



第二讲：生物神经元与感知机

Biological Neurons, McCulloch Pitts Neuron, Perceptrons, Multilayer Perceptrons
(MLPs)

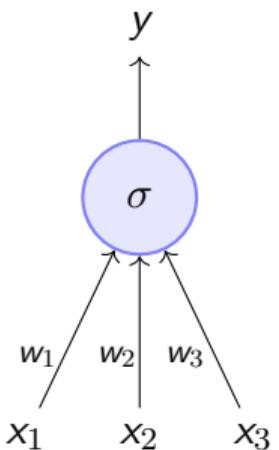
张盛平

s.zhang@hit.edu.cn

计算学部
哈尔滨工业大学

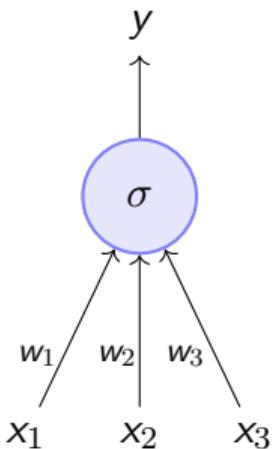
2021 年秋季学期

Biological Neurons (生物神经元)



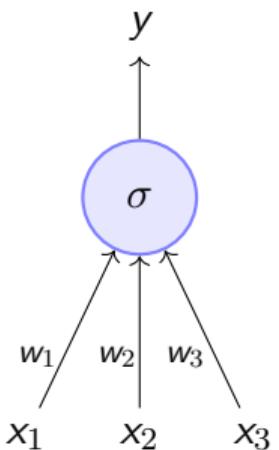
- 深度神经网络的最基本单元是人工神经元 (*artificial neuron*)

Artificial Neuron (人工神经元)



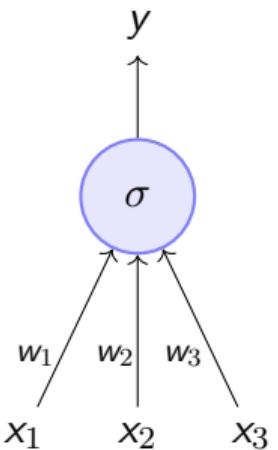
Artificial Neuron (人工神经元)

- 深度神经网络的最基本单元是人工神经元 (*artificial neuron*)
- 为什么称它为神经元? 它的设计灵感来自哪里?



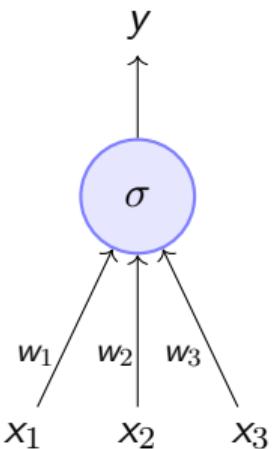
Artificial Neuron (人工神经元)

- 深度神经网络的最基本单元是人工神经元 (*artificial neuron*)
- 为什么称它为神经元? 它的设计灵感来自哪里?
- 灵感来自于生物学, 特别是来自我们的大脑 (*brain*)



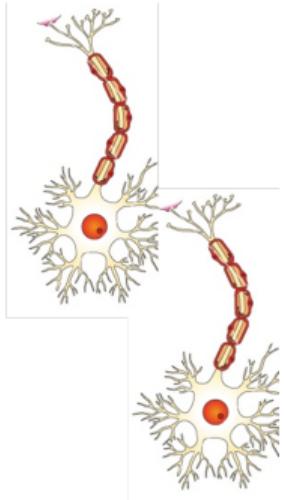
Artificial Neuron (人工神经元)

- 深度神经网络的最基本单元是人工神经元 (*artificial neuron*)
- 为什么称它为神经元? 它的设计灵感来自哪里?
- 灵感来自于生物学, 特别是来自我们的大脑 (*brain*)
- $biological\ neurons = neural\ cells = neural\ processing\ units$



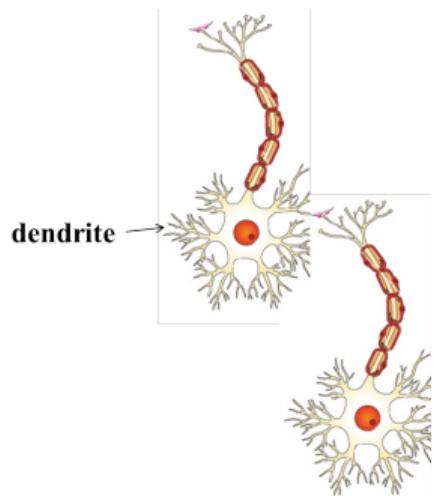
Artificial Neuron (人工神经元)

- 深度神经网络的最基本单元是人工神经元 (*artificial neuron*)
- 为什么称它为神经元? 它的设计灵感来自哪里?
- 灵感来自于生物学, 特别是来自我们的大脑 (*brain*)
- $\text{biological neurons} = \text{neural cells} = \text{neural processing units}$
- 我们将首先看看生物神经元长什么样...



