AI VIET NAM – COURSE 2024

AdaBoost and Gradient Boost

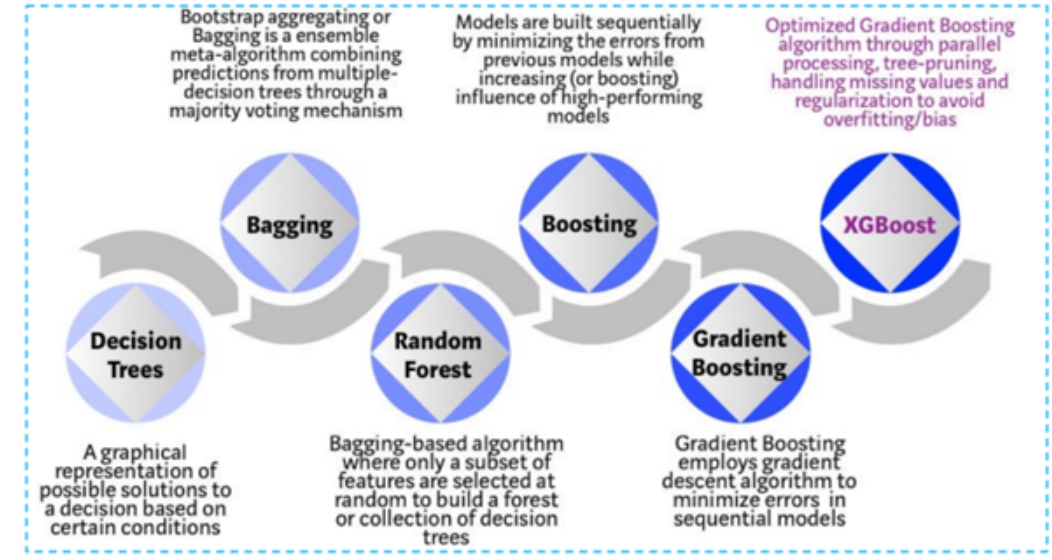
Ngày 14 tháng 3 năm 2024

Date of publication:	14/03/2024
Authors:	AIO 2024
Sources:	Link of Module 4 - AIO 2023
Keywords:	AdaBoost, Gradient Boost, Samme, Boosting, Stump, Amount of Say
Summary by:	C.T.Kiet

1. **Tóm tắt**

- Bài viết cô đọng một số nội dung quan trọng của bài giảng về Adaboost và Gradient Boost. Về bản chất, cả 2 thuật toán đều giúp cải thiện hiệu suất của mô hình máy học (Machine Learning) nhằm tối ưu kết quả dự đoán, tuy nhiên vẫn tồn tại sự khác biệt giữa chúng. AdaBoost tập trung nhiều hơn vào việc tối ưu hóa trọng số (weights) để các mẫu dữ liệu khó hoặc cho kết quả sai lệch được tập trung huấn luyện nhiều lần trong khi Gradient Boost tiếp cận dưới góc độ tối ưu hóa liên tục dựa trên lỗi của mô hình trước đó, gọi là các phần dư sai biệt (residual error). Đây là hai thuật toán quen thuộc thường được sử dụng cho các bài toán dự đoán (regression) và phân loại (classification) mức độ từ đơn giản đến phức tạp.

