Ensemble Learning: An Introduction

Ensemble Learning

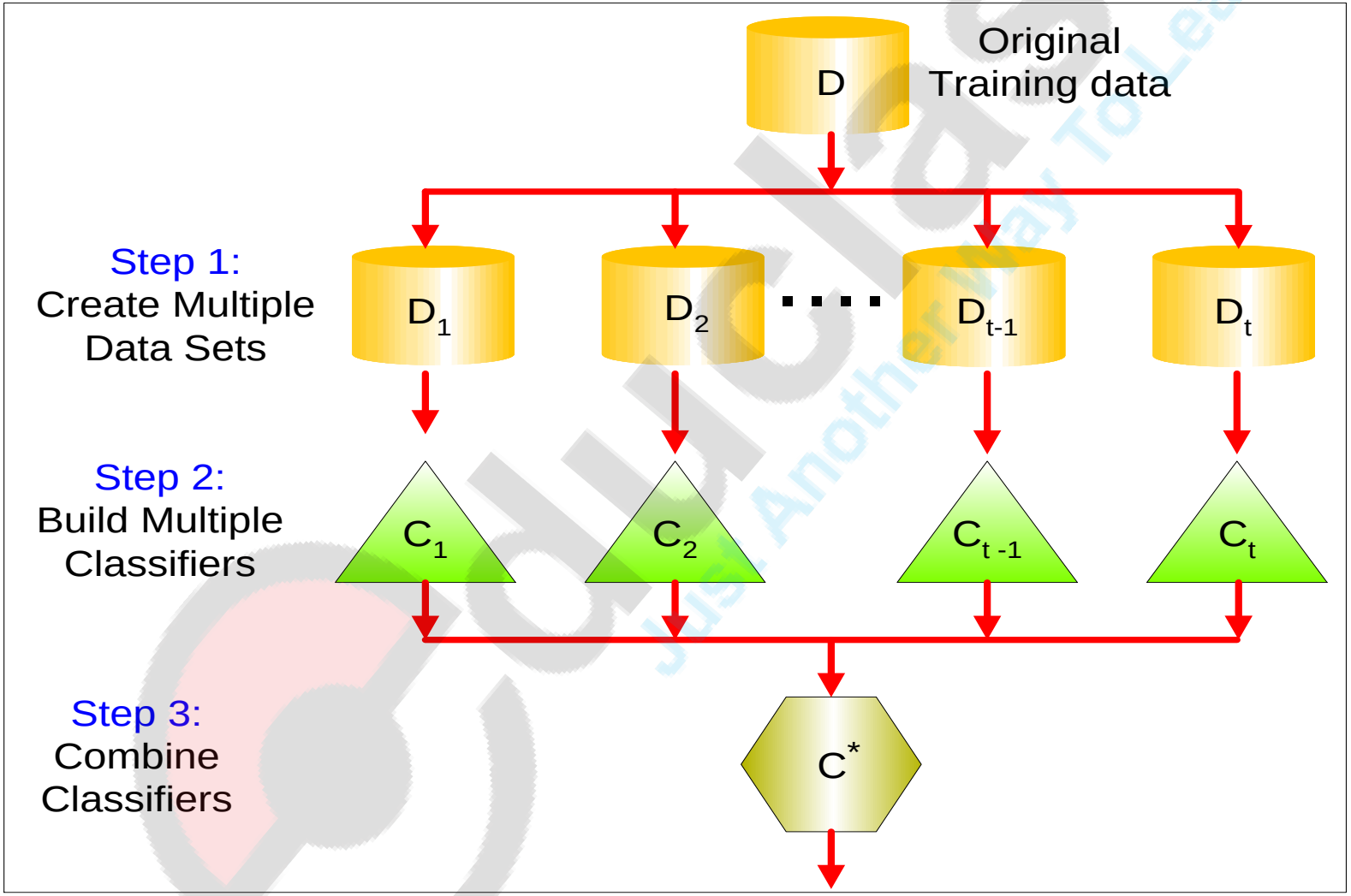
So far – learning methods that learn a single hypothesis, chosen form a hypothesis space that is used to make predictions.

Ensemble learning → select a collection (ensemble) of hypotheses and combine their predictions.

Example 1 - generate 100 different decision trees from the same or different training set and have them vote on the best classification for a new example.

