Boosting

ESLL, Chapter 10

Boosting

- Recall that an ensemble is a set of predictors whose individual decisions are combined in some way to classify new examples.
- (Previously) Bagging: Train classifiers independently on random

subsamples of the training data.

- (This lecture) Boosting: Train classifiers sequentially, each time focusing on training data points that were previously misclassified.
- Let's start with the concepts of weighted training sets and weak learner/classifier (or base classifiers).

Structure

- Abstract algo,
- Numerical example
- theory

*) Boosting:

Bag: D → D_1 → train a class./reg. using D_1 → $h_1(x)$

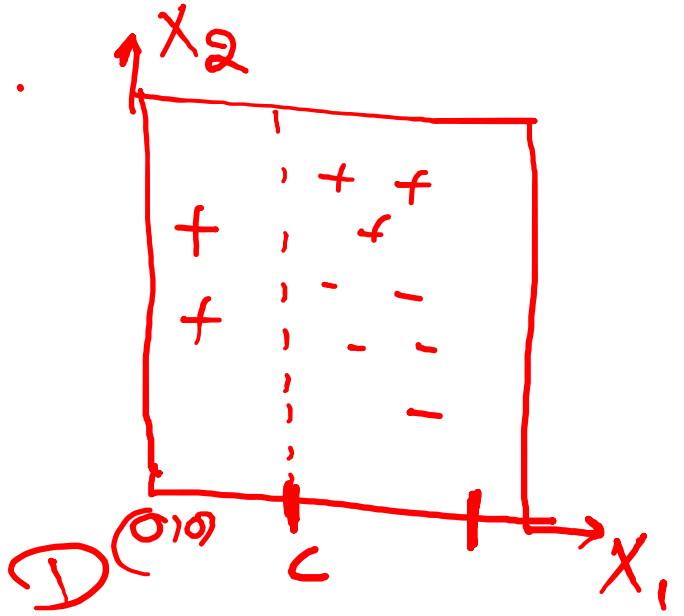
D → D_2 → $h_2(x)$

D → D_n → $h_n(x)$

Boosting → D, W W → weights for samples.

D, W → train a class. $\xrightarrow{h_1(x)}$ D, W', α_1 → train a new class $\xrightarrow{h_2(x)}$

Eg.



$$\text{loss } L = \frac{\sum_i w_i \, \mathbb{I}(y_i \neq h_1(x_i))}{\sum_i w_i}$$

Instead of Gini → we use this weighted loss.

$W = \frac{1}{10}$

$\mathbb{I}(\cdot)$ → indicator

= 1 → argument is true

= 0 → " " false.

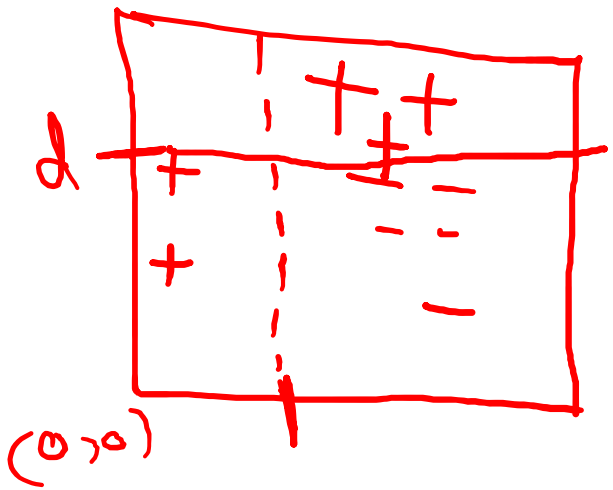
$$L = 3/10$$

$$\rightarrow \alpha = \frac{1}{2} \log \frac{1-L}{L} = \frac{1}{2} \log \frac{7}{3}$$

$\rightarrow w_i \leftarrow w_i e^{2\alpha}$ $\forall$ mis classified.

$$w_i \leftarrow \frac{1}{10} e^{2 \cdot \frac{1}{2} \log 7/3} = 7/30$$

for the 2nd iteration.

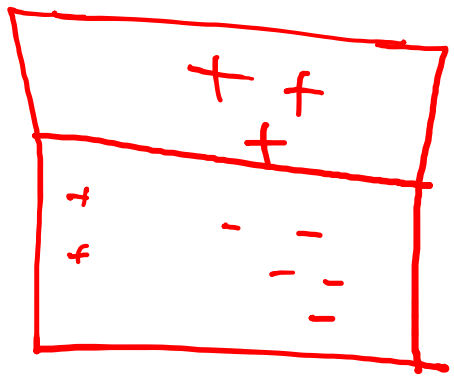


$h_2(x)$ for vertical split.

$$L = \frac{\frac{7}{30} \times 3}{3 \times \frac{7}{30} + 7 \cdot \frac{1}{10}}$$

for horizontal split. $= \frac{2 \cdot 1/10}{} = \frac{1}{7}$

$h_2(x)$



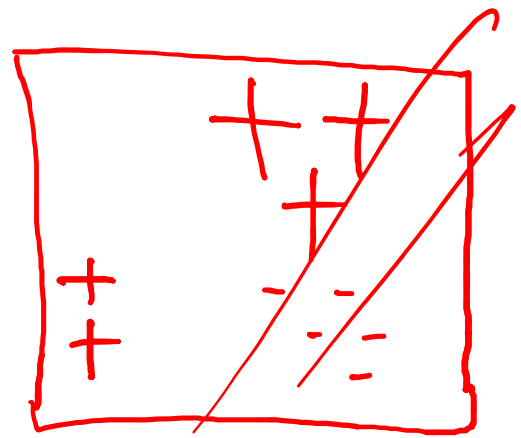
$$\alpha_2 = \frac{1}{2}\log\frac{1-1/7}{1/7} = \frac{1}{2}\log 6$$

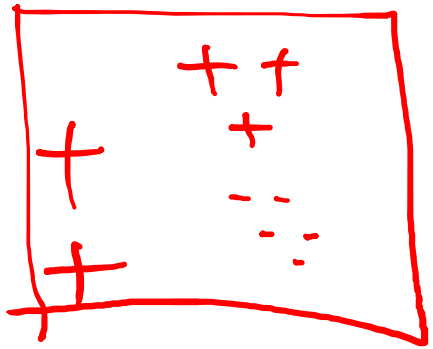
$$w_i \leftarrow w_i e^{2\alpha_2} \quad \forall \text{ mis-class.}$$

$$\leftarrow \frac{1}{T_0} e^{\log 6} = \frac{6}{T_0} = \frac{3}{5}$$

for round 3

D





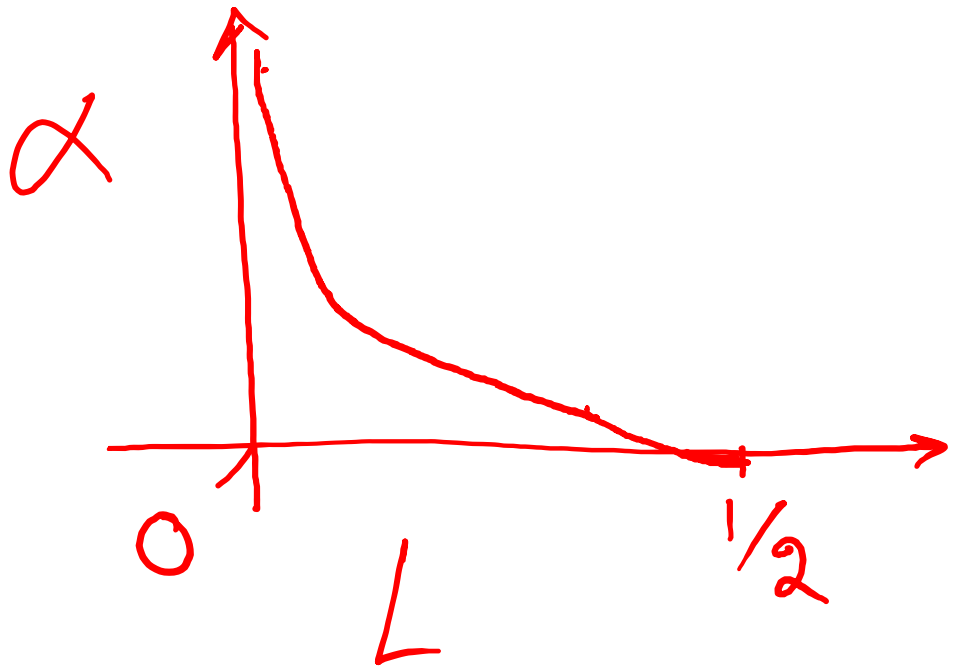
$$f(x) = \text{sign}\left(\alpha_1 h_1(x) + \alpha_2 h_2(x)\right)$$

$$= \text{sign}\left[\frac{1}{2}\log\frac{7}{3}\, h_1(x) + \frac{1}{2}\log 6\, h_2(x)\right]$$

$$= \text{sign}\left[\frac{1}{2}\log\frac{7}{3}\,\text{sign}(x_1 - c) + \frac{1}{2}\log 6\,\text{sign}(x_2 - d)\right]$$