2. **Sơ đồ phát triển thuật toán Machine Learning họ XGBoost:**

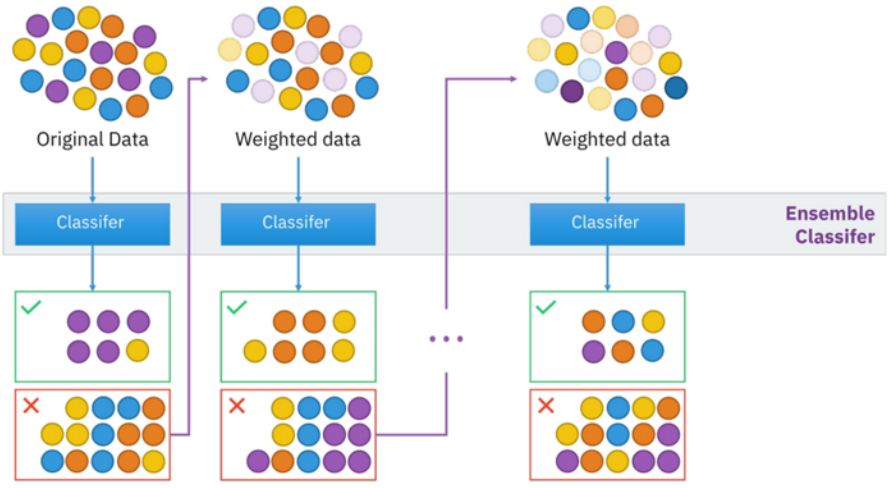


- **Level 1**: Decision Tree - Thuật toán giúp xây dựng 1 cây quyết định dựa trên toàn bộ tập dữ liệu huấn luyện.
  Level 2: Bagging - Thuật toán giúp lấy mẫu ngẫu nhiên để tạo ra các tập dữ liệu con để

hình thành nhiều cây quyết định và kết hợp chúng thông qua giá trị trung bình, các cây quyết định được xây dựng trên cơ sở ngẫn nhiên để đảm bảo tính độc lập (parallel ensemble method – phương pháp tổng hợp song song).
Level 3: Random Forest – Là thuật toán cải tiến của Bagging, bổ sung thêm tính ngẫu nhiên về thuộc tính khi xây dựng các cây quyết định.
Level 4: Boosting – Là thuật toán giúp khắc phục được một số nhược điểm của các thuật toán tiền nhiệm trong đó mỗi cây quyết định mới được tạo ra để sửa lỗi của mô hình trước đó, ta bảo các cây quyết định có ảnh hưởng lẫn nhau theo tính tuần tự (sequential ensemble method – phương pháp tổng hợp tuần tự).
Level 5: AdaBoost, Gradient Boost – Là các thuật toán thông dụng của họ Boosting. AdaBoost tập trung vào việc tối ưu hóa trọng số (weights) của các mẫu được phân loại sai, trong khi Gradient Boost tập trung vào việc tối ưu hóa các phần dư sai biệt (residual error) trong đó mỗi cây quyết định mới được tạo ra để sửa lỗi của mô hình trước đó.
Level 6: XGBoost – Là thuật toán nâng cấp từ Gradient Boost, trong đó quá tình huấn luyện có thể diễn ra song song giúp cải thiện hiệu suất và tốc độ huấn luyện, ngoài ra thuật toán có khả năng tự lược bỏ các thuộc tính không cần thiết, xử lý các giá trị bị khuyết và khắc phục lỗi hợp dữ liệu quá khớp (overfiting) hoặc thiên lệch (bias).

3. **AdaBoost là gì?**

- AdaBoost, viết tắt của Adaptive Boosting, là một thuật toán trong Machine Learning thuộc nhóm Boosting. Nó được phát triển để giải quyết điểm yếu của các thuật toán tiền nhiệm thông quả cải thiện hiệu suất và độ chính xác của mô hình từ các weak learners. Cơ chế hoạt động của AdaBoost là tạo ra các mô hình học yếu (weak learners) theo cách tuần tự (sequential), mô hình sau cải tiến hơn mô hình trước và tập trung vào các điểm dữ liệu khó hoặc dự đoán sai lệch mà mô hình trước đó chưa học được từ đó giúp mô hình được cải tiên tục và trở nên mạnh mẽ. AdaBoost thường được sử dụng cho các bài toán dự đoán, phân loại, và cả hồi quy trong một số trường hợp.



4. **Algorithm 1: AdaBoost (Freund and Schapire 1997)**

- **1. Khởi tạo trọng số**
Trọng số (weight) wi = 1/n, với i = 1, 2, 3, ..., n
2. Xây dựng stump đầu tiên
2.1. Xác định node gốc

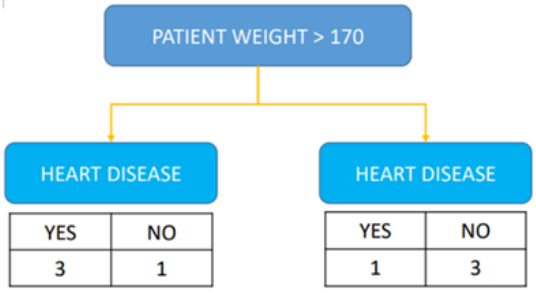
Tính chỉ số Gini cho từng thuộc tính, chọn thuộc tính có giá trị Gini nhỏ nhất

