

Ensemble Techniques

Introduction to Data Mining, 2nd Edition

by

Tan, Steinbach, Karpatne, Kumar

Ensemble Methods

- Construct a set of base classifiers learned from the training data
- Predict class label of test records by combining the predictions made by multiple classifiers (e.g., by taking majority vote)

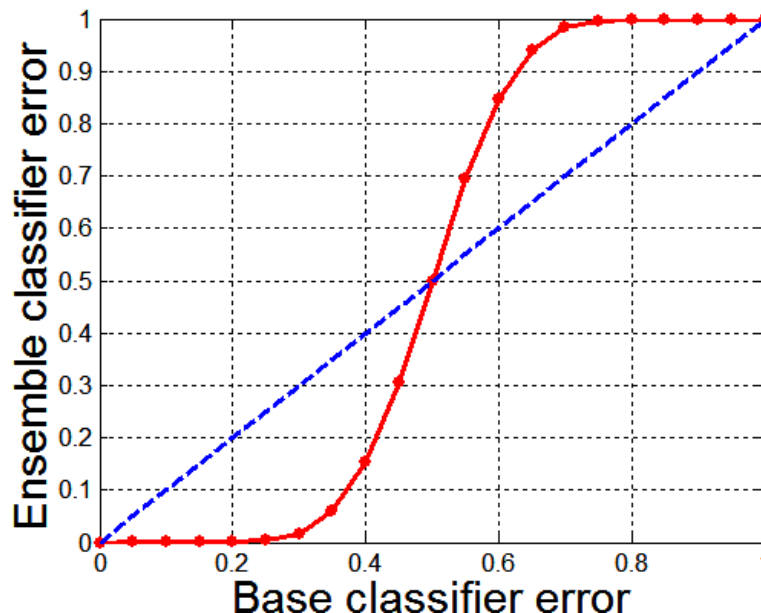
Example: Why Do Ensemble Methods Work?

- Suppose there are 25 base classifiers
 - Each classifier has error rate, $\epsilon = 0.35$
 - Majority vote of classifiers used for classification
 - If all classifiers are identical:
 - ◆ Error rate of ensemble = ϵ (0.35)
 - If all classifiers are independent (errors are uncorrelated):
 - ◆ Error rate of ensemble = probability of having more than half of base classifiers being wrong

$$e_{\text{ensemble}} = \sum_{i=13}^{25} \binom{25}{i} \epsilon^i (1 - \epsilon)^{25-i} = 0.06$$

Necessary Conditions for Ensemble Methods

- Ensemble Methods work better than a single base classifier if:
 1. All base classifiers are independent of each other
 2. All base classifiers perform better than random guessing (error rate < 0.5 for binary classification)



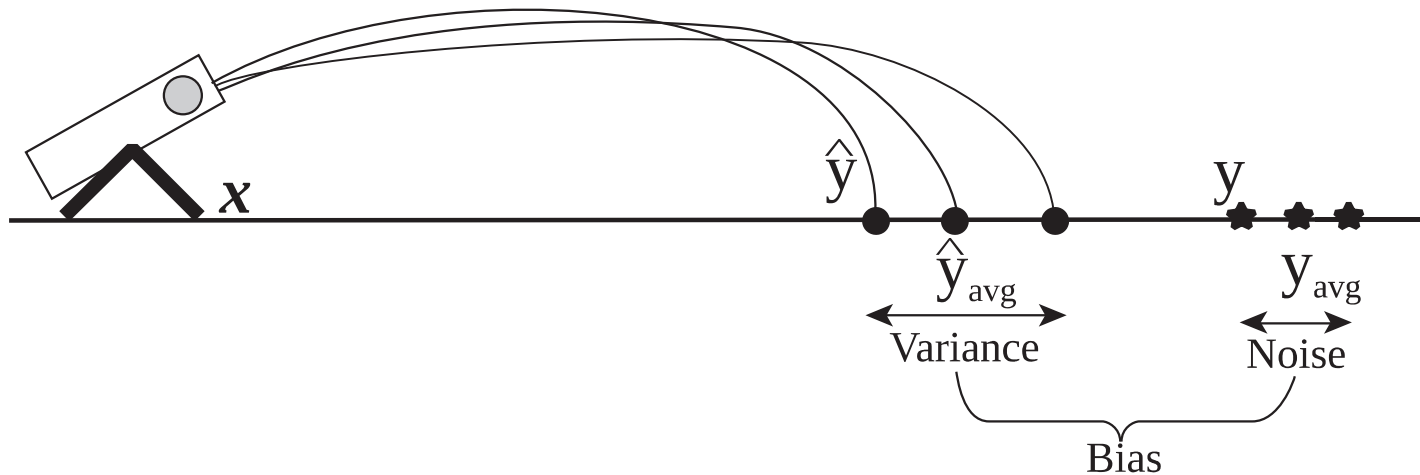
Classification error for an ensemble of 25 base classifiers, assuming their errors are uncorrelated.

