DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor
Khoury College of Computer Science
Northeastern University

November 3 2020

Announcements

- Projects
  - We have received 21 projects
  - You will get an assigned TA
  - Project milestone will be due on November 24
  - Report of 3-4 pages on progress and challenges
- HW4 will be out soon
  - Due on Nov 17
- Exam is on Thursday, November 19
  - Work on assignments and review class lectures
  - Will share a list of topics

Outline

- Ensemble learning
  - Review
- Boosting
  - General method
  - AdaBoost algorithm
  - Boosting vs Bagging
- Introduction to Deep Learning

Ensemble Learning

Consider a set of classifiers $h_1, \ldots, \mathrm{h}_\mathrm{L}$

Idea: construct a classifier $H(\mathbf{x})$ that combines the individual decisions of $h_1, \ldots, \mathrm{h}_\mathrm{L}$

- e.g., could have the member classifiers vote, or
- e.g., could use different members for different regions of the instance space

Successful ensembles require **diversity** IN THE MODELS

- Classifiers should make different mistakes
- Can have different types of base learners

How to Achieve Diversity

- Avoid overfitting
  - Vary the training data
- Features are noisy
  - Vary the set of features

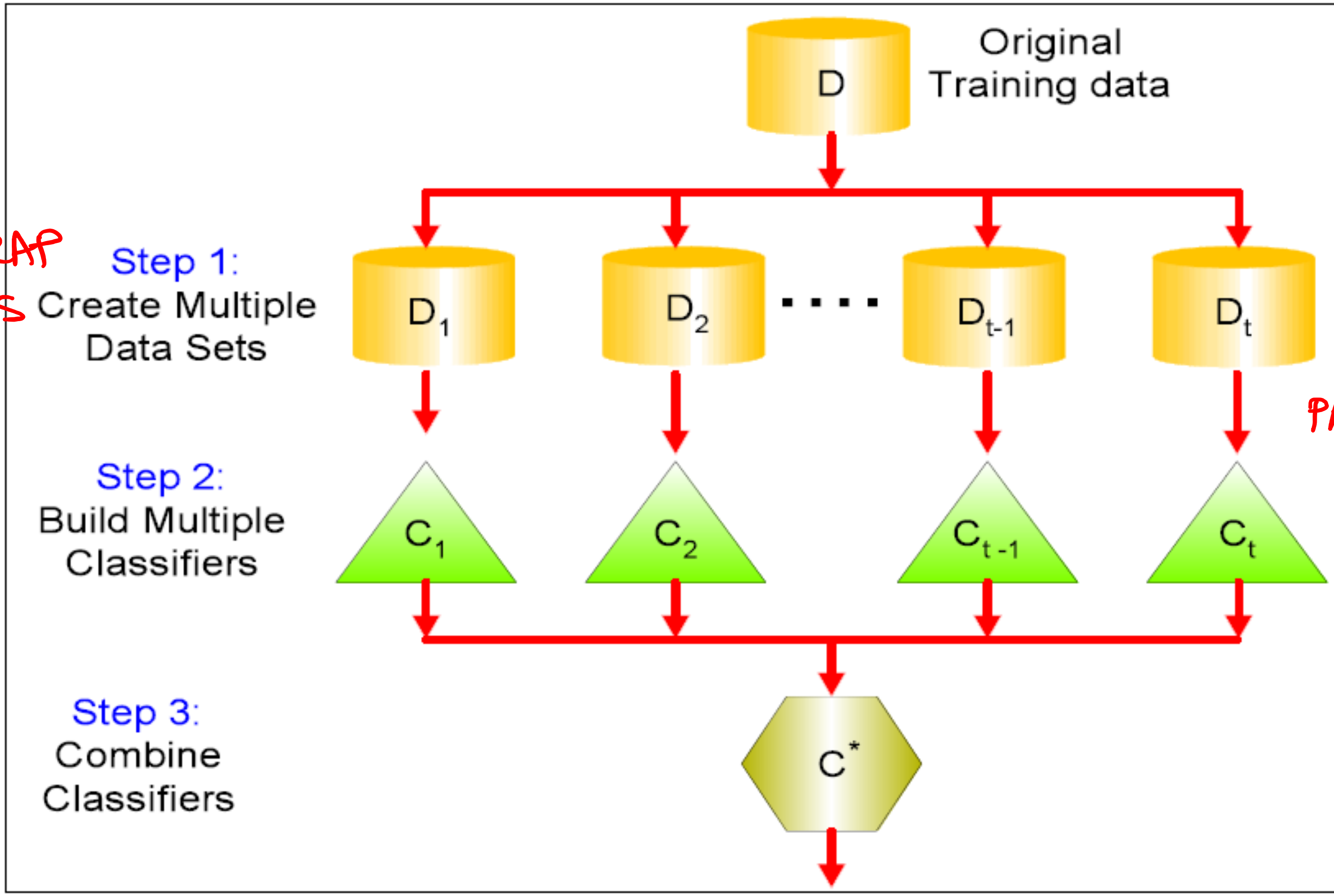
Two main ensemble learning methods

- Bagging (e.g., Random Forests) - BOOTSTRAP SAMPLES

  $\simeq \frac{2}{3}$ OF TRAINING DATA
- Boosting (e.g., AdaBoost)

