INT301 W2 Notes

Data

A set of data records (also called examples, instances or cases) described by:

- kattributes: A1, A2, ... Ak，如下图中：age，Has_Job，Own_House，Credit_Rating;
- a class: Each example is labelled with a pre-defined class，如下图 Class 中：Yes，和 No。

ID	Age	Has_Job	Own_House	Credit_Rating	Class
1	young	false	false	fair	No
2	young	false	false	good	No
3	young	true	false	good	Yes
4	young	true	true	fair	Yes
5	young	false	false	fair	No
6	middle	false	false	fair	No
7	middle	false	false	good	No
8	middle	true	true	good	Yes
9	middle	false	true	excellent	Yes
10	middle	false	true	excellent	Yes
11	old	false	true	excellent	Yes
12	old	false	true	good	Yes
13	old	true	false	good	Yes
14	old	true	false	excellent	Yes
15	old	false	false	fair	No

Fundamental assumption of learning

假设：training examples 的分布与 test examples 的分布相同（包括以后的 unseen examples）

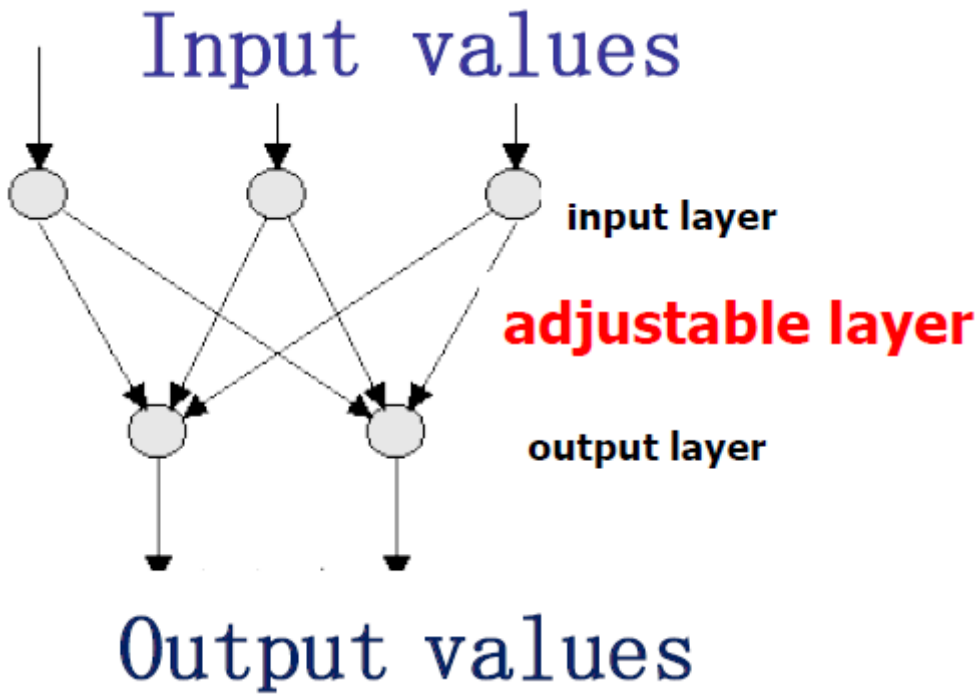
对上面假设的强烈违反显然会导致分类准确性差。为了在 test data 上实现良好的准确性，training examples 必须充分代表 test data。

Perceptron

感知器（perceptron）是神经网络，并可以使用错误校正规则（error-correcting rule）以"经验"改变。

注：上面这个意思就是，在网络出现错误的时候（比如分类错误），网络的权重就会改变（就是根据 loss 更新权重）。

最简单的感知器结构包括两层理想化的"神经元 (neurons)"，我们称之为 'units' of the network。这两层是：one layer of input units, and one layer of output units。



processing elements of the perceptron are the **abstract neurons**.

