Machine Learning (CE 40717) Fall 2024

Ali Sharifi-Zarchi

CE Department Sharif University of Technology

September 23, 2024





- 1 Introduction
- 2 Bagging
- 3 Random Forest
- 4 Boosting
- 6 AdaBoost
- **6** Comparison
- References



Introduction

- Condorcet's jury theorem Ensemble learning Ensemble methods
- 2 Bagging
- 3 Random Forest
- 4 Boosting
- 6 AdaBoost
- **6** Comparison



1 Introduction

Introduction

Condorcet's jury theorem

Ensemble learning
Ensemble methods

- 2 Bagging
- 3 Random Fores
- 4 Boosting
- 6 AdaBoost
- **6** Comparison

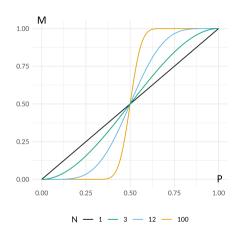




Condorcet's jury theorem

Introduction

- N voters wish to reach a decision by majority vote.
- Each voter has an independent probability
 p of voting for the correct decision.
- Majority votes for the correct decision with probability *M*.
- If p > 0.5 and $N \to \infty$, then $M \to 1$
 - How?



Adopted from Wikipedia



September 23, 2024

1 Introduction

Introduction

- Condorcet's jury theorem
- Ensemble learning

Ensemble method

- 2 Bagging
- 3 Random Fores
- 4 Boosting
- 5 AdaBoost
- **6** Comparison



6 / 70

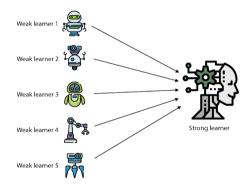
Strong vs. weak Learners

- **Strong learner:** we seek to produce one classifier for which the classification error can be made arbitrarily small.
 - So far we were looking for such methods.
- **Weak learner:** a classifier which is just better than random guessing (for now this will be our only expectation).



Basic idea

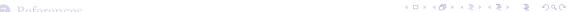
- Certain weak learners do well in modeling one aspect of the data, while others do well in modeling another.
- Learn several simple models and combine their outputs to produce the final decision.
- A composite prediction where the final accuracy is better than the accuracy of individual models.



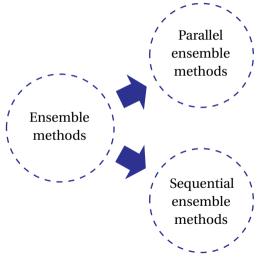
Adopted from [4]

Introduction

- Condorcet's jury theorem Ensemble learning
- Ensemble methods
- 2 Bagging
- 3 Random Fores
- 4 Boosting
- 6 AdaBoost
- **6** Comparison



Ensemble Methods



- Weak learners are generated in **parallel**.
 - Basic motivation is to use independence between the learners.

- Weak learners are generated **consecutively**.
- Basic motivation is to use dependence between the base learners.



What we talk about

- Weak or simple learners
 - Low variance: they don't usually overfit
 - **High bias**: they can't learn complex functions
- Bagging (parallel): To decrease the variance
 - Random Forest
- **Boosting** (sequential): To decrease the bias (enhance their capabilities)
 - AdaBoost



- Introduction
- 2 Bagging
 Basic idea & algorithm

Decision tree (quick review)

- 3 Random Fores
- 4 Boosting
- 6 AdaBoost
- 6 Comparisor
- 7 References



- 2 Bagging Basic idea & algorithm Decision tree (quick review)

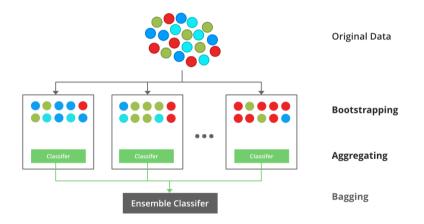


Basic idea

- Bagging = Bootstrap aggregating
- It uses bootstrap resampling to generate different training datasets from the original training dataset.
 - · Samples training data uniformly at random with replacement.
- On the training datasets, it trains different weak learners.
- During testing, it **aggregates** the weak learners by uniform averaging or majority voting.
 - Works best with unstable models (high variance models). Why?