Rationale for Ensemble Learning

- Ensemble Methods work best with **unstable base classifiers**
 - Classifiers that are sensitive to minor perturbations in training set, due to *high model complexity*
 - Examples: Unpruned decision trees, ANNs, ...

Bias-Variance Decomposition

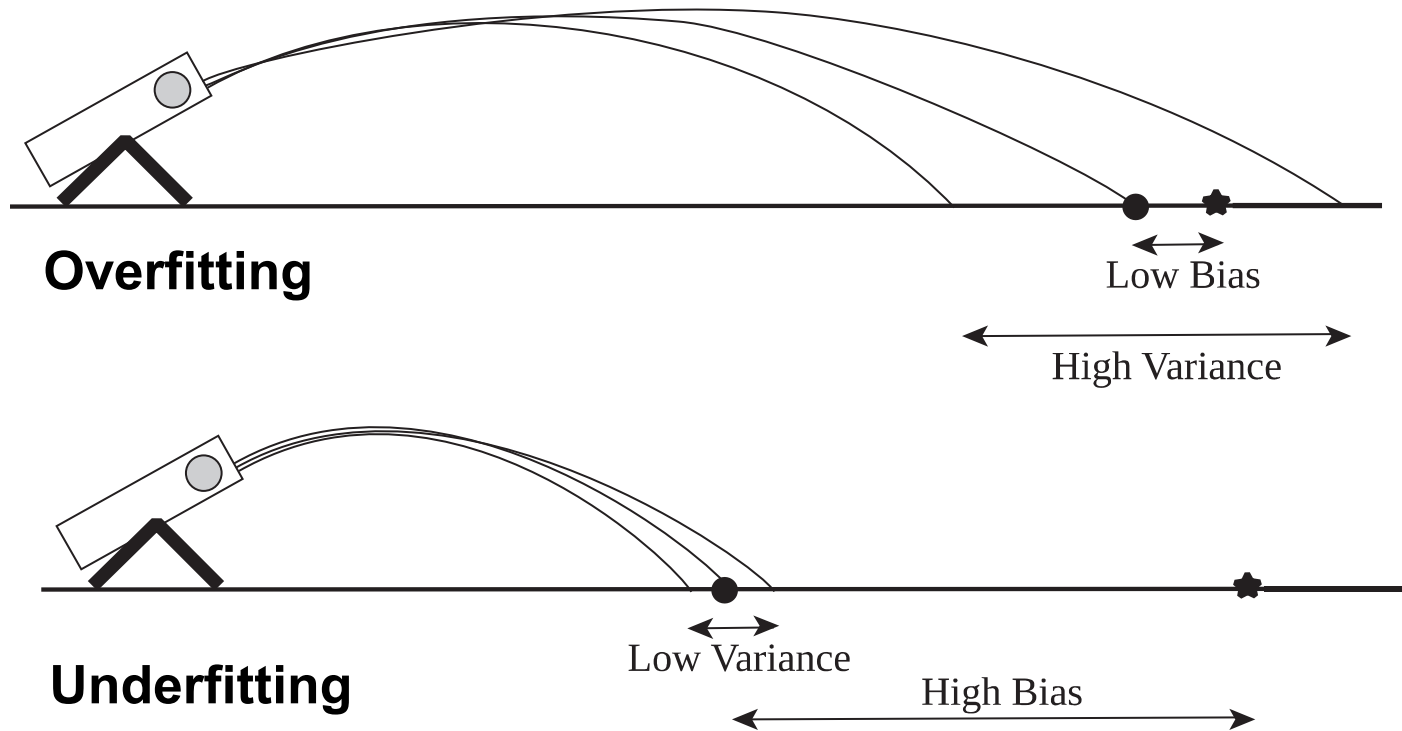
- Analogous problem of reaching a target y by firing projectiles from x (regression problem)