Bagging

BOOTSTRAP SAMPLES



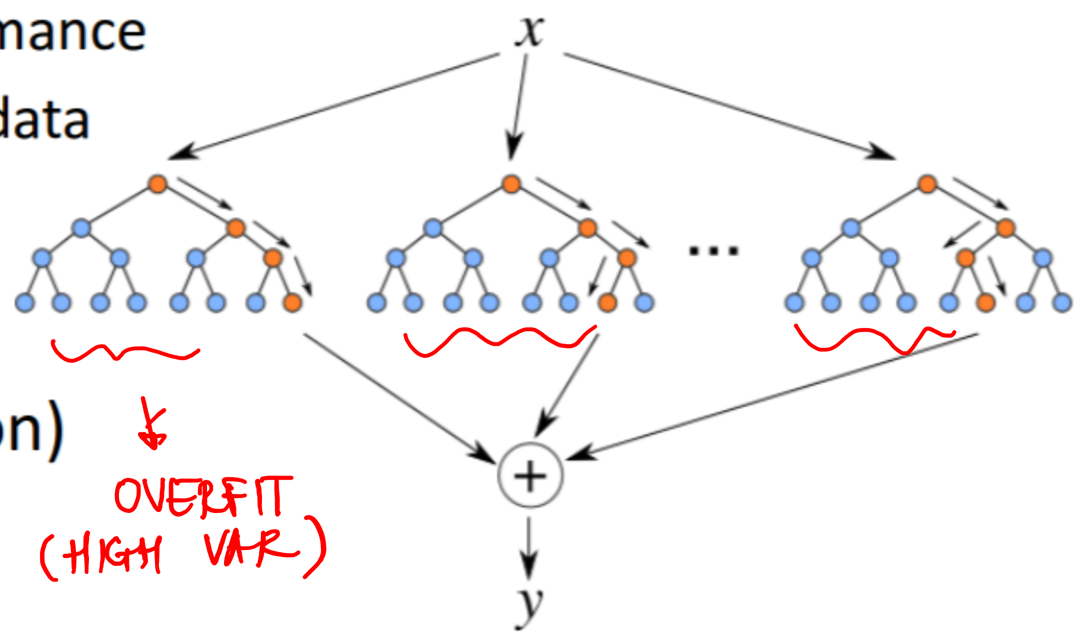
PARALLEL

Majority Votes

Random Forests

BAGGING FOR

- Construct decision trees on bootstrap replicas
  - Restrict the node decisions to a small subset of features picked randomly for each node
- Do not prune the trees
  - Estimate tree performance on out-of-bootstrap data
- Average the output of all trees (or choose mode decision)



OVERFIT
(HIGH VAR)

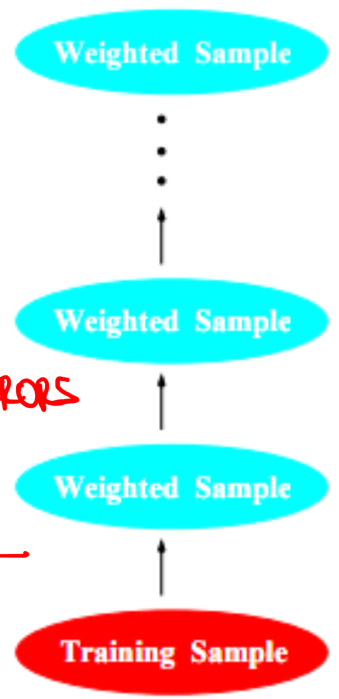
ENSEMBLES REDUCE VARIANCE

AdaBoost

- A meta-learning algorithm with great theoretical and empirical performance
- Turns a base learner (i.e., a “weak hypothesis”) into a high performance classifier
- Creates an ensemble of weak hypotheses by repeatedly emphasizing mispredicted instances

Adaptive Boosting
Freund and Schapire 1997

Overview of AdaBoost



$h_T(x)$

$h_2(x)$

$h_1(x)$

- USE WEIGHTS ON TRAINING
- INCREASE WEIGHTS ON ERRORS
- DECREASE WEIGHTS ON CORRECT POINTS

UNIFORM WEIGHTS

FINAL OUTPUT:

$$G(x) = \sum_{t=1}^{T} \beta_t h_t(x)$$

β_t PROPORTIONAL TO ACCURACY OF h_t

FIGURE 10.1. *Schematic of AdaBoost. Classifiers are trained on weighted versions of the dataset, and then combined to produce a final prediction.*

SEQUENTIAL

Boosting [Shapire '89]

- **Idea:** given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote

- On each iteration t:
  - weight each training example by how incorrectly it was classified
  - Learn a weak hypothesis – h_t
  - A strength for this hypothesis – β_t

- Final classifier: $\boxed{H(x) = sign(\sum \beta_t h_t(x))}$

$y_i \in \{-1, 1\}$

If each weak learner h_t is slightly better than random guessing ($\varepsilon_t < 0.5$), then training error of AdaBoost decays exponentially fast in number of rounds T.

AdaBoost

$y_i \in \{-1, 1\}$

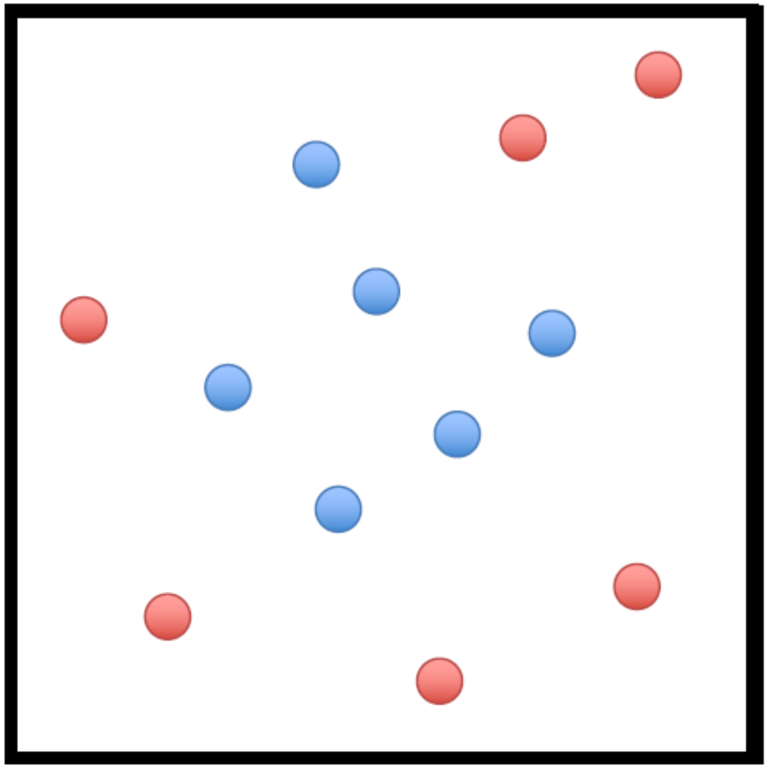
x_i: TRAINING EXAMPLES

1: Initialize a vector of n uniform weights $\mathbf{w}_1$
2: **for** $t = 1, \ldots, T$
3: Train model h_t on X, y with weights $\mathbf{w}_t$
4: Compute the weighted training error of h_t
5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