2.2. Tạo giá trị cho stump

2.2.1. Sơ đồ hóa

Kết nối thuộc tính và dữ liệu huấn luyện kèm trọng số đã khởi tạo

Patient Weight	Heart Diease	Sample Weight
205	Yes	1/8
180	Yes	1/8
210	Yes	1/8
167	Yes	1/8
156	No	1/8
125	No	1/8
168	No	1/8
172	No	1/8



2.2.2. Tính lõi giá trị

Tính Total Error (err): Tổng tất cả các trọng số của các phân lớp không chính xác

$$err^{(m)} = \sum_{i=1}^{n} w_i I(c_i \neq T^{(m)}(x_i)) / \sum_{i=1}^{n} w_i \tag{1}$$

Tính Amount of Say (alpha)

Amount of Say: Mức độ đóng góp của một Stump vào kết quả chung cuối cùng. Stump nào có Amount of Say lớn hơn thì mức độ đóng góp trong kết quả chung lớn hơn.

$$\alpha^{(m)} = \log(\frac{1 - err^{(m)}}{err^{(m)}}) \tag{2}$$

2.3. Xây dựng các stump tuần tự kế tiếp

Trọng số cho các mẫu dữ liệu dự đoán sai được sử dụng để tính "Amount of Say" cho từng stump hiện tại. Sử dụng trọng số các mẫu dữ liệu dự đoán sai để xây dựng các stump tiếp theo để khắc phục các dự đoán sai này.

2.3.1. Đặt lại tiêu chuẩn trọng số mới

Tăng trọng số cho các mẫu dữ liệu dự đoán sai, giảm trọng số cho các mẫu dữ liệu dự đoán đúng với label -1, 1; Hoặc; Tăng trọng số cho các mẫu dữ liệu dự đoán sai, giữ nguyên trọng số cho các mẫu dữ liệu dự đoán đúng với label 0, 1

$$w_i \leftarrow w_i.exp(\alpha^{(m)}.I(c_i \neq T^{(m)}(x_i)), i = 1, 2, ..., n \tag{3}$$

2.3.2. Chuẩn hóa trọng số mới

Mục tiêu: Tăng mức độ hiện diện của các mẫu dữ liệu dự đoán sai trong bảng dữ liệu mới.
Phương pháp: Tạo bảng Range theo xác suất cộng dồn, chạy hàm Random được giá trị Range nào thì lấy mẫu dữ liệu tương ứng đưa vào bảng dữ liệu mới

3. Kết quả sau cùng

$$C(x) = argmax_k \sum_{m=1}^{M} \alpha^{(m)}.I(T^{(m)}(x) = k) \tag{4}$$

5. **Algorithm 2: SAMME**

- Áp dụng: Bài toán có từ 2 lớp dự đoán trở lên
  Phương pháp: Tính Multi-class Exponential Loss Function.
  Lưu ý: Triển khai các bước giống Algorithm 1, chỉ khác cách tính Amount of Say

$$\alpha^{(m)} = \log(\frac{1 - err^{(m)}}{err^{(m)}}) + \log(K - 1) \tag{5}$$

6. **Example: Spam Classification**

- Áp dụng: Phân loại email có phải là spam hay không
  Tham khảo phần code bên dưới

http://archive.ics.uci.edu/dataset/94/spambase

```
# Fit model
ab = AIVNAdaBoost()
ab.fit(X_train, y_train, M = 50)

# Predict on test set
y_pred = ab.predict(X_test)
print('The accuracy_score of the model is:', round(accuracy_score(y_test, y_pred), 4))
```

```
The accuracy_score of the model is: 0.9392
```

