

CS 412

MARCH 5TH – ENSEMBLE METHODS

New Dataset

<https://archive.ics.uci.edu/ml/datasets/Blood+Transfusion+Service+Center>

After some hemming and hawing, I've decided to give you a blood donation predictor

Using UCI



Blood Transfusion Service Center Data Set

Download: [Data Folder](#), [Data Set Description](#)

Abstract: Data taken from the Blood Transfusion Service Center in Hsin-Chu City in Taiwan -- this is a classification problem.

Data Set Characteristics:	Multivariate	Number of Instances:	748	Area:	Business
Attribute Characteristics:	Real	Number of Attributes:	5	Date Donated	2008-10-03
Associated Tasks:	Classification	Missing Values?	N/A	Number of Web Hits:	193049

Using UCI

I want you to get practice with finding and downloading the UCI sets since a lot of you will be using one of them for your final project

For this assignment, the most important things I was looking for were:

- Task: Classification
- Attribute characteristics: Real
- Dataset characteristics: Multivariate
- Missing values: N/A
- Number of instances: 748

There are selection parameters to help with this when you go to the UCI site

Default Task - Undo
Classification (199)
Regression (79)
Clustering (51)
Other (8)
Attribute Type - Undo
Categorical (25)
Numerical (199)
Mixed (37)
Data Type - Undo
Multivariate (199)
Univariate (17)
Sequential (31)
Time-Series (59)
Text (23)
Domain-Theory (7)
Other (5)
Area
Life Sciences (50)
Physical Sciences (22)
CS / Engineering (87)
Social Sciences (4)
Business (13)
Game (0)
Other (21)
Attributes
Less than 10 (33)
10 to 100 (105)
Greater than 100 (50)
Instances
Less than 100 (7)
100 to 1000 (69)
Greater than 1000 (119)
Format Type
Matrix (157)
Non-Matrix (42)

Using UCI

Next, you'll want to look at the [data set description](#).

- This will describe how the data was collected, where it is from, which papers have used the data and most importantly, what the features are (and their summary statistics)

Attribute Information:

Given is the variable name, variable type, the measurement unit and a brief description. The "Blood Transfusion Service Center" is a classification problem. The order of this listing corresponds to the order of numerals along the rows of the database.

R (Recency - months since last donation),
F (Frequency - total number of donation),
M (Monetary - total blood donated in c.c.),
T (Time - months since first donation), and
a binary variable representing whether he/she donated blood in March 2007 (1 stand for donating blood; 0 stands for not donating blood).

Table 1 shows the descriptive statistics of the data. We selected 500 data at random as the training set, and the rest 248 as the testing set.

Table 1. Descriptive statistics of the data

Variable	Data	Type	Measurement	Description	min	max	mean	std
Recency	quantitative	Months	Input	0.03 74.4	9.74	8.07		
Frequency	quantitative	Times	Input	1 50	5.51	5.84		
Monetary	quantitative	c.c. blood	Input	250 12500	1378.68	1459.83		
Time	quantitative	Months	Input	2.27 98.3	34.42	24.32		
Whether he/she donated blood in March 2007	binary	1=yes 0=no	Output	0 1 1 (24%) 0 (76%)				

Using UCI

For this assignment, you may use all of the data points for your training and testing

Accuracy for this data will also be high, but should not be 100%
(as you have been getting with the digits data)

The data comes as a .csv **with a header**, so be prepared for that.

The prediction variable is the last column

Combining predictors

Given 3 predictors that are 70% correct:

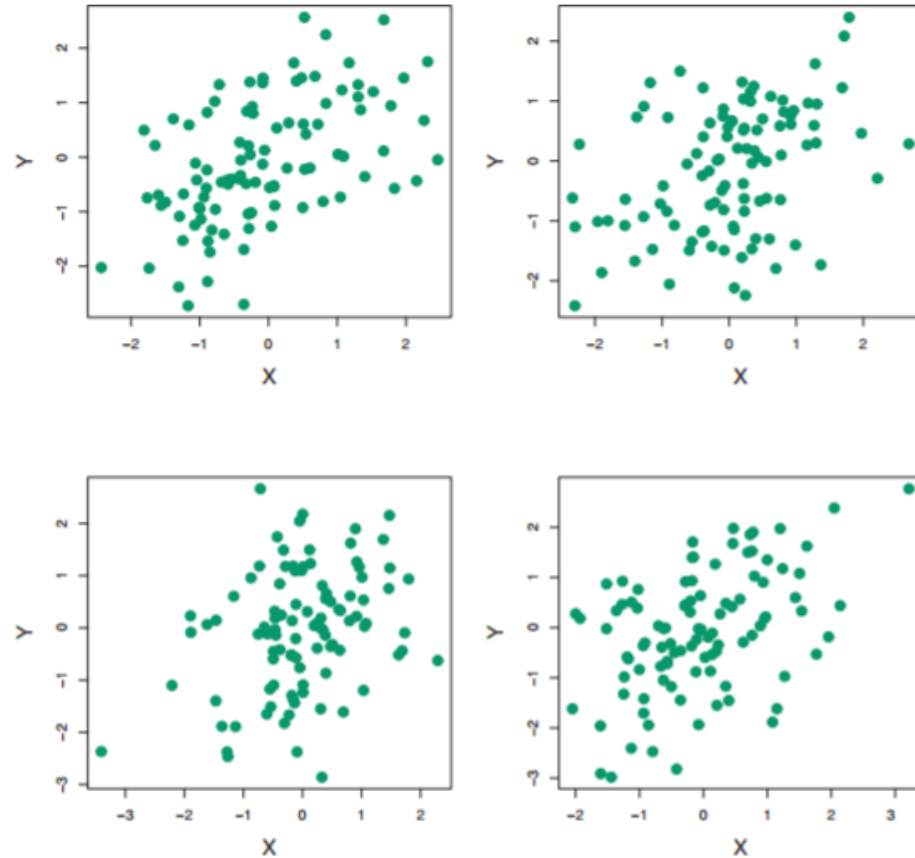
- can we obtain better predictions by combining?

If independent, what is probability that at least 2 are wrong?

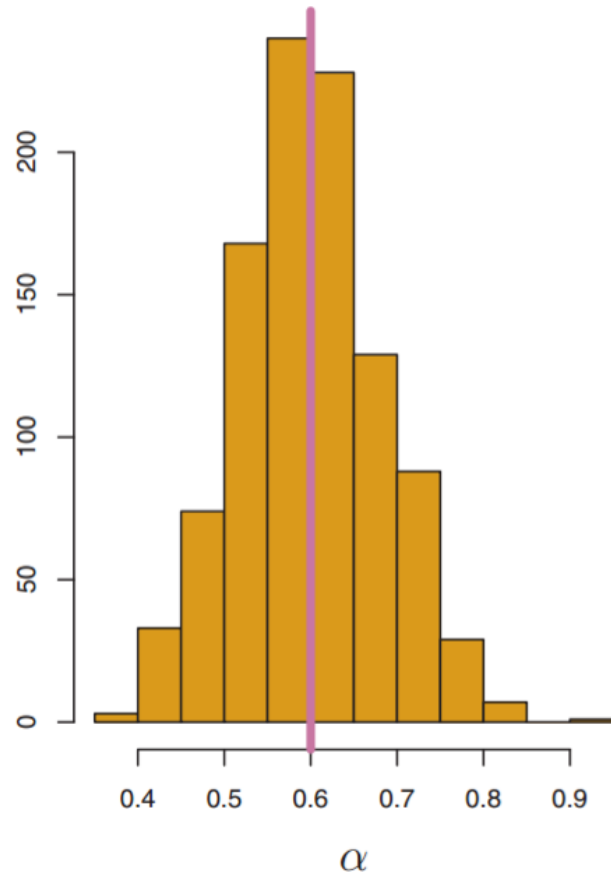
- Improved from 70% → 78.4%

$$1 * 0.3^3 + 3 * 0.3^2 * 0.7 = 0.216$$

Motivating Scenario



Motivating Scenario



If we repeat 1,000 times, this is a histogram of our estimate of α (and true is pink line)

But this required 100,000 samples!

