Boosting

DSE 220

Worksheet 10

Class 1 : $\omega_1 = (1,1)$, $b_1 = 0$ $\Rightarrow$ $f_1(x) = x_1 + x_2$

Class 2 : $\omega_2 = (1,0)$, $b_2 = 1$ $\Rightarrow$ $f_2(x) = x_1 + 1$

Class 3 : $\omega_3 = (0,1)$, $b_3 = -1$ $\Rightarrow$ $f_3(x) = x_2 - 1$

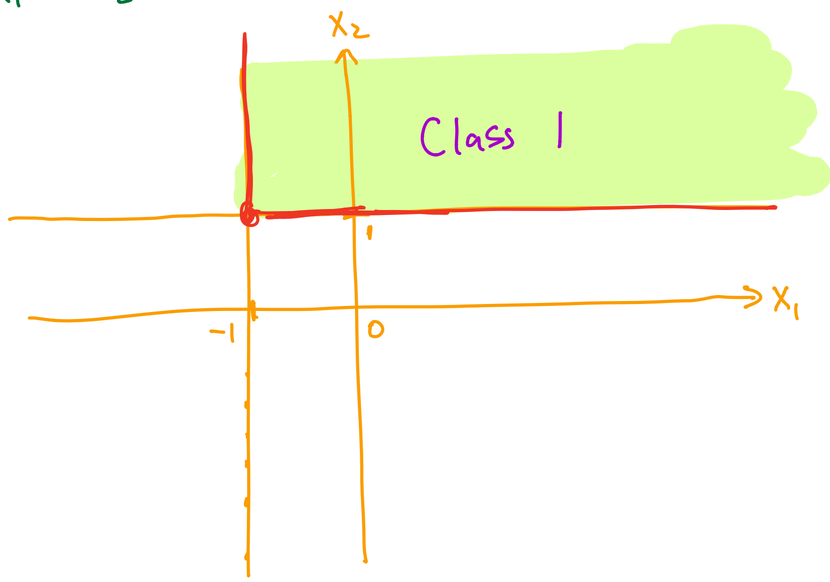
E.g. To classify point $(1,2)$

$$\left.\begin{aligned} f_1(1,2) &= 3 \\ f_2(1,2) &= 2 \\ f_3(1,2) &= 1 \end{aligned}\right\} \quad \therefore \text{ Predict class } \underline{\underline{1}}$$

$\therefore$ Predict class 1 when $f_1(x) \geq f_2(x)$ and $f_1(x) \geq f_3(x)$:

$$x_1 + x_2 \geq x_1 + 1 \iff x_2 \geq 1$$

$$x_1 + x_2 \geq x_2 - 1 \iff x_1 \geq -1$$



Choosing a classifier

So many choices:

- Nearest neighbor
- Different generative models
- Linear predictors with different loss functions
- Different kernels
- Neural nets
- etc.

Can one **combine** them?
And get a classifier that is better than any of them individually?

Voting or weighted majority ? (eg. can fit weights using logistic regression)

Combining simple classifiers

1. No one classifier is going to be the final product.
   So why not keep the individual components simple?

Combining simple classifiers

1. No one classifier is going to be the final product.
   So why not keep the individual components simple?

2. How to train each constituent classifier?
   On the full training set?

Combining simple classifiers

1. No one classifier is going to be the final product.
   So why not keep the individual components simple?

2. How to train each constituent classifier?
   On the full training set?

3. The full (combined) models may get enormous.
   Is this bad for generalization? (overfitting)

Weak learners

It is often easy to come up with a **weak classifier**, one that is marginally better than random guessing:

$$\Pr(h(X) \neq Y) \leq \frac{1}{2} - \epsilon$$

e.g. 45% error

A learning algorithm that can consistently generate such classifiers is called a **weak learner**. ← black-box learning alg

Is it possible to systematically boost the quality of a weak learner?

The blueprint for boosting

Weak learner takes a weighted data set

Given: data set $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$. ← $y \in \{-1, +1\}$

- Initially give all points equal weight.
- Repeat for $t = 1, 2, \ldots$:
  - Feed weighted data set to the weak learner, get back a weak classifier h_t
  - Reweight data to put more emphasis on points that h_t gets wrong
- Combine all these h_t's linearly

weak learner → h_1 coeff. α_1 ; increase weight on points h_1 got wrong

cannot be the same as h_1; weighting is constructed so that h_1 has 50% error on the new weights ← h_2 coeff. α_2 ; increase weight on points h_2 got wrong

h_3 coeff. α_3

$\vdots$

h_T coeff. α_T

Prediction on a new point x is: $\text{sign}\left(\alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_T h_T(x)\right)$

AdaBoost

Data set $(x^{(1)}, y^{(1)}), \ldots, (x^{(n)}, y^{(n)})$, labels $y^{(i)} \in \{-1, +1\}$.

