University of Cagliari

MSc programs in Computer Engineering, Cybersecurity and Artificial Intelligence, and in Electronic Engineering

Artificial Intelligence

Academic Year: 2025/2026

Instructor: Giorgio Fumera, Ambra Demontis

Exercises on machine learning

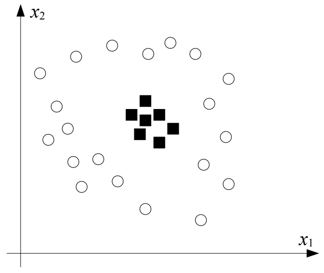
1. Define: (a) a feed-forward multi-layer perceptron network with a single hidden layer, and (b) a decision tree (trying to keep it as small as possible), to represent the Boolean function $f(a, b, c) = \overline{a} \cdot b \cdot c + \overline{b} \cdot c$.

2. Consider a two-class problem with two real-valued attributes x_1 and x_2, and a training set made up of the following six examples:

   - class 1: $\{(0.3, 0), (0.7, 0), (0.5, 0.3)\}$
   - class 0: $\{(0, 0.2), (0.5, 0.7), (1, 0.2)\}$

   Define, if possible, a *consistent* classifier, using: (i) a perceptron, (ii) a feed-forward multi-layer perceptron network, (iii) a decision tree.

3. The figure below shows the training set of a two-class problem ("black squares" vs "white circles"), characterised by two real-valued attributes x_1 and x_2:



   Determine whether it is possible to obtain a consistent classifier using:

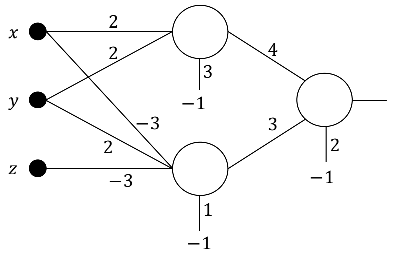
   - a decision tree,
   - a perceptron,
   - a feed-forward multi-layer perceptron network.

   For each affirmative answer define one possible consistent classifier of the corresponding type, trying to minimise its *complexity*.

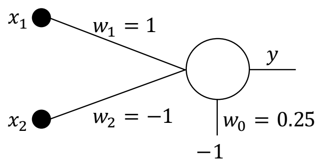
4. Consider the perceptron network in the figure below, where the inputs x, y and z are Boolean values represented as 0 and 1.

   (a) What is the Boolean function represented by this network?

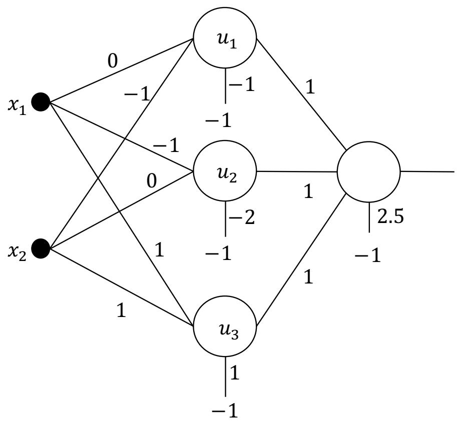
   (b) Define a decision tree that represents the same Boolean function.



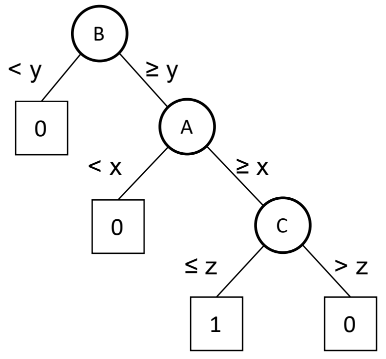
5. The figure below shows the initial values of the connection weights of a perceptron which has to be used as a two-class classifier, with two real-valued attributes x_1 and x_2. Assuming that the first training example processed by the perceptron learning algorithm is $x_1 = 0.5, x_2 = 0.5$ and that its class label is $t = 1$, determine whether the values of the connection weights w_0, w_1 and w_2 will be *increased*, *decreased* or left *unchanged*.

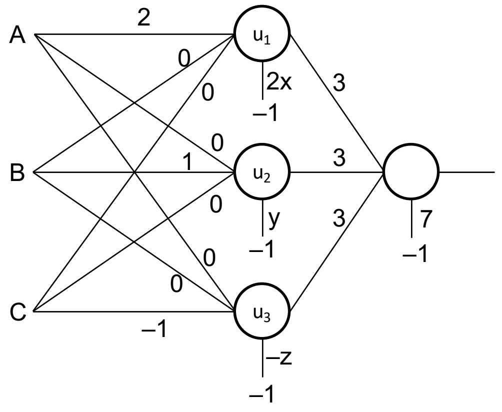


6. Consider the following feed-forward multi-layer *perceptron* network with real-valued inputs x_1 and x_2. Draw the corresponding decision regions in the input space, i.e., the regions of the $x_1 x_2$ plane where the network output is 0 or 1.



