Decision Trees: Some exercises

Exemplifying

how to compute information gains and how to work with *decision stumps*

CMU, 2013 fall, W. Cohen E. Xing, Sample questions, pr. 4

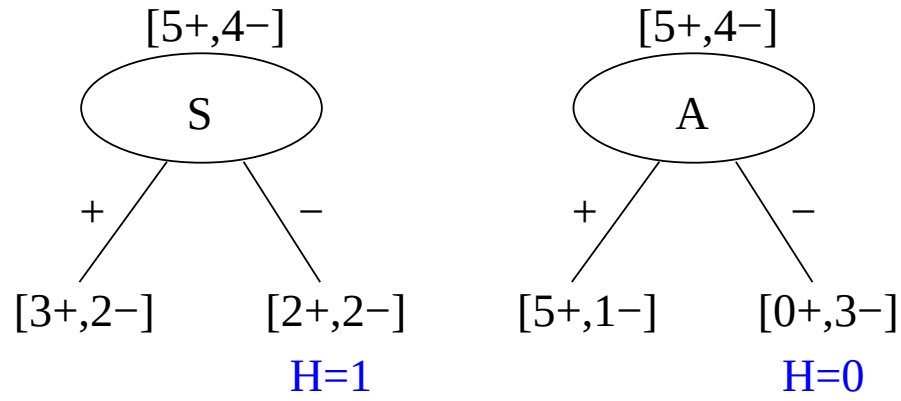
Timmy wants to know how to do well for ML exam. He collects those old statistics and decides to use decision trees to get his model. He now gets 9 data points, and two features: "whether stay up late before exam" (S) and "whether attending all the classes" (A). We already know the statistics is as below:

$$Set(all) = [5+, 4-]$$
$$Set(S+) = [3+, 2-], Set(S-) = [2+, 2-]$$
$$Set(A+) = [5+, 1-], Set(A-) = [0+, 3-]$$

Suppose we are going to use the feature to gain most information at first split, which feature should we choose? How much is the information gain?

You may use the following approximations:

N	3	5	7
$\log_2 N$	1.58	2.32	2.81



$$H(all) \overset{not.}{=} H[5+,4-] \overset{not.}{=} H\left(\frac{5}{9}\right) \overset{sim.}{=} H\left(\frac{4}{9}\right) \overset{def.}{=} \frac{5}{9}\log_2\frac{9}{5} + \frac{4}{9}\log_2\frac{9}{4} = \log_2 9 - \frac{5}{9}\log_2 5 - \frac{4}{9}\log_2 4$$

$$= 2\log_2 3 - \frac{5}{9}\log_2 5 - \frac{8}{9} = -\frac{8}{9} + 2\log_2 3 - \frac{5}{9}\log_2 5 = 0.991076$$

$$H(all|S) \overset{def.}{=} \frac{5}{9}\cdot H[3+,2-] + \frac{4}{9}\cdot H[2+,2-] = \ldots$$

$$= \frac{5}{9}\cdot 0.970951 + \frac{4}{9}\cdot 1 = 0.983861$$

$$H(all|A) \overset{def.}{=} \frac{6}{9}\cdot H[5+,1-] + \frac{3}{9}\cdot H[0+,3-] = \ldots$$

$$= \frac{6}{9}\cdot 0.650022 + \frac{4}{9}\cdot 0 = 0.433348$$

$$IG(all,S) \overset{def.}{=} H(all) - H(all|S) = 0.007215$$

$$IG(all,A) \overset{def.}{=} H(all) - H(all|A) = 0.557728$$

$$IG(all,S) < IG(all,A) \Leftrightarrow H(all|S) > H(all|A)$$

Exemplifying the application of the ID3 algorithm on a toy *mushrooms* dataset

CMU, 2002(?) spring, Andrew Moore, midterm example questions, pr. 2

You are stranded on a deserted island. Mushrooms of various types grow widely all over the island, but no other food is anywhere to be found. Some of the mushrooms have been determined as poisonous and others as not (determined by your former companions' trial and error). You are the only one remaining on the island. You have the following data to consider:

Example	*NotHeavy*	*Smelly*	*Spotted*	*Smooth*	*Edible*
A	1	0	0	0	1
B	1	0	1	0	1
C	0	1	0	1	1
D	0	0	0	1	0
E	1	1	1	0	0
F	1	0	1	1	0
G	1	0	0	1	0
H	0	1	0	0	0
U	0	1	1	1	?
V	1	1	0	1	?
W	1	1	0	0	?

You know whether or not mushrooms A through H are poisonous, but you do not know about U through W.

For the a–d questions, consider only mushrooms A through H.

a. What is the entropy of *Edible*?

b. Which attribute should you choose as the root of a decision tree? *Hint*: You can figure this out by looking at the data without explicitly computing the information gain of all four attributes.

c. What is the information gain of the attribute you chose in the previous question?

d. Build a ID3 decision tree to classify mushrooms as poisonous or not.

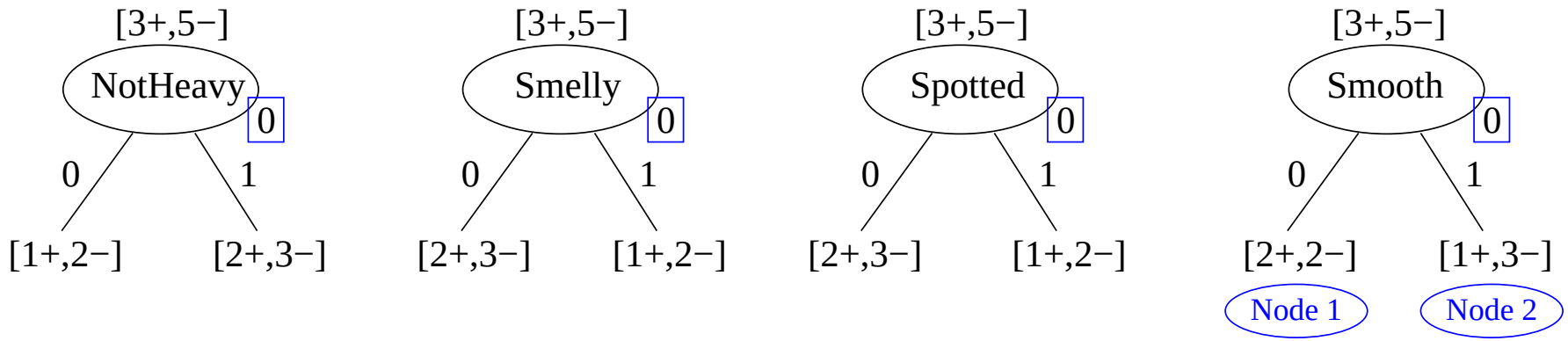
e. Classify mushrooms U, V and W using the decision tree as poisonous or not poisonous.

f. If the mushrooms A through H that you know are not poisonous suddenly became scarce, should you consider trying U, V and W? Which one(s) and why? Or if none of them, then why not?

a.

$$
\begin{aligned}
H_{Edible} &= H[3+,5-] \stackrel{def.}{=} -\frac{3}{8}\cdot\log_2\frac{3}{8} - \frac{5}{8}\cdot\log_2\frac{5}{8} = \frac{3}{8}\cdot\log_2\frac{8}{3} + \frac{5}{8}\cdot\log_2\frac{8}{5} \\
&= \frac{3}{8}\cdot 3 - \frac{3}{8}\cdot\log_2 3 + \frac{5}{8}\cdot 3 - \frac{5}{8}\cdot\log_2 5 = 3 - \frac{3}{8}\cdot\log_2 3 - \frac{5}{8}\cdot\log_2 5 \\
&\approx 0.9544
\end{aligned}
$$

b.


[3+,5−]
NotHeavy
0
0
1
[1+,2−]
[2+,3−]
[3+,5−]
Smelly
0
0
1
[2+,3−]
[1+,2−]
[3+,5−]
Spotted
0
0
1
[2+,3−]
[1+,2−]
[3+,5−]
Smooth
0
0
1
[2+,2−]
[1+,3−]
Node 1
Node 2

c.

$$
\begin{aligned}
H_{0/Smooth} &\overset{def.}{=} \frac{4}{8}H[2+,2-] + \frac{4}{8}H[1+,3-] = \frac{1}{2}\cdot 1 + \frac{1}{2}\left(\frac{1}{4}\log_2\frac{4}{1} + \frac{3}{4}\log_2\frac{4}{3}\right) \\
&= \frac{1}{2} + \frac{1}{2}\left(\frac{1}{4}\cdot 2 + \frac{3}{4}\cdot 2 - \frac{3}{4}\log_2 3\right) = \frac{1}{2} + \frac{1}{2}\left(2 - \frac{3}{4}\log_2 3\right) \\
&= \frac{1}{2} + 1 - \frac{3}{8}\log_2 3 = \frac{3}{2} - \frac{3}{8}\log_2 3 \approx 0.9056
\end{aligned}
$$

$$
\begin{aligned}
IG_{0/Smooth} &\overset{def.}{=} H_{Edible} - H_{0/Smooth} \\
&= 0.9544 - 0.9056 = 0.0488
\end{aligned}
$$

d.

$$
\begin{aligned}
H_{0/NotHeavy} &\overset{def.}{=} \frac{3}{8}H[1+,2-] + \frac{5}{8}H[2+,3-] \\
&= \frac{3}{8}\left(\frac{1}{3}\log_2\frac{3}{1} + \frac{2}{3}\log_2\frac{3}{2}\right) + \frac{5}{8}\left(\frac{2}{5}\log_2\frac{5}{2} + \frac{3}{5}\log_2\frac{5}{3}\right) \\
&= \frac{3}{8}\left(\frac{1}{3}\log_2 3 + \frac{2}{3}\log_2 3 - \frac{2}{3}\cdot 1\right) + \frac{5}{8}\left(\frac{2}{5}\log_2 5 - \frac{2}{5}\cdot 1 + \frac{3}{5}\log_2 5 - \frac{3}{5}\log_2 3\right) \\
&= \frac{3}{8}\left(\log_2 3 - \frac{2}{3}\right) + \frac{5}{8}\left(\log_2 5 - \frac{3}{5}\log_2 3 - \frac{2}{5}\right) \\
&= \frac{3}{8}\log_2 3 - \frac{2}{8} + \frac{5}{8}\log_2 5 - \frac{3}{8}\log_2 3 - \frac{2}{8} \\
&= \frac{5}{8}\log_2 5 - \frac{4}{8} \approx 0.9512
\end{aligned}
$$

$$
\Rightarrow IG_{0/NotHeavy} \overset{def.}{=} H_{Edible} - H_{0/NotHeavy} = 0.9544 - 0.9512 = 0.0032,
$$

$$
IG_{0/NotHeavy} = IG_{0/Smelly} = IG_{0/Spotted} = 0.0032 < IG_{0/Smooth} = 0.0488
$$

Important Remark (in Romanian)

În loc să fi calculat efectiv aceste câştiguri de informaţie, pentru a determina atributul cel mai ,,bun", ar fi fost suficient să comparăm valorile entropiilor condiţionale medii $H_{0/Smooth}$ şi $H_{0/NotHeavy}$:

$$IG_{0/Smooth} > IG_{0/NotHeavy} \Leftrightarrow H_{0/Smooth} < H_{0/NotHeavy}$$

$$\Leftrightarrow \quad \frac{3}{2} - \frac{3}{8}\log_2 3 < \frac{5}{8}\log_2 5 - \frac{1}{2} \Leftrightarrow 12 - 3\log_2 3 < 5\log_2 5 - 4$$

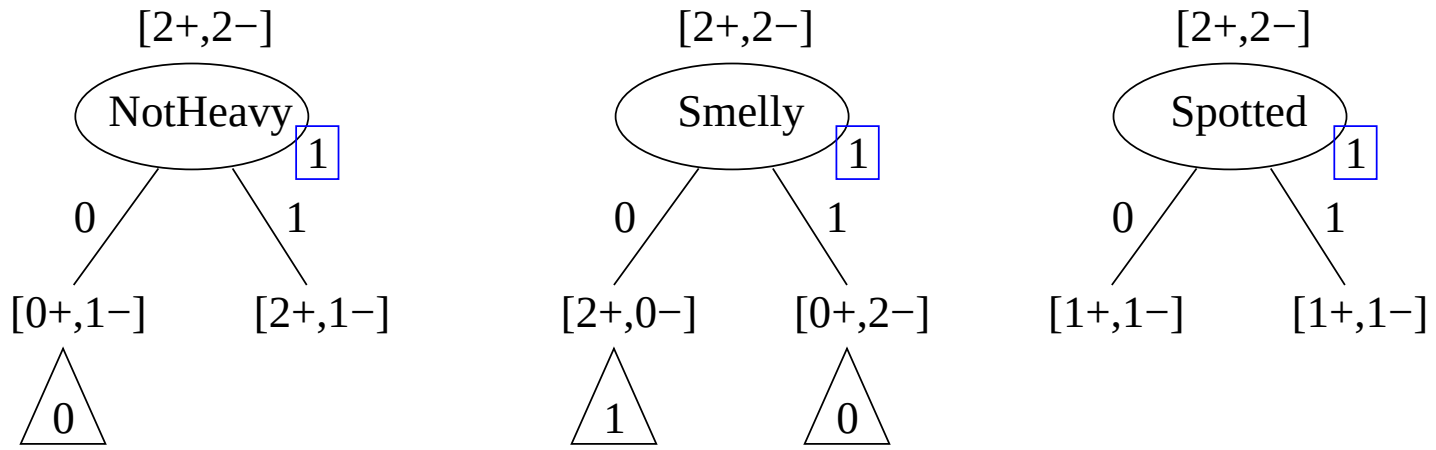
$$\Leftrightarrow \quad 16 < 5\log_2 5 + 3\log_2 3 \Leftrightarrow 16 < 11.6096 + 4.7548 \ \textbf{(adev.)}$$

În mod alternativ, ţinând cont de formulele de la problema UAIC, 2017 fall, S. Ciobanu, L. Ciortuz, putem proceda chiar *mai simplu* relativ la calcule (nu doar aici, *ori de câte ori nu avem de-a face cu un număr mare de instanţe*):

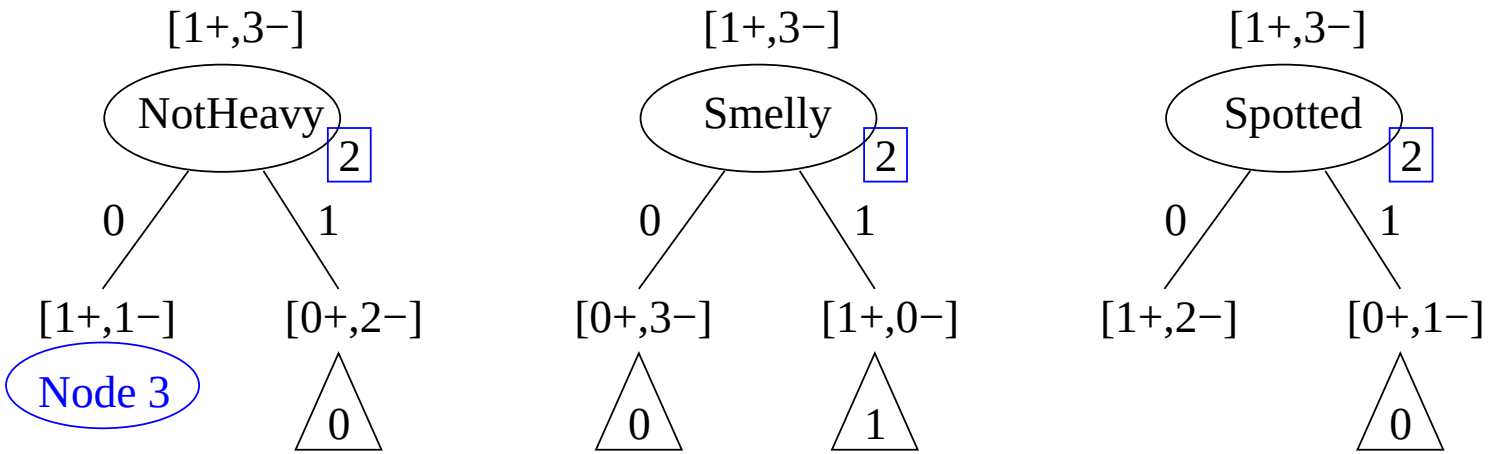
$$H_{0/Netedă} < H_{0/Uşoară} \Leftrightarrow \frac{4^4}{\cancel{2^2}\cdot\cancel{2^2}} \cdot \frac{4^4}{3^3} < \frac{5^5}{\cancel{2^2}\cdot\cancel{3^3}} \cdot \frac{\cancel{3^3}}{\cancel{2^2}} \Leftrightarrow \frac{4^8}{3^3} < 5^5 \Leftrightarrow 4^8 < 3^3 \cdot 5^5 \Leftrightarrow 2^{16} < 3^3 \cdot 5^5$$

$$\Leftrightarrow 64 \cdot \underbrace{2^{10}}_{1024} < 27 \cdot \underbrace{25 \cdot 125}_{> 3 \cdot \underbrace{8 \cdot 125}_{1000}} \quad \textbf{(adev.)}$$

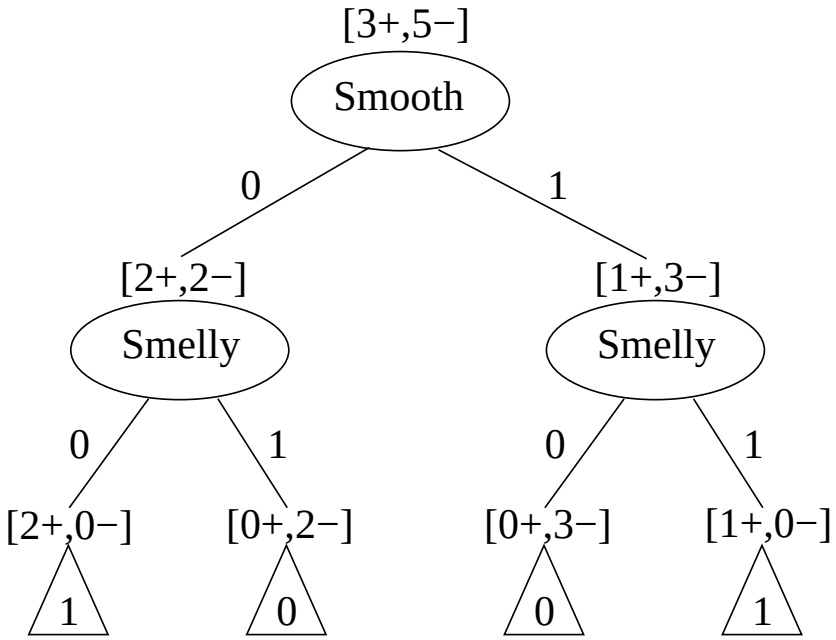
Node 1: Smooth = 0



Node 2: Smooth = 1



The resulting ID3 Tree



IF $(Smooth = 0$ **AND** $Smelly = 0)$ **OR**
$(Smooth = 1$ **AND** $Smelly = 1)$
THEN $Edibile$;
ELSE $\neg Edible$;

Classification of *test instances*:

U	$Smooth = 1,\ Smelly = 1 \Rightarrow Edible = 1$
V	$Smooth = 1,\ Smelly = 1 \Rightarrow Edible = 1$
W	$Smooth = 0,\ Smelly = 1 \Rightarrow Edible = 0$

Exemplifying the *greedy* character of the ID3 algorithm

CMU, 2003 fall, T. Mitchell, A. Moore, midterm, pr. 9.a

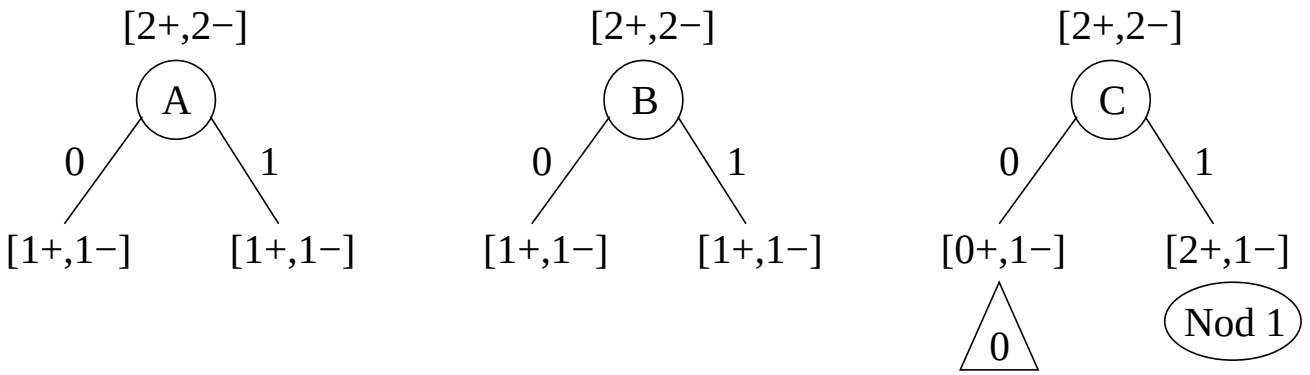
Fie atributele binare de intrare A, B, C, atributul de ieşire Y şi următoarele exemple de antrenament:

A	B	C	Y
1	1	0	0
1	0	1	1
0	1	1	1
0	0	1	0

a. Determinaţi arborele de decizie calculat de algoritmul ID3. Este acest arbore de decizie *consistent* cu datele de antrenament?

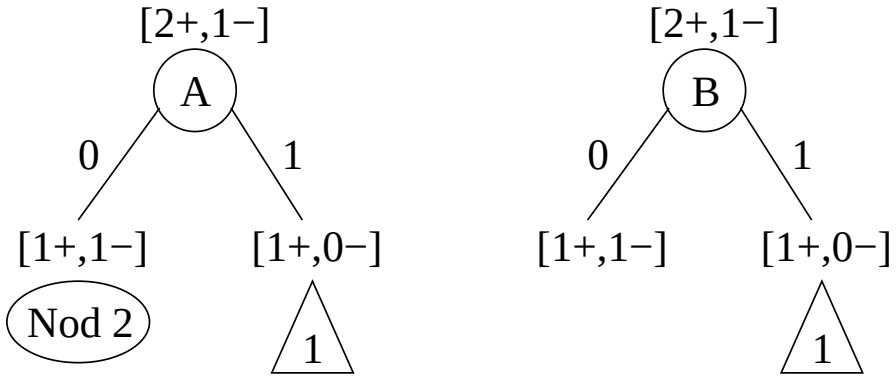
Răspuns

***Nodul 0*: (rădăcina)**



Se observă imediat că primii doi "compaşi de decizie" (engl. decision stumps) au $IG = 0$, în timp ce al treilea compas de decizie are $IG > 0$. Prin urmare, în nodul 0 (rădăcină) vom pune atributul C.

***Nodul 1*: Avem de clasificat instanţele cu $C = 1$, deci alegerea se face între atributele A şi B.**



Cele două entropii condiţionale medii sunt egale:

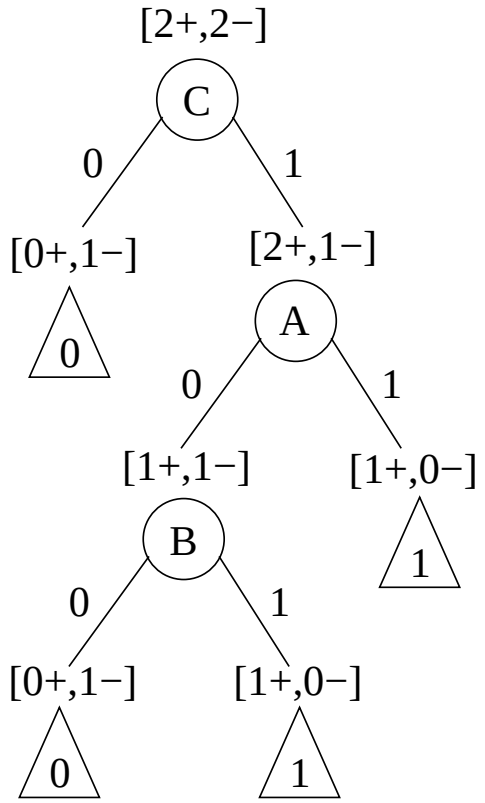
$$H_{1/A} = H_{1/B} = \frac{2}{3}H[1+, 1-] + \frac{1}{3}H[1+, 0-]$$

Aşadar, putem alege oricare dintre cele două atribute. Pentru fixare, îl alegem pe A.

***Nodul 2*: La acest nod nu mai avem disponibil decât atributul B, deci îl vom pune pe acesta.**

Arborele ID3 complet este reprezentat în figura alăturată.

Prin construcţie, algoritmul ID3 este *consistent* cu datele de antrenament dacă acestea sunt consistente (i.e., necontradictorii). În cazul nostru, se verifică imediat că datele de antrenament sunt consistente.



b. Există un arbore de decizie de adâncime mai mică (decât cea a arborelui ID3) consistent cu datele de mai sus? Dacă da, ce concept (logic) reprezintă acest arbore?

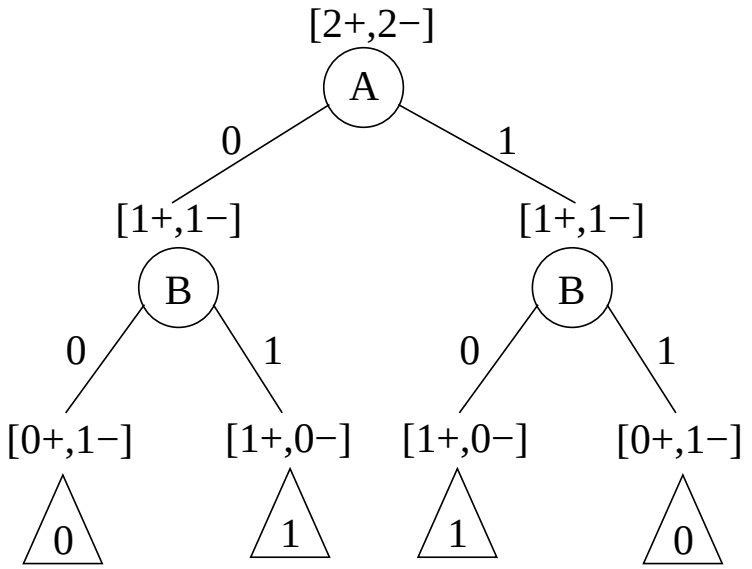
Răspuns:

Din date se observă că atributul de ieşire Y reprezintă de fapt funcţia logică A XOR B.

