CS 412

MARCH 31ST – RETURN TO MACHINE LEARNING

Welcome back from "Spring Break"

HW3
- Graded by end-of-week

HW4
- Out next Thursday after class
- Due two Thursdays after (4/16)

Midterm
- Next Thursday (4/9)
- Take home
- 12-8 pm CDT
- Should take around 2 hours

Mostly written, but there may be some small coding portion

HW2 regrades done by end of the week

previous midterm piazza

Submitted on gradescope

Next Tuesday will be exam review

Going Forward

Slack

Online classes
- Mostly blackboard collab during lectures.
- One panopto lecture over the weekend to catch up on material ← downloadable

Graded things left
- HW4 Out next week,
- HW5
- Midterm
- Final Exam
- Final Project

OH:
blackboard

Final Project

Official write up/description

Groups of up to 3

- Groups of 2, I expect ~50% more
- Groups of 3, ~100% more

Solo projects are fine

Every group will need to meet digitally with me (~15 minutes) between 4/13 and 4/17 to make sure the project is feasible

Project will be 5-10 pages, single-spaced, but with figures

- Problem introduction
- Previous work
- Novel attempt
- Results
- Future work/Conclusion

Final Project

Things to consider:

- Must involve a new dataset ← no digits
- Must involve at least one element not covered in the course material ← of ML
- Should also include a significant amount of the course material
- *For graduate students,* must include some ML research from the last 5 years

↳ a lot of researchers post their code

I **strongly** encourage you to find a project that aligns with your personal interests / research agenda

In general, I want to allow any reasonable machine learning project

Tips for finding a project

Was there something difficult for you in a homework assignment?

Is there a particular dataset that you would find interesting?

- For example, there is a large amount of medical data

What sorts of data do you think would give a different result from what you’ve seen?

Rationale for Ensemble Learning

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

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

ensemble

or higher precision

Different learners use different

- Algorithms
- Parameters
- Representations (Modalities)
- Training sets
- Subproblems

3 independent

70%

Bagging (Bootstrap aggregating)

ensembl method

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)

less independence when the model is stabl

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

- This is reduced with "stable" learners

unstable learners get a big variance reduction

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, $f^1, \ldots, f^B$

$f^t \to f^{t+1}$

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).

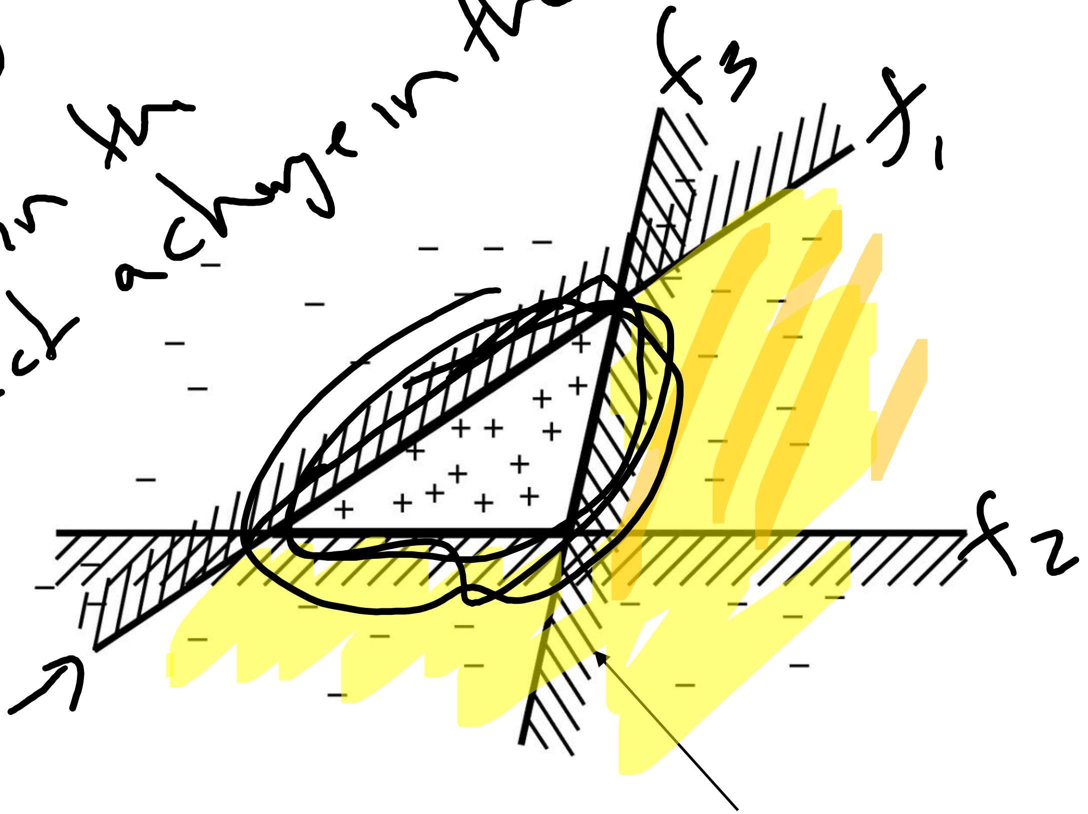
70% Each classifier is "weak" but the ensemble is "strong."