What should our certainty be using only 1,000 samples?

Solutions

What are some possible solutions to this problem?

Solutions

One possibility: get several samples from F .

Problem: it is impossible (or too expensive) to get multiple samples.

Solution: bootstrap

The general bootstrap algorithm

Let the original sample be $L=(x_1,x_2,\dots,x_n)$

Repeat B times:

- Generate a sample L_k of size n from L by sampling with replacement.
- Compute $\hat{\theta}^*$ for x^* .

→ Now we end up with bootstrap values

$$\hat{\theta}^* = (\hat{\theta}_1^*, \dots, \hat{\theta}_B^*)$$

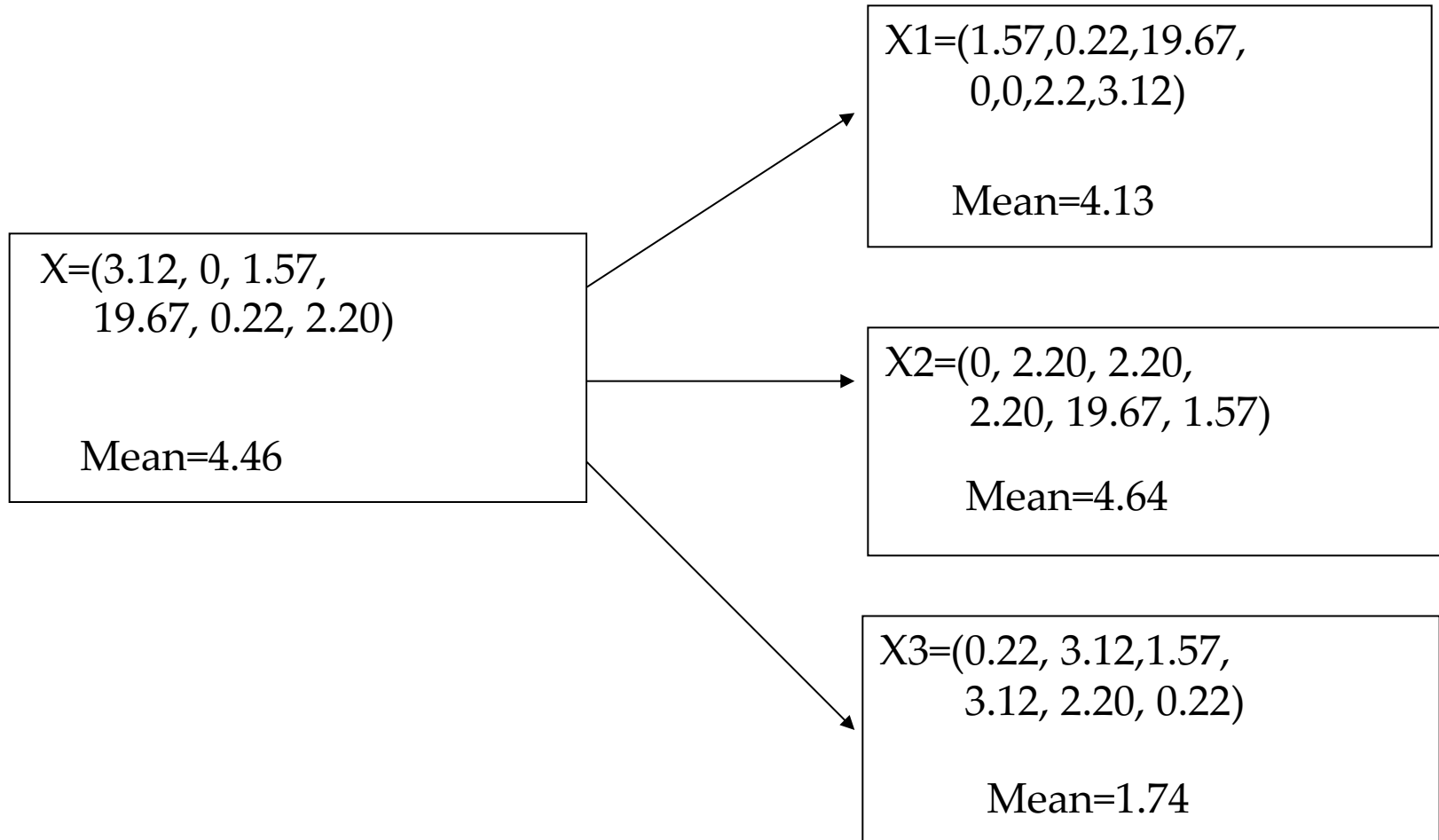
Use these values for calculating all the quantities of interest (e.g., standard deviation, confidence intervals)

An example

$X=(3.12, 0, 1.57,$
 $19.67, 0.22, 2.20)$

Mean=4.46

An example



Bootstrap distribution

The bootstrap does not replace or add to the original data.

We use bootstrap distribution as a way to **estimate the variation in a statistic** based on the original data.

Sampling distribution vs. bootstrap distribution

The population: certain unknown quantities of interest (e.g., mean)

Multiple samples → sampling distribution

Bootstrapping:

- One original sample → B bootstrap samples
- B bootstrap samples → bootstrap distribution

Bootstrap Distribution

Bootstrap distributions usually approximate the shape, spread, and bias of the actual sampling distribution.

Bootstrap distributions are centered at the value of the statistic from the original sample plus any bias.

The sampling distribution is centered at the value of the parameter in the population, plus any bias.

Cases where bootstrap does not apply

Small data sets: the original sample is not a good approximation of the population

Dirty data: outliers add variability in our estimates.

Dependence structures (e.g., time series, spatial problems): Bootstrap is based on the assumption of independence.

...

How many bootstrap samples are needed?

Choice of B depends on

Computer availability

Type of the problem: standard errors, confidence intervals, ...

Complexity of the problem

Bagging

Introduced by Breiman (1996)

“Bagging” stands for “bootstrap aggregating”.

It is an ensemble method: a method of combining multiple predictors.

Predictors

Let L be a training set $\{(x_i, y_i) \mid x_i \text{ in } X, y_i \text{ in } Y\}$, drawn from the set Λ of possible training sets.

A predictor $\Phi: X \rightarrow Y$ is a function that for any given x , it produces $y = \Phi(x)$.

A learning algorithm $\Psi: \Lambda \rightarrow \Phi$ that given any L in Λ , it produces a predictor $\Phi = \Psi(L)$ in Φ .