- For classification, the generalization error of model m can be given by:

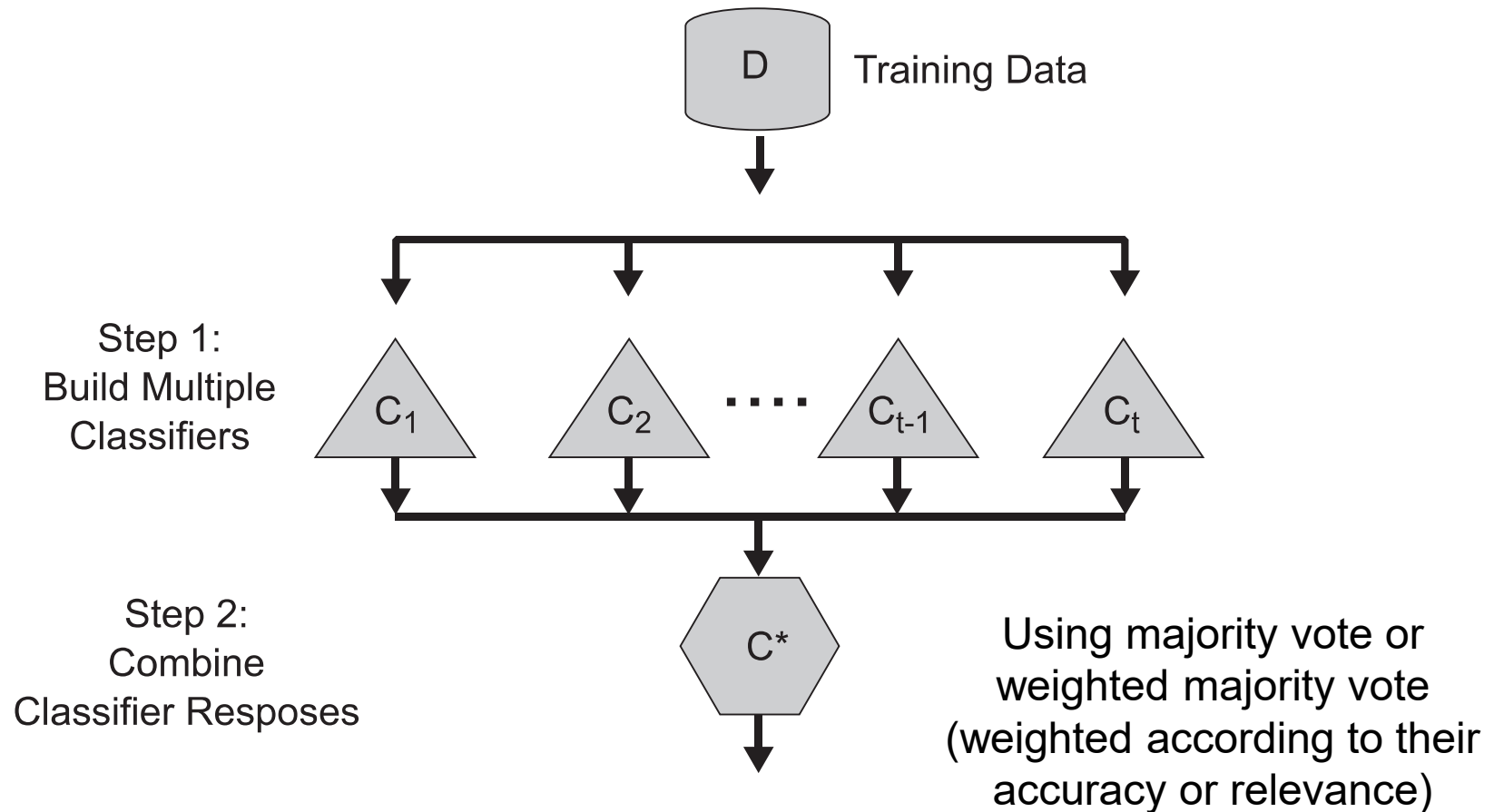
$$gen.error(m) = bias(m) + variance(m) + noise(m)$$