AdaBoost is a specific boosting method.

f^{t+1} "focus" on elements incorrectly classified by $f^t \to f^t$

Example

Stability: to what extend does a change in the data set effect a change in the model



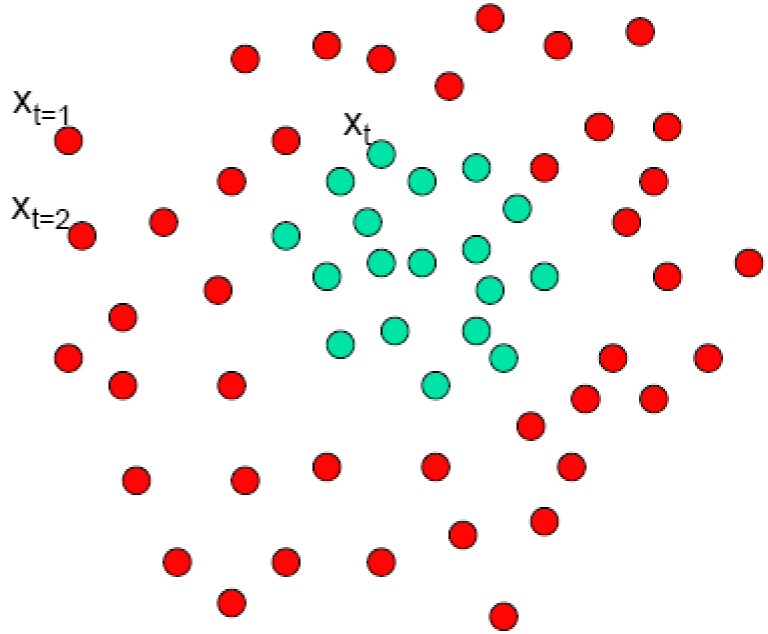
incorrectly classified by F_1

weight these points more when we build f_2

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

linear separators

Boosting - Example



Each data point has

a class label:

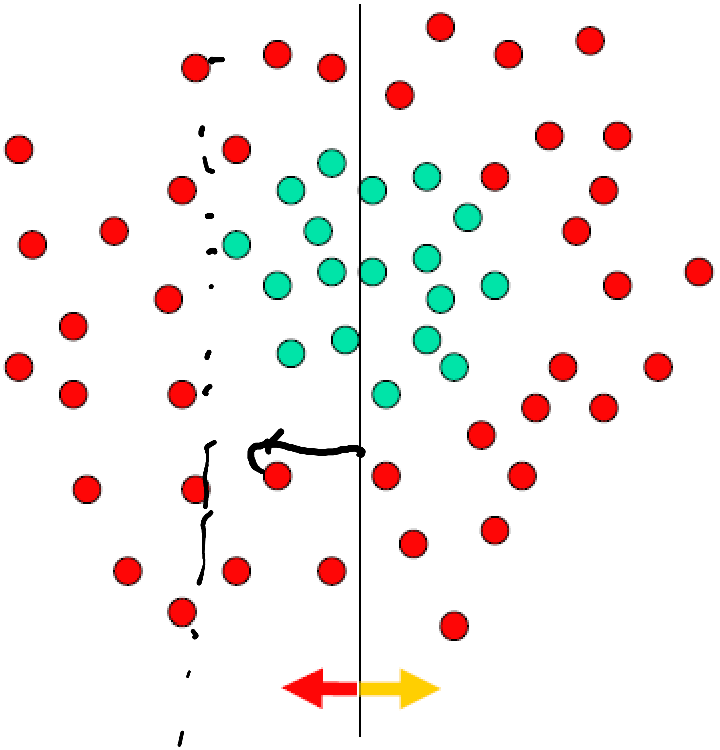
$$y_t = \begin{cases} +1 \; (\;) \\ -1 \; (\;) \end{cases}$$

and a weight:

$w_t = 1$

Boosting - Example

not a good ensemble learner



Each data point has a class label:

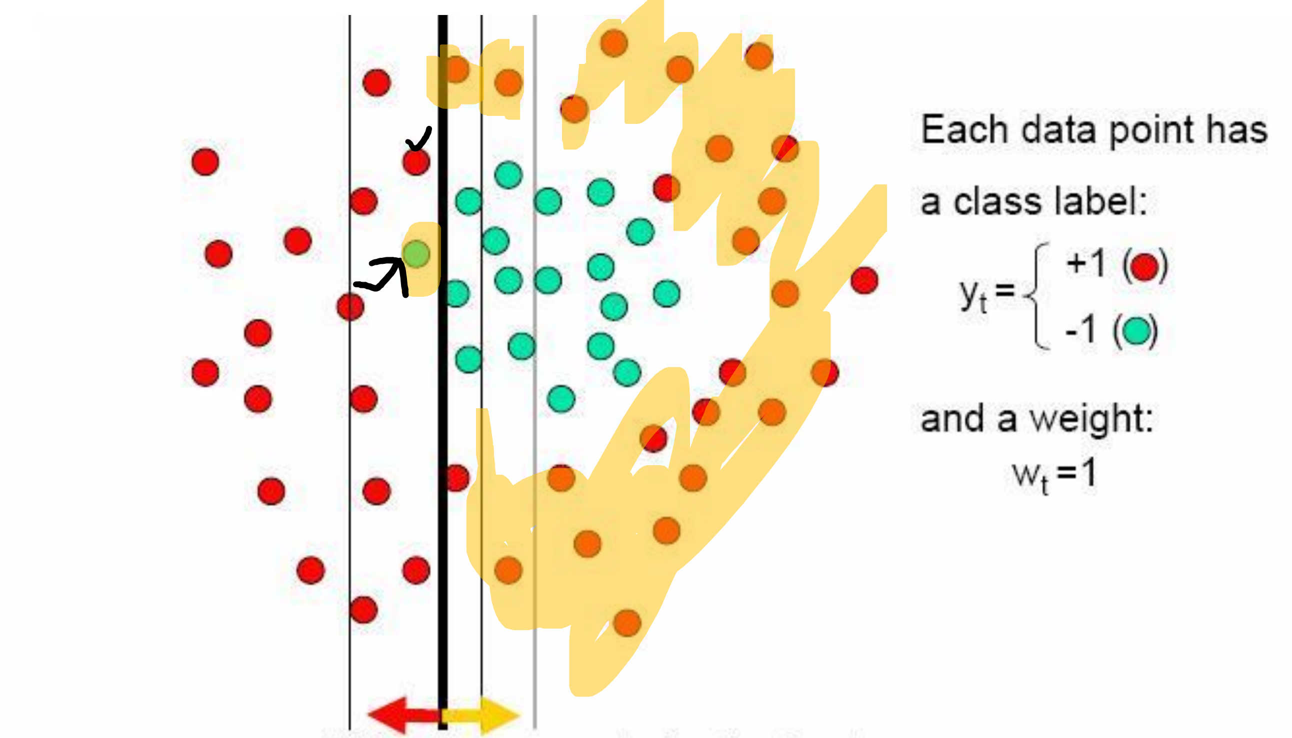
$$y_t = \begin{cases} +1 \ (\text{red}) \\ -1 \ (\text{green}) \end{cases}$$

and a weight:

$w_t = 1$

h => p(error) = 0.5 it is at chance

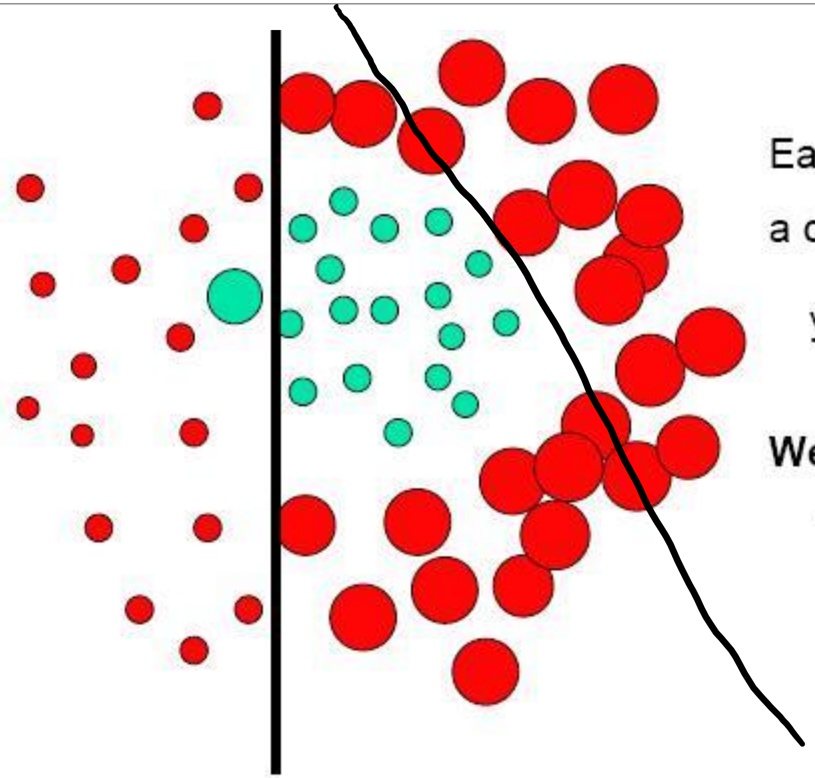
Boosting - Example



This one seems to be the best

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

Boosting - Example



Each data point has

a class label:

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

We update the weights:

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

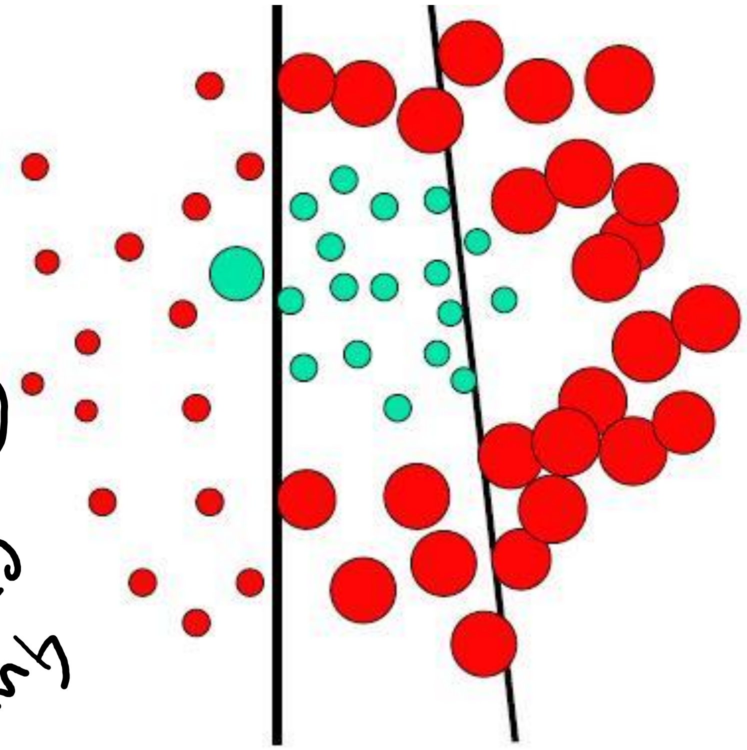
adaboost

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

Boosting - Example

Every point has a weight start at 1

reduces the weight of correctly classified points



Each data point has

a class label:

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

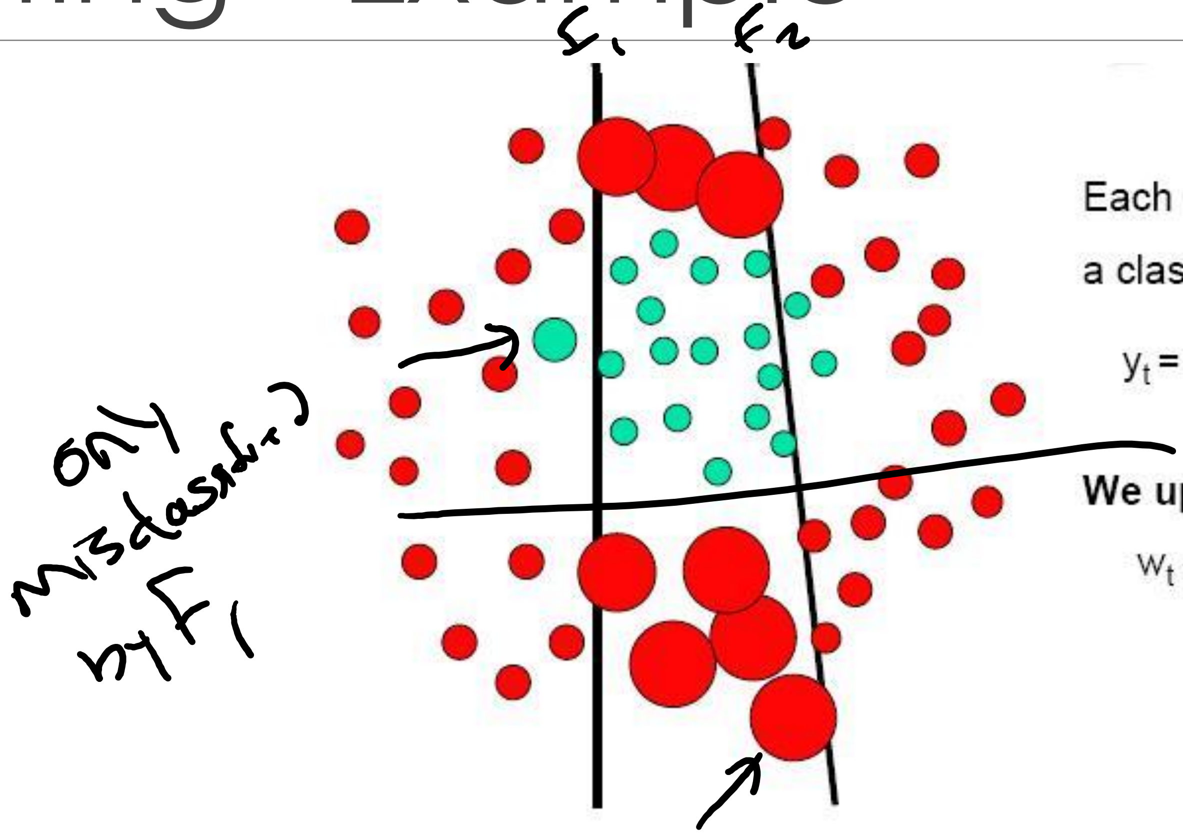
We update the weights:

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

$t \in$ Data Points

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

Boosting - Example



Each data point has

a class label:

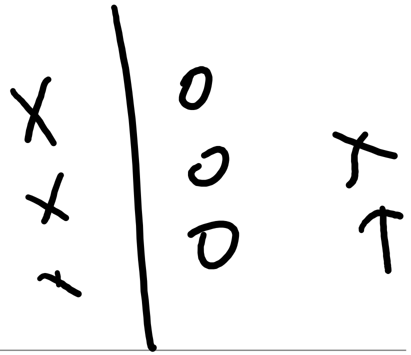
$$y_t = \begin{cases} +1 \\ -1 \end{cases}$$

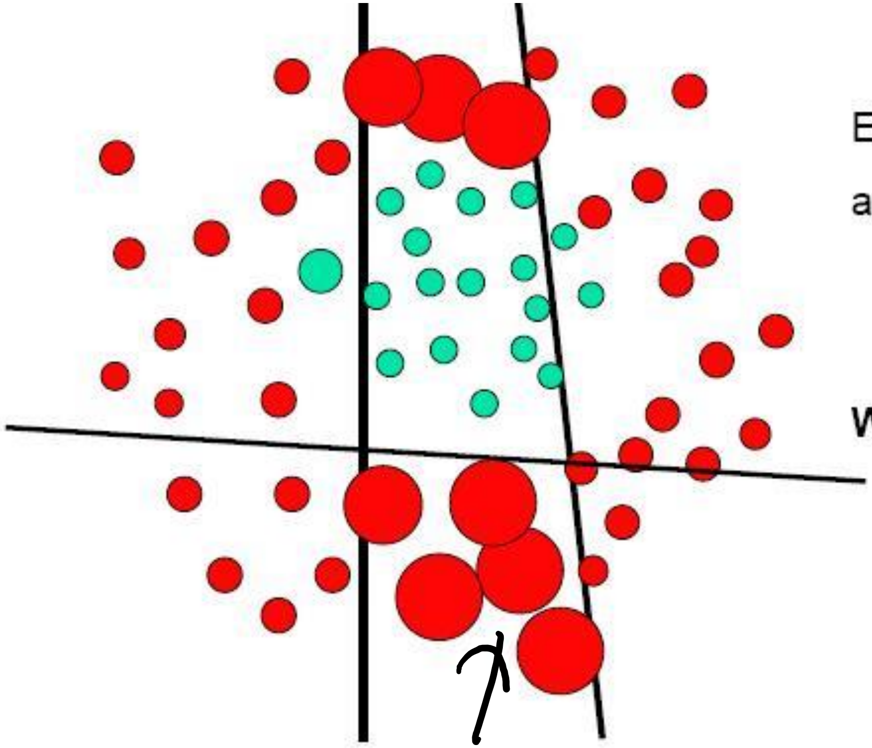
We update the weights:

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

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

Boosting - Example





Each data point has a class label:

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

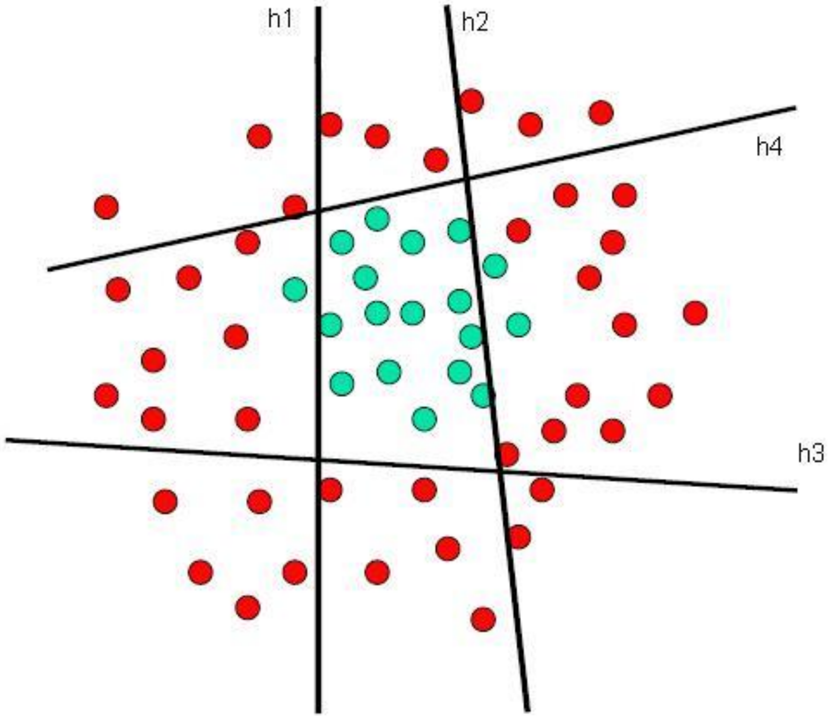
We update the weights:

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

cross-entropy of the decider

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

Boosting - Example



The strong (non- linear) classifier is built as the combination of all the weak (linear) classifiers.

Boosting Intuition

Weight = 2
= 2 points in the feature space

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. $F_1 \to \alpha_1 F_1$

↳ not strictly voting

Classifiers that 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?

↳ Bagging: bootstrap

↳ Boosting: weighted points

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

even with boosted data
learners can be similar w/ strong learners

boosted data causes a bigger difference
in models if the models are weak/unstable

Boosting

or weak models

Train a set of weak hypotheses: h1, …., hT.

hypotheses = T

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

- Each hypothesis h_t has a weight α_t.

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

α_t = each weight for each model t

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

- ➔ At round t, example xi has the weight Dt(i).

x_i $D_t(i)$

Boosting

$D_1(i) = 1/m$

$1, -1$

Binary classification problem

Training data:

h_1

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

m = number of data points

Dt(i): the weight of xi at round t. D1(i)=1/m.

A learner L that finds a weak hypothesis h_t: X ➔ Y given the training set and Dt

The error of a weak hypothesis h_t:

D = weight

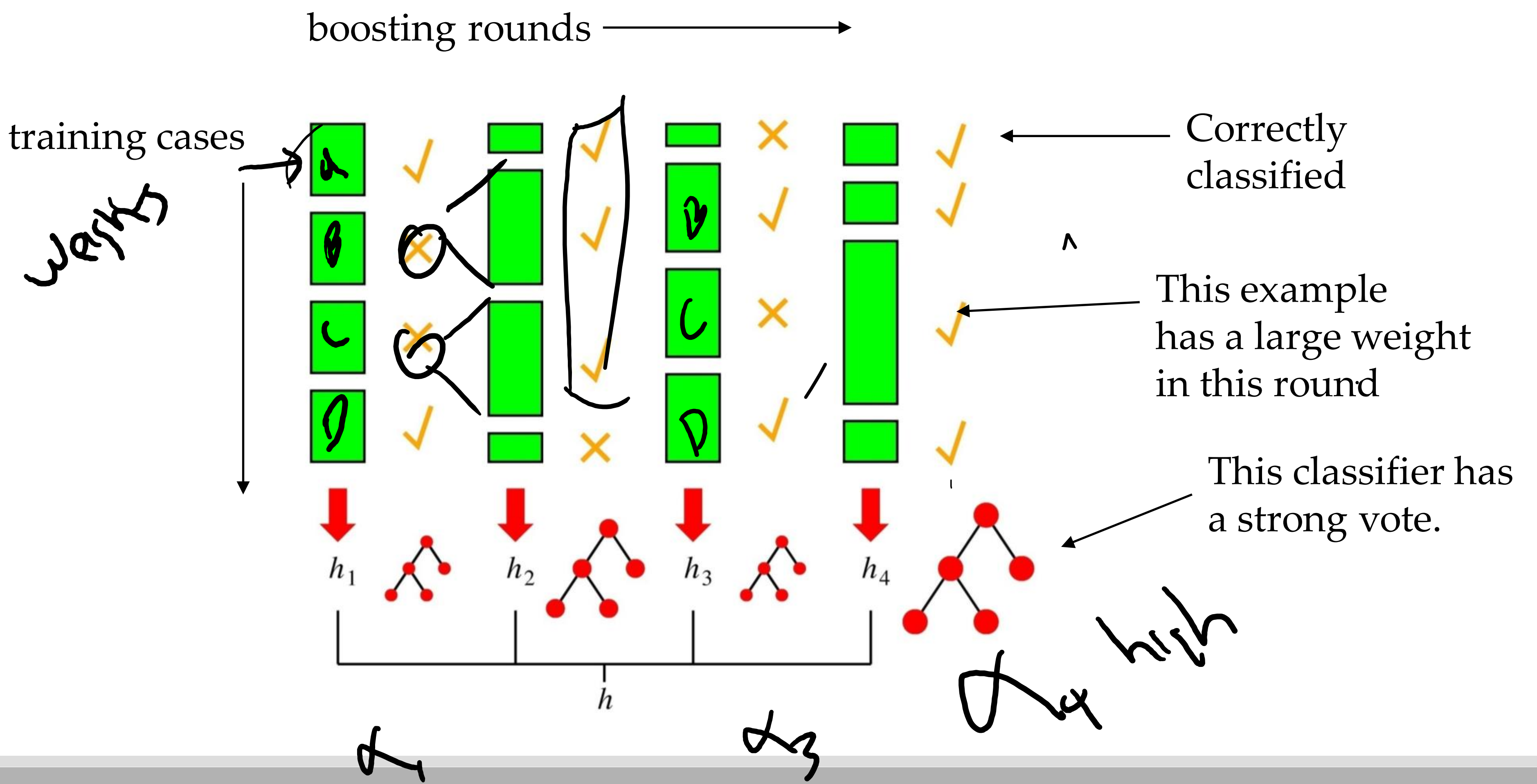
$$\varepsilon_t = \Pr_{i \sim D_t}[h_t(x_i) \neq y_i] = \sum_{i: h_t(x_i) \neq y_i} D_t(i)$$