$f(x) \rightarrow$ final classifier.

for a test sample $x^* = (x_1^*, x_2^*)$



$$\alpha : \frac{1}{g} \log \frac{1-L}{L}$$

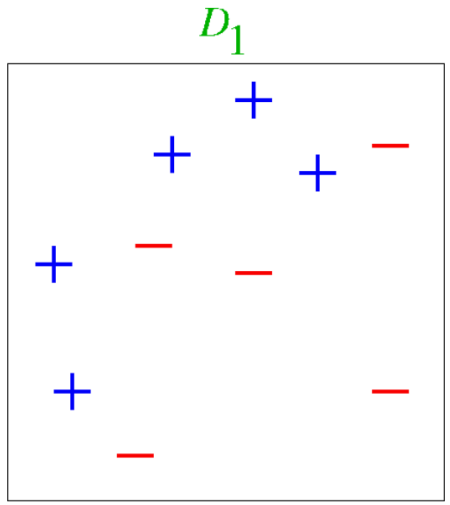
Preliminaries

- The misclassification rate $\frac{1}{N}\sum_{n=1}^{N}\mathbb{I}(h(x^n) \neq t^n)$ weighs each training example equally.
- x^n is the nth example, t^n is the label for x^n and h(.) is the classifier
- Key idea: we can learn a classifier using different costs (aka weights) for example. Classifier "tries harder" on examples with higher cost
- Change cost function: $\sum_{n=1}^{N} w^n \mathbb{I}(h(x^n) \neq t^n) \quad \sum w^n = 1$

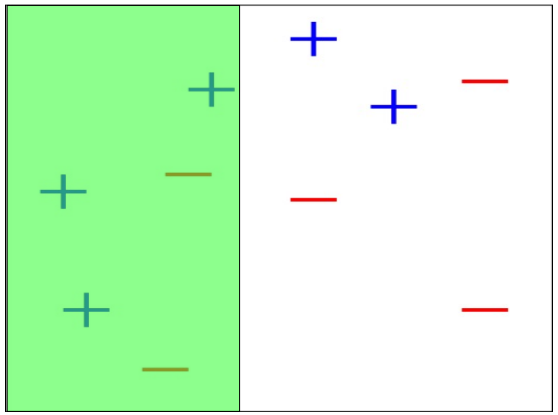
- Boosting uses weak learners.
- (Informal) Weak learner is a learning algorithm that outputs a hypothesis (e.g., a classifier) that performs slightly better than chance, e.g., it predicts the correct label with probability 0.51 in binary label case.
  - It gets slightly less than 0.5 error rate (the worst case is 0.5)
- We are interested in weak learners that are computationally efficient.
  - Decision trees
  - Even simpler: Decision Stump: A decision tree with a single split

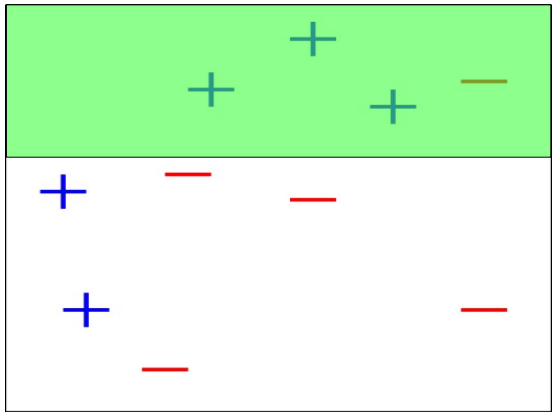
Weak Classifiers

These weak classifiers, which are decision stumps, consist of the set of horizontal and vertical half spaces.



2D data





Decision stump

AdaBoost.M1 (Adaptive Boosting) – learn from mistakes (Freud and Schapire'97)

- Boosting: Train classifiers sequentially, each time assigning higher weight to training data points that were previously misclassified.
- Key steps of AdaBoost:
  - At each iteration we re-weight the training samples by assigning larger weights to samples (i.e., data points) that were classified incorrectly.
  - We train a new weak classifier based on the re-weighted samples.
  - We add this weak classifier to the ensemble of weak classifiers. This ensemble is our new classifier.
  - We repeat the process many times.
- The weak learner needs to minimize weighted error.
- AdaBoost reduces bias by making each classifier focus on previous mistakes.

A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting*,
Yoav Freund and Robert E. Schapire, AT&T Labs, 180 Park Avenue, Florham Park, New Jersey 07932
2003 Gödel Prize winner

Abstract

Assumptions

- Input: Data $\mathrm{D} = \{x^n, t^n\}, t^n \in \{-1,1\}$
- A classifier or hypothesis $\mathcal{H}$: x → {-1,1}
- 0-1 loss $\mathbb{I}(h(x^n) \neq t^n) = 0.5(1 - \mathrm{h}(x^n)t^n)$

Algorithm

- Input: Data D, weak classifier (a classification procedure that returns a classifier h, e.g. best decision stump, from a set of classifiers $\mathcal{H}$, e.g. all possible decision stumps), number of iterations T
- Output: Classifier f(x)
- Initialize sample weights: uniform weights = 1/N
- For m = 1…T
  - Fit a classifier to data using weighted samples

$$h_t \leftarrow argmin_{h \in H} \sum_{n=1}^{N} w^n \mathbb{I}(h(x^n) \neq t^n)$$

Contd.

- Compute weighted error $err_m = \frac{\sum_{n=1}^{N} w^n \mathbb{I}(h(x^n) \neq t^n)}{\sum w^n}$
- Compute classifier coefficient $\alpha_m = 0.5 \log \frac{1-err_m}{err_m}$
- Update data weights $w^n \leftarrow w^n e^{-\alpha_m t^n h_m(x^n)} \equiv w^n e^{2\alpha_m \mathbb{I}(h(x^n) \neq t^n)}$
- $f(x) = sign(\sum_{m=1}^{T} \alpha_m h_m(x))$

AdaBoost.

