PHÂN HIỆU ĐẠI HỌC THỦY LỢI
BỘ MÔN CÔNG NGHỆ THÔNG TIN

BÁO CÁO BÀI TẬP MÔN HỌC
MÔN KHAI PHÁ DỮ LIỆU

Đề tài :
Phân loại cài đặt thuật toán ADBOOST

Giảng viên: Vũ Thị Hạnh

Sinh viên thực hiện: Bùi Văn Nhã

Lớp : S21-60TH1

I. GIỚI THIỆU VỀ THUẬT TOÁN ADABOOST

Thuật toán học adaboost:

AdaBoost (Adaptive Boost) là một thuật toán học mạnh, giúp đẩy nhan việc tạo ra một bộ phân loại mạnh (strong classifier) bằng cách chọn các đặc trưng tốt trong một họ các bộ phân loại yếu (weak classifer - bộ phân loại yếu) và kết hợp chúng lại tuyến tính bằng cách sử dụng các trọng số. Điều này thật sự cải thiện dần độ chính xác nhờ áp dụng hiệu quả một chuỗi các bộ phân loại yếu.

1. Cho một tập gồm n mẫu có đánh dấu (x1, y1), (x2, y2),…., (xn, yn) với xk ∈ (xk1, xk2,…, xkm) là vector đặc trưng và yk ∈ (-1,1) là nhãn của mẫu (1 ứng với object, -1 ứng với backgound).
2. Khởi tạo trọng số ban đầu cho tất cả các mẫu: với m là số mẫu đúng (ứng với object và y = 1) và l là số mẫu sai (ứng với background và y = -1)

$$W_{1,k} = \frac{1}{2m}, \frac{1}{2l}$$

3. Xây dựng T weak classifiers Lặp t=1,..., T Với mỗi đặc trưng trong vector đặc trưng, xây dựng một weak classifier hj với ngưỡng θj và lỗi εj:

$$\varepsilon_j = \sum_k^n W_{1,k} |h_j(x_k) - y_k|$$

- Chọn ra hj với εj nhỏ nhất, ta được ht: ht: X→{1, -1} Cập nhật lại trọng số:

$$W_{t+1,k} = \frac{W_{t,k}}{Z_t} \times \begin{cases} e^{-\alpha_t}, & h_t(x_k) = y_k \\ e^{\alpha^t} & h_t(x_k) \neq y_k \end{cases}$$

- Trong đó:

$$\alpha_t = \frac{1}{2} \ln \left(\frac{(1-\varepsilon_j)}{\varepsilon_j}\right)$$

- Zt: Hệ số dùng để đưa Wt+1 về đoạn [0, 1]

4. Strong classifier được xây dựng:

$$H(x) = Sign(\sum_{t=1}^{T} \propto_t h_t(x))$$

II. HIỂU BIẾT VỀ THUẬT TOÁN ADABOOT

- Adaboot dùng để dự đoán kết quả ví dụ như dự đoán kết quả người có bị ung thư hay không trong hồ sơ y tế của bệnh nhân.
- Thuật toán sẻ chia ra từng hiệp.
- Hiệp 1 Adaboot sẻ lấy mẫu trên tập huấn luyện và kiểm tra độ chính xác của mỗi người học là bao nhiêu. Kết quả trả về là người học có độ chính xác cao nhất.
- Ngươi học tốt nhất sẻ được đánh trọng số dựa vào độ chính xác .
- Hiệp 2: AdaBoost một lần nữa cố gắng tìm được người học có độ chính xác cao nhất. mẫu dữ liệu của tập huấn luyện hiện đang bị ảnh hưởng nhiều hơn bởi các trọng số phân lớp sai
- Hiệp cuối: ta còn lại một quần thể các người học được đánh trọng số sau nhiều lần được huấn luyện lặp đi lặp lại ở các hiệp trước trên các mẫu dữ liệu bị phân lớp sai

III. Môt số phân tích về đường cong

- Khởi tạo trọng số quan sát

$$t\, w_i = \frac{1}{N}, \forall i = 1, N.$$

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

a) Khớp mô hình f^b cho tập huấn luyện sử dụng trọng số wi cho mỗi quan sát (xi,yi).