Bias-Variance Trade-off and Overfitting



- Ensemble methods try to reduce the variance of complex models (with low bias) by *aggregating* responses of multiple base classifiers

General Approach of Ensemble Learning



Constructing Ensemble Classifiers

- By manipulating training set
 - Example: bagging, boosting, random forests
- By manipulating input features
 - Example: random forests
- By manipulating class labels
 - Example: error-correcting output coding
- By manipulating learning algorithm
 - Example: injecting randomness in the initial weights of ANN

Bagging (Bootstrap AGGregatING)

- Bootstrap sampling: sampling with replacement

Original Data	1	2	3	4	5	6	7	8	9	10
Bagging (Round 1)	7	8	10	8	2	5	10	10	5	9
Bagging (Round 2)	1	4	9	1	2	3	2	7	3	2
Bagging (Round 3)	1	8	5	10	5	5	9	6	3	7

- Build classifier on each bootstrap sample
- Probability of a training instance being selected in a bootstrap sample is:
 - $1 - (1 - 1/n)^n$ (n: number of training instances)
 - ~ 0.632 when n is large

Bagging Algorithm

Algorithm 4.5 Bagging algorithm.

- 1: Let k be the number of bootstrap samples.
 - 2: **for** $i = 1$ to k **do**
 - 3: Create a bootstrap sample of size N , D_i .
 - 4: Train a base classifier C_i on the bootstrap sample D_i .
 - 5: **end for**
 - 6: $C^*(x) = \underset{y}{\operatorname{argmax}} \sum_i \delta(C_i(x) = y).$
 $\{\delta(\cdot) = 1 \text{ if its argument is true and } 0 \text{ otherwise.}\}$
-

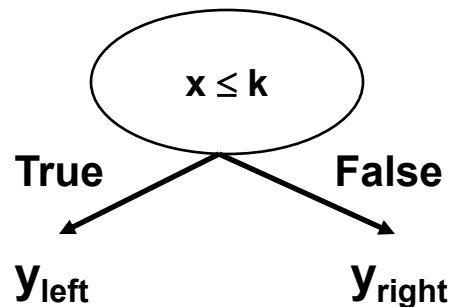
Bagging Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump (decision tree of size 1)
 - Decision rule: $x \leq k$ versus $x > k$
 - Split point k is chosen based on entropy



Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$

$x > 0.35 \rightarrow y = -1$

Bagging Example

Bagging Round 1:

x	0.1	0.2	0.2	0.3	0.4	0.4	0.5	0.6	0.9	0.9
y	1	1	1	1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$

$x > 0.35 \rightarrow y = -1$

Bagging Round 2:

x	0.1	0.2	0.3	0.4	0.5	0.5	0.9	1	1	1
y	1	1	1	-1	-1	-1	1	1	1	1

$x \leq 0.7 \rightarrow y = 1$

$x > 0.7 \rightarrow y = 1$

Bagging Round 3:

x	0.1	0.2	0.3	0.4	0.4	0.5	0.7	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.35 \rightarrow y = 1$