1. Initialize $D_1(i) = 1/n$ for all $i = 1, 2, \ldots, n$ ← $D_t(1), \ldots, D_t(n)$: weights on the n data points at time t; sum to 1

2. For $t = 1, 2, \ldots, T$:
   - Give D_t to weak learner (black box), get back some $h_t : \mathcal{X} \to [-1, 1]$ ← allow it to be more general than $\{-1,+1\}$ (measure of confidence, e.g.)
   - Compute h_t's margin of correctness:

$$r_t = \sum_{i=1}^{n} D_t(i) y^{(i)} h_t(x^{(i)}) \in [-1, 1]$$

← how accurate is h_t on D_t? +1 : perfect, 0 : random

$$\alpha_t = \frac{1}{2} \ln \frac{1 + r_t}{1 - r_t}$$

← coefficient of h_t in the final classifier

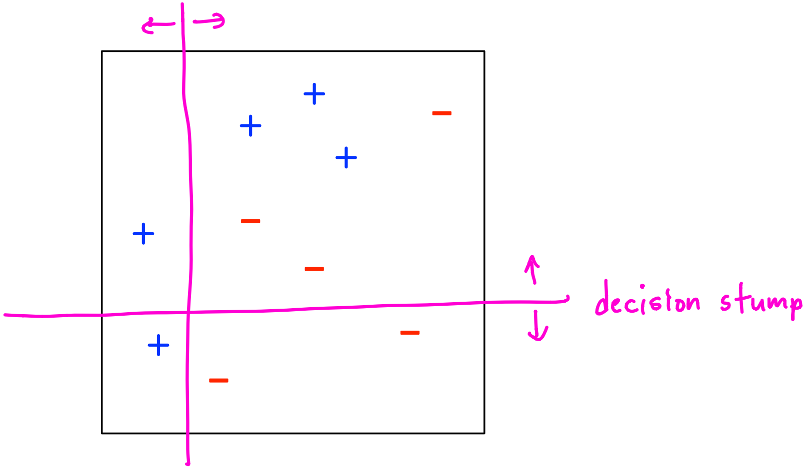
   - Update weights: $D_{t+1}(i) \propto D_t(i) \exp\left(-\alpha_t y^{(i)} h_t(x^{(i)})\right)$ ← put larger weight on the points that h_t got wrong

     ↑ "proportional to" (ie. normalize to sum to 1)

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

Example (Freund-Schapire)

Training set:

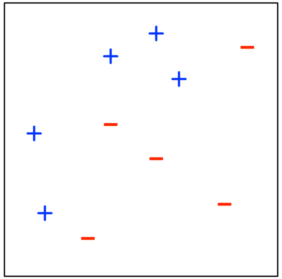


Use "decision stumps" (single-feature thresholds) as weak classifiers

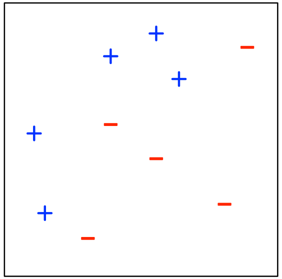
n data pts in $\mathbb{R}^d$: how many distinct decision stumps are there? $(n-1)\,d$

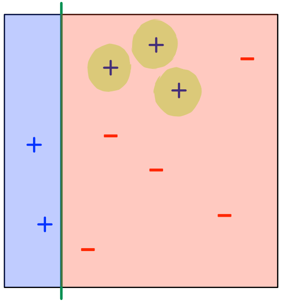
$\therefore$ Easy to choose best stump.

