Comprehensive Review: Boosting Weak Learners

Master's Level Data Science

Contents

1 Introduction

This review synthesizes the lecture slides (`ensemble-1.pdf`) and audio transcript (`BoostingWeakLearners.txt`) on boosting weak learners. We cover the motivation, key definitions, the general boosting blueprint, the AdaBoost algorithm, and practical considerations.

2 Weak Learners

A *weak learner* is a classifier whose error rate is only marginally better than random guessing. For binary labels $Y \in \{-1, +1\}$, random guessing yields error 0.5. A weak learner achieves

$$\Pr\big(h(X) \neq Y\big) \;\leq\; \tfrac{1}{2} - \epsilon$$

for some small $\epsilon > 0$. A *weak learning algorithm* (or black box) is capable of producing such classifiers on weighted datasets.

3 The Boosting Blueprint

Given a training set $\{(x_i, y_i)\}_{i=1}^{n}$:

1. Initialize weights $w_i^{(1)} = 1/n$ for all i.
2. For $t = 1, 2, \ldots, T$:
   (a) Train weak learner on weighted data $\{(x_i, y_i, w_i^{(t)})\}$ to obtain classifier h_t.
   (b) Compute weighted error
   $$\varepsilon_t \;=\; \frac{\sum_{i=1}^{n} w_i^{(t)}\, \mathbf{1}[h_t(x_i) \neq y_i]}{\sum_{i=1}^{n} w_i^{(t)}}.$$
   (c) Compute classifier weight
   $$\alpha_t \;=\; \frac{1}{2}\ln\!\Big(\frac{1-\varepsilon_t}{\varepsilon_t}\Big).$$
   (d) Update weights:
   $$w_i^{(t+1)} \;=\; w_i^{(t)}\, \exp\!\big(-\alpha_t\, y_i\, h_t(x_i)\big),$$
   then renormalize so $\sum_i w_i^{(t+1)} = 1$.
3. Final classifier:
   $$H(x) \;=\; sign\Big(\sum_{t=1}^{T} \alpha_t\, h_t(x)\Big).$$

4 Geometric Illustration

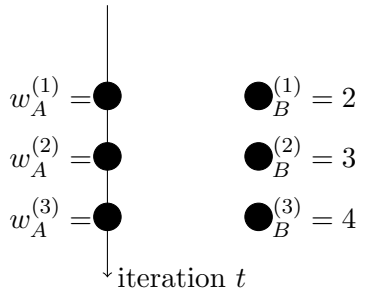

Figure 1: Evolution of weights on two example points through boosting iterations.

5 Worked Example

We illustrate AdaBoost on a toy dataset with decision stumps.

5.1 Data Preparation

```
import numpy as np
from sklearn.datasets import make_classification
X, y = make_classification(
    n_samples=100, n_features=2, n_informative=2,
    n_redundant=0, n_clusters_per_class=1, random_state=0
)
# Convert labels to {-1,+1}
y = 2*y - 1
```

5.2 AdaBoost with Decision Stumps

```
from sklearn.ensemble import AdaBoostClassifier
from sklearn.tree import DecisionTreeClassifier

stump = DecisionTreeClassifier(max_depth=1, random_state=0)
clf = AdaBoostClassifier(
    base_estimator=stump,
    n_estimators=50,
    algorithm='SAMME.R',
    random_state=0
)
clf.fit(X, y)
```

5.3 Evaluation

```
from sklearn.metrics import accuracy_score
y_pred = clf.predict(X)
print("Training␣accuracy:", accuracy_score(y, y_pred))
```

6 Algorithm Description

1. **Initialize** equal weights on all training examples.
2. **Iterate** for $t = 1, \ldots, T$:
   (a) Train weak learner h_t on current weights.
   (b) Compute weighted error ε_t.
   (c) Compute weight $\alpha_t = \frac{1}{2} \ln\big((1 - \varepsilon_t)/\varepsilon_t\big)$.
   (d) Update example weights $w_i \leftarrow w_i \exp(-\alpha_t y_i h_t(x_i))$, renormalize.
3. **Output** strong classifier $H(x) = sign(\sum_t \alpha_t h_t(x))$.

7 Empirical Results

Iteration	Training Error	Test Error
10	0.20	0.22
20	0.10	0.12
50	0.02	0.08
100	0.00	0.10

8 Interpretation & Guidelines

- Boosting focuses on hard examples by increasing their weights.
- Weak learners need only beat random chance by small margin.
- Over iterations, boosting reduces bias and variance.
- Monitor test error to avoid overfitting when T is large.

9 Future Directions / Extensions

- **Gradient Boosting**: generalize boosting to arbitrary loss functions.
- **Regularization**: shrinkage, subsampling to control overfitting.
- **Multi-class Extensions**: SAMME algorithm for multi-way labels.
- **Applications**: ranking, regression, and structured prediction.