CSC 461: Machine Learning

Fall 2024

Boosting

Prof. Marco Alvarez, Computer Science
University of Rhode Island

Introduction

- Boosting
  - powerful ensemble method in machine learning
  - **goal**: combine **weak learners** to create a strong predictor
  - **key idea**: sequentially build models, focusing on misclassified instances
  - backed by solid theoretical foundations
- Contrast with Bagging:
  - bagging → reduce variance by parallel ensembling
  - boosting → reduce bias by sequential ensembling

Weak learners

- Definition
  - classifiers performing slightly better than random guessing
  - e.g. if "buy now" in the message then predict "spam"
- Examples:
  - decision stumps (single-node decision trees), shallow decision trees, simple linear models
- Characteristics:
  - easy to construct, fast to train, often interpretable

The boosting principle

- Combining many "weak" rules can create a "strong" rule
- Mathematical formulation:

$$H(\mathbf{x}) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$$

  - $H(\mathbf{x})$ is the strong classifier
  - $h_t(\mathbf{x})$ are weak classifiers
  - α_t are weights assigned to each weak classifier
- Given sufficient data, a boosting algorithm can **provably** construct a single predictor with very high accuracy

General approach

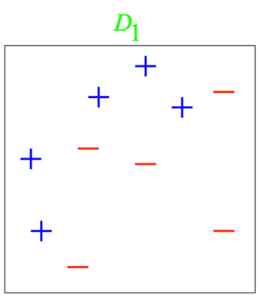
- Define a weight for each training instance
- For a number of iterations T
  - train a weak learner h_t with the weighted instances
  - calculate error ϵ_t and learner weight α_t
  - recalculate the training instance weights
    - weights of incorrect predictions are increased
    - weights of correct predictions are decreased
- Combine weak learners h_t into final model H

concentrate on "hardest" examples

Adaboost

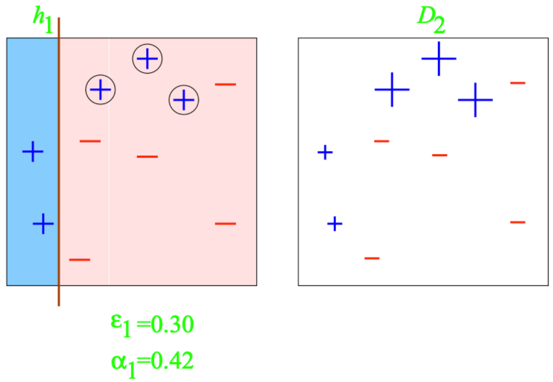
Adaboost

- Developed by Freund and Schapire (1995)
- First practical boosting algorithm
- Adaptively adjusts to the errors of weak learners

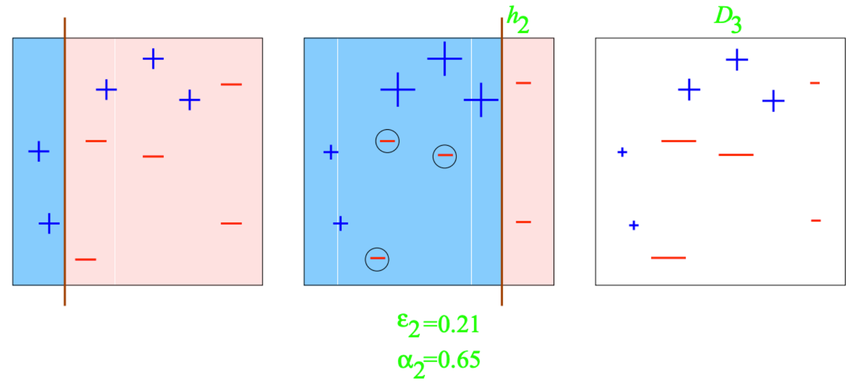


weak classifiers = vertical or horizontal half-planes

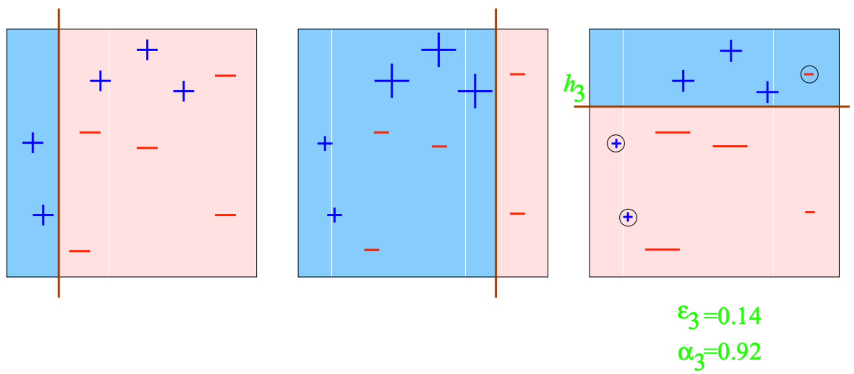
credit: Theory and applications of boosting, Rob Schapire