*) $D = \{x_i, y_i\}_{i=1}^{n}$,

$w_i \leftarrow$ weight of x_i, initialize with $1/n$

*) For any ~~stage~~ 'j'

$$h_j \leftarrow \text{argmin} \frac{\sum_{i=1}^{n} w_i I(y_i \neq h_j(x_i))}{\sum_{i=1}^{n} w_i}$$

$$\alpha_j \leftarrow \log\left(\frac{1-L_j}{L_j}\right)$$

$$w_i \leftarrow w_i e^{\alpha_j I(y_i \neq h_j(x_i))}$$

$$f(x) = \sum_{j=1}^{m} \beta_j h_j(x), \quad \beta_j = \frac{\alpha_j}{2}$$

Decision: $\underbrace{\text{sign}\left(f(x)\right)}_{f(x)}$

*) Say loss function $L(y, f(x))$ $\quad f(x) \in \{-1, 1\}$

~~Ly~~

X $\quad \underset{\beta}{\operatorname{argmin}}\, L\left(y, \operatorname{sign}\left(\sum_{j=1}^{m} \beta_j h_j(x)\right)\right)$

For a stage 'j', we know everything till 'j-1' and we need to learn the parameters for stage 'j' only.

$$L\left(y, f_{j-1} + \beta_j h_j(x)\right)$$

f_{j-1} is known, β_j & $h_j(x)$ are to be learnt

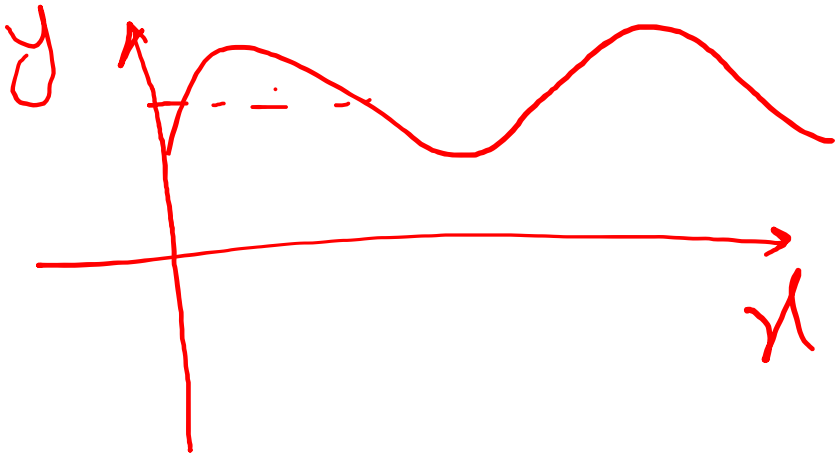
$$L(y, f(x)) = (y - f(x))^2$$

$$\min_{\beta_j, h_j} \left(y - \underbrace{f_{j-1}(x) - \beta_j h_j(x)}\right)^2$$

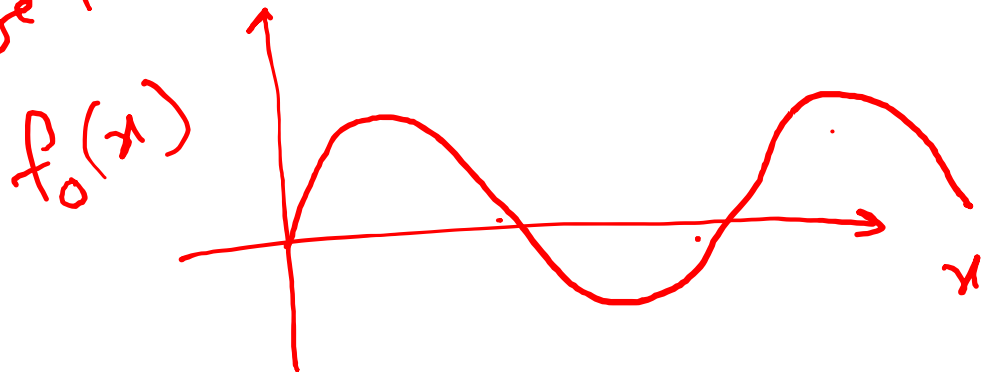
we fit $f_{j-1}(x)$ to the data,

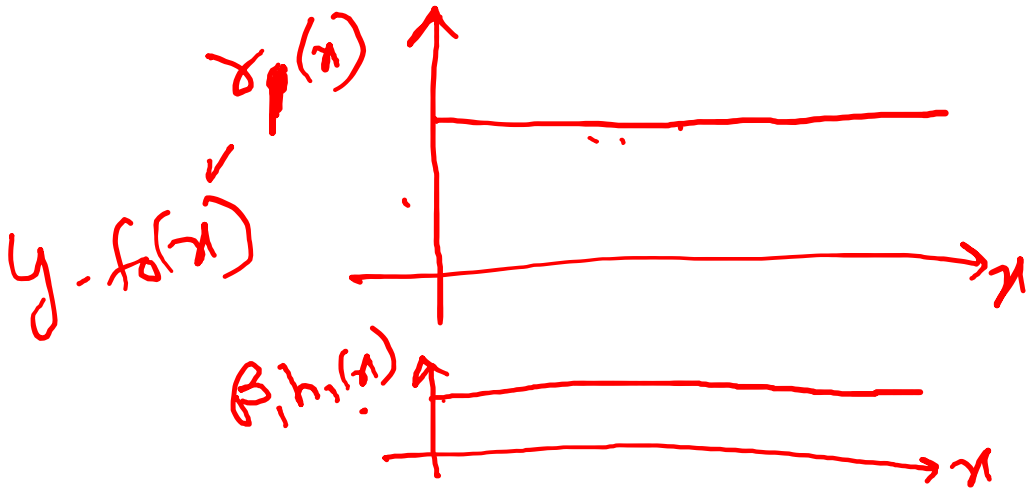
we get some residue $\gamma_j = y - f_{j-1}(x)$

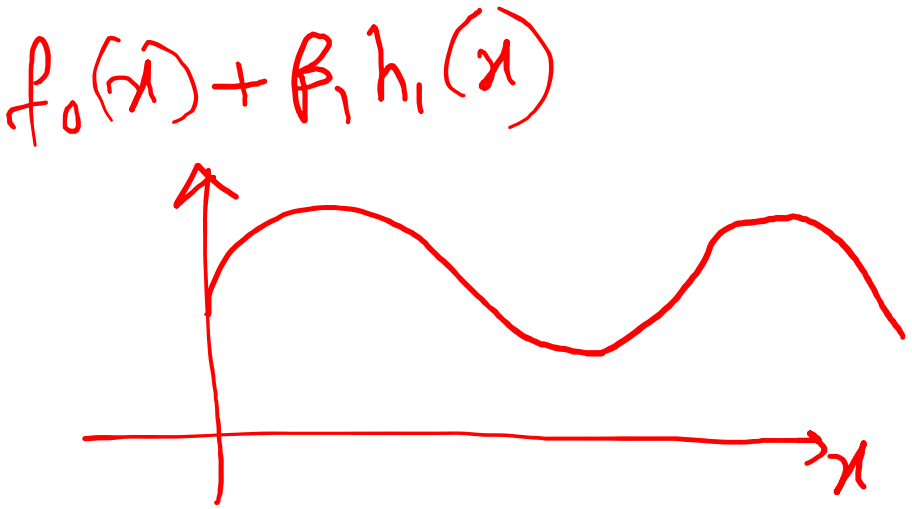
we augment $f_{j-1}(x)$ with $\beta_j h_j(x)$, which fit the residual γ_j



stage 1







$$L(y, f(x)) = e^{-y f(x)}$$

$y = f(x) \rightarrow L = e^{-1}$

$y \neq f(x) \rightarrow L = e^{1}$

$f(x) \in \{-1, 1\}, \; y \in \{-1, 1\}$

$h_j(x_i) \in \{-1, 1\}$

For 'n' points, $L(y, f(x)) = \sum_{i=1}^{n} e^{-y_i f(x_i)}$

$$f(x_i) = f_{j-1}(x_i) + \beta_j h_j(x_i)$$

$$L(y, f(x)) = \sum_{i=1}^{n} e^{-y_i \left[f_{j-1}(x_i) + \beta_j h_j(x_i) \right]}$$

$$L(y, f(x)) = \sum_{i=1}^{n} e^{-y_i f_{j-1}(x_i)} e^{-y_i \beta_j h_j(x_i)} \qquad a+b=1, \quad b=1-a$$

$$= \sum_{i=1}^{n} w_i e^{-y_i \beta_j h_j(x_i)}$$

$$= \sum_{y_i = h_j(x_i)} w_i e^{-\beta_j} + \sum_{y_i \neq h_j(x_i)} w_i e^{\beta_j}$$

$$= \sum_{i=1}^{n} w_i e^{-\beta_j} I\left[y_i = h_j(x_i)\right] + \sum_{i=1}^{n} w_i e^{\beta_j} I\left[y_i \neq h_j(x_i)\right]$$

$$= \sum_{j=1} e^{-\beta_j} \left\{ \sum_{i=1}^{n} w_i - \sum_{i=1}^{n} w_i I\left[y \neq h_j(x_i)\right] \right\} + \sum_{i=1}^{n} w_i e^{\beta_j} I\left[y_i \neq h_j(x_i)\right]$$