example m = 100

$D_1 = 1/100$

$\sum_{30} 1/100 = 30/100$

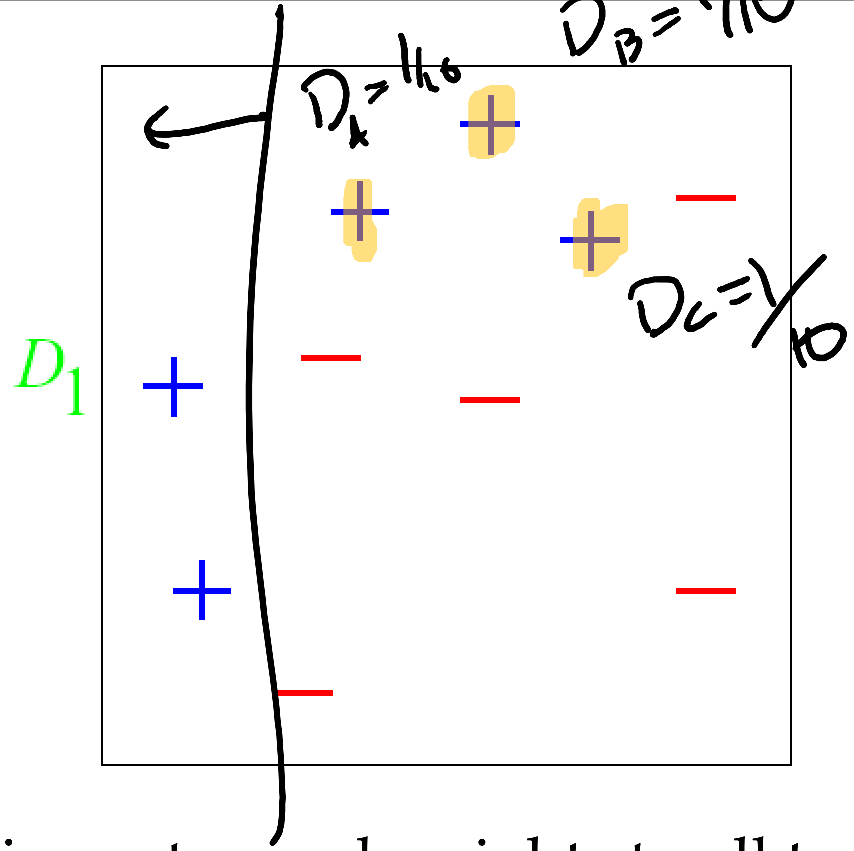
Boosting in a Picture



And in animation

$$\epsilon_t = \sum_{\text{misclass}} D$$

$$\alpha_t = \frac{\epsilon_t}{(1-\epsilon_t)}$$



$$\epsilon_1 = 0.30$$

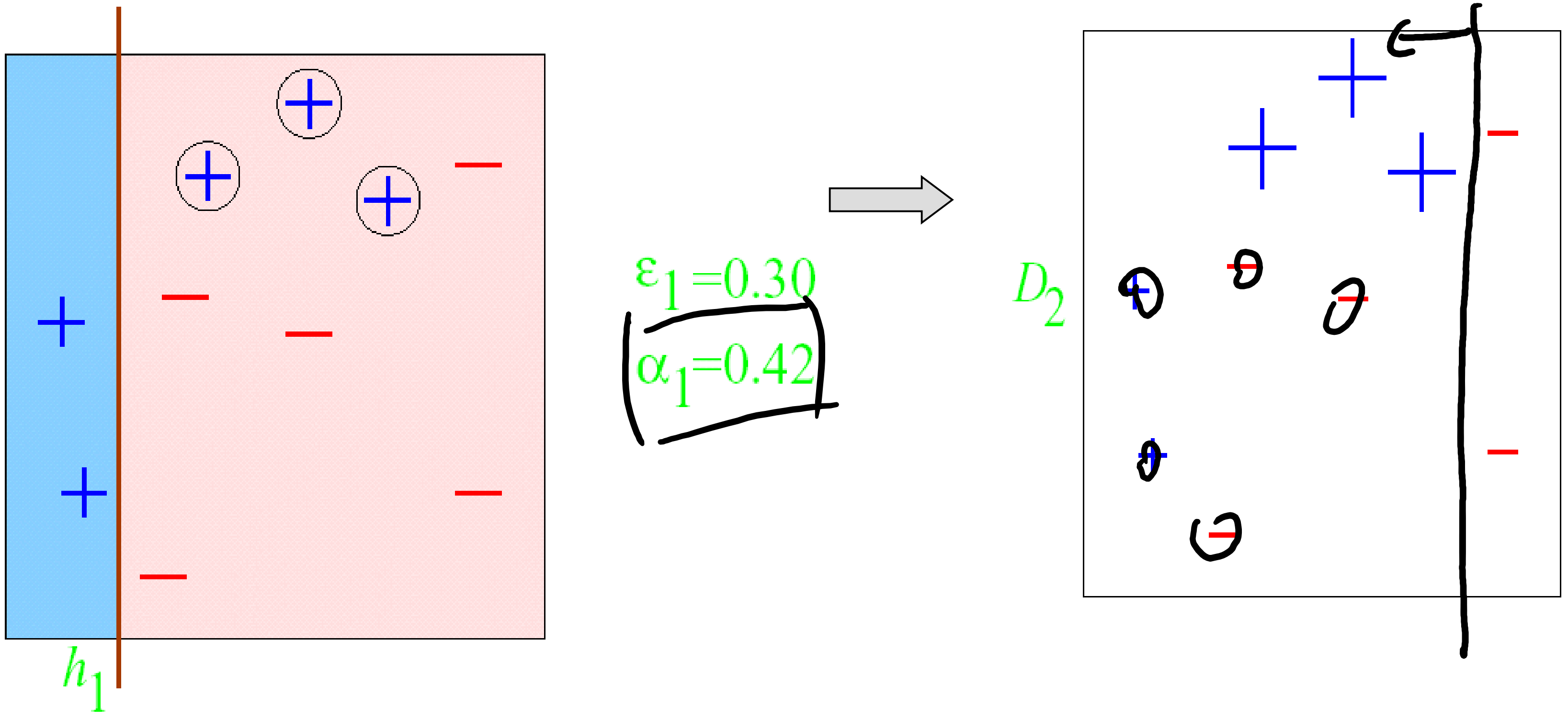