7. Consider a two-class classification problem with three real-valued attributes denoted by a, b and c, and let the class labels be denoted by 0 and 1. Two possible classifiers are shown below: a decision tree and a *perceptron* network, where symbols x, y and z denote three constant values. Do the decision tree and the perceptron network represent *identical* decision regions?

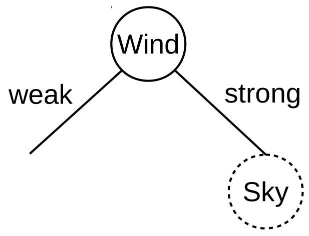




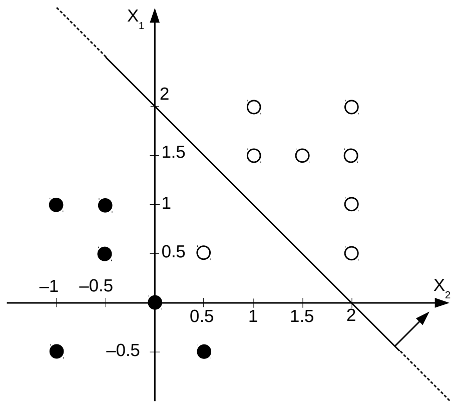
8. Mary is a tennis lover and would like to play tennis every day with her friend Alice, but sometimes she has to give up due to several reasons, including weather conditions. To predict whether Mary will be willing to play tennis given the weather conditions, Alice collected some statistics about sky conditions, temperature, humidity and wind over 14 different days, together with Mary's decision:

Day	Sky	Temperature	Humidity	Wind	Decision
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	high	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no

Now Mary would like to automatically infer a set of rules to predict Mary's decision, in the form of a decision tree. Assuming that the attribute `Wind` has already been associated with the root node, how does the ID3 learning algorithm evaluate the discriminant capability of the attribute `Sky` in the node shown below?



9. The figure below shows the training set of a two-class problem involving two real-valued attributes x_1 and x_2, and the decision boundary produced by a perceptron (the arrow points toward the region where the perceptron output is 1), assuming that the desired value of the perceptron output for the class "white circles" ($\circ$) is 1. Note that only one training example is misclassified, i.e., the "white circle" $(0.5, 0.5)$.



(a) Determine a possible set of values for the connection weights of the perceptron.

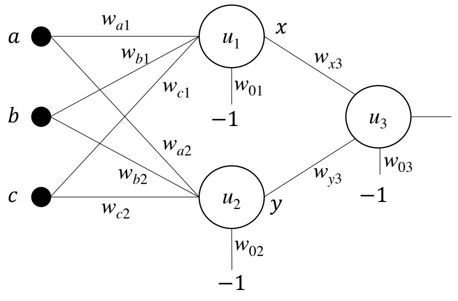
(a) What is the value of the error function of the perceptron learning algorithm for the misclassified training example, given the connection weights determined in the answer to question (a)?

(c) What values will the connection weights be assigned after the learning algorithm processes the misclassified training example?

10. Define: (a) a feed-forward multi-layer perceptron network with the lowest complexity to represent the Boolean function $f(a, b, c) = \overline{a} \cdot b \cdot c + a \cdot b \cdot c + a \cdot \overline{b} \cdot c + a \cdot \overline{b} \cdot \overline{c}$.

Solution

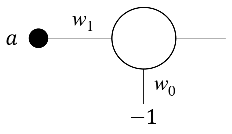
1. (a) Since a single perceptron can represent both the AND and OR functions between any number of variables, the required network can be defined in terms of two hidden units u_1 and u_2 that compute $\overline{a} \cdot b \cdot c$ and $\overline{b} \cdot c$, respectively (let their outputs be x and y), and one output unit u_3 which computes $x + y$:



To represent the AND function between n variables, $x_1 \cdot x_2 \cdot \ldots \cdot x_n$, one can set the weights $w_1, \ldots, w_n$ of all the input connections, e.g., to 1, which corresponds to an activation $\sum_{i=0}^{n} w_i x_i = \sum_{i=1}^{n} x_1 - w_0$; then, to have a non-negative activation (and thus a perceptron output equal to 1) only when *all* the inputs equal 1 ($x_1 = x_2 = \ldots = x_n = 1$), it is easy to see that the weight of the bias connection, w_0, can be set to any value in $(n-1, n]$, e.g., $w_0 = n - 0.5$.

Also to represent the OR function one can set $w_1 = w_2 = \ldots = w_n = 1$; then, to have a negative activation $\sum_{i=1}^{n} x_1 - w_0$ (and thus a perceptron output equal to 0) only when *all* the inputs equal 0 ($x_1 = x_2 = \ldots = x_n = 0$), the weight w_0 can be set to any value in $(0, 1]$, e.g., $w_0 = 0.5$.