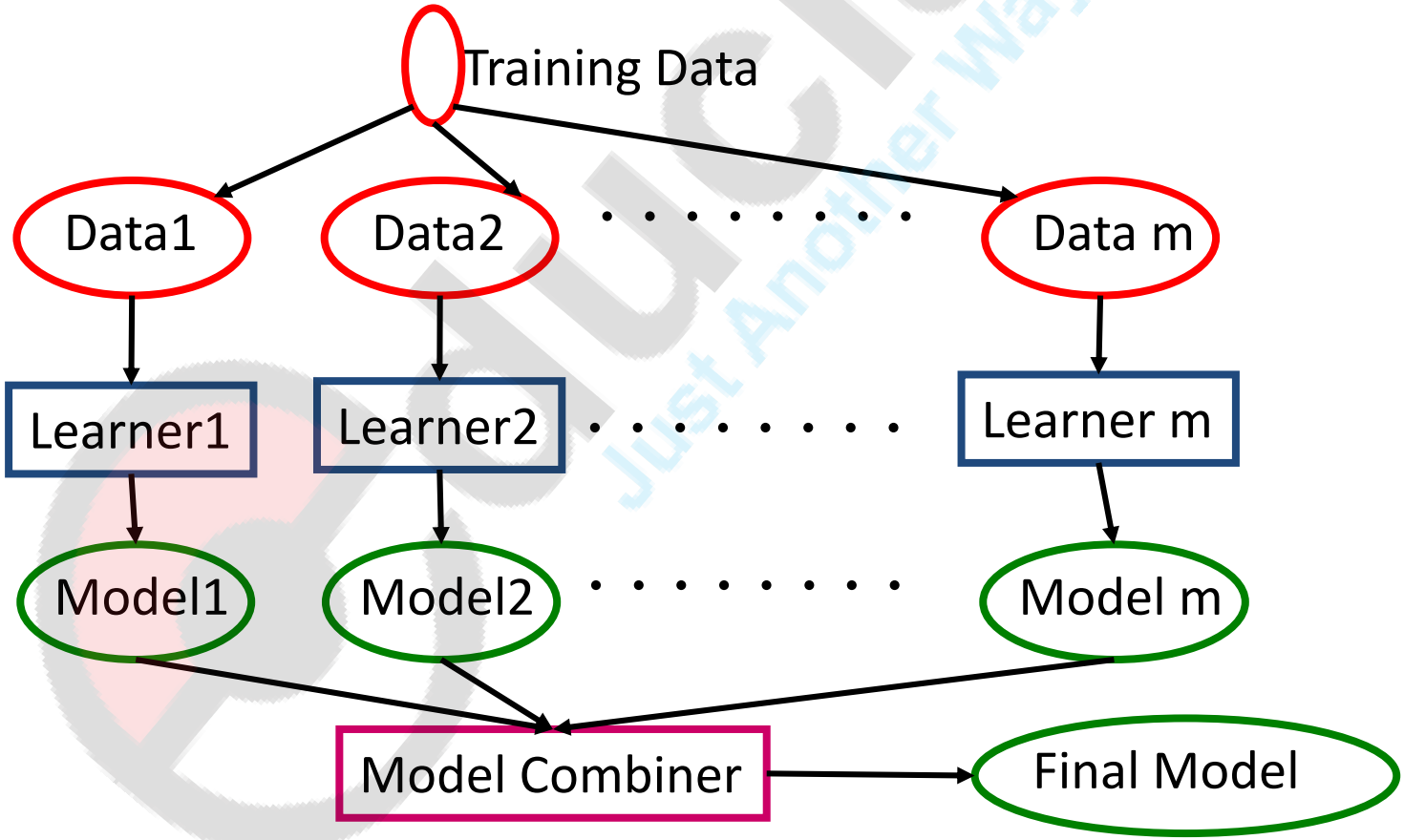
Key motivation: reduce the error rate.

