slides originally by
Dr. Richard Burns,
modified by
Dr. Stephanie Schwartz

ENSEMBLE METHODS

CSCI 452: Data Mining

Ensemble Methods

- Currently using *one single classifier* induced from training data as our model, to predict class of test instance
- What if we used multiple decision trees?
- *Motivation:* methods work surprisingly well, usually greatly improve decision tree accuracy
- Can also be applied to other learning algorithms

Ensemble Methods

- Ensemble Methods: construction and use of a *set* of base classifiers
  - Training multiple classifiers
  - How to predict a test instance?
  - *Idea:* Let each model have a vote on the correct class prediction. Assign the class with the most votes.

Rationale

- How can an ensemble method improve a classifier's performance?
- Assume we have 25 binary classifiers
- Each has error rate: $\varepsilon = 0.35$

1. If all 25 classifiers are identical:
   - They will vote the same way on each test instance
   - Ensemble error rate: $\varepsilon = 0.35$

Rationale

- How can an ensemble method improve a classifier's performance?
- Assume we have 25 binary classifiers
- Each has error rate: $\varepsilon = 0.35$

2. If all 25 classifiers are <u>independent</u> (errors are uncorrelated):
   - Ensemble method only makes a wrong prediction if more than half of the base classifiers predict incorrectly.

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

6% much less than 35%

Rati

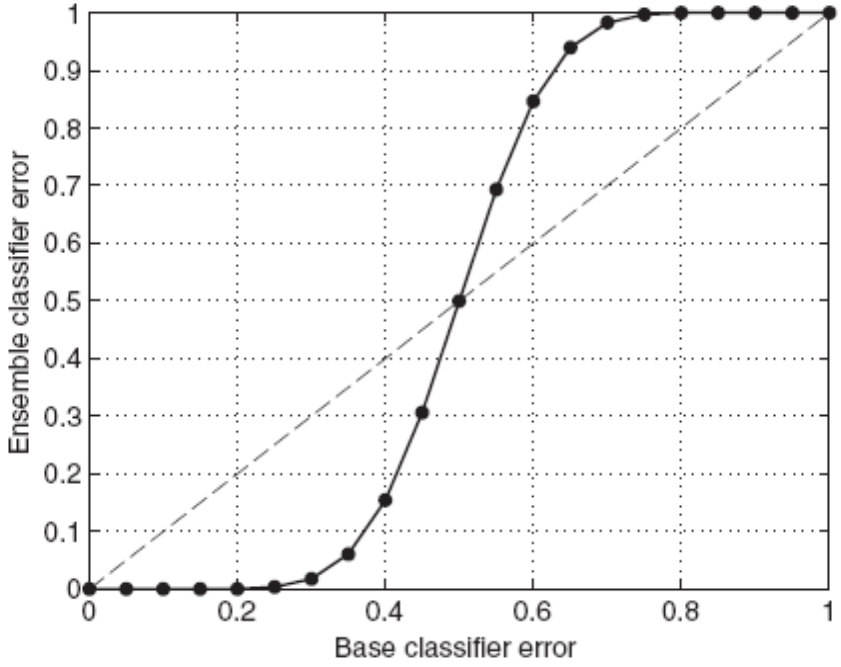
- In practice, difficult to have base classifiers than are completely independent.
- Ensemble methods have been shown to improve classification accuracies even when there is some correlation.

- Conditions necessary for an ensemble classifier to perform better than a single classifier:
  1. Base classifiers should be *independent* of each other
  2. Base classifiers should not do worse than a classifier doing random guessing
     - *Example:* for two-class problem, base classifier error rate: $\varepsilon < .5$

Error Rate Comparison

Ensemble method performs worse than single base classifier when $\varepsilon > 0.5$

