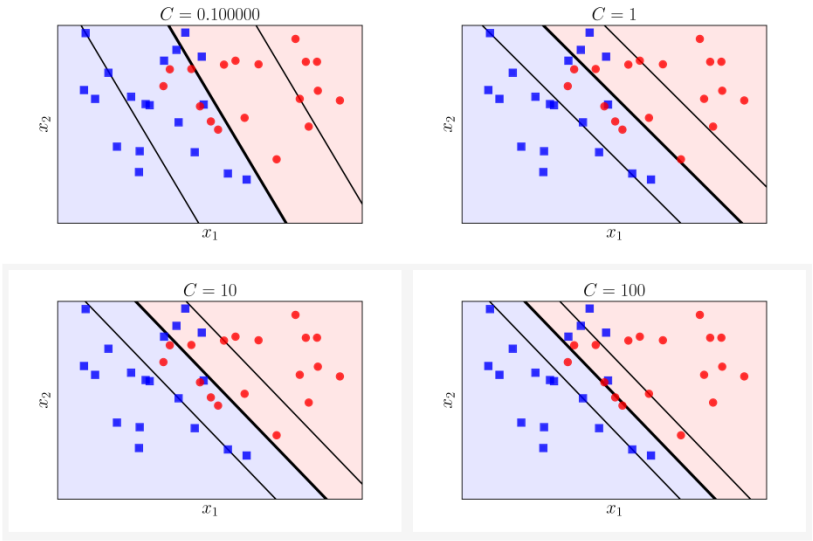
$$(W, b) = \arg\min_{W,b} \sum_{i=1}^{d} w_i^2 \forall n, y_i(kW^T X_i + kb) \geq 1$$

We have the following optimization problem for Soft-margin SVM with ξ in the optimization condition.

$$(W, b) = \arg\min_{W,b} \sum_{i=1}^{d} w_i^2 + C \sum_{i=1}^{N} \xi_i \forall n, y_i(kW^T X_i + kb) \geq 1 - \xi_i$$

Vai trò của tham số C ở đây là cân bằng giữa các thành phần trong hàm mục tiêu.

Với C nhỏ, SVM có đường biên mềm sẽ bỏ qua nhiều điểm để tối đa hóa độ rộng của đường biên. Với C lớn, SVM có đường biên mềm sẽ cố gắng bỏ qua ít điểm nhất. Trong trường hợp tập dữ liệu có thể phân tách tuyến tính, SVM có đường biên mềm sẽ tối ưu hóa để trở thành SVM có đường biên cứng tối ưu.

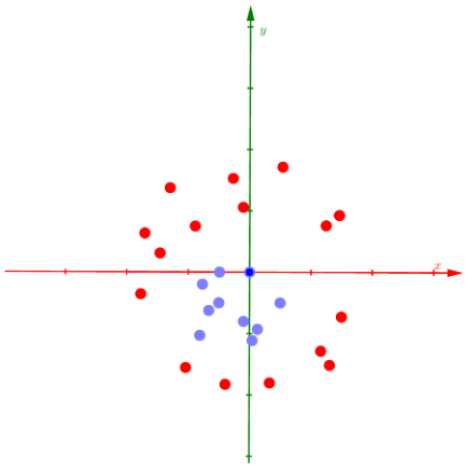


5.5 Kernel SVM

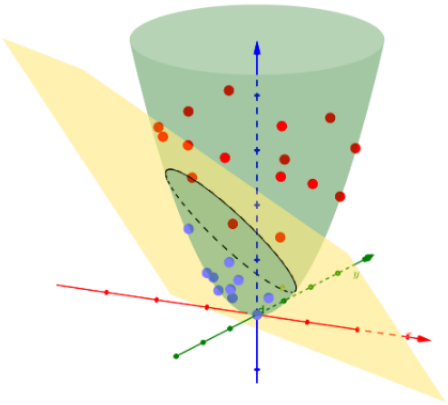
SVM ban đầu (SVM có đường biên cứng) hoạt động với các tập dữ liệu có thể phân tách tuyến tính.

SVM có đường biên mềm hoạt động với các tập dữ liệu có thể phân tách tuyến tính nhưng chứa nhiễu hoặc gần như có thể phân tách tuyến tính.

Vậy và tập dữ liệu không thể phân tách tuyến tính?



