UNSW

COMP9444 期末班

期末班第一节课讲解

讲师：文杰学长　　**时长：2 小时**

本周内容预览：

1. Perceptron 模型复习
2. Linear Separability
3. Perceptron Learning Algorithm
4. Multi Layer Perceptrons
5. Gradient Descent for Neural Networks
6. 对应习题讲解
7. 往年真题练习

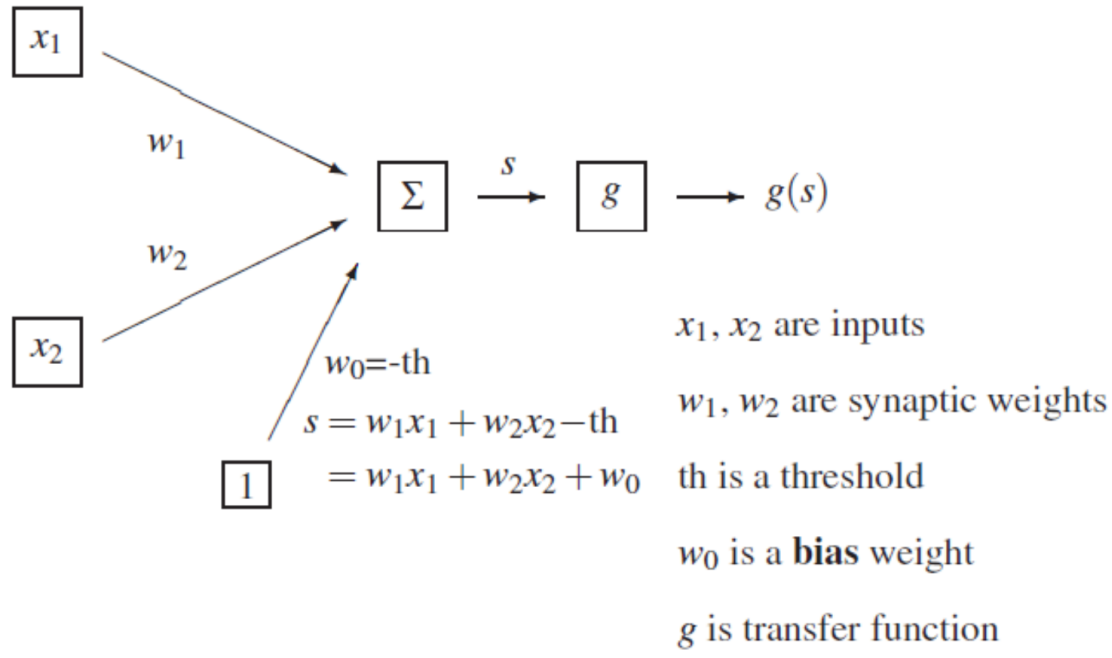
一、 Perceptron

In 1943, McCulloch and Pitts published a simplified mathematical model of a neuron. Each input x_i is multiplied by a corresponding **weight** w_i and the results are added together. An extra constant w_0 called the **bias** is also added.

$$s = w_0 + \sum_i w_i x_i$$

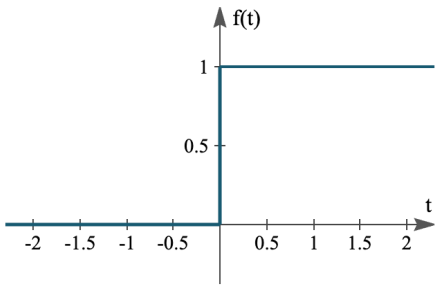
Finally, a non-linear **transfer function** is applied to this linear combination to produce the output of the neuron.

$$z = g(s) = g(w_0 + \sum_i w_i x_i)$$



Transfer function

The original transfer function used by McCulloch and Pitts was a discontinuous step function (also called the Heaviside function).