Comparison of 25 base classifiers when error rate is varied



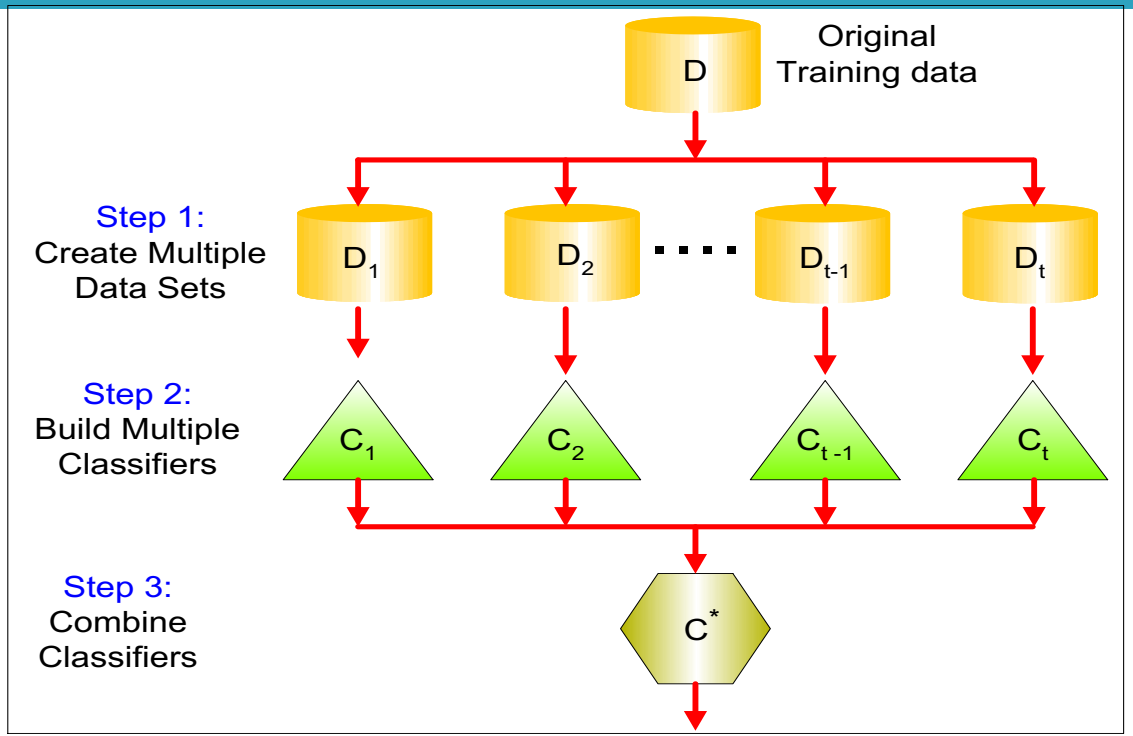
- Dashed diagonal line: when base classifiers are identical
- Solid curve: when base classifiers are independent

Figure 5.30. Comparison between errors of base classifiers and errors of the ensemble classifier.

Ensemble Methods

- Ensemble techniques: Improve classification accuracy by aggregating the predictions of multiple classifiers
  - (also sometimes called "classifier combination" methods)
- General Procedure:
  - Construct set of *base classifiers* from training data
  - Predict class label of previously unseen records by aggregating predictions made by each base classifier

Methods for Constructing an Ensemble Classifier



Methods for Constructing an Ensemble Classifier

1. By manipulating the training set.
2. By manipulating the input features.
3. By manipulating the class labels.
4. By manipulating the learning algorithm.

1. By manipulating the training set.

- Multiple training sets created by resampling original data
- Resample according to some sampling distribution
  - *Example:* equal probability
  - *Example:* weighted
  - Determines how likely it is that an example will be selected for training.
- Classifier built from each training set.
- Ensemble methods that manipulate the training set:
  1. Bagging
  2. Boosting

2. By manipulating the input features.

- Subset of input features is chosen at random from overall collection of features
- Each training set has different feature set
- Ensemble method that manipulates input features:
  1. Random Forest
     - Works well with datasets that contain highly redundant features

3. By manipulating the class labels.

- Used when there is a large number of classes
- Transform into many binary class problems
- Training Approach:
  1. Randomly partition class labels into two disjoint subsets A_0 (class 0) and A_1 (class 1)
  2. Train a base classifier based on this class reassignment.
  3. Repeat multiple times, once for each base classifier.
- Testing Approach:
  1. Each base classifier predicts test instance with its respective binary class subset
  2. All classes in subset receive a vote
  3. Class with highest count wins

