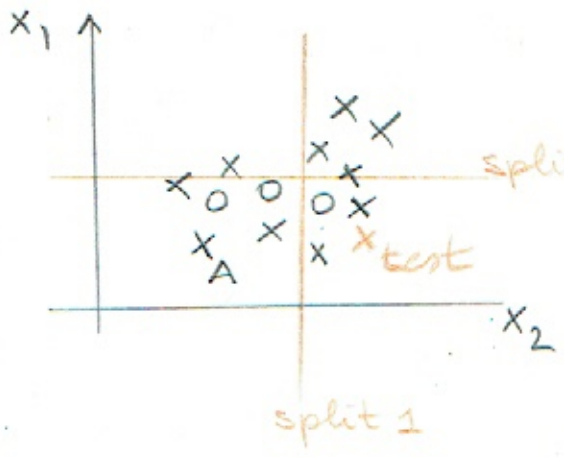
LESSON 11: DECISSION TREES, BAGGING, BOOSTING
ADABOOST, GRADIENT BOOSTING, XGBOOST
RANDOM FORESTS

KD Trees



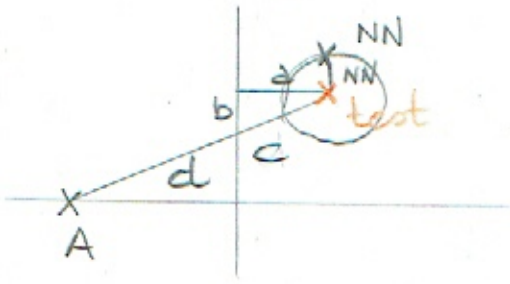
DATASET
↳ KNN classification
└ slow (calculate distances)
memory intensive

→ CAN we simplify ?
↳ YES

we will split data ALONG mean of AXIS WITH highest VARIANCE

→ apply KNN ON reduced dataset

CLAIM: POINT A IS further from X_{test} than X_{test}'s NN



$c^2 = a^2 + b^2 \rightarrow c = \sqrt{a^2+b^2}$

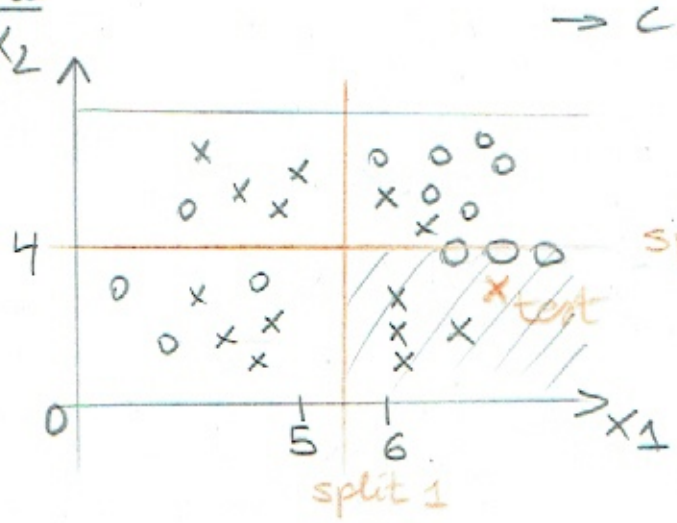
$\rightarrow c+d = \sqrt{a^2+b^2} + d$

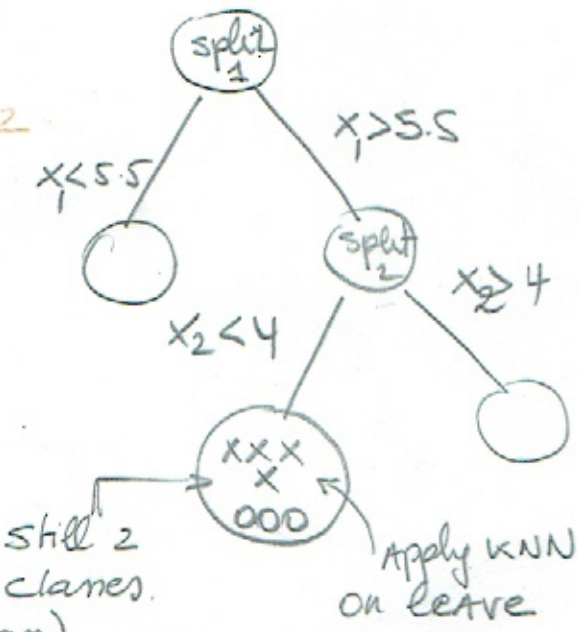
$\rightarrow c+d \geq \sqrt{a^2} + d$

$c+d \geq a+d$ $\quad a \geq NN$

$\rightarrow c+d \geq d \rightarrow \boxed{c+d \geq NN}$

EXAMPLE



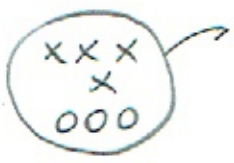


Define IMPURITY FUNCTIONS ⟸ | Ideally we have pure leaves (one class) ← still 2 classes.

IMPURITY < GINI / ENTROPY

↳ likelihood of classifying A POINT WTONG
QUALITY MEASURE FOR A SPLIT

GINI



$P(X_{test} = x) = 4/7$
$P(X_{test} = o) = 3/7$

S is set of labels of dataset → IN OUR CASE {x, o}

$S_k = \{(x,y) \in S \mid y = k\}$

$k = x \hookrightarrow S_x = \{(x,y) \in S \mid y = x\}$

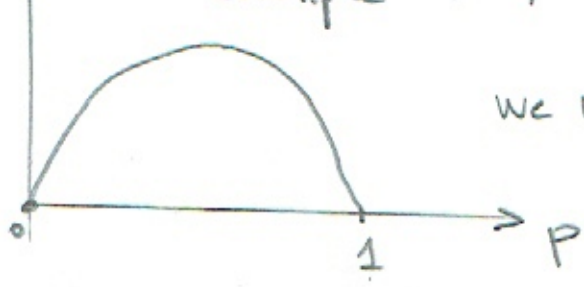
$P_k = \frac{S_k}{S}$

$S = S_1 \cup S_2 \cup .. \cup S_k$

$$G(S) = \sum P_k (1 - P_k)$$

if k=2 → probability of picking 1 class vs. another

↳ OUR EXAMPLE $\frac{4}{7} \cdot \frac{3}{7} + \frac{3}{7} \cdot \frac{4}{7} = 0.48$
↳ NOT VERY GOOD



we want value close to 1 OR 0

ENTROPY

↳ worst possible distribution IN A leaf IS when all classes are EQUALLY likely to be picked!

$d_1 = d_2 = .. = d_k = 1/k$ → WE DONT WANT d!!

KL-DIVERGENCE (Kullback-Leibler) : measure of relative similarity of 2 distributions

$$D_{KL}(P \| d) = \sum_{k=1}^{K} P_k \log \frac{P_k}{d_{k=1/k}}$$

↳ $1/k$

$$\rightarrow D_{KL}(p\|d) = \sum p_k \log p_k + \underbrace{\sum p_k \underbrace{\log(k)}_{c^t}}_{\log k \underbrace{\sum p_k}_{=1}}$$

$$\rightarrow D_{KL}(p\|d) = \sum p_k \log p_k + \log k$$

↳ drop when MAX/min as c^t

$$\longrightarrow \max_p D_{KL}\left(\sum p_k \log p_k\right)$$

$$\text{OR } \min_p D_{KL}\left(-\sum p_k \log p_k\right)$$

To be continued p6

Optional: ENTROPY - CROSS-ENTROPY - KL divergence

Supervised learning. → label y is known for TRAINING set

Regression

$C(w) = MSE$

$= \frac{1}{m} \sum (y_p - y)^2$

classification

$D \sim P(Y|X)$

↳ UNKNOWN

finding P is the essence of classification!

$C(w) = \cancel{MSE}$

$= -\log(\text{likelihood})$

↑ for logistic Regression

example : Random Variable X — A B C D (CAT, DOG.)

possible outcome

(animals pics)

1 training sample (picture of dog) → [0 0 0 1]

↳ distribution p

After training → model h. → [0.2 0.1 0.1 0.6]

↳ distribution q

How close is q to p ?