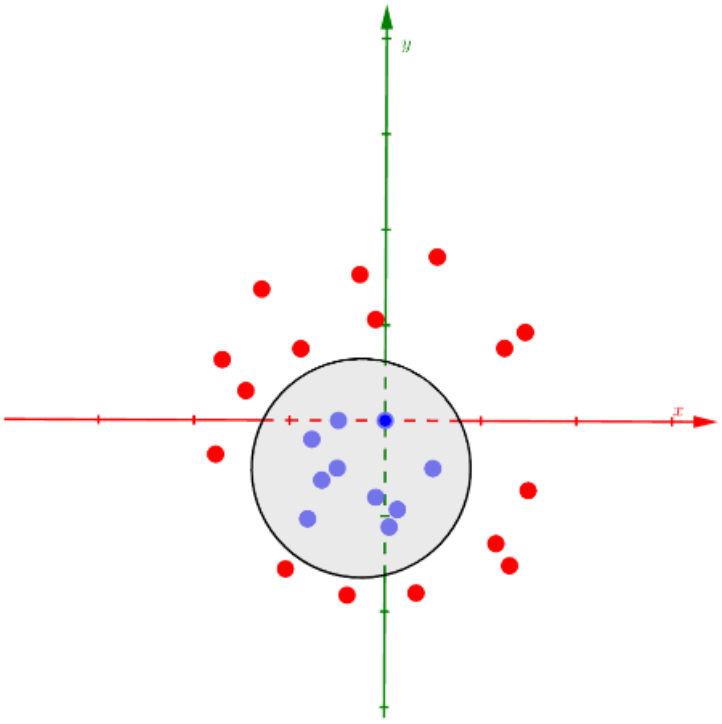
Trong trường hợp này, SVM Kernel sẽ đề xuất một hàm Φ để ánh xạ mỗi điểm dữ liệu x_i trong tập dữ liệu sang không gian dữ liệu mới. Và trong không gian này, tập dữ liệu không thể phân tách tuyến tính có thể trở thành tập dữ liệu có thể phân tách tuyến tính hoặc gần như có thể phân tách tuyến tính.



Trong không gian dữ liệu mới, chúng ta có thể sử dụng SVM có đường biên mềm hoặc SVM có đường biên cứng để tìm ra ranh giới phân loại. Với ranh giới phân loại này trong không gian dữ liệu mới, chúng ta có thể tìm ranh giới phân loại trong không gian dữ liệu ban đầu.

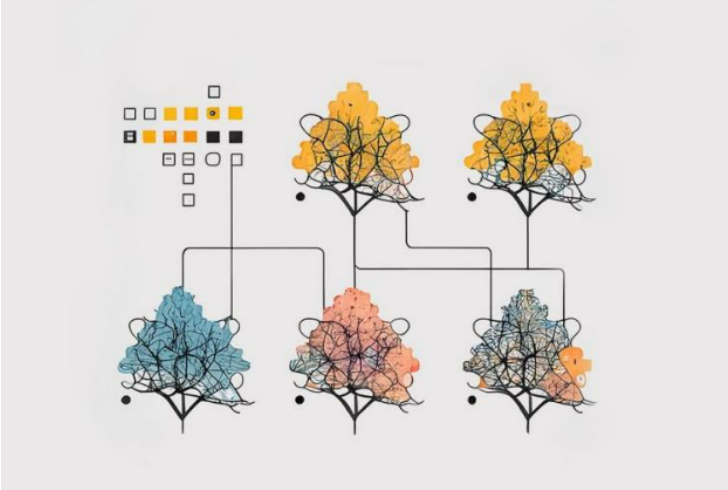
Để tiết kiệm chi phí tính toán, hàm Φ có thể là **kernel function**. Đó là lý do tại sao chúng ta gọi nó là **SVM Kernel.**

Thay vì ánh xạ trực tiếp điểm dữ liệu x_i trong tập dữ liệu sang không gian dữ liệu mới, hàm nhân cố gắng tính toán mối quan hệ giữa các điểm dữ liệu trong không gian dữ liệu mới.


y
x

6 Random forest

6.1 Random forest model



Hình 16: Random forest

Mô hình **forest** được huấn luyện dựa trên sự phối hợp giữa luật kết hợp **(ensembling)** và quá trình **lấy mẫu tái lặp** (boostrapping). Cụ thể thuật toán này tạo ra nhiều **cây quyết định** mà mỗi cây quyết định được huấn luyện dựa trên nhiều mẫu con khác nhau và kết quả dự báo là **bầu cử (voting)** từ toàn bộ những cây quyết định

Có thể dùng cho cả phân loại cả hồi quy.

⇒ một kết quả dự báo được tổng hợp từ nhiều mô hình nên kết quả của chúng sẽ không bị chệch. Đồng thời kết hợp kết quả dự báo từ nhiều mô hình sẽ có phương sai nhỏ hơn so với chỉ một mô hình.⇒ khắc phục được hiện tượng quá khớp.