4. By manipulating the learning algorithm.

- … so that applying algorithm on same training data may result in different models
- How to introduce randomness into decision tree induction?
  - Instead of choose best splitting attribute at each node, randomly choose one of top *k* attributes for splitting

Ensemble Methods

- Ensemble methods work best with unstable classifiers, base classifiers that are sensitive to minor perturbations in the training set.
  - *Example:* decision trees
  - *Unstable classifiers* have *high variability.*

Bagging

On average, each D_i will contain 63% of original training data.
Probability of sample being selected for D_i: $1 - (1 - (1/N)^N$

- Converges to: $1 - 1/e = 0.632$

- Ensemble method that "manipulates the training set"
- *Action:* repeatedly sample *with replacement* according to uniform probability distribution
  - Every instance has equal chance of being picked
  - Some instances may be picked multiple times; others may not be chosen
- *Sample Size:* same as training set
- D_i: each bootstrap sample
- *Footnote:* also called bootstrap aggregating

Bagging Algorithm

Model Generation:

- Let *n* be the number of instances in the training data.
- For each of *t* iterations:
  - Sample *n* instances with replacement from training data.
  - Apply the learning algorithm to the sample.
  - Store the resulting model.

Classification:

- For each of the *t* models:
  - Predict class of instance using model.
- Return class that has been predicted most often.

Bagging Example

Now going to apply bagging and create many decision stump base classifiers.

Dataset: 10 instances
Predictor Variable: x
Target Variable: y

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

- Decision Stump: one-level binary decision tree
- What's the best performance of a decision stump on this data?
- Splitting condition will be $x <= k$, where k is the split point
  - Best splits: $x <= 0.35$ or $x <= 0.75$
  - Best accuracy: 70%

Bagging Example

- First choose how many “bagging rounds” to perform
  - Chosen by analyst
- We’ll do 10 bagging rounds in this example:

In each round, create D_i by sampling with replacement

Bagging Example

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

Learn decision stump.

What stump will be learned?

If x <= 0.35 then y = 1

If x > 0.35 then y = -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	0.1	0.2	0.3	0.4	0.5	0.8	0.9	1	1	1
Y	1	1	1	-1	-1	1	1	1	1	1
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	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	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	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	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	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	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	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

If x <= 0.35 then y = 1
If x > 0.35 then y = -1

If x <= 0.65 then y = 1
If x > 0.65 then y = 1

If x <= 0.35 then y = 1
If x > 0.35 then y = -1

If x <= 0.3 then y = 1
If x > 0.3 then y = -1

If x <= 0.35 then y = 1
If x > 0.35 then y = -1

If x <= 0.75 then y = -1
If x > 0.75 then y = 1

If x <= 0.75 then y = -1
If x > 0.75 then y = 1

If x <= 0.75 then y = -1
If x > 0.75 then y = 1

If x <= 0.75 then y = -1
If x > 0.75 then y = 1

If x <= 0.05 then y = 1
If x > 0.05 then y = -1

10 bagging rounds. 10 D_i's. 10 learned models.

Classify test instance by using each base classifier and taking majority vote.

Bagging Example

10 test instances. Let's see how each classifier votes:

100% overall ensemble method accuracy (*improvement from 70%*)

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
Sign	1	1	1	-1	-1	-1	-1	1	1	1
True	1	1	1	-1	-1	-1	-1	1	1	1

Bagging Summary

- *In Previous Example:* even though base classifier was decision stump (*depth=1*), bagging aggregated the classifiers to effectively learn a decision tree of *depth=2*
- Bagging helps to reduce variance
- Bagging does not focus on any particular instance of training data
  - less susceptible to model overfitting with noisy data
  - robust to minor perturbations in training set
- If base classifiers are stable (not much variance), bagging can degrade performance
  - Training set is ~37% smaller than original data

Boosting

- Sample with nonuniform distribution
  - Unlike bagging where each instance had equal chance of being selected
  - *Motivation:* focus on instances that are harder to classify
  - *How:* give harder instances more weight in future rounds

Boosting Example

1. Initially instances are assigned weights of 1/N
   - Each is equally likely to be chosen for sample
2. Sample drawn *with replacement*: D_i
3. Classifier induced on D_i
4. Weights of training examples are updated:
   - Instances classified incorrectly have weights increased
   - Instances classified correctly have weights decreased

Boosting Example

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

- Suppose that *Instance #4* is hard to classify.
- Weight for this instance will be increased in future iterations, as it gets misclassified repeatedly.
- Examples not chosen in previous round (*Instances #1, #5*) also may have better chance of being selected in next round.
  - *Why?* Predictions in previous round are likely to be wrong since they weren't trained on.