The total input to the output unit j is:

$$S_j = \sum_{i=0}^{n} w_{ji} a_i$$

a_i : input value from the i-th input unit

w_{ji} : the weight of connection between i-th input and j-th output units

注意：上式中，i 是从 0 开始的。这是由于 input 层有一个 special bias input a_0。a_0 的输入恒为 1，但是它的 w_{j0} 会和其他权重一样变化。这个 special bias input a_0 相当于 b，后面会介绍。

和 MP neuron 类似，output unit j 的输出 X_j 也有一个阈值。

X_j is defined by the unit's threshold activation function.

$$X_j = f(S_j) = \begin{cases} 1, S_j \geq \theta_j \\ 0, S_j < \theta_j \end{cases}$$

最后，output 层的所有输出 $X = \{X_0, X_1, \ldots, X_n\}$ 叫做 **output vector** of the network。

Perceptron Training

Every processing element computes an output according its state and threshold:

$$S_j = \sum_{i=0}^{n} w_{ji} a_i$$

$$X_j = f(S_j) = \begin{cases} 1, S_j \geq \theta_j \\ 0, S_j < \theta_j \end{cases}$$

$$e_j = (t_j - X_j)$$

注：e_j 是 error，即 target output (期望) 和 instant output (预测) 的差异。

Perceptron Updating of the weights

在计算了每一个 output units 的 error 后：

$$e_j = (t_j - X_j)$$

现在需要对 weights 进行更新：

$$w_{ji}^{\text{new}} = w_{ji}^{\text{old}} + \Delta w_{ji}$$

Perceptron learning rule

where

$$\Delta w_{ji} = C e_j a_i = C(t_j - X_j) a_i$$

这也被称为 **delta rule**：

$$\Delta w = \text{learning rate} \times (\text{teacher} - \text{output}) \times \text{input}$$

举个例子：

Define our “features”:

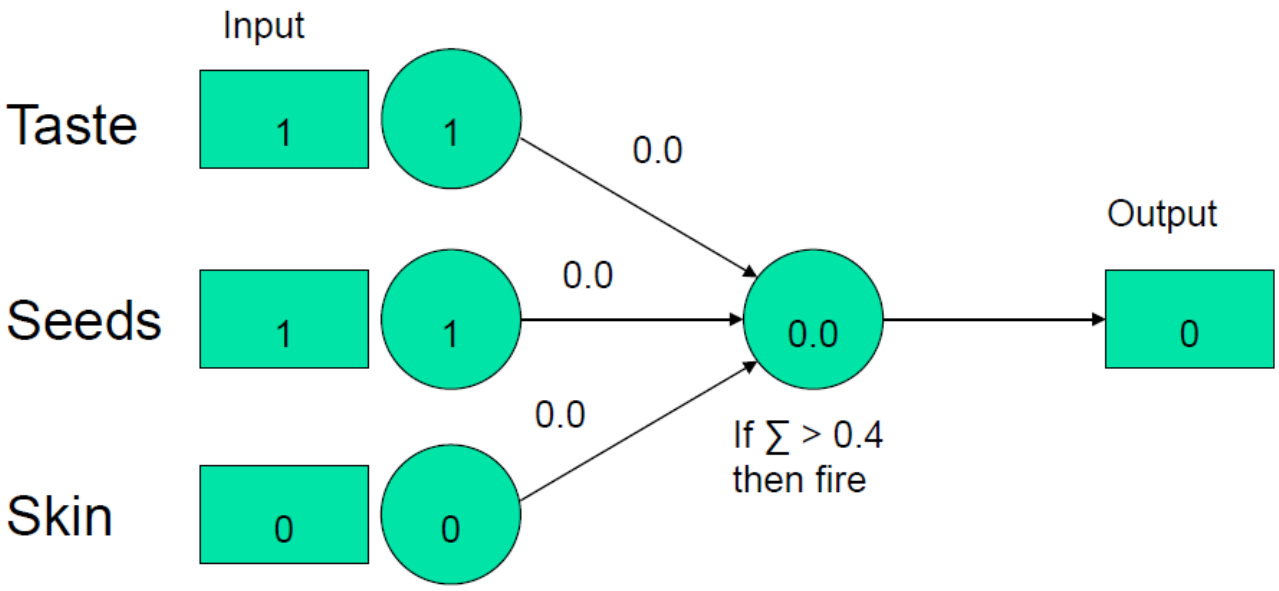
Taste	Sweet = 1, Not_Sweet = 0
Seeds	Edible = 1, Not_Edible = 0
Skin	Edible = 1, Not_Edible = 0

For output:

Good_Fruit = 1

Not_Good_Fruit = 0

Show it a banana:



设 learning rate C = 0.25，banana 的 label = 1。

Feature:	Learning rate:	(teacher - output):	Input:	Δw
taste	0.25	1	1	+0.25
seeds	0.25	1	1	+0.25
skin	0.25	1	0	0

The algorithm **converges (**收敛**)** to the correct classification, if:

- the training data is **linearly separable**;
- and the learning rate is sufficiently small,usually set below 1, which determines the **amount of correction made in a single iteration (**单次迭代中的矫正次数**)**.