Reprezentând această funcţie ca arbore de decizie, vom obţine arborele alăturat.

Acest arbore are cu un nivel mai puţin decât arborele construit cu algoritmul ID3.

Prin urmare, *arborele ID3 nu este optim* din punctul de vedere al numărului de niveluri.



Aceasta este o *consecinţă* a caracterului "greedy" al algoritmului ID3, datorat faptului că la fiecare iteraţie alegem ,,cel mai bun" atribut în raport cu criteriul câştigului de informaţie.

Se ştie că algoritmii de tip "greedy" nu grantează obţinerea optimului global.

Exemplifying the application of the ID3 algorithm

in the presence of both categorical *and* continue attributes

CMU, 2012 fall, Eric Xing, Aarti Singh, HW1, pr. 1.1

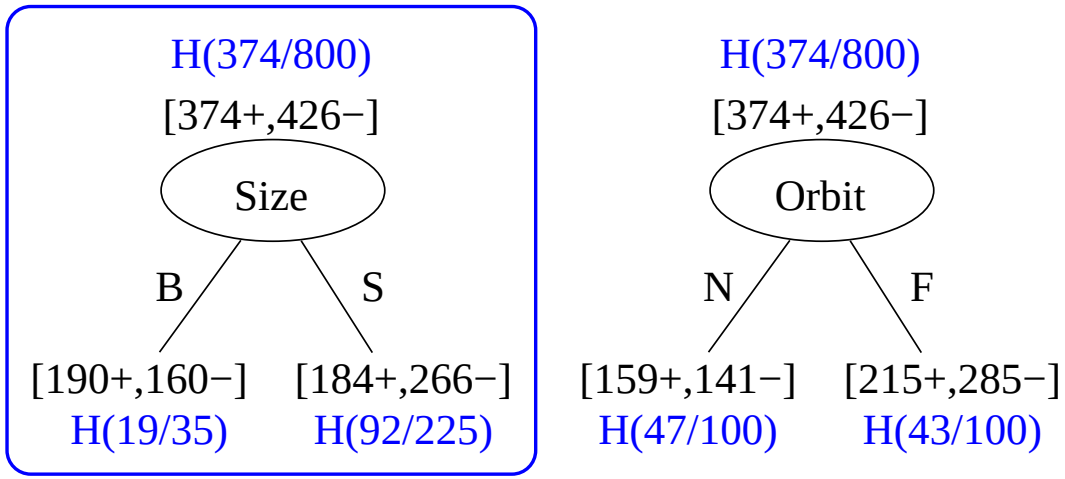
As of September 2012, 800 extrasolar planets have been identified in our galaxy. Super-secret surveying spaceships sent to all these planets have established whether they are habitable for humans or not, but sending a spaceship to each planet is expensive. In this problem, you will come up with decision trees to predict if a planet is habitable based only on features observable using telescopes.

a. In nearby table you are given the data from all 800 planets surveyed so far. The features observed by telescope are *Size* ("Big" or "Small"), and *Orbit* ("Near" or "Far"). Each row indicates the values of the features and habitability, and how many times that set of values was observed. So, for example, there were 20 "Big" planets "Near" their star that were habitable.

Size	Orbit	Habitable	Count
Big	Near	Yes	20
Big	Far	Yes	170
Small	Near	Yes	139
Small	Far	Yes	45
Big	Near	No	130
Big	Far	No	30
Small	Near	No	11
Small	Far	No	255

Derive and draw the decision tree learned by ID3 on this data (use the maximum information gain criterion for splits, don't do any pruning). Make sure to clearly mark at each node what attribute you are splitting on, and which value corresponds to which branch. By each leaf node of the tree, write in the number of habitable and inhabitable planets in the training data that belong to that node.

Answer: Level 1



$$H(Habitable|Size) = \frac{35}{80} \cdot H\left(\frac{19}{35}\right) + \frac{45}{80} \cdot H\left(\frac{92}{225}\right)$$

$$= \frac{35}{80} \cdot 0.9946 + \frac{45}{80} \cdot 0.9759 = 0.9841$$

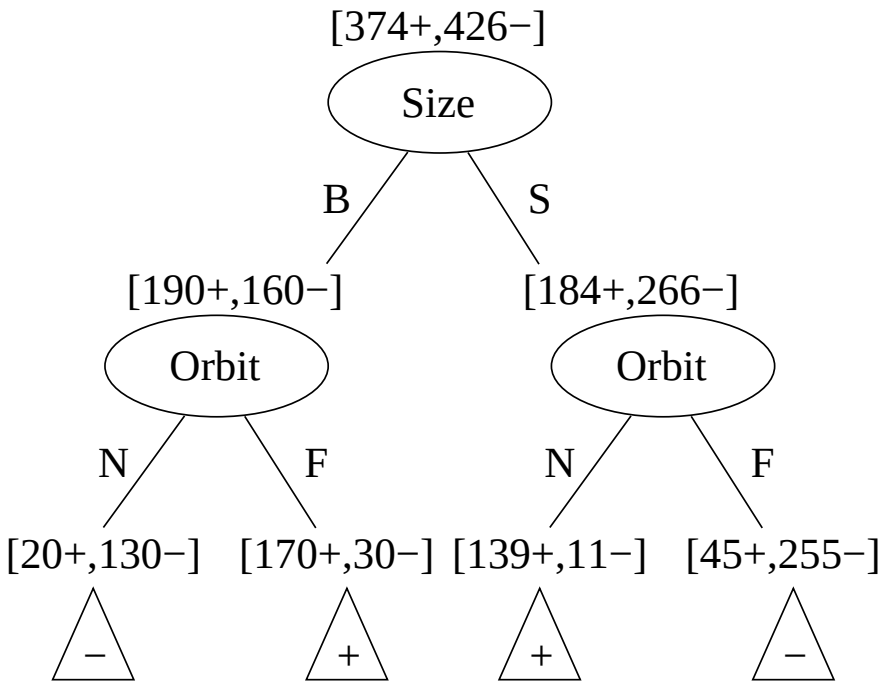
$$IG(Habitable; Size) = 0.0128$$

$$H(Habitable|Orbit) = \frac{3}{8} \cdot H\left(\frac{47}{100}\right) + \frac{5}{8} \cdot H\left(\frac{43}{100}\right)$$

$$= \frac{3}{8} \cdot 0.9974 + \frac{5}{8} \cdot 0.9858 = 0.9901$$

$$IG(Habitable; Orbit) = 0.0067$$

The final decision tree



b. For just 9 of the planets, a third feature, *Temperature* (in Kelvin degrees), has been measured, as shown in the nearby table.
Redo all the steps from part a on this data using all three features. For the *Temperature* feature, in each iteration you must maximize over all possible binary thresholding splits (such as $T \leq 250$ vs. $T > 250$, for example).

Size	Orbit	Temperature	Habitable
Big	Far	205	No
Big	Near	205	No
Big	Near	260	Yes
Big	Near	380	Yes
Small	Far	205	No
Small	Far	260	Yes
Small	Near	260	Yes
Small	Near	380	No
Small	Near	380	No

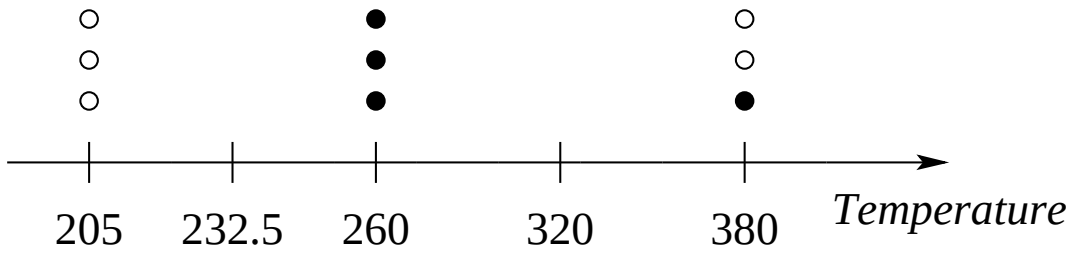
According to your decision tree, would a planet with the features (Big, Near, 280) be predicted to be habitable or not habitable?

***Hint*: You might need to use the following values of the entropy function for a Bernoulli variable of parameter p:**

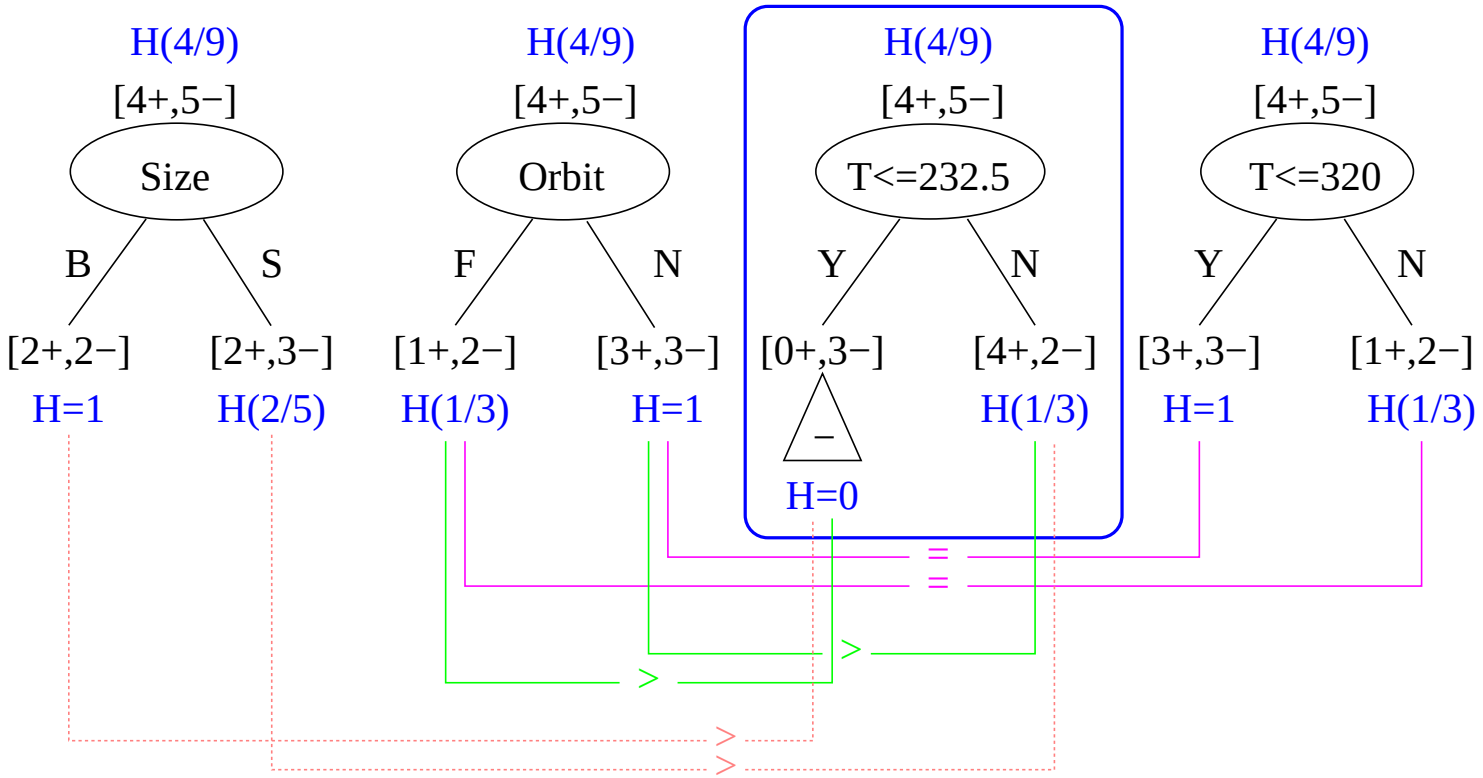
$H(1/3) = 0.9182$, $H(2/5) = 0.9709$, $H(92/225) = 0.9759$, $H(43/100) = 0.9858$, $H(16/35) = 0.9946$, $H(47/100) = 0.9974$.

Answer

Binary threshold splits for the continuous attribute *Temperature*:



Answer: Level 1



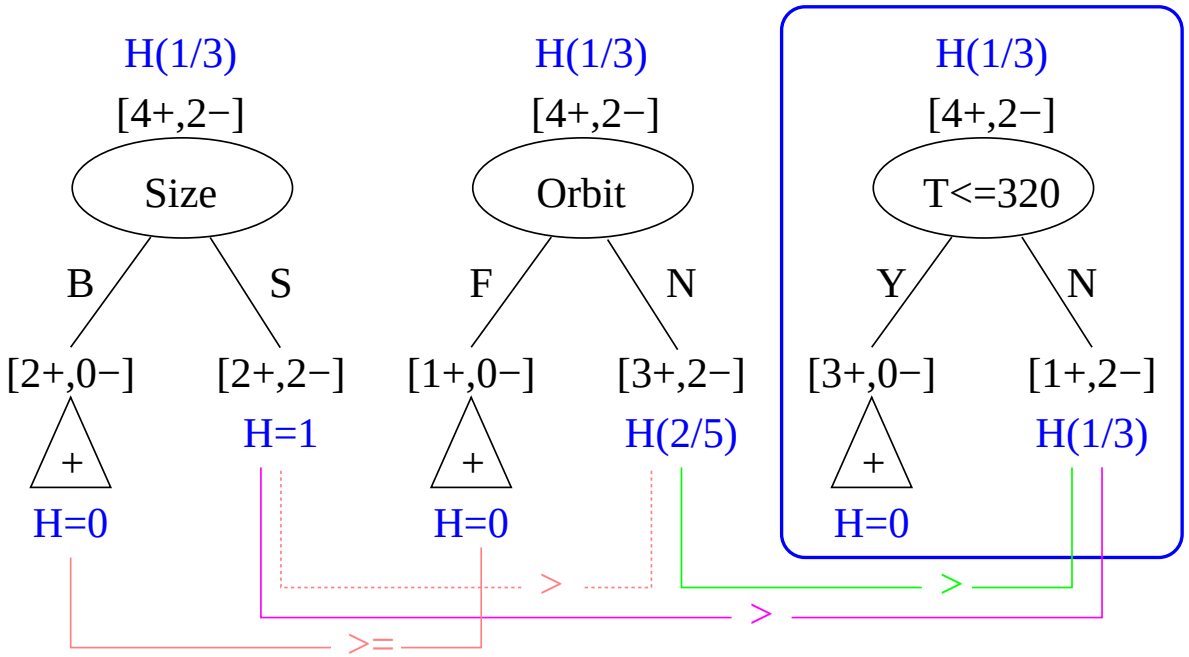
$$H(Habitable|Size) = \frac{4}{9} + \frac{5}{9} \cdot H\left(\frac{2}{5}\right) = \frac{4}{9} + \frac{5}{9} \cdot 0.9709 = 0.9838$$

$$H(Habitable|Temp \leq 232.5) = \frac{2}{3} \cdot H\left(\frac{1}{3}\right) = \frac{2}{3} \cdot 0.9182 = 0.6121$$

$$\begin{aligned} IG(Habitable; Size) &= H\left(\frac{187}{400}\right) - 0.9838 \\ &= 0.9969 - 0.9838 \\ &= 0.0072 \end{aligned}$$

$$IG(Habitable; Temp \leq 232.5) = 0.3788$$

Answer: Level 2

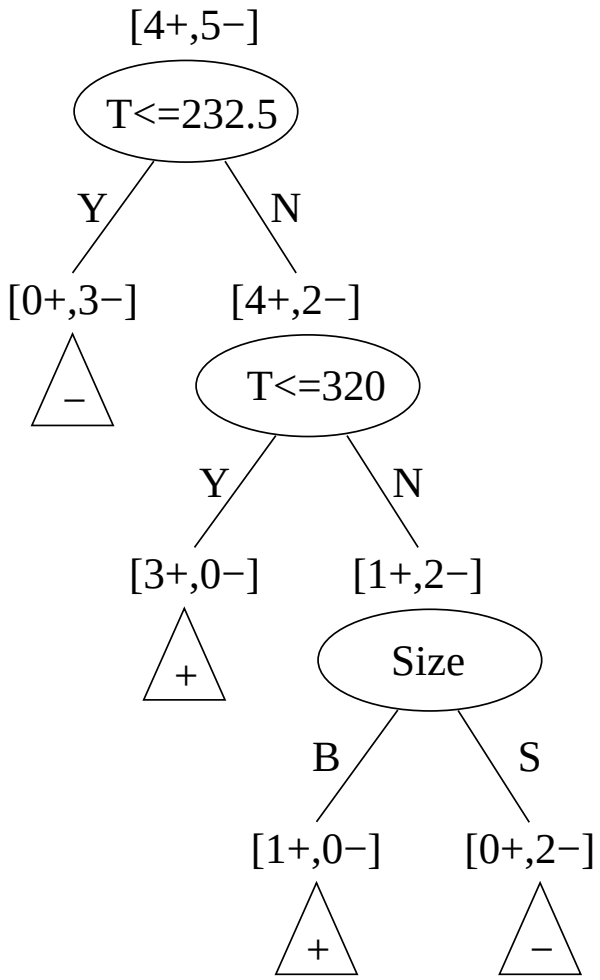


***Note*:**

The *plain lines* indicate that <u>both</u> the *specific conditional entropies* and their *coefficients* (weights) in the average conditional entropies satisfy the indicated relationship. (For example, $H(2/5) > H(1/3)$ and $5/6 > 3/6$.)

The *dotted lines* indicate that <u>only</u> the *specific conditional entropies* satisfy the indicated relationship. (For example, $H(2/2) = 1 > H(2/5)$ but $4/6 < 5/6$.)

The final decision tree:



c. According to your decision tree, would a planet with the features (Big, Near, 280) be predicted to be habitable or not habitable?

Answer: *habitable*.

Exemplifying

the application of the ID3 algorithm

on *continuous attributes*,

and in the presence of a *noise*.

Decision surfaces; decision boundaries.

The computation of the *LOOCV error*

CMU, 2002 fall, Andrew Moore, midterm, pr. 3

X	Y
1	0
2	0
3	0
4	0
6	1
7	1
8	1
8.5	0
9	1
10	1

Suppose we are learning a classifier with binary input values $Y = 0$ and $Y = 1$. There is one real-valued input X. The training data is given in the nearby table.

Assume that we learn a decision tree on this data. Assume that when the decision tree splits on the real-valued attribute X, it puts the *split threshold* halfway between the attributes that surround the split. For *example*, using information gain as splitting criterion, the decision tree would initially choos to split at $X = 5$, which is halfway between $X = 4$ and $X = 6$ datapoints.

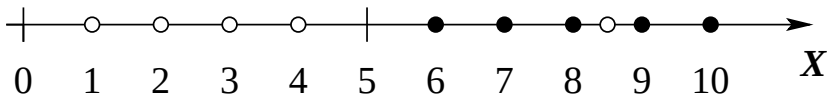
Let Algorithm DT2 be the method of learning a decision tree with only two leaf nodes (i.e., only one split).

Let Algorithm DT* be the method of learning a decision tree fully, with no prunning.

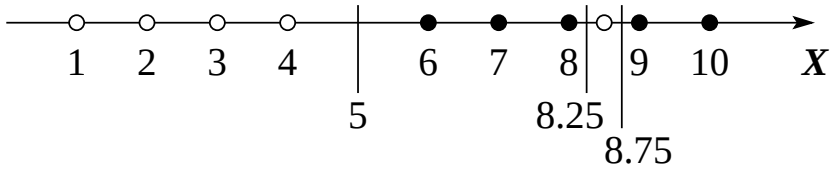
a. What will be the *training set error* for DT2 and respectively DT* on our data?

b. What will be the *leave-one-out cross-validation error* (LOOCV) for DT2 and respectively DT* on our data?

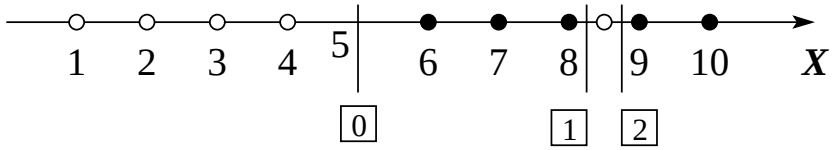
- **training data:**



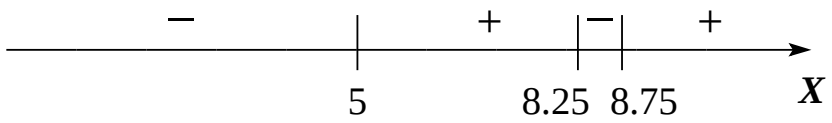
- **discretization / decision thresholds:**



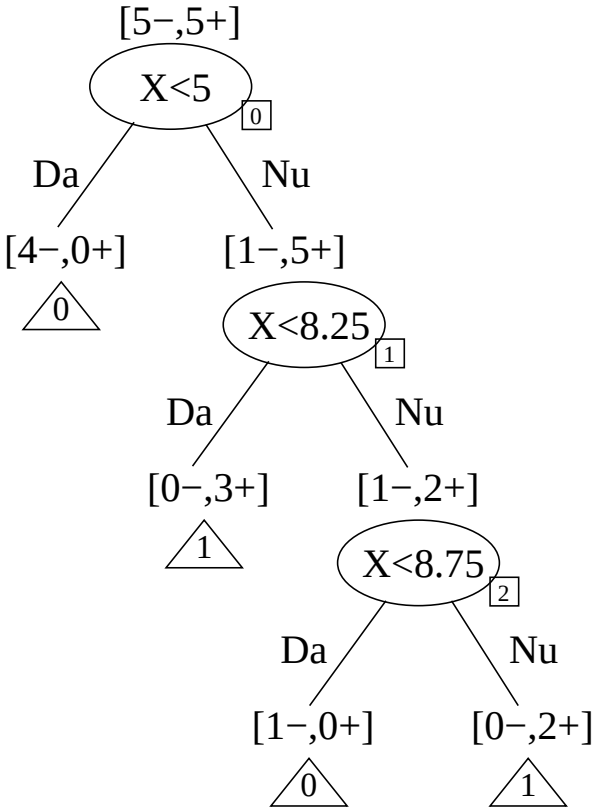
- **compact representation of the ID3 tree:**



- **decision "surfaces":**

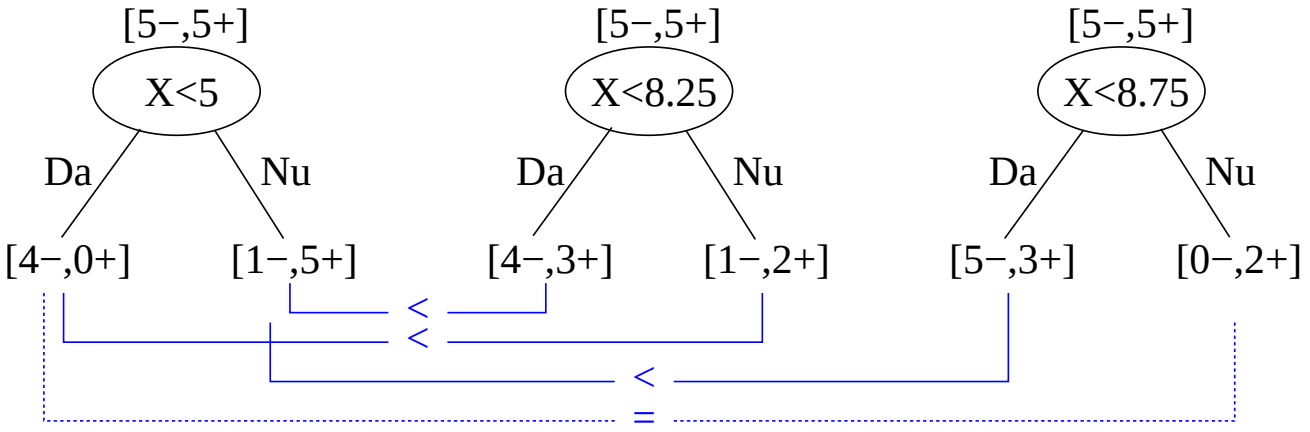


ID3 tree:

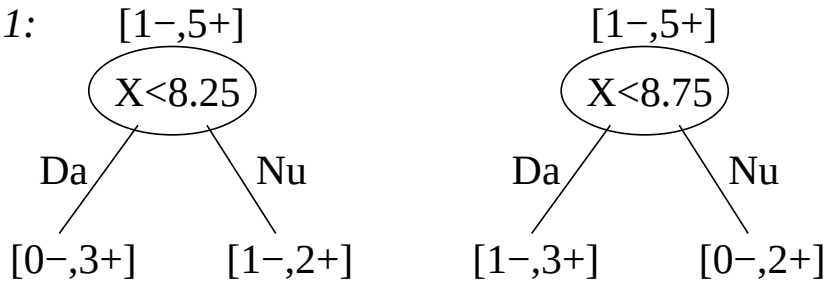


ID3:
IG computations

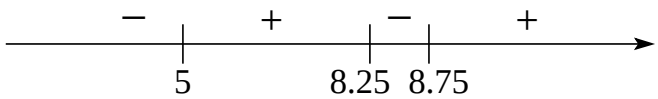
Level 0:



Level 1:

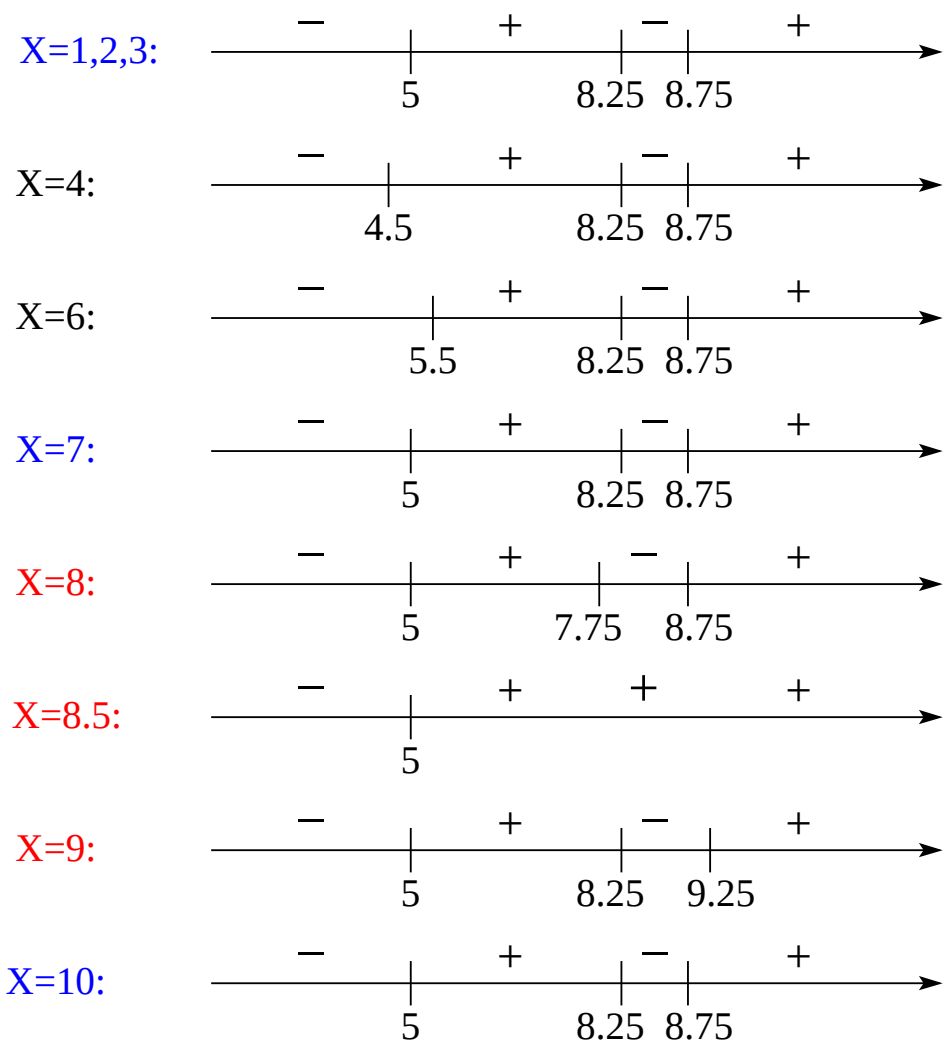


Decision "surfaces":

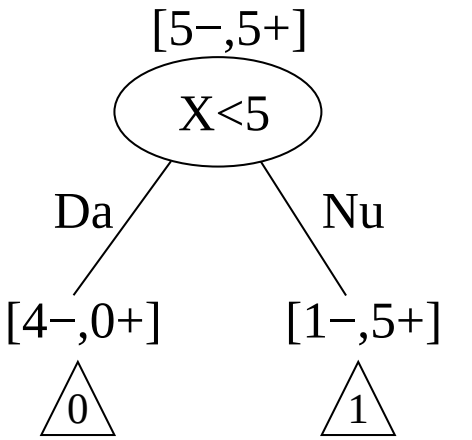


ID3, LOOCV:
Decision surfaces

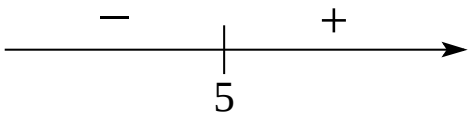
LOOCV error: 3/10



DT2



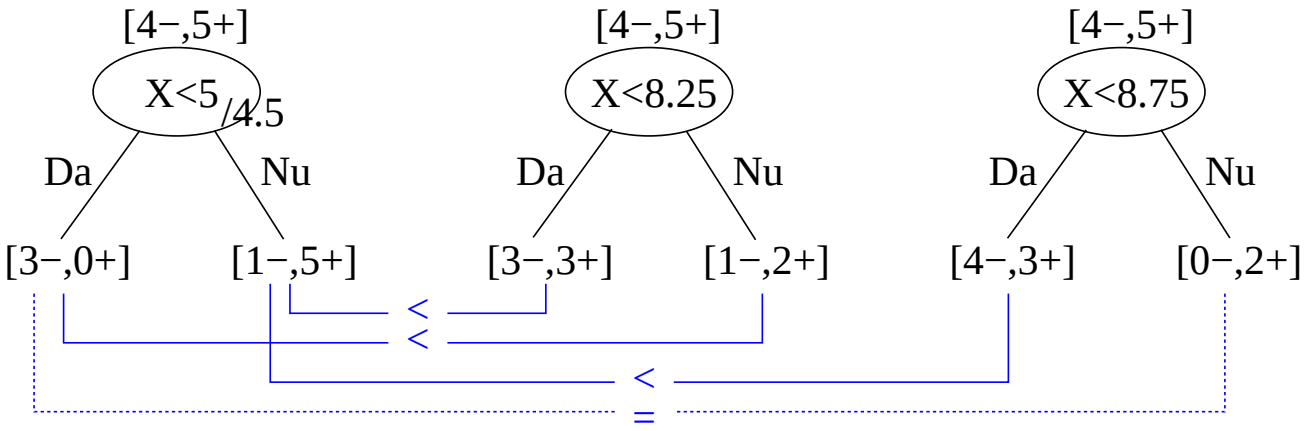
Decision "surfaces":



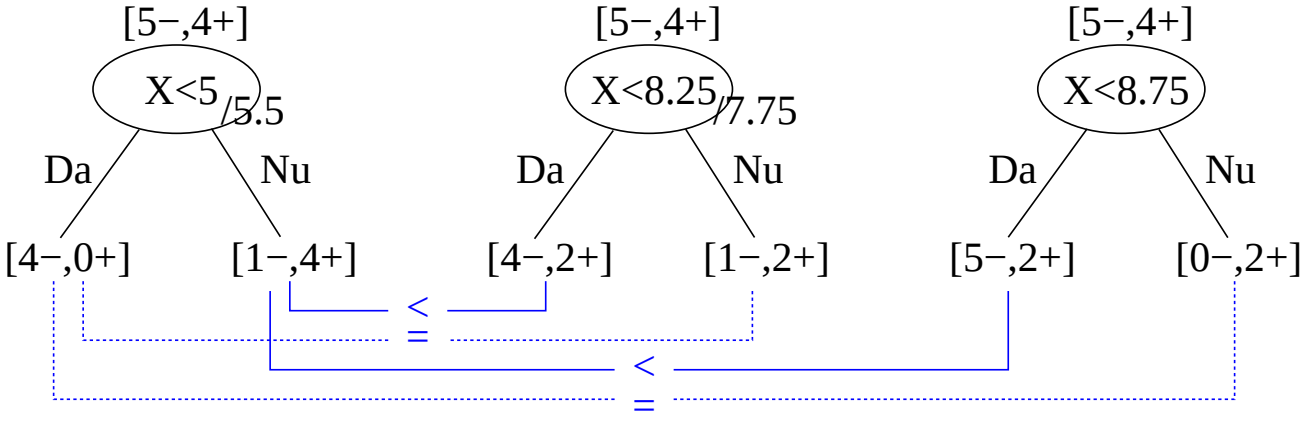
DT2, LOOCV

IG computations

Case 1: X=1, 2, 3, 4

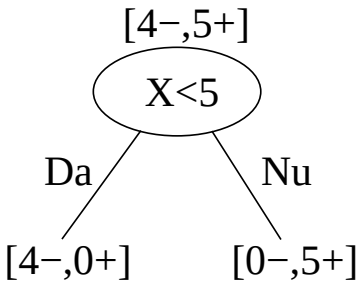


Case 2: X=6, 7, 8

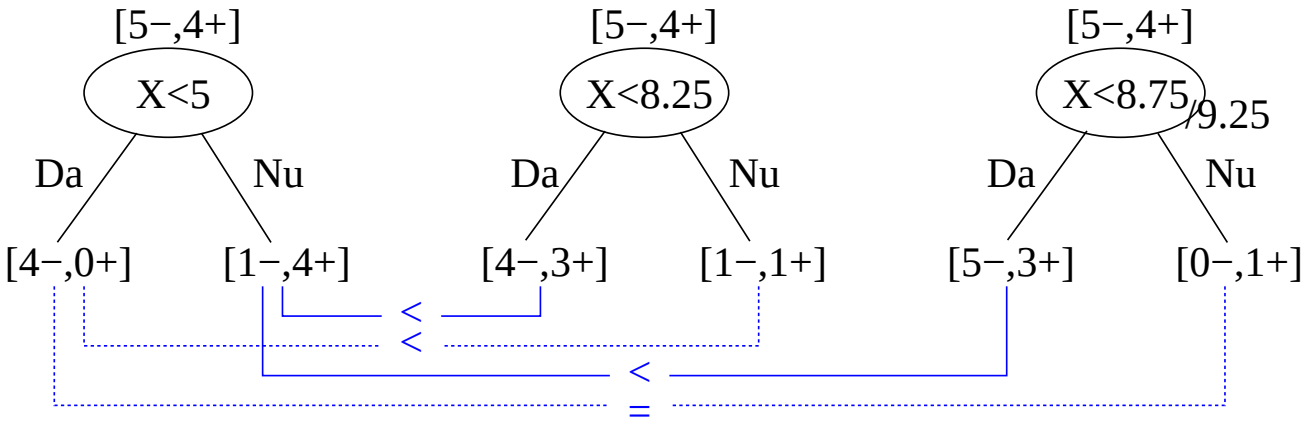


DT2, CVLOO
IG computations (cont'd)

Case 3: X=8.5



Case 2: X=9, 10



CVLOO error: 1/10

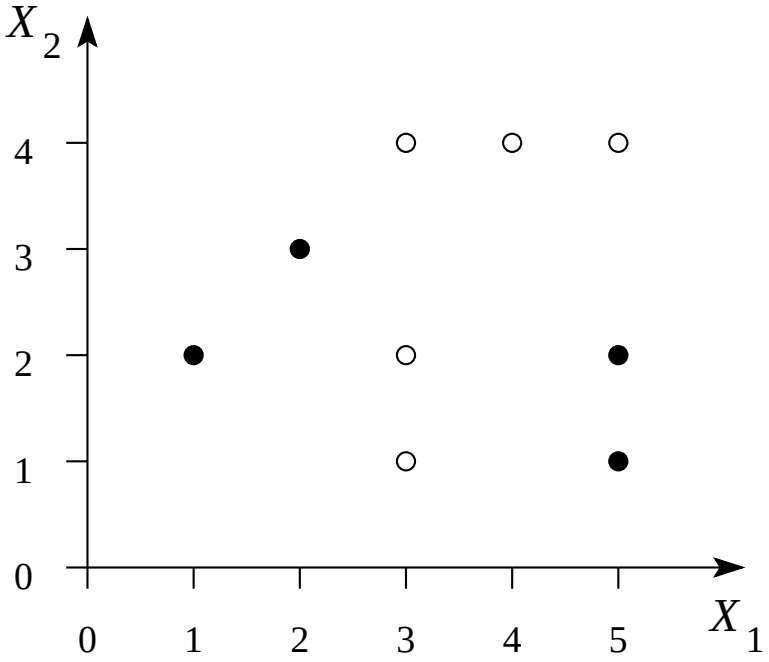
Applying ID3 on a dataset with two continuous attributes: decision zones

Liviu Ciortuz, 2017

Consider the training dataset in the nearby figure.
X_1 and X_2 are considered countinous attributes.

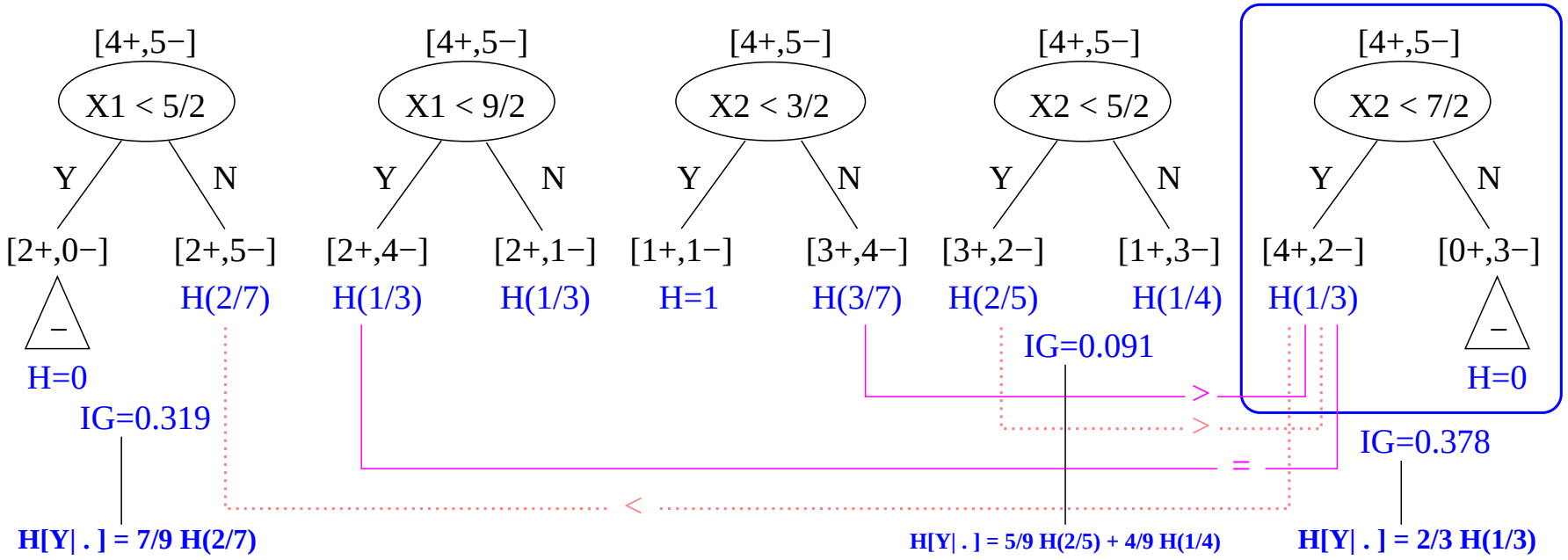
Apply the ID3 algorithm on this dataset. Draw the resulting decision tree.

Make a graphical representation of the decision areas and decision boundaries determined by ID3.

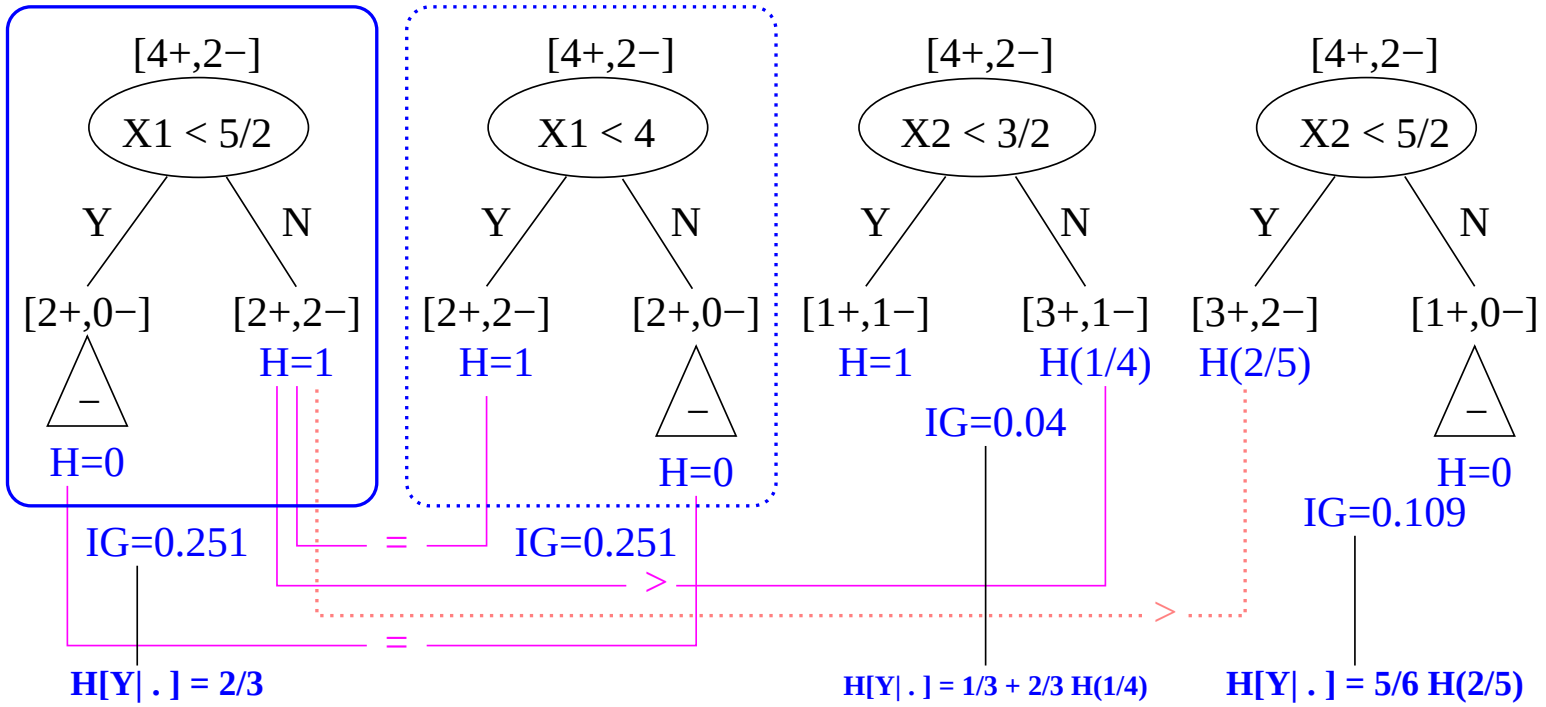


Solution

Level 1:



Level 2:



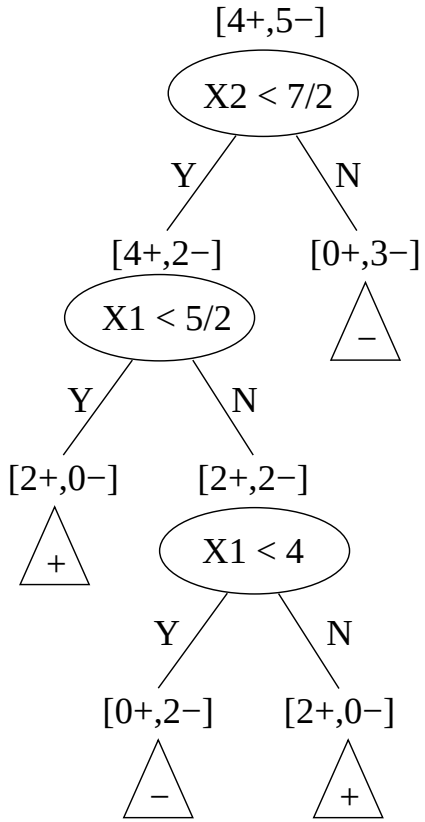
Notes:

1. Split thresholds for continuous attributes must be recomputed at each new iteration, because they may change. (For instance, here above, 4 replaces 4.5 as a threshold for X_1.)

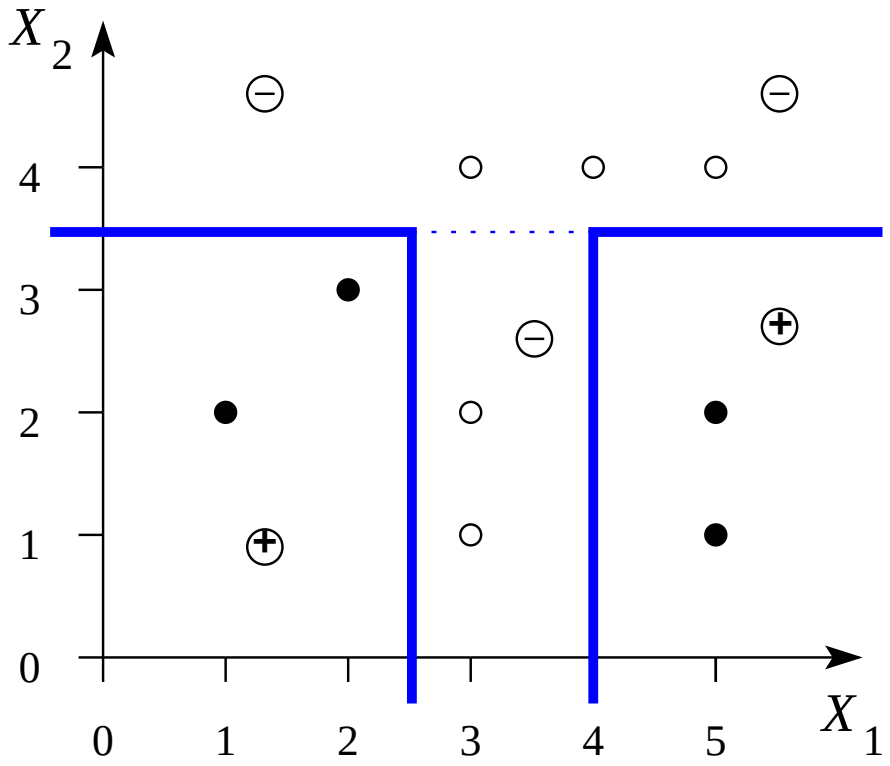
2. In the current stage, i.e., for the current node in the ID3 tree you may choose (as test) either $X_1 < 5/2$ or $X_1 < 4$.

3. Here above we have an example of reverse relationships between weighted and respectively un-weighted *specific entropies*: $H[2+,2-] > H[3+,2-]$ but $\frac{4}{6} \cdot H[2+,2-] < \frac{5}{6} \cdot H[3+,2-]$.

The final decision tree:



Decision areas:



Other criteria than IG for the best attribute selection in ID3:

Gini impurity / index and Misclassification impurity

CMU, 2003 fall, T. Mitchell, A. Moore, HW1, pr. 4

Entropy is a natural measure to quantify the impurity of a data set. The Decision Tree learning algorithm uses entropy as a splitting criterion by calculating the information gain to decide the next attribute to partition the current node.

However, there are other impurity measures that could be used as the splitting criteria too. Let's investigate two of them.

Assume the current node n has k classes $c_1, c_2, \ldots, c_k$.

- ***Gini Impurity*: $i(n) = 1 - \sum_{i=1}^{k} P^2(c_i)$.**
- ***Misclassification Impurity*: $i(n) = 1 - \max_{i=1}^{k} P(c_i)$.**

a. Assume node n has two classes, c_1 and c_2. Please draw a figure in which the three impurity measures (*Entropy*, *Gini* and *Misclassification*) are represented as the function of $P(c_1)$.

Answer

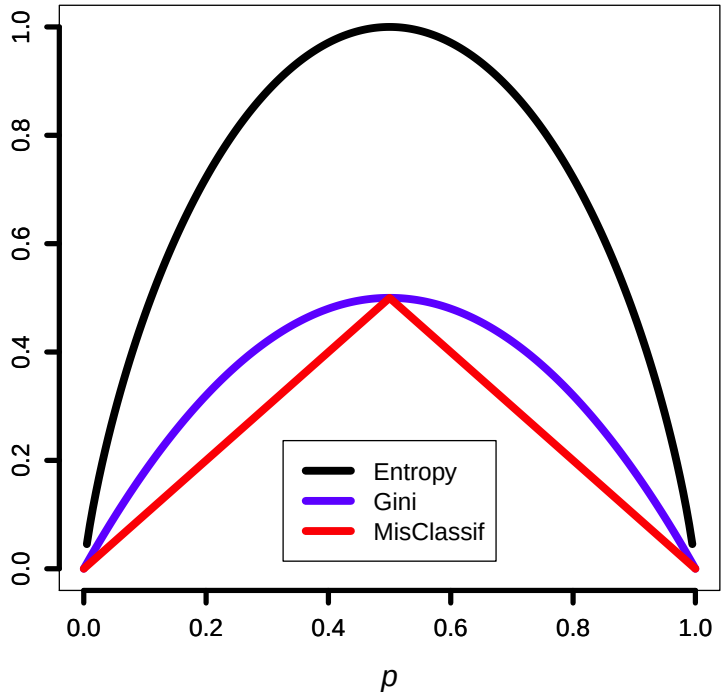
$$Entropy(p) = -p\log_2 p - (1-p)\log_2(1-p)$$

$$Gini(p) = 1 - p^2 - (1-p)^2 = 2p(1-p)$$

$$MisClassif(p) =$$

$$= \begin{cases} 1-(1-p), & \textbf{for } p \in [0;1/2) \\ 1-p, & \textbf{for } p \in [1/2;1] \end{cases}$$

$$= \begin{cases} p, & \textbf{for } p \in [0;1/2) \\ 1-p, & \textbf{for } p \in [1/2;1] \end{cases}$$



b. Now we can define new splitting criteria based on the *Gini* and *Misclassification* impurities, which is called *Drop-of-Impurity* in some literatures. That is the difference between the impurity of the current node and the weighted sum of the impurities of children.

For the binary category splits, *Drop-of-Impurity* is defined as

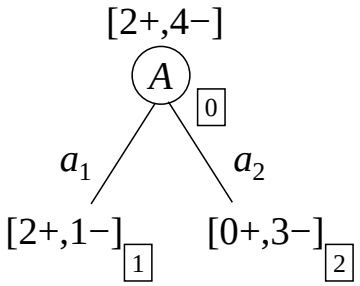
$$\Delta i(n) = i(n) - P(n_l)\, i(n_l) - P(n_r)\, i(n_r),$$

where n_l and n_r are the left and respectively the right child of node n after splitting.

Please calculate the *Drop-of-Impurity* (using both *Gini* and *Misclassification Impurity*) for the following example data set in which C is the class variable to be predicted.