Types of predictors:

- Classifiers: DTs, DLs, TBLs, ...
- Estimators: Regression trees
- Others: parsers

Bagging algorithm

Let the original training data be L

Repeat B times:

- Get a bootstrap sample L_k from L .
- Train a predictor using L_k .

Combine B predictors by

- Voting (for classification problem)
- Averaging (for estimation problem)
- ...

Bagging results

Table 1 Missclassification Rates (Percent)

Data Set	\bar{e}_S	\bar{e}_B	Decrease
waveform	29.0	19.4	33%
heart	10.0	5.3	47%
breast cancer	6.0	4.2	30%
ionosphere	11.2	8.6	23%
diabetes	23.4	18.8	20%
glass	32.0	24.9	22%
soybean	14.5	10.6	27%

1. Splitting the data set into training set T1 and test set T2.
2. Bagging using 50 bootstrap samples.
3. Repeat Steps 1-2 100 times, and calculate average test set misclassification rate.

Bagging k-nearest neighbor classifiers

Missclassification Rates for Nearest Neighbor

Data Set	\bar{e}_S	\bar{e}_B
waveform	26.1	26.1
heart	6.3	6.3
breast cancer	4.9	4.9
ionosphere	35.7	35.7
diabetes	16.4	16.4
glass	21.6	21.6

100 bootstrap samples. 100 iterations.
Bagging does not help.

Experiment results

Bagging works well for “unstable” learning algorithms.

Bagging can slightly degrade the performance of “stable” learning algorithms.

*What do we think we mean by a **stable** predictor?*

Learning algorithms

Unstable learning algorithms: small changes in the training set result in large changes in predictions.

- Neural network
- Decision tree
- Regression tree
- Subset selection in linear regression

Stable learning algorithms:

- K-nearest neighbors

Rationale

No Free Lunch Theorem: There is no algorithm that is always the most accurate

Generate a group of base-learners which when combined has higher accuracy

Different learners use different

- Algorithms
- Hyperparameters
- Representations /Modalities/Views
- Training sets
- Subproblems

Diversity vs accuracy

- How to generate base learners and how to combine them

Voting

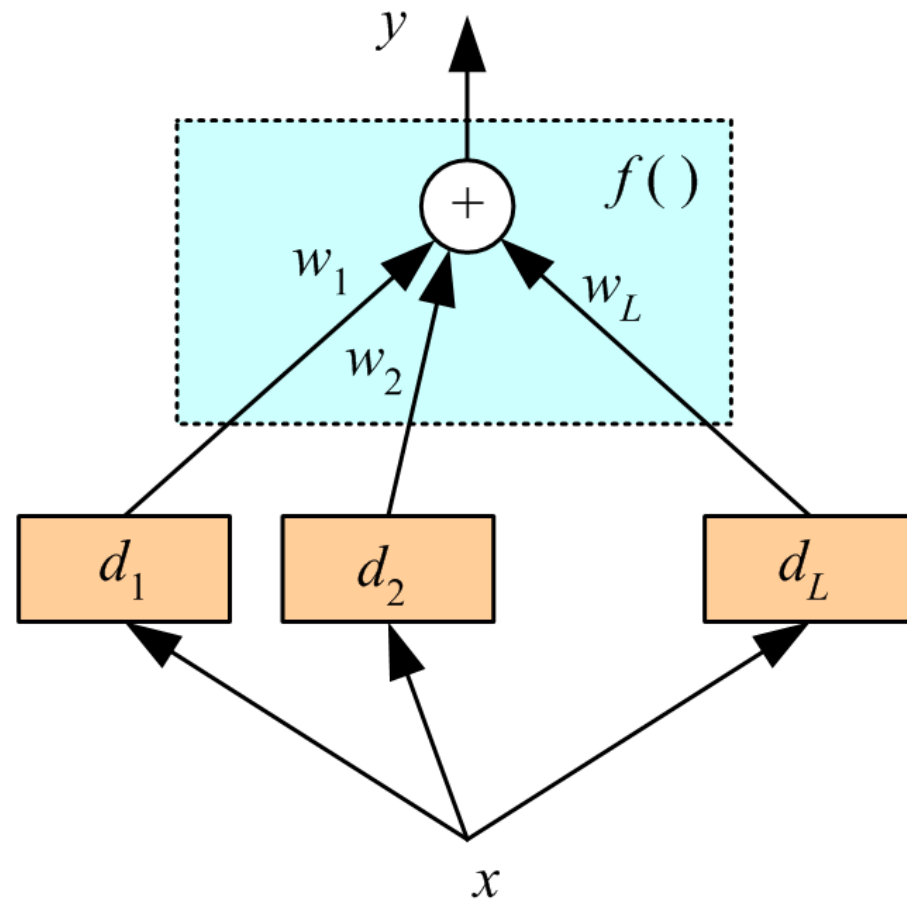
Linear combination

$$y = \sum_{j=1}^L w_j d_j$$

$$w_j \geq 0 \text{ and } \sum_{j=1}^L w_j = 1$$

Multi-class classification

$$y_i = \sum_{j=1}^L w_j d_{ji}$$



Bagging (Bootstrap aggregating)

Take M bootstrap samples (with replacement)

Train M different classifiers on these bootstrap samples

For a new query, let all classifiers predict and take an average (or majority vote)

If the classifiers make independent errors, then their ensemble can improve performance.

- This is reduced with “stable” learners

Stated differently: the variance in the prediction is reduced (we don't suffer from the random errors that a single classifier is bound to make).

Boosting

Boosting works differently.

1. Boosting does not involve bootstrap sampling
2. Decision models are grown sequentially: each model is created using information from previously grown trees
3. Like bagging, boosting involves combining a large number of models, f^1, \dots, f^B

Boosting

Train classifiers (e.g. decision trees) in a sequence.

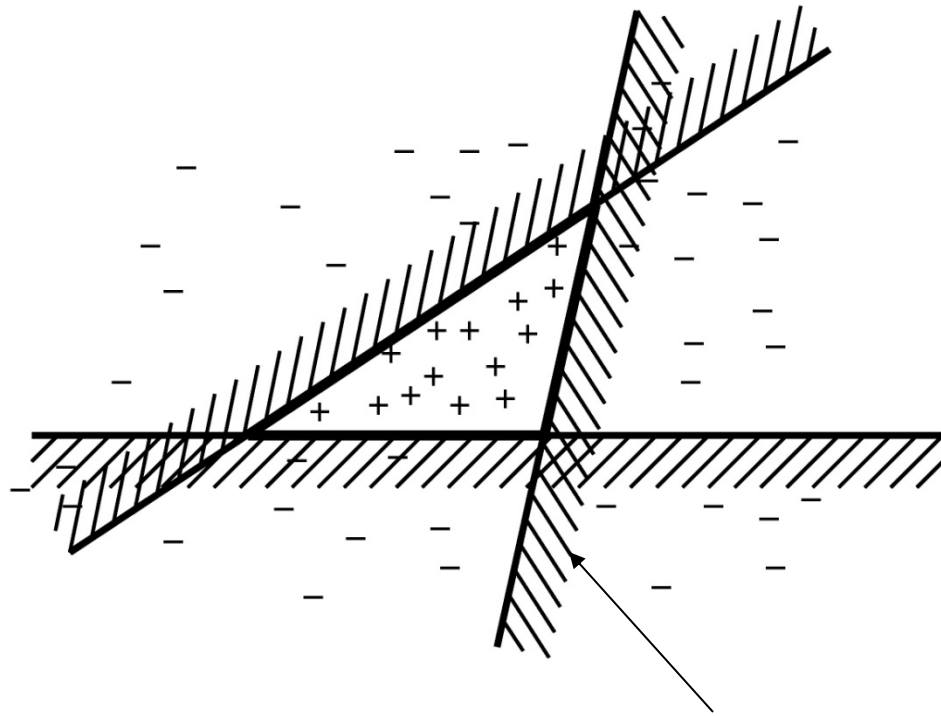
A new classifier should focus on those cases which were incorrectly classified in the last round.