$x > 0.35 \rightarrow y = -1$

Bagging Round 4:

x	0.1	0.1	0.2	0.4	0.4	0.5	0.5	0.7	0.8	0.9
y	1	1	1	-1	-1	-1	-1	-1	1	1

$x \leq 0.3 \rightarrow y = 1$

$x > 0.3 \rightarrow y = -1$

Bagging Round 5:

x	0.1	0.1	0.2	0.5	0.6	0.6	0.6	1	1	1
y	1	1	1	-1	-1	-1	-1	1	1	1

$x \leq 0.35 \rightarrow y = 1$

$x > 0.35 \rightarrow y = -1$

Bagging Example

Bagging Round 6:

x	0.2	0.4	0.5	0.6	0.7	0.7	0.7	0.8	0.9	1
y	1	-1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$
 $x > 0.75 \rightarrow y = 1$

Bagging Round 7:

x	0.1	0.4	0.4	0.6	0.7	0.8	0.9	0.9	0.9	1
y	1	-1	-1	-1	-1	1	1	1	1	1

$x \leq 0.75 \rightarrow y = -1$
 $x > 0.75 \rightarrow y = 1$

Bagging Round 8:

x	0.1	0.2	0.5	0.5	0.5	0.7	0.7	0.8	0.9	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$
 $x > 0.75 \rightarrow y = 1$

Bagging Round 9:

x	0.1	0.3	0.4	0.4	0.6	0.7	0.7	0.8	1	1
y	1	1	-1	-1	-1	-1	-1	1	1	1

$x \leq 0.75 \rightarrow y = -1$
 $x > 0.75 \rightarrow y = 1$

Bagging Round 10:

x	0.1	0.1	0.1	0.1	0.3	0.3	0.8	0.8	0.9	0.9
y	1	1	1	1	1	1	1	1	1	1

$x \leq 0.05 \rightarrow y = 1$
 $x > 0.05 \rightarrow y = 1$

Bagging Example

□ Summary of Trained Decision Stumps:

Round	Split Point	Left Class	Right Class
1	0.35	1	-1
2	0.7	1	1
3	0.35	1	-1
4	0.3	1	-1
5	0.35	1	-1
6	0.75	-1	1
7	0.75	-1	1
8	0.75	-1	1
9	0.75	-1	1
10	0.05	1	1

Bagging Example

- Use majority vote (sign of sum of predictions) to determine class of ensemble classifier

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	1	1	1	-1	-1	-1	-1	-1	-1	-1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
4	1	1	1	-1	-1	-1	-1	-1	-1	-1
5	1	1	1	-1	-1	-1	-1	-1	-1	-1
6	-1	-1	-1	-1	-1	-1	-1	1	1	1
7	-1	-1	-1	-1	-1	-1	-1	1	1	1
8	-1	-1	-1	-1	-1	-1	-1	1	1	1
9	-1	-1	-1	-1	-1	-1	-1	1	1	1
10	1	1	1	1	1	1	1	1	1	1
Sum	2	2	2	-6	-6	-6	-6	2	2	2
Predicted Class Sign	1	1	1	-1	-1	-1	-1	1	1	1

- Bagging can also increase the complexity (representation capacity) of simple classifiers such as decision stumps

Boosting

- An iterative procedure to adaptively change distribution of training data by focusing more on previously misclassified records
 - Initially, all N records are assigned equal weights (for being selected for training)
 - Unlike bagging, weights may change at the end of each boosting round

Boosting

- Records that are wrongly classified will have their weights increased in the next round
- Records that are classified correctly will have their weights decreased in the next round

Original Data	1	2	3	4	5	6	7	8	9	10
Boosting (Round 1)	7	3	2	8	7	9	4	10	6	3
Boosting (Round 2)	5	4	9	4	2	5	1	7	4	2
Boosting (Round 3)	4	4	8	10	4	5	4	6	3	4