A	a_1	a_1	a_1	a_2	a_2	a_2
C	c_1	c_1	c_2	c_2	c_2	c_2

Answer



Gini: $p = 2/6 = 1/3 \Rightarrow$

$$\left.\begin{array}{rcl} i(0) & = & 2 \cdot \frac{1}{3}(1 - \frac{1}{3}) = \frac{2}{3} \cdot \frac{2}{3} = \frac{4}{9} \\ i(1) & = & 2 \cdot \frac{2}{3}(1 - \frac{2}{3}) = \frac{4}{3} \cdot \frac{1}{3} = \frac{4}{9} \\ i(2) & = & 0 \end{array}\right\} \Rightarrow \Delta\, i(0) = \frac{4}{9} - \frac{3}{6} \cdot \frac{4}{9} = \frac{4}{9} - \frac{2}{9} = \frac{2}{9}.$$

Misclassification: $p = 1/3 < 1/2 \Rightarrow$

$$\left.\begin{array}{rcl} i(0) & = & p = \frac{1}{3} \\ i(1) & = & 1 - \frac{2}{3} = \frac{1}{3} \\ i(2) & = & 0 \end{array}\right\} \Rightarrow \Delta\, i(0) = \frac{1}{3} - \frac{1}{2} \cdot \frac{1}{3} = \frac{1}{6}.$$

c. We choose the attribute that can maximize the *Drop-of-Impurity* to split a node. Please create a data set and show that on this data set, *Misclassification Impurity* based $\Delta i(n)$ couldn't determine which attribute should be used for splitting (e.g., $\Delta i(n) = 0$ for all the attributes), but *Information Gain* and *Gini Impurity* based $\Delta i(n)$ can.

Answer

A	a_1	a_1	a_1	a_2	a_2	a_2	a_2
C	c_1	c_2	c_2	c_2	c_2	c_2	c_1

Entropy: $\Delta\, i(0) = H[5+, 2-] - \left(\frac{3}{7}H[2+, 1-] + \frac{4}{7}H[3+, 1-]\right) = 0.006 \neq 0$;

Gini: $2\left\{\frac{2}{7}\left(1-\frac{2}{7}\right) - \left[\frac{3}{7}\cdot\frac{1}{3}\left(1-\frac{1}{3}\right) + \frac{4}{7}\cdot\frac{1}{4}\left(1-\frac{1}{4}\right)\right]\right\} = 2\left\{\frac{10}{49} - \left[\frac{2}{21}+\frac{3}{28}\right]\right\}$

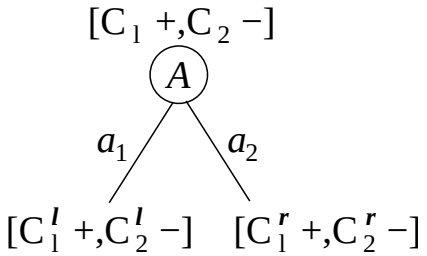
$= 2\left(\frac{10}{49} - \frac{17}{84}\right) \neq 0$;

Misclassification: $\Delta\, i(0) = \frac{2}{7} - \left(\frac{3}{7}\cdot\frac{1}{3} + \frac{4}{7}\cdot\frac{1}{4}\right) = 0.$

Note: A [quite bad] property

If $C_1 < C_2$, $C_1^l < C_2^l$ and $C_1^r < C_2^r$
(with $C_1 = C_1^l + C_1^r$ and $C_2 = C_2^l + C_2^r$),

then the *Drop-of-Impurity* based on *Misclassification* is 0.



Proof

$$\begin{aligned}\Delta i(n) &= \frac{C_1}{C_1+C_2} - \left(\frac{C_1^l + C_2^l}{C_1+C_2}\cdot\frac{C_1^l}{C_1^l+C_2^l} + \frac{C_1^r + C_2^r}{C_1+C_2}\cdot\frac{C_1^r}{C_1^r+C_2^r}\right)\\ &= \frac{C_1}{C_1+C_2} - \frac{C_1^l + C_1^r}{C_1+C_2} = \frac{C_1}{C_1+C_2} - \frac{C_1}{C_1+C_2} = 0.\end{aligned}$$

Exemplifying

pre- and post-pruning of decision trees

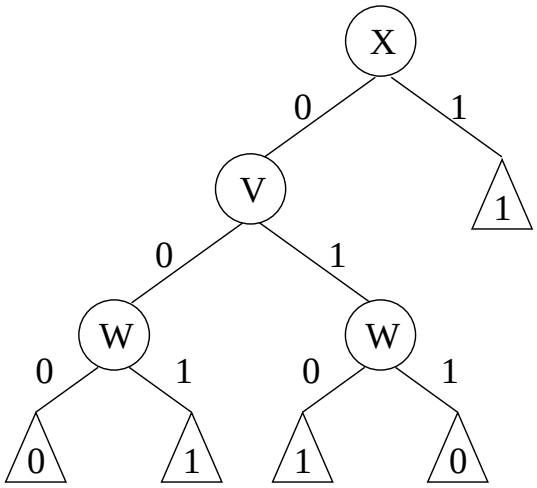
using a threshold for the Information Gain

CMU, 2006 spring, Carlos Guestrin, midterm, pr. 4

[adapted by Liviu Ciortuz]

Starting from the data in the following table, the ID3 algorithm builds the decision tree shown nearby.

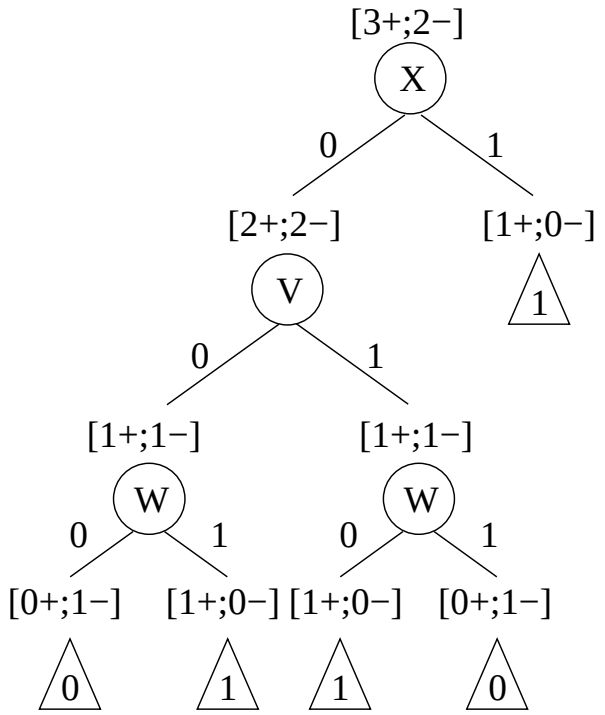
V	W	X	Y
0	0	0	0
0	1	0	1
1	0	0	1
1	1	0	0
1	1	1	1



a. One idea for pruning such a decision tree would be to start at the root, and prune *splits* for which the *information gain* (or some other criterion) is less than some small ε. This is called top-down pruning. What is the decision tree returned for $\varepsilon = 0.0001$? What is the training set error for this tree?

Answer

We will first augment the given decision tree with informations regarding the *data partitions* (i.e., the number of positive and, respectively, negative instances) which were assigned to each test node during the application of ID3 algorithm.



The information gain yielded by the attribute X in the root node is:

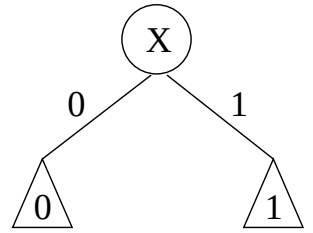
$$H[3+;2-] - 1/5 \cdot 0 - 4/5 \cdot 1 = 0.971 - 0.8 = 0.171 > \varepsilon.$$

Therefore, this node will not be eliminated from the tree.

The information gain for the attribute V (in the left-hand side child of the root node) is:

$$H[2+;2-] - 1/2 \cdot 1 - 1/2 \cdot 1 = 1 - 1 = 0 < \varepsilon.$$

So, the whole left subtree will be cut off and replaced by a decision node, as shown nearby. The training error produced by this tree is $2/5$.



b. Another option would be to start at the leaves, and prune subtrees for which the information gain (or some other criterion) of a split is less than some small ε. In this method, no ancestors of children with high information gain will get pruned. This is called bottom-up pruning. What is the tree returned for $\varepsilon = 0.0001$? What is the training set error for this tree?

Answer:

The information gain of V is $IG(Y;V) = 0$. A step later, the information gain of W (for either one of the descendent nodes of V) is $IG(Y;W) = 1$. So bottom-up pruning won't delete any nodes and the tree [given in the problem statement] remains unchanged.

The training error is 0.

c. Discuss when you would want to choose bottom-up pruning over top-down pruning and vice versa.

Answer:

Top-down pruning **is** ***computationally cheaper*****. When building the tree we can determine when to stop (no need for real pruning). But as we saw top-down pruning prunes too much.**

On the other hand, ***bottom-up pruning*** **is** ***more expensive*** **since we have to first build a full tree — which can be exponentially large — and then apply pruning. The second** ***problem*** **with bottom-up pruning is that supperfluous attributes may fullish it (see CMU, CMU, 2009 fall, Carlos Guestrin, HW1, pr. 2.4). The third** ***problem*** **with it is that in the lower levels of the tree** ***the number of examples*** **in the subtree** ***gets smaller*** **so information gain might be an inappropriate criterion for pruning, so one would usually use a** ***statistical test*** **instead.**

Exemplifying

χ^2-Based Pruning of Decision Trees

CMU, 2010 fall, Ziv Bar-Joseph, HW2, pr. 2.1

In class, we learned a decision tree pruning algorithm that iteratively visited subtrees and used a *validation dataset* to decide whether to remove the subtree. However, sometimes it is desirable to prune the tree after *training* on all of the available data.

One such approach is based on statistical hypothesis testing.

After learning the tree, we visit each internal node and *test* whether the *attribute* split at that node is actually *uncorrelated with* the *class labels*.

We hypothesize that the attribute is independent and then *use Pearson's chi-square test* to generate a test statistic that may provide evidence that we should *reject* this *"null" hypothesis*. If we fail to reject the hypothesis, we prune the subtree at that node.

a. At each internal node we can create a contingency table for the training examples that pass through that node on their paths to the leaves. The table will have the c class labels associated with the columns and the r values the split attribute associated with the rows.

Each entry $O_{i,j}$ in the table is the number of times we observe a training sample with that attribute value and label, where i is the row index that corresponds to an attribute value and j is the column index that corresponds to a class label.

In order to calculate the chi-square test statistic, we need a similar table of expected counts. The expected count is the number of observations we would expect if the class and attribute are independent.

Derive a formula for each expected count $E_{i,j}$ in the table.

***Hint*: What is the probability that a training example that passes through the node has a particular label? Using this probability and the independence assumption, what can you say about how many examples with a specific attribute value are expected to also have the class label?**

b. Given these two tables for the split, you can now calculate the chi-square test statistic

$$\chi^2 = \sum_{i=1}^{r} \sum_{j=1}^{c} \frac{(O_{i,j} - E_{i,j})^2}{E_{i,j}}$$

with degrees of freedom $(r-1)(c-1)$.

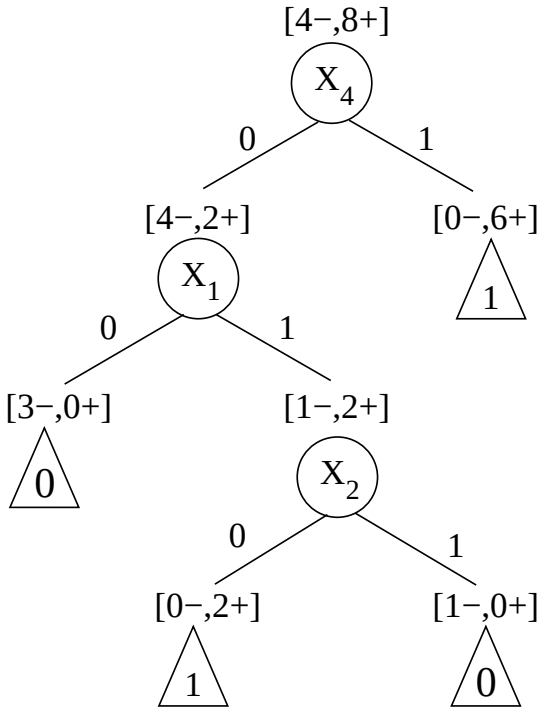
You can plug the test statistic and degrees of freedom into a software package[a] **or an online calculator**[b] **to calculate a** p**-value. Typically, if** $p < 0.05$ **we reject the null hypothesis that the attribute and class are independent and say the split is statistically significant.**

The decision tree given on the next slide was built from the data in the table nearby.
For each of the 3 internal nodes in the decision tree, show the p**-value for the split and state whether it is statistically significant.**
How many internal nodes will the tree have if we prune splits with $p \geq 0.05$**?**

[a] Use 1-chi2cdf(x,df) in MATLAB or CHIDIST(x,df) in Excel.
[b] (https://en.m.wikipedia.org/wiki/Chi-square_distribution#Table_of_.CF.872_value_vs_p-value.

Input:

X_1	X_2	X_3	X_4	***Class***
1	1	0	0	0
1	0	1	0	1
0	1	0	0	0
1	0	1	1	1
0	1	1	1	1
0	0	1	0	0
1	0	0	0	1
0	1	0	1	1
1	0	0	1	1
1	1	0	1	1
1	1	1	1	1
0	0	0	0	0



Idea

While traversing the ID3 tree [usually in bottom-up manner],
remove the nodes for which
there is not enough ("significant") statistical evidence that
there is a dependence between
the values of the input attribute tested in that node and the values of the output attribute (the labels),
supported by the set of instances assigned to that node.

Contingency tables

O_{X_4}	$\boldsymbol{Class\ =\ 0}$	$\boldsymbol{Class} = 1$
$X_4 = 0$	4	2
$X_4 = 1$	0	6

$$\overset{N=12}{\Rightarrow} \begin{cases} P(X_4 = 0) = \frac{6}{12} = \frac{1}{2},\ P(X_4 = 1) = \frac{1}{2} \\ P(\boldsymbol{Class} = 0) = \frac{4}{12} = \frac{1}{3},\ P(\boldsymbol{Class} = 1) = \frac{2}{3} \end{cases}$$

$O_{X_1 \mid X_4=0}$	$\boldsymbol{Class} = 0$	$\boldsymbol{Class} = 1$
$X_1 = 0$	3	0
$X_1 = 1$	1	2

$$\overset{N=6}{\Rightarrow} \begin{cases} P(X_1 = 0 \mid X_4 = 0) = \frac{3}{6} = \frac{1}{2} \\ P(X_1 = 1 \mid X_4 = 0) = \frac{1}{2} \\ P(\boldsymbol{Class} = 0 \mid X_4 = 0) = \frac{4}{6} = \frac{2}{3} \\ P(\boldsymbol{Class} = 1 \mid X_4 = 0) = \frac{1}{3} \end{cases}$$

$O_{X_2 \mid X_4=0, X_1=1}$	$\boldsymbol{Class} = 0$	$\boldsymbol{Class} = 1$
$X_2 = 0$	0	2
$X_2 = 1$	1	0

$$\overset{N=3}{\Rightarrow} \begin{cases} P(X_2 = 0 \mid X_4 = 0, X_1 = 1) = \frac{2}{3} \\ P(X_2 = 1 \mid X_4 = 0, X_1 = 1) = \frac{1}{3} \\ P(\boldsymbol{Class} = 0 \mid X_4 = 0, X_1 = 1) = \frac{1}{3} \\ P(\boldsymbol{Class} = 1 \mid X_4 = 0, X_1 = 1) = \frac{2}{3} \end{cases}$$

The reasoning that leads to the computation of the expected number of observations

$$P(A = i, C = j) = P(A = i) \cdot P(C = j)$$

$$P(A = i) = \frac{\sum_{k=1}^{c} O_{i,k}}{N} \textbf{ and } P(C = j) = \frac{\sum_{k=1}^{r} O_{k,j}}{N}$$

$$P(A = i, C = j) \overset{\textbf{\textit{indep.}}}{=} \frac{\left(\sum_{k=1}^{c} O_{i,k}\right)\left(\sum_{k=1}^{r} O_{k,j}\right)}{N^2}$$

$$E[A = i, C = j] = N \cdot P(A = i, C = j)$$

Expected number of observations

E_{X_4}	$\boldsymbol{Class} = 0$	$\boldsymbol{Class} = \boldsymbol{1}$
$X_4 = 0$	2	4
$X_4 = 1$	2	4

$E_{X_1 \mid X_4}$	$\boldsymbol{Class} = 0$	$\boldsymbol{Class} = \boldsymbol{1}$
$X_1 = 0$	2	1
$X_1 = 1$	2	1

$E_{X_2 \mid X_4, X_1 = 1}$	$\boldsymbol{Class} = 0$	$\boldsymbol{Class} = \boldsymbol{1}$
$X_2 = 0$	$\frac{2}{3}$	$\frac{4}{3}$
$X_2 = 1$	$\frac{1}{3}$	$\frac{2}{3}$

$E_{X_4}(X_4 = 0, \boldsymbol{Class} = 0)$:

$N = 12,\ P(X_4 = 0) = \frac{1}{2}$ **şi** $P(\boldsymbol{Class} = 0) = \frac{1}{3} \Rightarrow$

$$N \cdot P(X_4 = 0, \boldsymbol{Class} = 0) = N \cdot P(X_4 = 0) \cdot P(\boldsymbol{Class} = 0) = 12 \cdot \frac{1}{2} \cdot \frac{1}{3} = 2$$

χ^2 Statistics

$$\chi^2 = \sum_{i=1}^{r}\sum_{j=1}^{c}\frac{(O_{i,j}-E_{i,j})^2}{E_{i,j}}$$

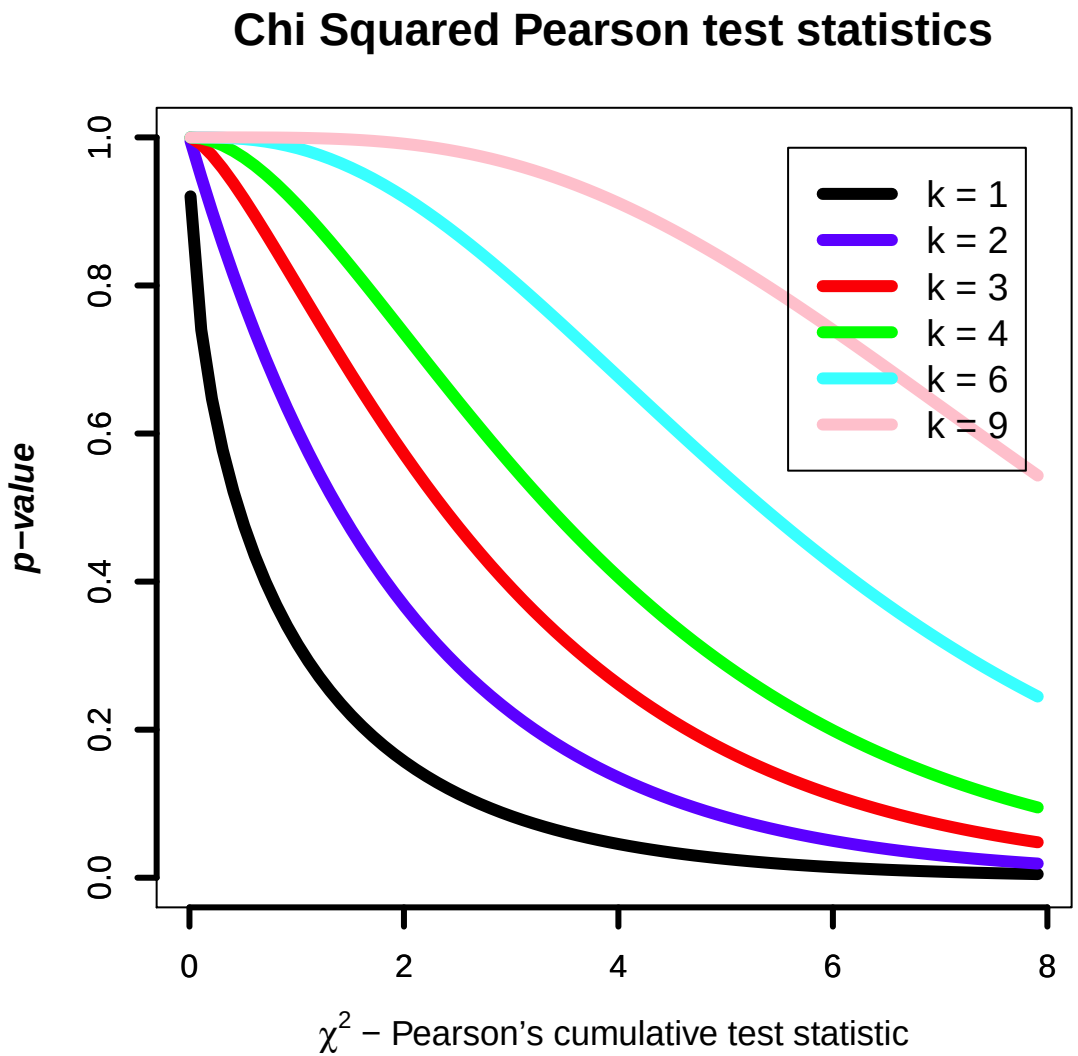
$$\chi^2_{X_4} = \frac{(4-2)^2}{2}+\frac{(0-2)^2}{2}+\frac{(2-4)^2}{4}+\frac{(6-4)^2}{4} = 2+2+1+1 = 6$$

$$\chi^2_{X_1|X_4=0} = \frac{(3-2)^2}{2}+\frac{(1-2)^2}{2}+\frac{(0-1)^2}{1}+\frac{(2-1)^2}{1} = 3$$

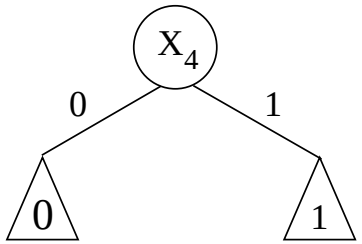
$$\chi^2_{X_2|X_4=0,X_1=1} = \frac{\left(0-\frac{2}{3}\right)^2}{\frac{2}{3}}+\frac{\left(1-\frac{1}{3}\right)^2}{\frac{1}{3}}+\frac{\left(2-\frac{4}{3}\right)^2}{\frac{4}{3}}+\frac{\left(0-\frac{2}{3}\right)^2}{\frac{2}{3}} = \frac{4}{9}\cdot\frac{27}{4} = 3$$

p-values: 0.0143, 0.0833, **and respectively** 0.0833.

The first one of these p-values is smaller than ε, therefore the root node (X_4) cannot be prunned.


Chi Squared Pearson test statistics
p−value
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8
k = 1
k = 2
k = 3
k = 4
k = 6
k = 9
χ² − Pearson's cumulative test statistic

Output (pruned tree) for 95% confidence level



The AdaBoost algorithm:

The convergence – in certain conditions to 0 – of the *training error*

CMU, 2015 fall, Ziv Bar-Joseph, Eric Xing, HW4, pr. 2.1-5
CMU, 2009 fall, Carlos Guestrin, HW2, pr. 3.1
CMU, 2009 fall, Eric Xing, HW3, pr. 4.2.2

Consider m training examples $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, where $x \in \mathcal{X}$ and $y \in \{-1, 1\}$. Suppose we have a *weak learning algorithm* A which produces a hypothesis $h : \mathcal{X} \to \{-1, 1\}$ given any distribution D of examples.

AdaBoost is an iterative algorithm which works as follows:

- **Begin with a uniform distribution $D_1(i) = \frac{1}{m}$, $i = 1, \ldots, m$.**
- **At each round $t = 1, \ldots, T$,**
  - **run the weak learning algorithm A on the distribution D_t and produce the hypothsis h_t;**

    ***Note*: Since A is a weak learning algorithm, the produced hypothesis h_t at round t is only slightly better than random guessing, say, by a margin γ_t:**

    $$\varepsilon_t = err_{D_t}(h_t) = \Pr_{x \sim D_t}[y \neq h_t(x)] = \frac{1}{2} - \gamma_t.$$

    [If at a certain iteration $t < T$ the weak classifier A cannot produce a hypothesis better than random guessing (i.e., $\gamma_t = 0$) or it produces a hypothesis for which $\varepsilon_t = 0$, then the AdaBoost algorithm should be stopped.]
  - **update $D_{t+1}(i) = \frac{1}{Z_t} \cdot D_t(i) \cdot e^{-\alpha_t y_i h_t(x_i)}$ for $i = 1, \ldots, m$,**

    where $\alpha_t \overset{not.}{=} \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$, and Z_t is the normalizer.
- **In the end, deliver $H_T = sign\left(\sum_{t=1}^{T} \alpha_t h_t\right)$ as the learned hypothesis, which will act as a *weighted majority vote*.**

We will prove that the training error $err_S(H_T)$ of AdaBoost decreases at a very fast rate, and in certain cases it converges to 0.

Important Remark

The above formulation of the AdaBoost algorithm states *no restriction* on the h_t hypothesis delivered by the weak classifier A at iteration t, except that $\varepsilon_t < 1/2$.

However, in another formulation of the AdaBoost algorithm (in a more general setup; see for instance MIT, 2006 fall, Tommi Jaakkola, HW4, problem 3), it is requested / reccommended that hypothesis h_t be chosen by *(approximately) maximizing* the [criterion of] *weighted training error* on a whole class of hypotheses like, for instance, decision trees of depth 1 (decision stumps).