$$\nabla_{\beta} L = -e^{-\beta_j} \left[\sum_{i=1}^{n} w_i - \sum_{i=1}^{n} w_i I\left(y_i \neq h_j(x_i)\right) \right]$$

$$+ e^{\beta_j} \sum_{i=1}^{n} w_i I\left(y_i \neq h_j(x_i)\right) = 0$$

$$e^{\beta_j} \sum_{i=1}^{n} w_i I\left(y_i \neq h_j(x_i)\right) + e^{-\beta_j} \sum_{i=1}^{n} w_i I\left(y_i \neq h_j(x_i)\right)$$

$$= e^{-\beta_j} \sum_{i=1}^{n} w_i$$

$$\left(e^{\beta_j} + e^{-\beta_j}\right) \sum_{i=1}^{n} w_i I(\cdot) = e^{-\beta_j} \sum_{i=1}^{n} w_i$$

$$\frac{e^{-\beta_j}}{e^{\beta_j} + e^{-\beta_j}} : \frac{\sum_{i=1}^{n} w_i I(\cdot)}{\sum_{i=1}^{n} w_i} = l_j$$

$$\frac{1 - l_j}{l_j} = \frac{1 - \frac{e^{-\beta_j}}{e^{-\beta_j} + e^{\beta_j}}}{\frac{e^{-\beta_j}}{e^{\beta_j} + e^{-\beta_j}}} = e^{2\beta_j}$$

$$\beta_j = \frac{1}{2} \log\left(\frac{1 - l_j}{l_j}\right)$$

$$= \alpha_j / 2$$

$$\sum_{i=1}^{n} w_i e^{-y_i \beta_j h_j(x_i)} \leftarrow \text{for stage 'j'}$$

$$\sum_{i=1}^{n} e^{-y_i f_{j-1}(x_i)} e^{-y_i \beta_j h_j(x_i)} \leftarrow \text{for stage 'j'}$$

$$\sum_{i=1}^{n} e^{-y_i f_j(x_i)} e^{-y_i \beta_{j+1} h_{j+1}(x_i)} \leftarrow \text{for stage 'j+1'}$$

$$e^{-y_i f_j(x_i)} = e^{-y_i [f_{j-1}(x_i) + \beta_j h_j(x_i)]}$$

$$= w_i e^{-\beta_j y_i h_j(x_i)}$$

$$-y_i h_j(x_i) = 2\,I\left(y_i \neq h_j(x_i)\right) - 1$$

$$w_i e^{-\beta_j y_i h_j(x_i)} = w_i e^{\beta_j \left[2\,I\left(y_i \neq h_j(x_i)\right) - 1\right]}$$

$$= \underbrace{w_i e^{\alpha_j I\left(y_i \neq h_j(x_i)\right)}}\; e^{-\beta_j}$$

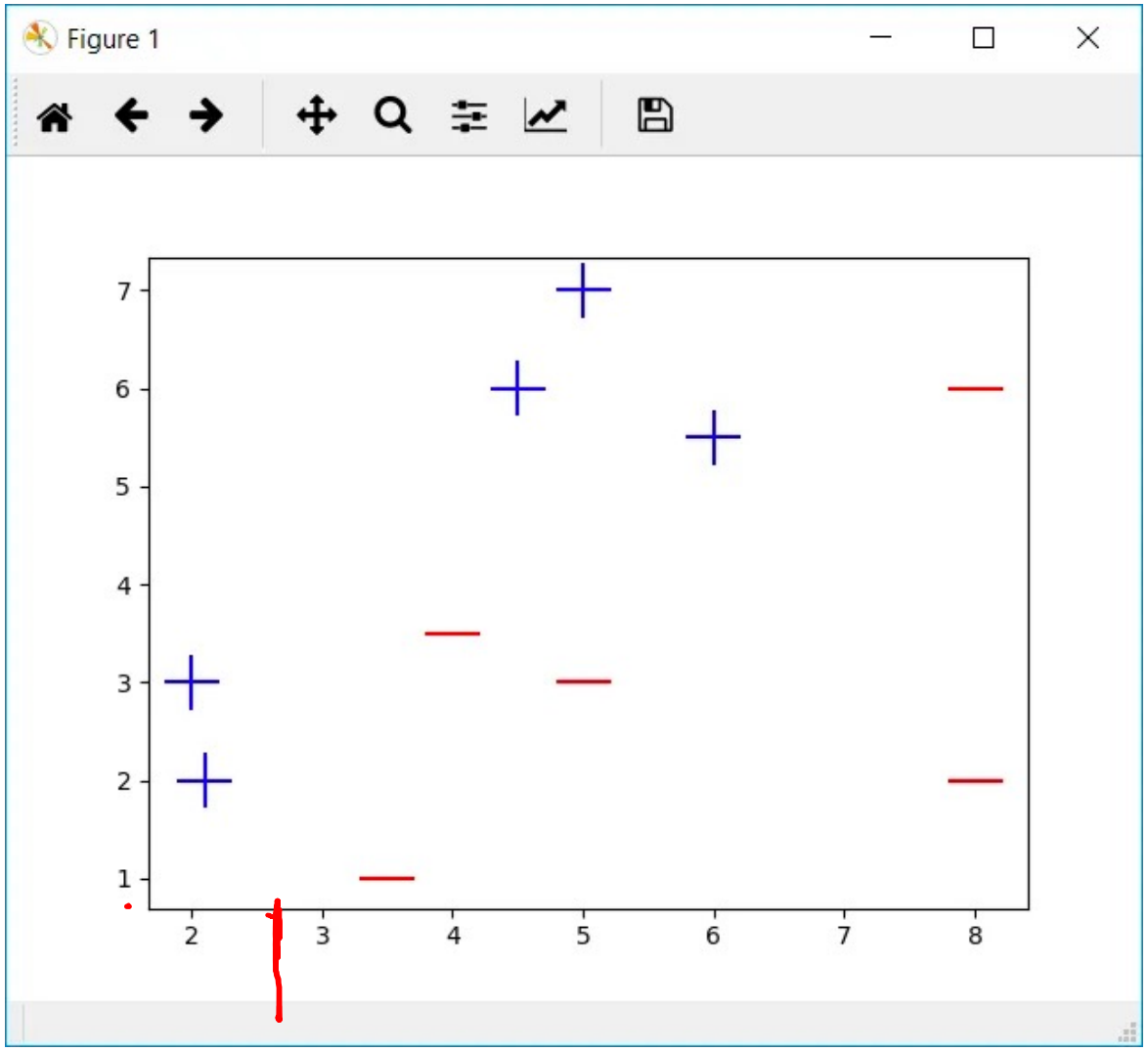
X

Numerical example

Data

https://sefiks.com/2018/11/02/a-step-by-step-adaboost-example/

x1,x2, label
2,3,1
2.1,2,1
4.5,6,1
4,3.5,-1
3.5,1,-1
5,7,1
5,3,-1
6,5.5,1
8,6,-1
8,2,-1



x1	x2	actual	weight	weighted_actual
2	3	1	0.1	0.1
2	2	1	0.1	0.1
4	6	1	0.1	0.1
4	3	-1	0.1	-0.1
4	1	-1	0.1	-0.1
5	7	1	0.1	0.1
5	3	-1	0.1	-0.1
6	5	1	0.1	0.1
8	6	-1	0.1	-0.1
8	2	-1	0.1	-0.1

All have uniform weights

x1	x2	actual G.T	weight 1/n	prediction	loss 0	weight * loss
2	3	1	0.1	1	0	0
2	2	1	0.1	1	0	0
4	6	1	0.1	-1	1	0.1
4	3	-1	0.1	-1	0	0
4	1	-1	0.1	-1	0	0
5	7	1	0.1	-1	1	0.1
5	3	-1	0.1	-1	0	0
6	5	1	0.1	-1	1	0.1
8	6	-1	0.1	-1	0	0
8	2	-1	0.1	-1	0	0

Error is 0.3 in this case.