Basic idea, Cont.



Adopted from GeeksForGeeks



15 / 70

Algorithm

Algorithm 1 Bagging

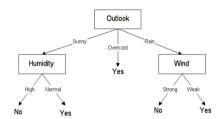
- 1: **Input:** M (required ensemble size), $D = \{(\mathbf{x}^{(1)}, \mathbf{v}^{(1)}), \dots, (\mathbf{x}^{(N)}, \mathbf{v}^{(N)})\}$ (training set)
- 2: **for** t = 1 to M **do**
- Build a dataset D_t by sampling N items randomly with replacement from D 3: ▶ Bootstrap resampling: like rolling N-face dice N times
- Train a model h_t using D_t and add it to the ensemble 4:
- 5: end for
- 6: $H(x) = \operatorname{sign}\left(\sum_{t=1}^{M} h_t(x)\right)$
 - > Aggregate models by voting for classification or by averaging for regression

- 1 Introduction
- 2 Bagging Basic idea & algorithm Decision tree (quick review)
- 3 Random Fores
- 4 Boosting
- 6 AdaBoost
- 6 Comparison
- 7 References



Structure

- **Terminal nodes** (leaves) represent target variable.
- Each internal node denotes a test on an attribute.



Outlook	Temperature	Humidity	Wind	Played football(yes/no
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Adopted from Medium



Learning

- Learning an optimal decision tree is NP-Complete.
 - Instead, we use a **greedy search** based on a heuristic.
 - We can't guarantee to return the globally-optimal decision tree.
- The most common strategy for DT learning is a greedy top-down approach.
- Tree is constructed by splitting samples into subsets based on an **attribute value test** in a recursive manner.

Algorithm

Algorithm 2 Constructing DT

```
1: procedure FINDTREE(S, A)
                                                                  \triangleright Input: S (samples), A (attributes)
        if A is empty or all labels in S are the same then
            status ← leaf
 3:
            class \leftarrow most common class in S
 4:
        else
 5:
            status ← internal
 6:
            a \leftarrow \text{bestAttribute}(S, A)
                                                                              ➤ The attribute value test
            LeftNode \leftarrow FindTree(S(a = 1), A \setminus \{a\})
 8:
 9:
            RightNode \leftarrow FindTree(S(a = 0), A \setminus \{a\})
        end if
10:
11: end procedure
```

Which attribute is the best?

• **Entropy** measures the uncertainty in a specific distribution.

$$H(X) = -\sum_{x_i \in x} P(x_i) \log P(x_i)$$

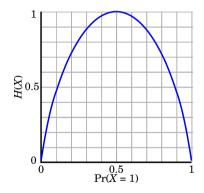
• Information Gain (IG)

$$Gain(S, A) = H_S(Y) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} H_{S_v}(Y)$$

A: variable used to split samples

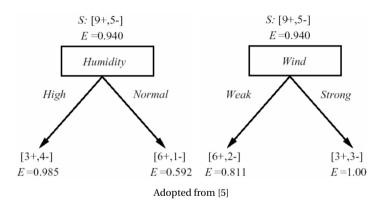
Y: target variable

S: samples



Adopted from Wikipedia

Example



Gain(S, Humidity) = 0.940 - (7/14)0.985 - (7/14)0.592 = 0.151

Gain(S, Humidity) = 0.940 - (8/14)0.811 - (6/14)1.0 = 0.48



- Introduction
- 2 Bagging
- 3 Random Forest
- 4 Boosting
- 6 AdaBoost
- **6** Comparison
- References

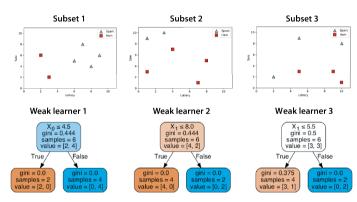


23 / 70

Bagging on decision trees?

Why decision trees?

- Interpretable
- Robust to outliers
- Low bias
- High variance



Adopted from [4]

Perfect candidates

- Why are **DTs** perfect candidates for ensembles?
 - Consider averaging many (nearly) **unbiased** tree estimators.
 - Bias remains similar, but variance is reduced.
- Remember Bagging?
 - Train many trees on bootstrapped data, then average the outputs.