In this problem we will not be concerned with such a *request*, but we will comply with it for instance in problem CMU, 2015 fall, Ziv Bar-Joseph, Eric Xing, HW4, pr2.6, when showing how AdaBoost works in practice.

a. Show that the [particular] choice of $\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$ **implies** $err_{D_{t+1}}(h_t) = 1/2$**.**

b. Show that $D_{T+1}(i) = \left(m \cdot \prod_{t=1}^{T} Z_t\right)^{-1} e^{-y_i f(x_i)}$**, where** $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$**.**

c. Show that $err_S(H_T) \leq \prod_{t=1}^{T} Z_t$**, where** $err_S(H_T) \overset{not.}{=} \frac{1}{m}\sum_{i=1}^{m} 1_{\{H_T(x_i)\neq y_i\}}$ **is the traing error produced by AdaBoost.**

d. Obviously, we would like to *minimize* test set error produced by AdaBoost, but it is hard to do so directly. We thus settle for *greedily* optimizing the *upper bound* on the *training error* found at part c**.** Observe that $Z_1, \ldots, Z_{t-1}$ are determined by the first $t-1$ iterations, and we cannot change them at iteration t. A greedy step we can take to minimize the training set error bound on round t is to minimize Z_t. **Prove that the value of** α_t **that minimizes** Z_t **(among all possible values for** α_t**) is indeed** $\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$ **(see the previous slide).**

e. Show that $\prod_{t=1}^{T} Z_t \leq e^{-2\sum_{t=1}^{T}\gamma_t^2}$**.**

f. From part c **and** d**, we know the training error decreases at exponential rate with respect to** T**. Assume that there is a number** $\gamma > 0$ **such that** $\gamma \leq \gamma_t$ **for** $t = 1, \ldots, T$**. (This is called *empirical* γ*-weak learnability*.) How many rounds are needed to achieve a training error** $\varepsilon > 0$**? Please express in big-**$\mathcal{O}$ **notation,** $T = \mathcal{O}(\cdot)$**.**

Solution

a. If we define the correct set $C = \{i : y_i h_t(x_i) \geq 0\}$ and the mistake set $M = \{i : y_i h_t(x_i) < 0\}$, we can write:

$$err_{D_{t+1}}(h_t) = \sum_{i=1}^{m} D_{t+1}(i) \cdot 1_{\{y_i \neq h_t(x_i)\}} = \sum_{i \in M} \frac{1}{Z_t} D_t(i) e^{\alpha_t} = \frac{1}{Z_t} \underbrace{\sum_{i \in M} D_t(i)}_{\varepsilon_t} e^{\alpha_t} = \frac{1}{Z_t} \cdot \varepsilon_t \cdot e^{\alpha_t}$$

$$Z_t = \sum_{i=1}^{m} D_t(i) e^{-\alpha_t y_i h_t(x_i)} = \sum_{i \in C} D_t(i) e^{-\alpha_t} + \sum_{i \in M} D_t(i) e^{\alpha_t} = (1 - \varepsilon_t) \cdot e^{-\alpha_t} + \varepsilon_t \cdot e^{\alpha_t} \qquad (1)$$

$$e^{\alpha_t} = e^{\frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}} = e^{\ln \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}}} = \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} \quad \textbf{and} \quad e^{-\alpha_t} = \frac{1}{e^{\alpha_t}} = \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}}$$

$$\Rightarrow Z_t = (1 - \varepsilon_t) \cdot \sqrt{\frac{\varepsilon_t}{1-\varepsilon_t}} + \varepsilon_t \cdot \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} = 2\sqrt{\varepsilon_t(1-\varepsilon_t)} \qquad (2)$$

$$\Rightarrow err_{D_{t+1}}(h_t) = \frac{1}{Z_t} \cdot \varepsilon_t \cdot e^{\alpha_t} = \frac{1}{2\sqrt{\varepsilon_t(1-\varepsilon_t)}} \cdot \varepsilon_t \cdot \sqrt{\frac{1-\varepsilon_t}{\varepsilon_t}} = \frac{1}{2}$$

Note that $\frac{1-\varepsilon_t}{\varepsilon_t} > 0$ because $\varepsilon_t \in (0, 1/2)$.

b. We will expand $D_t(i)$ recursively:

$$\begin{aligned}
D_{T+1}(i) &= \frac{1}{Z_T}\, D_T(i)\, e^{-\alpha_T\, y_i\, h_T(x_i)} = D_T(i)\, \frac{1}{Z_T}\, e^{-\alpha_T\, y_i\, h_T(x_i)} \\
&= D_{T-1}(i)\, \frac{1}{Z_{T-1}}\, e^{-\alpha_{T-1}\, y_i\, h_{T-1}(x_i)}\, \frac{1}{Z_T}\, e^{-\alpha_T\, y_i\, h_T(x_i)} \\
&\vdots \\
&= D_1(i)\, \frac{1}{\prod_{t=1}^{T} Z_t}\, e^{-\sum_{t=1}^{T} \alpha_t\, y_i\, h_t(x_i)} = \frac{1}{m \cdot \prod_{t=1}^{T} Z_t}\, e^{-y_i\, f(x_i)}.
\end{aligned}$$

c. We will make use of the fact that exponential loss upper bounds the 0-1 loss, i.e. $1_{\{x<0\}} \leq e^{-x}$:

$$\begin{aligned}
err_S(H_T) &= \frac{1}{m} \sum_{i=1}^{m} 1_{\{y_i f(x_i)<0\}} \\
&\leq \frac{1}{m} \sum_{i=1}^{m} e^{-y_i\, f(x_i)} = \frac{1}{m} \sum_{i=1}^{m} D_{T+1}(i) \cdot m \cdot \prod_{t=1}^{T} Z_t = \sum_{i=1}^{m} D_{T+1}(i) \prod_{t=1}^{T} Z_t \\
&= \underbrace{\left(\sum_{i=1}^{m} D_{T+1}(i) \right)}_{1} \cdot \left(\prod_{t=1}^{T} Z_t \right) = \prod_{t=1}^{T} Z_t.
\end{aligned}$$

d. We will start from the equation

$$Z_t = \varepsilon_t \cdot e^{\alpha_t} + (1 - \varepsilon_t) \cdot e^{-\alpha_t},$$

which has been proven at part a. Note that the right-hand side is constant with respect to ε_t (the error produced by h_t, the hypothesis produced by the weak classifier A at the current step).
Then we will proceed as usually, computing the partial derivative w.r.t. ε_t:

$$\frac{\partial}{\partial \alpha_t}\left(\varepsilon_t \cdot e^{\alpha_t} + (1 - \varepsilon_t) \cdot e^{-\alpha_t}\right) = 0 \Leftrightarrow \varepsilon_t \cdot e^{\alpha_t} - (1 - \varepsilon_t) \cdot e^{-\alpha_t} = 0$$

$$\Leftrightarrow \varepsilon_t \cdot (e^{\alpha_t})^2 = 1 - \varepsilon_t \Leftrightarrow e^{2\alpha_t} = \frac{1 - \varepsilon_t}{\varepsilon_t} \Leftrightarrow 2\alpha_t = \ln \frac{1 - \varepsilon_t}{\varepsilon_t} \Leftrightarrow \alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}.$$

Note that $\frac{1 - \varepsilon_t}{\varepsilon_t} > 1$ (and therefore $\alpha_t > 0$) because $\varepsilon_t \in (0,\, 1/2)$.

It can also be immediately shown that $\alpha_t = \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t}$ is indeed the value for which we reach the *minim* of the expression $\varepsilon_t \cdot e^{\alpha_t} + (1 - \varepsilon_t) \cdot e^{-\alpha_t}$, and therefore of Z_t too:

$$\varepsilon_t \cdot e^{\alpha_t} - (1 - \varepsilon_t) \cdot e^{-\alpha_t} > 0 \Leftrightarrow e^{2\alpha_t} - \frac{1 - \varepsilon_t}{\varepsilon_t} > 0 \Leftrightarrow \alpha_t > \frac{1}{2} \ln \frac{1 - \varepsilon_t}{\varepsilon_t} > 0.$$

e. Making use of the relationship (2) proven at part a, and using the fact that $1 - x \leq e^{-x}$ for all $x \in \mathbb{R}$, we can write:

$$\begin{aligned}\prod_{t=1}^{T} Z_t &= \prod_{t=1}^{T} 2 \cdot \sqrt{\varepsilon_t(1-\varepsilon_t)} = \prod_{t=1}^{T} 2 \cdot \sqrt{\left(\frac{1}{2} - \gamma_t\right)\left(1 - \left(\frac{1}{2} - \gamma_t\right)\right)} = \prod_{t=1}^{T} \sqrt{1 - 4\gamma_t^2} \\ &\leq \prod_{t=1}^{T} \sqrt{e^{-4\gamma_t^2}} = \prod_{t=1}^{T} \sqrt{(e^{-2\gamma_t^2})^2} = \prod_{t=1}^{T} e^{-2\gamma_t^2} = e^{-2\sum_{t=1}^{T} \gamma_t^2}.\end{aligned}$$

f. From the result obtained at parts c and d, we get:

$$err_S(H_T) \leq e^{-2\sum_{t=1}^{T} \gamma_t^2} \leq e^{-2T\gamma^2}$$

Therefore,

$$err_S(H_T) < \varepsilon \text{ if } -2T\gamma^2 < \ln \varepsilon \Leftrightarrow 2T\gamma^2 > -\ln \varepsilon \Leftrightarrow 2T\gamma^2 > \ln \frac{1}{\varepsilon} \Leftrightarrow T > \frac{1}{2\gamma^2} \ln \frac{1}{\varepsilon}$$

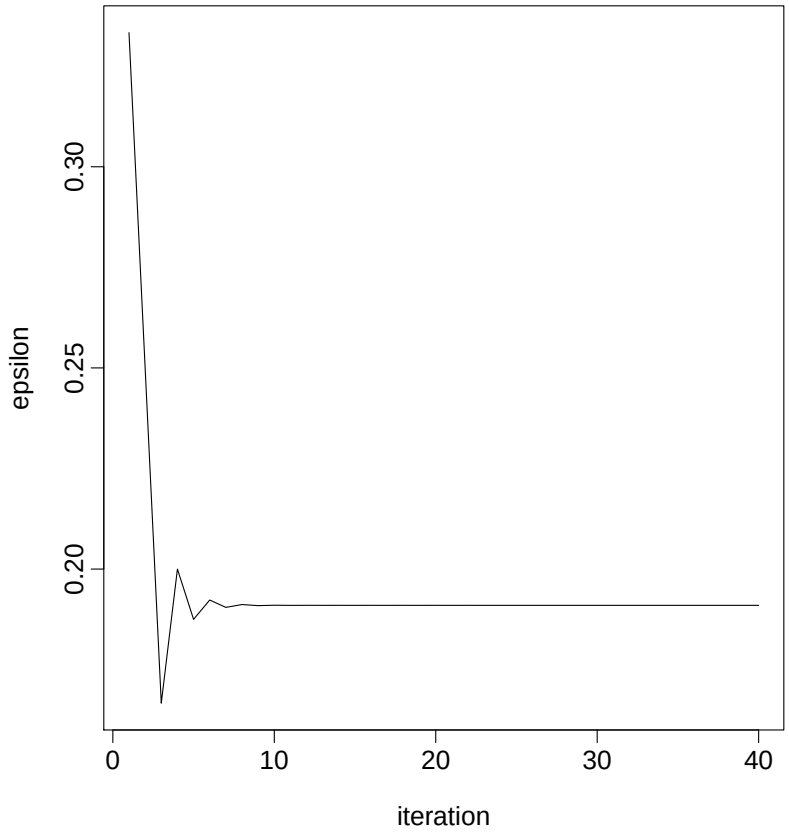
Hence we need $T = \mathcal{O}\left(\frac{1}{\gamma^2} \ln \frac{1}{\varepsilon}\right)$.

***Note*: It follows that $err_S(H_T) \to 0$ as $T \to \infty$.**

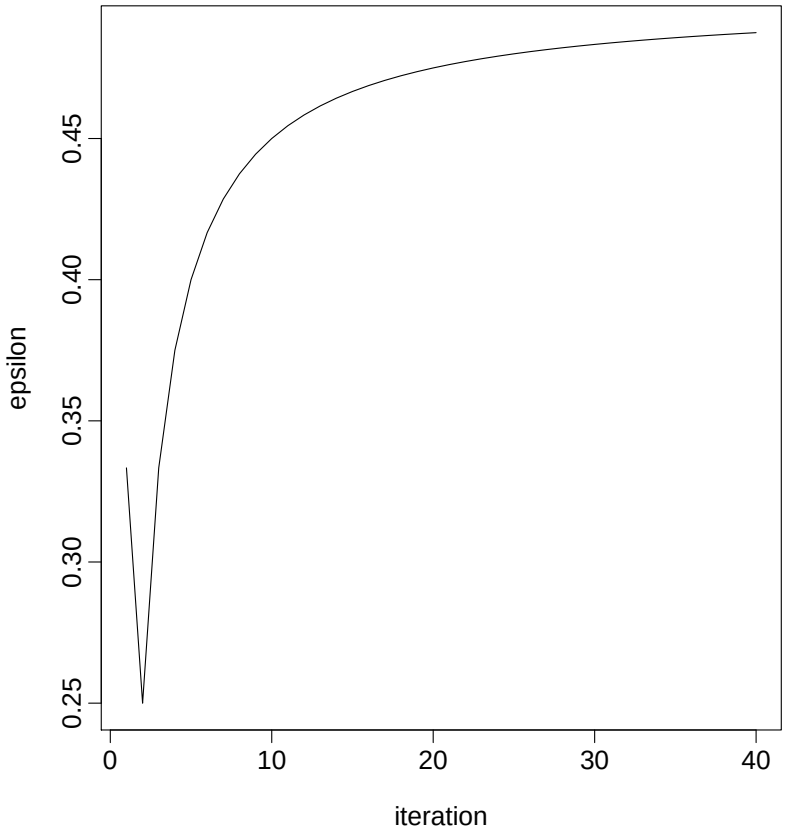
Exemplification:

Graphs made by MSc student Sebastian Ciobanu (2018 fall)
for CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, final, pr. 8.a-e

with outside thresold



without outside thresold

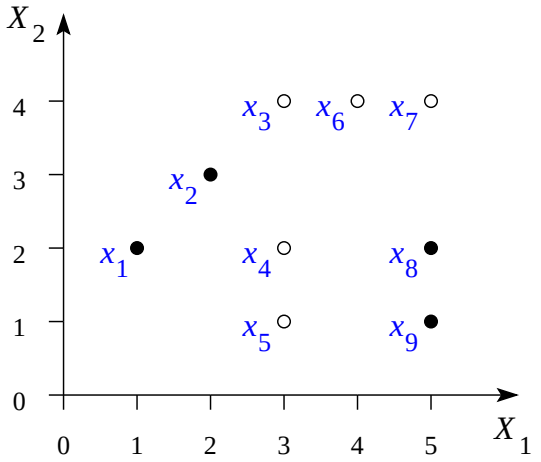


Exemplifying the application of AdaBoost algorithm

CMU, 2015 fall, Ziv Bar-Joseph, Eric Xing, HW4, pr. 2.6

Consider the training dataset in the nearby figure. Run $T = 3$ iterations of AdaBoost with decision stumps (axis-aligned separators) as the base learners. Illustrate the learned weak hypotheses h_t in this figure and fill in the table given below.

(For the pseudo-code of the AdaBoost algorithm, see CMU, 2015 fall, Ziv Bar-Joseph, Eric Xing, HW4, pr. 2.1-5. Please read the *Important Remark* that follows that pseudo-code!)



t	ε_t	α_t	$D_t(1)$	$D_t(2)$	$D_t(3)$	$D_t(4)$	$D_t(5)$	$D_t(6)$	$D_t(7)$	$D_t(8)$	$D_t(9)$	$err_S(H)$
1												
2												
3												

***Note*: The *goal* of this exercise is to help you understand how AdaBoost works in practice. It is advisable that — after understanding this exercise — you would implement a program / function that calculates the *weighted training error* produced by a given decision stump, w.r.t. a certain probabilistic distribution (D) defined on the training dataset. Later on you will extend this program to a full-fledged *implementation* of AdaBoost.**

Solution

Unlike the graphical reprezentation that we used until now for *decision stumps* (as trees of depth 1), here we will work with the following *analitical representation*: for a continuous attribute X taking values $x \in \mathbb{R}$ and for any threshold $s \in \mathbb{R}$, we can define two decision stumps:

$$sign(x-s) = \begin{cases} 1 & \text{if } x \geq s \\ -1 & \text{if } x < s \end{cases} \quad \text{and} \quad sign(s-x) = \begin{cases} -1 & \text{if } x \geq s \\ 1 & \text{if } x < s. \end{cases}$$

For convenience, in the sequel we will denote the first decision stump with $X \geq s$ and the second with $X < s$.

According to the *Important Remark* that follows the AdaBoost pseudo-code [see CMU, 2015 fall, Ziv Bar-Joseph, Eric Xing, HW4, pr. 2.1-5], at each iteration (t) the weak algorithm A selects the/a decision stump which, among all decision stumps, has the *minimum weighted training error* w.r.t. the current distribution (D_t) on the training data.

Notes

When applying the ID3 algorithm, for each continous attribute X, we used a threshold for each pair of examples (x_i, y_i), (x_{i+1}, y_{i+1}), with $y_i y_{i+1} < 0$ such that $x_i < x_{i+1}$, but no $x_j \in Val(X)$ for which $x_i < x_j < x_{i+1}$.

We will proceed similarly when applying AdaBoost with decision stumps and continous attributes.

[In the case of ID3 algorithm, there is a *theoretical result* stating that there is no need to consider other thresholds for a continuous attribute X apart from those situated beteen pairs of successive values $(x_i < x_{i+1})$ having opposite labels $(y_i \neq y_{i+1})$, because the Information Gain (IG) for the other thresholds $(x_i < x_{i+1}$, with $y_i = y_{i+1})$ is provably less than the maximal IG for X.

LC: A similar result can be proven, which allows us to simplify the application of the weak classifier (A) in the framework of the AdaBoost algorithm.]

Moreover, we will consider also a threshold from the outside of the interval of values taken by the attribute X in the training dataset. [The decision stumps corresponding to this "outside" threshold can be associated with the decision trees of depth 0 that we met in other problems.]

Iteration $t = 1$:

Therefore, at this stage (i.e, the first iteration of AdaBoost) the *thresholds* for the two continuous variables (X_1 and X_2) corresponding to the two coordinates of the training instances ($x_1, \ldots, x_9$) are

- $\frac{1}{2}$, $\frac{5}{2}$, **and** $\frac{9}{2}$ **for** X_1, **and**
- $\frac{1}{2}$, $\frac{3}{2}$, $\frac{5}{2}$ **and** $\frac{7}{2}$ **for** X_2.

One can easily see that we can get rid of the "outside" threshold $\frac{1}{2}$ **for** X_2, **because the decision stumps corresponding to this threshold act in the same as the decision stumps associated to the "outside" threshold** $\frac{1}{2}$ **for** X_1.

The *decision stumps* corresponding to this iteration together with their associated *weighted training errors* are shown on the next slide. When filling those tabels, we have used the equalities $err_{D_t}(X_1 \geq s) = 1 - err_{D_t}(X_1 < s)$ **and, similarly,** $err_{D_t}(X_2 \geq s) = 1 - err_{D_t}(X_2 < s)$, **for any threshold** s **and every iteration** $t = 1, 2, \ldots$. **These equalities are easy to prove.**

s	$\frac{1}{2}$	$\frac{5}{2}$	$\frac{9}{2}$
$err_{D_1}(X_1 < s)$	$\frac{4}{9}$	$\frac{2}{9}$	$\frac{4}{9} + \frac{2}{9} = \frac{2}{3}$
$err_{D_1}(X_1 \geq s)$	$\frac{5}{9}$	$\frac{7}{9}$	$\frac{1}{3}$

s	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{5}{2}$	$\frac{7}{2}$
$err_{D_1}(X_2 < s)$	$\frac{4}{9}$	$\frac{1}{9} + \frac{3}{9} = \frac{4}{9}$	$\frac{2}{9} + \frac{1}{9} = \frac{1}{3}$	$\frac{2}{9}$
$err_{D_1}(X_2 \geq s)$	$\frac{5}{9}$	$\frac{5}{9}$	$\frac{2}{3}$	$\frac{7}{9}$

It can be seen that the *minimal weighted training error* $(\varepsilon_1 = 2/9)$ **is obtained for the decision stumps** $X_1 < 5/2$ **and** $X_2 < 7/2$**. Therefore we can choose** $h_1 = sign\left(\frac{7}{2} - X_2\right)$ **as *best hypothesis* at iteration** $t = 1$**; the corresponding separator is the line** $X_2 = \frac{7}{2}$**. The** h_1 **hypothesis wrongly classifies the instances** x_4 **and** x_5**. Then**

$$\gamma_1 \quad = \quad \frac{1}{2} - \frac{2}{9} = \frac{5}{18} \textbf{ and } \alpha_1 = \frac{1}{2} \ln \frac{1 - \varepsilon_1}{\varepsilon_1} = \ln \sqrt{\left(1 - \frac{2}{9}\right) : \frac{2}{9}} = \ln \sqrt{\frac{7}{2}} \approx 0.626$$

Now the algorithm must get a new distribution (D_2) by altering the old one (D_1) so that the next iteration concentrates more on the misclassified instances.

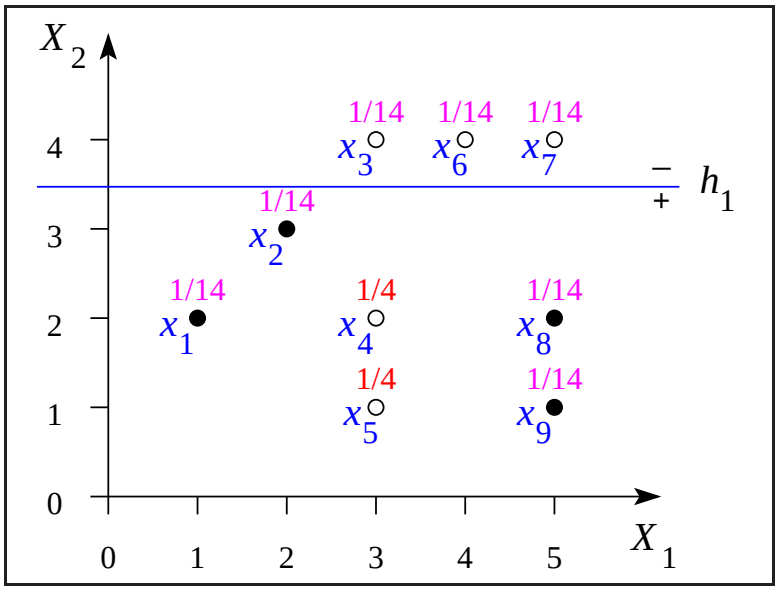
$$D_2(i) = \frac{1}{Z_1} D_1(i) \underbrace{(e^{-\alpha_1})}_{\sqrt{2/7}}{}^{y_i\, h_1(x_i)} = \begin{cases} \frac{1}{Z_1} \cdot \frac{1}{9} \cdot \sqrt{\frac{2}{7}} & \text{for } i \in \{1,2,3,6,7,8,9\}; \\ \frac{1}{Z_1} \cdot \frac{1}{9} \cdot \sqrt{\frac{7}{2}} & \text{for } i \in \{4,5\}. \end{cases}$$

Remember that Z_1 is a *normalization factor* for D_2. So,

$$Z_1 = \frac{1}{9}\left(7 \cdot \sqrt{\frac{2}{7}} + 2 \cdot \sqrt{\frac{7}{2}}\right) = \frac{2\sqrt{14}}{9} = 0.8315$$

Therefore,

$$D_2(i) = \begin{cases} \frac{9}{2\sqrt{14}} \cdot \frac{1}{9} \cdot \sqrt{\frac{2}{7}} = \frac{1}{14} & \text{for } i \notin \{4,5\}; \\ \frac{9}{2\sqrt{14}} \cdot \frac{1}{9} \cdot \sqrt{\frac{7}{2}} = \frac{1}{4} & \text{for } i \in \{4,5\}. \end{cases}$$



Note

If, instead of $sign\left(\frac{7}{2} - X_2\right)$ **we would have taken, as hypothesis** h_1**, the decision stump** $sign\left(\frac{5}{2} - X_1\right)$**, the subsequent calculus would have been slightly different (although both decision stumps have the same – minimal – weighted training error,** $\frac{2}{9}$**):** x_8 **and** x_9 **would have been allocated the weights** $\frac{1}{4}$**, while** x_4 **şi** x_5 **would have been allocated the weights** $\frac{1}{14}$**.**

(Therefore, the output of AdaBoost may not be uniquely determined!)