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

- As boosting rounds proceed, instances that are the hardest to classify become even more prevalent.

Boosting Algorithms

- Several different boosting algorithms exist
- Different by:
  1. How weights of training instances are updated after each boosting round
  2. How predictions made by each classifier are combined
     - Each boosting round produces one base classifier

AdaBoost

- AdaBoost is a popular boosting algorithm
- Regarding predictions of final ensemble classifier:
  - Importance of a base classier depends on its error rate

$$\varepsilon_i = \frac{1}{N}\left[\sum_{j=1}^{N} \omega_j I\left(C_i(x_j)\right) \neq y_j\right]$$

$$I(p) = 1 \text{ if predicate } p \text{ is true, and } 0 \text{ otherwise}$$

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

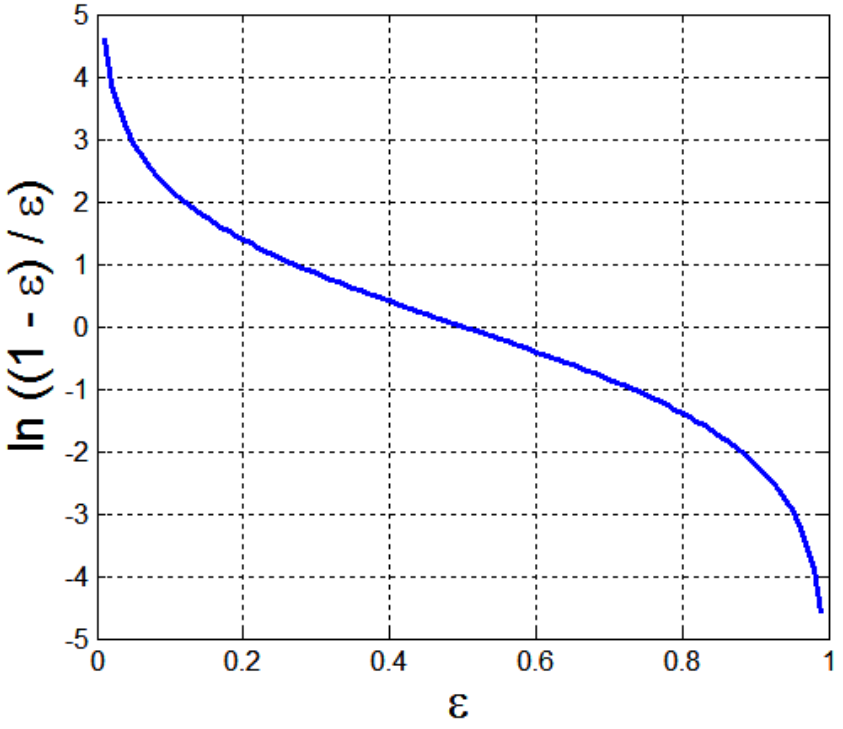
C_i: Base Classifier

ε_i: error rate

α_i: importance of classifier

AdaBoost

- α_i has a large positive value if error rate is close to 0
- α_i has a large negative value if error rate is close to 1



AdaBoost

- α_i also used to update weight of training examples after each boosting round

$$\omega_i^{(j+1)} = \frac{\omega_i^{(j)}}{Z_j} \times \begin{cases} \exp^{-\alpha_j} & \text{if } C_j(x_i) = y_i \\ \exp^{\alpha_j} & \text{if } C_j(x_i) \neq y_i \end{cases}$$

j: current round
j+1: next round
Z_j: normalization factor

$$\sum_i \omega_i^{(j+1)} = 1$$

- Increases weights of incorrectly classified instances
- Decreases weights of correctly classified instances

AdaBoost Example

Dataset: 10 instances
Predictor Variable: x
Target Variable: y

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

- Initially all instances have equal weights

AdaBoost Example

Initial Weights:	X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	ω_i	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	Σ=1
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
Updated Weights:	X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	ω_i	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01	Σ=1
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
Updated Weights:	X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
	ω_i	0.029	0.029	0.029	0.228	0.228	0.228	0.228	0.009	0.009	0.009	Σ=1
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

AdaBoost Example

Initial Weights:

X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
ω_i	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1

$\Sigma=1$

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

- Going to learn decision stump base classifier
- What is the model?