对于任何 linearly separable 的数据集，learning rule 保证在有限数量的步骤中找到解决方案。

Network Performance for Perceptron

The network performance during training session can be measured by a *root-mean-square (RMS) error value.*

$$RMS = \sqrt{\frac{\sum_{p=0}^{n_p}\sum_{j=0}^{n_o} e_{jp}^2}{n_p n_o}} = \sqrt{\frac{\sum_{p=0}^{n_p}\sum_{j=0}^{n_o} (t_{jp} - X_{jp})^2}{n_p n_o}}$$

where

n_p is the number of patterns in the training set and
n_o is the number of units in the output layer

由于 target output 的值 t_{jp}，和 n_p，n_o 的数量是常数，因此 RMS error 是一个只关于 X_{jp} 的函数。

而 instant output X_{jp} 是关于 input 值 a_{ip} 和权重 W_{ji} 的函数，a_{ip} 也是常数，因此:

$$X_{jp} = f(S_{jp} = \sum_{i=0}^{n_i} w_{ji} a_{ip}) = \tilde{f}(w_{ji}, a_{ip}),$$

RMS error 也是一个只关于连接权重 W_{ji} 的函数。

注：a_{ip} 指第 i 个输入的第 p 个 pattern。

网络的最佳性能与 RMS error 的最小值正相关，我们调整连接的权重以达到最小化 RMS error。

RMS on the Training Data

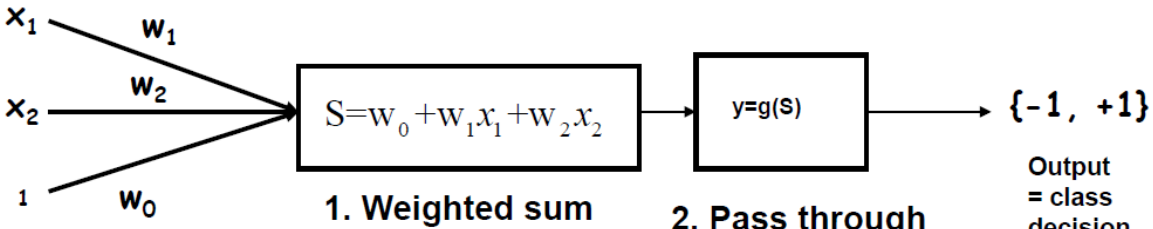
$$RMS^{training} = \sqrt{\frac{\sum_{p=0}^{n_p}\sum_{j=0}^{n_o}(t_{jp} - X_{jp}^{trained})^2}{n_p n_o}}$$

RMS on the Testing Data

$$RMS^{testing} = \sqrt{\frac{\sum_{p=0}^{n_p}\sum_{j=0}^{n_o}(t_{jp} - X_{jp}^{predicted})^2}{n_p n_o}}$$

Perceptron As a Classifier

For *d*-dimensional data, perceptron consists of d-weights, a bias, and a thresholding activation function. For 2D data example，we have:



View the bias as another weight from an input which is constantly on

1. Weighted sum of the inputs

2. Pass through Heaviside function:
T(S)= -1 if S < 0
T(S)= 1 if S>= 0

If we group the weights as a vector w , the net output y can be expressed as:

$$y = g(\mathrm{w} \cdot \mathrm{x} + \mathrm{w}_0)$$

14

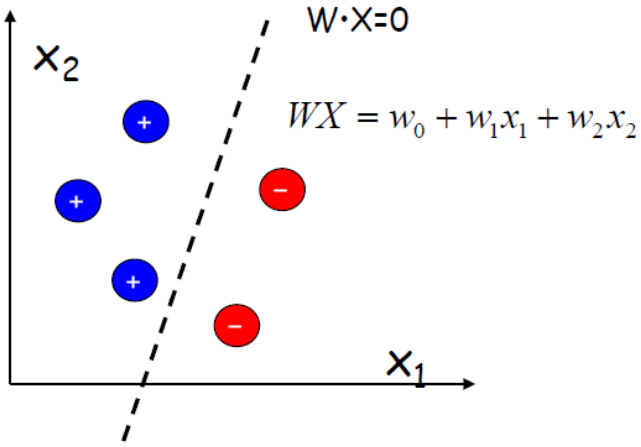
注：w_0 和之前的 a_0 一样，就是 b。

- A perceptron training is to compute weight vector:

$$\mathrm{W} = [\mathrm{w}_0, \mathrm{w}_1, \mathrm{w}_2, \ldots\ldots, \mathrm{w}_p]$$

to correctly classify all the training examples.

E.g.,
consider when *p*=2



W·X is a **hyperplane**, which in 2d is a straight line.

1

Neural Network as Classifier

对于二分类，可以把网络的 output 当作一个 discriminant function y(x, w)（判断函数），当 x 在 class 1，y(x, w) = 1；当 x 在 class 2，y(x, w) = -1。

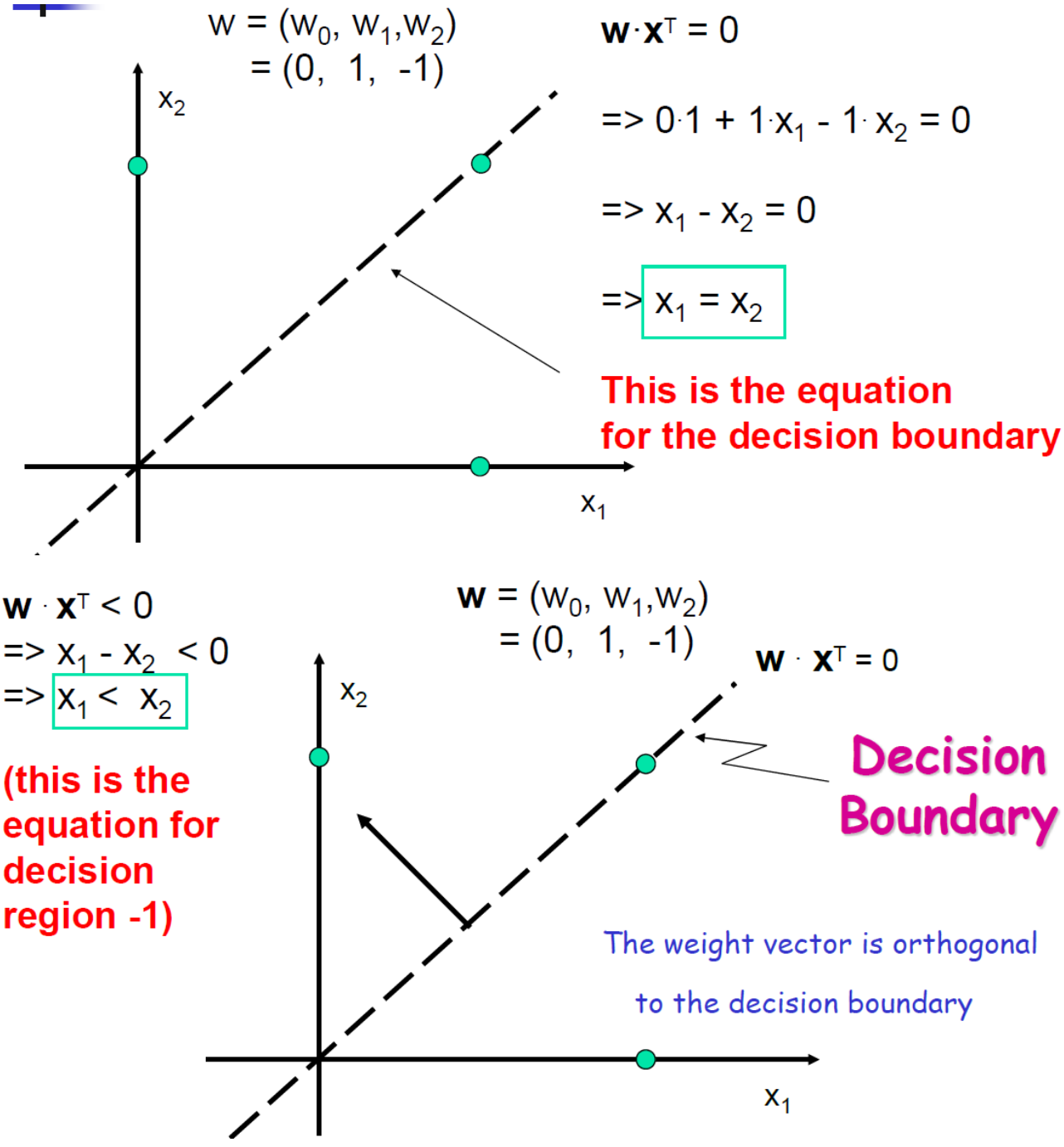
对于 m 个 classes，一个分类器（classifier）会把 feature space 分成 m 个 decision regions（决策区域）：