$$\alpha_1 = \frac{0.30}{0.7} = 0.42$$

Original training set: equal weights to all training samples

Boost example

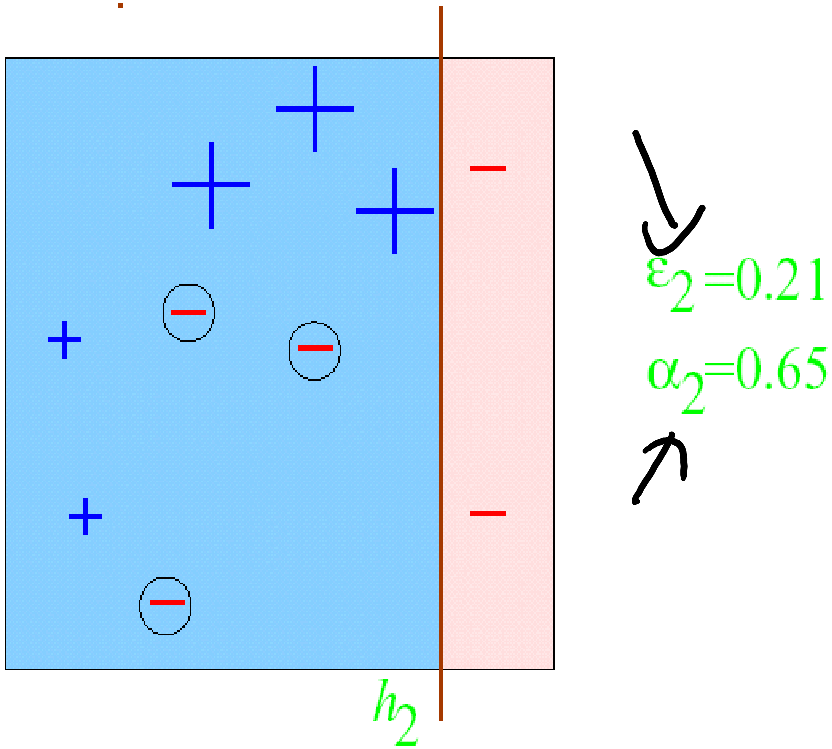
ε = error rate of classifier

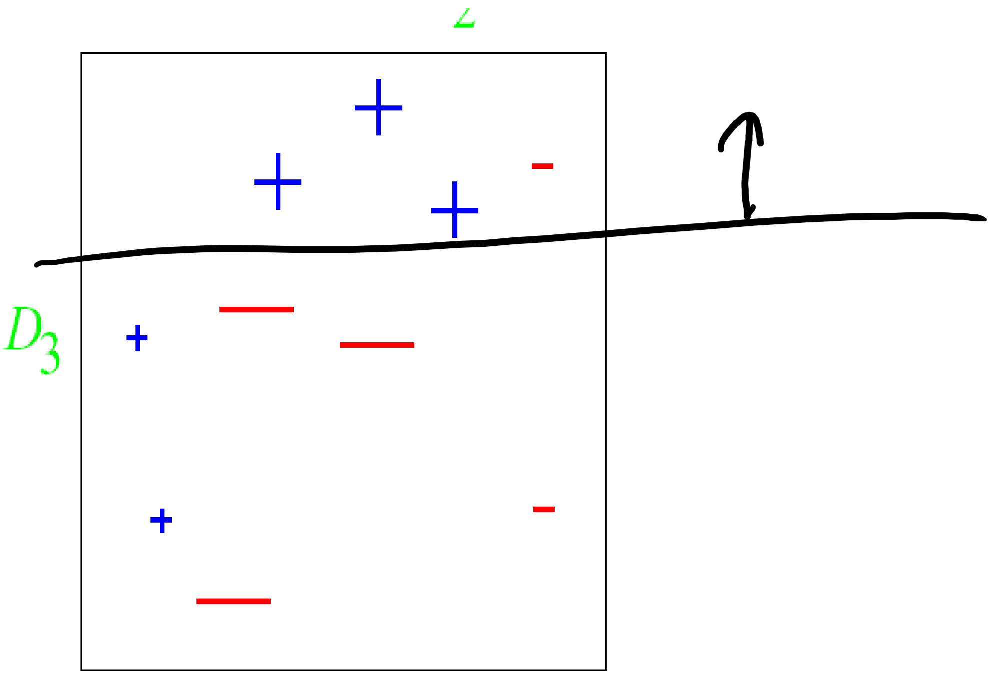
ROUND 1 α = weight of classifier e.g. [ε/(1-ε)]