Model:

If x <= 0.75 then y = -1

If x > 0.75 then y = 1

X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
True Y	1	1	1	-1	-1	-1	-1	1	1	1
Pred. Y	-1	-1	-1	-1	-1	-1	-1	1	1	1

$$\varepsilon_1 = \frac{1}{10}\left[\sum \omega_j \cdot I\right] = \frac{1}{10}\left[\sum .1 \cdot I\right] = 0.03$$

$$\text{Classifier Importance } \alpha_1 = \frac{1}{2}\ln\left(\frac{1-.03}{.03}\right) \approx 1.738$$

AdaBoost Example

X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
ω_1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
Update:	$0.1 \times e^{1.738}$	$0.1 \times e^{1.738}$	$0.1 \times e^{1.738}$	$0.1 \times e^{-1.738}$	$0.1 \times e^{-1.738}$	$0.1 \times e^{-1.738}$	$0.1 \times e^{-1.738}$	$0.1 \times e^{-1.738}$	$0.1 \times e^{-1.738}$	$0.1 \times e^{-1.738}$
	0.568	0.568	0.568	0.018	0.018	0.018	0.018	0.018	0.018	0.018

$\Sigma = .568 \times 3 + .018 \times 7 = 1.83$

Normalizing: (dividing each by sum)

ω_2	0.311	0.311	0.311	0.01	0.01	0.01	0.01	0.01	0.01	0.01

$\Sigma = 1$

AdaBoost Prediction

- Prediction made by each base classifier C_i is weighted by α_i
  - Instead of majority voting scheme (used in Bagging)

AdaBoost Example

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

Round	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
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

$$-1 \times 1.738 + 1 \times 2.7784 + 1 \times 4.1195 = 5.16$$

Boosting Algorithm

From Tan, Alg. 5.7

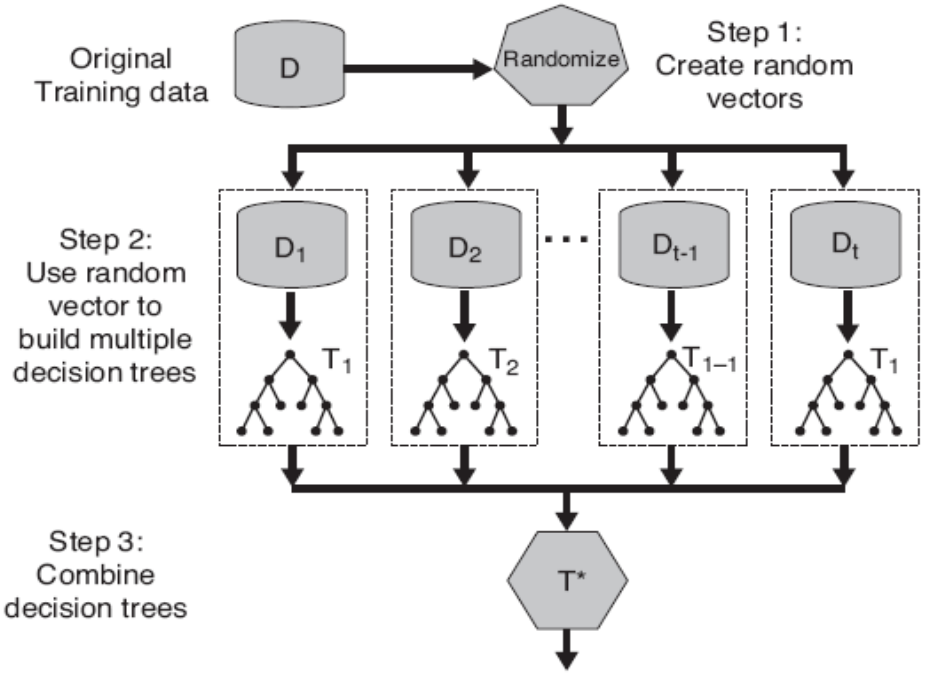
Algorithm 5.7 AdaBoost algorithm.