Consider now how to represent the NOT function, e.g., $f(a) = \overline{a}$:



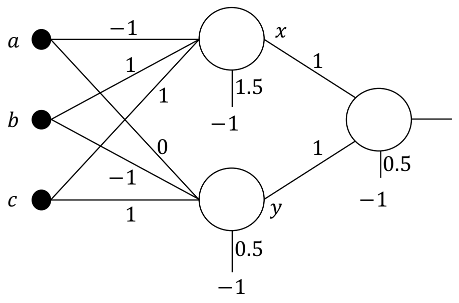
The activation of the above perceptron is $w_1 a - w_0$; it should be non-negative when $a = 0$, i.e., $-w_0 \geq 0$, wheras it should be negative when $a = 1$, i.e., $w_1 - w_0 < 0$. A possible choice of the two connection weights that fulfils the above conditions is $w_0 = 0, w_1 = -1$.

Using the above results it is not difficult to see how to set the connection weights of unit u_1 to represent the function $x = \overline{a} \cdot b \cdot c$. If one sets $w_{a1} = -1$, $w_{b1} = 1$ and $w_{c1} = 1$, the activation is $-a + b + c - w_{01}$. It should be non-negative only when $a = 0, b = 1, c = 1$, which implies $2 - w_{01} \geq 0$; note that for the other combinations of values of a, b and c the activation ranges from $-1 - w_{01}$ to $+1 - w_{01}$. It easily follows that w_{01} can be set to any value in $(1, 2]$, e.g., $w_{01} = 1.5$.

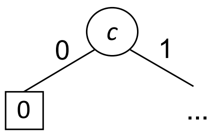
Similarly, to represent the function $y = \overline{b} \cdot c$ the connection weights of unit u_2 can be set to $w_{a2} = 0$, $w_{b2} = -1$ and $w_{c2} = 1$, and consequently w_{02} can be set, e.g., to 0.5.

Finally, since unit u_3 should represent the OR function between x and y, its connection weights can be set to $w_{x3} = w_{y3} = 1$, $w_{03} = 0.5$.

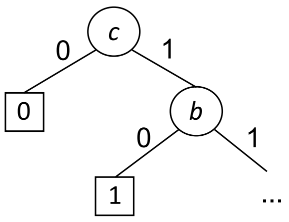
The corresponding perceptron network is:



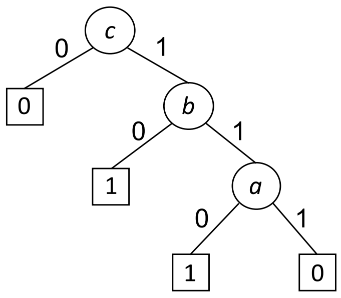
(b) Several different decision trees can represent the considered function. To build a small tree one can observe that, if $c = 0$, then $f = 0$: therefore c can be conveniently associated with the root node:



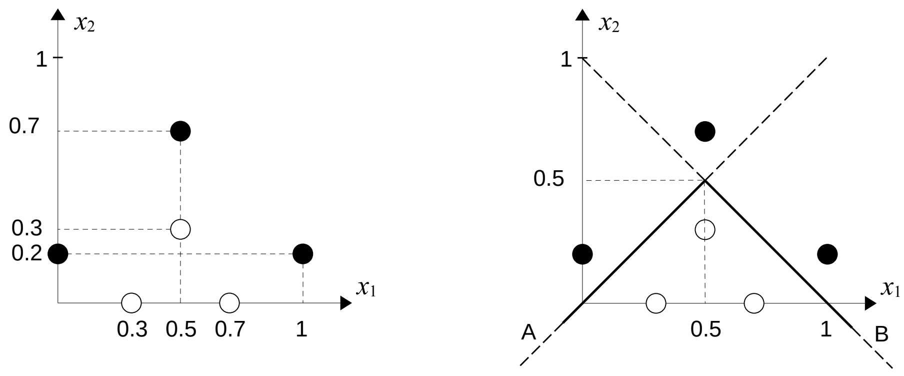
Moreover, when $c = 1$, the term $\overline{b} \cdot c$ implies that $f = 1$, if $b = 0$:



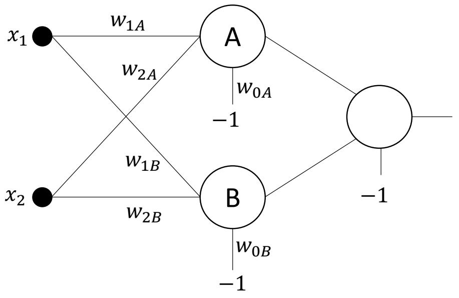
Finally, when $c = b = 1$ the value of f is determined by the term $\overline{a} \cdot b \cdot c$, that is, only by the value of the variable a: $f = 1$, if $a = 0$, and $f = 0$, if $a = 1$. The complete decision tree is therefore the following one:



2. The training examples are shown in the figure below on the left, as white circles for class 1 and black circles for class 0. It is easy to see that the examples of the two classes are not linearly separable, and thus a perceptron cannot provide a consistent classifier.