DEFINITION : INFORMATION content : IC

IC = 0 if info doesn't add value to what was known.

ex : $p(X=D) = 1$ ON TRAINING SAMPLE

$IC(x=D) \sim \frac{1}{P(X=D)} \rightarrow IC(X=D) = f(P(X=D))$

Assume 2 RVs X and Y ↳ whether ANIMAL is found in SINGAPORE OR NOT

X	Y
A	1
B	0
C	1
D	1

$IC(X \wedge Y) = IC(X) + IC(Y)$ if X, Y independent

$f(P(X=D))$ $f(P(Y=1))$

$f(P(X=D) \cdot P(Y=1)) = f(P(X=D)) + f(P(Y=1))$

=> f is log

$\rightarrow IC(X=D) = \log \frac{1}{P(X=D)} = -\log P(X=D)$

$\rightarrow$ **$IC\ p = -\log p$** for ANY distribution p

Expected VALUE of a random variable $\rightarrow E[x] = \sum_x p(X=x) \cdot x$

$E[IC] = -\log p(X=A) \cdot p(X=A) - \dots - \log p(X=D) \cdot p(X=D)$

ENTROPY $H(p) = -\sum_x P(X=x) \log p(X=x)$ (1)

If certainty $\rightarrow \log 1$
$\rightarrow 0$
$\rightarrow$ no contrib

CROSS-ENTROPY

1. UNknown distribution $P_y(Y|X) \rightarrow$ distr p
2. distribution q obtained throug training
3. $IC\ q = -\log q$

(1) $\rightarrow E[IC] = C(p||q) = -\sum_x p \log q(X=x)$

CROSS-ENTROPY $H(p||q) = -\sum_x p \log q$

cost FUNCTION
$\rightarrow$ MINIMIZE! $\rightarrow \frac{\partial}{\partial q_i} \overset{\text{set}}{=} 0$

WITH CONSTRAINT $\sum q_i = 1$

$\rightarrow \sum q_i - 1 = 0$

$$\frac{\partial}{\partial q_i} E[ic] = -\frac{p_i}{q_i} + \lambda \overset{\text{set}}{=} 0 \quad [\text{Lagrange}]$$

$$\rightarrow p_i = \lambda q_i \rightarrow \lambda = \frac{p_i}{q_i}$$

$$\rightarrow \boxed{p_i \approx q_i \text{ if } \lambda \approx 1}$$

λ measure for similarity of p and q

KL-divergence → relative entropy

$$D_{KL}(P \| Q) = H(P \| Q) - H(P)$$

↳ measure for comparing distributions p and q

$$H(P) = \sum p_i \log \frac{1}{p_i}$$

$$= \sum p_i \log \frac{q_i}{p_i} \cdot \frac{1}{q_i}$$

$$= \sum p_i \left(\log \frac{q_i}{p_i} - \log \frac{1}{q_i} \right)$$

$$= -\underbrace{\sum p_i \log \frac{p_i}{q_i}}_{\text{relative entropy}} + \underbrace{\sum p_i \log \frac{1}{q_i}}_{\text{CROSS-ENTROPY}}$$

→ describes mismatch between p and q.

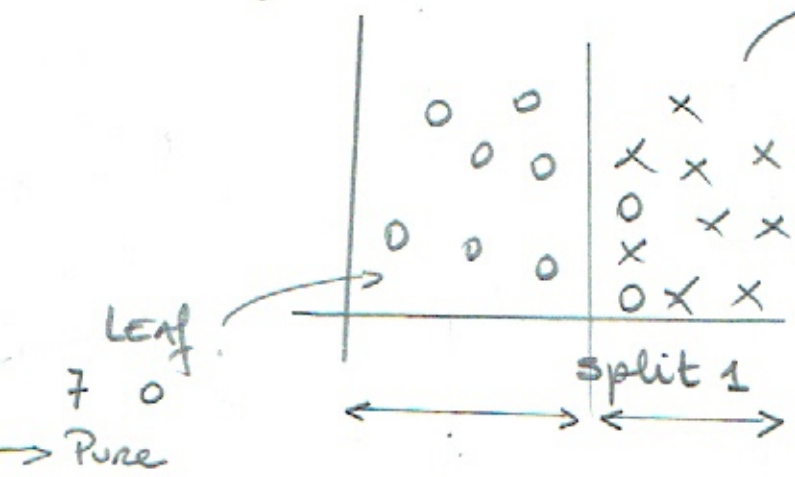
$$D_{KL}(P \| Q)$$

---

now we continue lesson from PAGE 3

$$\min_P D_{KL} \rightarrow \min_P \left(\underbrace{-\sum p_K \log p_K}_{H(P) \text{ ENTROPY}} \right)$$

ENTROPY OVER A TREE



LEAF 9 X
2 0

ENTROPY Tree
H(L) → H(R)
ENTROPY L — ENTROPY R

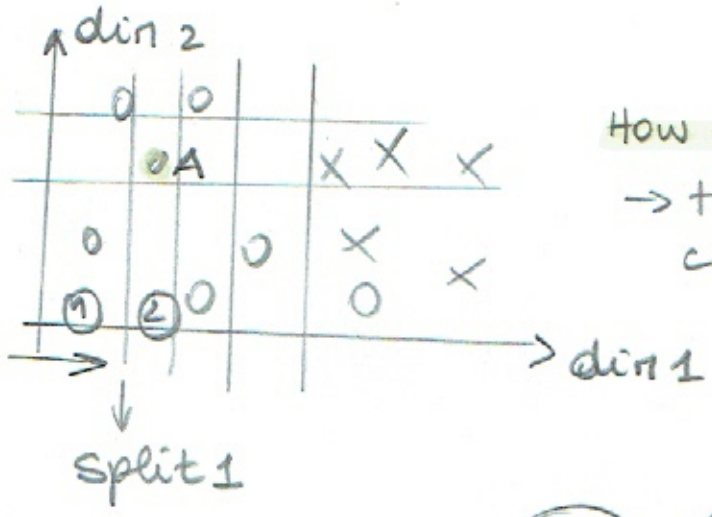
↓
more pure than R
↓
entropy lower

→ ENTROPY Tree = $H(L) \cdot p_L + H(R)\, p_R$

$H(S)$ — $p_L = \frac{S_L}{S}$ — $p_R = \frac{S_R}{R}$

$S_L = \{(x,y) \in S \mid y = L\}$ $\quad S_R = \{(x,y) \in S \mid y = R\}$

practical



How do you split ?
→ try them all !!
calculate ENTROPY
AND take lowest !

↓
CAN this be done more efficiently ?

YES

→ keep ALL calculations from previous split AND just take INTO ACCOUNT new point A

SITUATION 1

$H(L_1) \cdot P_{L_1} = \frac{S_{L1}}{S}$

SITUATION 2

$H(L_2) \cdot P_{L_2} = \frac{S_{L2}}{S}$

$S_{L2} = S_{L1} + 1$

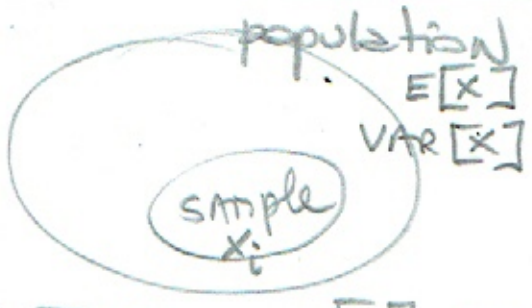
If you add x to the left side
→ H(L) will change

ENSEMBLE LEARNING — BAGGING → BREIMAN '96
BOOSTING

↳ group a set of learners in order to create a SUPER LEARNER!

↳ F. GALTON → weight of ox story

BAGGING: BOOTSTRap Aggregation
Can be done on any classifier WITH A VARIANCE problem



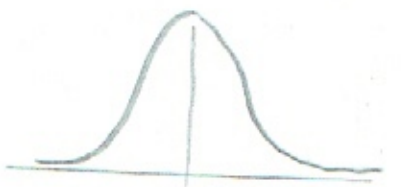
COIN TOSS → $E[x] = 50\%$

if $n \to \infty$

$E[\bar{x}] \to E[x]$

$\bar{x} = \frac{x_1 + \cdots + x_n}{n}$ — WLLN

x_i: 100 COIN TOSSES

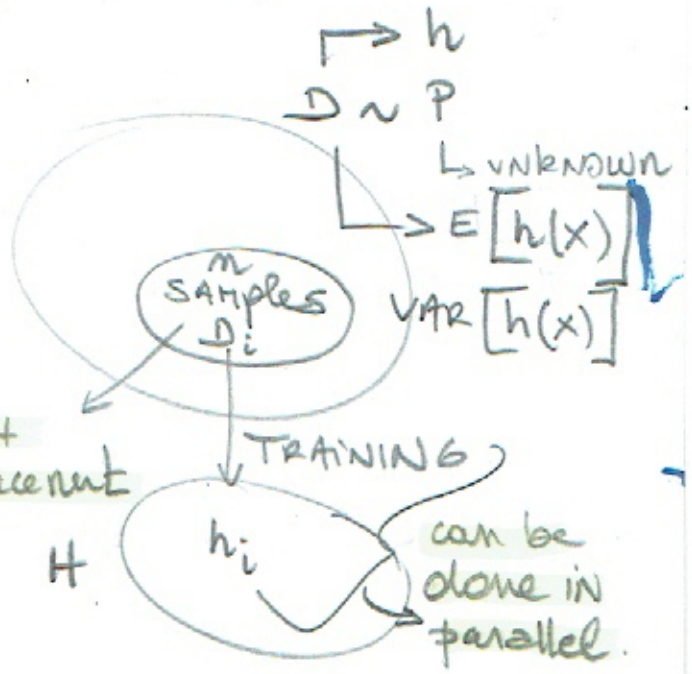


$\mathcal{N}(\mu, \sigma^2)$ NORMAL

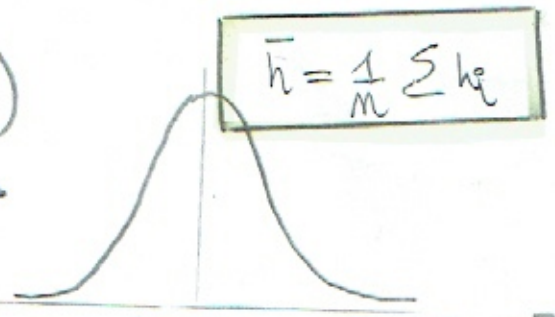
CENTRAL Limit THEOREM (CLT)

$VAR[\bar{x}] = \frac{VAR[x]}{n}$

This is how you resolve VARIANCE problem!