General Idea



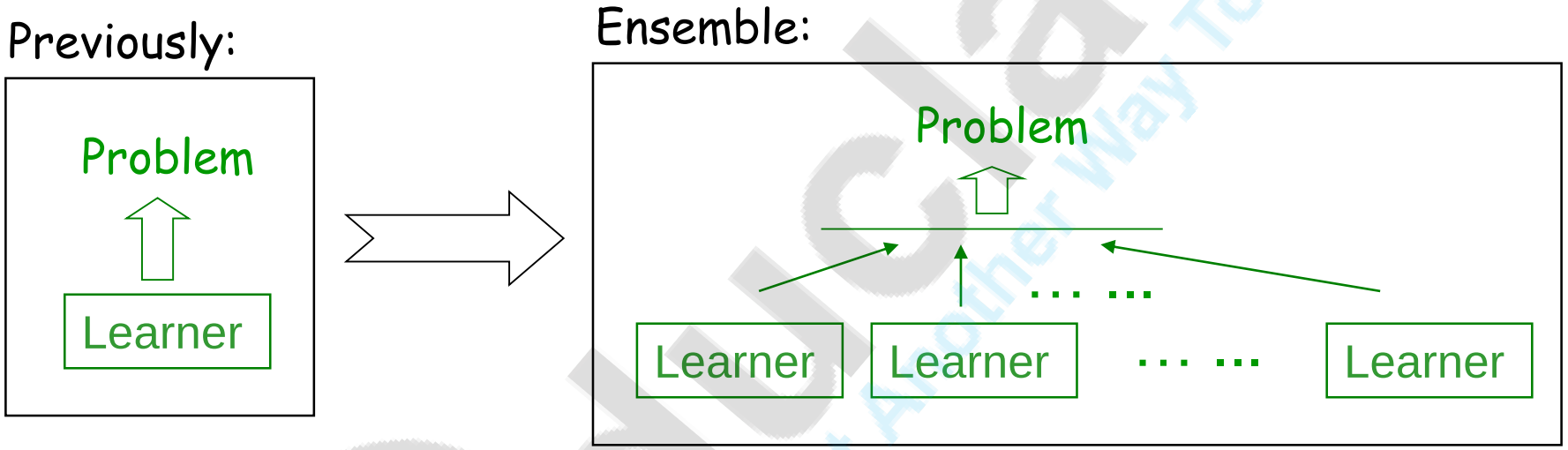
Learning Ensembles

Learn multiple alternative definitions of a concept using different training data or different learning algorithms.
Combine decisions of multiple definitions, e.g. using weighted voting.



Ensemble Learning

A machine learning paradigm where multiple learners are used to solve the problem



The generalization ability of the ensemble is usually significantly better than that of an individual learner

Boosting is one of the most important families of ensemble methods

Examples of Ensemble Methods

How to generate an ensemble of classifiers?

Bagging

Boosting

Bagging

Training Data

Data ID

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

Sampling with replacement

Build classifier on each bootstrap sample
Each sample has probability $(1 - 1/n)^n$ of being selected as test data
Training data = $1- (1 - 1/n)^n$ of the original data

Bagging (applied to training data)

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 Class	1	1	1	-1	-1	-1	-1	1	1	1

Figure 5.36. Example of combining classifiers constructed using the bagging approach.