Algorithm

Algorithm 3 Random Forest

- 1: **Input:** *T* (number of trees), *m* (number of variables used to split each node)
- 2: **for** t = 1 to T **do**
- 3: Draw a bootstrap dataset
- 4: Learn a tree on this dataset
 - ightharpoonup Select $m{m}$ features randomly out of $m{d}$ features as candidates before splitting
- 5: end for
- 6: Output:

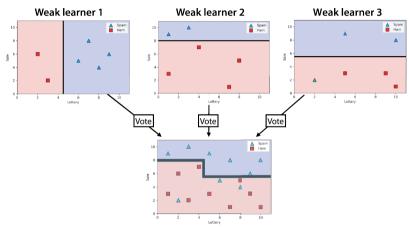
 $ightharpoonup Usually: m \le \sqrt{d}$

- 7: Regression: average of the outputs
- 8: Classification: majority voting



on Bagging **Random Forest** Boosting AdaBoost Comparison References

Example



Strong learner (random forest)

Adopted from [4]



- 1 Introduction
- 2 Bagging
- 3 Random Fores
- 4 Boosting

 Motivation & basic idea

 Algorithm
- 6 AdaBoos
- **6** Comparison
- References



28 / 70

- Introduction
- 2 Bagging
- 3 Random Fores
- 4 Boosting Motivation & basic idea Algorithm
- **6** AdaBoos
- 6 Comparisor
- 7 References



Problems with bagging

- Bagging created a diversity of **weak learners** by creating random datasets.
 - Examples: Decision stumps (shallow decision trees), Logistic regression, ...
- Did we have full control over the usefulness of the weak learners?
 - The **diversity** or **complementarity** of the weak learners is not controlled in any way, it is left to chance and to the instability of the models.



Basic idea

- We would expect a better performance if the weak learners also complemented each other.
 - They would have "expertise" on different subsets of the dataset.
 - So they would work better on different subsets.
- The basic idea of boosting is to generate a **series** of weak learners which complement each other.
 - For this, we will force each learner to focus on the mistakes of the previous learner.



Basic idea, Cont.



Adopted from GeeksForGeeks



- Introduction
- 2 Bagging
- 3 Random Fores
- 4 Boosting
 Motivation & basic idea
 Algorithm
- 6 AdaBoos
- 6 Comparison
- 7 References



33 / 70

Algorithm

- Try to combine many simple weak classifiers (in sequence) to find a single strong classifier.
 - Each component is a simple binary ±1 classifier
 - Voted combinations of component classifiers

$$H_m(\mathbf{x}) = \alpha_1 h(\mathbf{x}; \boldsymbol{\theta}_1) + \dots + \alpha_m h(\mathbf{x}; \boldsymbol{\theta}_m)$$

• To simplify notations: $h(x; \theta_i) = h_i(x)$

$$H_m(\mathbf{x}) = \alpha_1 h_1(\mathbf{x}) + \dots + \alpha_m h_m(\mathbf{x})$$

• **Prediction**: $\hat{y} = \text{sign}(H_m(x))$

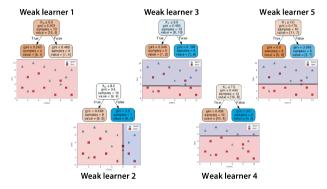


Candidate for $h_i(x)$

Decision stumps

 Each classifier is based on a single feature of x (e.g., x_k):

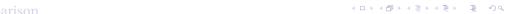
$$h(\mathbf{x}; \boldsymbol{\theta}) = \operatorname{sign}(w_1 \mathbf{x}_k - w_0)$$
$$\boldsymbol{\theta} = \{k, w_1, w_0\}$$



Adopted from [4]

35 / 70

- Introduction
- 2 Bagging
- 3 Random Fores
- 4 Boosting
- **5** AdaBoost
 - Basic idea & example
 - Algorithm
 - Loss function
 - Summary & example
 - Properties



- Introduction
- 2 Bagging
- 3 Random Forest
- 4 Boosting
- 6 AdaBoost Basic idea & example
 - Algorithm
 Loss function
 Summary & example