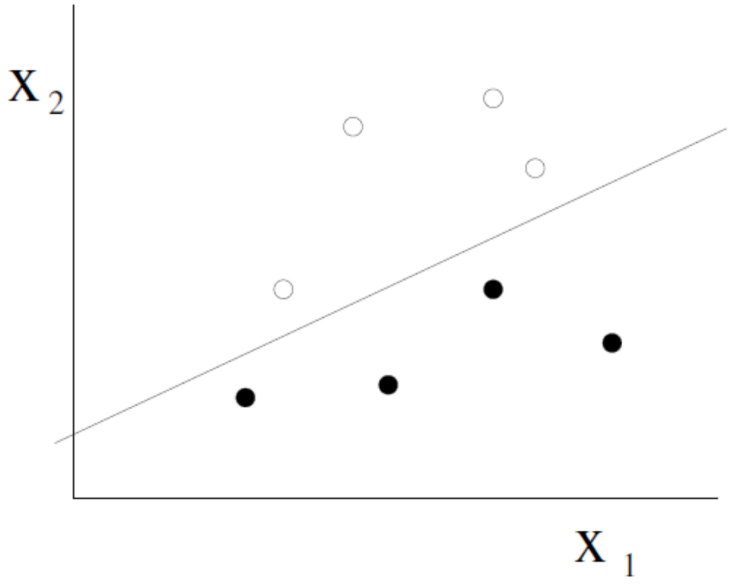


$$g(s) = \begin{cases} 1, & \text{if} \quad s \geq 0 \\ 0, & \text{if} \quad s < 0 \end{cases}$$

An artificial neuron with this transfer function is called a **Perceptron**.

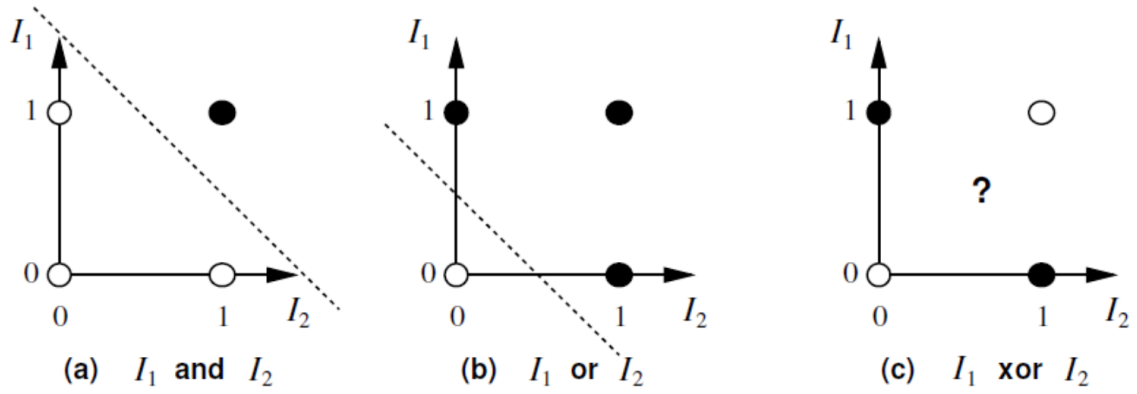
二、 Linear Separability

The weights and bias of a Perceptron determine a hyperplane which divides the space of inputs into two regions. For inputs on one side of this hyperplane, the output will be 0; for inputs on the other side, the output will be 1. Functions which can be computed in this way are called **Linearly Separable**.



Logical Functions

If we adopt the convention that $0 = \text{False}$ and $1 = \text{True}$, many simple logical functions like AND, OR and NOT become linearly separable. We can use our knowledge of plane geometry to build Perceptrons to compute these functions. This diagram shows the AND an (inclusive) OR functions, which can be computed by Perceptrons, as well as the Exclusive OR (XOR) which cannot be computed by a Perceptron, because it is not linearly separable.



These are the weights for Perceptrons to compute the AND and OR function, and the opposite of OR, sometimes called NOR which stands for "Not OR".

$$\begin{array}{lll} \text{AND} & w_1 = w_2 = 1.0, & w_0 = -1.5 \\ \text{OR} & w_1 = w_2 = 1.0, & w_0 = -0.5 \\ \text{NOR} & w_1 = w_2 = -1.0, & w_0 = 0.5 \end{array}$$

三、 Perceptron Learning Algorithm

The algorithm works as follows. The bias w_0 and weights $\{w_i\}$ of the Perceptron are at first initialised to (small) random values. Then, the training items are presented one at a time, and appropriate changes are made to w_0 and $\{w_i\}$.

Recall that the output of the Perceptron is $g(s)$ where $s = w_0 + \sum_i w_i\, x_i$.