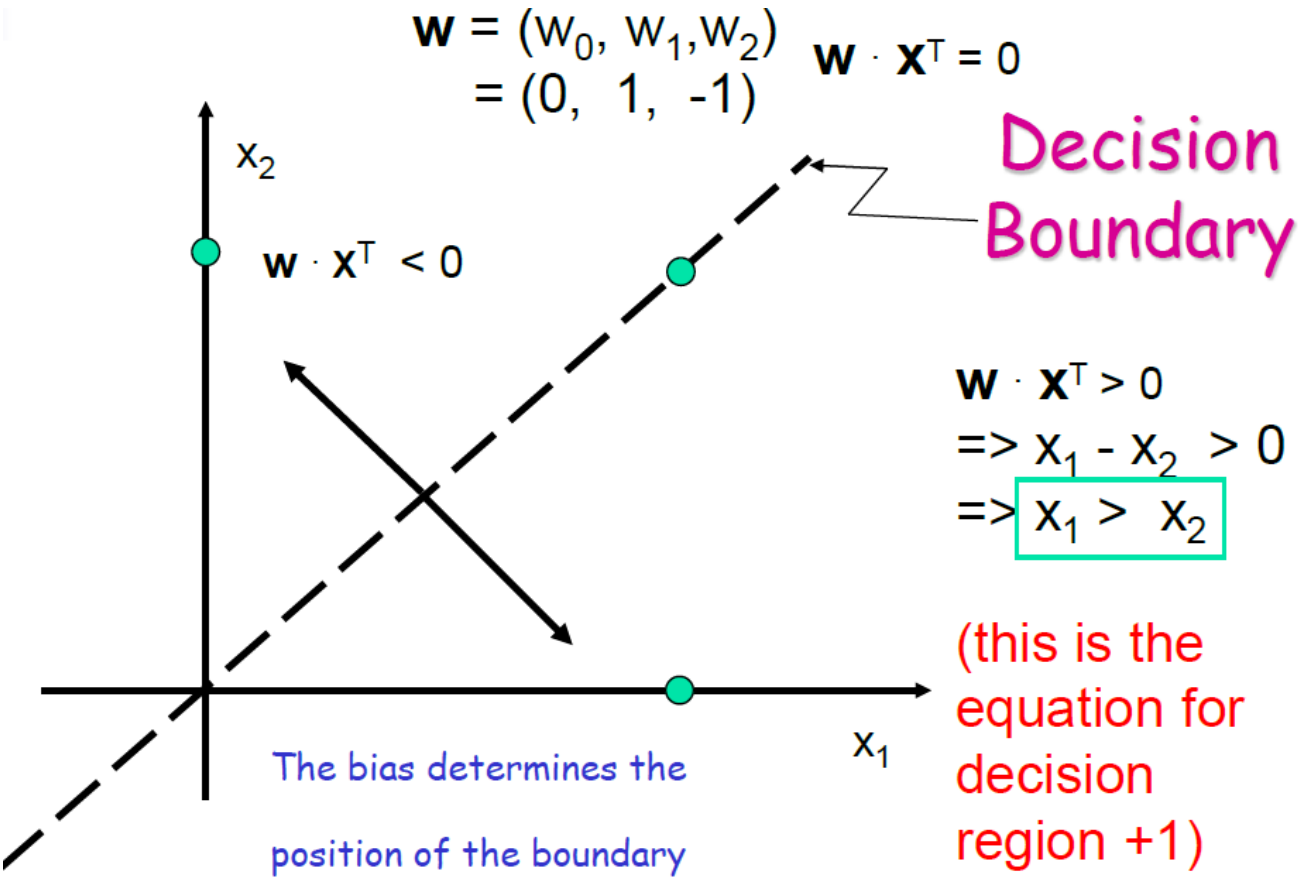
- 分割 class 的 line or curve 叫 decision boundary
- In more than 2 dimensions, this is a hyperplane