Biological Neurons*

*Image adapted from
<https://cdn.vectorstock.com/i/composite/12,25/neuron-cell-vector-81225.jpg>

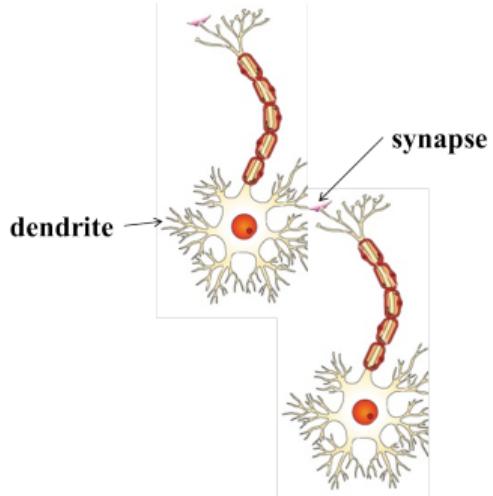


Biological Neurons*

- 树突 (dendrite) : 从其他神经元接收信号

*Image adapted from

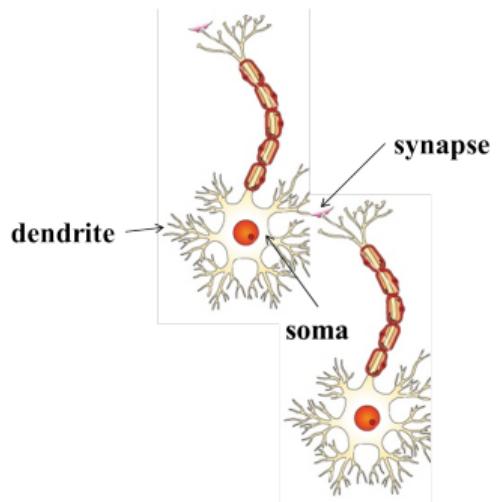
<https://cdn.vectorstock.com/i/composite/12,25/neuron-cell-vector-81225.jpg>



Biological Neurons*

*Image adapted from
<https://cdn.vectorstock.com/i/composite/12,25/neuron-cell-vector-81225.jpg>

- **树突 (dendrite)**：从其他神经元接收信号
- **突触 (synapse)**：一个神经元的激励传到另一个神经元相互接触的结构

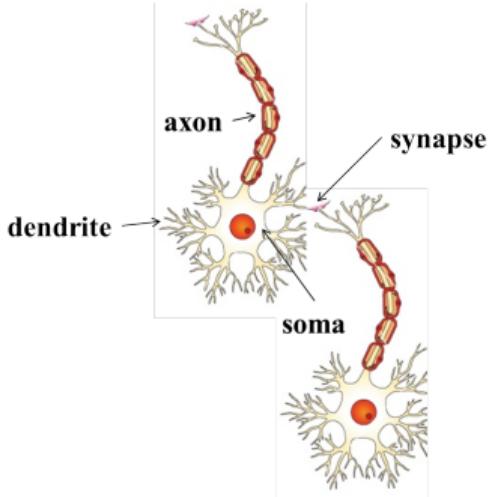


Biological Neurons*

- 树突 (dendrite)：从其他神经元接收信号
- 突触 (synapse)：一个神经元的激励传到另一个神经元相互接触的结构
- 胞体 (soma)：处理信息

*Image adapted from

<https://cdn.vectorstock.com/i/composite/12,25/neuron-cell-vector-81225.jpg>



Biological Neurons*

- **树突 (dendrite)**：从其他神经元接收信号
- **突触 (synapse)**：一个神经元的激励传到另一个神经元相互接触的结构
- **胞体 (soma)**：处理信息
- **轴突 (axon)**：传输神经元的输出

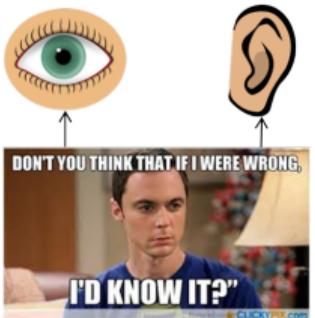
*Image adapted from

<https://cdn.vectorstock.com/i/composite/12,25/neuron-cell-vector-81225.jpg>

- 让我们借助一个卡通的例子来阐述神经元是如何工作的

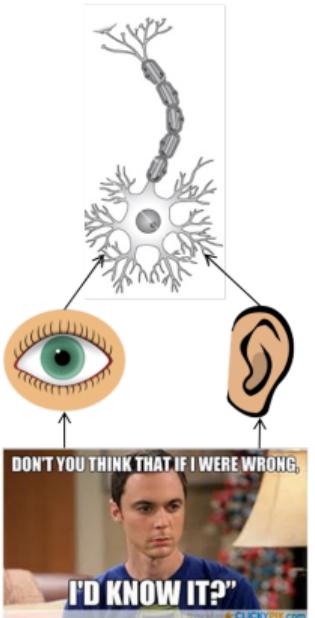


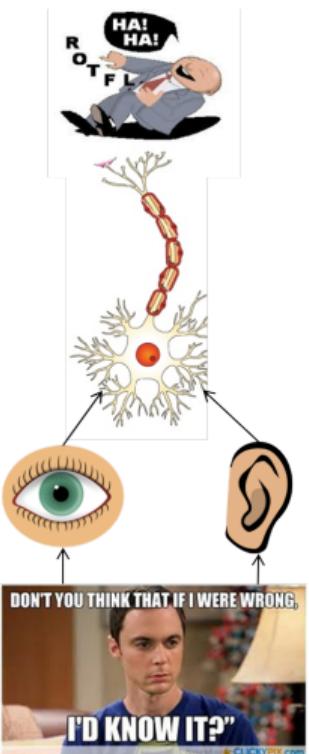
- 让我们借助一个卡通的例子来阐述神经元是如何工作的
- 我们的感官与外部世界进行交互