Suppose an input $\{x_i\}$ is presented for which the target output is 1 but the actual output of the Perceptron is 0. In this case, we want to adjust w_0 and $\{w_i\}$ in a way that will make s larger for this particular choice of $\{x_i\}$. We do this by making these adjustments, where $\eta > 0$ is a constant called the **learning rate**:

$$\begin{aligned} w_0 &\leftarrow w_0 + \eta \\ w_i &\leftarrow w_i + \eta\, x_i \\ \text{Consequently,} \quad s &\leftarrow s + \eta(1 + \sum_k x_k^2) \end{aligned}$$

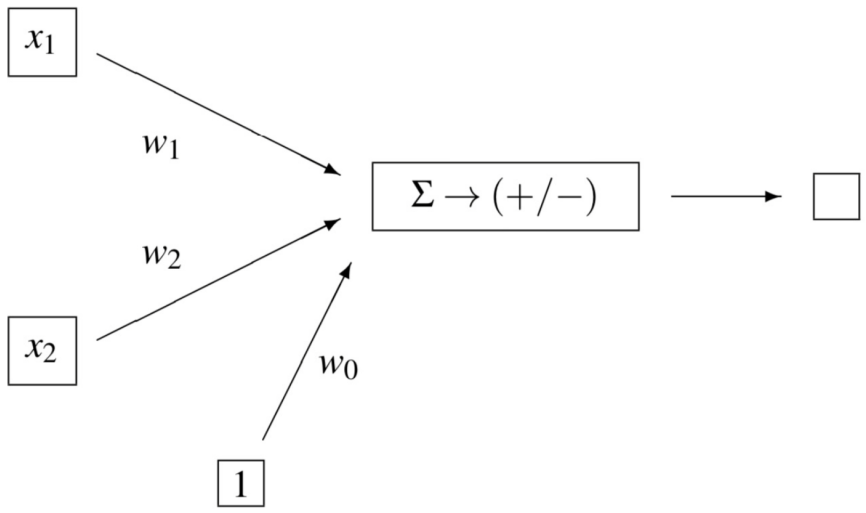
If the target output is 0 but the actual output is 1, we do the opposite, namely:

$$\begin{aligned} w_0 &\leftarrow w_0 - \eta \\ w_i &\leftarrow w_i - \eta\, x_i \\ \text{Consequently,} \quad s &\leftarrow s - \eta(1 + \sum_k x_k^2) \end{aligned}$$

If the actual output is equal to the target output, no change is made to the weights. We may need to cycle through the data several times but, if we reach a point where all the data are correctly classified, the algorithm terminates.

Rosenblatt proved that this algorithm will always learn to classify the training data successfully, provided the data are linearly separable.

Perceptron Learning Example:



Exercise 1:

Question 1

Construct by hand a Perceptron which correctly classifies the following data; use your knowledge of plane geometry to choose appropriate values for the weights w_0, w_1 and w_2.

Training example	x_1	x_2	Class
a.	0	1	-1
b.	2	0	-1
c.	1	1	$+1$

Identify the equation of the line and the weights of the Perceptron (including bias).

Exercise 2:

Question 2

✓ SUBMITTED

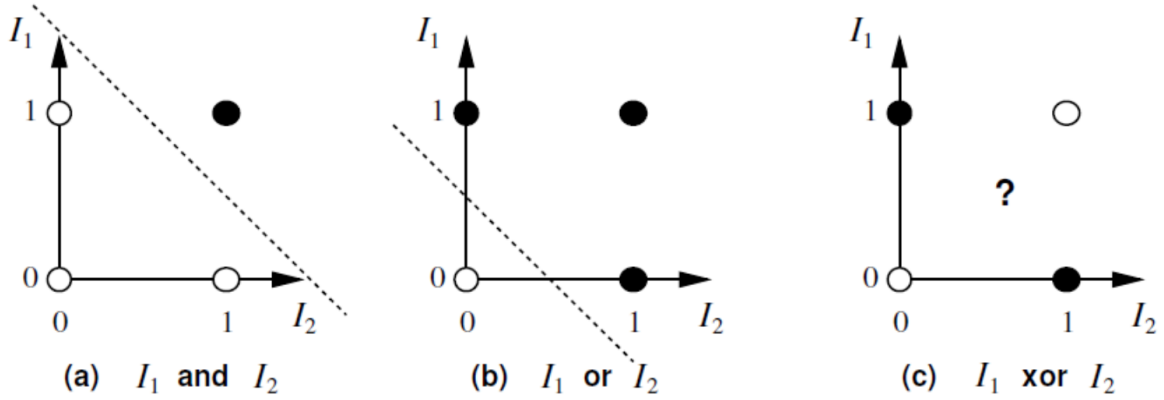
Demonstrate the Perceptron Learning Algorithm on the above data, using a learning rate of 1.0 and initial weight values of

$$\begin{aligned} w_0 &= -1.5 \\ w_1 &= 0 \\ w_2 &= 2 \end{aligned}$$

Continue until all items are correctly classified.