7: Normalize $\mathbf{w}_{t+1}$ to be a distribution
8: **end for**
9: **Return** the hypothesis

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



- Size of point represents the instance's weight

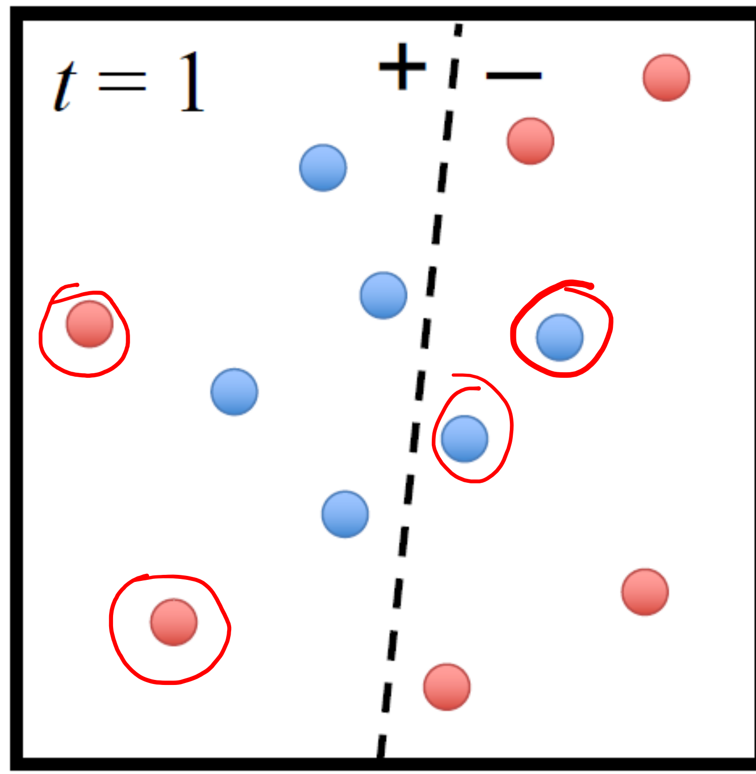
AdaBoost

Initialize a vector of n uniform weights $\mathbf{w}_1$
for $t = 1, \ldots, T$
    Train model h_t on X, y with weights $\mathbf{w}_t$
    Compute the weighted training error of h_t
    Choose $\beta_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
    Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

    Normalize $\mathbf{w}_{t+1}$ to be a distribution
end for
Return the hypothesis

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



If $\varepsilon_t \leq \frac{1}{2} \Rightarrow \frac{1-\varepsilon_t}{\varepsilon_t} \geq 1$

$\Rightarrow \beta_t = \frac{1}{2} \ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right) \geq 0$

$\varepsilon_1 = \frac{4}{12} = \frac{1}{3}$

$\boxed{\text{If } \varepsilon_t > \frac{1}{2} \Rightarrow \text{STOP}}$

AdaBoost

1: Initialize a vector of n uniform weights $\mathbf{w}_1$

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

3: Train model h_t on X, y with weights $\mathbf{w}_t$

4: Compute the weighted training error of h_t

5: Choose $\boxed{\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)}$

6: Update all instance weights:

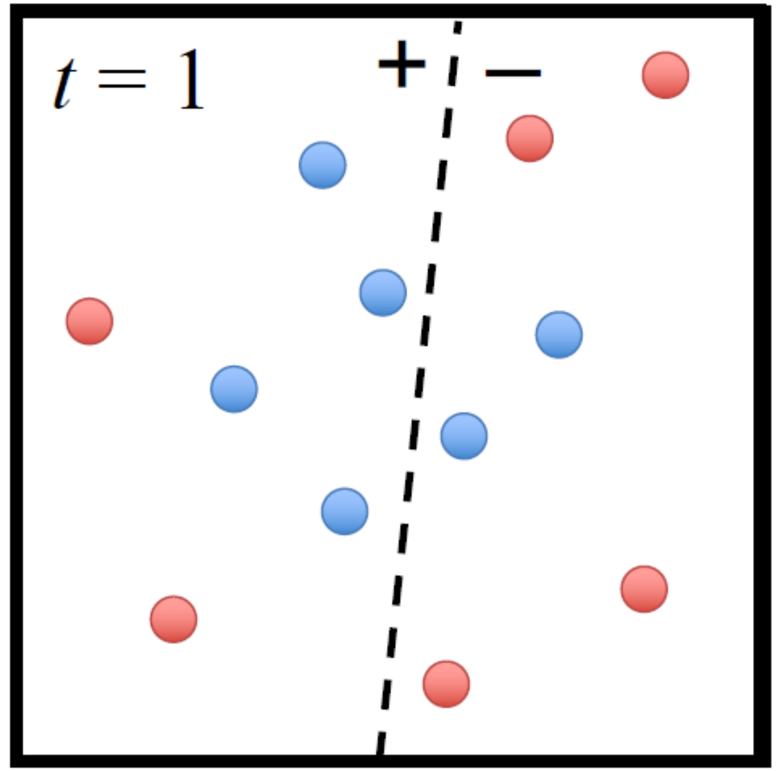
$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