Combine the classifiers by letting them vote on the final prediction (like bagging).

Each classifier is “weak” but the ensemble is “strong.”

AdaBoost is a specific boosting method.

Example



This line is one simple classifier saying that everything to the left + and everything to the right is -

Boosting Intuition

We adaptively weigh each data case.

Data cases which are wrongly classified get high weight (the algorithm will focus on them)

Each boosting round learns a new (simple) classifier on the weighed dataset.

These classifiers are weighed to combine them into a single powerful classifier.

Classifiers that obtain low training error rate have high weight.

We stop by using monitoring a hold out set (cross-validation).

Boosting Intuition

Why do we want to have

1. Weak learners?
2. Learners trained on different data?

Boosting Intuition

Why do we want to have

1. Weak learners?
2. Learners trained on different data?

We want the trainers to be as independent as possible

Boosting

Train a set of weak hypotheses: h_1, \dots, h_T .

The combined hypothesis H is a weighted majority vote of the T weak hypotheses.

- Each hypothesis h_t has a weight α_t .

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

During the training, focus on the examples that are misclassified.

- ➔ At round t , example x_i has the weight $D_t(i)$.

Boosting

Binary classification problem

Training data:

$(x_1, y_1), \dots, (x_m, y_m)$, where $x_i \in X, y_i \in Y = \{-1, 1\}$

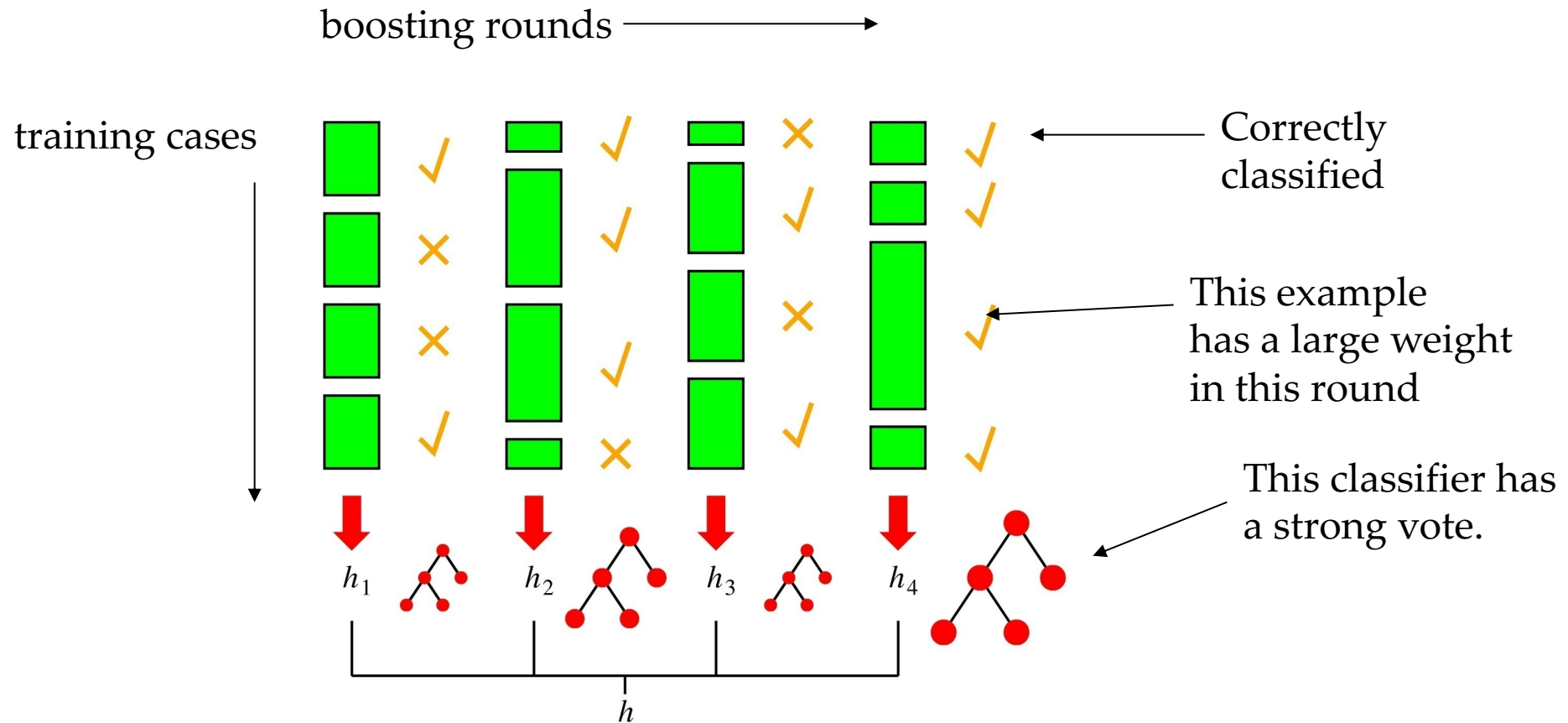
$D_t(i)$: the weight of x_i at round t . $D_1(i) = 1/m$.

A learner L that finds a weak hypothesis $h_t: X \rightarrow Y$ given the training set and D_t

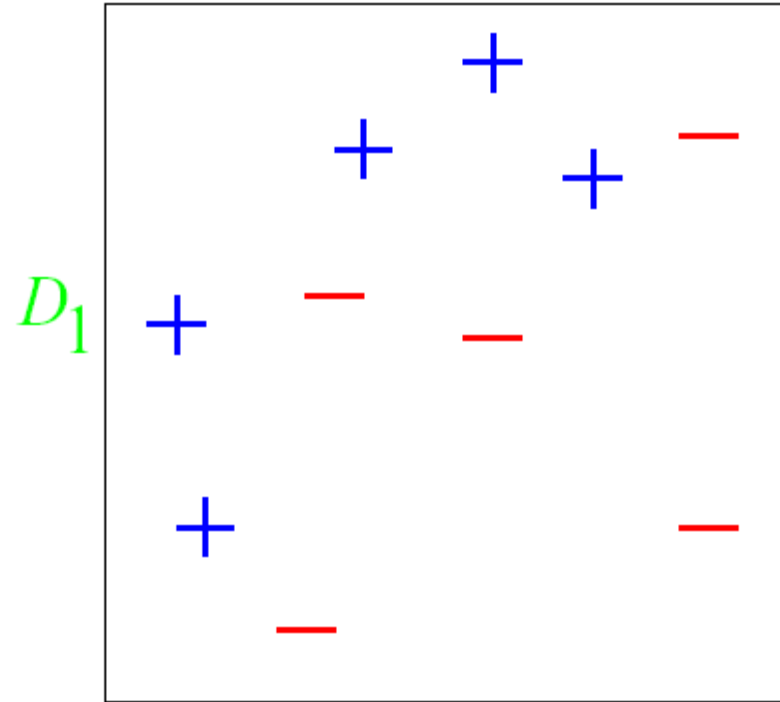
The error of a weak hypothesis h_t :

$$\varepsilon_t = \Pr_{i \sim D_t} [h_t(x_i) \neq y_i] = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$$