四、 Multi Layer Perceptrons

The main problem with Perceptrons is that many useful functions are not linearly separable.

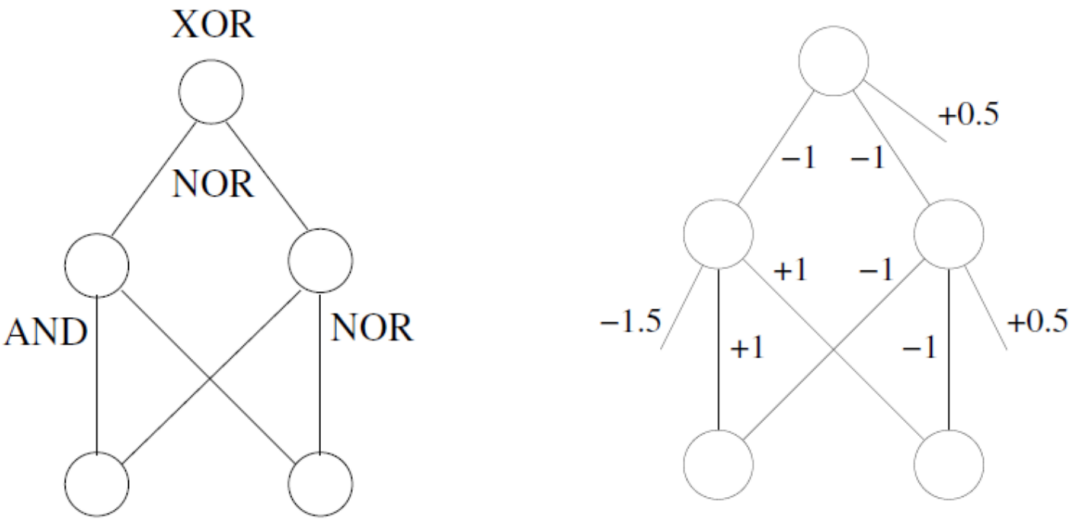


The simplest example of a logical function that is not linearly separable is the Exclusive OR or XOR function. Some languages have distinct words for inclusive and exclusive OR — for example, Latin has "aut" and "vel". In other languages, like English, this distinction needs to be inferred from the context. Consider these two sentences:

1. "All of his movies are either too long or too silly." [They could be both too long and too silly]
2. "Either you give me the money, or I will punch you in the face." [We understand from the context that one of these things will happen, but not both]

Multi Layer Perceptrons

One solution to this problem is to rewrite XOR in terms of linearly separable functions like AND, OR, NOR and then arrange several Perceptrons into a network which combines them in the same way. For example, x_1 XOR x_2 can be written as: (x_1 AND x_2) NOR (x_1 NOR x_2). We can therefore arrange three perceptrons in the following way to compute XOR:



五、 Gradient Descent for Neural Networks

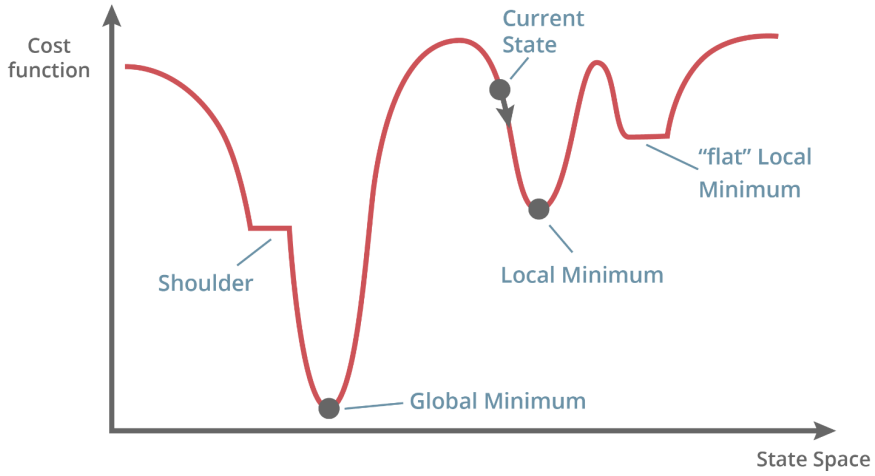
We saw previously how a Multi Layer Perceptron can be built to implement any logical function. However, normally we need to deal with raw data rather than an explicit logical expression. What we really want is a method, analogous to the perceptron learning algorithm, which can learn the weights of a neural network, based on a set of training items.