Here, we'll define a new variable beta.

beta = ln[(1-L)/L] / 2
= ln[(1 – 0.3)/0.3] / 2

beta = 0.42

I(y != h(x)) is denoted by loss here

$$L_j = D_j = \frac{\sum_i w_i I(y_i \neq h_j(x_i))}{\sum_i w_i}$$

- We'll use alpha = 2*beta to update weights in the next round.
- $w_{i+1} = w_i$ * .exp(alpha * I(yi != h(xi))) where i refers to instance number.
- $w_{i+1} = w_i$ * .exp(2*beta * I(yi != h(xi)))

x1	x2	actual	weight	prediction	w_(i+1)	norm(w_(i+1))
2	3	1	0.1	1	0.1	0.071
2	2	1	0.1	1	0.1	0.071
4	6	1	0.1	-1	0.23	0.167
4	3	-1	0.1	-1	0.1	0.071
4	1	-1	0.1	-1	0.1	0.071
5	7	1	0.1	-1	0.23	0.167
5	3	-1	0.1	-1	0.1	0.071
6	5	1	0.1	-1	0.23	0.167
8	6	-1	0.1	-1	0.1	0.071
8	2	-1	0.1	-1	0.1	0.071

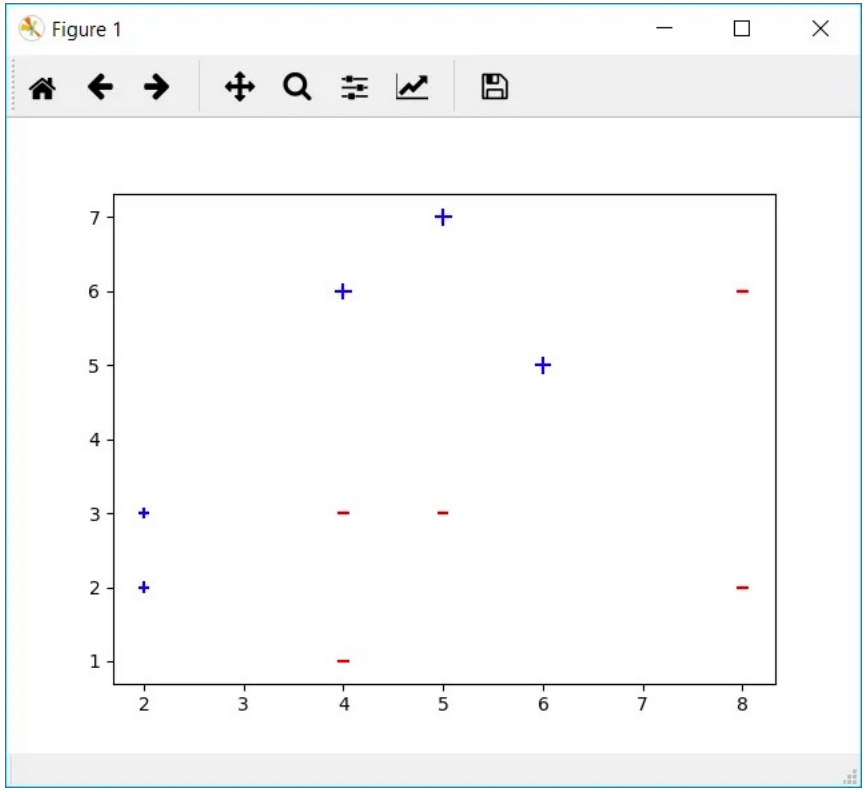
$w_{i+1} \leftarrow \frac{1}{10} e(0.84)$

Round 2

- shift normalized w_(i+1) column to weight column in this round. Then, build a decision stump. Still, x1 and x2 are features.

x1	x2	actual	weight
2	3	1	0.1
2	2	1	0.1
4	6	1	0.23
4	3	-1	0.1
4	1	-1	0.1
5	7	1	0.23
5	3	-1	0.1
6	5	1	0.23
8	6	-1	0.1
8	2	-1	0.1

- Graph of the new data set is demonstrated below. Weights of correct classified ones decreased whereas incorrect ones increased.



	x2	actual	weight	prediction	loss	weight * loss
	3	1	0.1	-1	1	0.1
	2	1	0.1	-1	1	0.1
	6	1	0.23	1	0	0
	3	-1	0.1	-1	0	0
	1	-1	0.1	-1	0	0
	7	1	0.23	1	0	0
	3	-1	0.1	-1	0	0
	5	1	0.23	1	0	0
	6	-1	0.1	1	1	0.1
	2	-1	0.1	-1	0	0

- calculate error and alpha values for round 2.
- L = 0.3/1.39, beta =0.525
- So, weights for the following round can be found.

$$\frac{\sum_i w_i I(\cdot)}{\sum_i w_i}$$

x1	x2	actual	weight	prediction	w_(i+1)	norm(w_(i+1))
2	3	1	0.1	-1	0.29	0.167
2	2	1	0.1	-1	0.29	0.167
4	6	1	0.23	1	0.23	0.106
4	3	-1	0.1	-1	0.1	0.045
4	1	-1	0.1	-1	0.1	0.045
5	7	1	0.23	1	0.23	0.106
5	3	-1	0.1	-1	0.1	0.045
6	5	1	0.23	1	0.23	0.106
8	6	-1	0.1	1	0.29	0.167
8	2	-1	0.1	-1	0.1	0.045

$0.1 \exp(1.04)$

Round 3

- L = $\frac{sum\ of\ w*loss}{sum\ of\ w}$
- L = 0.35
- Beta = 0.62

x1	x2	actual	weight	prediction	loss	w * loss	w_(i+1)	norm(w_(i+1))
2	3	1	0.29	1	0	0		
2	2	1	0.29	1	0	0		
4	6	1	0.23	-1	1	0.23		
4	3	-1	0.1	-1	0	0		
4	1	-1	0.1	-1	0	0		
5	7	1	0.23	-1	1	0.23		
5	3	-1	0.1	-1	0	0		
6	5	1	0.23	-1	1	0.23		
8	6	-1	0.29	-1	0	0		
8	2	-1	0.1	-1	0	0		