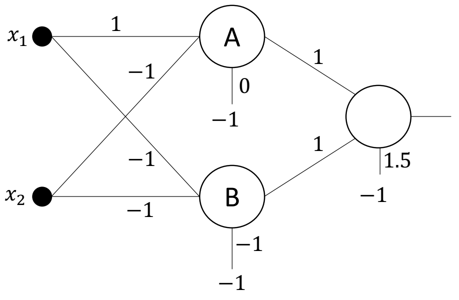
They can be separated, instead, by a piece-wise linear boundary made up of two half-lines, like the ones denoted by `A` and `B` in the figure above in the middle. This means that a consistent solution can be obtained by a feed-forward multi-layer perceptron network with one hidden layer made up of two hidden units (see figure below), whose activations correspond to the equations of line `A`: $x_2 = x_1$, and of line `B`: $x_2 = 1 - x_1$.



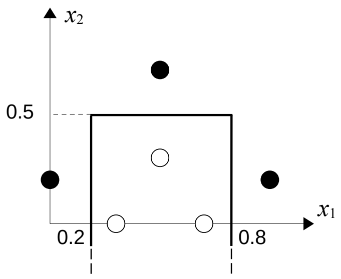
More precisely, denoting also the hidden units as `A` and `B`, and requiring that the output of the perceptron network should be 1 for the class of "white circles" and 0 for the class of "black circles", a convenient choice is to make unit `A` output the value 1 in the bottom-right half-plane defined by line `A`, i.e., $x_2 \leq x_1$, and to make unit `B` output the value 1 in the bottom-left half-plane defined by line `B`, i.e., $x_2 \leq 1 - x_1$. Accordingly, the output of the perceptron network should be $y = 1$, when the output of both units `A` and `B` equals 1, which corresponds to the `AND` logical function.

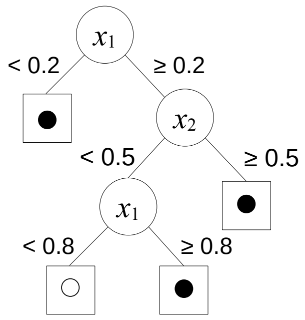
To determine the values of the connection weights of the three units, one can leverage the formulation to compute the equation of a line passing for two given points (x_{11}, x_{21}) and (x_{12}, x_{22}). First one should compute the line slope: $m = \frac{x_{22}-x_{21}}{x_{12}-x_{11}}$. Then, the slope and the coordinate of one of the two points, for example, the first can be employed to compute the equation: $x_2 - x_{21} = m(x_1 - x_{11})$. Considering the unit `A`, $(x_{11}, x_{21}) = (0, 0)$, $(x_{12}, x_{22}) = (0.5, 0.5)$, therefore, the slope is $\frac{0.5-0}{0.5-0} = \frac{0.5}{0.5} = 1$. Subsituting the first point and the slope we obtain the equation: $x_2 - 0 = 1(x1 - 0)$ which is equal to $x_2 = x_1$ if we move all the term on the same side we have $-x_1 + x_2 = 0$. Now we can take one point for which we would like the neuron to fire, e.g., the point $(1, 0)$. Substituting to x_1 and x_2 we have $-1 + 0$ which is lower than 0. Therefore, the equations of the neuron, to fire in this case is $-x_1 + x_2 \leq 0$ which is also equal to $+x_1 - x_2 \geq 0$. Considering the unit `B`, $(x_{11}, x_{21}) = (1, 0)$, $(x_{12}, x_{22}) = (0.5, 0.5)$, therefore, the slope is $\frac{0.5-0}{0.5-1} = \frac{0.5}{-0.5} = -1$. Subsituting the first point and the slope we obtain the equation: $x_2 - 0 = -1(x1 - 1)$ which is equal to $x_2 = -x_1 + 1$ if we move all the term on the same side we have $x_1 + x_2 - 1 = 0$. Now we can take one point for which we would like the neuron to fire, e.g., the point $(0, 0.5)$. Substituting to x_1 and x_2 we have $0 + 0.5 - 1$ which is lower than 0. Therefore, the equations of the neuron, to fire in this case is $x_1 + x_2 - 1 \leq 0$ which is also equal to $-x_1 - x_2 + 1 \geq 0$. Finally, the weight values of the output unit can be set as previously shown for the `AND` function.