The equation of the hyperplane is $w \cdot x^T = 0$.

Example of Perceptron Decision Boundary

Decision surface is the surface at which the output of the unit is precisely **equal to the threshold**, i.e. $\sum w_i x_i = \theta$.





Linear Separability Problem

如果两个类的 patterns 可以通过以线性方程表示的 decision boundary 来划分:

$$b + \sum_{i=1}^{n} x_i w_i = 0$$

那么，可以说它们是 linearly separable 的（反之，则是 linearly inseparable），感知器可以正确分类任何 patterns。

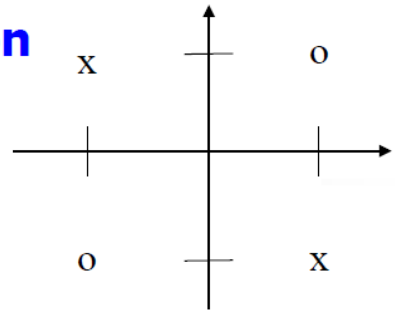
NOTE：如果没有 bias（即 b），hyperplane 会被迫和 origin（原点）相交。（这里的 b 和函数 y = ax + b 里的一样，就是让 hyperplane 可以移动）

Examples of linearly inseparable classes

- **Logical XOR (exclusive OR) function**

patterns (bipolar) decision boundary

x_1	x_2	y
-1	-1	-1
-1	1	1
1	-1	1
1	1	-1



x: class I (y = 1)
o: class II (y = -1)

No line can separate these two classes, as can be seen from the fact that the following linear inequality system has no solution

$$\begin{cases} b - w_1 - w_2 < 0 & (1) \\ b - w_1 + w_2 \geq 0 & (2) \\ b + w_1 - w_2 \geq 0 & (3) \\ b + w_1 + w_2 < 0 & (4) \end{cases}$$

because we have b < 0 from (1) + (4),
and b >= 0 from (2) + (3), which is a contradiction

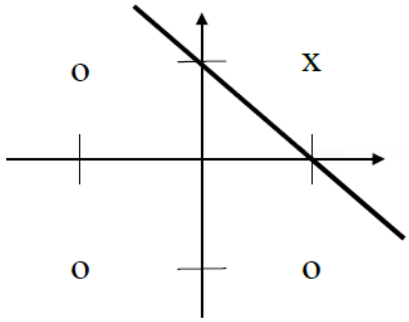
- Logical **AND** function

patterns (bipolar) decision boundary

x_1	x_2	y
-1	-1	-1
-1	1	-1
1	-1	-1
1	1	1

$w_1 = 1$
$w_2 = 1$
$b = -1$
$\theta = 0$

$-1 + x_1 + x_2 = 0$



x: class I (y = 1)
o: class II (y = -1)

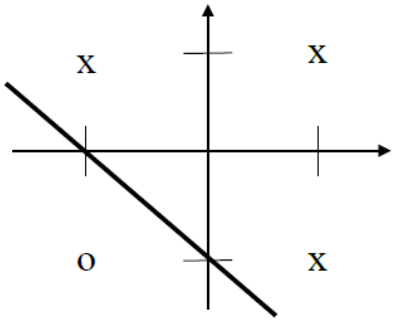
-Logical **OR** function

patterns (bipolar) decision boundary

x_1	x_2	y
-1	-1	-1
-1	1	1
1	-1	1
1	1	1

$w_1 = 1$
$w_2 = 1$
$b = 1$
$\theta = 0$

$1 + x_1 + x_2 = 0$



x: class I (y = 1)
o: class II (y = -1)