Ensemble Learning

Contents

- What is ensemble learning?
- Bagging
- Boosting
- Random Forest
- Different ways to combine classifiers

Ensemble Learning

- Combinations of models are generally known as *ensemble l*earning.
- *the most powerful techniques in machine* learning, often outperforming other methods.
- comes at the cost of increased algorithmic and model complexity.

How to build the model?

- to build an ensemble of slightly different models from the same training data
- key question here is how to achieve diversity between these different models
  - training models on random subsets of the data
  - random subsets of the available features

Ensemble methods in ML

- ensemble methods in machine learning have the following two things:
  - they construct multiple, diverse predictive models from adapted versions of the training data (most often reweighted or resampled)
  - they combine the predictions of these models in some way, often by simple averaging or voting (possibly weighted)

Ensemble Learning

- couple of questions to ask:
- which learners should we use?
- How should we ensure that they learn different things, and how should we combine their results?
- ensemble methods do very well when there is very little data as well as when there is too much.

Ensemble Learning

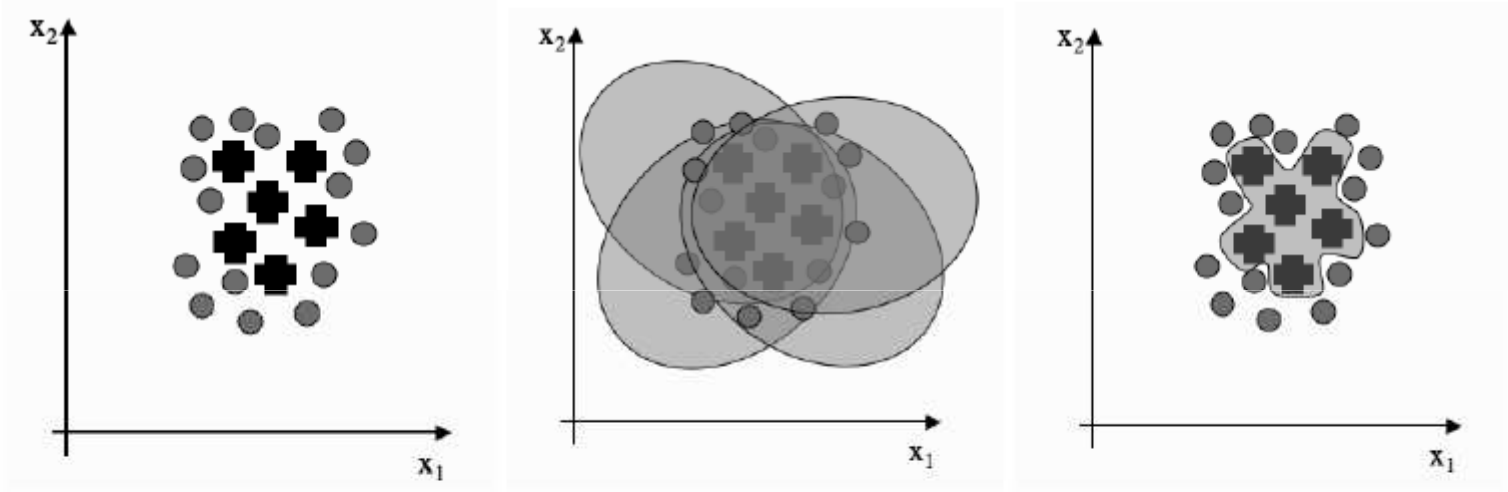
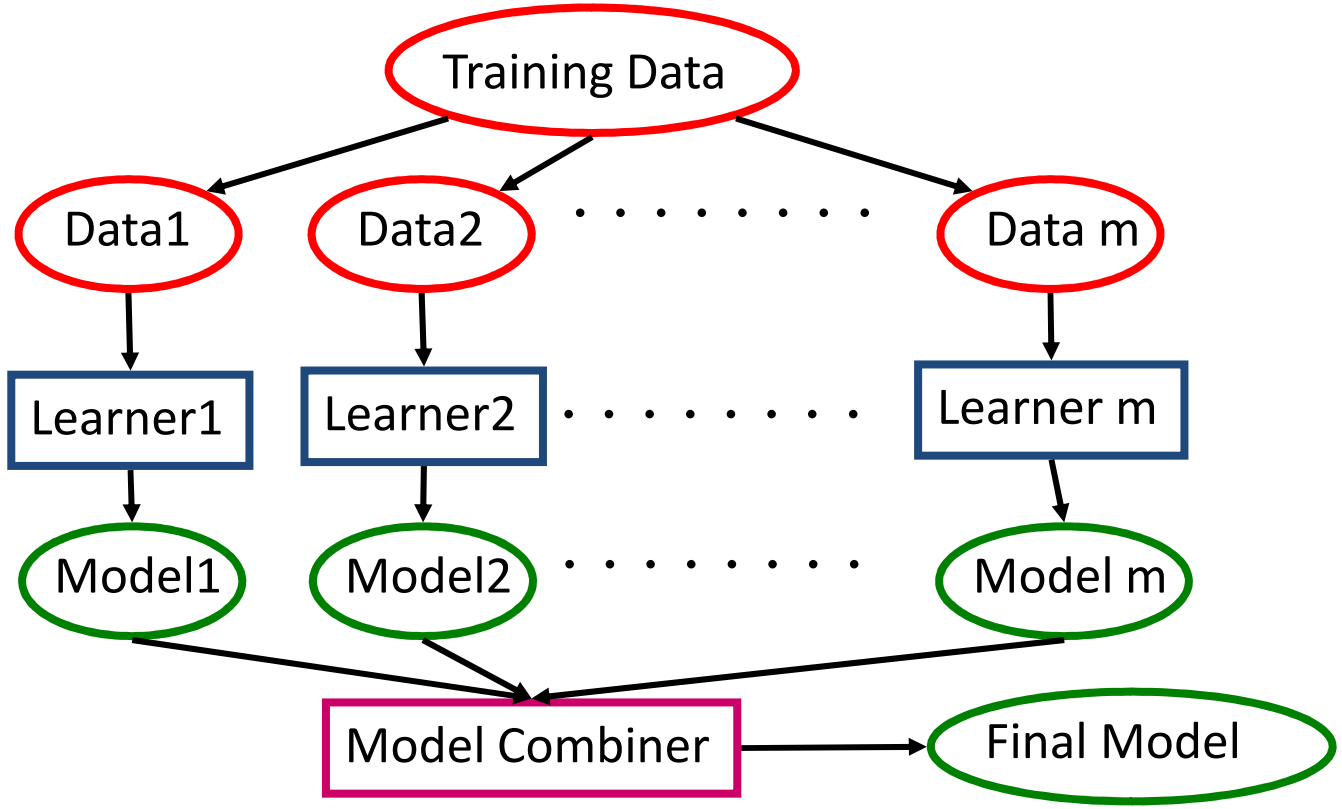