The resulting perceptron network is:

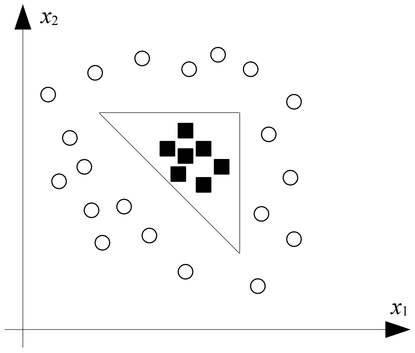


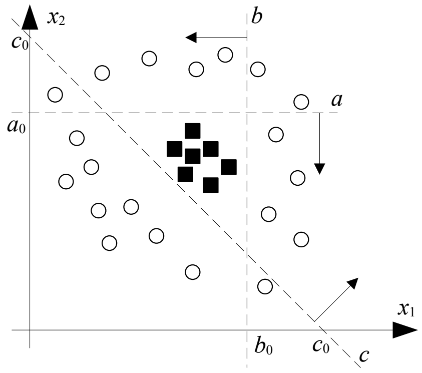
The examples of the two classes can also be separated by lines parallel to the axes, as shown in the figure below on the left. The corresponding decision regions can therefore be represented by a decision tree, like the one in the figre below on the right: the root node splits the attribute space through the line $x_1 = 0.2$, its right child splits the half-plane corresponding to $x_1 \geq 0.2$ through the line $x_2 = 0.5$, and finally the right child of the latter node splits the region delimited by $x_1 = 0.2$ and $x_1 \geq 0.2$ through the line $x_1 = 0.8$.





3. Also in this case the examples of the two classes are not linearly separable, thus a perceptron cannot provide a consistent classifier. However, the black squares can be separated from the white circles by enclosing them inside a polygon, i.e., through a convex and piece-wise linear class boundary; to this aim a triangle is sufficient as shown in the figure below on the left:

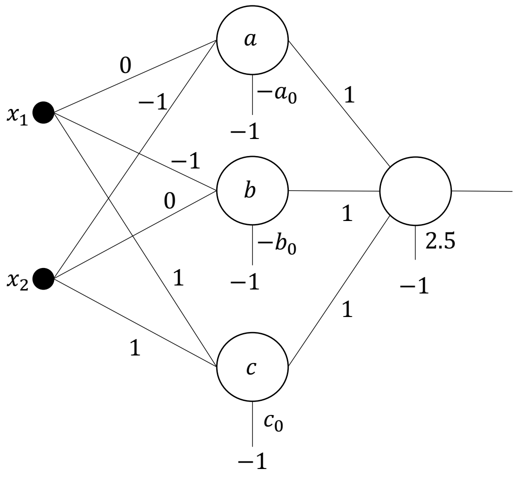




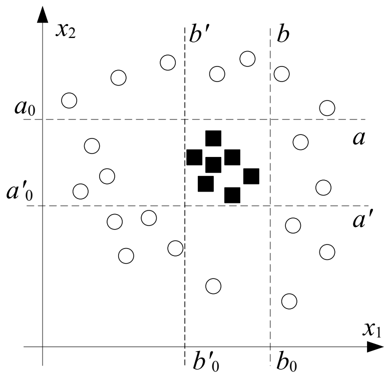
A consistent classifier whose decision regions are associated with the above triangle can be obtained using a feed-forward multi-layer perceptron network with one hidden layer made up of three units (one for each side of the triangle). To set the connection weights it is convenient to associate the "black squares" class with the value 1 of the network output, and to require that the output of each hidden unit equals 1 in the corresponding half-plane containing the black squares: this way, the output unit will have to implement the logical function `AND` (see the figure above on the right, where a, b and c denote the lines corresponding to the sides of the triangle, and the small arrows point toward the regions where the output of the corresponding hidden unit should be 1). Denoting the hidden units by a, b and c, as the corresponding lines, and the intercepts of such lines with the axes as a_0, b_0 and c_0 (see again the figure above on the right), it follows that the activations of the hidden units are given by:

$$a \, : \, -x_2 + a_0 \geq 0$$
$$b \, : \, -x_1 + b_0 \geq 0$$
$$c \, : \, x_1 + x_2 - c_0 \geq 0$$

The corresponding perceptron network is shown below:



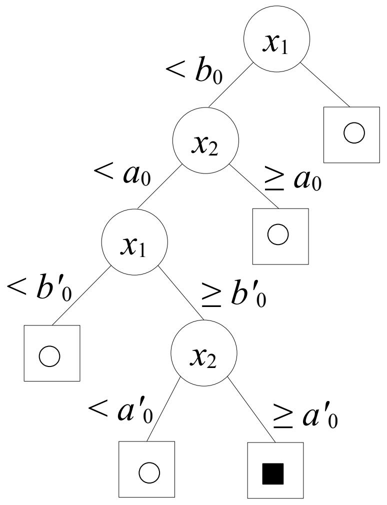
In the case of a decision tree, the class boundaries must be defined using lines *parallel* to the axes. Since the black squares can be separated by the white circles by enclosing them inside a rectangle, at least four lines are necessary: one possible choice of these lines is shown in the figure below.