Accuracy of ensemble classifier: 100% ☺

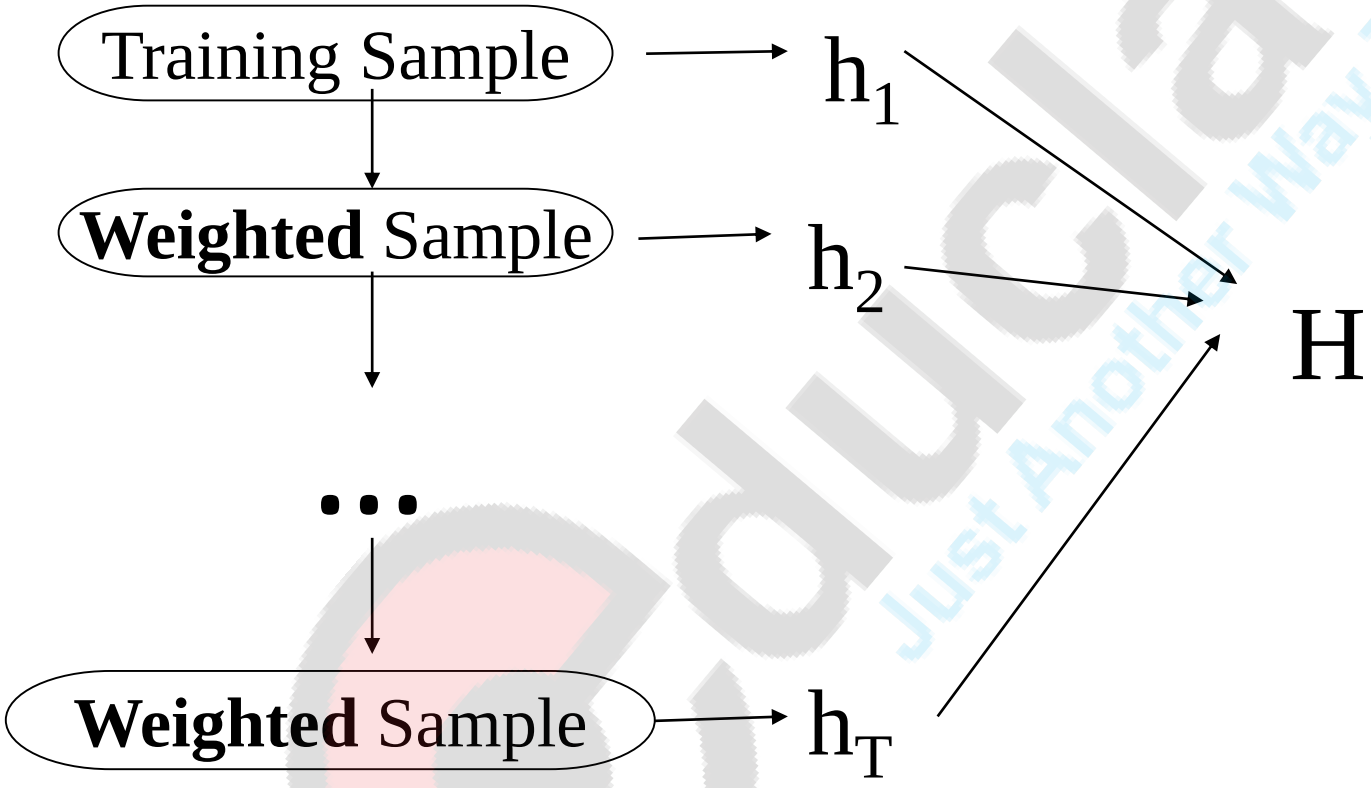
Boosting - definition

A machine learning algorithm
Perform supervised learning
Increments improvement of learned function
Forces the weak learner to generate new hypotheses that make less mistakes on “harder” parts.

Boosting - characteristics

iterative
successive classifiers depends upon its predecessors
look at errors from previous classifier step to decide how to focus on next iteration over data

Boosting



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
- Unlike bagging, weights may change at the end of a boosting round

Boosting

Records that are wrongly classified will have their weights increased
Records that are classified correctly will have their weights decreased

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

Boosting

Equal weights are assigned to each training instance (1/N for round 1) at first round
After a classifier C_i is learned, the weights are adjusted to allow the subsequent classifier C_{i+1} to "pay more attention" to data that were misclassified by C_i.
Final boosted classifier C* combines the votes of each individual classifier

- Weight of each classifier's vote is a function of its accuracy

Adaboost – popular boosting algorithm

AdaBoost is short for *adaptive boosting*.

AdaBoost works this way:

1. A weight is applied to every example in the training data.
2. We'll call the weight vector D. Initially, these weights are all equal.
3. A weak classifier is first trained on the training data. The errors from the weak classifier are calculated, and the weak classifier is trained a second time with the same dataset.
4. This second time the weak classifier is trained, the weights of the training set are adjusted so the examples properly classified the first time are weighted less and the examples incorrectly classified in the first iteration are weighted more.
5. To get one answer from all of these weak classifiers, AdaBoost assigns α values to each of the classifiers.
6. The values are based on the error of each weak classifier.

The error ε is given by

$$\varepsilon = \frac{number\ of\ incorrectly\ classified\ examples}{total\ number\ of\ examples}$$