7: Normalize $\mathbf{w}_{t+1}$ to be a distribution

8: **end for**

9: **Return** the hypothesis

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



If ε_t increases $\Rightarrow$ β_t decreases

$$\beta_t = \frac{1}{2} \ln \left(\frac{1}{\varepsilon_t} - 1 \right)$$

β_t = MEASURES IMPORTANCE OF h_t

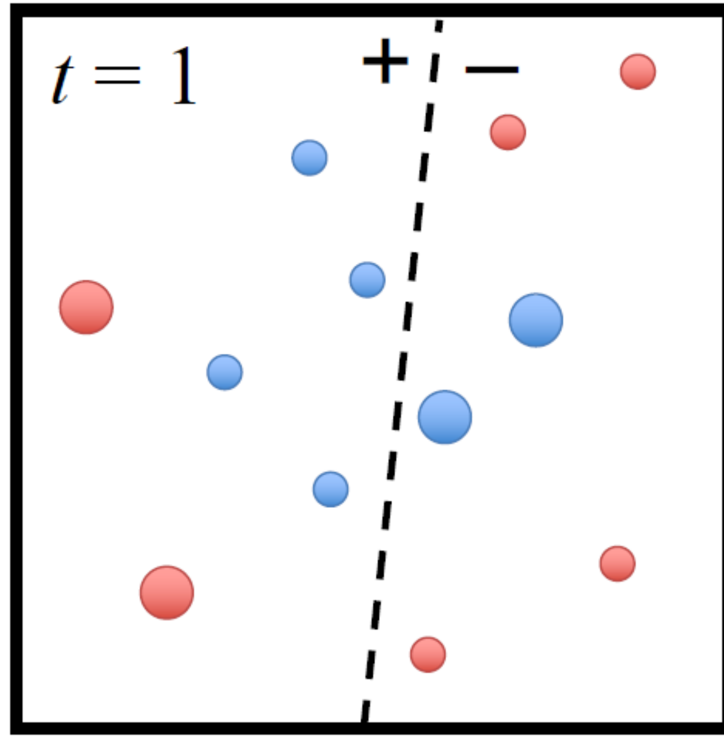
AdaBoost

Initialize a vector of n uniform weights $\mathbf{w}_1$
for $t = 1, \ldots, T$
    Train model h_t on X, y with weights $\mathbf{w}_t$
    Compute the weighted training error of h_t
    Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
    Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

    Normalize $\mathbf{w}_{t+1}$ to be a distribution
end for
Return the hypothesis

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



$$w_{t+1,i} = w_{t,i} \cdot e^{-\beta_t y_i h_t(x_i)}$$

1) If x_i is CORRECT by h_t $\Rightarrow$ $y_i \cdot h_t(x_i) = 1$

$$w_{t+1,i} = w_{t,i} \cdot \underbrace{e^{-\beta_t}} \leq w_{t,i}$$

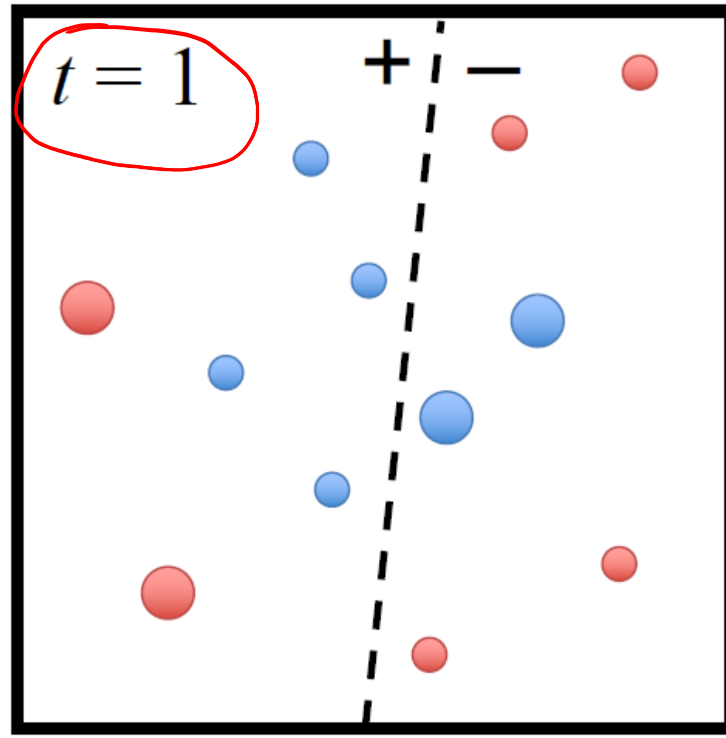
AdaBoost

1: Initialize a vector of n uniform weights $\mathbf{w}_1$
2: **for** $t = 1, \ldots, T$
3: Train model h_t on X, y with weights $\mathbf{w}_t$
4: Compute the weighted training error of h_t
5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

7: Normalize $\mathbf{w}_{t+1}$ to be a distribution
8: **end for**
9: **Return** the hypothesis

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



2) IF x_i IS MISCLASSIFIED BY $h_t \Rightarrow y_i h_t(x_i) = 1$

$$w_{t+1,i} = w_{t,i} \cdot e^{\beta_t} \geq w_{t,i}$$

(7) $\sum_{i=1}^{N} w_{t,i} = 1$

AdaBoost

Initialize a vector of n uniform weights $\mathbf{w}_1$

for $t = 1, \ldots, T$

$\boxed{\text{Train model } h_t \text{ on } X, y \text{ with weights } \mathbf{w}_t}$

Compute the weighted training error of h_t

Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$

Update all instance weights:

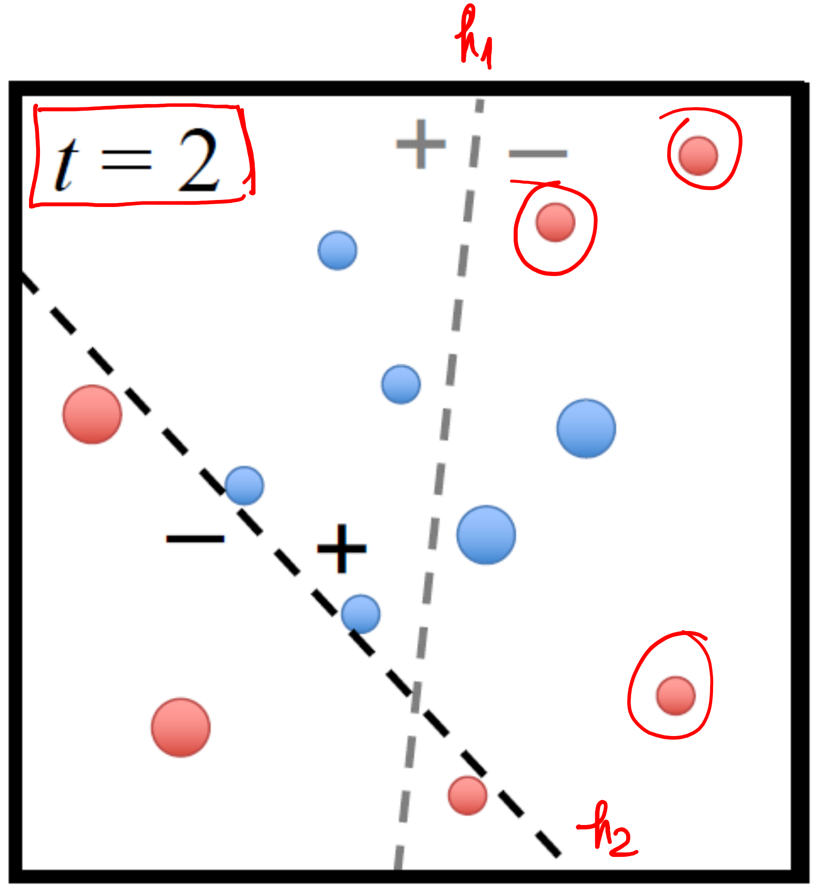
$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

Normalize $\mathbf{w}_{t+1}$ to be a distribution

end for

Return the hypothesis

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



$$\varepsilon_2 = \frac{3}{12} = \frac{1}{4}$$

$$\beta_2 = \frac{1}{2} \ln \left(\frac{1-\varepsilon_2}{\varepsilon_2} \right)$$

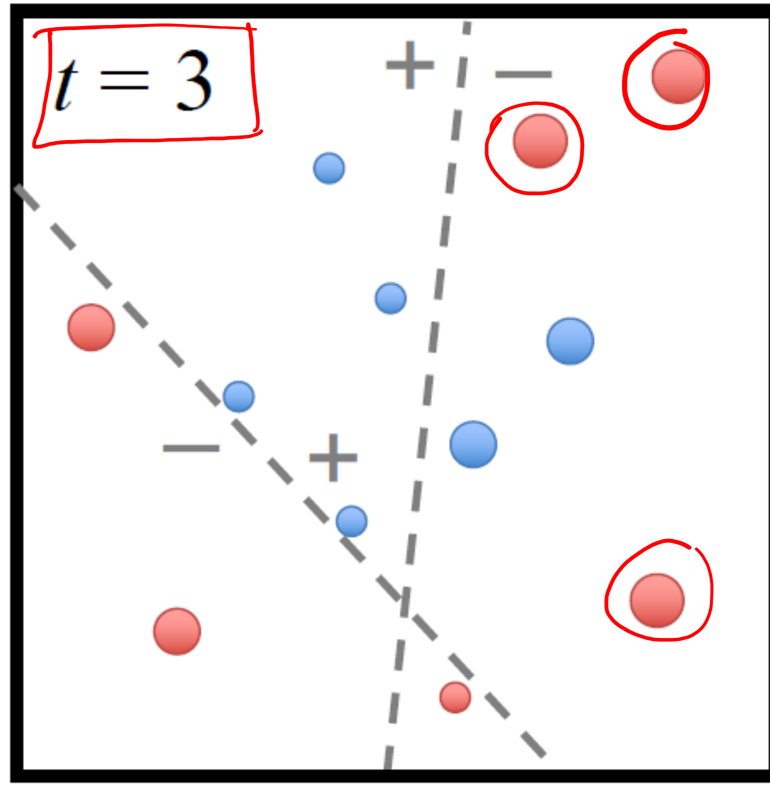
AdaBoost

1: Initialize a vector of n uniform weights $\mathbf{w}_1$
2: **for** $t = 1, \ldots, T$
3: Train model h_t on X, y with weights $\mathbf{w}_t$
4: Compute the weighted training error of h_t
5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

7: Normalize $\mathbf{w}_{t+1}$ to be a distribution
8: **end for**
9: **Return** the hypothesis

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



AdaBoost

1: Initialize a vector of n uniform weights $\mathbf{w}_1$

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

3: Train model h_t on X, y with weights $\mathbf{w}_t$

4: Compute the weighted training error of h_t

5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$

6: Update all instance weights:

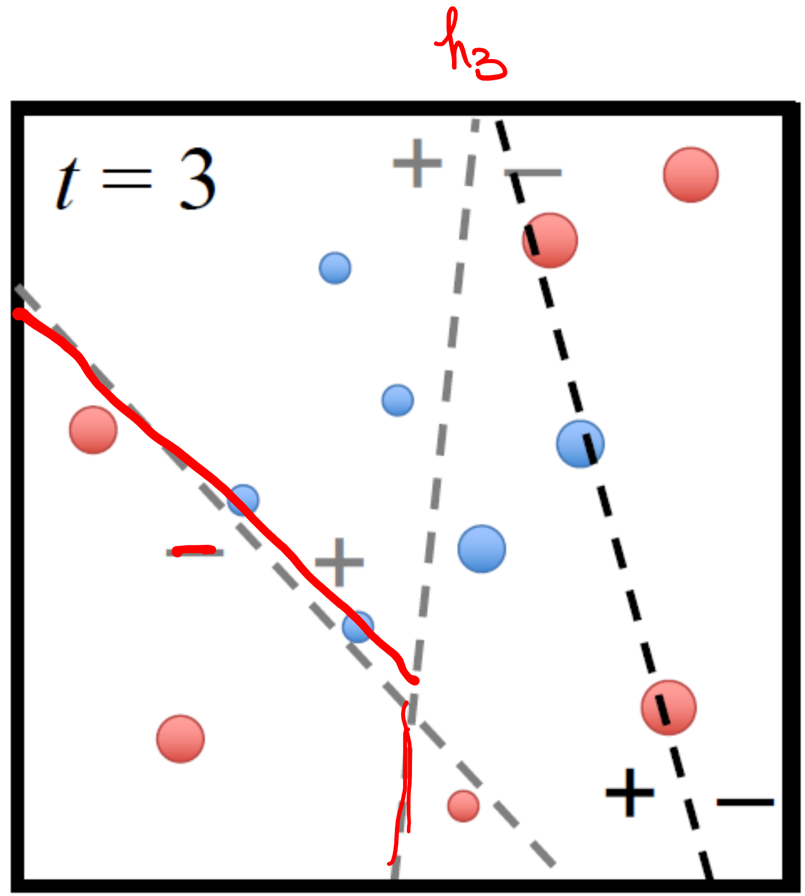
$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