1: $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$. {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: $\epsilon_i = \frac{1}{N}\left[\sum_j w_j \, \delta\big(C_i(x_j) \neq y_j\big)\right]$ {Calculate the weighted error.}
8: **if** $\epsilon_i > 0.5$ **then**
9: $\mathbf{w} = \{w_j = 1/N \mid j = 1, 2, \ldots, N\}$. {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 5.69.
14: **end for**
15: $C^*(\mathbf{x}) = \underset{y}{\operatorname{argmax}} \sum_{j=1}^{T} \alpha_j \delta(C_j(\mathbf{x}) = y)\big)$.

Random Forests

- Designed for decision tree classifiers
  - Bagging and Boosting can be applied to other learning algorithms
- Example of "manipulating the input features"
- *Idea:* combines predictions made by multiple decision trees
  - each tree is induced based on an independent set of random vectors
    - Random forests are generated from *fixed probability distribution* (unlike Boosting)
    - Bagging using decision trees is a special case of Random Forests

Random Forests



Different variations of Random Forests exist

Random Forests Algorithm

- "Normal" decision tree induction utilizes *full* set of features to determine each split
- Random forest injects randomness:
  - Forest-RI: Selection of random *subset* of features to determine each split
  - How large should subset be?
    - A subset of size ($\log_2 d + 1$) for d features is commonly used
  - Tree is grown to full size with pruning
- Overall prediction is majority vote from all individually trained trees

Ensemble Method Performance on Classic Data Sets

Data Set	Number of (Attributes, Classes, Records)	Decision Tree (%)	Bagging (%)	Boosting (%)	RF (%)
Anneal	(39, 6, 898)	92.09	94.43	95.43	95.43
Australia	(15, 2, 690)	85.51	87.10	85.22	85.80
Auto	(26, 7, 205)	81.95	85.37	85.37	84.39
Breast	(11, 2, 699)	95.14	96.42	97.28	96.14
Cleve	(14, 2, 303)	76.24	81.52	82.18	82.18
Credit	(16, 2, 690)	85.8	86.23	86.09	85.8
Diabetes	(9, 2, 768)	72.40	76.30	73.18	75.13
German	(21, 2, 1000)	70.90	73.40	73.00	74.5
Glass	(10, 7, 214)	67.29	76.17	77.57	78.04
Heart	(14, 2, 270)	80.00	81.48	80.74	83.33
Hepatitis	(20, 2, 155)	81.94	81.29	83.87	83.23
Horse	(23, 2, 368)	85.33	85.87	81.25	85.33
Ionosphere	(35, 2, 351)	89.17	92.02	93.73	93.45
Iris	(5, 3, 150)	94.67	94.67	94.00	93.33
Labor	(17, 2, 57)	78.95	84.21	89.47	84.21
Led7	(8, 10, 3200)	73.34	73.66	73.34	73.06
Lymphography	(19, 4, 148)	77.03	79.05	85.14	82.43
Pima	(9, 2, 768)	74.35	76.69	73.44	77.60
Sonar	(61, 2, 208)	78.85	78.85	84.62	85.58
Tic-tac-toe	(10, 2, 958)	83.72	93.84	98.54	95.82
Vehicle	(19, 4, 846)	71.04	74.11	78.25	74.94
Waveform	(22, 3, 5000)	76.44	83.30	83.90	84.04
Wine	(14, 3, 178)	94.38	96.07	97.75	97.75
Zoo	(17, 7, 101)	93.07	93.07	95.05	97.03

Ensemble Methods Summary

- Advantages:
  - Astonishingly good performance
  - Modeling human behavior: making judgment based on many trusted advisors, each with their own specialty
- Disadvantage:
  - Combined models rather hard to analyze
    - Tens or hundreds of individual models
  - Current research: making models more comprehensible

References

- *Introduction to Data Mining*, 1st edition, Tan et al.