D_1



D_1



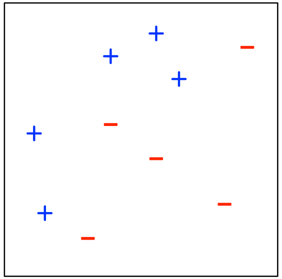


h_1

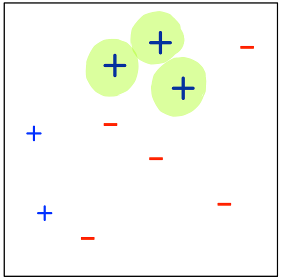
$r_1 = 0.40$, $\alpha_1 = 0.42$

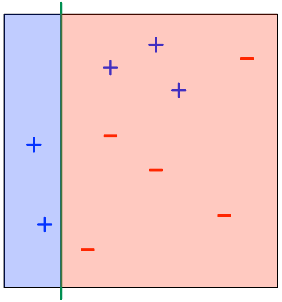
70% accurate

D_1



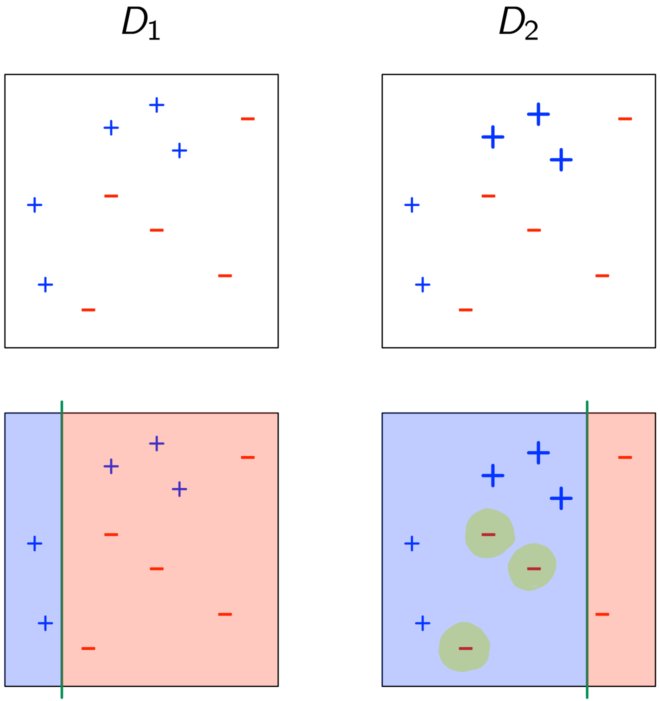
D_2





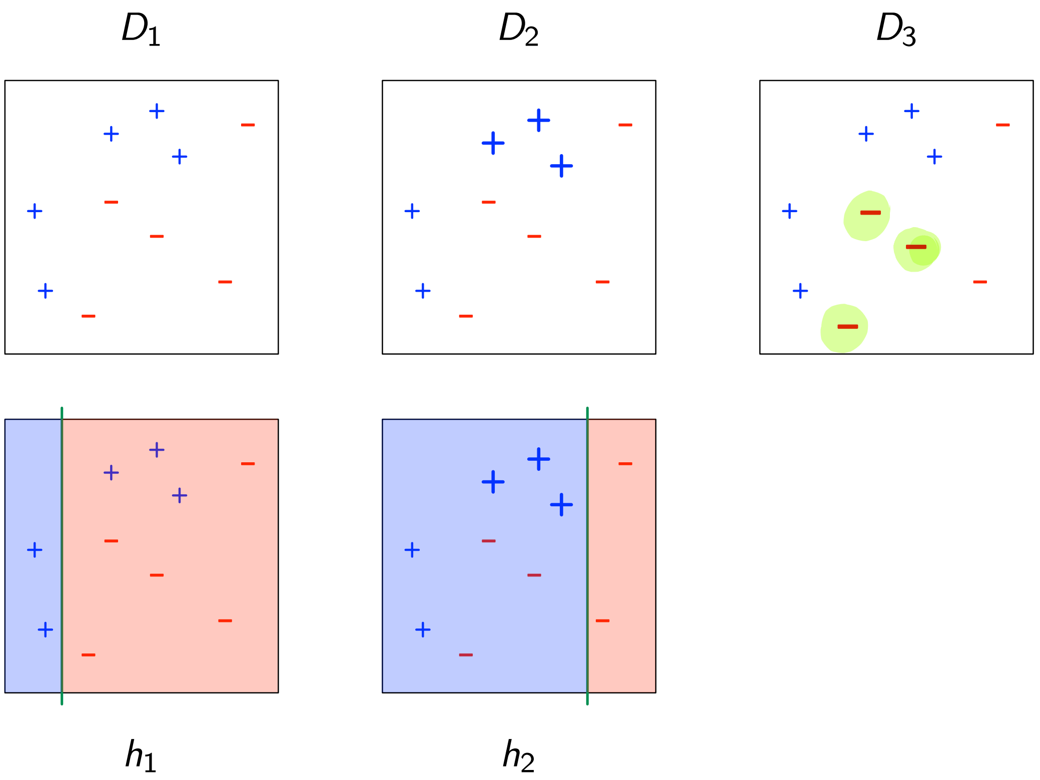
h_1

$r_1 = 0.40, \alpha_1 = 0.42$

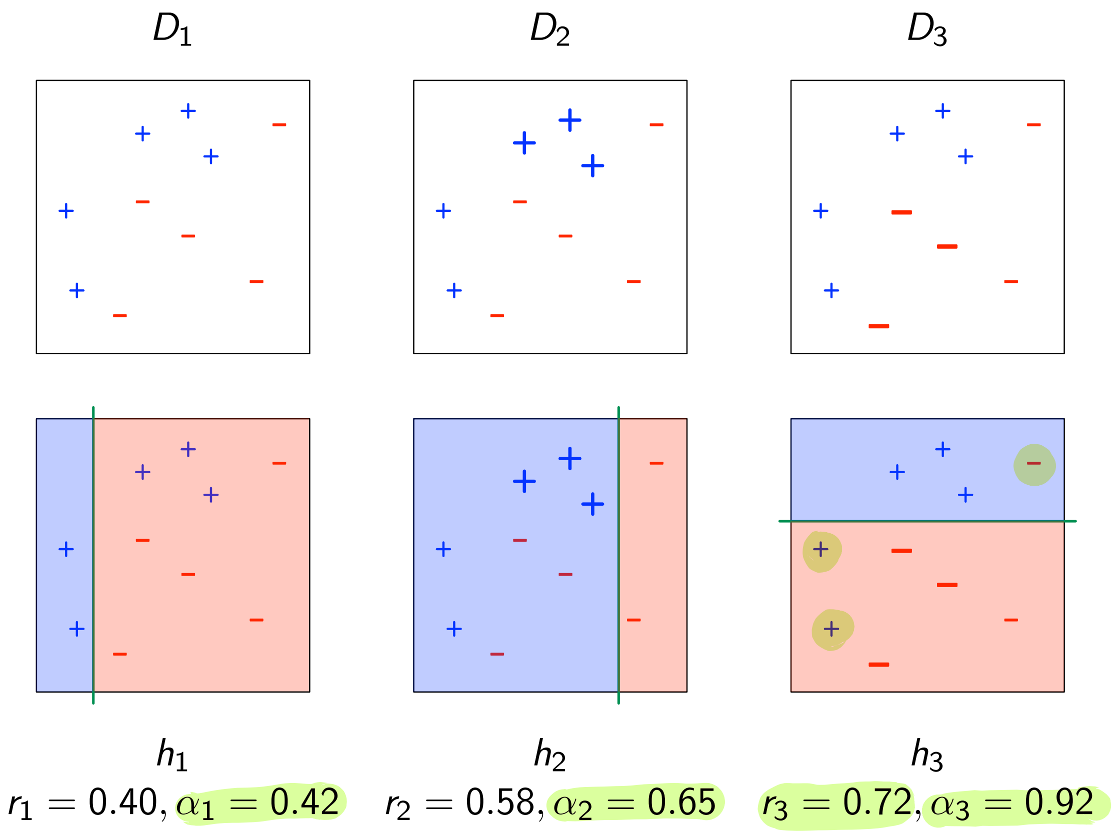


