Boosting

CSC 461: Machine Learning

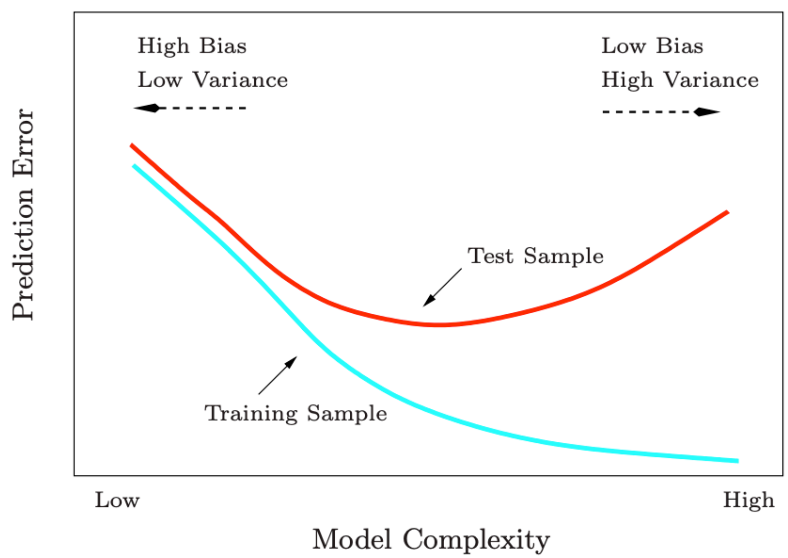
Fall 2022

Prof. Marco Alvarez
University of Rhode Island

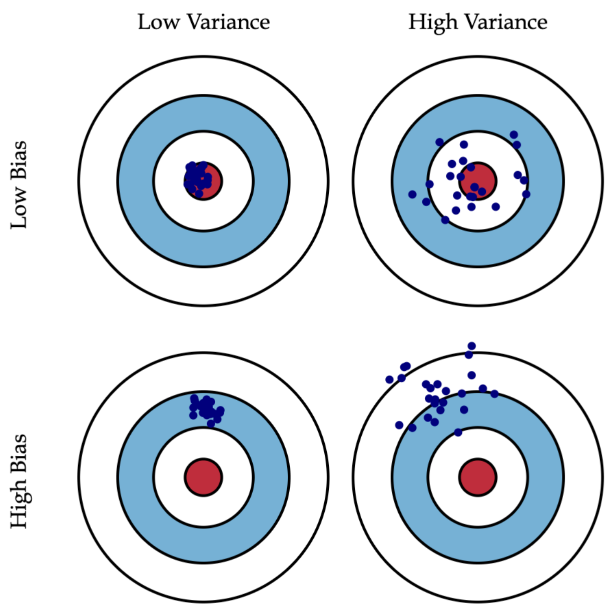
Bias-Variance decomposition

- Expected loss
  - ✓ **bias**: how wrong the expected prediction is
  - ✓ **variance**: the amount of variability in the predictions
  - ✓ **Bayes error**: the inherent unpredictability of the targets (e.g. noise)

$$\mathbb{E}[(y-t)^2] = \underbrace{(y^* - \mathbb{E}[y])^2}_{\text{bias}} + \underbrace{\mathrm{Var}(y)}_{\text{variance}} + \underbrace{\mathrm{Var}(t)}_{\text{Bayes error}}$$



The Elements of Statistical Learning, Hastie, Tibshirani, Friedman, 2nd Ed.



http://scott.fortmann-roe.com/docs/BiasVariance.html

Context

- From bagging …
  - ✓ reduce **variance** by ensembling hypotheses in parallel (ensemble)
- Boosting …
  - ✓ reduce **bias** by ensembling (weak) hypotheses sequentially

Weak learners

- Simple rules (learners) may give reasonable performance
  - ✓ perform at least better than chance (e.g. 51% accuracy)
  - ✓ e.g. if "bank account" then "spam"
- Ideas
  - ✓ create an ensemble of weak learners — highly accurate predictor (e.g. 99% accuracy)
  - ✓ each learner focuses on different examples
  - ✓ each learner contributes differently to the final model

Weak learners (examples)

- Decision stumps
  - ✓ tree with a single node
- Very shallow trees
- Linear models