Boosting in a Picture



And in animation



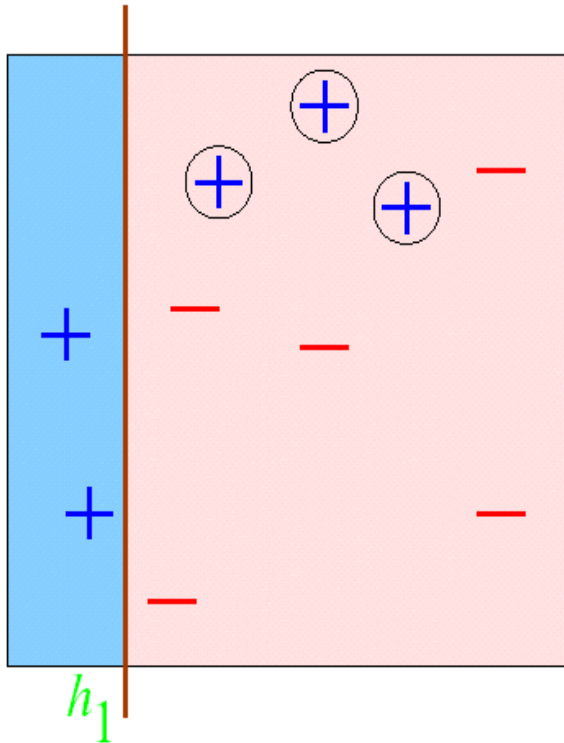
Original training set: equal weights to all training samples

Boost example

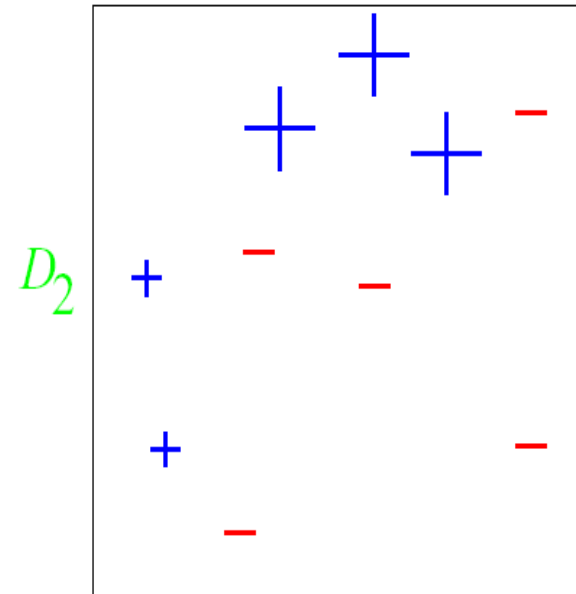
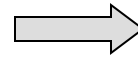
ROUND 1

ε = error rate of classifier

α = weight of classifier e.g. $[\varepsilon/(1-\varepsilon)]$

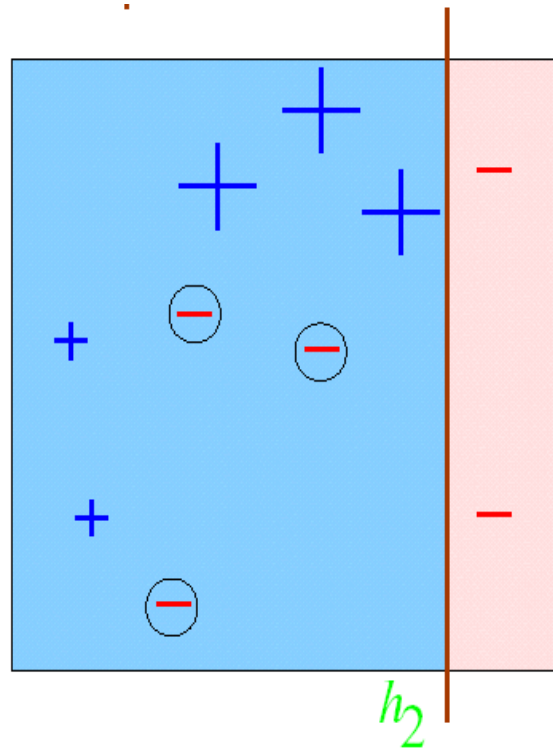


$\varepsilon_1 = 0.30$
 $\alpha_1 = 0.42$

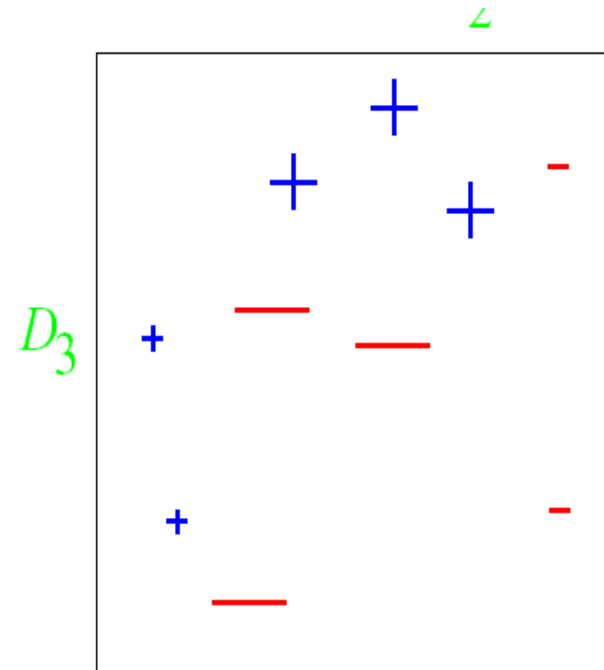
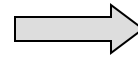