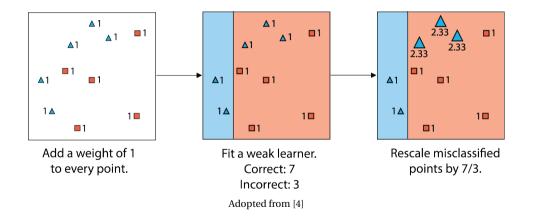


Basic idea

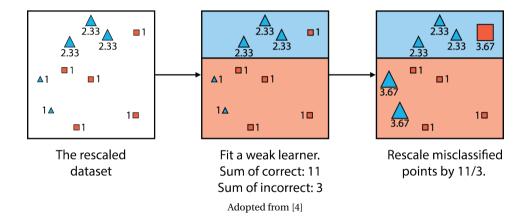
- Sequential production of classifiers
 - Iteratively add the classifier whose addition will be most helpful.
- Represent the important of each sample by assigning weights to them.
 - Correct classification \implies smaller weights
 - Misclassified samples ⇒ larger weights
- Each classifier is **dependent** on the previous ones.
 - Focuses on the **previous ones' error**.



Example

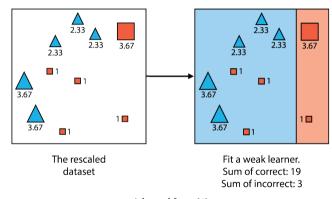


Example, Cont.





Example, Cont.

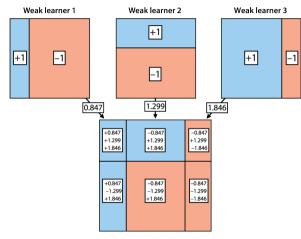


Adopted from [4]



n Bagging Random Forest Boosting **AdaBoost** Comparison References

Example, Cont.

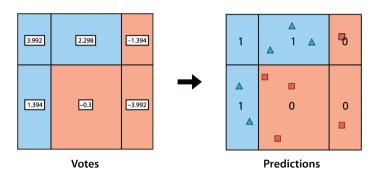


Strong learner

Adopted from [4]



Example, Cont.



Adopted from [4]

- Introduction
- 2 Bagging
- 3 Random Fores
- 4 Boosting
- **5** AdaBoost

Basic idea & example

Algorithm

Loss function
Summary & example
Properties



Algorithm

Algorithm 4 AdaBoost

1: Initialize data weight $w_1^{(i)} = \frac{1}{N}$ for all N samples

2: **for**
$$m = 1$$
 to M **do**

3:
$$J_m = \sum_{i=1}^N w_m^{(i)} \times I(y^{(i)} \neq h_m(\mathbf{x}^{(i)}))$$

 \triangleright Find $h_m(\mathbf{x})$ by minimizing the weighted error

3:
$$J_{m} = \sum_{i=1}^{N} w_{m}^{(i)} \times I(y^{(i)} \neq h_{m}(\mathbf{x}^{(i)}))$$
4:
$$\epsilon_{m} = \frac{\sum_{i=1}^{N} w_{m}^{(i)} \times I(y^{(i)} \neq h_{m}(\mathbf{x}^{(i)}))}{\sum_{i=1}^{N} w_{m}^{(i)}}$$

 \triangleright Find the weighted error of $h_m(x)$

5:
$$\alpha_{m} = \ln \left(\frac{1 - \epsilon_{m}}{\epsilon_{m}} \right)$$
6:
$$w_{m+1}^{(i)} = w_{m}^{(i)} e^{\alpha_{m} I(y^{(i)} \neq h_{m}(\mathbf{x}^{(i)}))}$$

> Assign votes based on the error

6:
$$w_{m+1}^{(i)} = w_m^{(i)} e^{\alpha_m I(y^{(i)} \neq h_m(\mathbf{x}^{(i)}))}$$

> Update normalized weights

7: end for

8: Combined classifier: $\hat{y} = \text{sign}(H_M(\mathbf{x})), H_M(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})$

Notations & conditions

- $w_m^{(i)}$: weighting coefficient of data point *i* in iteration *m*
- α_m : weighting coefficient of *m*-th base classifier in the final ensemble
 - ϵ_m : weighted error rate of *m*-th classifier

- Only when $h_m(x)$ with $\epsilon_m < 0.5$ (better than chance) is found, boosting continues.
 - Condorcet's jury theorem?