7: Normalize $\mathbf{w}_{t+1}$ to be a distribution

8: **end for**

9: **Return** the hypothesis

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



- Compute importance of hypothesis β_t
- Update weights w_t

AdaBoost

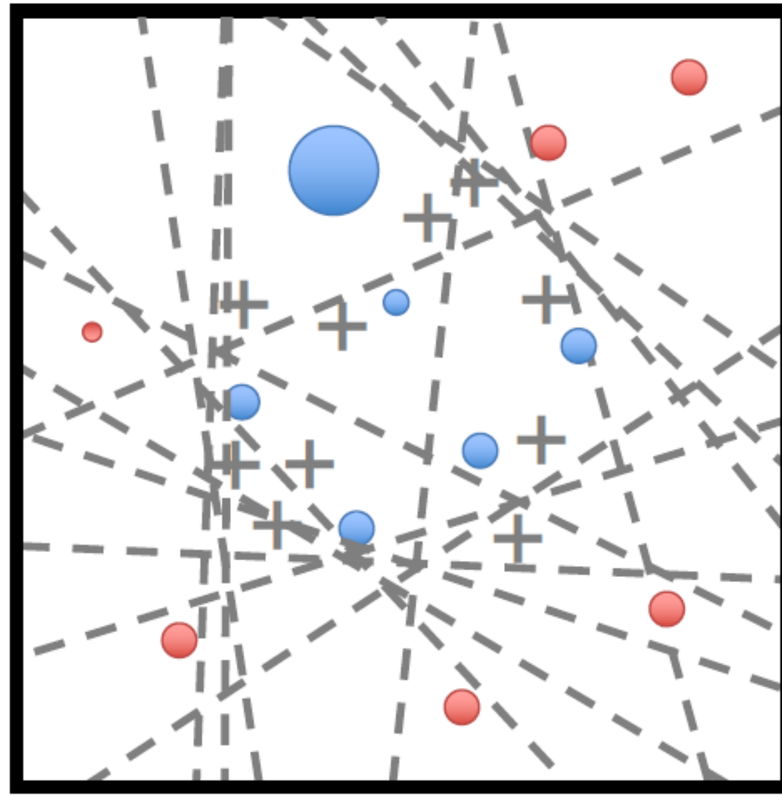
Initialize a vector of n uniform weights $\mathbf{w}_1$
for $t = 1, \ldots, T$
    Train model h_t on X, y with weights $\mathbf{w}_t$
    Compute the weighted training error of h_t
    Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
    Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

    Normalize $\mathbf{w}_{t+1}$ to be a distribution
end for
Return the hypothesis

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

$t = \mathrm{T}$



AdaBoost

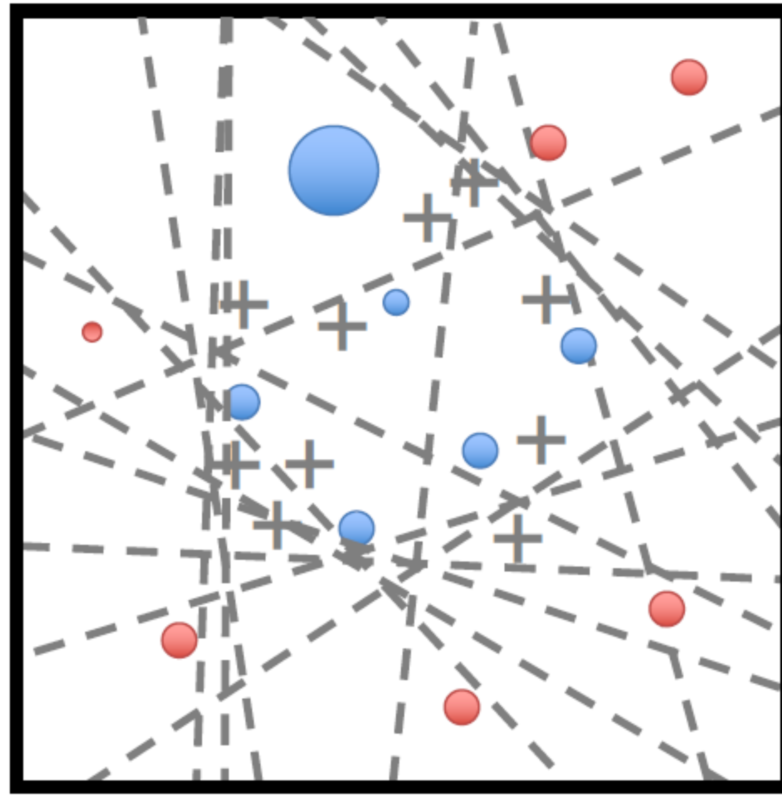
Initialize a vector of n uniform weights $\mathbf{w}_1$
for $t = 1, \ldots, T$
    Train model h_t on X, y with weights $\mathbf{w}_t$
    Compute the weighted training error of h_t
    Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1-\epsilon_t}{\epsilon_t} \right)$
    Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right)$$