FIGURE 13.1 By combining lots of simple classifiers (here that simply put an elliptical decision boundary onto the data), the decision boundary can be made much more complicated, enabling the difficult separation of the pluses from the circles.

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.



Source: Ray Mooney

Value of Ensembles

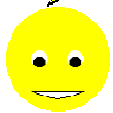
- "No Free Lunch" Theorem
  - No single algorithm wins all the time!

- When combing multiple independent and diverse decisions each of which is at least more accurate than random guessing, random errors cancel each other out, correct decisions are reinforced.

- Examples: Human ensembles are demonstrably better
  - How many jelly beans in the jar?: Individual estimates vs. group average.
  - Who Wants to be a Millionaire: Audience vote.

Source: Ray Mooney

Example: Weather Forecast

Reality							
1							
2							
3							
4							
5							
Combine							

Different Learners

- Different learning algorithms
- Algorithms with different choice for parameters
- Data set with different features
- Data set = different subsets

Homogenous Ensembles

- Use a single, arbitrary learning algorithm but manipulate training data to make it learn multiple models.
  - Data1 ≠ Data2 ≠ … ≠ Data m
  - Learner1 = Learner2 = … = Learner m
- Different methods for changing training data:
  - Bagging: Resample training data
  - Boosting: Reweight training data

Bagging

Bagging

- Bagging, short for ‘bootstrap aggregating’, is a simple but highly effective ensemble method that creates diverse models on different random samples of the original data set.
- These samples are taken uniformly with replacement and are known as *bootstrap samples.*
- bootstrap sample will in general contain duplicates, and hence some of the original data points will be missing even if the bootstrap sample is of the same size as the original data set.
- as differences between the bootstrap samples will create diversity among the models in the ensemble.