$\bar{h} = \frac{1}{n} \sum h_i$



if $n \to \infty$

$$VAR[\bar{h}] = \frac{VAR[h_i]}{n}$$

⚠ WLLN CLT ASSUMES i.i.d

① identical ✓
D ∧ P

② independent ✗
↳ not fulfilled!!
↓
cannot use WLLN & CLT

Nevertheless: BAGGING works well!

RANDOM Forests

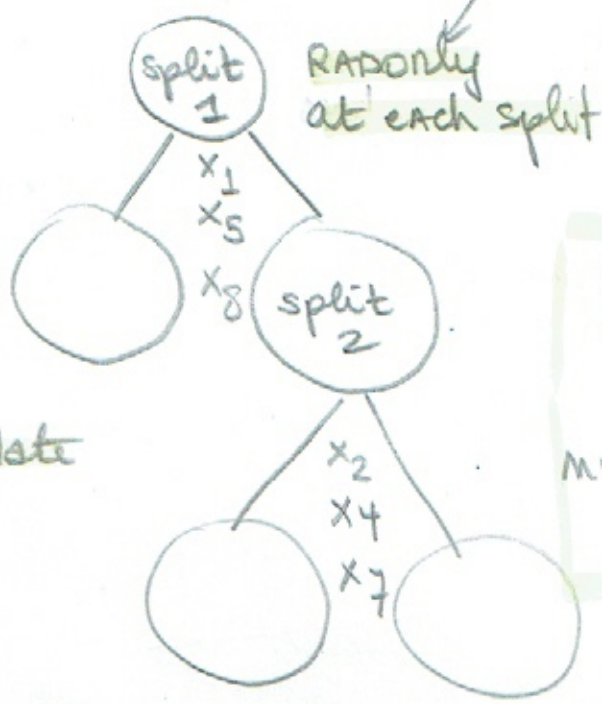
↳ BAGGING ON TREES!
↳ BREIMAN '99

① D ∧ P

→ run through all trees until all leaves are pure
→ overfitting

→ split on $k < d$ features
↓
features

example: $d = 9$
$k = 3$



RANDOMLY at each split

→ you can evaluate importance of certain features by splitting on them and calculate IMPURITY!

$k = \sqrt{d}$!

M: as large as possible
↓
no hyperp. TUNING

Boosting

Freund & Shapire '96
ADAPTIVE Boosting
||
ADABOOST
↳ classification only!

Friedman '99
GRADIENT Boosting — class / Regr.

IDEA: combine a set of 'WEAK' learners to make a super learner!

→ high BIAS

SEQUENTIAL buildup of classifier $H(x)$ over a few rounds

POPULAR at Kaggle competition

① XGBOOST 2014 Tianqi Chen
② LIGHTGBM (MICROSOFT) 2017.

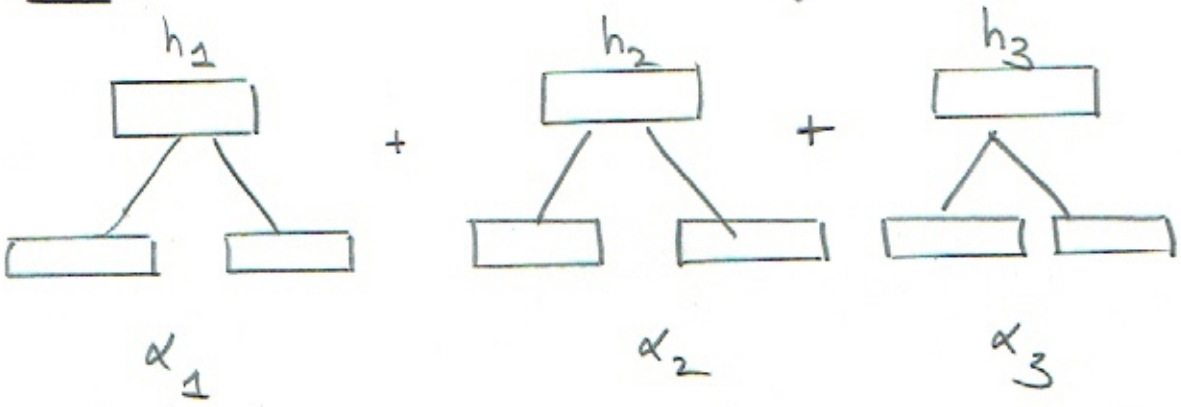
↓

FAST
Scalable
Accurate

ADABOOST → STUMPS (Tree of depth 1)

h_i → weak classifiers

IDEA:



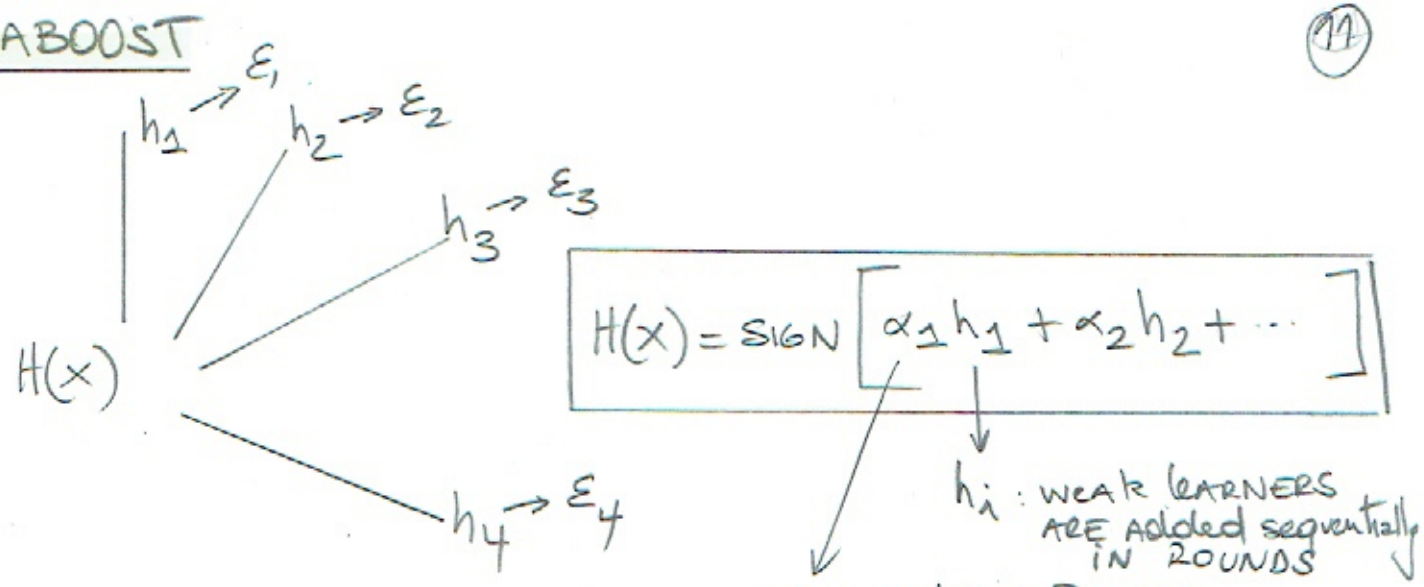
$$\alpha_i \sim \frac{1}{\varepsilon_i}$$