    Normalize $\mathbf{w}_{t+1}$ to be a distribution
end for
Return the hypothesis

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

$t = \mathrm{T}$



- Final model is a weighted combination of members
  - Each member weighted by its importance

AdaBoost

$y_i \in \{-1, 1\}$

INPUT: training data $X, y = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$,
the number of iterations T

1: Initialize a vector of n uniform weights $\mathbf{w}_1 = [\underbrace{\frac{1}{n}, \ldots, \frac{1}{n}}]$
2: **for** $t = 1, \ldots, T$
3: Train model h_t on X, y with instance weights $\mathbf{w}_t$
4: Compute the weighted training error rate of h_t:

$$\epsilon_t = \sum_{i : y_i \neq h_t(\mathbf{x}_i)} w_{t,i} \quad \in [0,1]$$

5: Choose $\beta_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$ IMPORTANCE OF h_t
6: Update all instance weights:

$$w_{t+1,i} = w_{t,i} \exp\left(-\beta_t y_i h_t(\mathbf{x}_i)\right) \quad \forall i = 1, \ldots, n$$

7: Normalize $\mathbf{w}_{t+1}$ to be a distribution:

$$w_{t+1,i} = \frac{w_{t+1,i}}{\sum_{j=1}^n w_{t+1,j}} \quad \forall i = 1, \ldots, n \qquad \sum_{i=1}^{n} w_{t+1,i} = 1$$

8: **end for**
9: **Return** the hypothesis

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

BINARY.

Train with Weighted Instances

TRAINING DATA $x_1 \ldots \quad x_N$

$2 \quad 2 \quad 1 \ldots \quad 1$

$x_1, x_1, x_2, x_2, x_3, \ldots, x_N$

/ CHANGE TRAINING DATA

1) WEIGHT TRAINING DATA:

- CREATE BOOTSTRAP SAMPLES

SAMPLE POINTS BASED ON w_i

GENERAL; APPLY TO ANY MODEL

2) OBJECTIVE (LOSS) + GRADIENT DESCENT

LOSS: $J(\theta) = \sum_{i=1}^{N} \text{cost}(x_i)$

- APPLIES TO LOGISTIC REGRESSION, SVM, DEEP NETS

TRAIN WITH WEIGHTS: $J(\theta) = \sum_{i=1}^{N} w_i \, \text{cost}(x_i)$

Train with Weighted Instances

(2) • For algorithms like logistic regression, can simply incorporate weights w into the cost function
  - Essentially, weigh the cost of misclassification differently for each instance

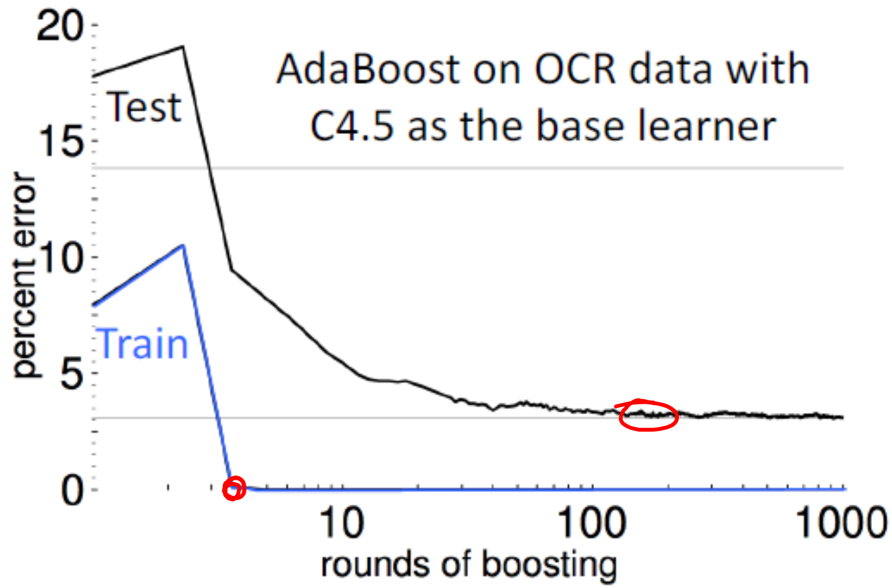
$$J_{\text{reg}}(\boldsymbol{\theta}) = -\sum_{i=1}^{n} w_i \left[y_i \log h_{\boldsymbol{\theta}}(\mathbf{x}_i) + (1 - y_i) \log \left(1 - h_{\boldsymbol{\theta}}(\mathbf{x}_i)\right)\right] + \lambda \|\boldsymbol{\theta}_{[1:d]}\|_2^2$$

(1) • For algorithms that don't directly support instance weights (e.g., ID3 decision trees, etc.), use weighted bootstrap sampling
  - Form training set by resampling instances with replacement according to w

Properties

- If a point is repeatedly misclassified
  - Its weight is increased every time
  - Eventually it will generate a hypothesis that correctly predicts it
- In practice AdaBoost does not typically overfit
- Does not use explicitly regularization

Resilience to overfitting



• USES WEAK, SIMPLE CLASSIFIER

- Empirically, boosting resists overfitting
- Note that it continues to drive down the test error even AFTER the training error reaches zero

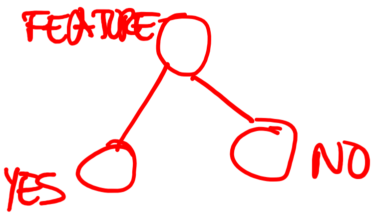
Increases confidence in prediction when adding more rounds

Base Learner Requirements

- AdaBoost works best with “weak” learners
  - Should not be complex
  - Typically high bias classifiers
  - Works even when weak learner has an error rate just slightly under 0.5 (i.e., just slightly better than random)
    - Can prove training error goes to 0 in O(log n) iterations