Boost example

ROUND 2

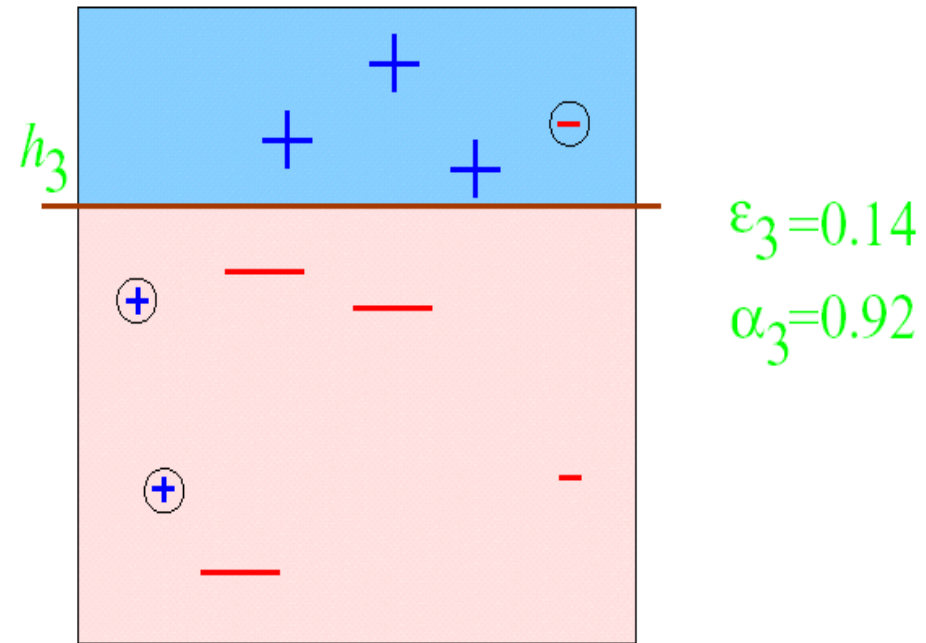


$$\epsilon_2 = 0.21$$
$$\alpha_2 = 0.65$$



Boost example

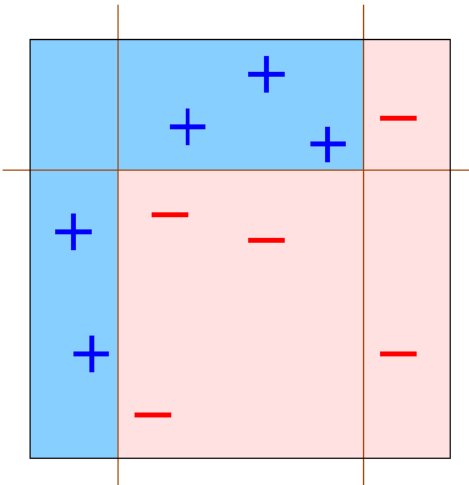
ROUND 3



Boost example

$$H_{\text{final}} = \text{sign} \left(0.85 \begin{array}{|c|} \hline \text{blue} \\ \hline \end{array} + 1.32 \begin{array}{|c|} \hline \text{blue} \\ \hline \end{array} + 1.82 \begin{array}{|c|} \hline \text{blue} \\ \hline \end{array} \right)$$

=



AdaBoost - Introduction

Linear classifier with all its desirable properties

Has good generalization properties

Is a feature selector with a principled strategy (minimisation of upper bound on empirical error)

Close to sequential decision making

AdaBoost

Is an algorithm for constructing a “strong” classifier as linear combination

$$f(x) = \sum_{t=1}^T \alpha_t h_t(x)$$

of simple “weak” classifiers $h_t(x)$.

$h_t(x)$ - “weak” or basis classifier, hypothesis, “feature”

$H(x) = \text{sign}(f(x))$ - “strong” or final classifier/hypothesis

AdaBoost

Input – a training set: $S = \{(x_1, y_1); \dots ; (x_m, y_m)\}$

- $x_i \in X$, X instance space
- $y_i \in Y$, Y finite label space
 - in binary case $Y = \{-1, +1\}$

Each round, $t=1, \dots, T$, AdaBoost calls a given weak or base learning algorithm – accepts as input a sequence of training examples (S) and a set of weights over the training example ($D_t(i)$)

AdaBoost

The weak learner computes a weak classifier (h_t), : $h_t : X \rightarrow \mathbb{R}$

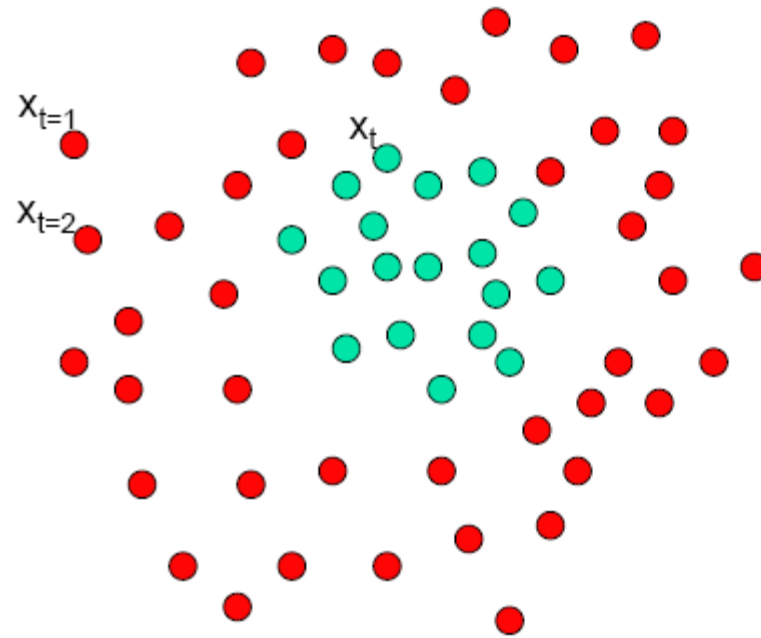
Once the weak classifier has been received, AdaBoost chooses a parameter ($\alpha_t \in \mathbb{R}$) – intuitively measures the importance that it assigns to h_t .

The main idea of AdaBoost

to use the weak learner to form a highly accurate prediction rule by calling the weak learner repeatedly on different distributions over the training examples.

initially, all weights are set equally, but each round the weights of incorrectly classified examples are increased so that those observations that the previously classifier poorly predicts receive greater weight on the next iteration.

Boosting - Example



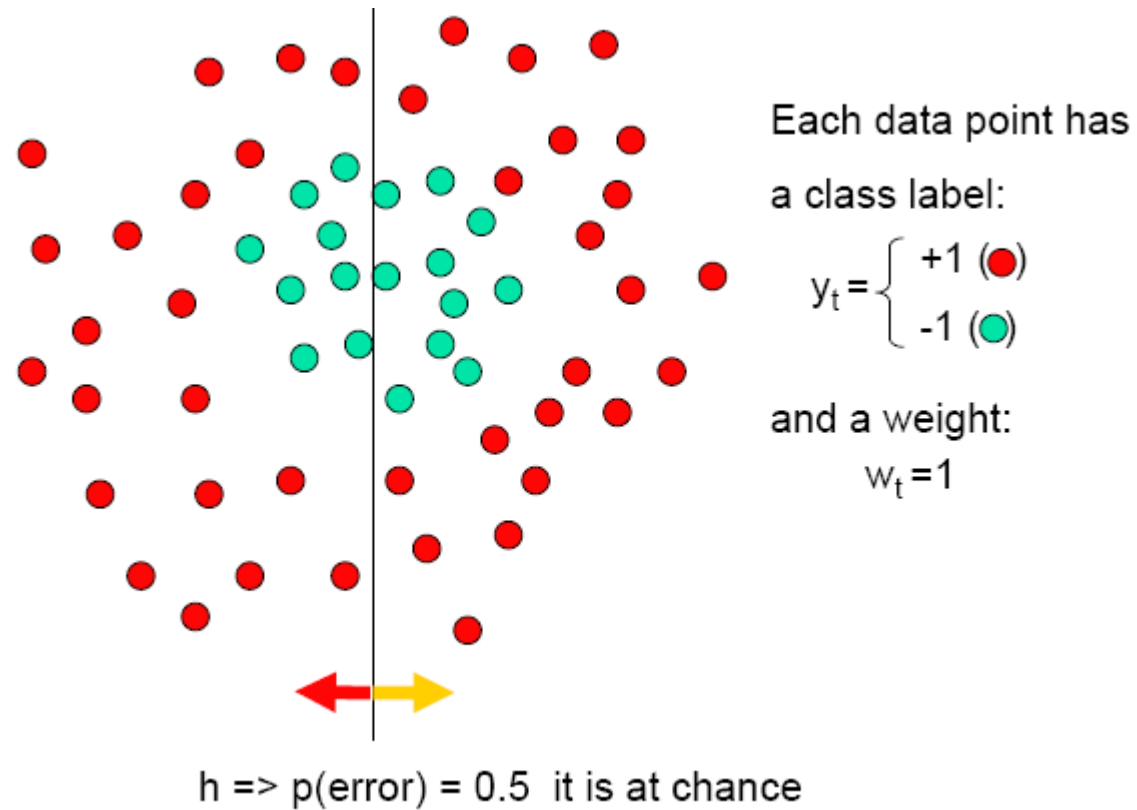
Each data point has
a class label:

$$y_t = \begin{cases} +1 & (\text{red circle}) \\ -1 & (\text{cyan circle}) \end{cases}$$

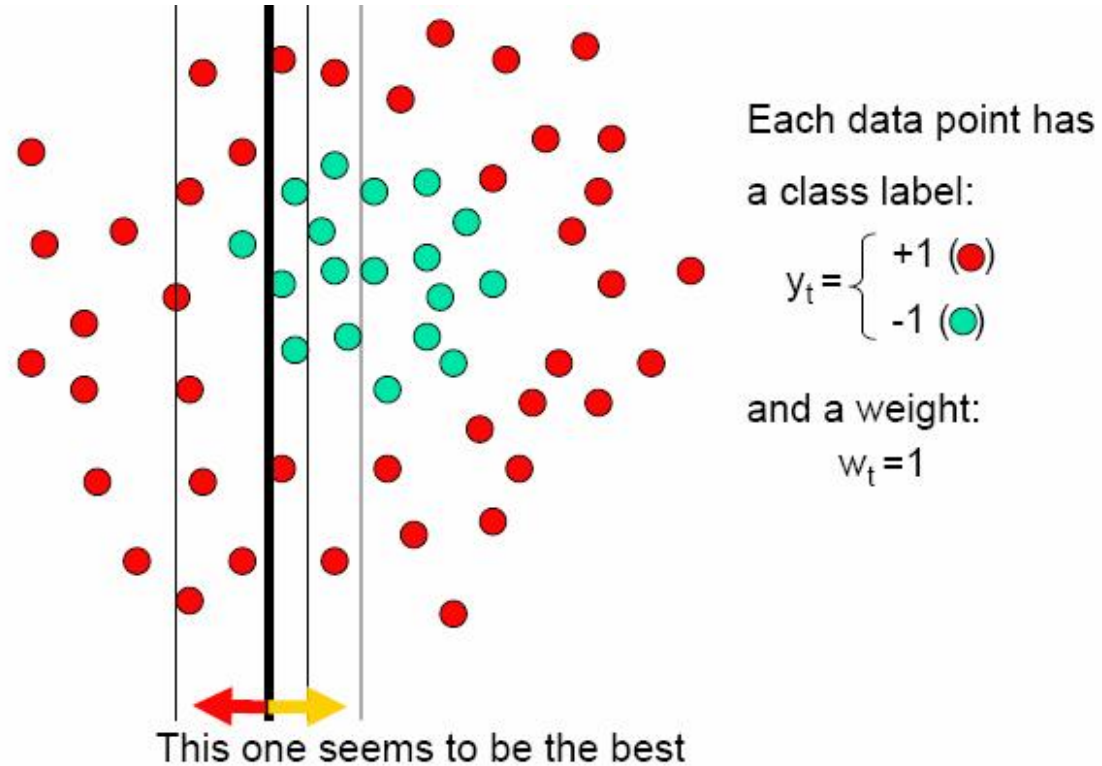
and a weight:

$$w_t = 1$$

Boosting - Example

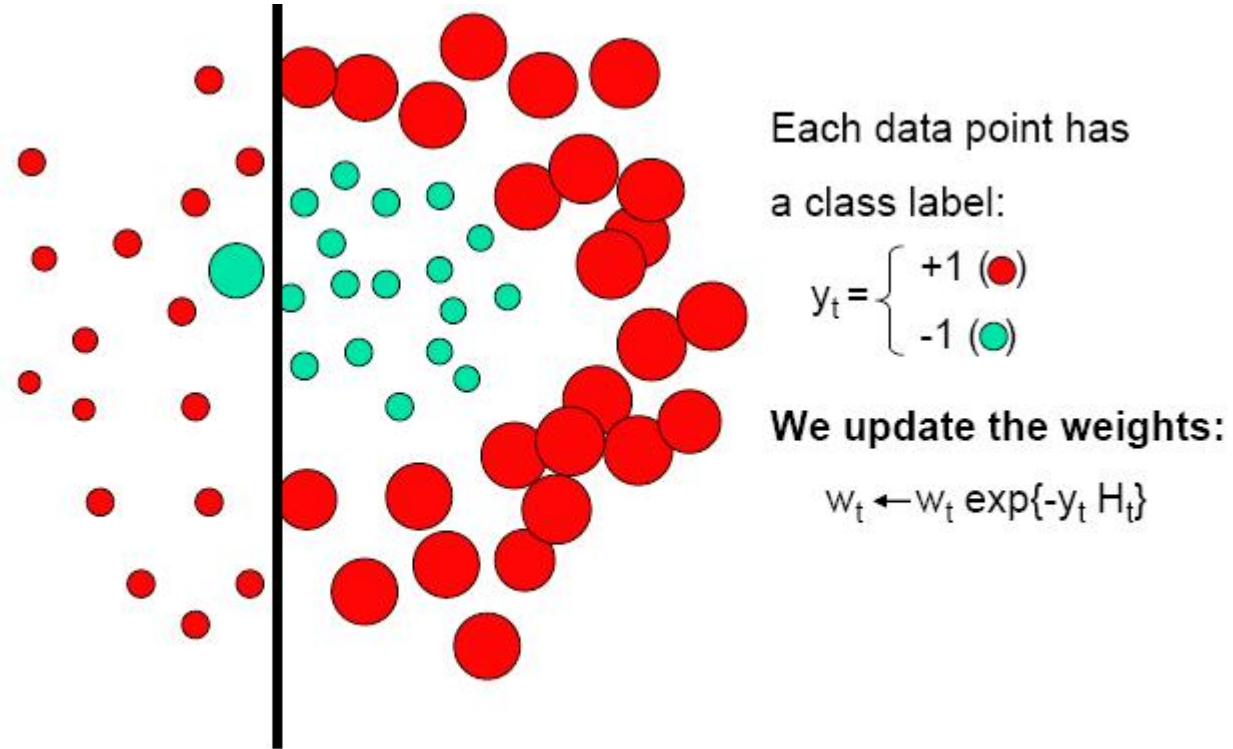


Boosting - Example



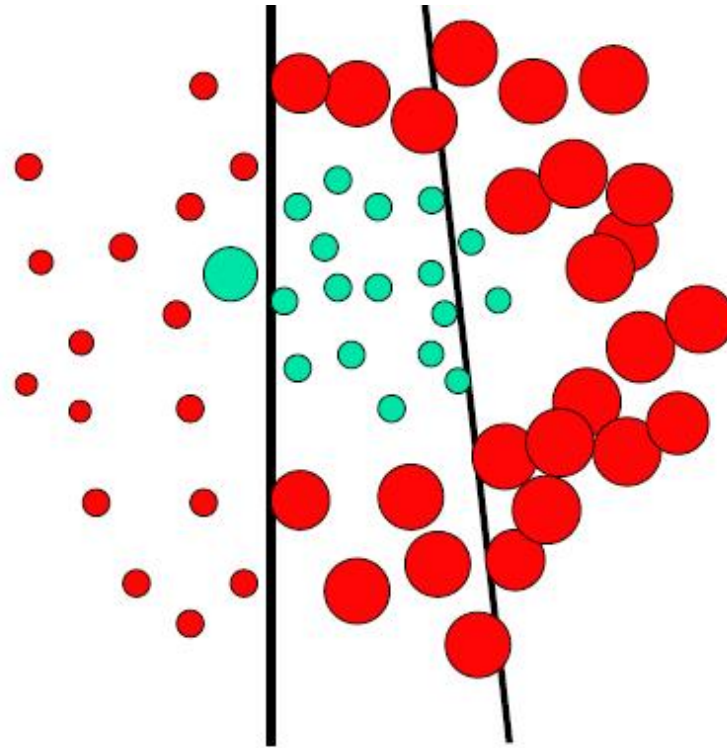
This is a '**weak classifier**': It performs slightly better than chance.

Boosting - Example



We set a new problem for which the previous weak classifier performs at chance again

Boosting - Example



Each data point has
a class label:

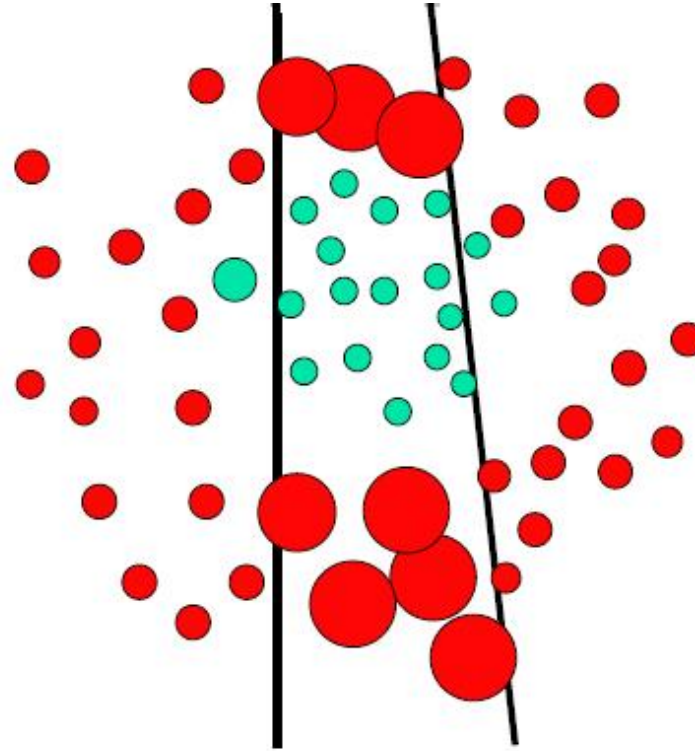
$$y_t = \begin{cases} +1 & (\text{red}) \\ -1 & (\text{teal}) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Boosting - Example



Each data point has
a class label:

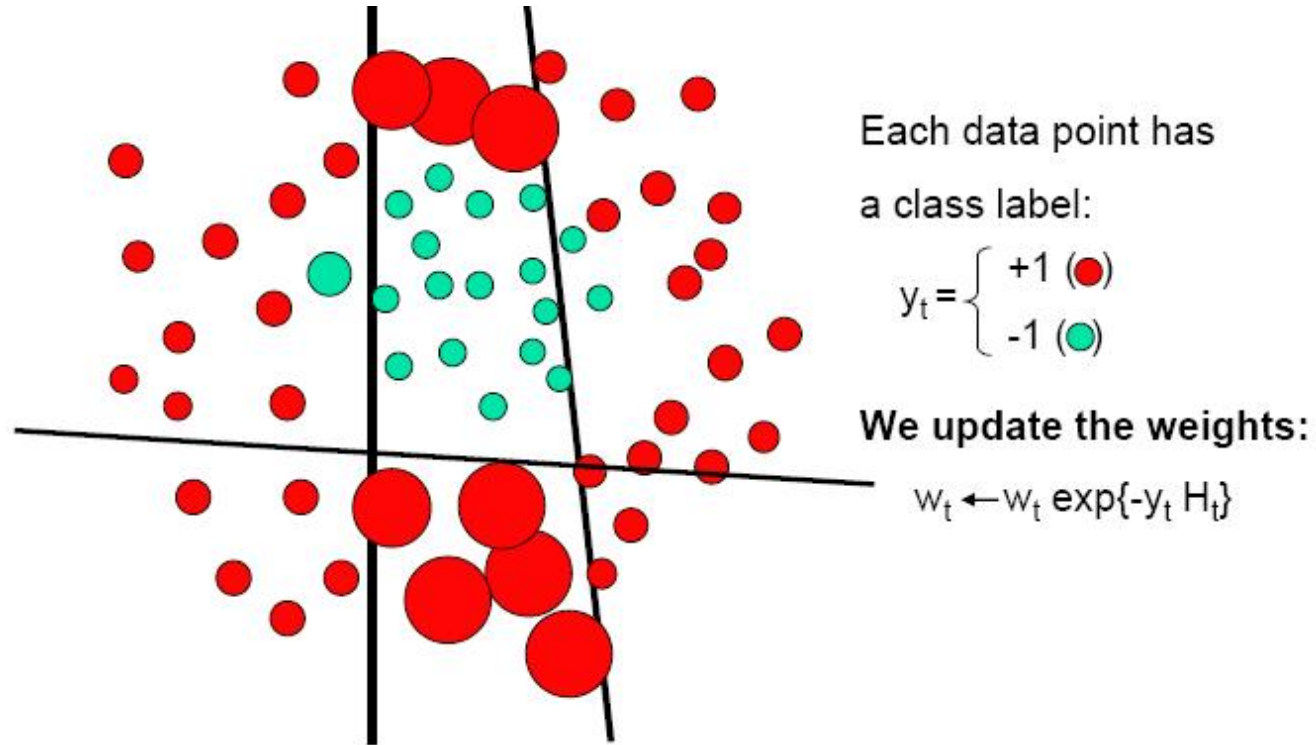
$$y_t = \begin{cases} +1 & (\text{red circle}) \\ -1 & (\text{cyan circle}) \end{cases}$$

We update the weights:

$$w_t \leftarrow w_t \exp\{-y_t H_t\}$$

We set a new problem for which the previous weak classifier performs at chance again

Boosting - Example



We set a new problem for which the previous weak classifier performs at chance again