VOTING POWER — ERROR of h_i

$$H(x) = \text{sign}\left[\alpha_1 h_1 + \alpha_2 h_2 + \dots\right]$$

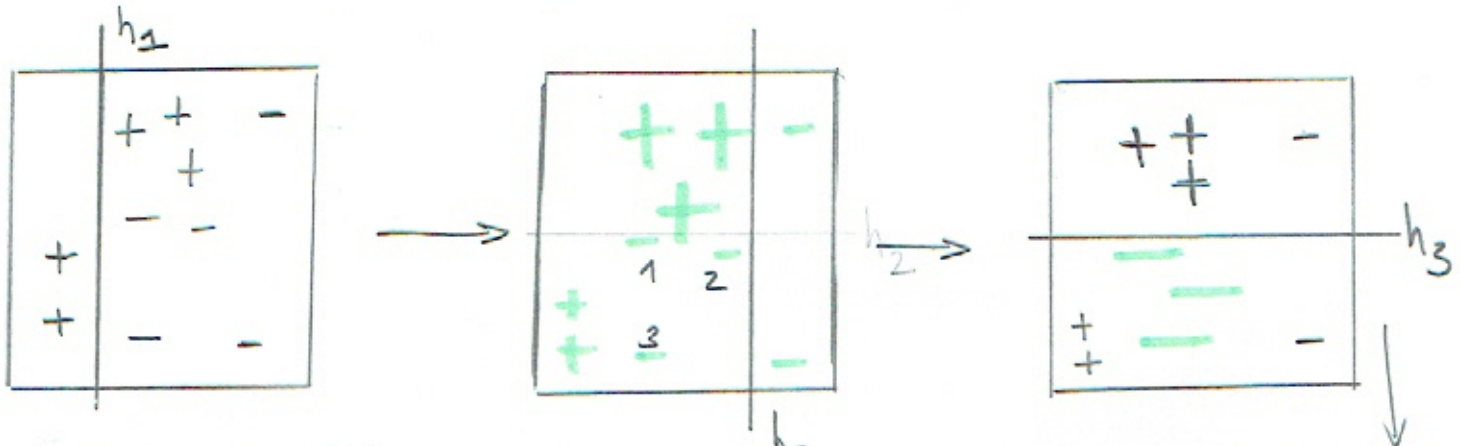
Stop when target reached or when all points correctly classi

ADABOOST



$$\alpha = \frac{1}{2} \ln \frac{1-\varepsilon}{\varepsilon}$$

WEAK LEARNERS are STUMPS (simple trees)



① INITIALISE weights
$w_i = \frac{1}{n}$ n is # points.

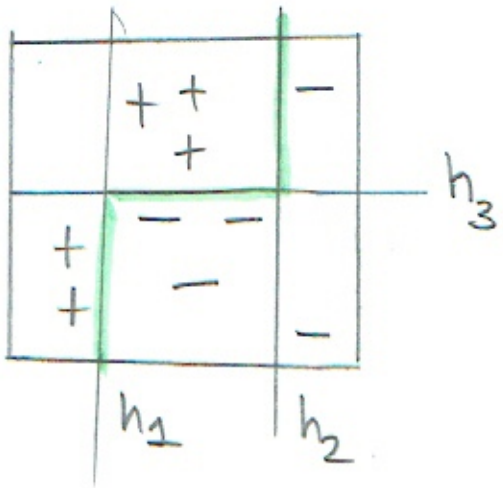
② pick best WEAK LEARNER
h_1 (lowest ERROR)

③ $h_1 \rightarrow$ 3 POINTS misclassified
↳ INCREASE weights
± → reduce weights other points.
↳ ASSIGN voting power to h_1 $\rightarrow \alpha_1 \rightarrow \alpha_1 = 0.41$
If ε is large we want α to be small
↳ Σ weights of misclassified points

④ Append $\alpha_i h_i$ to $H(x)$

h_2
↓
1,2,3 misclassified
$\alpha_2 = 0.66$

$\alpha_3 = 0.93$
↓
high weight points are CORRECT

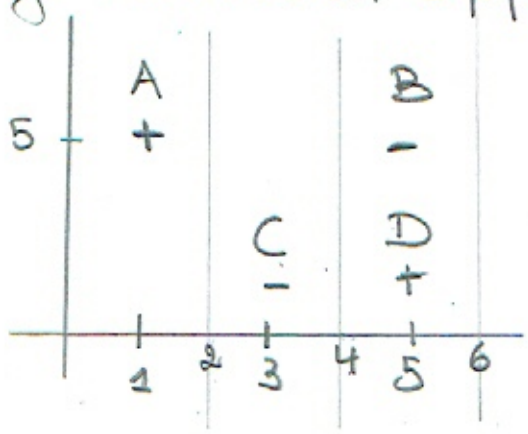


$$H(x) = \text{sign}\left[0.42 \; \square + 0.66 \; \square + 0.93 \; \square\right]$$

↓

We are done as all pts correctly classified!

Example of ADABOOST application with 4 points



4 pts = N → $\frac{1}{N} = \frac{1}{4}$

① Stumps of KD Tree

→ $\sum_{\text{wrong}} w_i$

	Misclan	ε RND1	ε RND2	ε RN3
X < 2	D	1/4	3/6	9/24
X < 4	C, D	2/4	4/6	15/24
X < 6	B, C	2/4	2/6	12/24
X > 2	A, B, C	3/4	3/6	15/24
X > 4	B, A	2/4	2/6	9/24
X > 6	A, D	2/4	4/6	12/24

	init	RND 2	RND 3
w_A	1/4	1/6	3/24
w_B	1/4	1/6	6/24
w_C	1/4	1/6	6/24
w_D	1/4	3/6	9/24

② Assign weights → $\frac{1}{N}$ at start

③ Calculate ε

④ Pick best h

$$\alpha = \frac{1}{2} \ln \frac{1-\varepsilon}{\varepsilon}$$

	RND 1	RD 2	RND 3
h	X<2	X<6	X>4
ε	1/4	1/3	3/8
α	$\frac{1}{2}\ln 3$	$\frac{1}{2}\ln 2$	$\frac{1}{2}\ln\frac{5}{3}$

D voted correctly by (1) α (2)

⑤ Append H(x) = $\text{sgn}\left[\frac{1}{2}\ln 3\,(x<2) + \frac{1}{2}\ln 2\,(x<6) + \frac{1}{2}\ln\frac{5}{3}\,(x>4)\right]$

D by BC (1) — $\frac{1}{2}\ln 3 \approx 1.09$

D by BA (2) — $\frac{1}{2}\ln 2 \approx 0.69$

$\frac{1}{2}\ln\frac{5}{3} \approx 0.51$

Are we done ? → no D is misclassified!

↓

update weights → ROUND 2

$$w_{new} = \begin{cases} w_{old} \cdot \frac{1}{2}\frac{1}{1-\varepsilon} & \text{if correctly classified} \\ w_{old} \cdot \frac{1}{2}\frac{1}{\varepsilon} & \text{if wrongly classified} \end{cases}$$

$w_A = 1/4 \cdot \frac{1}{2} \cdot \frac{4}{3} = 1/6$

$w_B = 1/4 \cdot \frac{1}{2} \cdot \frac{4}{3} = 1/6$

$w_C = 1/4 \cdot \frac{1}{2} \cdot \frac{4}{3} = 1/6$

$w_D = 1/4 \cdot \frac{1}{2}^{4} = 1/2$ ✓

Are we done ? → B, C still misclassified!

→ ROUND 3 → we need it to have the necessary weight + classify D correctly

$w_A = \frac{1}{6} \cdot \frac{1}{2} \cdot 3 = \frac{3}{12} = \frac{6}{24}$ ✓

$w_B = \frac{1}{6} \cdot \frac{1}{2} \cdot 3 = \frac{3}{12} = \frac{6}{24}$ ✓

$w_C = \frac{1}{6} \cdot \frac{1}{2} \cdot 3 = \frac{3}{12} = \frac{6}{24}$ ✓

misclassified weights $\leq$ to $1/2$ and same for correct classified weight

$w_D = \frac{3}{6} \cdot \frac{1}{2} \cdot \frac{3}{2} = \frac{9}{24}$

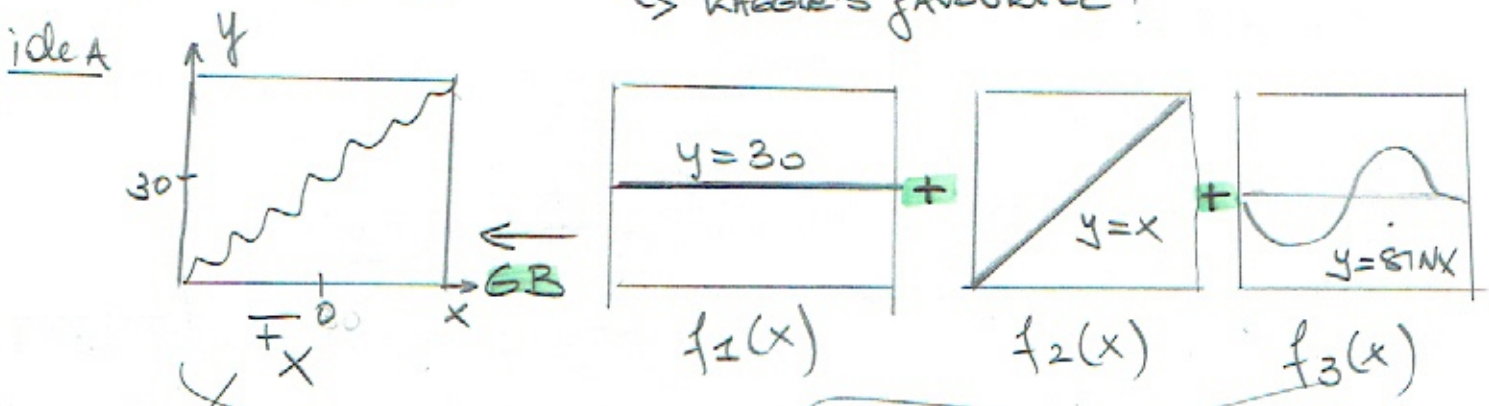
Are we done ? → yes as no more points misclassified.