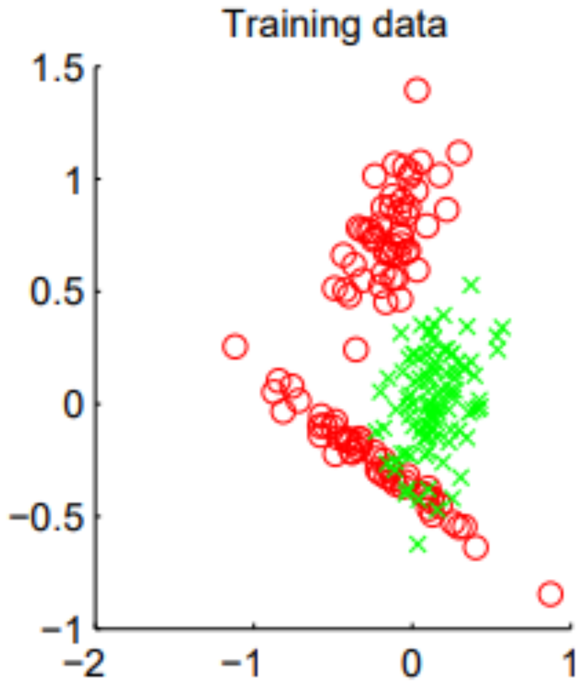
BOOSTING REDUCES BIAS

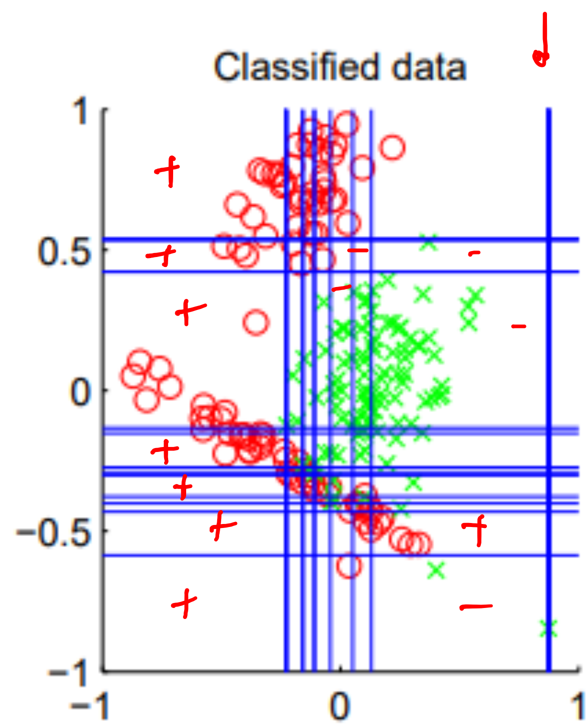
- Examples:
  - Decision stumps (1 level decision trees)
  - Depth-limited decision trees
  - Linear classifiers



AdaBoost with Decision Stumps

DECISION STUMP





AdaBoost in Practice

Strengths:

- Fast and simple to program
- No parameters to tune (besides T) CROSS-VALIDATION.
- No assumptions on weak learner $\varepsilon_t < \frac{1}{2}$

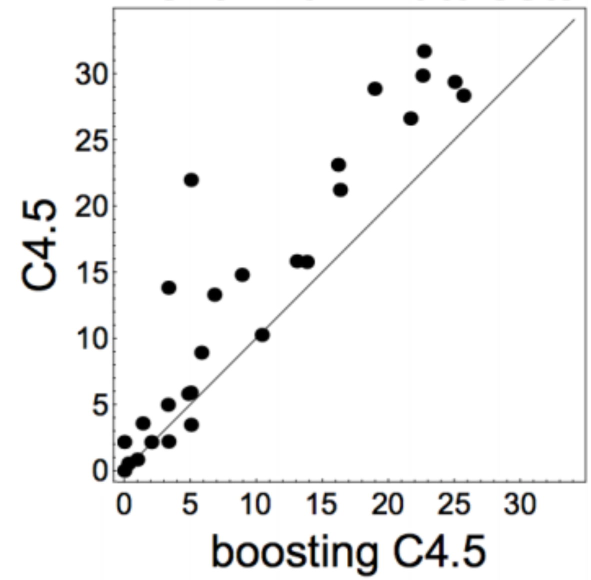
When boosting can fail:

- Given insufficient data
- Overly complex weak hypotheses
- Can be susceptible to noise
- When there are a large number of outliers

Boosted Decision Trees

- Boosted decision trees are one of the best "off-the-shelf" classifiers
  - i.e., no parameter tuning
- Limit member hypothesis complexity by limiting tree depth
- Gradient boosting methods are typically used with trees in practice

Error Rates on 27 Benchmark Data Sets



"AdaBoost with trees is the best off-the-shelf classifier in the world" -Breiman, 1996
(Also, see results by Caruana & Niculescu-Mizil, ICML 2006)

Bagging vs Boosting

	BAGGING	BOOSTING
VARIANCE	↓	↓
BIAS	~ SAME	↓ MORE COMPLEX
MODELS	COMPLEX DECISION TREES	SIMPLE DECISION STUMPS
DIVERSITY	BOOTSTRAP SAMPLES SUBSET OF FEATURES	WEIGHTING TRAINING EXAMPLES
PREDICTION	UNIFORM CONTRIBUTION	WEIGHTED CONTRIBUTION
OUTLIERS	MORE RESILIENT	

Bagging vs Boosting

Bagging	**vs.**	**Boosting**
Resamples data points		Reweights data points (modifies their distribution)
Weight of each classifier is the same		Weight is dependent on classifier's accuracy
Only variance reduction		Both bias and variance reduced – learning rule becomes more complex with iterations

Applicable to complex models with low bias, high variance

Applicable to weak models with high bias, low variance

Review Ensembles

- Ensemble learning are powerful learning methods
  - Better accuracy than standard classifiers
- Bagging uses bootstrapping (with replacement), trains T models, and averages their prediction
  - Random forests vary training data and feature set at each split
- Boosting is an ensemble of T weak learners that emphasizes mis-predicted examples
  - AdaBoost has great theoretical and experimental performance
  - Can be used with linear models or simple decision trees (stumps, fixed-depth decision trees)

Acknowledgements

- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
- Thanks!