Ý tưởng: để đưa 1 phán đoán thì chúng ta dùng phán đoán của rất nhiều cây khác nhau sau đó chúng ta lấy trung bình của tất cả cây đó. (tận dụng sức mạnh của nhiều cây khác nhau – để đưa ra 1 quyết định con người cần hỏi nhiều người, bỏ phiếu – do khách quan – nhiều người cùng đưa ra 1 ý kiến thì ý kiến đó tốt – ước lượng không chệch).

Cách triển khai đơn giản dùng các cây hoàn toàn ngẫu nhiên. Cho đến này random forest vẫn là 1 trong những pp hiệu quả nhất.

6.2 3 thành phần chính của RF

Randomization and no pruning:

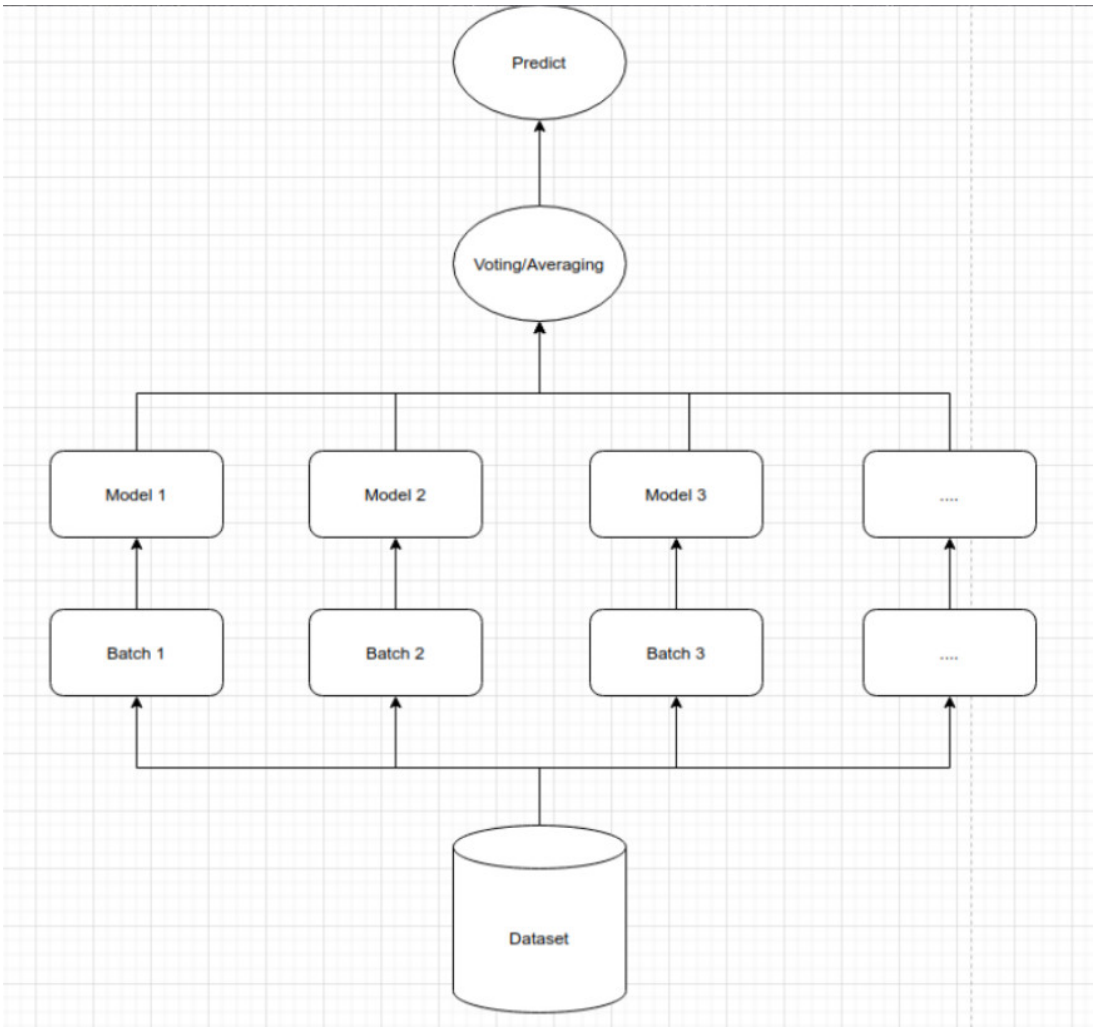
- Với mỗi cây và tại mỗi nút, chúng ta chọn ngẫu nhiên một tập con của các thuộc tính.