- Example 4 is hard to classify
- Its weight is increased, therefore it is more likely to be chosen again in subsequent rounds

AdaBoost

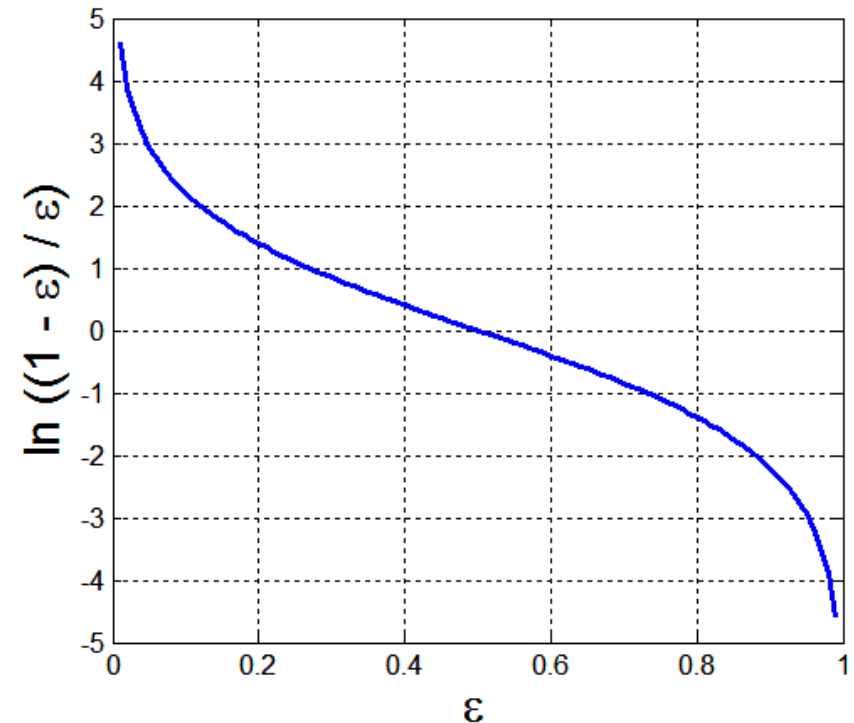
□ Base classifiers: $C_1, C_2, \dots, C_T$

□ Error rate of a base classifier:

$$\epsilon_i = \frac{1}{N} \sum_{j=1}^N w_j^{(i)} \delta(C_i(x_j) \neq y_j)$$

□ Importance of a classifier:

$$\alpha_i = \frac{1}{2} \ln \left(\frac{1 - \epsilon_i}{\epsilon_i} \right)$$



AdaBoost Algorithm

- Weight update: (eq. 1)

$$w_j^{(i+1)} = \frac{w_j^{(i)}}{Z_i} \times \begin{cases} e^{-\alpha_i} & \text{if } C_i(x_j) = y_j \\ e^{\alpha_i} & \text{if } C_i(x_j) \neq y_j \end{cases}$$

Where Z_i is the normalization factor

- If any intermediate rounds produce error rate higher than 50%, the weights are reverted back to $1/n$ and the resampling procedure is repeated
- Classification:

$$C^*(x) = \arg \max_y \sum_{i=1}^T \alpha_i \delta(C_i(x) = y)$$

AdaBoost Algorithm

Algorithm 4.6 AdaBoost algorithm.

- 1: $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \dots, N\}$. {Initialize the weights for all N examples.}
 - 2: Let k be the number of boosting rounds.
 - 3: **for** $i = 1$ to k **do**
 - 4: Create training set D_i by sampling (with replacement) from D according to $\mathbf{w}$.
 - 5: Train a base classifier C_i on D_i .
 - 6: Apply C_i to all examples in the original training set, D .
 - 7: $\epsilon_i = \frac{1}{N} \left[\sum_j w_j \delta(C_i(x_j) \neq y_j) \right]$ {Calculate the weighted error.}
 - 8: **if** $\epsilon_i > 0.5$ **then**
 - 9: $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \dots, N\}$. {Reset the weights for all N examples.}
 - 10: Go back to Step 4.
 - 11: **end if**
 - 12: $\alpha_i = \frac{1}{2} \ln \frac{1-\epsilon_i}{\epsilon_i}$.
 - 13: Update the weight of each example according to Equation 1, slide 21
 - 14: **end for**
 - 15: $C^*(\mathbf{x}) = \operatorname{argmax}_y \sum_{j=1}^T \alpha_j \delta(C_j(\mathbf{x}) = y)$.
-