and α is given by

$$\alpha = \frac{1}{2} ln\left(\frac{1-\varepsilon}{\varepsilon}\right)$$

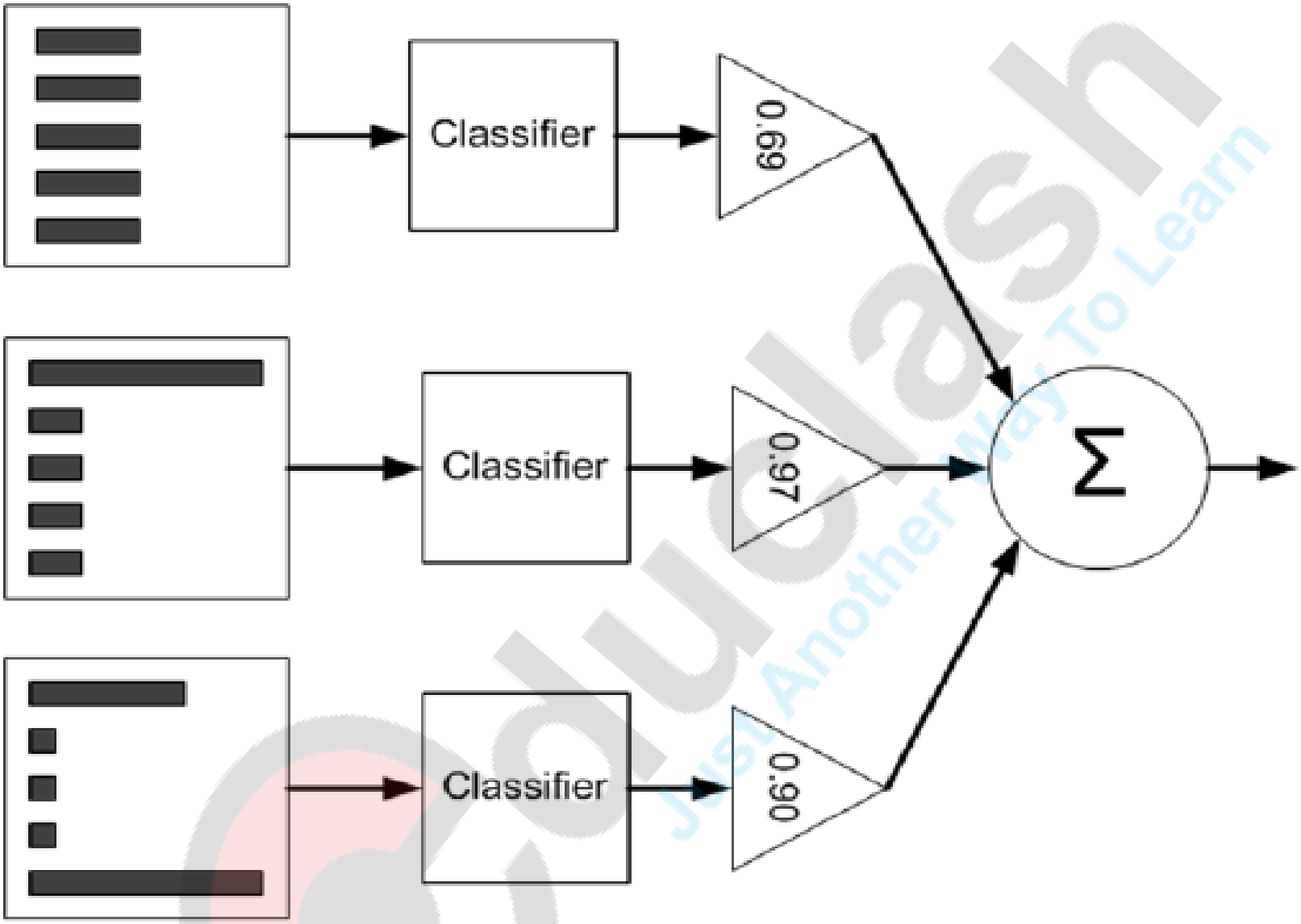


Figure 7.1 Schematic representation of AdaBoost; with the dataset on the left side, the different widths of the bars represent weights applied to each instance. The weighted predictions pass through a classifier, which is then weighted by the triangles (α values). The weighted output of each triangle is summed up in the circle, which produces the final output.

Ada Boost

- Assume 2-class problem, with labels +1 and -1
  - y^i in {-1,1}
- Ada boost produces a discriminant function:

$$g(x) = \sum_{t=1}^{T} \alpha_t h_t(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \ldots \alpha_T h_T(x)$$

- Where $h_t(x)$ is a weak classifier, for example:

$$h_t(x) = \begin{cases} -1 & \text{if email has word "money"} \\ 1 & \text{if email does not have word "money"} \end{cases}$$

- The final classifier is the sign of the discriminant function

$$f_{final}(x) = \text{sign}[g(x)]$$

Idea Behind Ada Boost

- Algorithm is iterative
- Maintains distribution of weights over the training examples
- Initially weights are equal
- Main Idea: at successive iterations, the weight of misclassified examples is increased
- This forces the algorithm to concentrate on examples that have not been classified correctly so far

Idea Behind Ada Boost

- Examples of high weight are shown more often at later rounds
- Face/nonface classification problem:

Round 1

1/7	1/7	1/7	1/7	1/7	1/7	1/7
best weak classifier:	✓	-	✓	✓	=	✓	-
change weights:	1/16	1/4	1/16	1/16	1/4	1/16	1/4

Round 2

best weak classifier:	✓	✓	✓	-	-	-	✓	✓	✓	✓
change weights:		1/8	1/32	11/32		1/2		1/8	1/32	1/32

Idea Behind Ada Boost

Round 3

- out of all available weak classifiers, we choose the one that works best on the data we have at round 3
- we assume there is always a weak classifier better than random (better than 50% error)
-  image is half of the data given to the classifier
- chosen weak classifier **has to** classify this image correctly

More Comments on Ada Boost

- Ada boost is very simple to implement, provided you have an implementation of a "weak learner"
- Will work as long as the "basic" classifier $\mathbf{h}_t(\mathbf{x})$ is at least slightly better than random
  - will work if the error rate of $\boldsymbol{h}_t(\boldsymbol{x})$ is less than 0.5
  - 0.5 is the error rate of a random guessing for a 2-class problem
- Can be applied to boost any classifier, not necessarily weak
  - but there may be no benefits in boosting a "strong" classifier