- 让我们借助一个卡通的例子来阐述神经元是如何工作的
- 我们的感官与外部世界进行交互
- 感官将信号传递给神经元

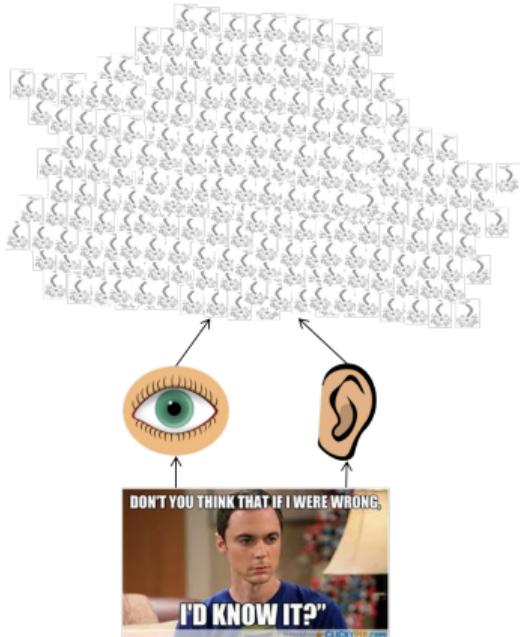




- 让我们借助一个卡通的例子来阐述神经元是如何工作的
- 我们的感官与外部世界进行交互
- 感官将信号传递给神经元
- 神经元被激活，产生一个响应（例如，例子中的『哈哈大笑』）

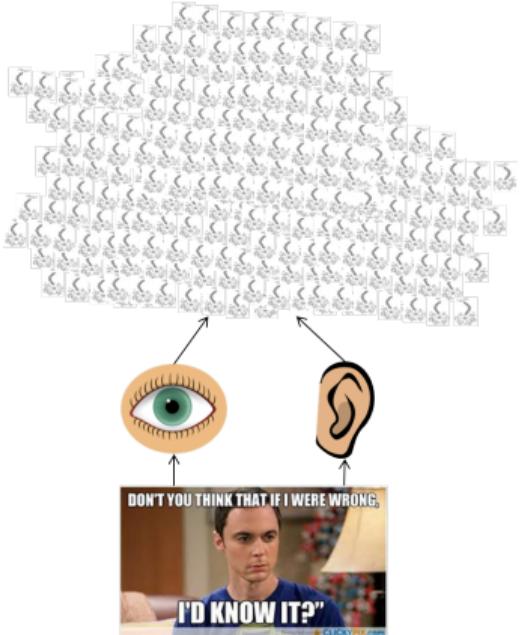


- 在现实中，并非是一个神经元来完成这件事情

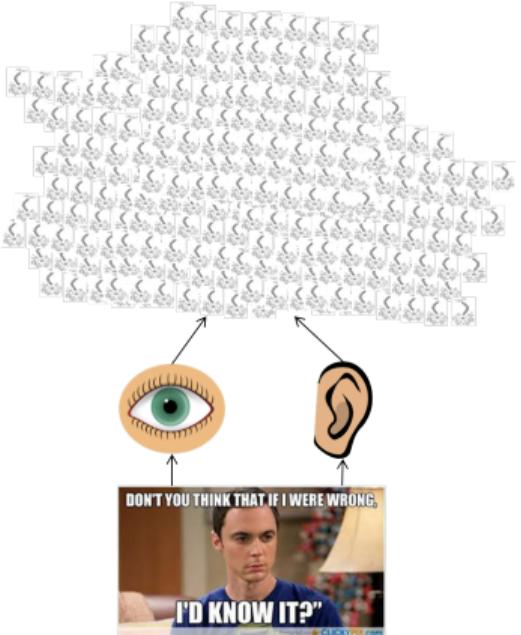




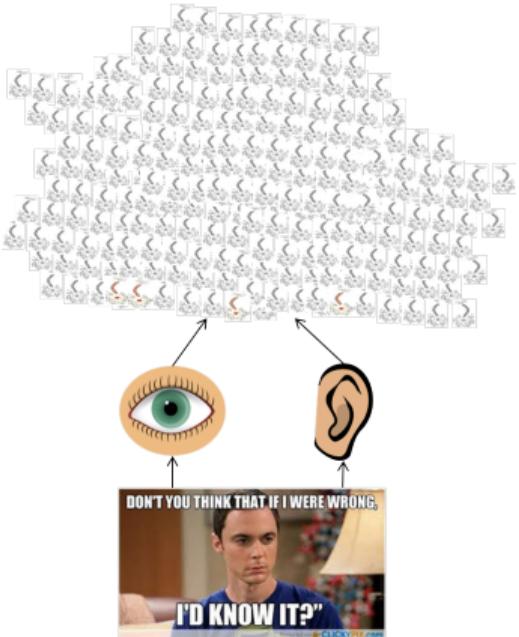
- 在现实中，并非是一个神经元来完成这件事情
- 而是由大量神经元并行相连而构成的一个网络