- 1 Introduction
- 2 Bagging
- 3 Random Fores
- 4 Boosting
- **5** AdaBoost

Basic idea & example

Algorithn

Loss function

Summary & example

Comporison



Loss function

- We need a loss function for the combination
 - To Determine the new component, $h(x; \theta)$
 - And how many votes it should receive, α

$$H_m(\mathbf{x}) = \alpha_1 h_1(\mathbf{x}) + \dots + \alpha_m h_m(\mathbf{x})$$

- Many options for the loss function
 - AdaBoost is equivalent to using the following **exponential loss**

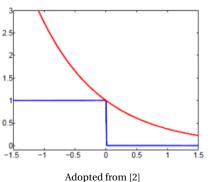
$$\operatorname{Loss}(y,\hat{y}) = e^{-y \times H_m(x)}$$

$$\hat{y} = \text{sign}(H_m(\mathbf{x}))$$



Why the exponential loss?

- Differentiable approximation (bound) of the 0/1 loss
 - Easy to optimize
 - Optimizing an upper bound on classification error.



Calculations

• Consider adding the *m*-th component:

$$H_m(\mathbf{x}) = \frac{1}{2} [\alpha_1 h_1(\mathbf{x}) + \dots, + \alpha_m h_m(\mathbf{x})]$$

For a cleaner form later

$$E = \sum_{i=1}^{N} e^{-y^{(i)} H_m(\mathbf{x}^{(i)})} = \sum_{i=1}^{N} e^{-y^{(i)} [H_{m-1}(\mathbf{x}^{(i)}) + \frac{1}{2} \alpha_m h_m(\mathbf{x}^{(i)})]}$$

$$= \sum_{i=1}^{N} e^{-y^{(i)} H_{m-1}(\mathbf{x}^{(i)})} e^{-\frac{1}{2} \alpha_m y^{(i)} h_m(\mathbf{x}^{(i)})} = \sum_{i=1}^{N} \underbrace{w_m^{(i)}}_{e^{-y^{(i)} H_{m-1}(\mathbf{x}^{(i)})}} e^{-\frac{1}{2} \alpha_m y^{(i)} h_m(\mathbf{x}^{(i)})}$$

Suppose it is fixed at stage *m*

Should be optimized at stage m by seeking $h_m(x)$ and α_m

Weighted exponential loss

$$E = \sum_{i=1}^{N} w_m^{(i)} e^{-\frac{1}{2}\alpha_m y^{(i)} h_m(\mathbf{x}^{(i)})}$$

- Sequentially adds a new component trained on reweighted training samples.
- $w_m^{(i)}$: history of classification of $x^{(i)}$ by H_{m-1}
 - Loss weighted towards mistakes
- Iteration *m* optimization:
 - Choose the new component, $h_m = h(x; \theta_m)$
 - And the vote that optimizes the weighted exponential loss, α_m .



Minimizing loss: finding h_m

$$E = \sum_{i=1}^{N} w_{m}^{(i)} e^{-\frac{1}{2}\alpha_{m} y^{(i)} h_{m}(\mathbf{x}^{(i)})}$$

$$= e^{\frac{-\alpha_{m}}{2}} \left(\sum_{y^{(i)} = h_{m}(\mathbf{x}^{(i)})} w_{m}^{(i)} \right) + e^{\frac{\alpha_{m}}{2}} \left(\sum_{y^{(i)} \neq h_{m}(\mathbf{x}^{(i)})} w_{m}^{(i)} \right)$$

$$= (e^{\frac{\alpha_{m}}{2}} - e^{\frac{-\alpha_{m}}{2}}) \left(\sum_{y^{(i)} \neq h_{m}(\mathbf{x}^{(i)})} w_{m}^{(i)} \right) + e^{\frac{-\alpha_{m}}{2}} \left(\sum_{i=1}^{N} w_{m}^{(i)} \right)$$

$$J_{m} = \sum_{i=1}^{N} w_{m}^{(i)} \times I \left(y^{(i)} \neq h_{m}(\mathbf{x}^{(i)}) \right)$$