Bagging - Aggregate Bootstrapping

- Given a standard training set D of size n
- For i = 1 .. M
  - Draw a sample of size $n^* < n$ from D uniformly and with replacement
  - Learn classifier C_i
- Final classifier is a vote of C_1 .. C_M
- Increases classifier stability/reduces variance

Algorithm

Algorithm Bagging($D, T, \mathscr{A}$)

Input : data set D; ensemble size T; learning algorithm $\mathscr{A}$.

Output : ensemble of models whose predictions are to be combined by voting or averaging.

1 **for** $t = 1$ to T **do**

2 build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;

3 run $\mathscr{A}$ on D_t to produce a model M_t;

4 **end**

5 **return** $\{M_t | 1 \leq t \leq T\}$

Random forest

- Bagging is particularly useful in combination with tree models, which are quite sensitive to variations in the training data.
- When applied to tree models, bagging is often combined with another idea: to build each tree from a different random subset of the features, a process also referred to as *subspace sampling.*
- *This encourages the diversity* in the ensemble even more, and has the additional advantage that the training time of each tree is reduced. The resulting ensemble method is called *random forests.*

Algorithm

Algorithm RandomForest(D, T, d)

Input : data set D; ensemble size T; subspace dimension d.

Output : ensemble of tree models whose predictions are to be combined by voting or averaging.

1 **for** $t = 1$ to T **do**

2 build a bootstrap sample D_t from D by sampling $|D|$ data points with replacement;

3 select d features at random and reduce dimensionality of D_t accordingly;

4 train a tree model M_t on D_t without pruning;

5 **end**

6 **return** $\{M_t | 1 \leq t \leq T\}$

Random Forest Training Algorithm

- For each of N trees:
  - create a new bootstrap sample of the training set
  - use this bootstrap sample to train a decision tree
  - at each node of the decision tree, randomly select m features, and compute the information gain (or Gini impurity) only on that set of features, selecting the optimal one
  - repeat until the tree is complete

Boosting

- Boosting is an ensemble technique that is superficially similar to bagging
- Simple and appealing
- uses a more sophisticated technique than bootstrap sampling to create diverse training sets
- train a linear classifier on a data set and find that its training error rate is e.

Methods

- another classifier to the ensemble that does better on the misclassifications of the first classifier.
  - One way to do that is to duplicate the misclassified instances: if our base model is the basic linear classifier, this will shift the class means towards the duplicated instances.
  - A better way to achieve is to give the misclassified instances a higher weight, and to modify the classifier to take these weights into account (e.g., the basic linear classifier can calculate the class means as a weighted average)

AdaBoost

- Initialise all weights to $1/N$, where N is the number of datapoints
- While $0 < \epsilon_t < \frac{1}{2}$ (and $t < T$, some maximum number of iterations):
  - train classifier on $\{S, w^{(t)}\}$, getting hypotheses $h_t(x_n)$ for datapoints x_n
  - compute training error $\epsilon_t = \sum_{n=1}^{N} w_n^{(t)} I(y_n \neq h_t(x_n))$
  - set $\alpha_t = \log\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
  - update weights using:

$$w_n^{(t+1)} = w_n^{(t)} \exp(\alpha_t I(y_n \neq h_t(x_n))/Z_t, \tag{13.1}$$

    where Z_t is a normalisation constant
- Output $f(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$