To define the corresponding decision tree, note that the rectangular region enclosing the black squares can be represented by the following rule:

$$\texttt{IF } x_1 < b_0 \texttt{ AND } x_2 < a_0 \texttt{ AND } x_1 \geq b'_0 \texttt{ AND } x_2 \geq a'_0 \texttt{ THEN class} = \blacksquare$$

Clearly, if *any* of the above conditions is not satisfied, then the corresponding point (x_1, x_2) lies outside the rectangle, and can therefore be labelled as "white circle". This leads to the following decision tree:



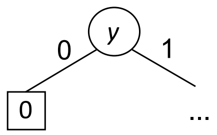
4. (a) The activation of the upper hidden unit is $2x + 2y - 3$, which is non-negative only when $x = 1$ and $y = 1$. Therefore this unit implements the `AND` Boolean function, $x \cdot y$.

   The activation of the lower hidden unit is $-3x + 2y - 3z - 1$, which is non-negative only when $x = z = 0$ and $y = 1$. Accordingly, this unit implements the Boolean function $\overline{x} \cdot y \cdot \overline{z}$.

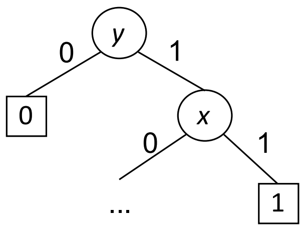
   Finally, denoting as u_1 and u_2 the output values of the hidden units, the activation of the output unit is $4u_1 + 3u_2 - 2$, which is non-negative when at least one among u_1 and u_2 equals 1: this corresponds to the `OR` Boolean function, $u_1 + u_2$.

   The Boolean function represented by the whole perceptron network is therefore $x \cdot y + \overline{x} \cdot y \cdot \overline{z}$.

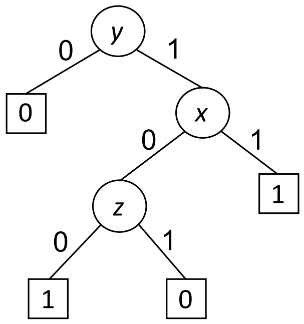
   (b) Since both operands of the `OR` operator in the Boolean function $f = x \cdot y + \overline{x} \cdot y \cdot \overline{z}$ include the term y, it follows that $f = 0$, if $y = 0$. It is therefore convenient to associate the root node of the corresponding decision tree with the variable y:



   Now the fact that $f = 1$ when $y = 1$ and $x = 1$ (corresponding to the first term $x \cdot y$) can be represented by adding an inner node associated with the variable x to the right child of the root node:



   Finally, when $y = 1$ and $x = 0$, the second term $\overline{x} \cdot y \cdot \overline{z}$ implies that $f = 1$ when $z = 0$, whereas $f = 0$ when $z = 1$: this can be represented by adding a "decision stump" sub-tree to the branch corresponding to $y = 1$ and $x = 0$:



5. The activation of the perceptron is:

$$a = w_1 x_1 + w_2 x_2 - w_0 = 1 \times 0.5 - 1 \times 0.5 - 1 \times 0.25 = -0.25 \ .$$

   Since $a < 0$, the output is $y = 0$, and therefore the considered training example is misclassified. This means that the values of the connection weights will be *updated* by the perceptron learning algorithm. The corresponding error function is defined as $E = -t \times a = -t \times (w_1 x_1 + w_2 x_2 - w_0)$. For the considered training example, $E = 0.25$. The weight update rule of the perceptron learning algorithm is:

$$w_i \leftarrow w_i - \frac{\partial E}{\partial w_i} \ . \tag{1}$$

In particular:

$$\frac{\partial E}{\partial w_i} = -tx_i,\ i = 1,2; \quad \frac{\partial E}{\partial w_0} = t\ .$$

The rationale of the above update rule is to *reduce* the value of the error function, which can be achieved by "pushing" the activation toward a higher value, if $a < 0$ and $t = +1$, or toward a lower value, if $a > 0$ and $t = -1$. For the considered values of w_1, w_2, x_1, x_2 and t, it follows that:

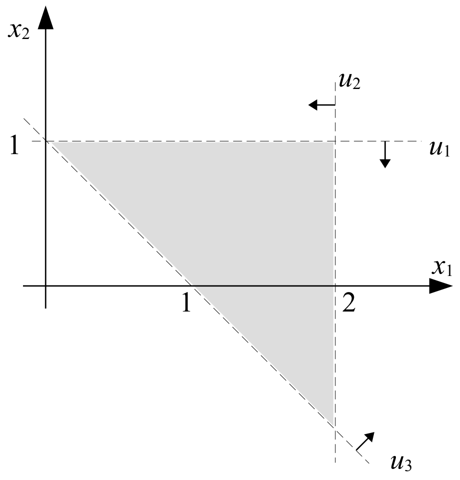
$$\frac{\partial E}{\partial w_0} = t = 1, \quad \frac{\partial E}{\partial w_1} = -tx_1 = -0.5, \quad \frac{\partial E}{\partial w_2} = -tx_2 = -0.5\ .$$

From the above expressions and from Eq. (1) it is easy to see that w_0 will be *decreased*, whereas w_1 and w_2 will be *increased*.

6. The activations of the three hidden units are non-negative (and therefore the corresponding outputs equal 1), when the following inequalities are satisfied:

   - $u_1 :\ -x_2 + 1 \geq 0$,
   - $u_2 :\ -x_1 + 2 \geq 0$,
   - $u_3 :\ x_1 + x_2 - 1 \geq 0$.

   The corresponding regions in the input space are shown by the three dashed lines u_1, u_2 and u_3 in the figure below (the small arrows point toward the half-plane where the outputs of the hidden units is 1).



   It is easy to see that the output unit implements the `AND` Boolean function between the outputs of the hidden units. Therefore the region where the network output equals 1 is the shaded area in the figure above; outside this region the network output equals 0.

7. The decision rule represented by the decision tree for class 1 can be expressed as:

   `IF` $B \geq y$ `AND` $A \geq x$ `AND` $C \leq z$ `THEN` class label $= 1$

   If the above conditions are not satisfied, the class label assigned by the decision tree is 0.

   Consider now the perceptron network. The hidden unit u_1 outputs the value $y_1 = 1$, if $2A - 2x \geq 0$; the hidden unit u_2 outputs the value $y_2 = 1$, if $B - y \geq 0$; the hidden unit u_3 outputs the value $y_3 = 1$, if $-C + z \geq 0$, i.e., if $C - z \leq 0$. The output unit, in turn, outputs the value $y = 1$, if $3y_1 + 3y_2 + 3y_3 - 7 \geq 0$, that is, only if the output of each hidden unit equals 1 (this means that the output unit implements the `AND` Boolean function). Accordingly, the network output is 1, if and only if $2A - 2x \geq 0$ and $B - y \geq 0$ and $C - z \leq 0$, which are the same conditions under which the output of the decision tree is 1: therefore the decision regions of the two classifiers are identical.

8. The attribute `Sky` has three possible values: `sunny`, `overcast` and `rain`. Its discriminant capability, related to the considered node of the decision tree, is computed by ID3 as the conditional entropy of the probability density function $P(Decision|Wind = \text{strong}, Sky)$:

$$H(Decision|Wind = \text{strong}, Sky) = \sum_{s \in \{\text{sunny, overcast, rain}\}} P(Sky = s|Wind = \text{strong}) H(Decision|Wind = \text{strong}, Sky = s) \ . \qquad (2)$$

The entropy of the conditional densities corresponding to each value of s, $H(Decision|Wind = \text{strong}, Sky = s)$, can be estimated as follows.

The node associated with the attribute `Sky` is reached by the six training examples whose attribute `Wind` has the value `strong`, i.e., by training examples 2, 6, 7, 11, 12 and 14. Among them:

- Examples 2 and 11 correspond to `Sky=sunny`, and the corresponding decision is respectively `no` and `yes`. Therefore the density $P(Decision|Wind = \text{strong}, Sky = \text{sunny})$ can be estimated as:
  - $P(Decision = \text{yes}|Wind = \text{strong}, Sky = \text{sunny}) = \frac{1}{2} = 0.5$,
  - $P(Decision = \text{no}|Wind = \text{strong}, Sky = \text{sunny}) = \frac{1}{2} = 0.5$.

  It easily follows that the entropy of the above density function is 1.
- Examples 7 and 12 correspond to `Sky=overcast`, and for both of them the decision is `yes`. Therefore the entropy of the (estimated) density function $P(Decision|Wind = \text{strong}, Sky = \text{overcast})$ is 0.
- Examples 6 and 14 correspond to `Sky=rain`, and for both of them the decision is `no`. Therefore also the entropy of $P(Decision|Wind = \text{strong}, Sky = \text{rain})$ is 0.