Iteration $t = 2$:

s	$\frac{1}{2}$	$\frac{5}{2}$	$\frac{9}{2}$
$err_{D_2}(X_1 < s)$	$\frac{4}{14}$	$\frac{2}{14}$	$\frac{2}{14} + \frac{2}{4} + \frac{2}{14} = \frac{11}{14}$
$err_{D_2}(X_1 \geq s)$	$\frac{10}{14}$	$\frac{12}{14}$	$\frac{3}{14}$

s	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{5}{2}$	$\frac{7}{2}$
$err_{D_2}(X_2 < s)$	$\frac{4}{14}$	$\frac{1}{4} + \frac{3}{14} = \frac{13}{28}$	$\frac{2}{4} + \frac{1}{14} = \frac{8}{14}$	$\frac{2}{4} = \frac{1}{2}$
$err_{D_2}(X_2 \geq s)$	$\frac{10}{14}$	$\frac{15}{28}$	$\frac{6}{14}$	$\frac{1}{2}$

***Note*: According to the theoretical result presented at part a of CMU, 2015 fall, Ziv Bar-Joseph, Eric Xing, HW4, pr. 2.1-5, computing the weighted error rate of the decision stump [corresponding to the test] $X_2 < 7/2$ is now superfluous, because this decision stump has been chosen as optimal hypothesis at the previous iteration. (Nevertheless, we had placed it into the tabel, for the sake of a thorough presentation.)**

Now the best hypothesis is $h_2 = sign\left(\frac{5}{2} - X_1\right)$; the corresponding separator is the line $X_1 = \frac{5}{2}$.

$$\varepsilon_2 = P_{D_2}(\{x_8, x_9\}) = \frac{2}{14} = \frac{1}{7} = 0.143 \qquad \Rightarrow \gamma_2 = \frac{1}{2} - \frac{1}{7} = \frac{5}{14}$$

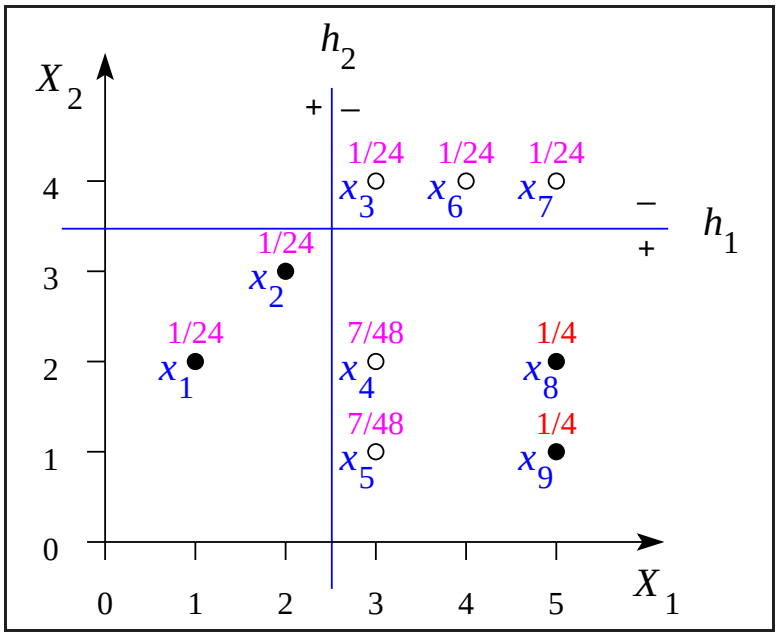
$$\alpha_2 = \ln\sqrt{\frac{1-\varepsilon_2}{\varepsilon_2}} = \ln\sqrt{\left(1 - \frac{1}{7}\right) : \frac{1}{7}} = \ln\sqrt{6} = 0.896$$

$$D_3(i) = \frac{1}{Z_2} \cdot D_2(i) \cdot (\underbrace{e^{-\alpha_2}}_{1/\sqrt{6}})^{y_i\, h_2(x_i)} = \begin{cases} \frac{1}{Z_2} \cdot D_2(i) \cdot \frac{1}{\sqrt{6}} & \text{if } h_2(x_i) = y_i; \\ \frac{1}{Z_2} \cdot D_2(i) \cdot \sqrt{6} & \text{otherwise} \end{cases}$$

$$= \begin{cases} \frac{1}{Z_2} \cdot \frac{1}{14} \cdot \frac{1}{\sqrt{6}} & \text{for } i \in \{1,2,3,6,7\}; \\ \frac{1}{Z_2} \cdot \frac{1}{4} \cdot \frac{1}{\sqrt{6}} & \text{for } i \in \{4,5\}; \\ \frac{1}{Z_2} \cdot \frac{1}{14} \cdot \sqrt{6} & \text{for } i \in \{8,9\}. \end{cases}$$

$$Z_2 = 5 \cdot \frac{1}{14} \cdot \frac{1}{\sqrt{6}} + 2 \cdot \frac{1}{4} \cdot \frac{1}{\sqrt{6}} + 2 \cdot \frac{1}{14} \cdot \sqrt{6} = \frac{5}{14\sqrt{6}} + \frac{1}{2\sqrt{6}} + \frac{\sqrt{6}}{7} = \frac{12+12}{14\sqrt{6}} = \frac{24}{14\sqrt{6}} = \frac{2\sqrt{6}}{7} \approx 0.7$$

$$D_3(i) = \begin{cases} \frac{7}{2\sqrt{6}} \cdot \frac{1}{14} \cdot \frac{1}{\sqrt{6}} = \frac{1}{24} & \text{for } i \in \{1,2,3,6,7\}; \\ \frac{7}{2\sqrt{6}} \cdot \frac{1}{4} \cdot \frac{1}{\sqrt{6}} = \frac{7}{48} & \text{for } i \in \{4,5\}; \\ \frac{7}{2\sqrt{6}} \cdot \frac{1}{14} \cdot \sqrt{6} = \frac{1}{4} & \text{for } i \in \{8,9\}. \end{cases}$$



Iteration $t = 3$:

s	$\frac{1}{2}$	$\frac{5}{2}$	$\frac{9}{2}$
$err_{D_3}(X_1 < s)$	$\frac{2}{24} + \frac{2}{4} = \frac{7}{12}$	$\frac{2}{4}$	$\frac{2}{24} + 2 \cdot \frac{7}{48} + 2 \cdot \frac{1}{4} = \frac{21}{24}$
$err_{D_3}(X_1 \geq s)$	$\frac{5}{12}$	$\frac{2}{4}$	$\frac{3}{24} = \frac{1}{8}$

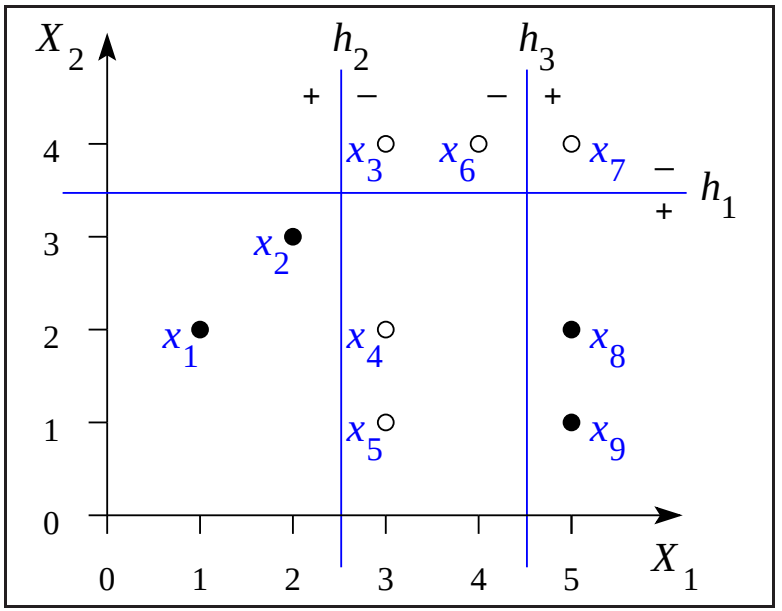
s	$\frac{1}{2}$	$\frac{3}{2}$	$\frac{5}{2}$	$\frac{7}{2}$
$err_{D_3}(X_1 < s)$	$\frac{7}{12}$	$\frac{7}{48} + \frac{2}{24} + \frac{1}{4} = \frac{23}{48}$	$2 \cdot \frac{7}{48} + \frac{1}{24} = \frac{1}{3}$	$2 \cdot \frac{7}{48} = \frac{7}{24}$
$err_{D_3}(X_1 \geq s)$	$\frac{5}{12}$	$\frac{25}{48}$	$\frac{2}{3}$	$\frac{17}{24}$

The new best hypothesis is $h_3 = sign\left(X_1 - \frac{9}{2}\right)$; the corresponding separator is the line $X_1 = \frac{9}{2}$.

$$\varepsilon_3 = P_{D_3}(\{x_1, x_2, x_7\}) = 2 \cdot \frac{1}{24} + \frac{1}{24} = \frac{3}{24} = \frac{1}{8}$$

$$\gamma_3 = \frac{1}{2} - \frac{1}{8} = \frac{3}{8}$$

$$\alpha_3 = \ln \sqrt{\frac{1-\varepsilon_3}{\varepsilon_3}} = \ln \sqrt{\left(1 - \frac{1}{8}\right) : \frac{1}{8}} = \ln \sqrt{7} = 0.973$$



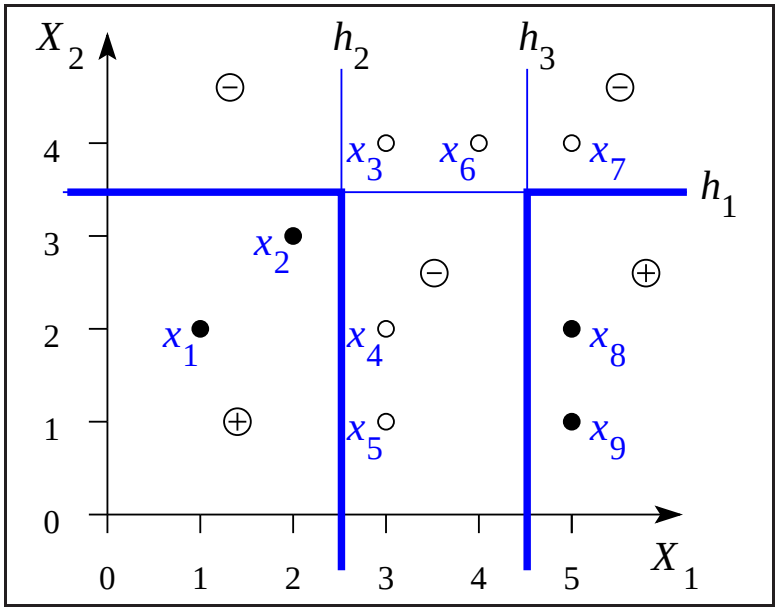
Finally, after filling our results in the given table, we get:

t	ε_t	α_t	$D_t(1)$	$D_t(2)$	$D_t(3)$	$D_t(4)$	$D_t(5)$	$D_t(6)$	$D_t(7)$	$D_t(8)$	$D_t(9)$	$err_S(H)$
1	2/9	$\ln\sqrt{7/2}$	1/9	1/9	1/9	1/9	1/9	1/9	1/9	1/9	1/9	2/9
2	2/14	$\ln\sqrt{6}$	1/14	1/14	1/14	1/4	1/4	1/14	1/14	1/14	1/14	2/9
3	1/8	$\ln\sqrt{7}$	1/24	1/24	1/24	7/48	7/48	1/24	1/24	1/4	1/4	0

***Note*: The following table helps you understand how $err_S(H)$ was computed; remember that $H(x_i) \stackrel{def.}{=} sign\left(\sum_{t=1}^{T} \alpha_t\, h_t(x_i)\right)$.**

t	α_t	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9
1	0.626	+1	+1	−1	+1	+1	−1	−1	+1	+1
2	0.896	+1	+1	−1	−1	−1	−1	−1	−1	−1
3	0.973	−1	−1	−1	−1	−1	−1	+1	+1	+1
	$H(x_i)$	+1	+1	−1	−1	−1	−1	−1	+1	+1

***Remark*: One can immediately see that the [test] instance $(1, 4)$ will be classified by the hypothesis H learned by AdaBoost as negative (since $-\alpha_1 + \alpha_2 - \alpha_3 = -0.626 + 0.896 - 0.973 < 0$). After making other similar calculi, we can conclude that the *decision zones* and the *decision boundaries* produced by AdaBoost for the given training data will be as indicated in the nearby figure.**

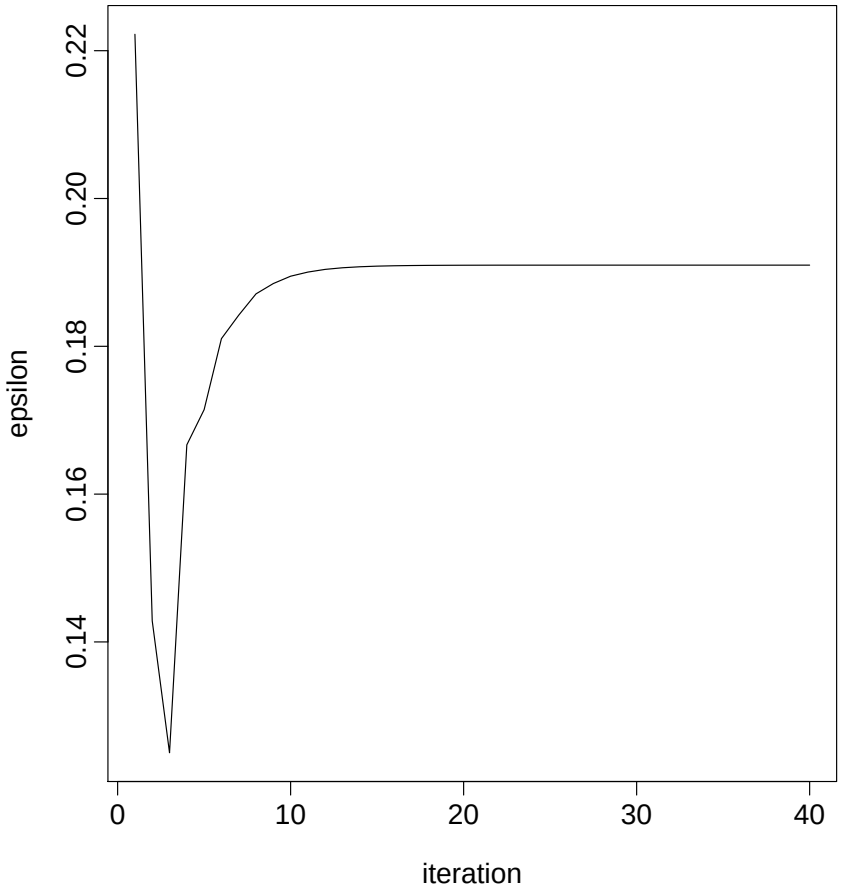


***Remark*: The execution of AdaBoost could continue (if we would have taken initially $T > 3$...), although we have obtained $err_S(H) = 0$ at iteration $t = 3$. By elaborating the details, we would see that for $t = 4$ we would obtain as optimal hypothesis $X_2 < 7/2$ (which has already been selected at iteration $t = 1$). This hypothesis produces now the weighted training error $\varepsilon_4 = 1/6$. Therefore, $\alpha_4 = \ln\sqrt{5}$, and this will be added to $\alpha_1 = \ln\sqrt{7/2}$ in the new output H. In this way, the confidence in the hypothesis $X_2 < 7/2$ would be strenghened.**

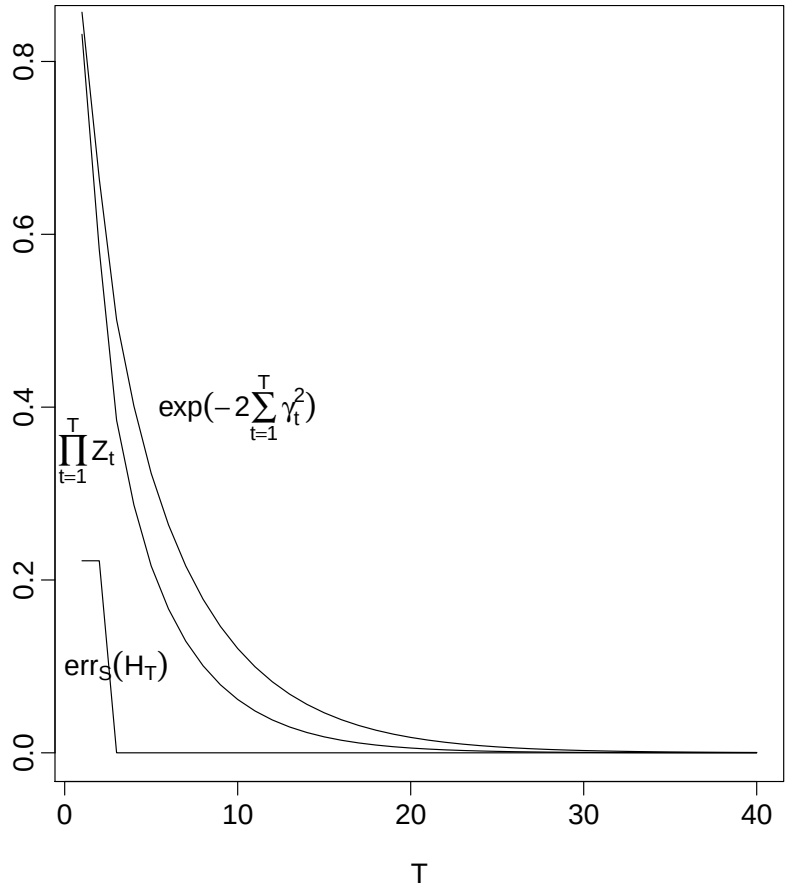
So, we should keep in mind that AdaBoost can select several times a certain weak hypothesis (but never at consecutive iterations, cf. CMU, 2015 fall, E. Xing, Z. Bar-Joseph, HW4, pr. 2.1).

Graphs made by MSc student Sebastian Ciobanu (2018 fall)

The variation of ε_t w.r.t. t:



The two upper bounds of the empirical error of H_T:



Seeing AdaBoost as an *optimization algorithm*, w.r.t. the [inverse] exponential *loss function*

CMU, 2008 fall, Eric Xing, HW3, pr. 4.1.1

CMU, 2008 fall, Eric Xing, midterm, pr. 5.1

At CMU, 2015 fall, Ziv Bar-Joseph, Eric Xing, HW4, pr. 2.1-5, part d, we have shown that in AdaBoost, we try to [indirectly] minimize the training error $err_S(H)$ by sequentially minimizing its upper bound $\prod_{t=1}^{T} Z_t$, i.e. at each iteration t $(1 \leq t \leq T)$ we choose α_t so as to minimize Z_t (viwed as a function of α_t).

Here you will see that another way to explain AdaBoost is by sequentially minimizing the *negative exponential loss*:

$$E \stackrel{def.}{=} \sum_{i=1}^{m} \exp(-y_i f_T(x_i)) \stackrel{not.}{=} \sum_{i=1}^{m} \exp(-y_i \sum_{t=1}^{T} \alpha_t h_t(x_i)). \tag{3}$$

That is to say, at the t-th iteration $(1 \leq t \leq T)$ we want to choose besides the appropriate classifier h_t the corresponding weight α_t so that the overall loss E (accumulated up to the t-th iteration) is minimized.

Prove that this [new] strategy will lead to the same update rule for α_t used in AdaBoost, i.e., $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$.

***Hint*: You can use the fact that $D_t(i) \propto \exp(-y_i f_{t-1}(x_i))$, and it [LC: the proportionality factor] can be viewed as constant when we try to optimize E with respect to α_t in the t-th iteration.**

Solution

At the t-th iteration, we have

$$
\begin{aligned}
E &= \sum_{i=1}^{m} \exp(-y_i f_t(x_i)) = \sum_{i=1}^{m} \exp\left(-y_i \left(\sum_{t'=1}^{t-1} \alpha_{t'} h_{t'}(x_i)\right) - y_i \alpha_t h_t(x_i)\right) \\
&= \sum_{i=1}^{m} \exp(-y_i f_{t-1}(x_i)) \cdot \exp(-y_i \alpha_t h_t(x_i)) = \sum_{i=1}^{m} \left(m \prod_{i=1}^{t-1} Z_t\right) \cdot D_t(i) \cdot \exp(-y_i \alpha_t h_t(x_i)) \\
&\propto \sum_{i=1}^{m} D_t(i) \cdot \exp(-y_i \alpha_t h_t(x_i)) \overset{not.}{=} E' \text{ (see CMU, 2015 fall, Z. Bar-Joseph, E. Xing, HW4, pr. 2.1-5, part } b\text{)}
\end{aligned}
$$

Further on, we can rewrite E' as

$$
\begin{aligned}
E' &= \sum_{i=1}^{m} D_t(i) \cdot \exp(-y_i \alpha_t h_t(x_i)) = \sum_{i \in C} D_t(i) \exp(-\alpha_t) + \sum_{i \in M} D_t(i) \exp(\alpha_t) \\
&= (1 - \varepsilon_t) \cdot e^{-\alpha_t} + \varepsilon_t \cdot e^{\alpha_t}, \qquad (4)
\end{aligned}
$$

where C is the set of examples which are correctly classified by h_t, and M is the set of examples which are mis-classified by h_t.

The relation (4) **is identical with the expression** (1) **from part** a **of CMU, 2015 fall, Ziv Bar-Joseph, Eric Xing, HW4, pr. 2.1-5 (see the solution). Therefore,** E **will reach its minim for** $\alpha_t = \frac{1}{2} \ln \frac{1-\varepsilon_t}{\varepsilon_t}$**.**

AdaBoost algorithm: the notion of [*voting*] *margin*; some properties

CMU, 2016 spring, W. Cohen, N. Balcan, HW4, pr. 3.3

Despite that *model complexity* increases with each iteration, *AdaBoost does not usually overfit*. The reason behind this is that the model becomes more "confident" as we increase the number of iterations. The "confidence" can be expressed mathematically as the [*voting*] *margin*. Recall that after the algorithm AdaBoost terminates with T iterations the [output] classifier is

$$H_T(x) = sign\left(\sum_{t=1}^{T} \alpha_t \, h_t(x)\right).$$

Similarly, we can define the *intermediate weighted classifier* after k iterations as:

$$H_k(x) = sign\left(\sum_{t=1}^{k} \alpha_t \, h_t(x)\right).$$

As its output is either -1 or 1, it does not tell the confidence of its judgement. Here, without changing the decision rule, let

$$H_k(x) = sign\left(\sum_{t=1}^{k} \bar{\alpha}_t \, h_t(x)\right),$$

where $\bar{\alpha}_t = \dfrac{\alpha_t}{\sum_{t'=1}^{k} \alpha_{t'}}$ so that the weights on each weak classifier are normalized.

Define the *margin* after the k-th iteration as [the sum of] the [normalized] weights of h_t voting correctly minus [the sum of] the [normalized] weights of h_t voting incorrectly.

$$Margin_k(x) = \sum_{t:h_t(x)=y} \bar{\alpha}_t - \sum_{t:h_t(x)\neq y} \bar{\alpha}_t.$$

a. Let $f_k(x) \stackrel{not.}{=} \sum_{t=1}^{k} \bar{\alpha}_t \, h_t(x)$. Show that $Margin_k(x_i) = y_i \, f_k(x_i)$ for all training instances x_i, with $i = 1, \ldots, m$.

b. If $Margin_k(x_i) > Margin_k(x_j)$, which of the samples x_i and x_j will receive a higher weight in iteration $k+1$?

***Hint*: Use the relation $D_{k+1}(i) = \dfrac{1}{m \cdot \prod_{t=1}^{k} Z_t} \cdot \exp(-y_i f_k(x_i))$ which was proven at CMU, 2015 fall, Z. Bar-Joseph, E. Xing, HW4, pr. 2.2.**

Solution

a. We will prove the equality starting from its right hand side:

$$\begin{aligned} y_i\, f_k(x_i) &= y_i \sum_{t=1}^{k} \bar{\alpha}_t\, h_t(x_i) = \sum_{t=1}^{k} \bar{\alpha}_t\, y_i\, h_t(x_i) = \sum_{t:h_t(x_i)=y_i} \bar{\alpha}_t - \sum_{t:h_t(x_i)\neq y_i} \bar{\alpha}_t \\ &= Margin_k(x_i). \end{aligned}$$

b. According to the relationship already proven at part a,

$$Margin_k(x_i) > Margin_k(x_j) \Leftrightarrow y_i\, f_k(x_i) > y_j\, f_k(x_j) \Leftrightarrow$$
$$-y_i\, f_k(x_i) < -y_j\, f_k(x_j) \Leftrightarrow \exp(-y_i\, f_k(x_i)) < \exp(-y_j\, f_k(x_j)).$$

Based on the given *Hint*, it follows that $D_{k+1}(i) < D_{k+1}(j)$.

Important Remark

It can be shown that boosting tends to increase the margins of training examples — see the relation (3) at CMU, 2008 fall, Eric Xing, HW3, pr. 4.1.1 —, and that a large margin on training examples reduces the generalization error.

Thus we can explain why, although the number of "parameters" of the model created by AdaBoost increases with 2 at every iteration — therefore *complexity* rises —, it usually doesn't overfit.

AdaBoost: a sufficient condition for γ-weak learnability, based on the voting margins

CMU, 2016 spring, W. Cohen, N. Balcan, HW4, pr. 3.1.4

At CMU, 2015 fall, Z. Bar-Joseph, E. Xing, HW4, pr. 2.5 we encountered the notion of empirical γ-weak learnability. When this condition — $\gamma \leq \gamma_t$ for all t, where $\gamma_t \stackrel{def.}{=} \frac{1}{2} - \varepsilon_t$, with ε_t being the weighted training error produced by the weak hypothesis h_t — is met, it ensures that AdaBoost will drive down the training error quickly. However, this condition does not hold all the time.

In this problem we will prove a *sufficient condition* for empirical weak learnability [to hold]. This condition refers to the notion of *voting margin* which was presented in CMU, 2016 spring, W. Cohen, N. Balcan, HW4, pr. 3.3.

Namely, we will prove that
if there is a constant $\theta > 0$ such that the [voting] margins of all training instances are lower-bounded by θ at each iteration of the AdaBoost algorithm, then the property of empirical γ-weak learnability is "guaranteed", with $\gamma = \theta/2$.

[Formalisation]

Suppose we are given a training set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, such that for some weak hypotheses $h_1, \ldots, h_k$ from the hypothesis space $\mathcal{H}$, and some non-negative coefficients $\alpha_1, \ldots, \alpha_k$ with $\sum_{j=1}^{k} \alpha_j = 1$, there exists $\theta > 0$ such that

$$y_i(\sum_{j=1}^{k} \alpha_j h_j(x_i)) \geq \theta, \ \forall (x_i, y_i) \in S.$$

***Note*: according to CMU, 2016 spring, W. Cohen, N. Balcan, HW4, pr. 3.3,**

$$y_i(\sum_{j=1}^{k} \alpha_j h_j(x_i)) = Margin_k(x_i) = f_k(x_i), \ \textbf{where} \ f_k(x_i) \overset{not.}{=} \sum_{j=1}^{k} \alpha_j h_j(x_i).$$

Key idea: We will show that if the condition above is satisfied (for a given k), then *for any* distribution D over S, there exists a hypothesis $h_l \in \{h_1, \ldots, h_k\}$ with weighted training error at most $\frac{1}{2} - \frac{\theta}{2}$ over the distribution D.

It will follow that when the condition above is satisfied for any k, the training set S is empirically γ-weak learnable, with $\gamma = \frac{\theta}{2}$.

a. Show that — if the condition stated above is met — there exists a weak hypothesis h_l from $\{h_1, \ldots, h_k\}$ such that $E_{i\sim D}[y_i h_l(x_i)] \geq \theta$.

***Hint*: Taking expectation under the same distribution does not change the inequality conditions.**

b. Show that the inequality $E_{i\sim D}[y_i h_l(x_i)] \geq \theta$ is equivalent to

$$\underbrace{\Pr_{i\sim D}[y_i \neq h_l(x_i)]}_{err_D(h_l)} \leq \frac{1}{2} - \frac{\theta}{2},$$

meaning that the weighted training error of h_l is at most $\frac{1}{2} - \frac{\theta}{2}$, and therefore $\gamma_t \geq \frac{\theta}{2}$.