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 1(yi≠f^b(xi)) chính là những quan sát bị dự báo sai ở mô hình thứ b. Giá trị rb∈[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 ∀i=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 wi thì giá trị của chúng sẽ được chuẩn hoá bằng cách chia cho tổng ∑i=1Nwi.

.- Cập nhật dự báo cuối cùng:

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

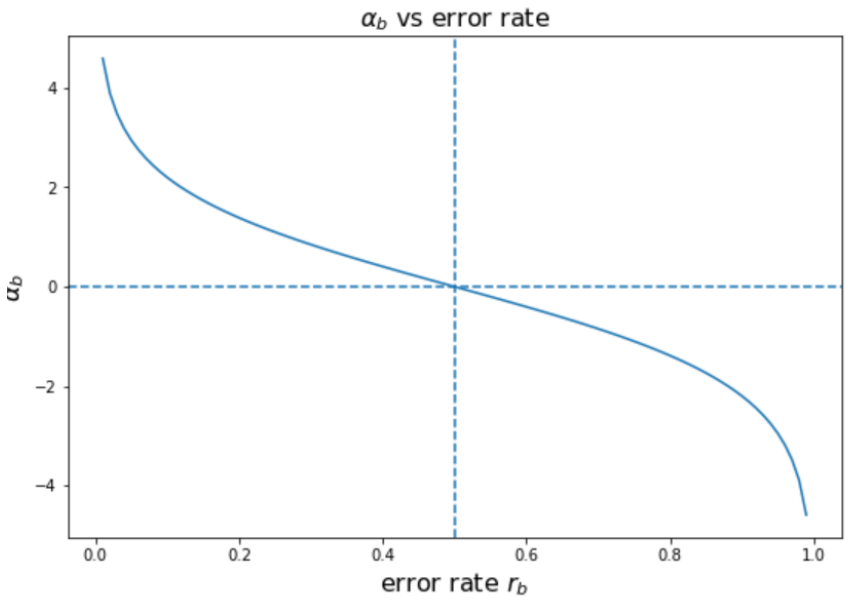
VD : Trọng số αi được tính ở bước thứ 2 thể hiện vai trò quan trọng trong việc ra quyết định của mô hình thứ i. Giá trị này được tính theo một hàm nghịch biến với sai số của mô hình. Chúng ta cùng phân tích hàm này bên dưới:

```
import numpy as np
import matplotlib.pyplot as plt

plt.figure(figsize = (9, 6))
r = np.linspace(0.01, 0.99, 100)

def alpha(z):
  return np.log((1-z)/z)

y = alpha(r)
plt.plot(r, y)
plt.xlabel(r'error rate $r_b$', fontsize=16)
plt.ylabel(r'$\alpha_b$', fontsize=16)
plt.axvline(0.5, linestyle='--')
plt.axhline(0, linestyle='--')
plt.title(r'$\alpha_b$ vs error rate', fontsize=16)
plt.show()
```



Giá trị của *trọng số quyết định* αb theo sai số rb. Đây là một hàm nghịch biến theo rb và có giá trị từ $(-\infty,+\infty)$

Bên dưới ta sẽ xét 3 trường hợp đối với sai số dự báo rb:

- Khi rb=0.5 tương ứng với kết quả từ một mô hình dự báo ngẫu nhiên. Trường hợp này có αb=0. Khi đó mô hình không có đóng góp gì vào hàm dự báo được thể hiện ở công thức (1). Điều này là hợp lý vì một giá trị dự báo ngẫu nhiên thì không có ích cho việc phân loại. Đồng thời trọng số sau cập nhật wiexp(αb)=wi, tức là vai trò của các quan sát được giữ cố định.
- Khi rb tiến dần tới 0, chẳng hạn rb=0.1, tương ứng với mô hình dự báo có tỷ lệ sai số thấp và đây là một mô hình khá mạnh. Khi đó $\alpha_b = \log \frac{1-0.1}{0.1} = \log 9 = 2.197$ và $\exp(\alpha_b) = 9$. Như vậy đối với những quan sát bị dự báo sai thì trọng số của nó được gấp lên 9 lần, điều này giúp cho những mô hình sau sẽ điều chỉnh lại *cây quyết định* sao cho tập trung vào dự báo đúng những quan sát này. Đồng thời αbf^b(x)=2.197 f^b(x) cho thấy các dự báo từ mô hình này được đánh giá rất cao và góp phần gia tăng điểm số dự báo cuối cùng theo như công thức (1).
- Khi rb tiến dần tới 1, chẳng hạn rb=0.9 cho thấy đây là một mô hình rất yếu vì có tỷ lệ sai số dự báo cao. Khi đó $\alpha_b = \log \frac{1-0.9}{0.9} = \log \frac{1}{9} = -2.197$ là một giá trị âm tương đối nhỏ và exp(αb)=19 là một giá trị gần 0. Như vậy trọng số wiexp(αb) sẽ bị giảm gấp 9 lần so với wi. Lưu ý rằng trong trường hợp này mô hình đang dự báo hầu hết là sai nên nếu mô hình dự báo sai thì dường như những quan sát đó lại dễ được dự báo đúng và ít quan trọng. Điều này cũng giống như một người dự báo sai tới 90% thì khả năng ta lấy kết quả ngược lại của anh ta sẽ được mô hình dự báo đúng 90% và những trường hợp anh ta dự báo sai thường dễ dàng được phân loại đúng nhờ làm ngược lại. Do đó ta cần giảm trọng số wi cho những quan sát mà mô hình dự báo sai, trong trường hợp này là giảm đi 9 lần. Đồng thời đóng góp từ kết quả dự báo vào mô hình

là αbf^b(x)=−2.197 f^b(x) cho thấy kết quả từ mô hình này sẽ được cập nhật ngược chiều vào điểm số cuối cùng. Điều này cũng giống như chúng ta làm ngược lại gợi ý của một người hay phán đoán sai để thu được phán đoán đúng. Quá trình *tăng cường* mô hình sẽ tiếp tục như vậy cho đến khi mô hình đạt số lượng tối đa hoặc toàn bộ các quan sát trên tập kiểm tra được phân loại đúng. Một lưu ý đó là các mô hình *cây quyết định* con trong phương pháp *tăng cường* thường có độ sâu thấp, thông thường chỉ gồm 1 node gốc với hai node lá, trường hợp cây quyết định chỉ gồm một node gốc được gọi là mô hình *gốc cây* (*stump*). Sở dĩ chúng ta không cần yêu cầu các *cây quyết định* phải quá phức tạp là để ngăn ngừa hiện tượng *quá khớp* có thể xảy ra và đồng thời tăng khả năng giải thích cho mô hình.

- Số lần lặp trên tập dữ liệu

Bước 1: Bộ phân loại yếu (ví dụ: gốc quyết định) được thực hiện trên dữ liệu huấn luyện dựa trên các mẫu có trọng số. Ở đây, trọng lượng của mỗi mẫu cho biết tầm quan trọng của việc phân loại chính xác. Ban đầu, đối với gốc cây đầu tiên, chúng tôi cho tất cả các mẫu có trọng lượng bằng nhau.

Bước 2: Chúng tôi tạo một gốc quyết định cho từng biến và xem mỗi gốc phân loại mẫu cho các lớp mục tiêu của chúng tốt như thế nào. Ví dụ, trong sơ đồ dưới đây, chúng tôi kiểm tra Tuổi, Ăn Đồ ăn vặt và Tập thể dục. Chúng tôi sẽ xem có bao nhiêu mẫu được phân loại đúng hoặc không chính xác là Phù hợp hoặc Không phù hợp cho từng gốc cây riêng lẻ.

Bước 3: Thêm trọng lượng được chỉ định cho các mẫu được phân loại không chính xác để chúng được phân loại chính xác trong gốc quyết định tiếp theo. Trọng lượng cũng được chỉ định cho mỗi bộ phân loại dựa trên độ chính xác của bộ phân loại, có nghĩa là độ chính xác cao = trọng lượng cao!

Bước 4: Lặp lại từ Bước 2 cho đến khi tất cả các điểm dữ liệu đã được phân loại chính xác hoặc đã đạt đến mức lặp tối đa.

Link Githhub bài tập cá nhân

https://github.com/VanNha586/buivannha_baitapcanhan