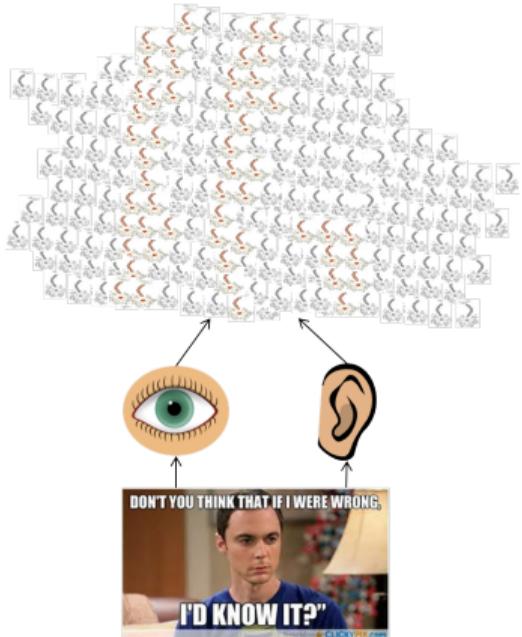


- 在现实中，并非是一个神经元来完成这件事情
- 而是由大量神经元并行相连而构成的一个网络
- 我们的感官将信息传递给最低层的神经元

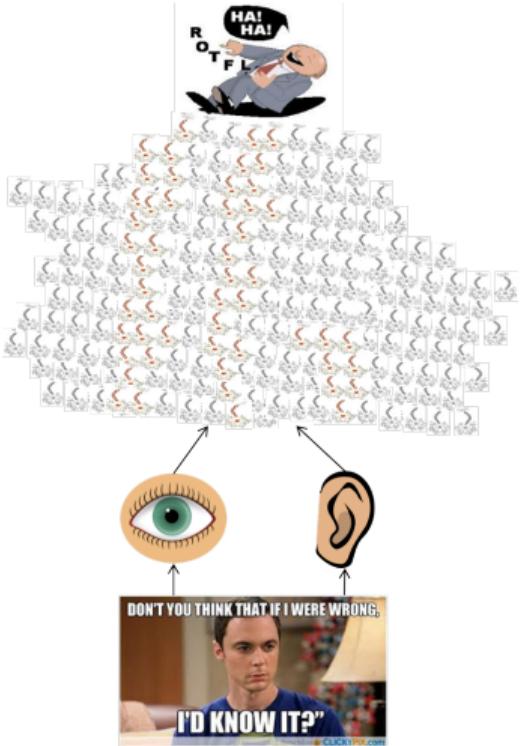


- 在现实中，并非是一个神经元来完成这件事情
- 而是由大量神经元并行相连而构成的一个网络
- 我们的感官将信息传递给最低层的神经元
- 这些神经元中的一些会被激活（红色）来响应接收到的信号，并将信息传递给与之相连的其他神经元

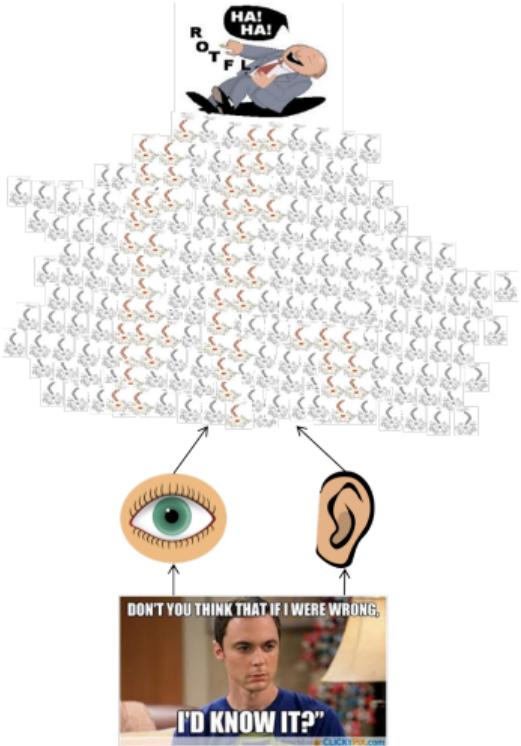




- 在现实中，并非是一个神经元来完成这件事情
- 而是由大量神经元并行相连而构成的一个网络
- 我们的感官将信息传递给最低层的神经元
- 这些神经元中的一些会被激活（红色）来响应接收到的信号，并将信息传递给与之相连的其他神经元
- 上层相连的神经元中会有一些神经元会被再次激活（红色），重复这个过程，



