17-AdaBoost

October 20, 2024

1 Adaboost

AdaBoost (Adaptive Boosting) is an ensemble learning technique that combines multiple weak classifiers to form a strong classifier. It works by training weak classifiers (often decision stumps or shallow trees) sequentially, where each classifier focuses more on the mistakes made by the previous ones.

The key idea behind AdaBoost is to assign more weight to the misclassified instances from the previous classifier, so that the next classifier can focus on these harder-to-classify examples. By combining the predictions of these weak learners, AdaBoost can create a highly accurate model.

AdaBoost = Adaptive + Boosting

- **Boosting**: Combining weak classifiers to create a strong one.
- **Adaptive**: Adjusting the importance (weights) of misclassified examples.

When to Use AdaBoost? AdaBoost is particularly useful when:

- You want to improve the performance of a weak learner (such as a decision stump).
- You are dealing with binary classification problems, although it can be extended to multi-class classification as well.
- You need an efficient model that can provide high accuracy without too much computational cost.
- Your dataset has moderate noise. However, AdaBoost can be sensitive to outliers, so noisy datasets should be preprocessed carefully.

How Does AdaBoost Work? AdaBoost trains classifiers sequentially, each one focusing more on the mistakes of the previous one. Here's the step-by-step breakdown:

1. Initialize Weights:

- All instances in the training set are given equal weights initially. For N data points, the weight for each data point is 1/N.

2. Train the First Weak Classifier:

A weak classifier (usually a decision stump) is trained on the weighted data. It predicts the class for each example, and the weighted error is calculated.

3. Update Weights:

- Increase the weights of the misclassified examples so that the next classifier pays more attention to these points.
- Decrease the weights of correctly classified examples.

4. **Repeat:**
   - Train a new weak classifier using the updated weights.
   - Adjust the weights based on the errors of the new classifier.
   - This process continues for a specified number of iterations or until a stopping criterion is met.
5. **Combine Weak Learners:**
   - After multiple iterations, the weak classifiers are combined to make the final prediction.
   - Each classifier is given a weight based on its accuracy, with more accurate classifiers contributing more to the final decision.

1.0.1 Who Should Use AdaBoost?

- **Data scientists and machine learning engineers**: Who want to improve the performance of simple models in tasks like binary classification.
- **Beginners and intermediate learners**: AdaBoost is relatively simple to understand, making it suitable for learners who are exploring boosting techniques.
- **Industries that rely on classification tasks**: AdaBoost can be used in fraud detection, text classification, image recognition, and customer churn prediction.

Advantages of AdaBoost:

- **Improves weak learners**: Transforms weak classifiers into a strong one.
- **No parameter tuning**: Unlike many machine learning algorithms, AdaBoost often works well with minimal tuning.
- **Efficient and scalable**: It can handle large datasets efficiently.
- **Feature selection**: AdaBoost implicitly performs feature selection by giving more importance to features that are useful for classification.

Disadvantages of AdaBoost:

- **Sensitive to noisy data**: Outliers can disrupt the performance of AdaBoost because they get higher weights, causing overfitting.
- **Computation cost**: For very large datasets, the sequential training of weak learners can become computationally expensive.
- **Works best with weak learners**: Using a strong learner as the base model can lead to overfitting.

[43]:
```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.datasets import make_classification
from sklearn.ensemble import AdaBoostClassifier
```

```
from sklearn.metrics import accuracy_score, confusion_matrix,␣
 ↪classification_report

from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

[44]:
```
X, y = make_classification(n_samples=2000, n_features=10, n_informative=8,␣
 ↪n_redundant=2, random_state=11)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
 ↪2,random_state=11)
abc = AdaBoostClassifier()
abc.fit(X_train, y_train)
```

```
c:\Users\ikiga\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
```

[44]:
```
AdaBoostClassifier()
```

[45]:
```
y_pred = abc.predict(X_test)
y_pred
```

[45]:
```
array([0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 1, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0,
       1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1,
       0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1,
       1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       0, 1, 0, 0])
```

[46]:
```
print(accuracy_score(y_test, y_pred))
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
0.815
[[171  34]
 [ 40 155]]
              precision    recall  f1-score   support