Boost example

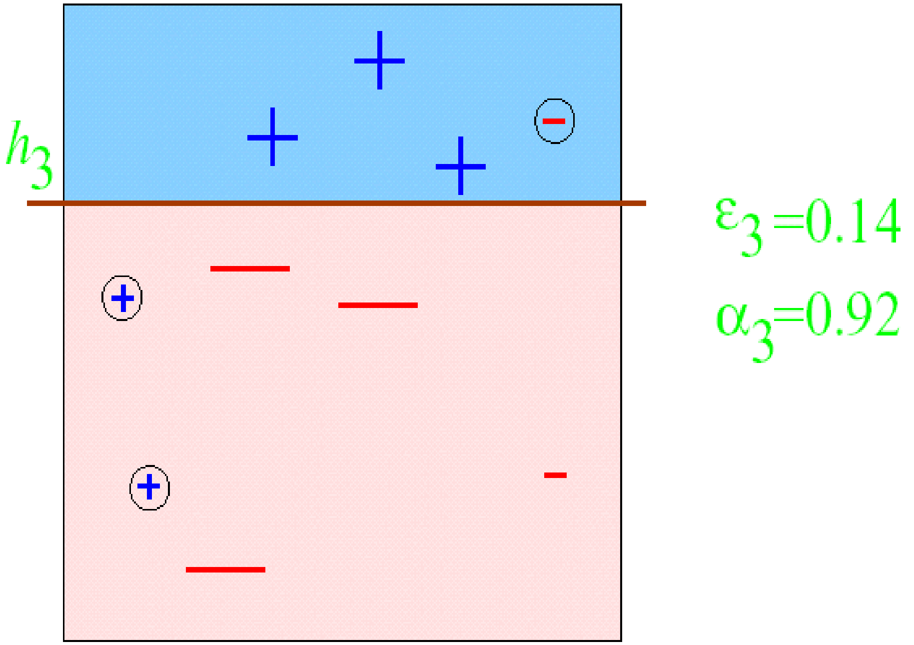
ROUND 2



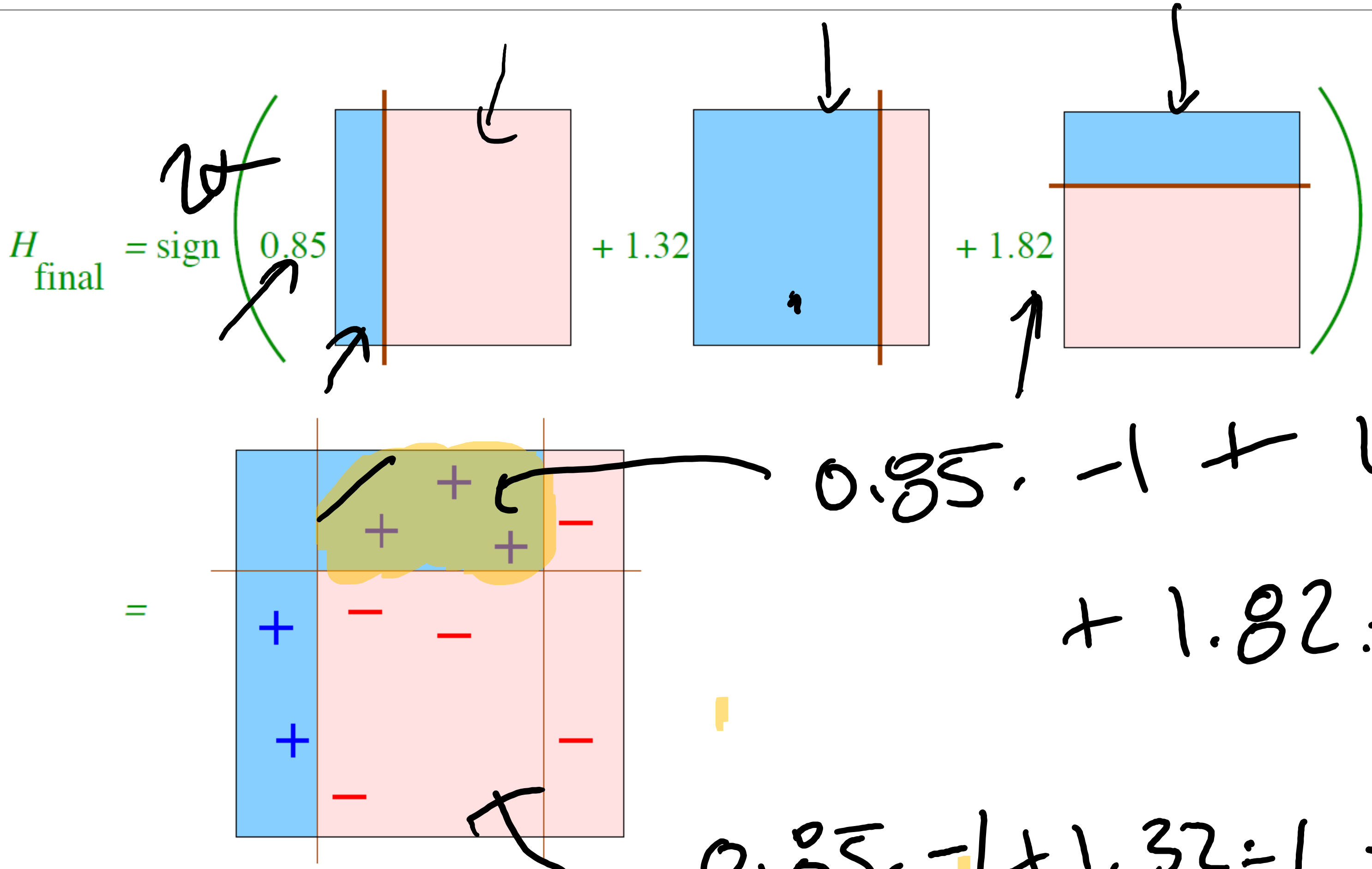


Boost example

ROUND 3



Boost example



$0.85 \cdot -1 + 1.32 \cdot 1$
$+ 1.82 \cdot 1 \geq 0$

$0.85 \cdot -1 + 1.32 \cdot 1 - 1.82 \cdot 1 \leq 0$

AdaBoost - Introduction

Linear classifier with all its desirable properties ← Decision stump - horizontal or vertical separator

Has good generalization properties

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

Close to sequential decision making

↳ stacking
cascading

$h_t(x)$
returns
$(-1, 1)$

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 ht(x).

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

H(x) = sign(f(x)) – "strong" or final classifier/hypothesis ensemble

AdaBoost

Input – a training set: S = {(x1, y1); … ;(xm, ym)}

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

classifiers

Each round, t=1,…,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 ($D_t(i)$)

$D_{t(i)}$ = weight for point i in weak learner # t

AdaBoost

The weak learner computes a weak classifier (h_t), : $h_t : X \to R$

Once the weak classifier has been received, AdaBoost chooses a parameter ($\alpha_t \varepsilon R$) – 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

Perform 3 iterations of the boosting algorithm on the following dataset. Double the error at each point and weight each model with $\alpha_i = (1-\varepsilon)/\varepsilon$

# of learners