Solution

a. Since $y_i(\sum_{j=1}^{k} \alpha_j h_j(x_i)) \geq \theta \Leftrightarrow y_i f_k(x_i) \geq \theta$ **for** $i = 1, \ldots, m$**, it follows (according to the *Hint*) that**

$$E_{i\sim D}[y_i f_k(x_i)] \geq \theta \text{ where } f_k(x_i) \stackrel{not.}{=} \sum_{j=1}^{k} \alpha_j h_j(x_i). \tag{5}$$

On the other side, $E_{i\sim D}[y_i h_l(x_i)] \geq \theta \stackrel{def.}{\Leftrightarrow} \sum_{i=1}^{m} y_i h_l(x_i) \cdot D(i) \geq \theta$**.**

Suppose, on contrary, that $E_{i\sim D}[y_i h_l(x_i)] < \theta$**, that is** $\sum_{i=1}^{m} y_i h_l(x_i) \cdot D(i) < \theta$ **for** $l = 1, \ldots, k$**. Then** $\sum_{i=1}^{m} y_i h_l(x_i) \cdot D(i) \cdot \alpha_l < \theta \cdot \alpha_l$ **for** $l = 1, \ldots, k$**. By summing up these inequations for** $l = 1, \ldots, k$ **we get**

$$\sum_{l=1}^{k}\sum_{i=1}^{m} y_i h_l(x_i) \cdot D(i) \cdot \alpha_l < \sum_{l=1}^{k} \theta \cdot \alpha_l \Leftrightarrow \sum_{i=1}^{m} y_i\, D(i) \left(\sum_{l=1}^{k} h_l(x_i)\alpha_l\right) < \theta \sum_{l=1}^{k} \alpha_l \Leftrightarrow$$

$$\sum_{i=1}^{m} y_i f_k(x_i) \cdot D(i) < \theta, \tag{6}$$

because $\sum_{j=1}^{k} \alpha_j = 1$ **şi** $f_k(x_i) \stackrel{not.}{=} \sum_{l=1}^{k} \alpha_l h_l(x_i)$**.**

The inequation (6) can be written as $E_{i\sim D}[y_i f_k(x_i)] < \theta$**. Obviously, it contradicts the relationship (5). Therefore, the previous supposition is false. In conclusion, there exist** $l \in \{1, \ldots, k\}$ **such that** $E_{i\sim D}[y_i h_l(x_i)] \geq \theta$**.**

Solution (cont'd)

b. We already said that $E_{i\sim D}[y_i h_l(x_i)] \geq \theta \Leftrightarrow \sum_{i=1}^{m} y_i\, h_l(x_i) \cdot D(i) \geq \theta$**.**
Since $y_i \in \{-1,+1\}$ **and** $h_l(x_i) \in \{-1,+1\}$ **for** $i = 1,\ldots,m$ **and** $l = 1,\ldots,k$**, we have**

$$\sum_{i=1}^{m} y_i\, h_l(x_i) \cdot D(i) \geq \theta \;\; \Leftrightarrow \sum_{i:y_i=h_l(x_i)} D(x_i) - \sum_{i:y_i\neq h_l(x_i)} D(x_i) \geq \theta \Leftrightarrow (1-\varepsilon_l) - \varepsilon_l \geq \theta \Leftrightarrow$$

$$1 - 2\varepsilon_l \geq \theta \Leftrightarrow 2\varepsilon_l \leq 1-\theta \Leftrightarrow \varepsilon_l \leq \frac{1}{2} - \frac{\theta}{2} \overset{def.}{\Leftrightarrow} err_D(h_l) \leq \frac{1}{2} - \frac{\theta}{2}.$$

AdaBoost:
Any set of consistently labelled instances from $\mathbb{R}$ is empirically γ-weak learnable by using decision stumps

Stanford, 2016 fall, Andrew Ng, John Duchi, HW2, pr. 6.abc

At CMU, 2015, Z. Bar-Joseph, E. Xing, HW4, pr. 2.5 we encountered the notion of empirical γ-weak learnability. When this condition — $\gamma \leq \gamma_t$ for all t, where $\gamma_t \stackrel{def.}{=} \frac{1}{2} - \varepsilon_t$, with ε_t being the weighted training error produced by the weak hypothesis h_t — is met, it ensures that AdaBoost will drive down the training error quickly.

In this problem we will assume that our input attribute vectors $x \in \mathbb{R}$, that is, they are one-dimensional, and we will show that [LC] when these vectors are consitently labelled, decision stumps based on thresholding provide a weak-learning *guarantee* (γ).

Decision stumps: analytical definitions / formalization

Thresholding-based decision stumps can be seen as functions indexed by a threshold s and sign $+/-$, such that

$$\phi_{s,+}(x) = \begin{cases} 1 & \text{if } x \geq s \\ -1 & \text{if } x < s \end{cases} \quad \text{and } \phi_{s,-}(x) = \begin{cases} -1 & \text{if } x \geq s \\ 1 & \text{if } x < s. \end{cases}$$

Therefore, $\phi_{s,+}(x) = -\phi_{s,-}(x)$**.**

Key idea for the proof

We will show that given a consistently labelled training set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$**, with** $x_i \in \mathbb{R}$ **and** $y_i \in \{-1, +1\}$ **for** $i = 1, \ldots, m$**, there is some** $\gamma > 0$ **such that for any distribution** p **defined on this training set there is a threshold** $s \in \mathbb{R}$ **for which**

$$error_p(\phi_{s,+}) \leq \frac{1}{2} - \gamma \quad \textbf{or} \quad error_p(\phi_{s,-}) \leq \frac{1}{2} - \gamma,$$

where $error_p(\phi_{s,+})$ **and** $error_p(\phi_{s,-})$ **denote the weighted training error of** $\phi_{s,+}$ **and respectively** $\phi_{s,-}$**, computed according to the distribution** p**.**

Convention: In our problem we will assume that our training instances $x_1, \ldots x_m \in \mathbb{R}$ are distinct. Moreover, we will assume (without loss of generality, but this makes the proof notationally simpler) that

$$x_1 > x_2 > \ldots > x_m.$$

a. Show that, given S, for each threshold $s \in \mathbb{R}$ there is some $m_0(s) \in \{0, 1, \ldots, m\}$ such that

$$error_p(\phi_{s,+}) \stackrel{def.}{=} \sum_{i=1}^{m} p_i \cdot 1_{\{y_i \neq \phi_{s,+}(x_i)\}} = \frac{1}{2} - \frac{1}{2} \underbrace{\left(\sum_{i=1}^{m_0(s)} y_i p_i \; - \sum_{i=m_0(s)+1}^{m} y_i p_i \right)}_{\text{not.: } f(m_0(s))}$$

and

$$error_p(\phi_{s,-}) \stackrel{def.}{=} \sum_{i=1}^{m} p_i \cdot 1_{\{y_i \neq \phi_{s,-}(x_i)\}} = \frac{1}{2} - \frac{1}{2} \underbrace{\left(\sum_{i=m_0(s)+1}^{m} y_i p_i \; - \sum_{i=1}^{m_0(s)} y_i p_i \right)}_{\text{not.: } -f(m_0(s))}$$

***Note*: Treat sums over empty sets of indices as zero. Therefore, $\sum_{i=1}^{0} a_i = 0$ for any a_i, and similarly $\sum_{i=m+1}^{m} a_i = 0$.**

b. Prove that, given S, there is some $\gamma > 0$ (which may depend on the training set size m) such that for any set of probabilities p on the training set (therefore $p_i \geq 0$ and $\sum_{i=1}^{m} p_i = 1$) we can find $m_0 \in \{0, \ldots, m\}$ so as

$$|f(m_0)| \geq 2\gamma, \text{ where } f(m_0) \overset{not.}{=} \sum_{i=1}^{m_0} y_i p_i - \sum_{i=m_0(s)+1}^{m} y_i p_i,$$

***Note*: γ should not depend on p.**

***Hint*: Consider the difference $f(m_0) - f(m_0 - 1)$.**

What is your γ?

c. Based on your answers to parts a and b, what *edge* can thresholded decision stumps guarantee on any training set $\{x_i, y_i\}_{i=1}^m$, where the raw attributes $x_i \in \mathbb{R}$ are all distinct? Recall that the edge of a weak classifier $\phi : \mathbb{R} \to \{-1, 1\}$ is the constant $\gamma \in (0, 1/2)$ such that

$$error_p(\phi) \stackrel{def.}{=} \sum_{i=1}^{m} p_i \cdot 1_{\{\phi(x_i) \neq y_i\}} \leq \frac{1}{2} - \gamma.$$

d. Can you give an upper bound on the number of thresholded decision stumps required to achieve zero error on a given training set?

Solution

a. We perform several algebraic steps.
Let $sign(t) = 1$ **if** $t \geq 0$**, and** $sign(t) = -1$ **otherwise.**
Then

$$1_{\{\phi_{s,+}(x) \neq y\}} = 1_{\{sign(x-s) \neq y\}} = 1_{\{y \cdot sign(x-s) \leq 0\}},$$

where the symbol $1_{\{\ \}}$ **denotes the well known *indicator function*.**
Thus we have

$$\begin{aligned} error_p(\phi_{s,+}) &\overset{def.}{=} \sum_{i=1}^{m} p_i \cdot 1_{\{y_i \neq \phi_{s,+}(x_i)\}} = \sum_{i=1}^{m} p_i \cdot 1_{\{y_i \cdot sign(x_i - s) \leq 0\}} \\ &= \sum_{i: x_i \geq s} p_i \cdot 1_{\{y_i = -1\}} + \sum_{i: x_i < s} p_i \cdot 1_{\{y_i = 1\}} \end{aligned}$$

Thus, if we let $m_0(s)$ **be the index in** $\{0, \ldots, m\}$ **such that** $x_i \geq s$ **for** $i \leq m_0(s)$ **and** $x_i < s$ **for** $i > m_0(s)$**, which we know must exist because** $x_1 > x_2 > \ldots$ **, we have**

$$error_p(\phi_{s,+}) \overset{def.}{=} \sum_{i=1}^{m} p_i \cdot 1_{\{y_i \neq \phi_{s,+}(x_i)\}} = \sum_{i=1}^{m_0(s)} p_i \cdot 1_{\{y_i = -1\}} + \sum_{i=m_0(s)+1}^{m} p_i \cdot 1_{\{y_i = 1\}}.$$

Now we make a *key observation*:
we have

$$1_{\{y=-1\}} = \frac{1-y}{2} \text{ and } 1_{\{y=1\}} = \frac{1+y}{2},$$

because $y \in \{-1, 1\}$.

Consequently,

$$\begin{aligned} error_p(\phi_{s,+}) &\stackrel{def.}{=} \sum_{i=1}^{m} p_i \cdot 1_{\{y_i \neq \phi_{s,+}(x_i)\}} = \sum_{i=1}^{m_0(s)} p_i \cdot 1_{\{y_i=-1\}} + \sum_{i=m_0(s)+1}^{m} p_i \cdot 1_{\{y_i=1\}} \\ &= \sum_{i=1}^{m_0(s)} p_i \cdot \frac{1-y_i}{2} + \sum_{i=m_0(s)+1}^{m} p_i \cdot \frac{1+y_i}{2} = \frac{1}{2}\sum_{i=1}^{m} p_i - \frac{1}{2}\sum_{i=1}^{m_0(s)} p_i y_i + \frac{1}{2}\sum_{i=m_0(s)+1}^{m} p_i y_i \\ &= \frac{1}{2} - \frac{1}{2}\left(\sum_{i=1}^{m_0(s)} p_i y_i - \sum_{i=m_0(s)+1}^{m} p_i y_i\right). \end{aligned}$$

The last equality follows because $\sum_{i=1}^{m} p_i = 1$.

The case for $\phi_{s,-}$ is symmetric to this one, so we omit the argument.

Solution (cont'd)

b. For any $m_0 \in \{1, \ldots, m\}$ we have

$$f(m_0) - f(m_0 - 1) = \sum_{i=1}^{m_0} y_i p_i \; - \sum_{i=m_0+1}^{m} y_i p_i - \sum_{i=1}^{m_0-1} y_i p_i \; + \sum_{i=m_0}^{m} y_i p_i = 2 y_{m_0} p_{m_0}.$$

Therefore, $|f(m_0) - f(m_0 - 1)| = 2|y_{m_0}|\, p_{m_0} = 2p_{m_0}$ for all $m_0 \in \{1, \ldots, m\}$.
Because $\sum_{i=1}^{m} p_i = 1$, there must be at least one index m_0' with $p_{m_0'} \geq \frac{1}{m}$.
Thus we have $|f(m_0') - f(m_0' - 1)| \geq \frac{2}{m}$, and so it must be the case that at least one of

$$|f(m_0')| \geq \frac{1}{m} \quad \textbf{or} \quad |f(m_0' - 1)| \geq \frac{1}{m}$$

holds.
Depending on which one of those two inequations is true, we would then "return" m_0' or $m_0' - 1$.
(*Note*: If $|f(m_0' - 1)| \geq \frac{1}{m}$ and $m_0' = 1$, then we have to consider an "outside" threshold, $s > x_1$.)
Finally, we have $\gamma = \frac{1}{2m}$.

Solution (cont'd)

c. The inequation proven at part b, $|f(m_0)| \geq 2\gamma$ implies that either $f(m_0) \geq 2\gamma$ or $f(m_0) \leq -2\gamma$. So,

$$\textbf{either } f(m_0) \geq 2\gamma \Leftrightarrow -f(m_0) \leq -2\gamma \Leftrightarrow \underbrace{\frac{1}{2} - \frac{1}{2} f(m_0)}_{error_p(\phi_{s,+})} \leq \frac{1}{2} - \frac{1}{2} \cdot 2\gamma = \frac{1}{2} - \gamma$$

$$\textbf{or } f(m_0) \leq -2\gamma \Leftrightarrow \underbrace{\frac{1}{2} + \frac{1}{2} f(m_0)}_{error_p(\phi_{s,-})} \leq \frac{1}{2} - \frac{1}{2} \cdot 2\gamma = \frac{1}{2} - \gamma$$

for any $s \in (x_{m_0+1}, x_{m_0}]$,[a]

Therefore thresholded decision stumps are *guaranteed* to have an *edge* of at least $\gamma = \dfrac{1}{2m}$ over random guessing.

[a] In the case described by the *Note* at part b, we must consider $s > x_1$.

Summing up

At each iteration t executed by AdaBoost,

- **a probabilistic distribution p (denoted as D_t in CMU, 2015, Z. Bar-Joseph, E. Xing, HW4, pr. 2.1-5) is in use;**
- **[at part b of the present exercise we proved that] there is at least one m_0 (better denoted $m_0(p)$) in $\{0, \ldots, m\}$ such that**

$$|f(m_0)| \geq \frac{1}{m} \stackrel{not.}{=} 2\gamma, \text{ where } f(m_0) \stackrel{def.}{=} \sum_{i=1}^{m_0} y_i p_i - \sum_{i=m_0+1}^{m} y_i p_i$$

- **[the proof made at parts a and c of the present exercise implies that] for any $s \in (x_{m_0+1}, x_{m_0}]$,** [a]

$$error_p(\phi_{s,+}) \leq \frac{1}{2} - \gamma \text{ or } error_p(\phi_{s,-}) \leq \frac{1}{2} - \gamma, \text{ where } \gamma \stackrel{not.}{=} \frac{1}{2m},$$

As a consequence, AdaBoost can choose at each iteration a weak hypothesis (h_t) for which $\gamma_t \geq \gamma = \frac{1}{2m}$.

[a] See the previous footnote.

Solution (cont'd)

d. Boosting takes $\frac{\ln m}{2\gamma^2}$ iterations to achieve zero [training] error, as shown at CMU, 2015, Z. Bar-Joseph, E. Xing, HW4, pr. 2.5, so with decision stumps we will achieve zero [training] error in at most $2m^2 \ln m$ iterations of boosting.

Each iteration of boosting introduces a single new weak hypothesis, so at most $2m^2 \ln m$ thresholded decision stumps are necessary.

A generalized version of the AdaBoost algorithm

MIT, 2003 fall, Tommy Jaakkola, HW4, pr. 2.1-3

Here we derive a *boosting algorithm* from a slightly *more general* perspective than the AdaBoost algorithm in CMU, 2015 fall, Z. Bar-Joseph, E. Xing, HW4, pr. 2.1-5, that will be applicable *for a class of loss functions* including the exponential one.

The *goal* is to generate *discriminant functions* of the form

$$f_K(x) = \alpha_1 h(x;\theta_1) + \ldots + \alpha_K h(x;\theta_K),$$

where both x belong to $\mathbb{R}^d$, θ are parameters, and you can *assume* that the weak classifiers $h(x;\theta)$ are decision stumps whose predictions are ± 1; any other set of weak learners would be fine without modification.

We *successively* add components to the overall discriminant function in a manner that will separate the *estimation of [the parameters of] the weak classifiers* from the *setting of the votes* α to the extent possible.

A useful definition

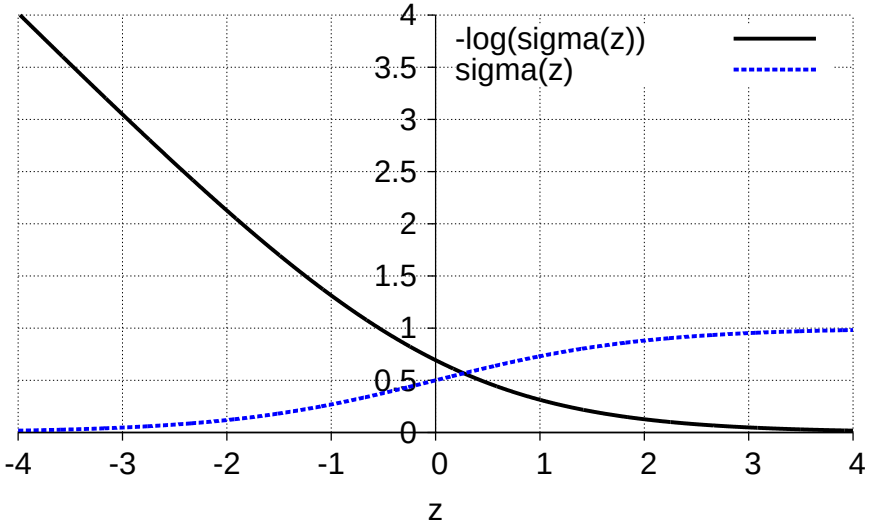
Let's start by defining a set of *useful loss functions*. The only *restriction* we place on the loss is that it should be a *monotonically decreasing* and *differentiable* function of its argument. The argument in our context is $y_i\, f_K(x_i)$ so that the more the discriminant function agrees with the ± 1 label y_i, the smaller the loss.
The simple *exponential loss* we have already considered [at CMU, 2015 fall, Z. Bar-Joseph, E. Xing, HW4, pr. 2.1-5], i.e.,

$$\text{Loss}(y_i f_K(x_i)) = \exp(-y_i f_K(x_i))$$

certainly conforms to this notion.

And so does the *logistic loss*

$$\begin{aligned} &\text{Loss}(y_i f_K(x_i)) \\ &\quad = \quad \ln(1+\exp(-y_i f_K(x_i))). \end{aligned}$$



Remark

Note that the logistic loss has a nice interpretation as a negative log-probability. Indeed, [recall that] for an additive *logistic regression* model

$$-\ln P(y=1|x,w) = -\ln \frac{1}{1+\exp(-z)} = \ln(1+\exp(-z)),$$

where $z = w_1\phi_1(x)+\ldots+w_K\phi_K(x)$ and we omit the bias term (w_0) for simplicity.

By replacing the additive combination of *basis functions* ($\phi_i(x)$) with the combination of weak classifiers ($h(x;\theta_i)$), we have an *additive logistic regression model* where the weak classifiers serve as the basis functions. The *difference* is that both the basis functions (weak classifiers) and the coefficients multiplying them will be estimated. In the logistic regression model we typically envision a fixed set of basis functions.

Let us now try to *derive the boosting algorithm* in a manner that can accomodate *any loss function* of the type discussed above. To this end, suppose we have already included $k-1$ component classifiers

$$f_{k-1}(x) = \hat{\alpha}_1 h(x; \hat{\theta}_1) + \ldots + \hat{\alpha}_{k-1} h(x; \hat{\theta}_{k-1}), \tag{7}$$

and we wish to add another $h(x;\theta)$. The *estimation criterion* for the overall discriminant function, including the new component with votes α, is given by

$$J(\alpha, \theta) = \frac{1}{m} \sum_{i=1}^{m} \text{Loss}(y_i f_{k-1}(x_i) + y_i \, \alpha \, h(x_i; \theta)).$$

Note that we explicate only how the objective depends on the choice of the last component and the corresponding votes since the parameters of the $k-1$ previous components along with their votes have already been set and won't be modified further.

We will *first* try to find the new component or parameters θ so as to maximize its potential in reducing the empirical loss, potential in the sense that we can subsequently adjust the votes to actually reduce the empirical loss. More precisely, we set θ so as to minimize the derivative

$$\begin{aligned}\frac{\partial}{\partial\alpha}J(\alpha,\theta)_{|\alpha=0} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial\alpha}\text{Loss}(y_i f_{k-1}(x_i)+y_i\,\alpha\,h(x_i;\theta))_{|\alpha=0}\\ &= \frac{1}{m}\sum_{i=1}^{m}dL(y_i f_{k-1}(x_i))\,y_i\,h(x_i;\theta),\end{aligned}\tag{8}$$

where $dL(z)\overset{not.}{=}\dfrac{\partial\text{Loss}(z)}{\partial z}$**.**

Note that this derivative $\dfrac{\partial}{\partial\alpha}J(\alpha,\theta)_{|\alpha=0}$ **precisely captures the amount by which we would start to reduce the empirical loss if we gradually increased the vote (α) for the new component with parameters θ. Minimizing this reduction seems like a sensible estimation criterion for the new component or θ. This plan permits us to first set θ and then subsequently optimize α to actually minimize the empirical loss.**

Let's rewrite the algorithm slightly to make it look more like a *boosting algorithm*. First, let's define the following weights and normalized weights on the training examples:

$$W_i^{(k-1)} = -dL(y_i f_{k-1}(x_i)) \text{ and}$$

$$\tilde{W}_i^{(k-1)} = \frac{W_i^{(k-1)}}{\sum_{j=1}^{m} W_j^{(k-1)}}, \text{ for } i = 1, \ldots, m.$$

These weights are guaranteed to be non-negative since the loss function is a decreasing function of its argument (its derivative has to be negative or zero).

Now we can rewrite the expression (8) as

$$\begin{aligned}
\frac{\partial}{\partial\alpha} J(\alpha,\theta)_{|\alpha=0} &= -\frac{1}{m}\sum_{i=1}^{m} W_i^{(k-1)}\, y_i\, h(x_i;\theta) \\
&= -\frac{1}{m}\Big(\sum_j W_j^{(k-1)}\Big)\cdot\sum_{i=1}^{m}\frac{W_i^{(k-1)}}{\sum_j W_j^{(k-1)}}\, y_i\, h(x_i;\theta) \\
&= -\frac{1}{m}\Big(\sum_j W_j^{(k-1)}\Big)\cdot\sum_{i=1}^{m}\tilde{W}_i^{(k-1)}\, y_i\, h(x_i;\theta).
\end{aligned}$$

By ignoring the multiplicative constant (i.e., $\frac{1}{m}\sum_j W_j^{(k-1)}$, which is constant at iteration k), we will estimate θ by *minimizing*

$$-\sum_{i=1}^{m}\tilde{W}_i^{(k-1)} y_i h(x_i;\theta), \tag{9}$$

where the normalized weights $\tilde{W}_i^{(k-1)}$ sum to 1. (This is the same as maximizing the weighted agreement with the labels, i.e., $\sum_{i=1}^{m}\tilde{W}_i^{(k-1)} y_i h(x_i;\theta)$.)

Some remarks (by Liviu Ciortuz)

1. Using some familiar notations, we can write

$$\begin{aligned}
\sum_{i=1}^{m} \tilde{W}_i^{(k-1)} y_i h(x_i;\theta) &= \sum_{i\in C} \tilde{W}_i^{(k-1)} y_i h(x_i;\theta) + \sum_{i\in M} \tilde{W}_i^{(k-1)} y_i h(x_i;\theta) \\
&= \underbrace{\sum_{i\in C} \tilde{W}_i^{(k-1)}}_{1-\varepsilon_k} - \underbrace{\sum_{i\in M} \tilde{W}_i^{(k-1)}}_{\varepsilon_k} = 1 - 2\varepsilon_k \\
\Rightarrow \varepsilon_k &= \frac{1}{2}\left(1 - \sum_{i=1}^{m} \tilde{W}_i^{(k-1)} y_i h(x_i;\hat{\theta}_k)\right)
\end{aligned}$$

2. Because $\tilde{W}_i^{(k-1)} \geq 0$ and $\sum_{i=1}^{m} \tilde{W}_i^{(k-1)} = 1$, it follows that $\sum_{i=1}^{m} \tilde{W}_i^{(k-1)} y_i h(x_i;\theta) \in [-1,+1]$, therefore

$1 - (\sum_{i\in C} \tilde{W}_i^{(k-1)} - \sum_{i\in M} \tilde{W}_i^{(k-1)}) = 1 - \underbrace{\sum_{i=1}^{m} \tilde{W}_i^{(k-1)} y_i h(x_i;\hat{\theta}_k)}_{\in[-1,+1]} \in [0,+2]$**, and so**

$\frac{1}{2}\left(1 - \sum_{i=1}^{m} \tilde{W}_i^{(k-1)} y_i h(x_i;\hat{\theta}_k)\right) \in [0,+1]$**.**

We are now ready to cast the steps of the boosting algorithm in a form similar to the AdaBoost algorithm given at CMU, 2015 fall, Z. Bar-Joseph, E. Xing, HW4, pr. 2.1-5.

[Assume $\tilde{W}_i(0) = \dfrac{1}{m}$ **and** $f_0(x_i) = 0$ **for** $i = 1, \ldots . m$**.]**

***Step 1*: Find any classifier** $h(x; \hat{\theta}_k)$ **that performs better than chance with respect to the weighted training error:**

$$\varepsilon_k = \frac{1}{2}\left(1 - \sum_{i=1}^{m} \tilde{W}_i^{(k-1)} y_i h(x_i; \hat{\theta}_k)\right). \tag{10}$$

***Step 2*: Set the votes** α_k **for the new component by minimizing the overall empirical loss:**

$$J(\alpha, \hat{\theta}_k) \quad = \quad \frac{1}{m}\sum_{i=1}^{m} \text{Loss}(y_i\, f_{k-1}(x_i) + y_i\, \alpha\, h(x_i; \hat{\theta}_k)), \text{ and so } \alpha_k = \arg\min_{\alpha \geq 0} J(\alpha, \hat{\theta}_k).$$

***Step 3*: Recompute the normalized weights for the next iteration according to**

$$\tilde{W}_i^{(k)} = -c_k \cdot dL(\underbrace{y_i\, f_{k-1}(x_i) + y_i\, \alpha_k\, h(x_i; \hat{\theta}_k)}_{y_i\, f_k(x_i)}) \text{ for } i = 1, \ldots, m, \tag{11}$$

where c_k **is chosen so that** $\sum_{i=1}^{m} \tilde{W}_i^{(k)} = 1$**.**

One more remark (by Liviu Ciortuz),

now concerning Step 1:

Normally there should be such $\varepsilon_k \in (0, 1/2)$ (in fact, some corresponding $\hat{\theta}_k$), because if for some h we would have $\varepsilon_k \in (1/2, 1)$, then we can take $h' = -h$, and the resulting ε'_k would belong to $(0, 1/2)$.