As early as the 1960's, engineers understood how to use Gradient Descent to optimize over a family of functions that are continuous and differentiable. The basic idea is this:

We define an error function or **loss** function E to be (half) the sum over all input items of the square of the difference between the actual output and target output:

$$\mathrm{E} = \frac{1}{2}\sum_i (z_i - t_i)^2$$

If we think of E as height, this defines an error landscape on the weight space. The aim is to find a set of weights for which E is very low.



If the functions involved are smooth, we can use multi-variable calculus to adjust the weights in such a way as to take us in the steepest downhill direction.

$$w \leftarrow w - \eta \frac{\partial E}{\partial w}$$

Parameter η is called the **learning rate**.

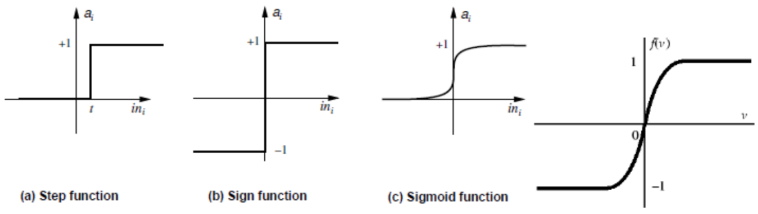
Continuous Activation Functions

The key idea is to replace the (discontinuous) step function with a differentiable function, such as the sigmoid:

$$g(s) = \frac{1}{1+e^{-s}}$$

or hyperbolic tangent:

$$g(s) = \tanh(s) = \frac{e^s - e^{-s}}{e^s + e^{-s}} = 2\left(\frac{1}{1+e^{-2s}}\right) - 1$$



Exercise 3:

Conjunctive Normal Form

Before beginning this exercise, we note that any logical function can be converted into Conjunctive Normal Form (CNF), meaning it is a conjunction of terms where each term is a disjunction of (possibly negated) literals. This is an example of an expression in CNF:

$$(\mathrm{A} \vee \mathrm{B}) \wedge (\neg \mathrm{B} \vee \mathrm{C} \vee \neg \mathrm{D}) \wedge (\mathrm{D} \vee \neg \mathrm{E})$$

Computing any Logical Function with a two-layer network

Assuming False$=0$ and True$=1$, explain how each of the following could be constructed.

You should include the bias for each node, and the value of all the weights (input-to-output or input-to-hidden, hidden-to-output, as appropriate).

Question 1

Perceptron to compute the OR function of m inputs

Question 2

Perceptron to compute the AND function of n inputs.

Question 3 ✓ SUBMITTED

Two-layer Neural Network to compute the function $(A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E)$

With reference to this example, explain how a two-layer neural network could be constructed to compute any (given) logical expression, assuming it is written in Conjunctive Normal Form.

Hint: first consider how to construct a Perceptron to compute the OR function of m inputs, with k of the m inputs negated.

Part B: Numerical (Questions 16-21)

Part B: Question 16 (3 marks)

Consider a perceptron whose output is given by $h(w_0 + w_1x_1 + w_2x_2)$ where x_1, x_2 are inputs and $h()$ is the Heaviside (step) function. The current weights are $w_0 = -0.5, w_1 = -1, w_2 = -2$.