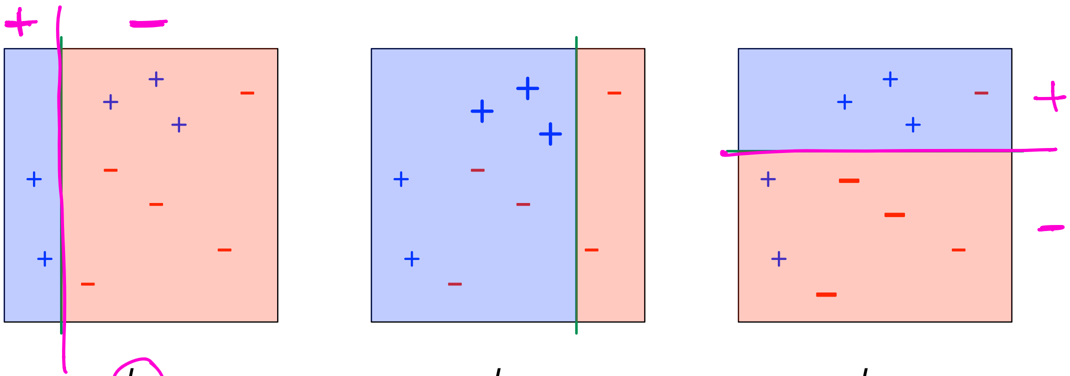
h_1 h_2

$r_1 = 0.40, \alpha_1 = 0.42$ $r_2 = 0.58, \alpha_2 = 0.65$



$r_1 = 0.40, \alpha_1 = 0.42$ $\quad$ $r_2 = 0.58, \alpha_2 = 0.65$

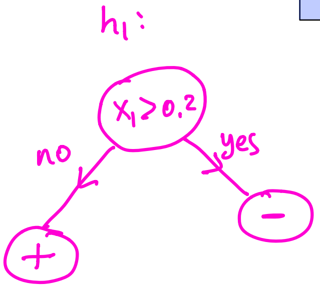

D_1
D_2
D_3
h_1
h_2
h_3
$r_1 = 0.40, \alpha_1 = 0.42$
$r_2 = 0.58, \alpha_2 = 0.65$
$r_3 = 0.72, \alpha_3 = 0.92$



