

Tối ưu hóa mô hình XGBoost & Bias-Variance Tradeoff

Mục tiêu bài học

- Hiểu các tham số cần tinh chỉnh trong mô hình XGBoost.
- Nắm được ý nghĩa của từng tham số và ảnh hưởng của chúng đến **Bias** và **Variance**.
- Hiểu khái niệm và vai trò của **Cross-Validation** trong việc chọn tham số tối ưu.

I. Các tham số cần tinh chỉnh trong XGBoost

“Dưới đây là 8 tham số phổ biến ảnh hưởng đến hiệu suất của mô hình:

1. `n_estimators` / `num_boost_round` - Số vòng boosting

- Mô tả: Số cây (trees) được huấn luyện trong mô hình.
- Tăng quá cao → mô hình overfitting (variance cao).
- Quá thấp → mô hình underfitting (bias cao).

Ví dụ:

```
xgb.XGBClassifier(n_estimators=100) # Thử 100 vòng boosting
```

2. `eta` - Learning Rate (tốc độ học)

- Mô tả: Mỗi cây đóng góp bao nhiêu vào kết quả cuối cùng.
- Giá trị thấp → mô hình học chậm → variance cao.
- Giá trị cao → mô hình học nhanh → dễ bị bias cao.

Gợi ý: Dùng `eta` thấp (0.01-0.3) + tăng số vòng.

Ví dụ:

```
xgb.XGBClassifier(eta=0.1) # Tốc độ học vừa phải
```

3. `min_child_weight` - Trọng số tối thiểu cho mỗi leaf

- Mô tả: Quy định ngưỡng tối thiểu để tách cây.
- Ngăn không cho cây học từ các quan sát không đủ "trọng số".

Ví dụ:

```
xgb.XGBClassifier(min_child_weight=5) # Tránh cây quá phức tạp
```

4. `max_depth` - Độ sâu tối đa của cây

- Mô tả: Cây càng sâu càng học kỹ → dễ overfitting.
- Tăng `max_depth` nếu `eta` thấp.

Ví dụ:

```
xgb.XGBClassifier(max_depth=6) # Cây vừa phải
```

5. `gamma` - Ngưỡng để tách (split) cây

- Mô tả: Cây chỉ tách nếu đem lại lợi ích vượt qua gamma.
- Giá trị lớn → khó chia → giảm overfitting.

Ví dụ:

```
xgb.XGBClassifier(gamma=0.2) # Thận trọng khi chia
```

6. `subsample` - Tỷ lệ mẫu lấy ra từ tập dữ liệu

- Mô tả: Ngẫu nhiên lấy 1 phần dữ liệu cho mỗi vòng.
- Giá trị thấp → giúp giảm variance.
- Giá trị cao (≈ 1.0) → học toàn bộ dữ liệu.

Ví dụ:

```
xgb.XGBClassifier(subsample=0.8)
```

7. `colsample_bytree` - Tỷ lệ cột sử dụng cho mỗi cây

- Mô tả: Giảm số lượng đặc trưng (features) dùng để huấn luyện mỗi cây.
- Tác dụng tương tự như `subsample` nhưng trên **features**.

Ví dụ:

```
xgb.XGBClassifier(colsample_bytree=0.7)
```

8. Tổng quan ảnh hưởng đến Bias vs. Variance

Tham số	Giá trị thấp → Ảnh hưởng	Giá trị cao → Ảnh hưởng
eta	Variance ↑	Bias ↑
max_depth	Bias ↑	Variance ↑
min_child_weight	Variance ↑	Bias ↑
gamma	Variance ↑	Bias ↑
subsample	Variance ↑	Bias ↑
colsample_bytree	Variance ↑	Bias ↑

II. Kỹ thuật Cross-Validation - Tối ưu tham số

Mục tiêu:

“ Tìm **bộ tham số tốt nhất** bằng cách **chia nhỏ dữ liệu huấn luyện** thành nhiều phần → thử nghiệm mô hình trên từng phần → trung bình kết quả.

Ý tưởng chính

- Chia dữ liệu huấn luyện thành **k phần (folds)**.
- Dùng (k-1) phần để huấn luyện, 1 phần để kiểm tra.
- Lặp lại k lần → lấy trung bình độ chính xác.

Ưu điểm:

- Đảm bảo mô hình không chỉ tốt trên một tập dữ liệu cụ thể.
- Giúp chọn ra tham số tổng quát hóa tốt.

Ví dụ: Sử dụng `GridSearchCV` để tìm tham số

```
from sklearn.model_selection import GridSearchCV
from xgboost import XGBClassifier

params = {
    'max_depth': [3, 6],
    'learning_rate': [0.01, 0.1],
    'subsample': [0.7, 1],
```

```
}

model = XGBClassifier()
grid = GridSearchCV(estimator=model, param_grid=params, cv=5)
grid.fit(X_train, y_train)

print("Best Params:", grid.best_params_)
```

Ghi nhớ

Vấn đề	Cách giải quyết
Overfitting (Variance cao)	Tăng <code>min_child_weight</code> , tăng <code>gamma</code> , giảm <code>max_depth</code> , <code>subsample</code> , <code>colsample_bytree</code>
Underfitting (Bias cao)	Giảm <code>min_child_weight</code> , giảm <code>gamma</code> , tăng <code>max_depth</code> , <code>n_estimators</code>

Tổng kết

- Bạn nên tinh chỉnh tham số qua **GridSearch** hoặc **RandomizedSearch** kết hợp **Cross-Validation**.
- Luôn cân nhắc **Bias - Variance Tradeoff** khi chọn giá trị.
- XGBoost hỗ trợ song song hóa → xử lý Cross-Validation rất hiệu quả.

Tác giả: **Đỗ Ngọc Tú**
Công Ty Phần Mềm **VHTSoft**