- Tìm cách phân chia tốt nhất và sau đó phát triển các cây con thích hợp.
- Mỗi cây sẽ được phát triển đến kích thước lớn nhất mà không cần cắt tỉa.

Combination: Mỗi lần phán đoán, đứa cái phán đoán đó cho tất cả các cây và lấy trung bình

Bagging: mỗi cây sinh ra tập ngẫu nhiêu từ tập ban đầu.

6.3 Lấy mẫu tái lặp



Lấy mẫu có trùng lặp: 1 mẫu lấy ra có thể được tái sử dụng

Các vấn đề xoay quay việc lấy mẫu tái lặp:

Đầu vào: tập data D

Cách huấn luyện: Training independence

Ví dụ với cây thứ i:

Sinh tập dữ liệu train D_i bằng cách lấy mẫu có trùng lập. Sao cho D_i = D. Sau đó phát triển D_i từ D (mỗi đỉnh cho 1 thuộc tính). Cho cây phát triển hết cỡ, không cắt tỉa.

⇒ overfiting.

D_i có thể chứa khoảng 2/3(đã chứng minh) các quan sát trong D (bagging) – Còn lại là trùng lặp.

Tại sao D_i lại có các phần tử trùng nhau thay vì là các hoán vị.

⇒ Để tăng tính ngẫu nhiên, tăng cái verrian(sự khác biệt giữa các cây).

Sự ngẫu nhiên thể hiện ở 2 chỗ, các phần tử có thể trùng. Và chọn 1 phát tử bất kỳ làm node.

⇒ Tạo ra sự khác biệt nhiều. Mỗi 1 cây tạo ra sẽ cố gắng tạo ra 1 vài đặc trưng của không gian dữ liệu 2 cây bất kỳ trùng lặp gần = 0.

Để cho thuật toán đạt hiệu quả thì mỗi cây phải **độc lập**, khác biệt (nhớ lại Becnulli)

6.4 Thuật toán

- Input: training data D
- Learning: grow K trees as follows ¨
  - Generate a training set Di by sampling with replacement from
  - Learn the ith tree from Di
  - At each node:
    - Select randomly a subset S of attributes.
    - Split the node into subtrees according to S.
  - Grow this tree upto its largest size without pruning.
- Prediction: taking the average of all predictions from the individual trees.

RF là 1 ví dụ cho unsample learning, adaboosts, multiBoost

6.5 Tại sao nói mô hình Random forest là tốt

RF được so sánh rộng rãi với các phương pháp khác

- Bởi Fernández-Delgado và cộng sự.

- Sử dụng 55 bài toán khác nhau.
- Sử dụng độ chính xác trung bình (uP) làm thước đo

6.6 Source code tham khảo

Tham khảo source tại đây:

7 Tài liệu tham khảo

Tài liệu

[1] Report Machine Learning(MCI)
https://drive.google.com/drive/folders/1-vRdFZmvBtUCkCd6KM-GYvLB7TFILiag,

[2] Slide của mọi người
googleshet.com,

[3] OpenAI Chat Platform
https://chat.openai.com,

[4] Bài giảng Model Performance Improvements - Vương Hải Yến
https://chat.openai.com,

[5] Bài toán phân loại lớp,
https://thinkingneuron.com/german-credit-risk-classification.../,

[6] Documentation Tek4 - Course Machine Learning,
https://tek4.vn,

[7] Dũng lại lập trình - course AI
https://amis.misa.vn/31462/van-hoa-doanh-nghiep-viettel/

[8] Machine learning - Thân Quang Khoát
https://users.soict.hust.edu.vn/khoattq/ml-dm-course/
L7-Random-forests.pdf

[9] Các mô hình học máy - Thân Quang Khoát
https://users.soict.hust.edu.vn/khoattq/ml-dm-course/
L7-Random-forests.pdf

[10] Bài giảng các mô hình ML freecodecamp
https://youtobe.com

[11] 8 base algo in Machine Learning
https://datasciencedojo.com/blog/machine-learning-algorithms-explanation/
?fbclid=IwAR1ygGCnuLyQa4oHeT-3InAOob14Pv9HHAHfT7S-tqbIR1gG2PwWiu7d0Jg#

[12] Total documentation GitHub for Vietnamese
https://github.com/afshinea/stanford-cs-230-deep-learning/blob/
master/vi/super-cheatsheet-deep-learning.pdf?fbclid=IwAR2y9l8_
Af6-fiRinqGf1abiIwQkkfxATblKeMLQr48gibax3aLnIzsNY4o