Implementing AdaBoost in Python with Scikit-Learn: A Practical Guide

Learn how to train an AdaBoost classifier using the scikit-learn library for improved classification performance.

Hernández Pérez Erick Fernando

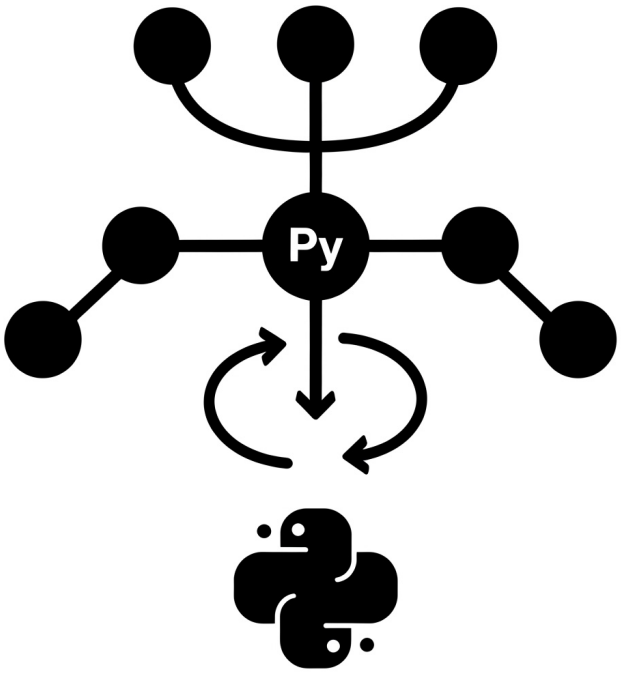

Figure 1

AdaBoost (Adaptive Boosting) is one of the most influential ensemble learning techniques, designed to improve the performance of weak classifiers by combining them into a single, strong model. Originally introduced by Freund and Schapire in 1996, AdaBoost works by training multiple models sequentially, where each subsequent model focuses on the mistakes made by its predecessors. By adjusting the weights of misclassified samples, the algorithm directs more learning effort toward difficult cases, leading to improved accuracy and robustness.

One of the key advantages of AdaBoost is its ability to reduce both bias and variance. Unlike traditional machine learning models that may struggle with underfitting or overfitting, AdaBoost dynamically adapts to the dataset, making it particularly effective for complex classification tasks. Additionally, AdaBoost is relatively resistant to overfitting when used with simple base learners, such as decision stumps (shallow decision trees). This property makes it a powerful choice for many applications, including image recognition, text classification, and medical diagnostics.

In this article, we will focus on implementing AdaBoost in Python using the scikit-learn library. We will walk through the process of training an AdaBoost classifier, evaluating its performance, and understanding how it makes predictions.

AdaBoostClassifier

AdaBoostClassifier

```
class sklearn.ensemble.AdaBoostClassifier(estimator=None, *, n_estimators=50,
↪  learning_rate=1.0, algorithm='deprecated', random_state=None)
```

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. (Scikit-learn Developers, 2024) [5]

Parameters

- **estimator** (*object*, default=None). Defines the base learner for the boosted ensemble. It must support sample weighting and provide the `classes_` and `n_classes_` attributes. If set to `None`, a `DecisionTreeClassifier` with `max_depth=1` is used by default.
  Note: Since version 1.2, ***`base_estimator`*** *has been renamed to* ***`estimator`****.*
- **n_estimators** (*int*, default=50). Specifies the maximum number of weak classifiers in the ensemble. The boosting process stops early if a perfect fit is achieved. Must be within the range $[1, \infty)$.
- **learning_rate** (*float*, default=1.0). Determines the weight of each classifier in every boosting iteration. Higher values increase the impact of individual classifiers. There is a trade-off between `learning_rate` and `n_estimators`. Must be within the range $(0.0, \infty)$.
- **random_state** (*int*, `RandomState` instance, or `None`, default=`None`). Controls the random seed assigned to each estimator at every boosting iteration. Useful for reproducibility when the base estimator supports `random_state`.

Attributes

- **estimator_** (*estimator*). The base estimator from which the ensemble is built.
- **estimators_** (*list of classifiers*). A list of fitted sub-estimators (weak classifiers) in the ensemble.
- **classes_** (*ndarray of shape (n_classes,)*). The class labels associated with the target variable.
- **n_classes_** (*int*). The number of distinct classes in the dataset.
- **estimator_weights_** (*ndarray of floats*). The weights assigned to each estimator in the boosted ensemble.
- **estimator_errors_** (*ndarray of floats*). The classification error for each estimator in the boosted ensemble.
- **feature_importances_** (*ndarray of shape (n_features,)*). The impurity-based feature importances, which indicate the contribution of each feature to the model.
- **n_features_in_** (*int*). The number of features observed during the fitting process.
- **feature_names_in_** (*ndarray of shape (n_features_in_,)*). The names of the features observed during the fitting process. Defined only when the input data `X` contains feature names that are all strings.