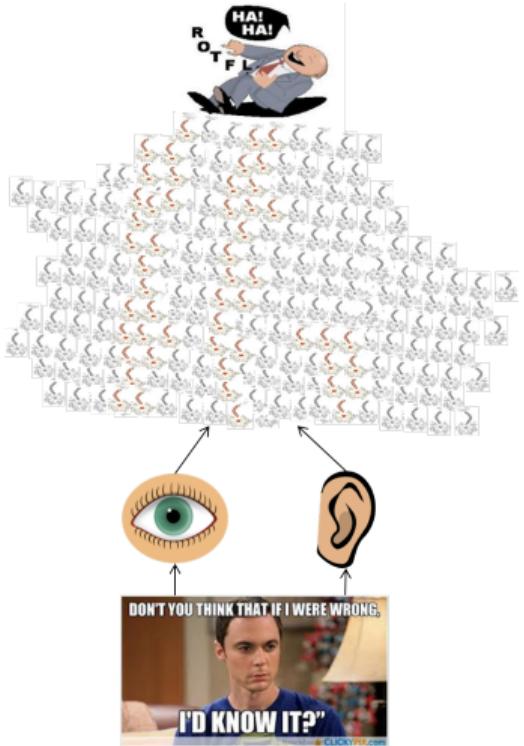
- 在现实中，并非是一个神经元来完成这件事情
- 而是由大量神经元并行相连而构成的一个网络
- 我们的感官将信息传递给最低层的神经元
- 这些神经元中的一些会被激活（红色）来响应接收到的信号，并将信息传递给与之相连的其他神经元
- 上层相连的神经元中会有一些神经元会被再次激活（红色），重复这个过程，最终产生一个响应（『哈哈大笑』）

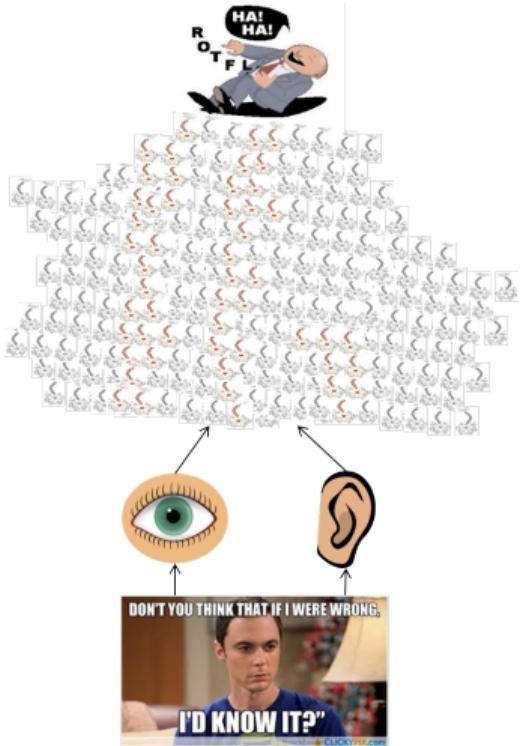


- 在现实中，并非是一个神经元来完成这件事情
- 而是由大量神经元并行相连而构成的一个网络
- 我们的感官将信息传递给最低层的神经元
- 这些神经元中的一些会被激活（红色）来响应接收到的信号，并将信息传递给与之相连的其他神经元
- 上层相连的神经元中会有一些神经元会被再次激活（红色），重复这个过程，最终产生一个响应（『哈哈大笑』）
- 一个普通人的大脑有大约 10^{11} (100 billion) 个神经元！