Find $h_m(x)$ that minimizes J_m

0000000**00000000**00000000000

Minimizing loss: finding α_m

$$\begin{split} \frac{\partial E}{\partial \alpha_m} &= 0 \\ \implies \frac{1}{2} (e^{\frac{\alpha_m}{2}} + e^{\frac{-\alpha_m}{2}}) \left(\sum_{y^{(i)} \neq h_m(\mathbf{x}^{(i)})} w_m^{(i)} \right) - \frac{1}{2} e^{\frac{-\alpha_m}{2}} \left(\sum_{i=1}^N w_m^{(i)} \right) = 0 \\ \implies \frac{e^{\frac{-\alpha_m}{2}}}{(e^{\frac{\alpha_m}{2}} + e^{\frac{-\alpha_m}{2}})} &= \frac{\sum_{y^{(i)} \neq h_m(\mathbf{x}^{(i)})} w_m^{(i)}}{\sum_{i=1}^N w_m^{(i)}} \end{split}$$

$$\epsilon_m = \frac{\sum_{i=1}^{N} w_m^{(i)} I\left(y^{(i)} \neq h_m(\boldsymbol{x}^{(i)})\right)}{\sum_{i=1}^{N} w_m^{(i)}}, \qquad \alpha_m = \ln\left(\frac{1 - \epsilon_m}{\epsilon_m}\right)$$



Updating weights

• Updating weights in AdaBoost algorithm:

$$w_i^{m+1} = w_i^m e^{-\frac{1}{2}\alpha_m y^{(i)} h_m(\mathbf{x}^{(i)})}$$

$$\xrightarrow{y^{(i)}h_m(\mathbf{x}^{(i)})=1-2I(y^{(i)}\neq h_m(\mathbf{x}^{(i)}))} w_i^{m+1} = w_i^m e^{-\frac{1}{2}\alpha_m} e^{\alpha_m I(y^{(i)}\neq h_m(\mathbf{x}^{(i)}))}$$

Independent of i and can be ignored

$$\implies w_i^{m+1} = w_i^m e^{\alpha_m I\left(y^{(i)} \neq h_m(\boldsymbol{x}^{(i)})\right)}$$



- Introduction
- 2 Bagging
- 3 Random Fores
- 4 Boosting
- **5** AdaBoost

Basic idea & example

Algorithm

Loss function

Summary & example

Properties



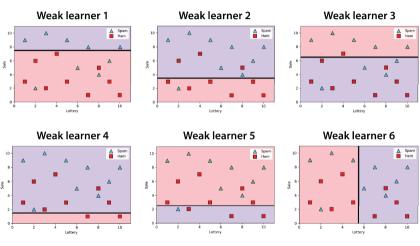
Summary

Algorithm 5 AdaBoost Summary

- 1: **for** i = 1 to N **do**
- 2: Initialize the data weight $w_1^{(i)} = \frac{1}{N}$
- 3: end for
- 4: **for** m = 1 to M **do**
- 5: Find a classifier $h_m(x)$ by minimzing the weighted error function
- 6: Find the normalized weighted error of $h_m(\mathbf{x})$ as ϵ_m
- 7: Compute the new component weight as α_m
- 8: Update example weights for the next iteration $w_{m+1}^{(i)}$
- 9: end for
- 10: Combined classifier $\hat{y} = \text{sign}(H_M(\mathbf{x}))$ where $H_M(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m h_m(\mathbf{x})$



Example

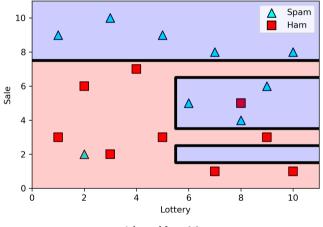






Bagging Random Forest Boosting **AdaBoost** Comparison References

Example, Cont.