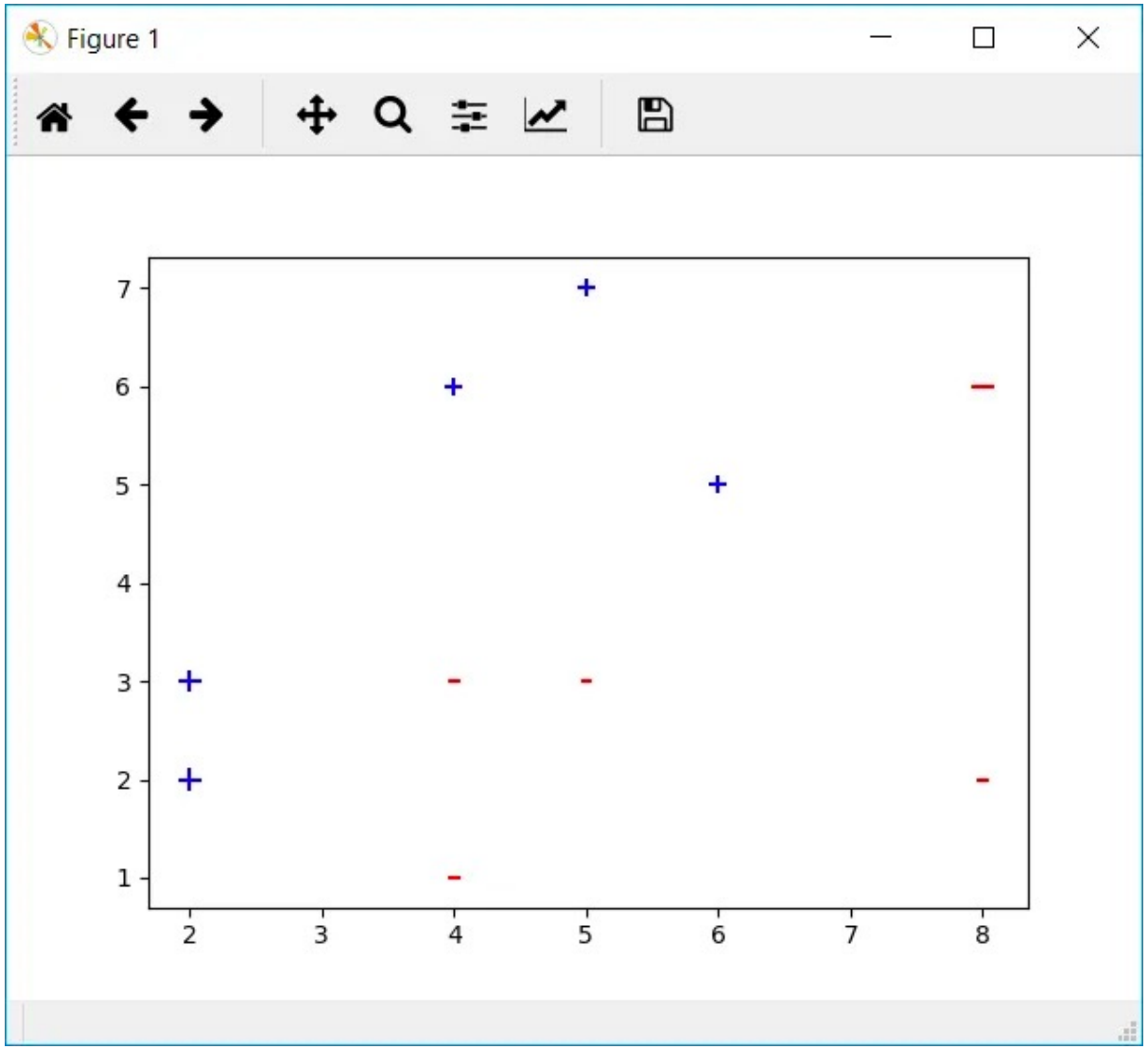


Figure 1
7
6
5
4
3
2
1
2
3
4
5
6
7
8

Round 4

- L = 0.15
- beta = 1.17

Prediction

- Cumulative sum of each round's beta times prediction gives the final prediction
- For example, prediction of the 1st instance will be

0.42 x 1 + 0.525 x (-1) + 0.62 x 1 + 1.17x1 = 1.58

- And we will apply sign function

- Sign(01.58) = +1 which is correctly classified.

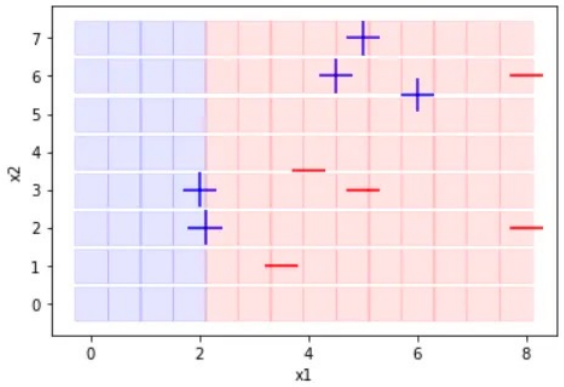
round 1 alpha	round 2 alpha	round 3 alpha	round 4 alpha
0.42	0.525	0.62	1.17
round 1 prediction	**round 2 prediction**	**round 3 prediction**	**round 4 prediction**
1	-1	1	1
1	-1	1	1
-1	1	-1	1
-1	-1	-1	1
-1	-1	-1	1
-1	1	-1	1
-1	-1	-1	1
-1	1	-1	1
-1	1	-1	-1
-1	-1	-1	-1

Weak learners

β_1

0.4236489301936017

x



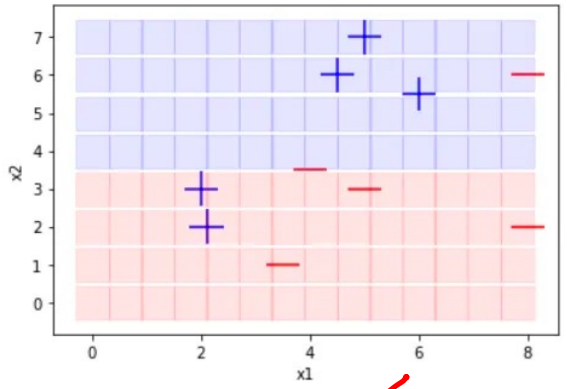
Loss = 0.3

+

$0.525 = \beta_2$

0.6496414920651304

x



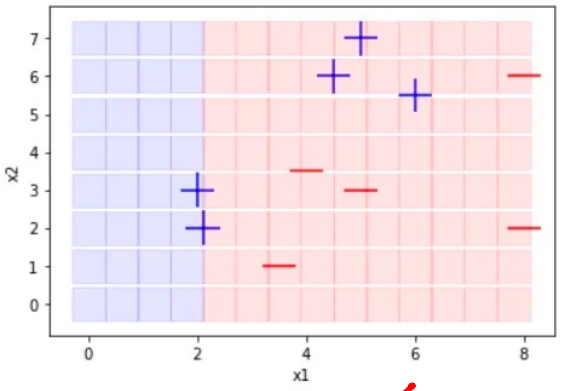
Loss = 0.21

+

$\beta_3 = 0.62$

0.38107902602344847

x



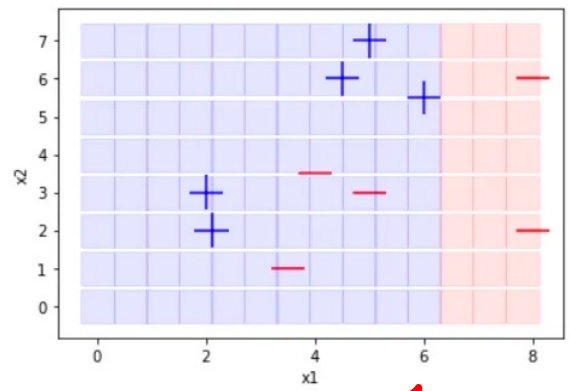
Loss = 0.31

+

$\beta_4 = 1.17$

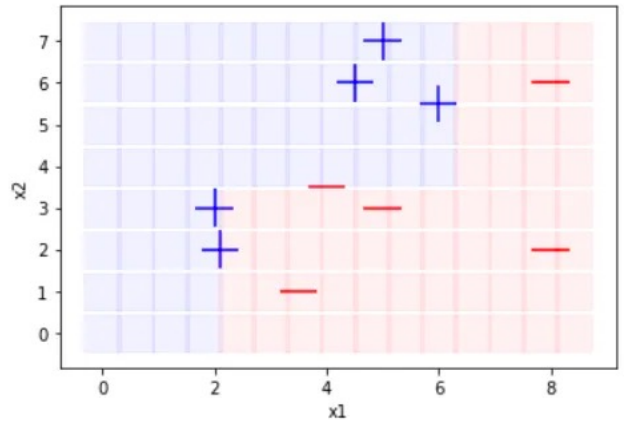
1.0986122886681096

x



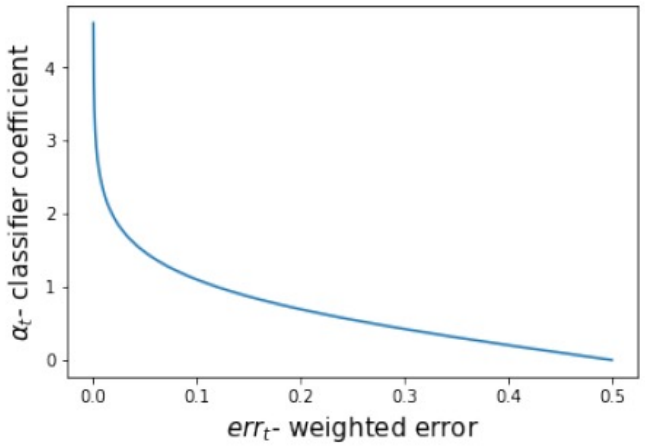
Loss = 0.1

Final decision boundary



Weighting Intuition

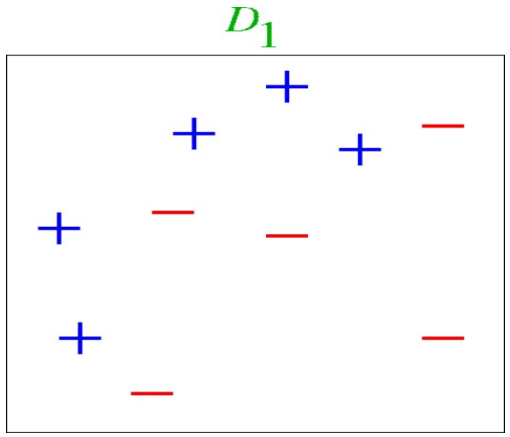
- Weak classifiers which get lower weighted error get more weight in the final classifier