Ada Boost for 2 Classes

Initialization step: for each example **x**, set

$$D(x) = \frac{1}{N}$$, where **N** is the number of examples

Iteration step (for **t** = 1...T):

1. Find best weak classifier $h_t(x)$ using weights $D(x)$
2. Compute the error rate ε_t as

$$\varepsilon_t = \sum_{i=1}^{N} D(x^i) \cdot [y^i \neq h_t(x^i)]$$

$$= \begin{cases} 1 & \text{if } y^i \neq h_t(x^i) \\ 0 & \text{otherwise} \end{cases}$$

3. compute weight α_t of classifier h_t

$$\alpha_t = \log((1-\varepsilon_t)/\varepsilon_t)$$

4. For each x^i, $D(x^i) = D(x^i) \cdot \exp(\alpha_t \cdot I[y^i \neq h_t(x^i)])$

5. Normalize $D(x^i)$ so that $\sum_{i=1}^{N} D(x^i) = 1$

$$f_{final}(x) = \text{sign}\left[\sum \alpha_t h_t(x)\right]$$

Ada Boost: Step 1

1. Find best weak classifier $h_t(x)$ using weights $D(x)$
   - some classifiers accept weighted samples, but most don't
   - if classifier does not take weighted samples, sample from the training samples according to the distribution $D(x)$

1/16 1/4 1/16 1/16 1/4 1/16 1/4

- Draw **k** samples, each **x** with probability equal to **D(x)**:

re-sampled examples

Ada Boost: Step 1

1. Find best weak classifier $h_t(x)$ using weights $D(x)$

- Give to the classifier the re-sampled examples:

- To find the best weak classifier, go through **all** weak classifiers, and find the one that gives the smallest error on the re-sampled examples

weak classifiers	$h_1(x)$	$h_2(x)$	$h_3(x)$	.....	$h_m(x)$
errors:	0.46	0.36	0.16		0.43

the best classifier $h_t(x)$ to choose at iteration t

Ada Boost: Step 2

2. Compute ε_t the error rate as

$$\varepsilon_t = \sum_{i=1}^{N} D(x^i) \cdot I\left[y^i \neq h_t(x^i)\right] \qquad = \begin{cases} 1 & \text{if } y^i \neq h_t(x^i) \\ 0 & \text{otherwise} \end{cases}$$

1/16 ✓ 1/4 ✓ 1/16 ✓ 1/16 - 1/4 - 1/16 ✓ 1/4 ✓

$$\varepsilon_t = \frac{1}{4} + \frac{1}{16} = \frac{5}{16}$$

- ε_t is the weight of all misclassified examples added
  - the error rate is computed over original examples, not the re-sampled examples
- If a weak classifier is better than random, then $\varepsilon_t < ½$

Ada Boost: Step 3

3. compute weight α_t of classifier h_t

$$\alpha_t = \log((1 - \varepsilon_t)/\varepsilon_t)$$

In example from previous slide:

$$\varepsilon_t = \frac{5}{16} \Rightarrow \alpha_t = \log \frac{1 - \frac{5}{16}}{\frac{5}{16}} = \log \frac{11}{5} \approx 0.8$$