Gradient Boosting → Algorithm used by search engines

Trees ← { - XGBOOST
- LightGBM

→ Kaggle's favourite !

idea



Target function !

Additive modelling.

$$F(x) = f_1(x) + f_2(x) + f_3(x)$$

functions : M ← hyperparameter

$$\rightarrow F_M(x) = \sum_{m=1}^{M} f_m(x) \quad (1)$$

in machine learning : function = model

y_p
h

OR $F_M(x) = F_{m-1}(x) + f_m(x)$

For intuition we will use a regression Tree ↔ fitting function trough (x_i, y_i)

→ classification can be found on Youtube

→ StatQuest : Gradient Boost Part 4 : classification details.

| MATH is challenging

Example: house price prediction!

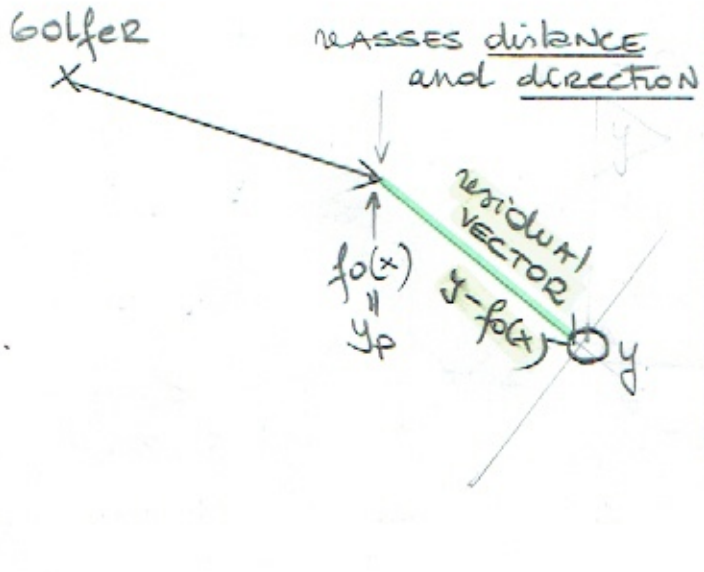
↳ simple: 1 feature: size house

i	size (m^2) $x^{(i)}$	price (R$) $y^{(i)}$	$F_0(x^{(i)})$	$y - F_0 = r_{i1}$	$F_1(x^{(i)})$	$y - F_1 = r_{i2}$
1	50	60	398	-338	+378	-318
2	100	180	398	-218	+378	-198
3	150	250	398	-148	+378	-128
4	200	300	398	-98	+378	-78
5	250	700	398	302	+438	262
6	300	900	398	502	+438	462
	350	?	↑ first prediction	↑ residual r_{i1}	↑ 2nd prediction	↑ r_{i2}

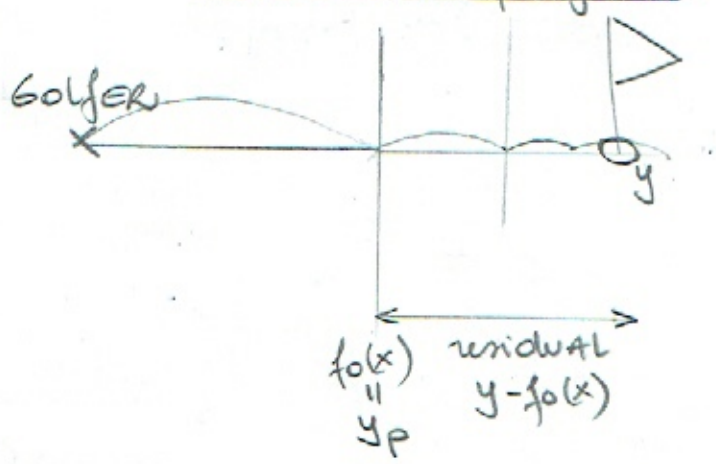
$i: 1 \to 6$ $m = 6$ samples → fitting function that best connects $(x^{(i)}, y^{(i)})$

Analogy: Golf

Top view projection



side view projection



(1) p15 → $$y_p = \sum_{m=1}^{M} f_m(x) \quad (2)$$

↳ WE ARE IN FUNCTION SPACE !

Cost function $$\frac{1}{2}(y - y_p)^2 = C, \quad (3)$$ → must be differentiable!

$$\frac{dC}{dy_p} \overset{\text{CHAIN RULE}}{=} \frac{1}{2} \cdot 2(y - y_p) \cdot (-1) \quad \rightarrow C(y, y_p)$$

$$= -(y - y_p)$$

residual !

now: back to housing price exercise

① initialise model → $y_{p_0} = F_0(x)$

(2) p17 → $$F_0(x) = \underset{y_p}{\text{ARGMIN}} \sum_{i=1}^{6} \cdot C(y^{(i)}, y_p^{(i)})$$

INITIAL prediction ↑ target ↑ WE need to find y_p that MINIMIZES this SUM !

(3) → $$\frac{1}{2}(60 - y_p)^2 + \dots + \frac{1}{2}(900 - y_p)^2$$

$$\frac{\partial C}{\partial y_p} = -(60 - y_p) - (180 - y_p) - \dots - (900 - y_p)$$

$$= 6y_p - 2390 \overset{\text{set}}{=} 0 \rightarrow y_p = \frac{2390}{6} = 398$$

↑ Average of $y_s^{(i)}$

$$\Rightarrow \boxed{F_0(x) = 398}$$

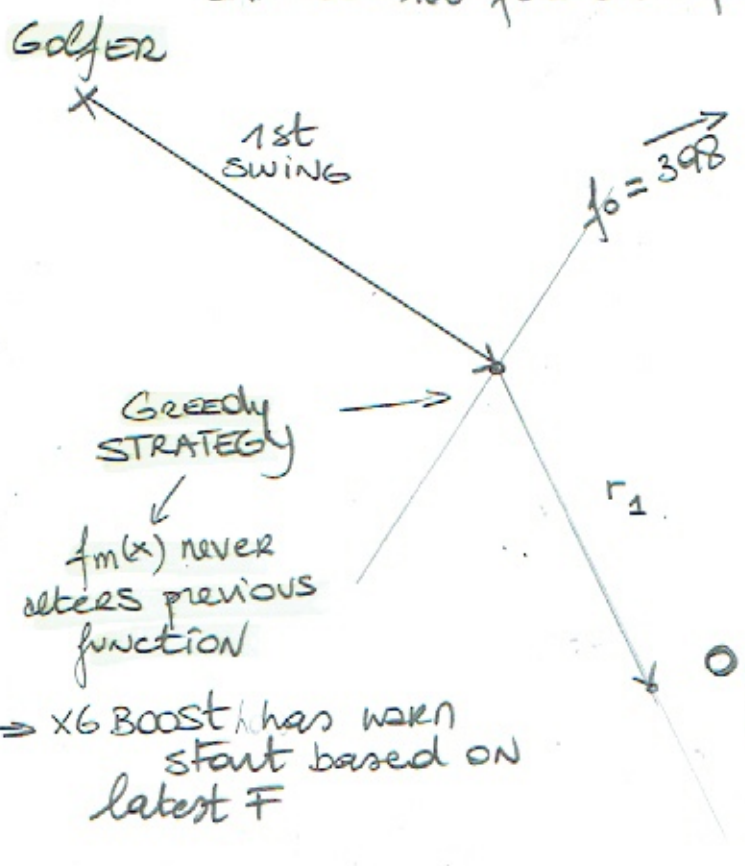
↳ is just a leaf → ? is predicted to be 398k $!

(2) → build M trees to predict the residuals !

↓ instead of price of the House

CAN be 100 for example

m : 1 → 100.



→ XG Boost has warn start based on latest F

USE Size of house to predict r_{i1} etc...