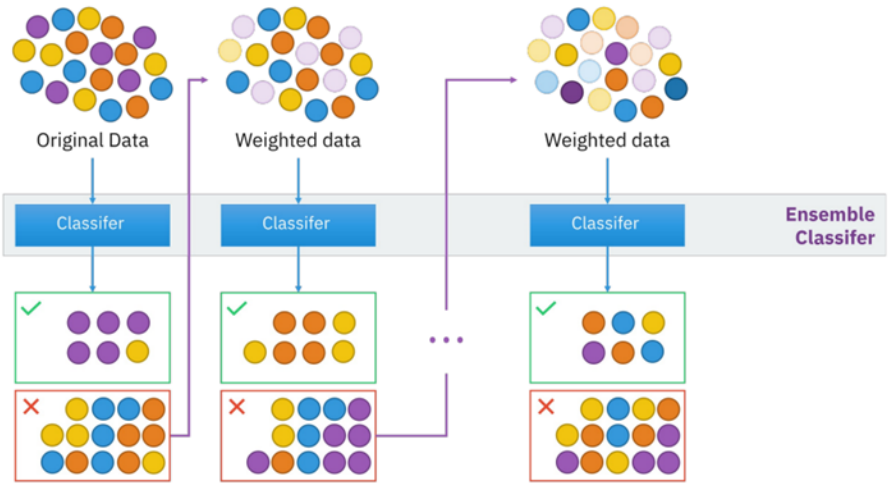
```
from sklearn.ensemble import AdaBoostClassifier

ab_sk = AdaBoostClassifier(n_estimators = 50)
ab_sk.fit(X_train, y_train)
y_pred_sk = ab_sk.predict(X_test)
print('The accuracy_score of the model is:', round(accuracy_score(y_test, y_pred_sk), 4))
```

```
The accuracy_score of the model is: 0.9435
```

7. **Gradient Boost là gì?**

- Gradient Boosting là một thuật toán Machine Learning thuộc lớp các thuật toán Boosting. Ý tưởng chính của Gradient Boosting là xây dựng một mô hình dự đoán mạnh mẽ (strong learner) bằng cách kết hợp nhiều mô hình dự đoán yếu (weak learners). Gradient Boost thường được sử dụng cho các bài toán dự đoán, phân loại, và cả hồi quy.

8. **Algorithm: Gradient Boost**



- **Bước 1: Xây dựng cây quyết định đầu tiên (cây 1 node gốc)**
  Tính toán giá trị trong số trung bình (average of weights)
  Giá trị trọng số trung bình này là giá trị khởi tạo cho cây đầu tiên

$$F_0(x) = argmin_\gamma \sum_{i=1}^{n} L(y_i, \gamma) \tag{6}$$

Bước 2: Khởi tạo M cây với n mẫu dữ liệu
Tính phần dư sai biệt (residual error) và hạn chế lỗi quá khớp (overfitting)

$$r_i m = -[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}]_{F(x)=F_{m-1}(x)} for i = 1, ..., n \tag{7}$$

Tính đầu ra của node lá thỏa hàm loss đạt giá tri cực tiểu

$$\gamma_m = argmin_\gamma \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)) \tag{8}$$

Cập nhật giá trị tìm được và tiếp tục dự đoám

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{9}$$

Bước 3: Kết quả sau cùng
Chọn giá trị tối ưu của hàm cập nhật

9. **Example: Sales Prediction**

- Tham khảo phần code bên dưới

TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9
8.7	48.9	75	7.2
57.5	32.8	23.5	11.8
120.2	19.6	11.6	13.2
8.6	2.1	1	4.8
199.8	2.6	21.2	15.6

```python
import numpy as np
from sklearn.tree import DecisionTreeRegressor

class AIVNGradientBooster:

    def __init__(self, max_depth=8, min_samples_split=5, min_samples_leaf=5, max_features=3, lr=0.1, num_iter=50):
        self.max_depth = max_depth
        self.min_samples_split = min_samples_split
        self.min_samples_leaf = min_samples_leaf
        self.max_features = max_features
        self.lr = lr
        self.num_iter = num_iter
        self.y_mean = 0
```

```python
#READ DATA
data = pd.read_csv("/content/advertising.csv")
data.fillna(0,inplace=True)
#X,y
X = data.iloc[:,:-1].values
y = data.iloc[:,-1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=100)
#scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.fit_transform(X_test)
y_train = np.array(y_train).reshape(X_train.shape[0],1)
y_test = np.array(y_test).reshape(X_test.shape[0],1)
#TRAIN
G = AIVNGradientBooster()
models, losses, pred_0 = G.train(X_train,y_train)
```

```python
sns.set_style('darkgrid')
ax = sns.lineplot(x=range(50),y=losses)
ax.set(xlabel='Epoch',ylabel='Loss',title='Loss vs Epoch')
```