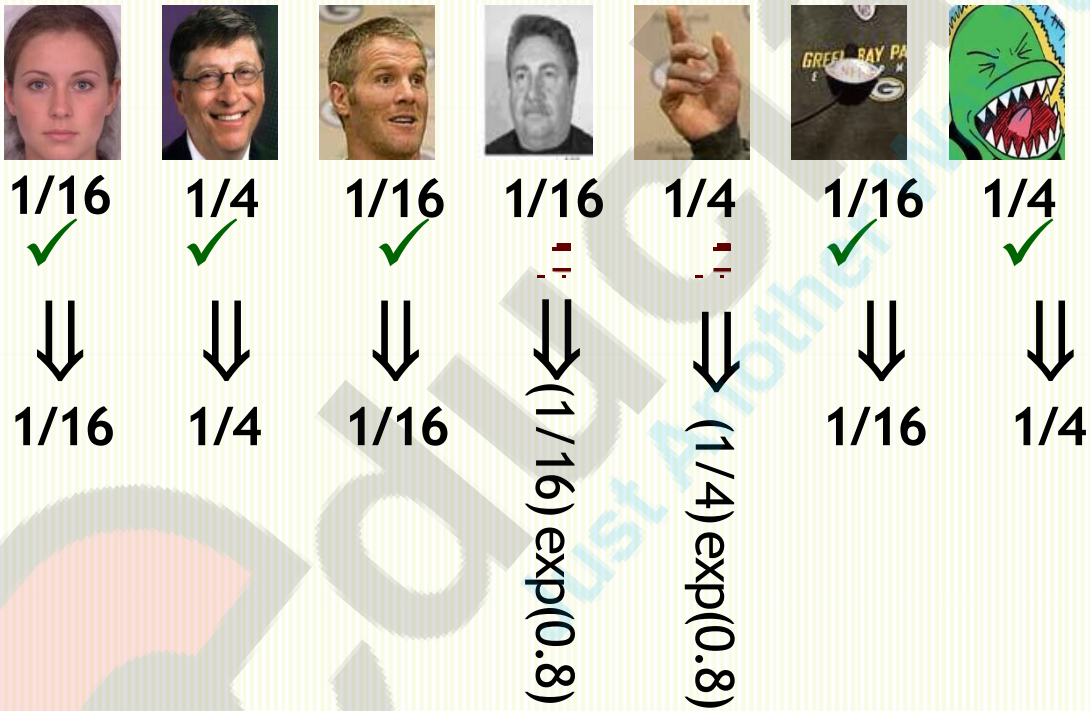
- Recall that $\varepsilon_t < ½$
- Thus $(1 - \varepsilon_t)/\varepsilon_t > 1 \Rightarrow \alpha_t > 0$
- The smaller is ε_t, the larger is α_t, and thus the more importance (weight) classifier $h_t(x)$

$$\text{final}(x) = \text{sign}\left[\sum \alpha_t h_t(x)\right]$$

Ada Boost: Step 4

4. For each x^i, $D(x^i) = D(x^i) \cdot \exp(\alpha_t \cdot I[y^i \neq h_t(x^i)])$

from previous slide $\alpha_t = 0.8$



- weight of misclassified examples is increased

Ada Boost: Step 5

5. Normalize $D(x^i)$ so that $\sum D(x^i) = 1$

from previous slide:

1/16 1/4 1/16 0.14 0.56 1/16 1/4

- after normalization

0.05 0.18 0.05 0.10 0.40 0.05 0.18

- In Matlab, if D is weights vector, normalize with

D = D./sum(D)

Unsupervised Learning

Supervised learning vs. unsupervised learning

Supervised learning: discover patterns in the data that relate data attributes with a target (class) attribute.

These patterns are then utilized to predict the values of the target attribute in future data instances.

Unsupervised learning: The data have no target attribute.

We want to explore the data to find some intrinsic structures in them.

Clustering

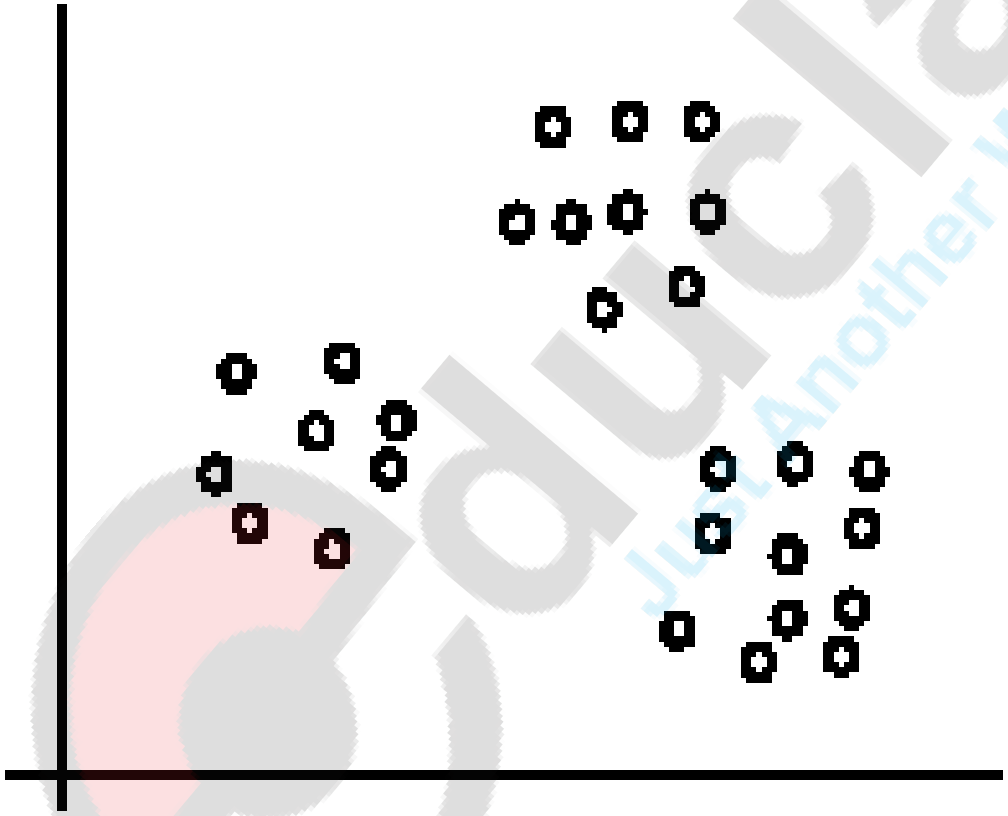
Clustering is a technique for finding similarity groups in data, called **clusters**. I.e.,