- Also: $w^n e^{2\alpha_m \mathbb{I}(h(x^n) \neq t^n)}$
- If err_m tends to 0, α_m high so misclassified examples get more attention
- If err_m tends to 0.5, α_m low so misclassified examples are not emphasized

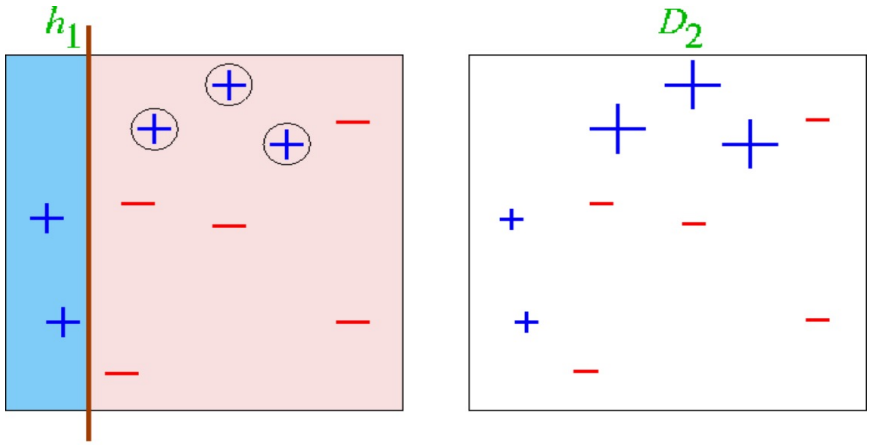
Example

- Training data



- $\mathcal{H}$: decision trees with a single split (decision stumps)

- Round 1

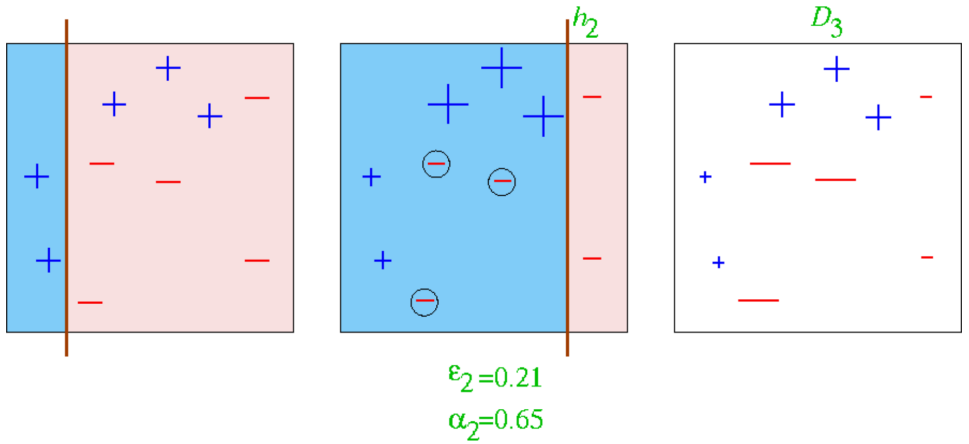


- W = (1/10 …1/10) → Train a classifier (using w))

$$err_1 = \frac{\sum_{n=1}^{10} w^n \mathbb{I}(h(x^n) \neq t^n)}{\sum w^n} = 3/10 \qquad \alpha_1 = 0.5 \log \frac{1-.3}{.3} \approx .42$$

$$f(x) = sign(\alpha_1 h_1(x))$$

- Round 2

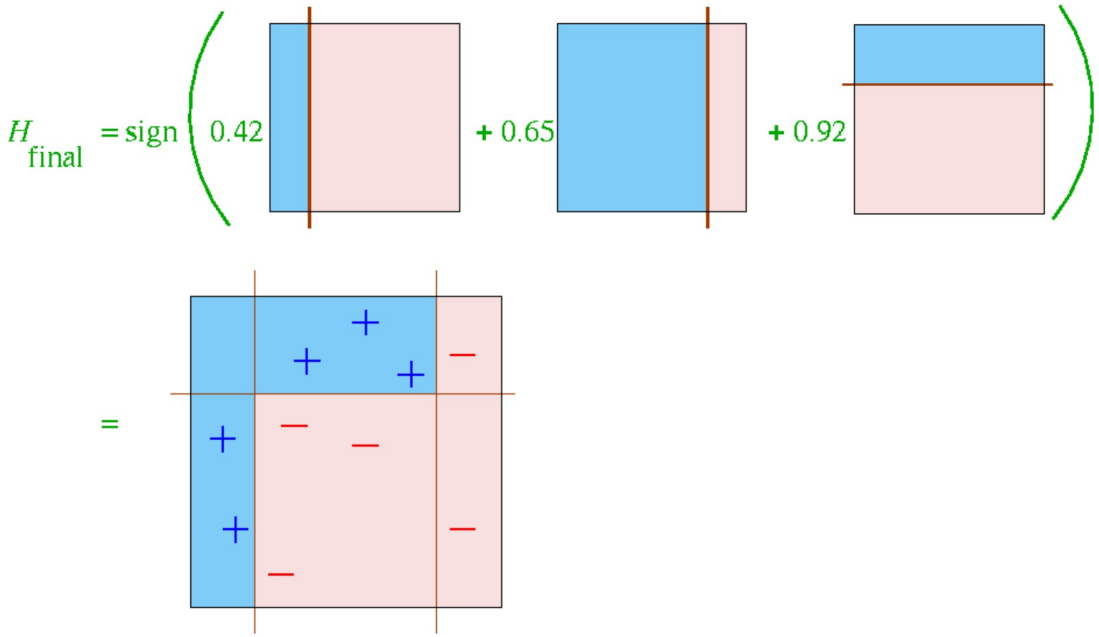


- new weights w; Train a classifier (using new w)) err2

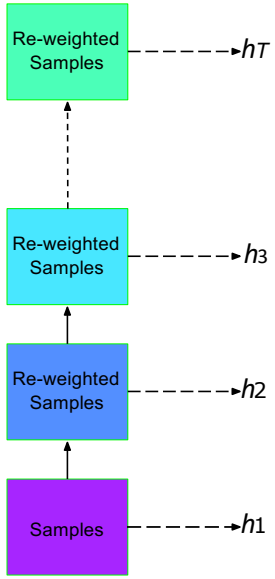
$$err_2 = \frac{\sum_{n=1}^{10} w^n \mathbb{I}(h(x^n) \neq t^n)}{\sum w^n} = .21 \quad \alpha_1 \approx .0.66$$

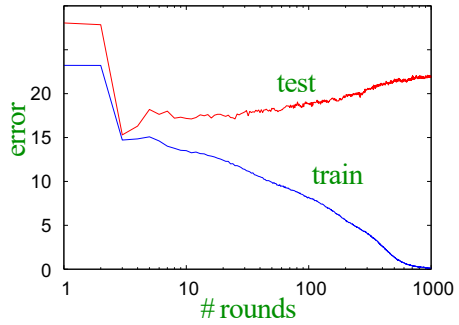
$$f(x) = sign(\alpha_1 h_1(x) + \alpha_2 h_2(x))$$

Final classifier

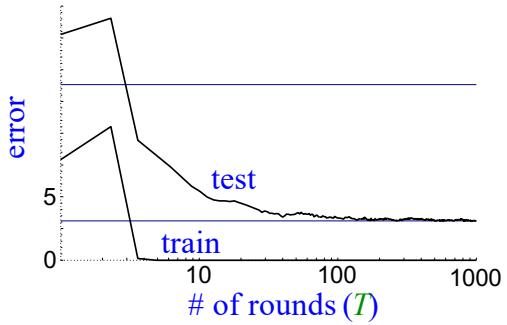


Algo and generalization error





If one runs AdaBoost long enough, it can in fact overfit.