The conditional density function $P(Sky = s|Wind = \text{strong})$ can be easily estimated from the frequencies of the corresponding events in the training set: for each value of s it is easy to see that the corresponding relative frequency equals $2/6 = 1/3$ (e.g., among the six examples for which `Wind=strong`, two have `Sky=sunny`). Therefore, from Eq. (2) one obtains:

$$\sum_{s \in \{\text{sunny, overcast, rain}\}} P(Sky = s|Wind = \text{strong}) H(Decision|Wind = \text{strong}, Sky = s) = \frac{1}{3}(1 + 0 + 0) = \frac{1}{3} \ .$$

To compute the gain ratio, we need to compute the information gain. The information gain for the attribute `Sky` in branch $Wind = strong$ is $H(Decision|Wind = \text{strong}) - H(Decision|Wind = \text{strong}, sky)$. Note that the second term is what we have computed before. Whereas we should compute the first term, which is the entropy of the branch strong of the root node wind before adding a further node. In that branch goes: 3 examples for which the decision is No and 3 examples for which the decision is Yes, therefore the entropy of that branch is 1. The information gain is thus $1 - \frac{1}{3} = \frac{3-1}{3} = \frac{2}{3} \simeq 0.66$. This will be the numerator of our gain ratio. Now we should compute the denominator of the gain ratio, namely the split information. If we consider the attribute `Sky`, the split info is thus $\frac{2}{6}log_2(\frac{2}{6}) - \frac{2}{6}log_2(\frac{2}{6}) - \frac{2}{6}log_2(\frac{2}{6}) \simeq 1.585$. The gain ratio is thus $\frac{information_gain}{split_info} \simeq \frac{0.66}{1.585} \simeq 0.416$.

9. The region of the attribute space where the output of the considered perceptron equals 1 corresponds to the half-plane defined by $x_1 + x_2 - 2 \geq 0$.

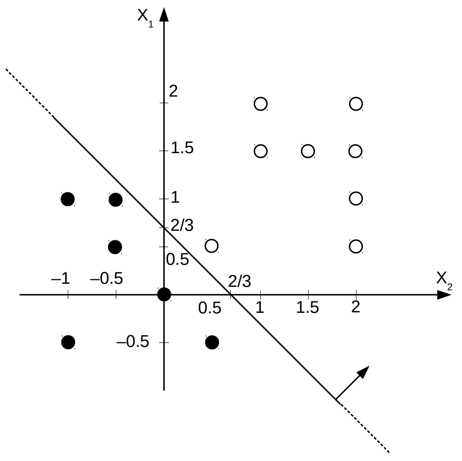
   (a) Considering the expression $x_1 + x_2 - 2$ as the activation of the perceptron, the values of the corresponding connection weights are $w_1 = +1$, $w_2 = +1$ and $w_0 = 2$.

   (b) The value of the error function for the misclassified example $(0.5, 0.5)$, with class label $t = +1$, is given by $E = -t \times (w_1 x_1 + w_2 x_2 - w_0) = -1 \times (0.5 w_1 + 0.5 w_2 - w_0) = +1$.

   (c) The weight update rule is $w_k \leftarrow w_k - \frac{\partial E}{\partial w_k}$, which leads to:

$$
\begin{aligned}
w_1 \leftarrow w_1 - \frac{\partial\,(0.5w_1 + 0.5w_2 - w_0)}{\partial w_1} &= 1 + 0.5 = 1.5\ , \\
w_2 \leftarrow w_2 - \frac{\partial\,(0.5w_1 + 0.5w_2 - w_0)}{\partial w_2} &= 1 + 0.5 = 1.5\ , \\
w_0 \leftarrow w_0 - \frac{\partial\,(0.5w_1 + 0.5w_2 - w_0)}{\partial w_0} &= 2 - 1 = 1\ .
\end{aligned}
$$

   The activation of the perceptron becomes now $1.5x_1 + 1.5x_2 - 1$, and the corresponding decision boundary is shown in the figure below. Now the previously misclassified training example gets correctly classified, and all the other ones remain correctly classified.

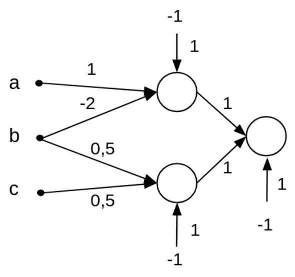


10. The Karnaugh map can be used to simplify the function:

		AB			
		00	01	11	10
C	0	0	0	0	$\boxed{1}$
	1	0	$\boxed{1}$	$\boxed{1}$	$\boxed{1}$

The ones are gruped in the biggest squares possible of an area with a power of two so that all the ones are covered by at least one square. For each group we consider the variables that do not change their value. The simplified function is: $a \cdot \overline{b} + b \cdot c$

we can thus create a network with two nodes in the hidden layer and one node in the output layer. $b \cdot c$ is an and function. in $a \cdot \overline{b}$ a should favor the activation, whereas b should favor the not activation. The network can thus be the following:


-1
1
a
1
-2
1
b
0,5
1
c
0,5
1
1
-1
-1