- 在这种由大量神经元并行相连的网络中，每个神经元是有具体分工的

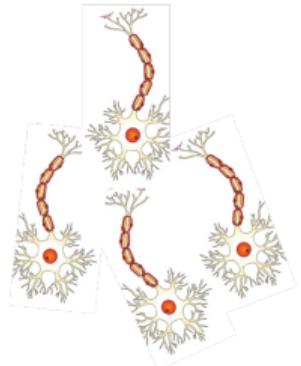




- 在这种由大量神经元并行相连的网络中，每个神经元是有具体分工的
- 每个神经元扮演一个特定的角色或对一个特定的激励进行响应



- 在这种由大量神经元并行相连的网络中，每个神经元是有具体分工的
- 每个神经元扮演一个特定的角色或对一个特定的激励进行响应



A simplified illustration



- 在这种由大量神经元并行相连的网络中，每个神经元是有具体分工的
- 每个神经元扮演一个特定的角色或对一个特定的激励进行响应



A simplified illustration



- 在这种由大量神经元并行相连的网络中，每个神经元是有具体分工的
- 每个神经元扮演一个特定的角色或对一个特定的激励进行响应



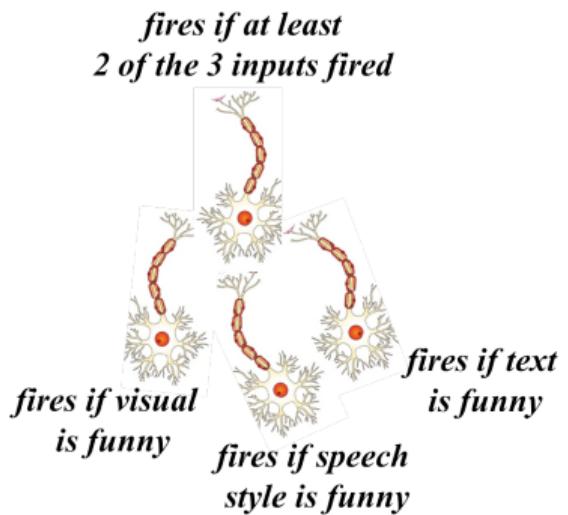
A simplified illustration



- 在这种由大量神经元并行相连的网络中，每个神经元是有具体分工的
- 每个神经元扮演一个特定的角色或对一个特定的激励进行响应



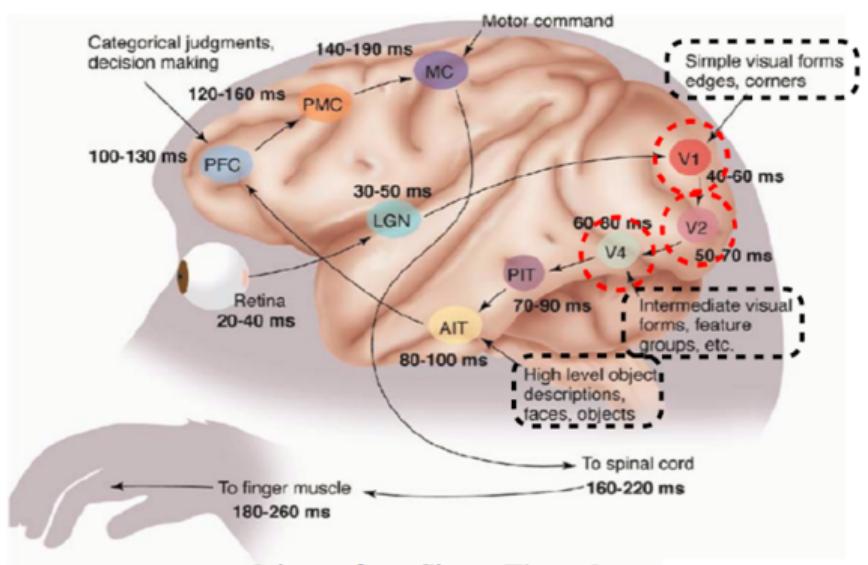
A simplified illustration



- 在这种由大量神经元并行相连的网络中，每个神经元是有具体分工的
- 每个神经元扮演一个特定的角色或对一个特定的激励进行响应

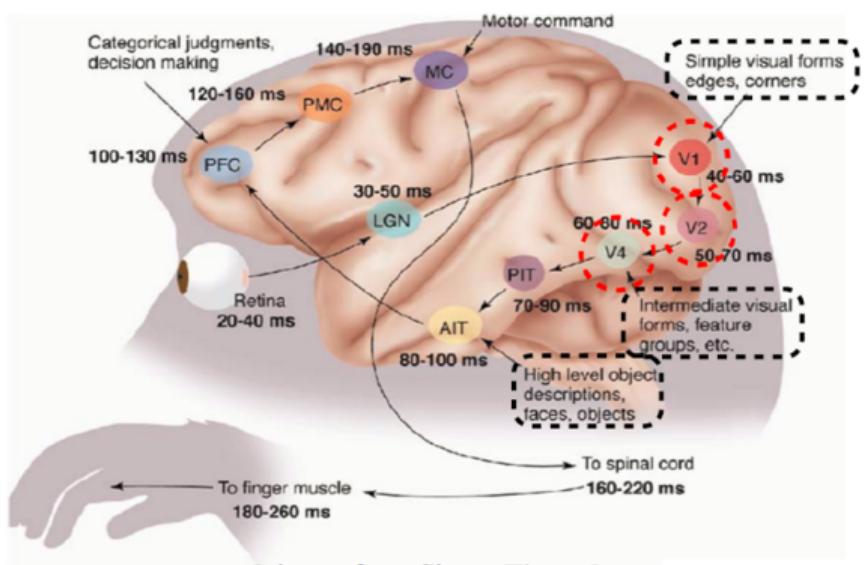
A simplified illustration

- 大脑中的神经元是以一种层次化的方式组织的



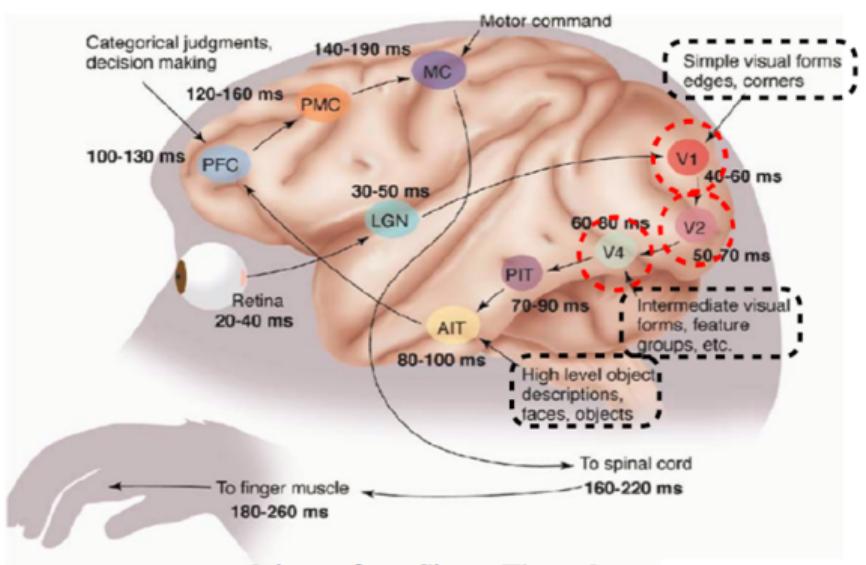
[picture from Simon Thorpe]