           0       0.81      0.83      0.82       205
           1       0.82      0.79      0.81       195

    accuracy                           0.81       400
   macro avg       0.82      0.81      0.81       400
weighted avg       0.82      0.81      0.81       400
```

1.1 Logistic Regression

[47]:
```
abclog = AdaBoostClassifier(estimator=LogisticRegression())
abclog.fit(X_train, y_train)
```

```
c:\Users\ikiga\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
```

[47]:
```
AdaBoostClassifier(estimator=LogisticRegression())
```

[48]:
```
y_pred2 = abclog.predict(X_test)
y_pred2
```

[48]:
```
array([0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1,
       1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0])
```

[49]:
```
print(accuracy_score(y_test, y_pred2))
print(confusion_matrix(y_test, y_pred2))
print(classification_report(y_test, y_pred2))
```

```
0.785
[[167  38]
 [ 48 147]]
              precision    recall  f1-score   support

           0       0.78      0.81      0.80       205
           1       0.79      0.75      0.77       195

    accuracy                           0.79       400
   macro avg       0.79      0.78      0.78       400
weighted avg       0.79      0.79      0.78       400
```

1.2 Support Vector Machine

[50]:
```
svc = SVC(kernel='linear', probability=True)

abcsvm = AdaBoostClassifier(estimator=svc, n_estimators=25, learning_rate=0.1)
abcsvm.fit(X_train, y_train)
```

```
c:\Users\ikiga\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
```

[50]:
```
AdaBoostClassifier(estimator=SVC(kernel='linear', probability=True),
                   learning_rate=0.1, n_estimators=25)
```

[51]:
```
y_pred3 = abcsvm.predict(X_test)
y_pred3
```

[51]:
```
array([0, 0, 1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1,
       0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 1, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
       1, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1,
       1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 0, 1, 0,
       1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
```

```
       1, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1,
       1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0])
```

[52]:
```
print(accuracy_score(y_test, y_pred3))
print(confusion_matrix(y_test, y_pred3))
print(classification_report(y_test, y_pred3))
```

```
0.7925
[[168  37]
 [ 46 149]]
              precision    recall  f1-score   support

           0       0.79      0.82      0.80       205
           1       0.80      0.76      0.78       195

    accuracy                           0.79       400
   macro avg       0.79      0.79      0.79       400
weighted avg       0.79      0.79      0.79       400
```

1.3 Hyperparameter Tuning

[54]:
```
param_grid = {
    'n_estimators': [1,5,10,25,50,100,500],
    'learning_rate': [0.00001, 0.0001, 0.001, 0.1, 0.5, 1.0]
}

abc_grid = GridSearchCV(abc, param_grid, cv=3, n_jobs=-1)
abc_grid.fit(X_train, y_train)
```

```
c:\Users\ikiga\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
```

[54]:
```
GridSearchCV(cv=3, estimator=AdaBoostClassifier(), n_jobs=-1,
             param_grid={'learning_rate': [1e-05, 0.0001, 0.001, 0.1, 0.5, 1.0],
                         'n_estimators': [1, 5, 10, 25, 50, 100, 500]})
```

[55]:
```
abc_grid.best_params_
```

[55]:
```
{'learning_rate': 0.1, 'n_estimators': 500}
```

[56]:
```
abc2 = AdaBoostClassifier(learning_rate=0.2, n_estimators=500)
abc2.fit(X_train, y_train)
```

```
c:\Users\ikiga\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\ensemble\_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
```

[56]:
```
AdaBoostClassifier(learning_rate=0.2, n_estimators=500)
```

[58]:
```
y_pred4 = abc2.predict(X_test)
y_pred4
```

[58]:
```
array([0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 1,
       1, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 1, 0,
       0, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 0,
       1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0,
       0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1,
       0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0,
       1, 0, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1,
       1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1,
       1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
       0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       0, 1, 0, 0])
```

[59]:
```
print(accuracy_score(y_test, y_pred4))
print(confusion_matrix(y_test, y_pred4))
print(classification_report(y_test, y_pred4))
```

```
0.8275
[[173  32]
 [ 37 158]]
              precision    recall  f1-score   support

           0       0.82      0.84      0.83       205
           1       0.83      0.81      0.82       195

    accuracy                           0.83       400
   macro avg       0.83      0.83      0.83       400
weighted avg       0.83      0.83      0.83       400
```