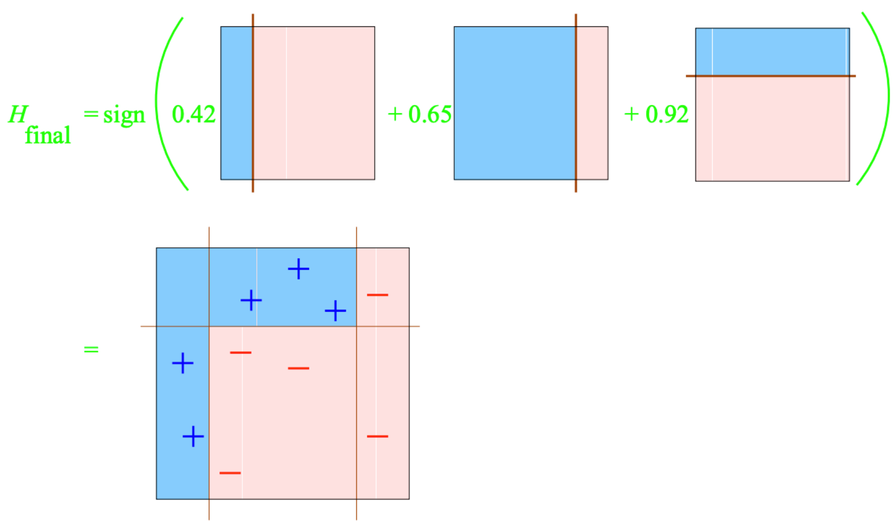
credit: Theory and applications of boosting, Rob Schapire



credit: Theory and applications of boosting, Rob Schapire



credit: Theory and applications of boosting, Rob Schapire



credit: Theory and applications of boosting, Rob Schapire

Algorithm

- given training set $(x_1, y_1), \dots, (x_m, y_m)$
- $y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$
- for $t = 1, \dots, T$:
  - construct distribution D_t on $\{1, \dots, m\}$
  - find weak classifier ("rule of thumb")

    $$h_t : X \to \{-1, +1\}$$

    with small error ϵ_t on D_t:

    $$\epsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$$
- output final classifier H_{final}

Algorithm

- constructing D_t:
  - $D_1(i) = 1/m$
  - given D_t and h_t:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & \text{if } y_i = h_t(x_i) \\ e^{\alpha_t} & \text{if } y_i \neq h_t(x_i) \end{cases}$$

$$= \frac{D_t(i)}{Z_t} \exp(-\alpha_t \, y_i \, h_t(x_i))$$

where Z_t = normalization constant

$$\alpha_t = \tfrac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) > 0$$

- final classifier:
  - $H_{\text{final}}(x) = \text{sign}\left(\sum_t \alpha_t h_t(x)\right)$

Adaboost

$\textsc{AdaBoost}(S = ((x_1, y_1), \dots, (x_m, y_m)))$

for $i \leftarrow 1$ **to** m **do**

  $D_1(i) \leftarrow \frac{1}{m}$

for $t \leftarrow 1$ **to** T **do**

  $h_t \leftarrow$ base classifier in H with small error $\epsilon_t = \Pr_{D_t}[h_t(x_i) \neq y_i]$

  $\alpha_t \leftarrow \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$

  $Z_t \leftarrow 2[\epsilon_t(1 - \epsilon_t)]^{\frac{1}{2}}$ ▷ normalization factor