Training Example	x_1	x_2	Class
a.	2	1	+
b.	–2	2	+
c.	–1	–1	–

Suppose the Perceptron Learning Algorithm is applied to the current weights, using a learning rate of $\eta = 1.0$. We consider only the first step, where **item (a)** is learned. The new values for w_0, w_1 and w_2 after learning **item (a) only** will be:

i. (1 mark) w_0 :

ii. (1 mark) w_1 :

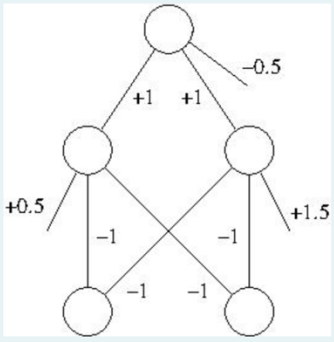
iii. (1 mark) w_2 :

Question 17
Answer saved
Marked out of 2.00
Flag question

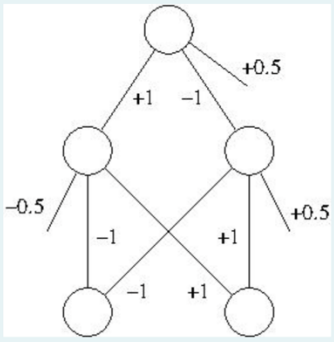
If 0=FALSE and 1=TRUE, which of these networks (with threshold activations at both the hidden and output layer) correctly computes the XOR function of two inputs?

Select one:

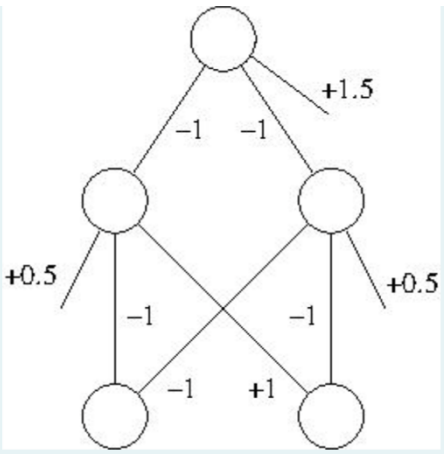
a.



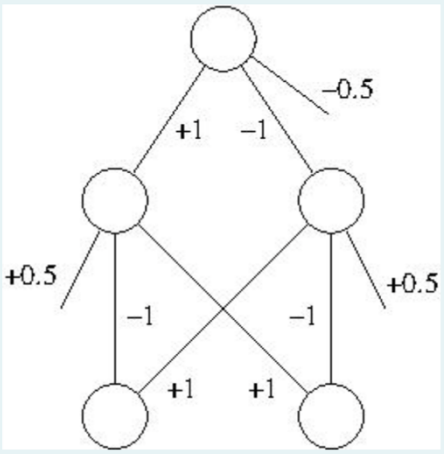
b.



c.



d.



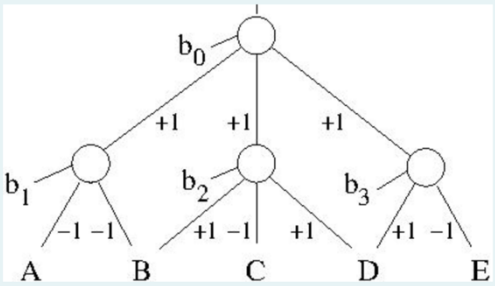
Question **14**

Answer saved

Marked out of 2.00

Flag question

Consider the following multi-layer perceptron, using the threshold activation function, and assume that TRUE is represented by 1; FALSE by 0.



For which values of the biases b_0, b_1, b_2 and b_3 would this network compute the logical function

(¬A ∨ ¬B) ∧ (B ∨ ¬C ∨ D) ∧ (D ∨ ¬E)

* b_0= −2.5
* b_1= +1.5
* b_2= +0.5
* b_3= +0.5

Question **13**

Answer saved

Marked out of 2.00

Flag question

Consider a Perceptron whose output is given by $h(w_0+w_1x_1+w_2x_2)$, where x_1, x_2 are inputs and $h()$ is the Heaviside (step) function.

Assume this Perceptron is being trained on the data in the following table, and that the current values of the weights are w_0= 0.5, w_1= 1 and w_2= -2.

Training Example	x_1	x_2	Class
(a)	−2	2	Pos
(b)	2	1	Neg
(c)	−1	−1	Neg

If the Perceptron Learning Rule is applied to the current weights, using training item (a) and a learning rate of η=1.0, the new values for w_0, w_1 and w_2 at the end of this training step will be:

* w_0: +1.5
* w_1: −1
* w_2: 0