- 大脑中的神经元是以一种层次化的方式组织的
- 我们通过视觉皮层的示意图来展示这种层次化组织

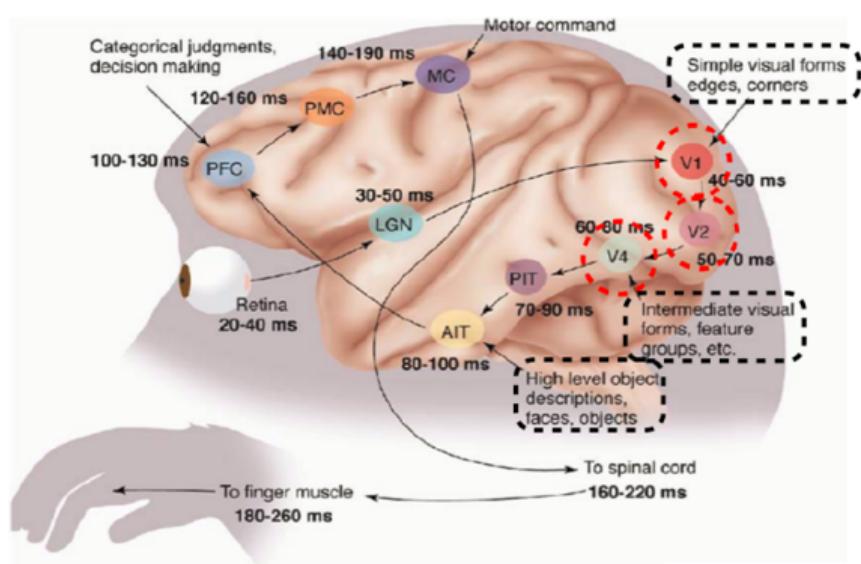


[picture from Simon Thorpe]

- 大脑中的神经元是以一种层次化的方式组织的
- 我们通过视觉皮层的示意图来展示这种层次化组织
- 从视网膜接收信息 (retina) 开始，信息沿着箭头方向传递



- 大脑中的神经元是以一种层次化的方式组织的
- 我们通过视觉皮层的示意图来展示这种层次化组织
- 从视网膜接收信息 (retina) 开始，信息沿着箭头方向传递
- 我们观察到层 V1, V2 到 AIT 形成了一个层次化的方式（响应简单的视觉模式到高级的物体）



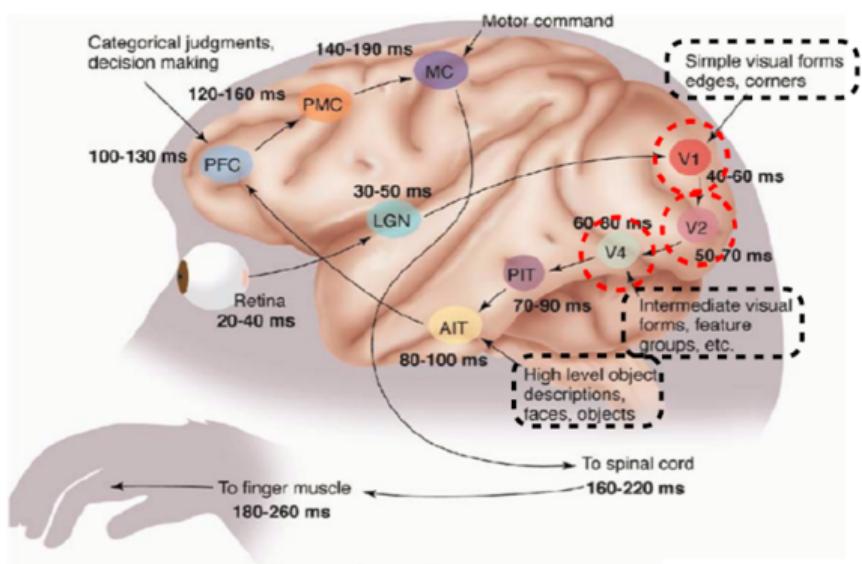
[picture from Simon Thorpe]



Layer 1: detect edges & corners

Sample illustration of hierarchical processing*

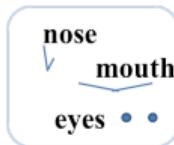
*Idea borrowed from Hugo Larochelle's lecture slides



[picture from Simon Thorpe]



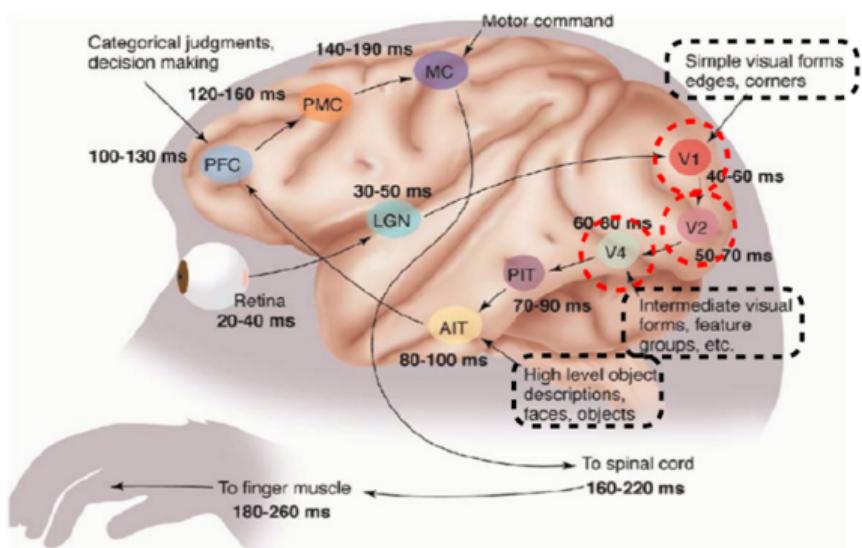
Layer 1: detect edges & corners



Layer 2: form feature groups

Sample illustration of hierarchical processing*

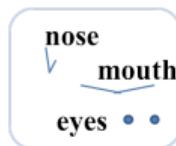
*Idea borrowed from Hugo Larochelle's lecture slides



[picture from Simon Thorpe]



Layer 1: detect edges & corners



Layer 2: form feature groups



Layer 3: detect high level objects, faces, etc.