Example

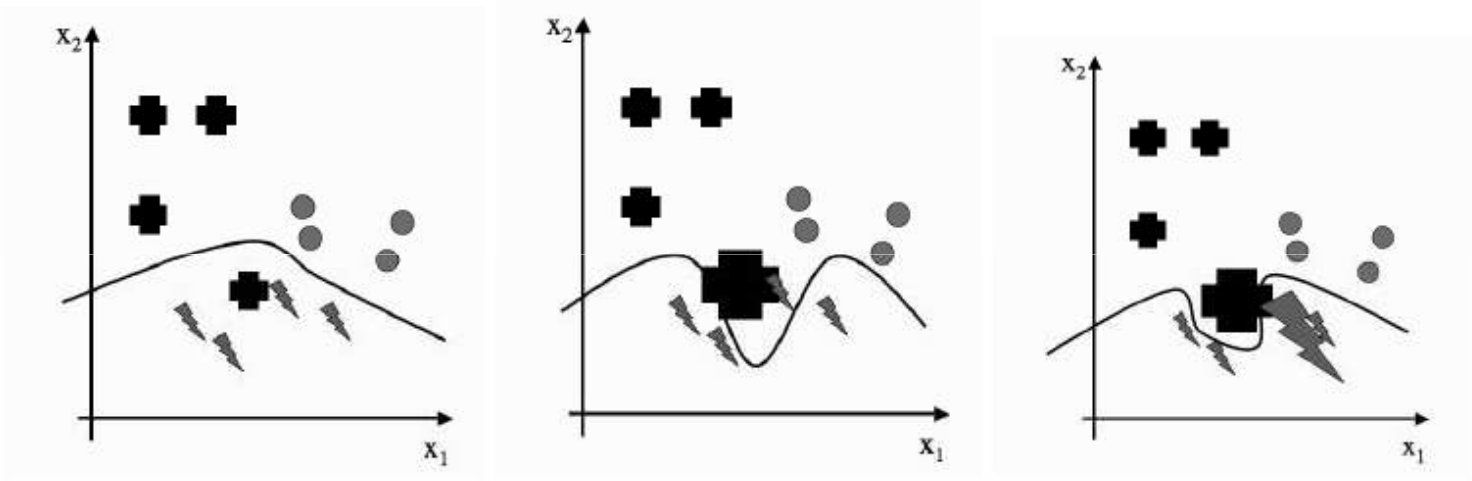


FIGURE 13.2 As points are misclassified, so their weights increase in boosting (shown by the datapoint getting larger), which makes the importance of those datapoints increase, making the classifiers pay more attention to them.

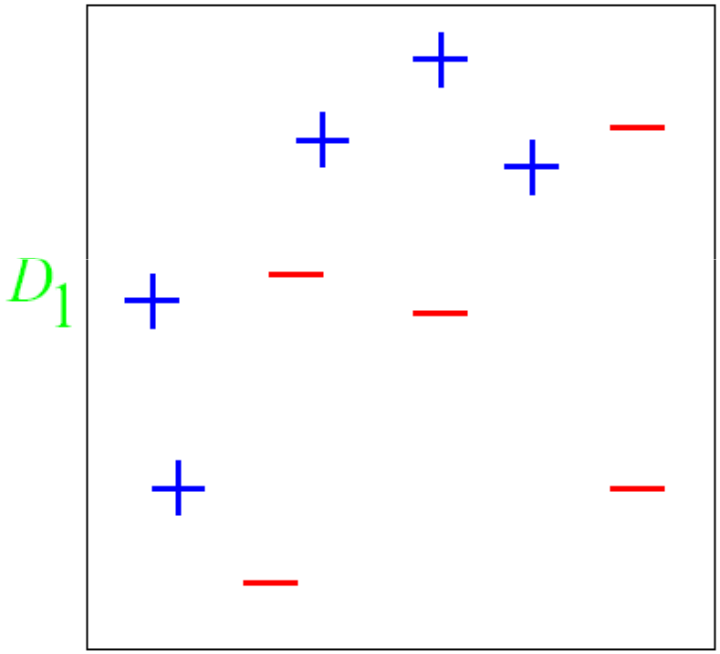
Weights change?

- The idea is that half of the total weight is assigned to the misclassified examples, and the other half to the rest.
- Start with uniform weights that sum to 1
- the current weight assigned to the misclassified examples is exactly the error rate e. So, multiply their weights by 1/2e. Assuming $e < 0.5$ - increase in weight as desired.
- the weights of the correctly classified examples get multiplied by 1/2(1−*e), so the adjusted weights again sum to 1.*
- *In the next round* we do exactly the same, except we take the non-uniform weights into account when evaluating the error rate

Contd…

- one more ingredient in our boosting algorithm and that is a confidence factor *α for each model in the ensemble*
- *α to increase while* decreasing e