↓

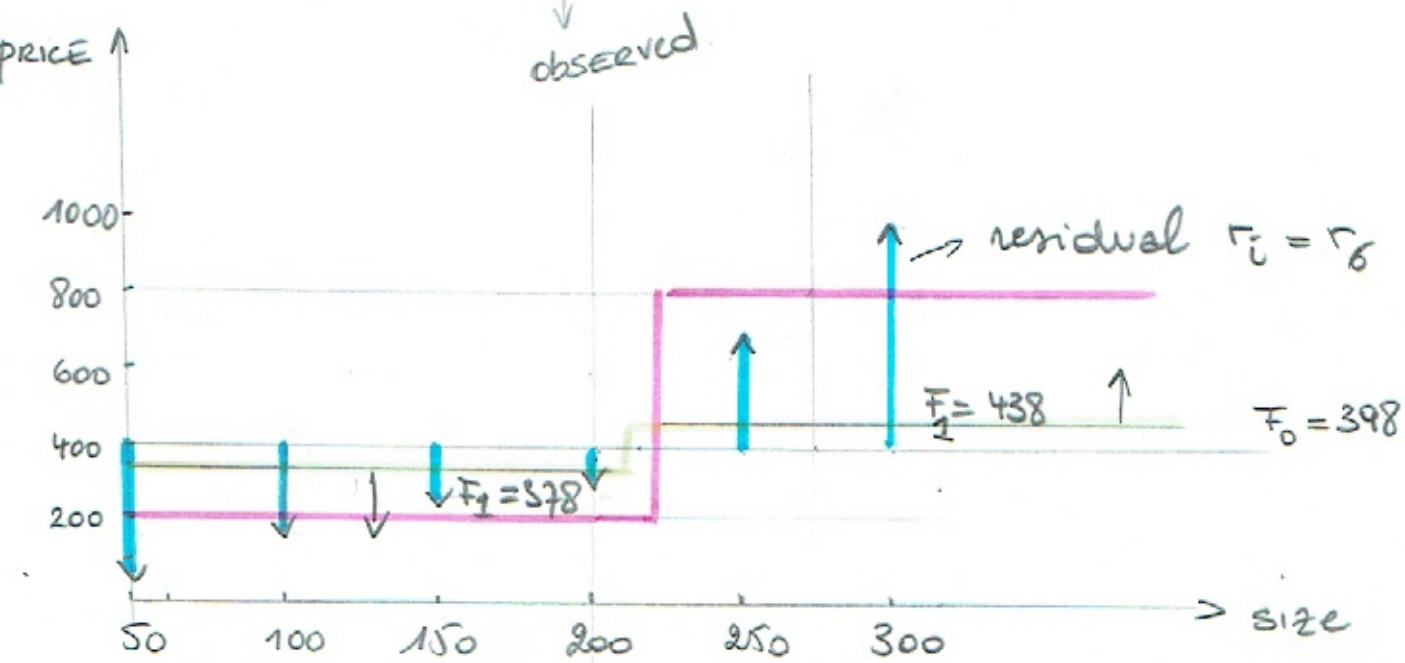
|| WE TRAIN ON THE RESIDUALS

m = 1 → COMPUTE residual r

$$r_{i1} = (y^{(i)} - y^{(i)}_{F_0})$$

1st SAMPLE ← 1st tree → $F_0(x)$ ↳ Predicted

→ fill IN IN column r_{i1}



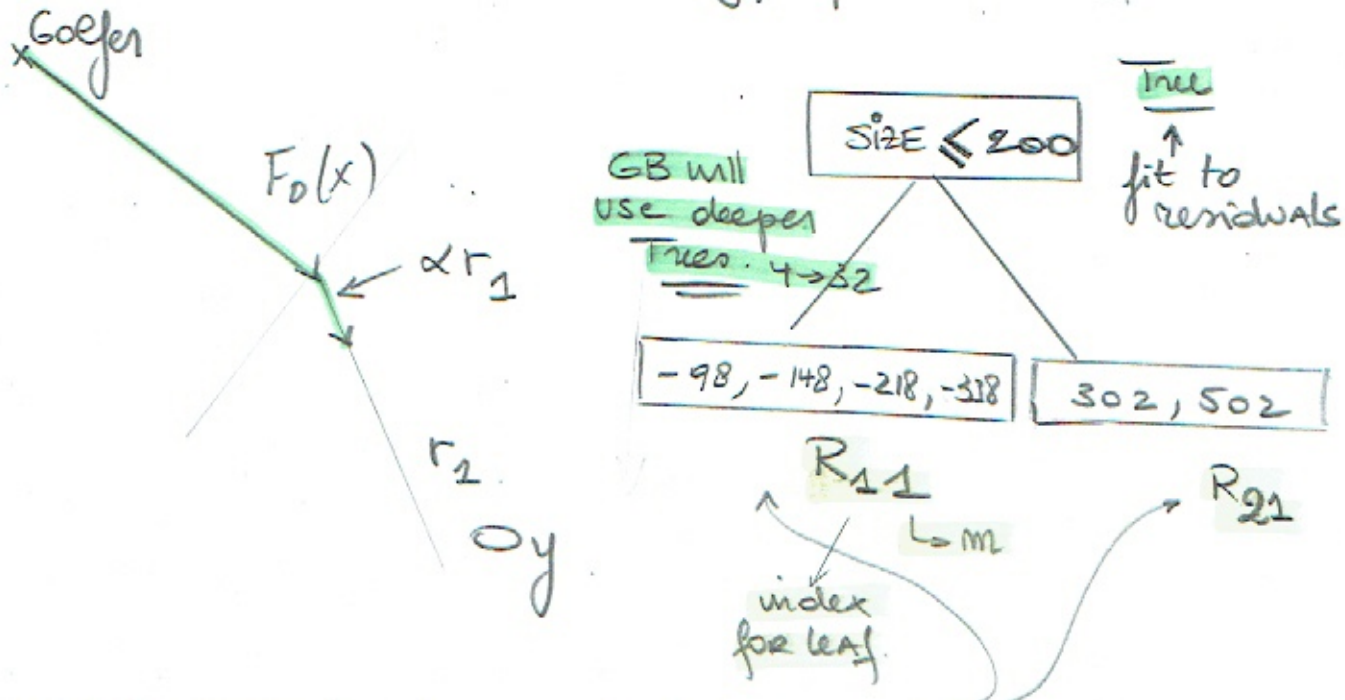
A perfect r_1 would be $F_1(x) = F_0(x) + [y - F_0(x)]$ (19)

$= y$ ↯ model is NOT PERFECT

⇓ r_1 vector does not end in y!

let's define → $\boxed{F_m(x) = F_{m-1}(x) + \alpha r_1}$

↓ learning Rate hyperparameter!



now need to create terminal Regions : R_{jm}

& calculate outputs

R_{11} → $y_{pj1} = \underset{y_p}{\text{ARGMIN}} \sum_{\substack{x^{(i)} \\ \in \\ R_{ij}}} C\left(y^{(i)}, \overbrace{F_0(x^{(i)})}^{y_{p_1}} + y_p\right)$ (1)

↳ you take previous PREDICTION INTO ACCOUNT

(1) → $y_{pj1} = \underset{y_p}{\text{ARGMIN}} \sum \frac{1}{2}\left(y^{(i)} - \left(F_{m-1}(x^{(i)}) + y_p\right)\right)^2$

→ $\frac{\partial C}{\partial y_p} = \frac{1}{2} \cdot 2\left[60 - (398 + y_p)\right] + \frac{1}{2} 2[\ ] \ldots\ldots$

Team 1

TERM 1 : $(-338 - y_p)(-1)$
2 : $(-218 - y_p)(-1)$
3 : $(-148 - y_p)(-1)$
4 : $(-98 - y_p)(-1)$

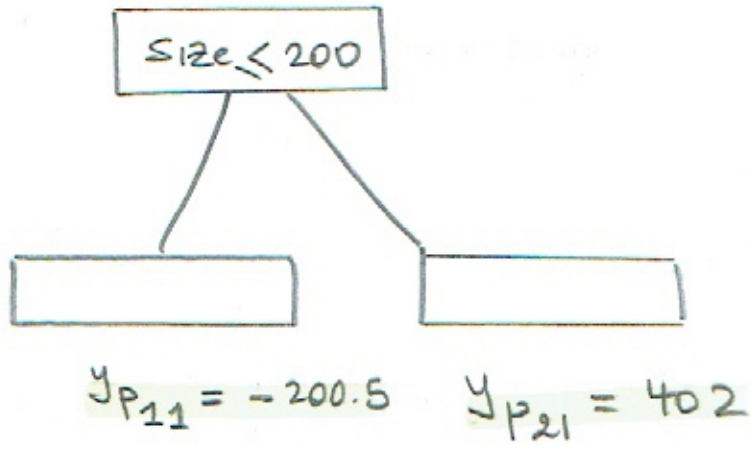
$\rightarrow \Sigma \Rightarrow +4y_p + 802 \overset{\text{set}}{=} 0$