Set up

$$\mathscr{D} = \{(\mathbf{x}^{(1)}, y^{(1)}), \ldots, (\mathbf{x}^{(n)}, y^{(n)})\}$$

$$\mathbf{x}^{(i)} \in \mathbb{R}^d, y^{(i)} \in \{-1, +1\}$$

Assumptions

- Weak learners can consistently find rules better than random
  - ✓ binary classification settings
- Given sufficient data, a boosting algorithm can **provably** construct a model with high accuracy
- Many algorithms can directly incorporate **weights** into the loss
  - ✓ alternatively, can subsample the data proportional to weights

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
  - ✓ recalculate the weights
    - ✓ weights of incorrect predictions are increased
    - ✓ weights of correct predictions are decreased
- Combine weak learners into final model
  - ✓ each model has a coefficient

Formal description of Boosting

given training set $(x_1, y_1), \ldots, (x_m, y_m)$

$y_i \in \{-1, +1\}$ correct label of instance $x_i \in X$

for $t = 1, \ldots, T$:

- construct distribution D_t on $\{1, \ldots, m\}$
- 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}

From: Boosting: Foundations and Algorithms, Rob Schapire

A decision-theoretic generalization of on-line learning and an application to boosting*

Yoav Freund Robert E. Schapire

AT&T Labs
180 Park Avenue
Florham Park, NJ 07932
{yoav, schapire}@research.att.com

December 19, 1996

Abstract

In the first part of the paper we consider the problem of dynamically apportioning resources among a set of options in a worst-case on-line framework. The model we study can be interpreted as a broad, abstract extension of the well-studied on-line prediction model to a general decision-theoretic setting. We show that the multiplicative weight-update rule of Littlestone and Warmuth [20] can be adapted to this model yielding bounds that are slightly weaker in some cases, but applicable to a considerably more general class of learning problems. We show how the resulting learning algorithm can be applied to a variety of problems, including gambling, multiple-outcome prediction, repeated games and prediction of points in $\mathbb{R}^n$. In the second part of the paper we apply the multiplicative weight-update technique to derive a new boosting algorithm. This boosting algorithm does not require any prior knowledge about the performance of the weak learning algorithm. We also study generalizations of the new boosting algorithm to the problem of learning functions whose range, rather than being binary, is an arbitrary finite set or a bounded segment of the real line.

AdaBoost

$\text{AdaBoost}(S = ((x_1, y_1), \ldots, (x_m, y_m)))$

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

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

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

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

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

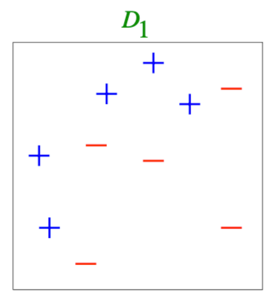
$\quad Z_t \leftarrow 2[\epsilon_t(1-\epsilon_t)]^{\frac{1}{2}}$ $\quad \triangleright$ normalization factor

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

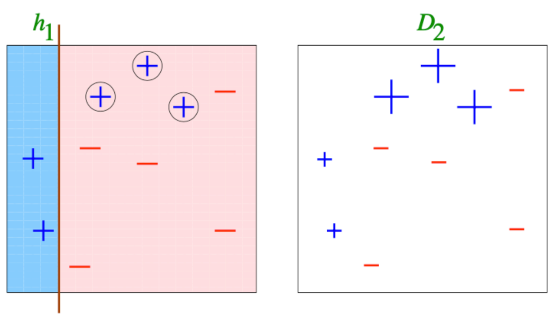
$\qquad D_{t+1}(i) \leftarrow \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t}$

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

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



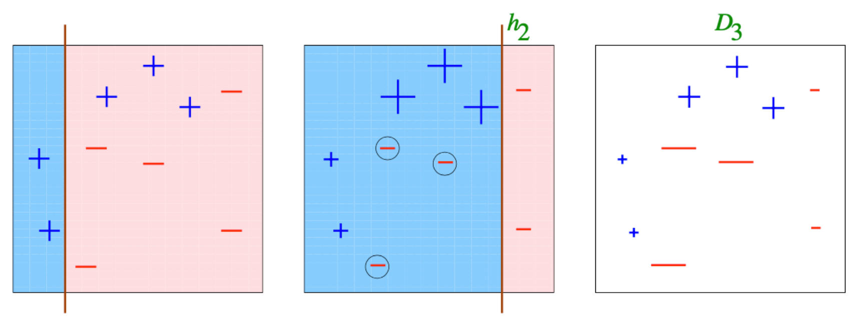
From: Boosting: Foundations and Algorithms, Rob Schapire