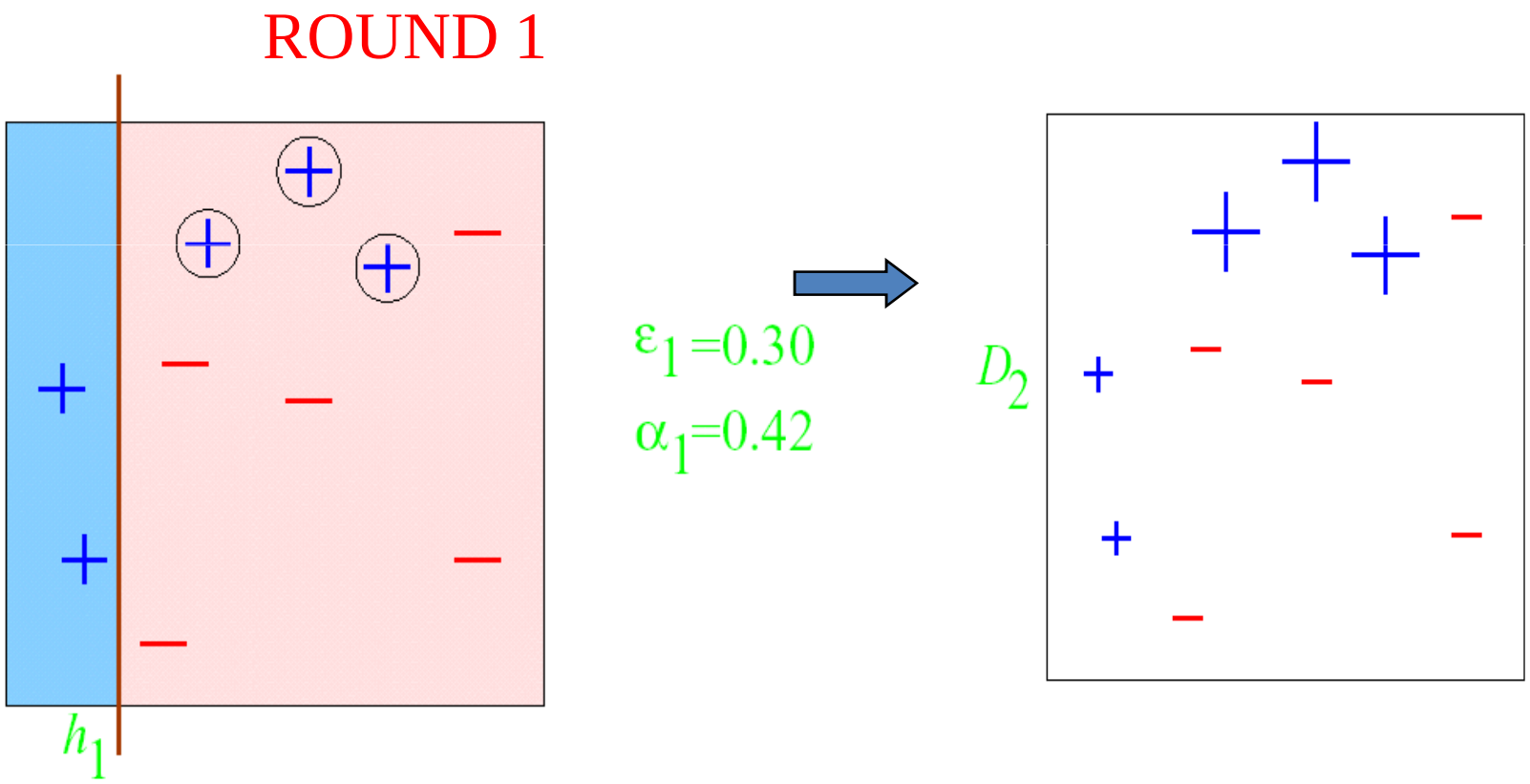
AdaBoost(Example)