$\downarrow$

$y_{p11} = -200.5$

Average of residuals that ended in leaf 1

$R_{21} \rightarrow y_{p21} = +402$



$y_{p_{11}} = -200.5$ $y_{p_{21}} = 402$

Now: Update F $\rightarrow$ $F_1(x) = F_0(x) + \alpha \text{Tree}$

↳ NEW Prediction! for EACH Sample

SET = 0.1 must $\in [0,1]$

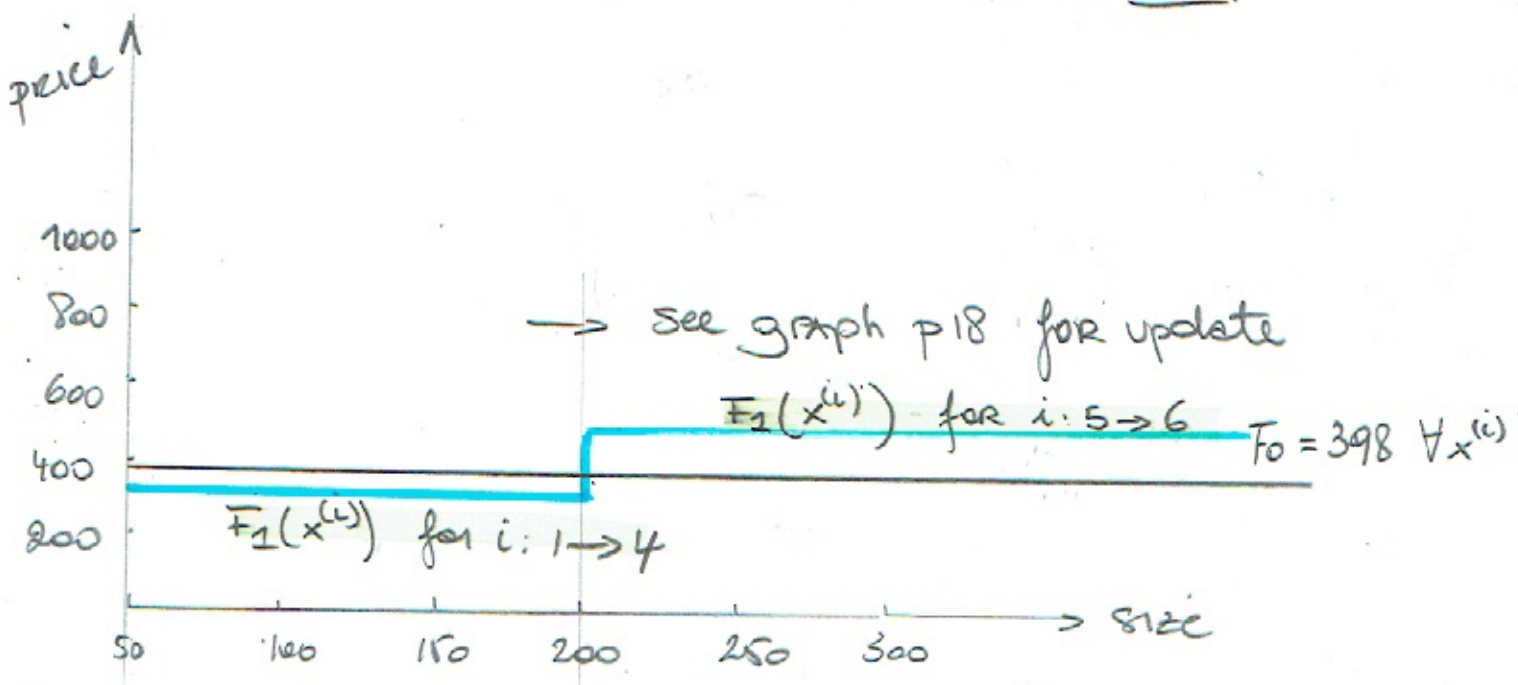
$$\Rightarrow F_1(x) = F_0(x) + \alpha \sum_{j=1}^{m} y_{pj}$$

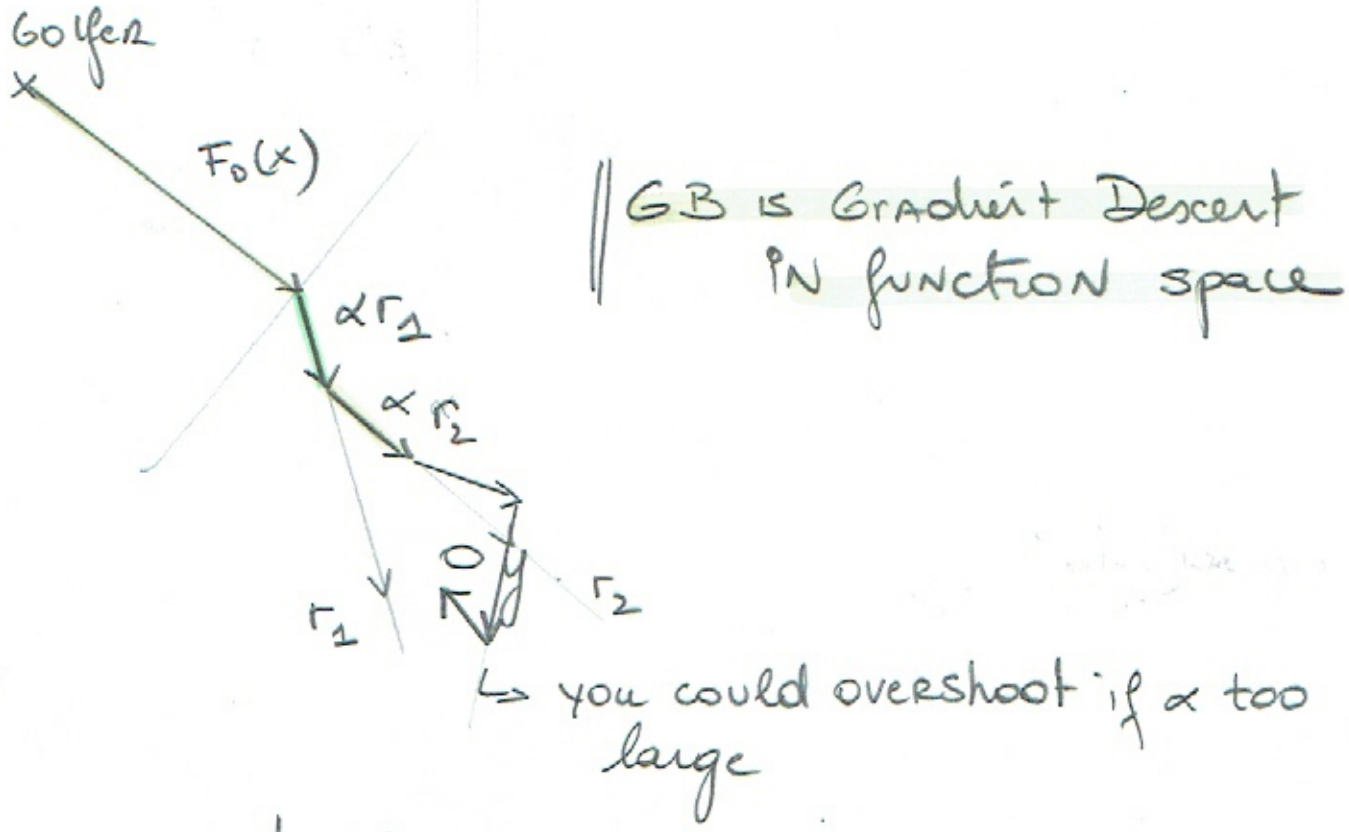
m → #leaves

$F_1(x^{(1)}) = +398 + 0.1\ (-200.5) = \boxed{378}$ ✓

→ same for $x^{(2)}, x^{(3)}, x^{(4)}$

$F_1(x^{(5)}) = F_1(x^{(6)}) = +398 + 40 = \boxed{438}$ ✓





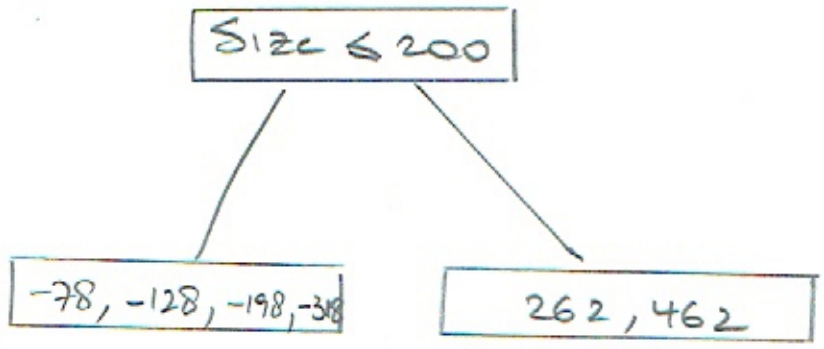
GB is Gradient Descent in function space