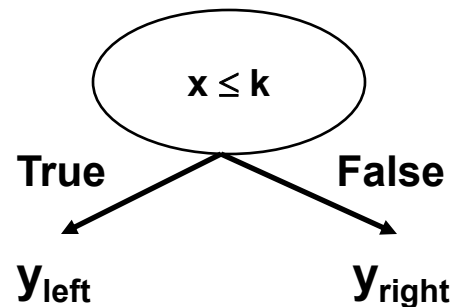
AdaBoost Example

- Consider 1-dimensional data set:

Original Data:

x	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y	1	1	1	-1	-1	-1	-1	1	1	1

- Classifier is a decision stump
 - Decision rule: $x \leq k$ versus $x > k$
 - Split point k is chosen based on entropy



AdaBoost Example

- Training sets for the first 3 boosting rounds:

Boosting Round 1:

x	0.1	0.4	0.5	0.6	0.6	0.7	0.7	0.7	0.8	1
y	1	-1	-1	-1	-1	-1	-1	-1	1	1

Boosting Round 2:

x	0.1	0.1	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
y	1	1	1	1	1	1	1	1	1	1

Boosting Round 3:

x	0.2	0.2	0.4	0.4	0.4	0.4	0.5	0.6	0.6	0.7
y	1	1	-1	-1	-1	-1	-1	-1	-1	-1

- Summary:

Round	Split Point	Left Class	Right Class	alpha
1	0.75	-1	1	1.738
2	0.05	1	1	2.7784
3	0.3	1	-1	4.1195

AdaBoost Example

□ Weights

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01
3	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009

□ Classification

Round	x=0.1	x=0.2	x=0.3	x=0.4	x=0.5	x=0.6	x=0.7	x=0.8	x=0.9	x=1.0
1	-1	-1	-1	-1	-1	-1	-1	1	1	1
2	1	1	1	1	1	1	1	1	1	1
3	1	1	1	-1	-1	-1	-1	-1	-1	-1
Sum	5.16	5.16	5.16	-3.08	-3.08	-3.08	-3.08	0.397	0.397	0.397
Sign	1	1	1	-1	-1	-1	-1	1	1	1

Predicted
Class

Random Forest Algorithm

- Construct an ensemble of decision trees by manipulating training set as well as features
 - Use bootstrap sample to train every decision tree (similar to Bagging)
 - Use the following tree induction algorithm:
 - ◆ At every internal node of decision tree, randomly sample p attributes for selecting split criterion
 - ◆ Repeat this procedure until all leaves are pure (unpruned tree)

Characteristics of Random Forest

- Base classifiers are unpruned trees and hence are *unstable classifiers*
- Base classifiers are *decorrelated* (due to randomization in training set as well as features)
- Random forests reduce variance of unstable classifiers without negatively impacting the bias
- Selection of hyper-parameter p
 - Small value ensures lack of correlation
 - High value promotes strong base classifiers
 - Common default choices: $\sqrt{d}$, $\log_2(d + 1)$

Gradient Boosting

- Constructs a series of models
 - Models can be any predictive model that has a differentiable loss function
 - Commonly, trees are the chosen model
 - ◆ XGboost (extreme gradient boosting) is a popular package because of its impressive performance
- Boosting can be viewed as optimizing the loss function by iterative functional gradient descent.
- Implementations of various boosted algorithms are available in Python, R, Matlab, and more.