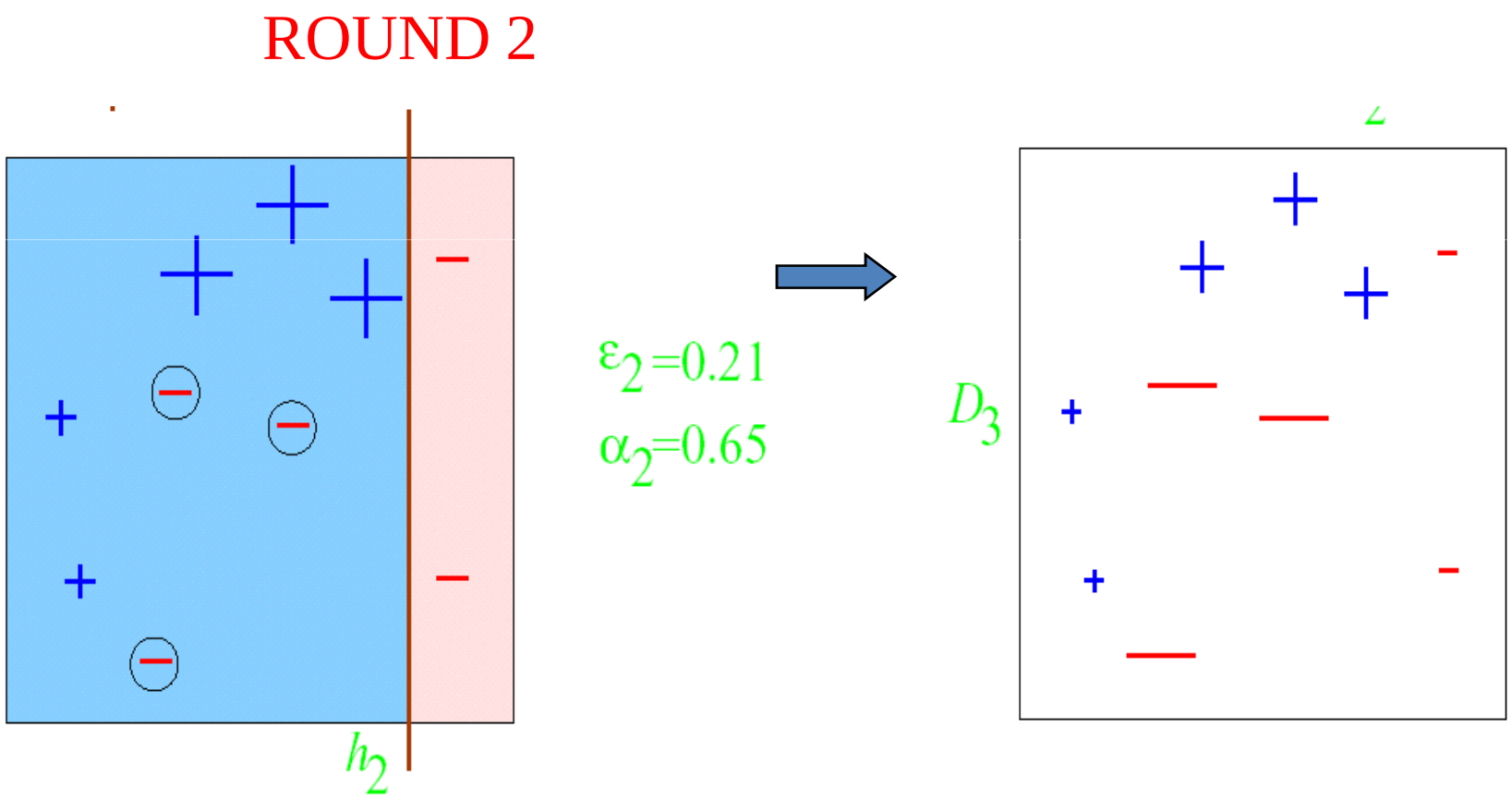
Original Training set : Equal Weights to all training samples

Taken from "**A Tutorial on Boosting" by** Yoav Freund and Rob Schapire

AdaBoost(Example)

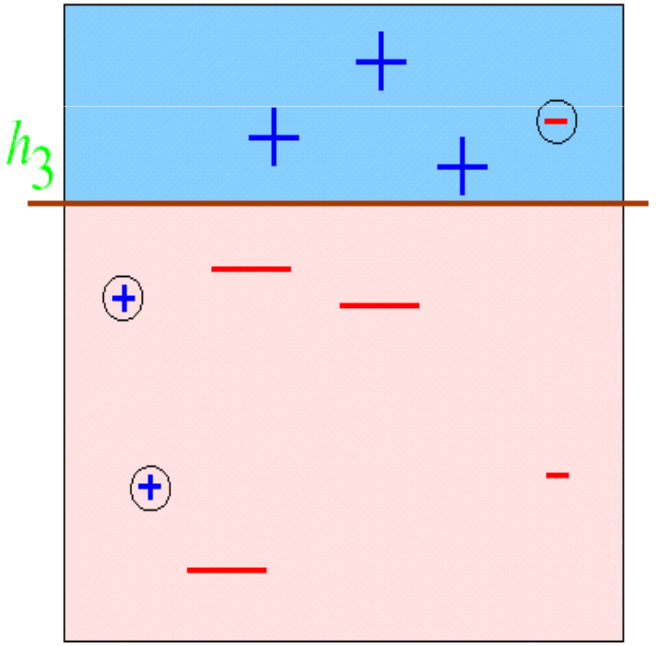


AdaBoost(Example)



AdaBoost(Example)

ROUND 3



$\varepsilon_3 = 0.14$

$\alpha_3 = 0.92$

AdaBoost(Example)

H_{final}