↳ you could overshoot if α too large

If α too small → slow

m=2

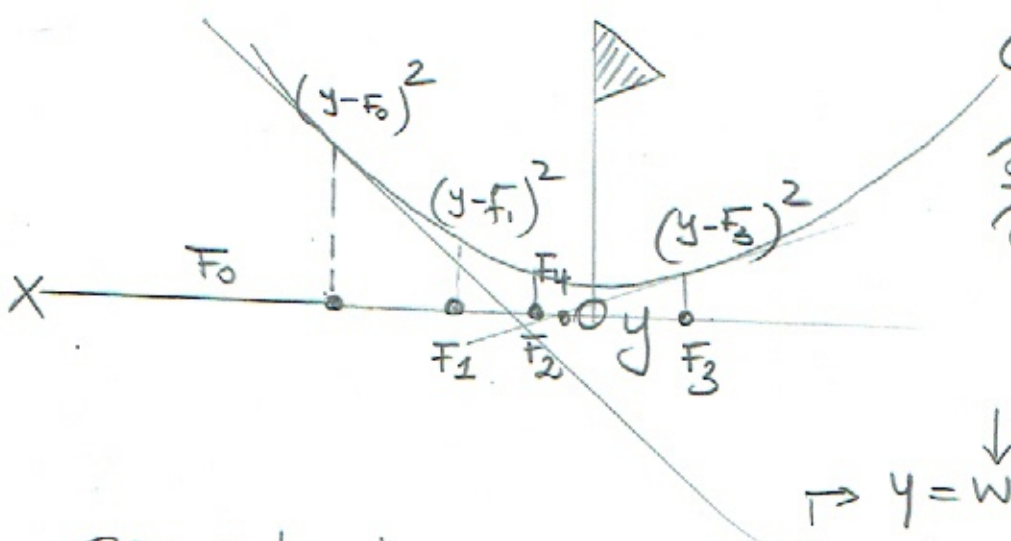
Tree



→ Repeat m tines 1 → M → gradually move to y

hyperparameters — M → # trees (4 → 32)
— α : learning rate

Gradient Descent? INTUITION GOLFER



↦ cost function

$C(y^{(i)} - y_p^{(i)})$

$\frac{\partial C}{\partial y_p}$ gives → direction INTO which y_p has to move

↦ $y = w^T x$ (x → $\bar{x}$)

GDescent: tweaking parameters/weights of features in order to minimize $(y - y_p)$

GBoosting: Adding models together sequentially with the goal to optimize the composite model

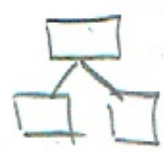
↓ boost model OUTPUT

prediction (y_p) so as to bring it as close as possible to y.

3) **Prediction** ASSUME M=2

$F_2(x)$ = FINAL OUTPUT → OUTPUT Tree 2 : 362
→ OUTPUT Tree 1
→ $x > 200$ 402

→ new data $x^{(7)} = 350\ m^2$

$F_2(x) = 398 + 0.1$  $+ 0.1$ 

Tree 1 Tree 2

$= 398 + 40 + 36 = 474$

Classification

↳ same method but different Cost FUNCTION

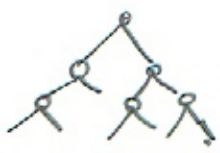
↙ $-\log(\text{likelihood})$ (LR)

pseudo residuals ←

XG Boost

eXtreme GRADient Boosting

BIAS → ADABOOST
GRADIENT Boosting



2014 Tianqi chen VAR

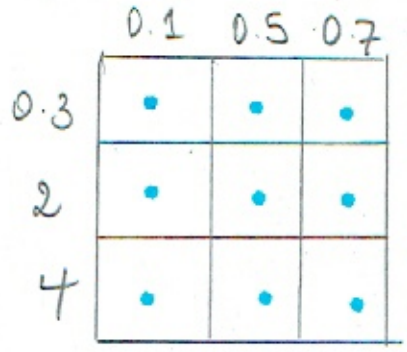
XG Boost library

→ BAGGING
Random forests

PARAMETER TUNING → Hyperparameters
→ you as data scientist will need to choose

① GridSearchCV

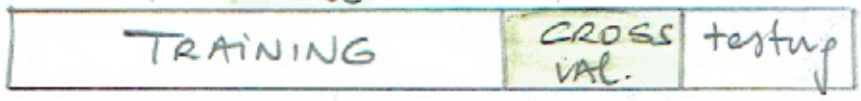
α	0.1	0.5	0.7
β	0.3	2	4



works when:
① # parameters is small
② # samples is relatively small

↳ 2 parameters each with 3 values!
→ 9 scenarios to train α
cross-validate

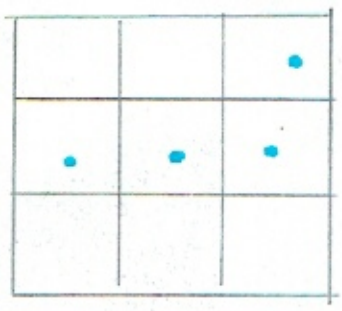
k-fold cross-validation → average the performance at end



↑ used to tune parameters!

→ Brute force → takes long time
↳ does not learn from previous rounds

② RandomSearchCV



→ A bit more efficient but still does not learn from previous rounds.
→ randomly sample from grid!

Conclusion: we do trial & error Engineering!
↓ defaults will not lead you best possible results
Black Box

BAYESIAN OPTIMIZATION

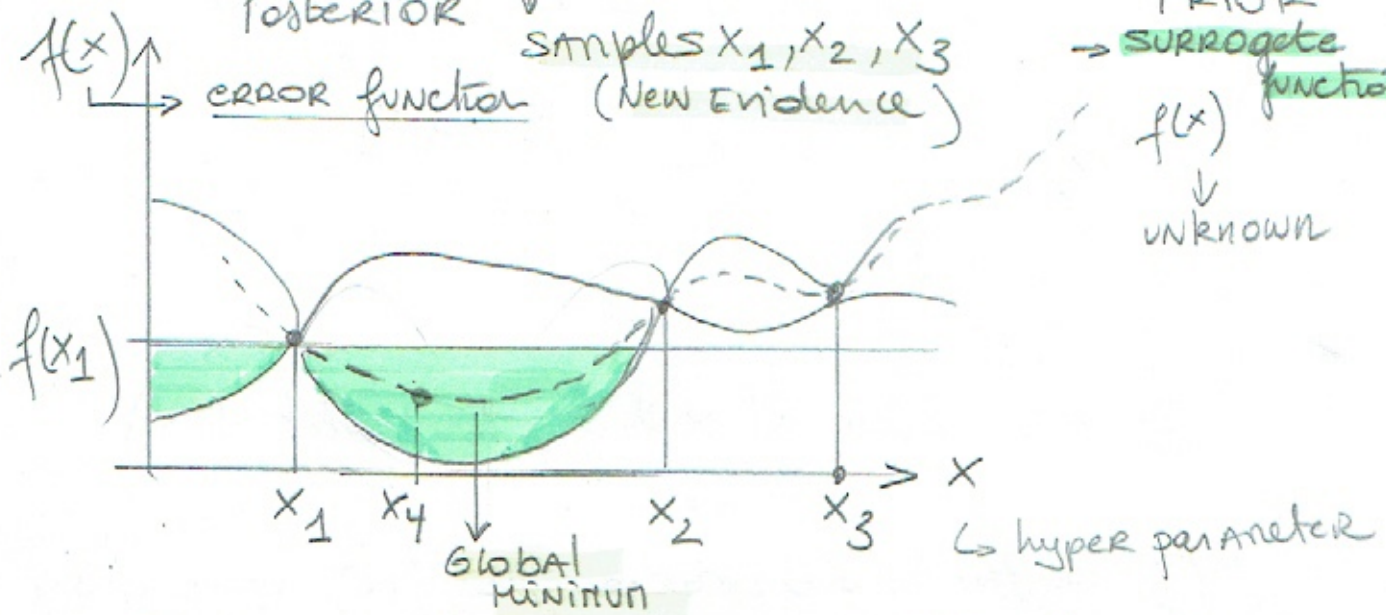
- consider hyper parameters as random variables with a distribution that we don't know

→ optimization problem for finding the MINIMUM of a function $f(x)$ without calculating gradients

3 options
① GAUSSIAN Processes → $P(f(x)|X) \sim N(\mu, \sigma)$
② RANDOM FORESTS
③ TREE PARZEN ESTIMATORS

Bayes: $P(f(x)|X) \sim P(X|f(x))\ P(f(x))$

Posterior — samples X_1, X_2, X_3 (NEW EVIDENCE) — PRIOR → SURROGATE function



↳ hyper parameter

How to find correct zone to sample X_4 from ?

- look for zones where the expected IMPROVEMENT is MAXIMUM and where $f(x) \leq f(x_1)$

$$EI = E\left[\max\left(f(x) - f(x_1)\right), 0\right]$$ → ACQUISITION FUNCTION

→ take X_4 in green zone

parameters XGBoost

① learning rate α $F_1 = F_0 + \alpha r_1$
0.1 → 1

② n_estimators → $F_M = F_{M-1} + \alpha r_m$
↳ M ↳ # trees M
(25 → 1000)

③ gamma → regularization factor
(0 → 5)

④ min_child_weight → min # of items in leaf before you continue to split
(2 → 10)

⑤ max_depth → against overfitting
(1 → 32)

⑥ subsample
⑦ colsample } analogues with Bagging
(0.5 → 1)

$D \leftarrow D_1, D_2, D_3$

→ split $k < d$.