AdaBoostClassifier

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

# Create a synthetic classification dataset
X, y = make_classification(n_samples=1000, n_features=4,
                           n_informative=2, n_redundant=0,
                           random_state=0, shuffle=False)
```

```python
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize and fit AdaBoost classifier
clf = AdaBoostClassifier(n_estimators=100, random_state=0)
clf.fit(X_train, y_train)

# Predict on the test set
y_pred = clf.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)

# Output results
print(f"Accuracy: {accuracy:.4f}")
print("Confusion Matrix:")
print(conf_matrix)
print("Classification Report:")
print(class_report)

# Predict on a single sample
sample = [[0, 0, 0, 0]]
sample_prediction = clf.predict(sample)
print(f"Prediction for {sample}: {sample_prediction}")
```

AdaBoostRegressor

AdaBoostRegressor

```
class sklearn.ensemble.AdaBoostRegressor(estimator=None, *, n_estimators=50,
↪  learning_rate=1.0, loss='linear', random_state=None)
```

An AdaBoost regressor is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction. As such, subsequent regressors focus more on difficult cases. (Scikit-learn Developers, 2024) [6]

Parameters

- **estimator** (*object*, default=None). The base estimator from which the boosted ensemble is built. If None, the base estimator is `DecisionTreeRegressor` initialized with `max_depth=3`.
- **n_estimators** (*int*, default=50). The maximum number of estimators at which boosting is terminated. In case of perfect fit, the learning procedure is stopped early. Values must be in the range $[1, \infty)$.
- **learning_rate** (*float*, default=1.0). Weight applied to each regressor at each boosting iteration. A higher learning rate increases the contribution of each regressor. There is a trade-off between `learning_rate` and `n_estimators`. Values must be in the range $(0.0, \infty)$.
- **loss** (*'linear', 'square', 'exponential'*, default='linear'). The loss function to use when updating the weights after each boosting iteration.
- **random_state** (*int, RandomState instance or None*, default=None). Controls the random seed given at each estimator at each boosting iteration. It is only used when estimator exposes a `random_state`. It also controls the bootstrap of the weights used to train the estimator at each boosting iteration. Pass an int for reproducible output across multiple function calls.

Attributes

- **estimator_** (*estimator*). The base estimator from which the ensemble is grown.
- **estimators_** (*list of regressors*): The collection of fitted sub-estimators in the ensemble.
- **estimator_weights_** (*ndarray of floats*). Weights for each estimator in the boosted ensemble.
- **estimator_errors_** (*ndarray of floats*). Regression error for each estimator in the boosted ensemble.
- **feature_importances_** (*ndarray of shape (n_features,)*). The impurity-based feature importances.
- **n_features_in_** (*int*): The number of features seen during the fit process.
- **feature_names_in_** (*ndarray of shape (n_features_in_,)*). Names of features seen during the fit process. Defined only when the input data `X` has feature names that are all strings.

AdaBoostRegressor

```python
from sklearn.ensemble import AdaBoostRegressor
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Create a synthetic regression dataset
X, y = make_regression(n_samples=1000, n_features=4, n_informative=2,
                       noise=0.1, random_state=0, shuffle=False)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Initialize and fit AdaBoost regressor
regr = AdaBoostRegressor(n_estimators=100, random_state=0)
regr.fit(X_train, y_train)

# Predict on the test set
y_pred = regr.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Output results
print(f"Mean Squared Error: {mse:.4f}")
print(f"R-squared: {r2:.4f}")

# Predict on a single sample
sample = [[0, 0, 0, 0]]
sample_prediction = regr.predict(sample)
print(f"Prediction for {sample}: {sample_prediction}")
```

References

[1] Ganaie, M., Hu, M., Malik, A., Tanveer, M. & Suganthan, P. (2022). *Ensemble deep learning: A review.* Engineering Applications of Artificial Intelligence, 115, 105151. https://doi.org/10.1016/j.engappai.2022.105151

[2] Morales, A. (2015). *Uso de características no lineales para identificar llantos de recién nacidos con un conjunto clasificador* (Bachelor's thesis). Universidad de las Américas Puebla. https://catarina.udlap.mx/u_dl_a/tales/documentos/lmt/morales_s_aa/

[3] Rojas, R. (2009). *AdaBoost and the Super Bowl of classifiers: A tutorial introduction to adaptive boosting.*

Computer Science Department, Freie Universität Berlin. https://www.inf.fu-berlin.de/inst/ag-ki/adaboost4.pdf

[4] Rudin, C. (2012). *Boosting (MIT 15.097 course notes)*. Massachusetts Institute of Technology. https://ocw.mit.edu/courses/15-097-prediction-machine-learning-and-statistics-spring-2012/bd1326207712d907fe3418bcc9043b55_MIT15_097S12_lec10.pdf

[5] Scikit-learn Developers. (2024). *AdaBoostClassifier*. Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

[6] Scikit-learn Developers. (2024). *AdaBoostRegressor*. Scikit-learn. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostRegressor.html