h_1 $\alpha_1 = 0.42$

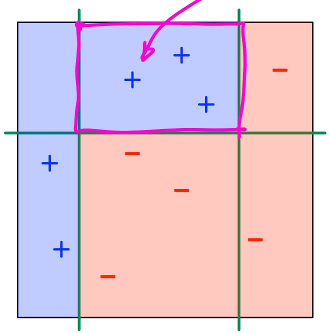
h_2 $\alpha_2 = 0.65$

h_3 $\alpha_3 = 0.92$



Final classifier:

$$\mathrm{sign}\left(0.42h_1(x) + 0.65h_2(x) + 0.92h_3(x)\right)$$



h_1: − 0.42
h_2: + 0.65
h_3: + 0.92

∴ +

The surprising power of weak learning

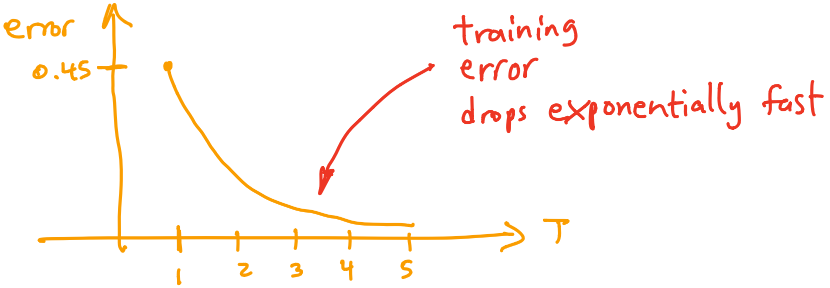
black box

Suppose that on each round t, the weak learner returns a rule h_t whose error on the time-t weighted data distribution is $\leq 1/2 - \gamma$. e.g. 0.05 (at most 45% error)

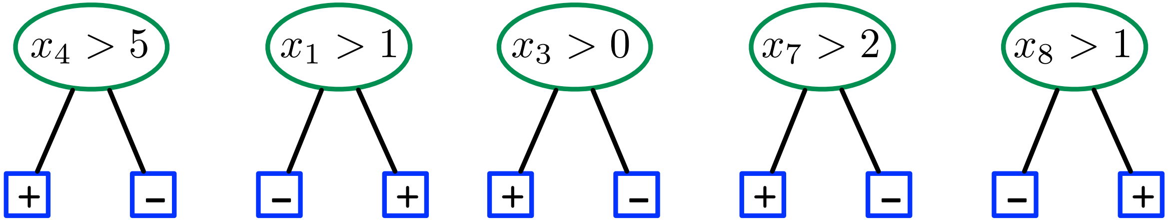
Then, after T rounds, the training error of the combined rule

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

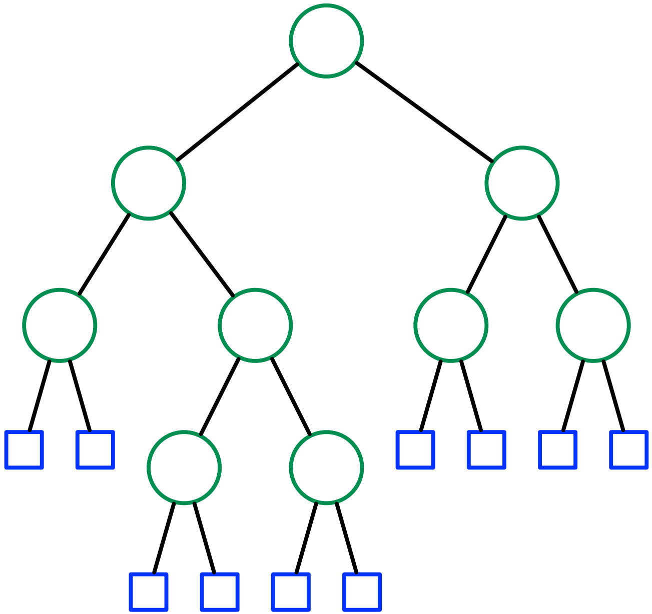
is at most $e^{-\gamma^2 T/2}$.



Boosting decision stumps



Boosting decision trees



Boosting decision trees