  for $i \leftarrow 1$ **to** m **do**

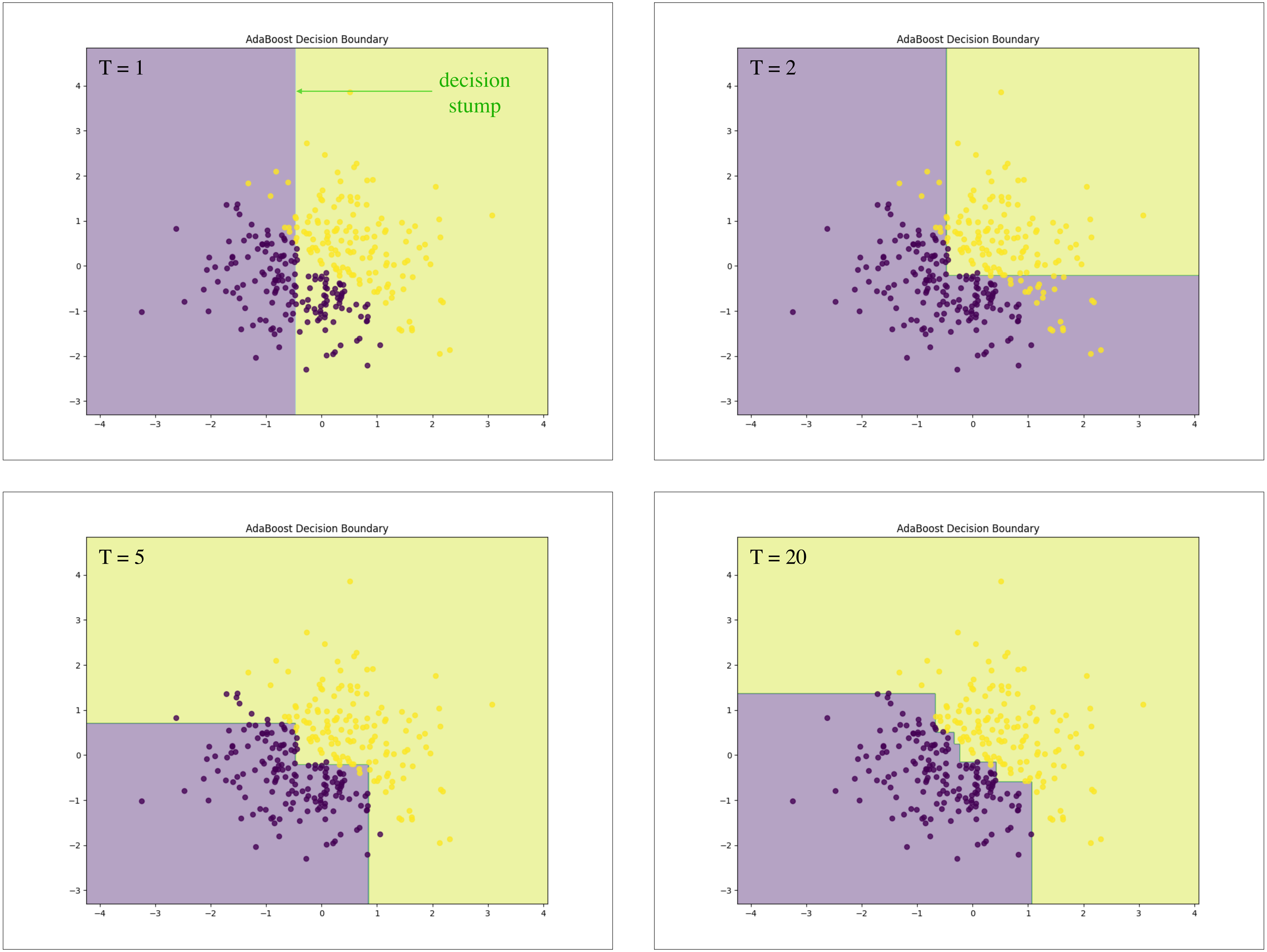
    $D_{t+1}(i) \leftarrow \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

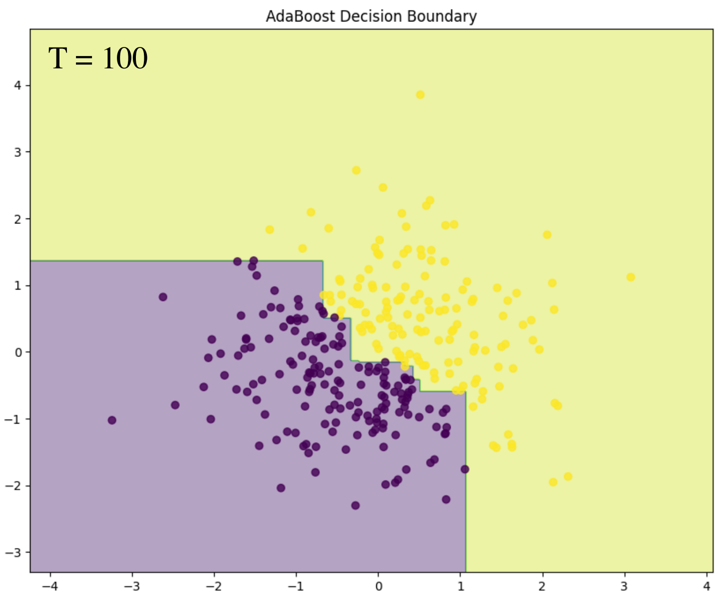
$f \leftarrow \sum_{t=1}^{T} \alpha_t h_t$