- it groups data instances that are similar to (near) each other in one cluster and data instances that are very different (far away) from each other into different clusters.

Clustering is often called an **unsupervised learning** task as no class values denoting an *a priori* grouping of the data instances are given, which is the case in supervised learning.

An illustration

The data set has three natural groups of data points, i.e., 3 natural clusters.



What is clustering for?

Let us see some real-life examples

Example 1: groups people of similar sizes together to make “small”, “medium” and “large” T-Shirts.

- Tailor-made for each person: too expensive
- One-size-fits-all: does not fit all.

Example 2: In marketing, segment customers according to their similarities

- To do targeted marketing.

What is clustering for? (cont…)

Example 3: Given a collection of text documents, we want to organize them according to their content similarities,

To produce a topic hierarchy

In fact, clustering is one of the most utilized data mining techniques.

It has a long history, and used in almost every field, e.g., medicine, psychology, botany, sociology, biology, archeology, marketing, insurance, libraries, etc.

In recent years, due to the rapid increase of online documents, text clustering becomes important.

Clustering: Application Examples

Information retrieval: document clustering
Land use: Identification of areas of similar land use in an earth observation database
Marketing: Help marketers discover distinct groups in their customer bases, and then use this knowledge to develop targeted marketing programs
City-planning: Identifying groups of houses according to their house type, value, and geographical location
Climate: understanding earth climate, find patterns of atmospheric and ocean

Aspects of clustering

A clustering algorithm

- Partitional clustering
- Hierarchical clustering
- …

Types of clustering:

1. **Hierarchical algorithms**: these find successive clusters using previously established clusters.
   1. Agglomerative ("bottom-up"): Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters.
   2. Divisive ("top-down"): Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.
2. **Partitional clustering:** Partitional algorithms determine all clusters at once. They include:

   ***K*-means and derivatives**

   Fuzzy *c*-means clustering

   QT clustering algorithm

K-means clustering

K-means is a partitional clustering algorithm

Let the set of data points (or instances) D be

$$\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\},$$

where $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ir})$ is a vector in a real-valued space $X \subseteq R^r$, and r is the number of attributes (dimensions) in the data.

The *k*-means algorithm partitions the given data into *k* clusters.

Each cluster has a cluster **center**, called **centroid**.

k is specified by the user

The *K-Means* Clustering Method

Given *k*, the *k-means* algorithm is implemented in four steps:

- Partition objects into *k* nonempty subsets
- Compute seed points as the centroids of the clusters of the current partitioning (the centroid is the center, i.e., *mean point*, of the cluster)
- Assign each object to the cluster with the nearest seed point
- Go back to Step 2, stop when the assignment does not change

Using K-means clustering, cluster the following data into two clusters and show each step.

{2, 4, 10, 12, 3, 20, 30, 11, 25}

Solution:

Given: {2, 4, 10, 12, 3, 20, 30, 11, 25}

Step 1: Randomly assign the means: m1 = 3, m2 = 4

Step 2: Group the numbers close to mean m1 = 3 are grouped into cluster k1 and m2 = 4 are grouped into cluster k2

Step 3: k1 = {2, 3}, k2 = {4, 10, 12, 20, 30, 11, 25}, m1= 2.5, m2 = 16

Step 4: k2 = {2, 3, 4}, k2 = {10, 12, 20, 30, 11, 25}, m1= 3 m2 = 18

Step 5: k1 = {2, 3, 4, 10}, k2= {12, 20, 30, 11, 25}, m1= 4.75, m2 = 19.6

Step 6: k1 = {2, 3, 4, 10, 11, 12}, k2 = {20, 30, 25}, m1 = 7, m2 = 25

Step 7: k1 = {2, 3, 4, 10, 11, 12}, k2 = {20, 30, 25}, m1 = 7, m2 = 25

Step 8: Stop. The clusters in step 6 and 7 are same.

Final answer: k1 = {2, 3, 4, 10, 11, 12} and k2 = {20, 30, 25}