But often it does not!

Sometimes the test error decreases even after the training error is zero!

Theory

ADDITIVE LOGISTIC REGRESSION: A STATISTICAL VIEW OF BOOSTING

By Jerome Friedman, Trevor Hastie, and Robert Tibshirani,

Stanford University

Additive Models

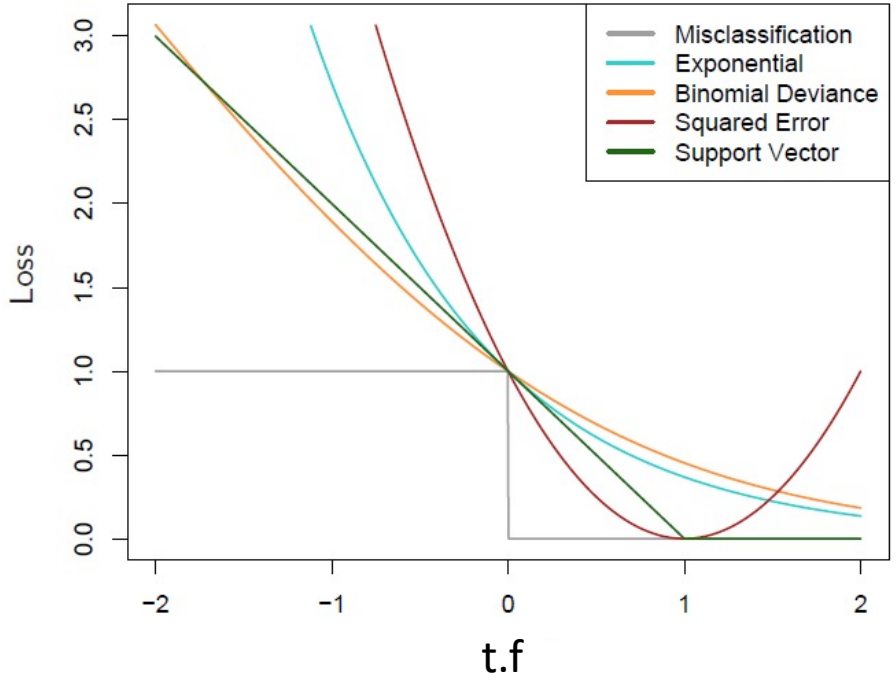
- Next, we'll now interpret AdaBoost as a way of fitting an additive model.
- An additive model with m terms is given by
- $H_m(x) = \sum_{i=1}^{m} \alpha_i h_i(x)$
- Observe that we're taking a linear combination of base classifiers $h_i(x)$, just like in boosting.
- The goal is to show how to learn optimal h, α and update of weights for any stage.

Stagewise Training of Additive Models

- A greedy approach to fitting additive models, known as stagewise training:
- Initialize $H_0(x) = 0$
- For m = 1 to T:
- Compute the m-th hypothesis $H_m(x) = H_{m-1}(x) + \alpha_m h_m$ i.e. h_m and α_m assuming previous additive model H_{m-1} is fixed:
- $(h_m, \alpha_m) \leftarrow argmin_{h \in \mathcal{H}, \alpha} \sum_{i=1}^{N} L(H_{m-1}(x^i) + \alpha h(x^i), t^i)$
- Add it to the additive model $H_m = H_{m-1} + \alpha_m h_m$

Additive Models with Exponential Loss

- Consider the exponential loss $L(f,t) = \exp(-ft)$
- Loss functions for two-class classification. The response is t = ±1; the prediction is f, with class prediction sign(f). The losses are misclassification: Indicator(sign(f) ≠ t); exponential: exp(−tf); binomial deviance: log(1 + exp(−2tf)); squared error: (t − f)^2; and support vector: $(1 - tf)_+$ Each function has been scaled so that it passes through the point(0, 1).



- $(h_m, \alpha_m) \leftarrow argmin_{h \in \mathcal{H}, \alpha} \sum_{i=1}^{N} L(H_{m-1}(x^i) + \alpha h(x^i), t^i)$
- $(h_m, \alpha_m) \leftarrow argmin_{h \in \mathcal{H}, \alpha} \sum_{i=1}^{N} \exp\{-H_{m-1}(x^i)t^i - \alpha h(x^i)t^i\}$
- $= \sum_{i=1}^{N} \exp\{-H_{m-1}(x^i)t^i\} \exp\{-\alpha h(x^i)t^i\}$
- $= \sum_{i=1}^{N} w_m^i \exp\{-\alpha h(x^i)t^i\}$

$w_m^i = \exp\{-H_{m-1}(x^i)t^i\}$

- We want to solve the following minimization problem:

$$(h_m, \alpha_m) \leftarrow argmin_{h \in \mathcal{H}, \alpha} \sum_{i=1}^{N} w_m^i \exp\{-\alpha h(x^i)t^i\}$$

- If $h(x^n) = t^n$ we have exp(- α $h(x^n)t^n$)=exp(- α), otherwise exp(α).

  $\sum_{i=1}^{N} w_m^i \exp\{-\alpha h(x^i)t^i\}$

  = exp(- α) $\sum_{i=1}^{N} w_m^i \mathbb{I}(h(x^n) = t^n)$ + exp(α) $\sum_{i=1}^{N} w_m^i \mathbb{I}(h(x^n) \neq t^n)$

=

$$\exp(-\alpha) \sum_{i=1}^{N} w_m^i \mathbb{I}(h(x^n) = t^n) + \exp(\alpha) \sum_{i=1}^{N} w_m^i \mathbb{I}(h(x^n) \neq t^n) - \exp(-\alpha) \sum_{i=1}^{N} w_m^i \mathbb{I}(h(x^n) \neq t^n)\} + \exp(-\alpha) \sum_{i=1}^{N} w_m^i \mathbb{I}(h(x^n) \neq t^n)$$

Combine 1st and last, and, 2nd and 3rd

- = {exp(α)−exp(− α)} $\sum_{i=1}^{N} w_m^i \mathbb{I}(h(x^n) \neq t^n)$ + exp(− α) [$\sum_{i=1}^{N} w_m^i [\mathbb{I}(h(x^n) \neq t^n) + \mathbb{I}(h(x^n) = t^n)]$

- The second term includes all samples, so essentially is summation of weights only, which is independent of h.
- So we can optimize only the first term

$$argmin_{h \in \mathcal{H}} \sum_{i=1}^{N} w_m^i \mathbb{I}(h(x^n) \neq t^n)$$

- Thus the classifier that minimizes the loss is the solution

- Suppose hm is the solution, then
- We can plug in hm and find α by taking derivative and setting to 0

$$[\{\exp(\alpha) - \exp(-\alpha)\} \sum_{i=1}^{N} w_m^i \mathbb{I}(h(x^n) \neq t^n)] + \exp(-\alpha) \sum_{i=1}^{N} w_m^i$$

$$= \{\exp(\alpha) - \exp(-\alpha)\} err_m \sum_{i=1}^{N} w_m^i + \exp(-\alpha) \sum_{i=1}^{N} w_m^i$$

$$err_m = \frac{\sum_{n=1}^{N} w^n \mathbb{I}(h(x^n) \neq t^n)}{\sum w^n}$$

$$w_{m+1}^{i} = \exp\{-H_m(x^i)t^i\} = \exp\{-H_{m-1}(x^i)t^i - \alpha_m h_m(x^i)t^i\}$$

$$= \exp\{-H_{m-1}(x^i)t^i\}\exp\{-\alpha_m h_m(x^i)t^i\}$$

$$= w_m^i \exp\{-\alpha_m h_m(x^i)t^i\}$$

- Finally $H_m(x) = H_{m-1}(x) + \alpha_m h_m$