return $h = \text{sgn}(f)$

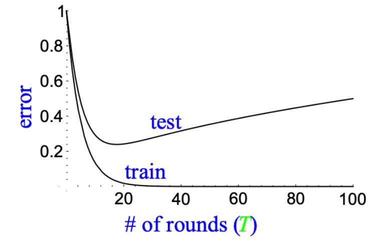
Show me the code

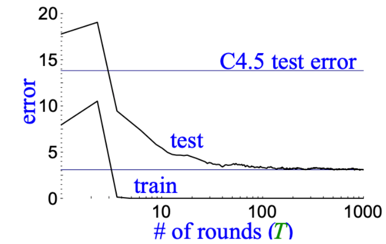
```python
# generate toy dataset
np.random.seed(42)
X = np.random.randn(300, 2)
y = (X[:, 0] + X[:, 1] > 0).astype(int)

# train AdaBoost
clf = AdaBoostClassifier(
    DecisionTreeClassifier(max_depth=1),
    n_estimators=100
)
clf.fit(X, y)
```


AdaBoost Decision Boundary
T = 1
decision stump
AdaBoost Decision Boundary
T = 2
AdaBoost Decision Boundary
T = 5
AdaBoost Decision Boundary
T = 20



Generalization





	# rounds		
	5	100	1000
train error	0.0	0.0	0.0
test error	8.4	3.3	3.1

A first guess on the behavior of the test error is that it would increase as the final classifier becomes “too complex”

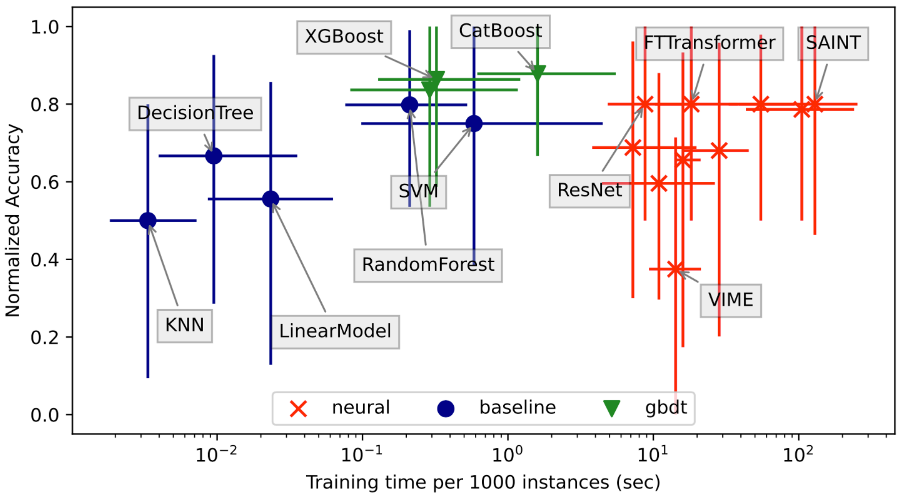
Typical run on boosting C4.5 on the “letter” dataset (surprisingly resistant to overfitting, test error continues to improve even after training error is zero)

Final remarks

- Practical advantages over previous attempts
  - fast, simple, easy to code
  - no hyperparameters to tune (except T)
  - flexible — can use any weak learner
- Caveats
  - performance depends on data and weak learners
- Can fail if …
  - strong base learners => overfitting
  - too weak base learners => underfitting
  - susceptible to noise

Modern variants and applications

- Multi-class AdaBoost
- Gradient Boosting
  - XGBoost, LightGBM, CatBoost
  - state-of-the-art performance in many ML competitions



Median runtime vs. median normalized accuracy for each algorithm, over 104 datasets. The bars span the 20th to 80th percentile over all datasets.

When Do Neural Nets Outperform Boosted Trees on Tabular Data?, NeurIPS 2023