- Introduction
- 2 Bagging
- 3 Random Fores
- 4 Boosting
- **5** AdaBoost

Basic idea & example Algorithm Loss function

Properties

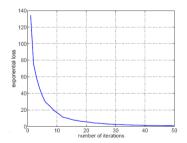


Exponential loss properties

• In each boosting iteration, assuming we can find $h(x; \hat{\theta}_m)$ whose weighted error is better than chance.

$$H_m(x) = \frac{1}{2} [\hat{\alpha}_1 h(\boldsymbol{x}; \hat{\boldsymbol{\theta}}_1) + \dots + \hat{\alpha}_m h(\boldsymbol{x}; \hat{\boldsymbol{\theta}}_m)]$$

• Thus, **lower exponential loss** over training data is guaranteed.

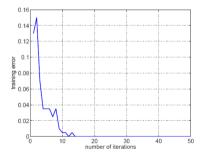


Adopted from [6]



Training error properties

Boosting iterations typically **decrease** the **training error** of $H_m(\mathbf{x})$ over training examples.



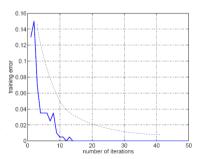
Adopted from [6]



Training error properties, Cont.

• Training error has to go down exponentially fast if the weighted error of each h_m is strictly better than chance (i.e., $\epsilon_m < 0.5$)

$$E_{\text{train}}(H_M) \le \prod_{m=1}^{M} 2\sqrt{\epsilon_m(1-\epsilon_m)}$$



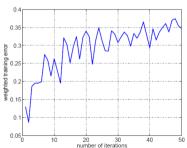
Adopted from [6]



Weighted error properties

 Weighted error of each new component classifier tends to increase as a function of boosting iterations.

$$\epsilon_{m} = \frac{\sum_{i=1}^{N} w_{m}^{(i)} I(y^{(i)} \neq h_{m}(\mathbf{x}^{(i)}))}{\sum_{i=1}^{N} w_{m}^{(i)}}$$

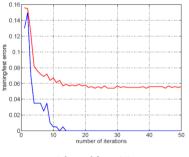


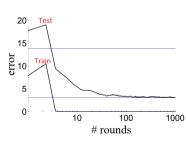
Adopted from [6]



Test error properties

- **Test error** can still **decrease** after training error is flat (even zero).
- But, is it robust to overfitting?
 - May easily overfit in the presence of labeling noise or overlap of classes.





Adopted from [6]

Adopted from [3]

Typical behavior

- Exponential loss goes strictly down.
- Training error of H goes down.
- Weighted error ϵ_m goes $\mathbf{up} \implies \text{votes } \alpha_m$ go \mathbf{down} .
- Test error decreases even after a flat training error.



•0

- Introduction

- **6** Comparison



Bagging Random Forest Boosting AdaBoost **Comparison** References

Bagging vs. Boosting

	Bagging	Boosting
Training Strategy	Parallel training	Sequential training
Data Sampling	Bootstrapping (random subsets)	Weighted (by instance importance)
Learners Dependency	Independent	Dependent (on the previous models)
Learner Weighting	Equal weights	Varying weights (based on importance)
Tolerance to Noise	More robust (due to aggregation)	More sensitive (may overfit to noise)
Properties	Reduces bias	Reduces bias and variance (focus on bias)



- Introduction
- 2 Bagging
- 3 Random Forest
- 4 Boosting
- 6 AdaBoost
- **6** Comparison
- References



Contributions

- This slide has been prepared thanks to:
 - · Nikan Vasei
 - · Mahan Bayhaghi



- [1] C. M., *Pattern Recognition and Machine Learning*. Information Science and Statistics, New York, NY: Springer, 1 ed., Aug. 2006.
- [2] M. Soleymani Baghshah, "Machine learning." Lecture slides.
- [3] R. E. Schapire, "The boosting approach to machine learning: An overview," *Nonlinear estimation and classification*, pp. 149–171, 2003.
- [4] L. Serrano, Grokking machine learning.New York, NY: Manning Publications, Jan. 2022.
- [5] T. Mitchell, Machine Learning. McGraw-Hill series in computer science, New York, NY: McGraw-Hill Professional, Mar. 1997.
- [6] T. Jaakkola, "Machine learning course slides." Lecture slides.