The are only two *exceptions*, which correspond to the case when for any hypothesis h we would have

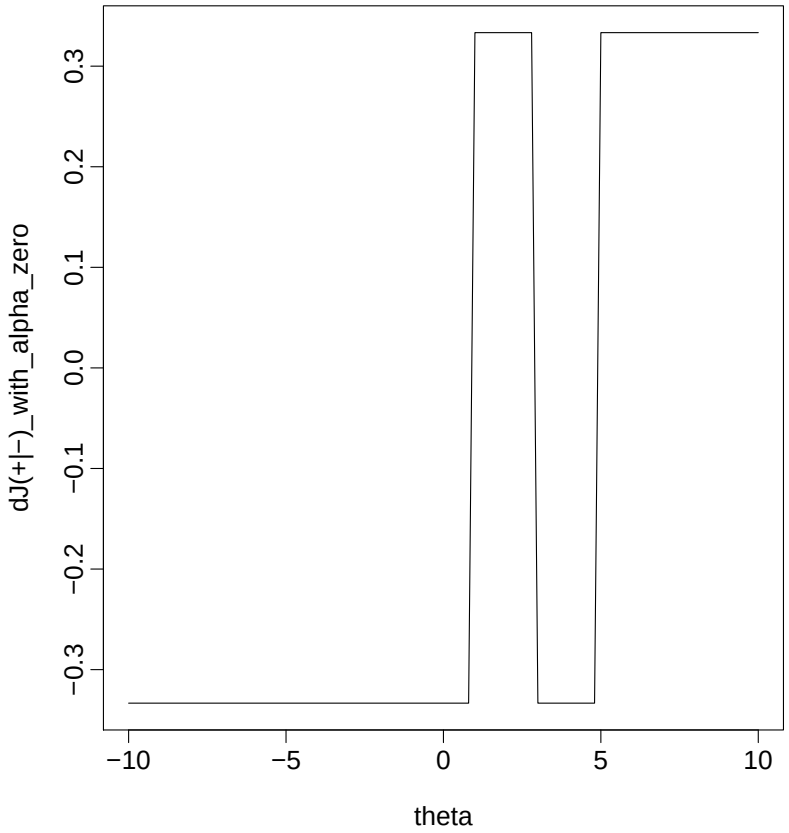
– either $\varepsilon_k = 1/2$, in which case $\sum_{i \in C} \tilde{W}_i^{(k-1)} = \sum_{i \in M} \tilde{W}_i^{(k-1)}$

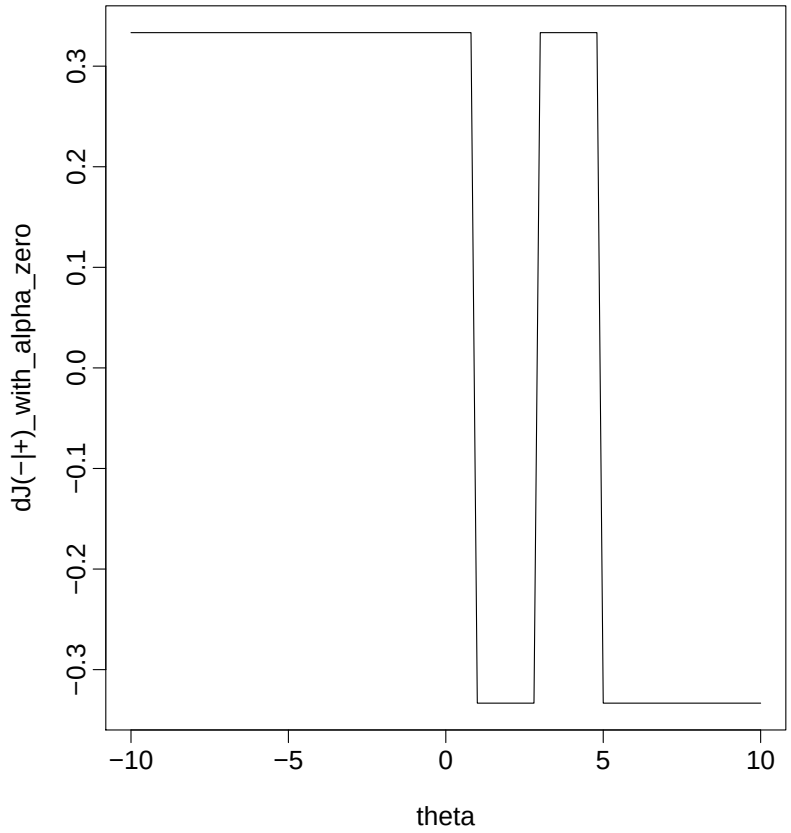
– or $\varepsilon_k \in \{0, 1\}$, in which case either h or $h' = -h$ is a *perfect* (therefore not *weak*) classifier for the given training data.

Exemplifying Step 1 on data from CMU, 2012 fall, T. Mitchell, Z. Bar-Joseph, final, pr. 8.a-e

[graphs made by MSc student Sebastian Ciobanu, FII, 2018 fall]

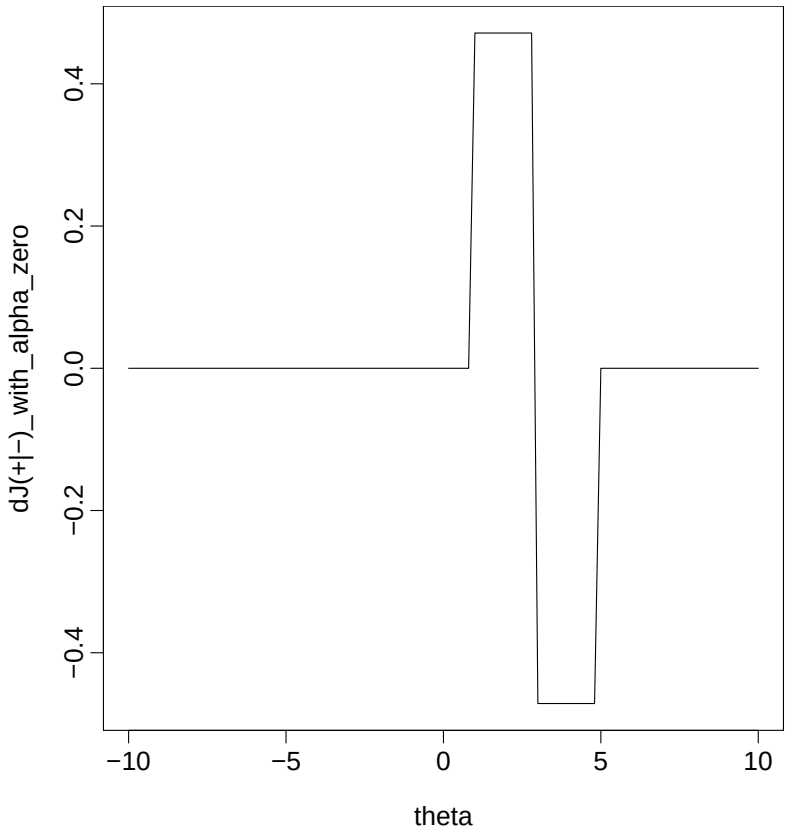
Iteration 1

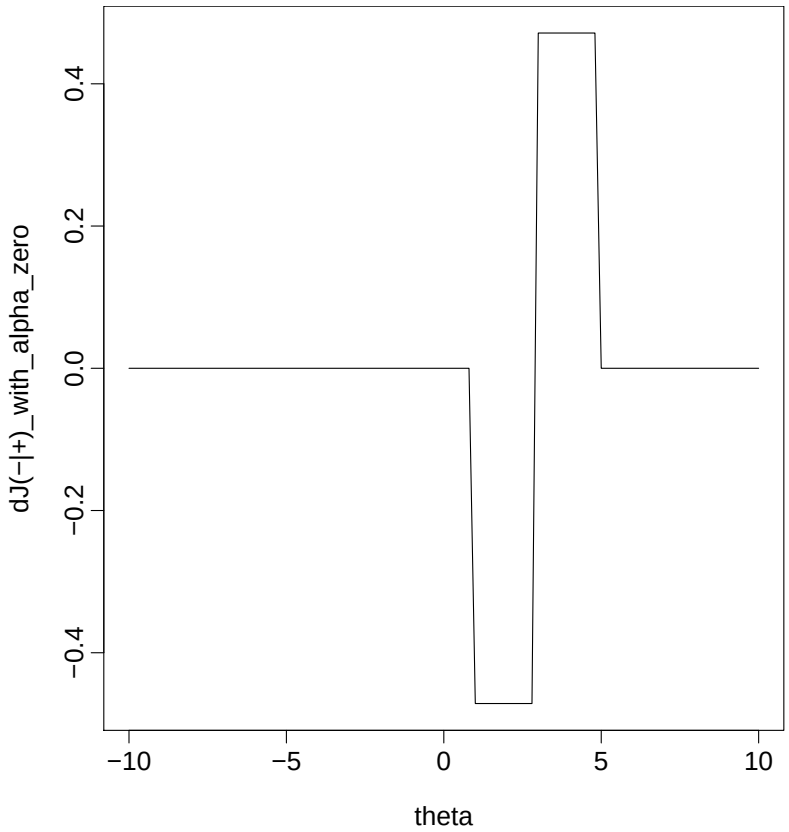




[graphs made by MSc student Sebastian Ciobanu, FII, 2018 fall]

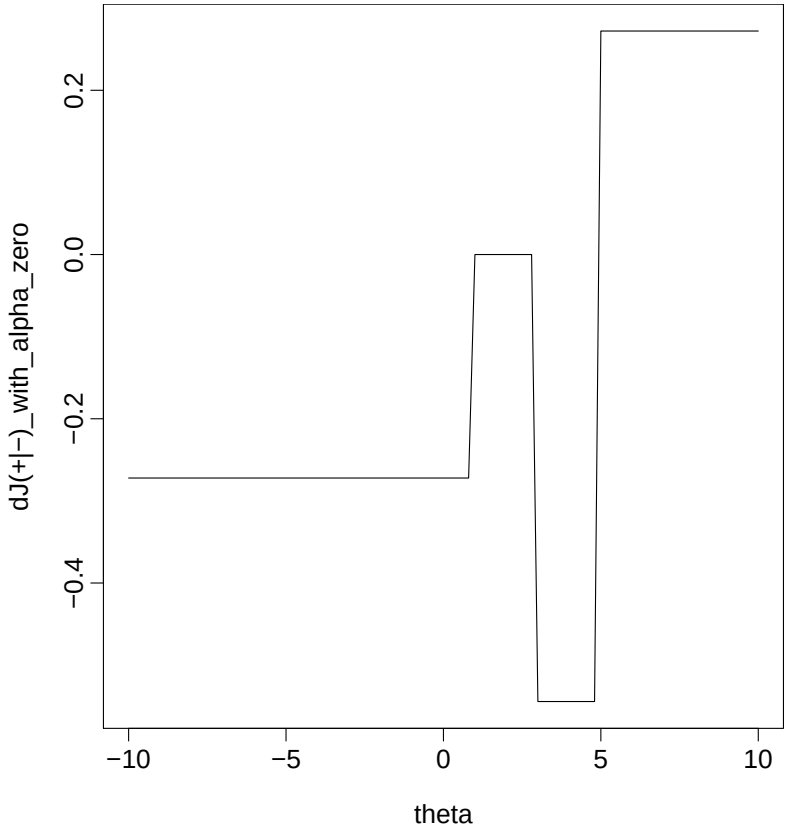
Iteration 2

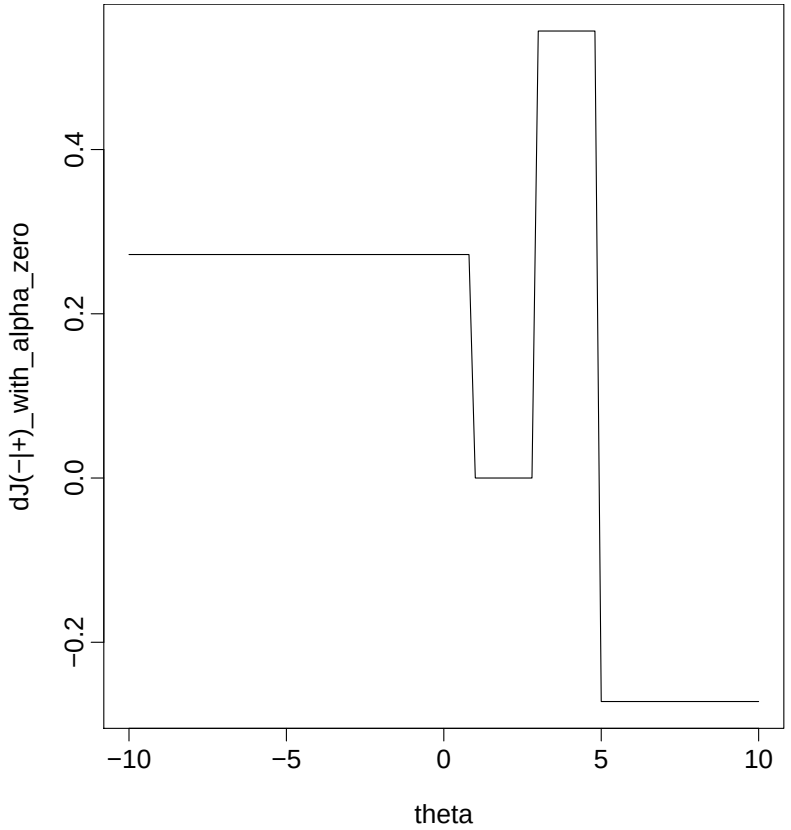




[graphs made by MSc student Sebastian Ciobanu, FII, 2018 fall]

Iteration 3





a. Show that the three steps in the algorithm correspond exactly to AdaBoost when the loss function is the exponential loss $\text{Loss}(z) = \exp(-z)$.

More precisely, show that in this case the setting of α_k based on the new weak classifier and the weight update to get $\tilde{W}_i^{(k)}$ would be identical to AdaBoost. (In CMU, 2015 fall, Z. Bar-Joseph, E. Xing, HW4, pr. 2.1-5, $\tilde{W}_i^{(k)}$ corresponds to $D_k(i)$.)

Solution

For the first part, we will show that the minimization in Step 2 of the general algorithm (LHS below), with $\text{Loss}(z) = e^{-z}$, is the same as the minimization performed by AdaBoost (RHS below), i.e. that

$$\arg\min_{\alpha>0} \sum_{i=1}^{m} \text{Loss}(y_i f_{k-1}(x_i) + \alpha y_i h(x_i; \hat{\theta}_k)) = \arg\min_{\alpha>0} \sum_{i=1}^{m} \tilde{W}_i^{(k-1)} \exp(-\alpha y_i h(x_i; \hat{\theta}_k)),$$

with (from AdaBoost)

$$\tilde{W}_i^{(k-1)} = c_{k-1} \cdot \exp(-y_i f_{k-1}(x_i)),$$

where c_{k-1} is a normalization constant (weights sum to 1).

Solution (cont'd)

Evaluating the objective in LHS gives:

$$\sum_{i=1}^{m} \text{Loss}(y_i f_{k-1}(x_i) + \alpha y_i h(x_i; \hat{\theta}_k))$$

$$= \sum_{i=1}^{m} \exp(-y_i f_{k-1}(x_i)) \, \exp(-\alpha y_i h(x_i; \hat{\theta}_k)) \overset{(11)}{=} \frac{1}{c_{k-1}} \sum_{i=1}^{m} \tilde{W}_i^{(k-1)} \exp(-\alpha y_i h(x_i; \hat{\theta}_k))$$

$$= \frac{1}{c_{k-1}} \Big[\sum_{i \in \{i | y_i h(x_i; \hat{\theta}_k) = 1\}} \left(\tilde{W}_i^{(k-1)} \right) e^{-\alpha} + \sum_{i \in \{i | y_i h(x_i; \hat{\theta}_k) = -1\}} \left(\tilde{W}_i^{(k-1)} \right) e^{\alpha} \Big],$$

which is proportional to the objective minimized by AdaBoost so that minimizing value of α is the same for both algorithms (see CMU, 2008 fall, Eric Xing, HW3, pr. 4.1.1).

For the second part, note that the weight assignment in Step 3 of the general algorithm (for stage k) is

$$\tilde{W}_i^{(k)} = -c_k \cdot dL(y_i f_k(x_i)) = c_k \cdot \exp(-y_i f_k(x_i)),$$

which is the same as in AdaBoost (see CMU, 2015 fall, Z. Bar-Joseph, E. Xing, HW4, pr. 2.2).

136.

b. Show that for any valid loss function of the type discussed above, the new component $h(x;\hat{\theta}_k)$ just added at the k-th iteration would have weighted training error exactly $1/2$ relative to the updated weights $\tilde{W}_i^{(k)}$. If you prefer, you can show this only in the case of the logistic loss.

Solution

At stage k, α_k is chosen to minimize $J(\alpha,\hat{\theta}_k)$, i.e. to solve $\dfrac{\partial J(\alpha,\hat{\theta}_k)}{\partial\alpha}=0$. In general,

$$\begin{aligned}\frac{\partial}{\partial\alpha}J(\alpha,\hat{\theta}_k) &= \frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial\alpha}\mathrm{Loss}(y_i f_{k-1}(x_i)+y_i\alpha h(x_i;\hat{\theta}_k))\\ &= \frac{1}{m}\sum_{i=1}^{m}\underbrace{dL(y_i f_{k-1}(x_i)+y_i\alpha h(x_i;\hat{\theta}_k))}_{-\frac{\tilde{W}_i^{(k)}}{c_k}}y_i h(x_i;\hat{\theta}_k)\propto\sum_{i=1}^{m}\tilde{W}_i^{(k)}y_i h(x_i;\hat{\theta}_k),\end{aligned}$$

so that we must have $\tilde{W}_i^{(k)}y_i h(x_i;\hat{\theta}_k)=0$.Then, the weighted training error for $h(x;\hat{\theta}_k)$ (relative to the updated weights $\tilde{W}_i^{(k)}$ determined by α_k) can be computed in a similarly way to (10):

$$\frac{1}{2}\Big(1-\underbrace{\sum_{i=1}^{m}\tilde{W}_i^{(k)}y_i h(x_i;\hat{\theta}_k)}_{0}\Big)=\frac{1}{2}(1-0)=\frac{1}{2}.$$

c. Now, suppose that we change the objective function to $J(f_k) = \sum_{i=1}^{m}(y_i - f_k(x_i))^2$ and we still want to optimize it sequentially.[a] What is the new update rule for α_k?

Solution

We will compute the derivative of $J(f_k) = \sum_{i=1}^{m}(y_i - f_k(x_i))^2$ and set it to zero to find the value of α_k.

$$\frac{\partial J(f_k)}{\partial \alpha_k} = \frac{\partial \sum_{i=1}^{m}(y_i - f_k(x_i))^2}{\partial \alpha_k} = \sum_{i=1}^{m} 2(y_i - f_k(x_i)) \frac{\partial (y_i - f_k(x_i))}{\partial \alpha_k}.$$

We also know that

$$f_k(x_i) = f_{k-1}(x_i) + \alpha_k h_k(x_i).$$

In this equation, f_{k-1} is independent of α_k. Substituting this in the derivative equation, we get

$$\frac{\partial J(f_k)}{\partial \alpha_k} = 2\sum_{i=1}^{m}(y_i - f_k(x_i)) \frac{\partial (y_i - f_{k-1}(x_i) - \alpha_k h_k(x_i))}{\partial \alpha_k} = 2\sum_{i=1}^{m}(y_i - f_k(x_i))(-h_k(x_i))$$

[a] LC: Note that $(y_i - f(x_i))^2 = [y_i(1 - y_i f_k(x_i))]^2 = (1 - y_i f_k(x_i))^2 = (1 - z_i)^2$, where $z_i = y_i f_k(x_i)$. The function $(1 - z)^2$ is derivable and convex; it is decreasing on $(-\infty, 1]$ and increasing on $[1, +\infty)$.

Solution (cont'd)

Setting the derivative to zero, we get

$$\frac{\partial J(f_k)}{\partial \alpha_k} = 0 \Leftrightarrow \sum_{i=1}^{m}(y_i - f_k(x_i))h_k(x_i) = 0 \Leftrightarrow \sum_{i=1}^{m}(y_i - \alpha_k h_k(x_i) - f_{k-1}(x_i))h_k(x_i) = 0$$

$$\sum_{i=1}^{m}(y_i - f_{k-1}(x_i))h_k(x_i) = \alpha_k \sum_{i=1}^{m} h_k^2(x_i) \Leftrightarrow \alpha_k = \frac{\sum_{i=1}^{m}(y_i - f_{k-1}(x_i))h_k(x_i)}{\sum_{i=1}^{m} \underbrace{h_k^2(x_i)}_{1}}$$

$$\Leftrightarrow \alpha_k = \frac{1}{m}\sum_{i=1}^{m}(y_i - f_{k-1}(x_i))h_k(x_i).$$

MIT, 2006 fall, Tommi Jaakkola, HW4, pr. 3.a
MIT, 2009 fall, Tommi Jaakkola, HW3, pr. 2.1

d. Show that if we use the *logistic loss* instead [of the exponential loss] the unnormalized weights $W_i^{(k)}$ are bounded by 1.

Solution

$W_i^{(k)}$ **was defined as** $-\mathrm{dL}(y_i f_k(x_i))$**, with** $\mathrm{dL}(z) \overset{not.}{=} -\frac{\partial}{\partial z}(\ln(1+e^{-z}))$**. Therefore,**

$$W_i^{(k)} = \frac{e^{-z_i}}{1+e^{-z_i}} = \frac{1}{1+e^{z_i}} < 1, \text{ where } z_i = y_i f_k(x_i).$$

e. When using the *logistic loss*, what are the normalized weights, $\tilde{W}_i^{(k)}$? Express the weights as a function of the agreements $y_i f_k(x_i)$, where we have already included the k-th weak learner.

What can you say about the resulting normalized weights for examples that are clearly misclassified in comparison to those that are just slightly misclassified by the current ensemble?

If the training data contains mislabeled examples, why do we prefer the logistic loss over the exponential loss, $\text{Loss}(z) = \exp(-z)$?

Solution

The normalized weights are given by $\tilde{W}_i^{(k)} = c_k \cdot \dfrac{\exp(-y_i f_k(x_i))}{1+\exp(-y_i f_k(x_i))}$, **with the normalization constant** $c_k = \left(\sum_{i=1}^{m} \dfrac{\exp(-y_i f_k(x_i))}{1+\exp(-y_i f_k(x_i))} \right)^{-1}$.

[Answer from MIT, 2011 fall, Leslie P. Kaelbling, HW5, pr. 1.1]

For clearly misclassified examples, $y_i\, h_k(x_i)$ is a large negative, so $W_i^{(k)}$ is close to [and less than] 1, while for slightly misclassifed examples, $W_i^{(k)}$ is close to [and greater than] $1/2$. Thus, the normalized weights for the two respective cases will be in a ratio of at most $2:1$, i.e. a single clearly misclassifed outlier will never be worth more than two completely uncertain points. This is why boosting [with logistic loss function] is robust to outliers.

Solution (cont'd) in Romanian

LC: Pentru ultima parte de la punctul e nu am găsit deloc răspuns la MIT, însă pot gândi astfel:

În cazul funcţiei de pierdere logistice, dacă avem un x_i care ar avea de drept eticheta $y_i = +1$, dar se consideră (în mod eronat) $y_i = -1$, pierderea este de aproximativ $f_k(x_i)$ dacă $f_k(x_i) > 0$,[a] în vreme ce în cazul funcţiei de pierdere [negativ] exponenţiale pierderea este $\exp(f_k(x_i))$ care este în general mult mai mare decât $f_k(x_i)$. Cazurile simetrice ($f_k(x_i) \leq 0$ şi apoi $y_i = -1 \to +1$) se tratează în mod similar.

[a] Vedeţi graficul funcţiei de *pierdere logistică* din enunţul problemei de faţă.

MIT, 2006 fall, Tommi Jaakkola, HW4, pr. 3.b

f. Suppose we use logistic loss and the training set is linearly separable. We would like to use a linear support vector machine (no slack penalties) as a base classifier [LC: or any linear separator consitent with the training data]. Assume that the generalized AdaBoost algorithm minimizes the ε_k weighted error at Step 1. In the first boosting iteration, what would the resulting α_1 be?

Solution

In Step 1, we pick θ_1. We wish to find θ_1 to minimize $\frac{\partial J(\alpha,\theta)}{\partial \alpha}|_{\alpha=0}$. Equivalently, this θ_1 is chosen to minimize the weighted sum: $2\varepsilon_1 - 1 = -\sum_{i=1}^{m} \tilde{W}_0(i) y_i h(x_i;\theta)$, where $\tilde{W}_0(i) = \frac{1}{m}$ for all $i = 1, 2, \ldots, m$. If the training set is linearly separable with offset, then the no-slack SVM problem is feasible. Hence, the base classifier in this case will thus be an affine (linear with offset) separator $h(\cdot;\theta_1)$, which satisfies the inequality $y_i h(x_i;\theta_1) \geq 1$ for all $i = 1, 2, \ldots, m$.

In Step 2, we pick α_1 to minimize $J(\alpha_1,\theta_1) = \sum_{i=1}^{m} L(y_i h_0(x_i) + \alpha_1 y_i h(x_i;\theta_1)) = \sum_{i=1}^{m} L(\alpha_1 y_i h(x_i;\theta_1))$. Note that $J(\alpha_1,\theta_1)$ is a sum of terms that are strictly decreasing in α_1 (as $y_i h(x_i;\theta_1) \geq 1$); therefore, it itself is also strictly decreasing in α_1. It follows that the boosting algorithm will take $\alpha_1 = \infty$ in order to minimize $J(\alpha_1,\theta_1)$.

This makes sense, because if we can find a base classifier that perfectly separates the data, we will weight it as much as we can to minimize the boosting loss. The lesson here is simple: when doing boosting, we need to use base classifiers that are not powerful enough to perfectly separate the data.