ε_1=0.30

α_1=0.42

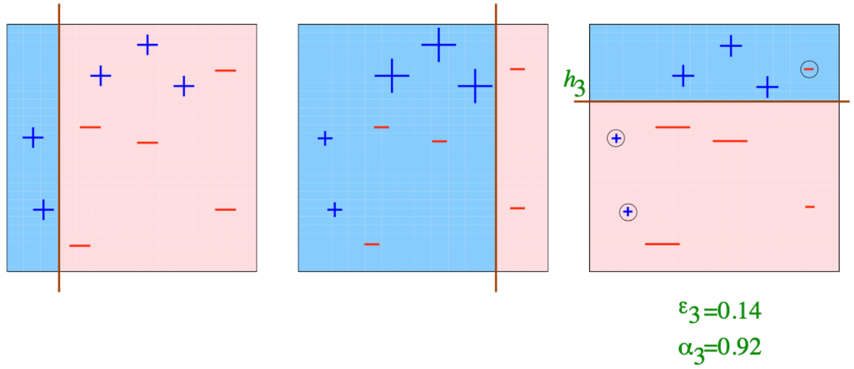
From: Boosting: Foundations and Algorithms, Rob Schapire

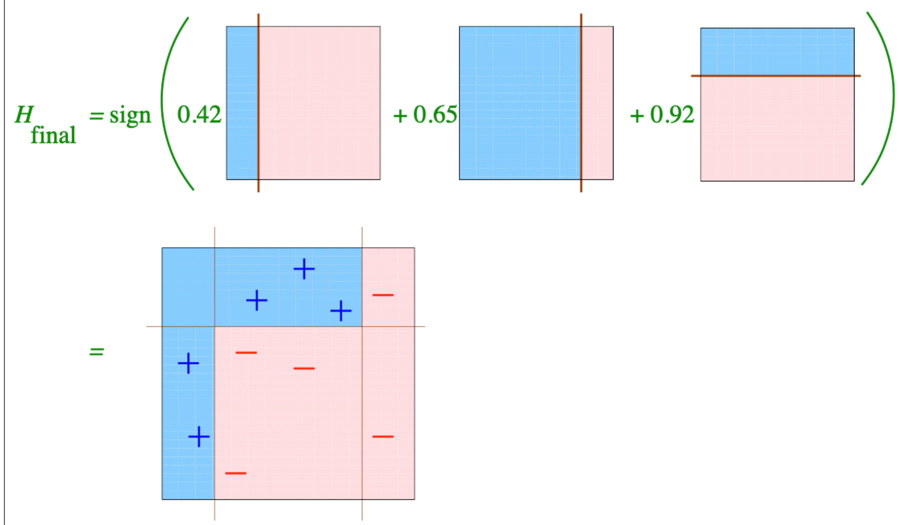


ε_2=0.21

α_2=0.65

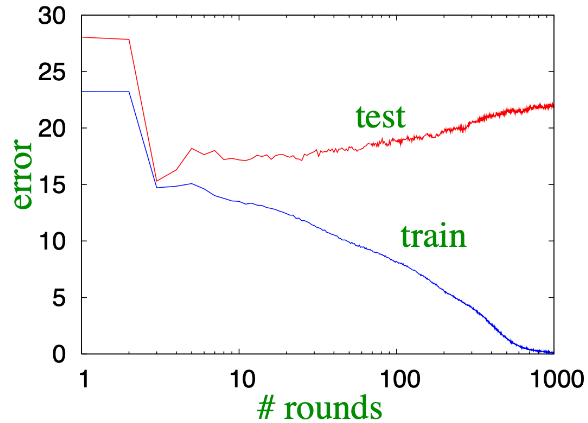
From: Boosting: Foundations and Algorithms, Rob Schapire



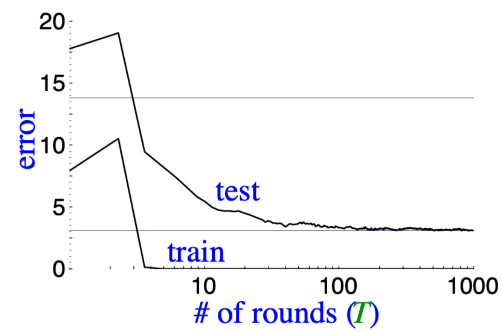


Overfitting

- It may happen … often doesn't



Typical run



(boosting C4.5 on "letter" dataset)

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

Adaboost

- Practical advantages over previous attempts
  - ✓ fast, simple, easy to code
  - ✓ no hyperparameters to tune (except T)
  - ✓ 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