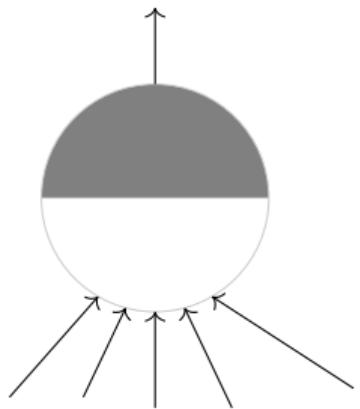
Sample illustration of hierarchical processing*

*Idea borrowed from Hugo Larochelle's lecture slides

M-P 神经元 (McCulloch Pitts Neuron)

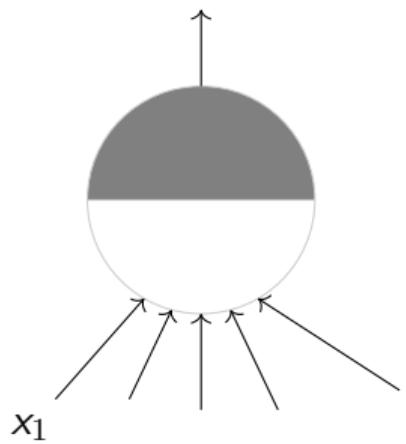


- 1943 年，神经科学家 McCulloch 和逻辑学家 Pitts 提出了一个非常简化的神经元计算模型



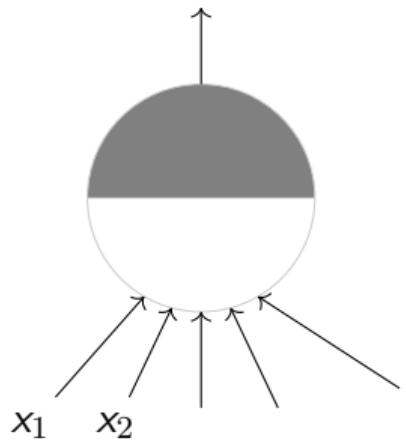


- 1943 年，神经科学家 McCulloch 和逻辑学家 Pitts 提出了一个非常简化的神经元计算模型



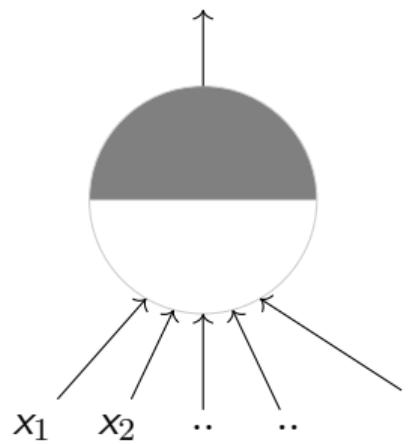


- 1943 年，神经科学家 McCulloch 和逻辑学家 Pitts 提出了一个非常简化的神经元计算模型



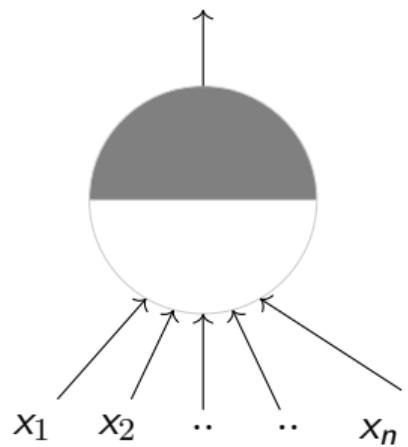


- 1943 年，神经科学家 McCulloch 和逻辑学家 Pitts 提出了一个非常简化的神经元计算模型



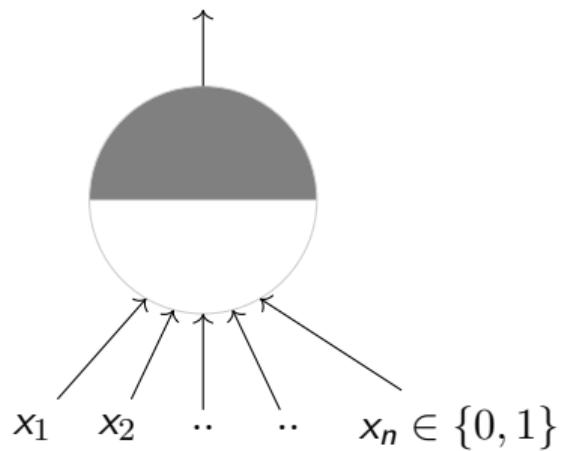


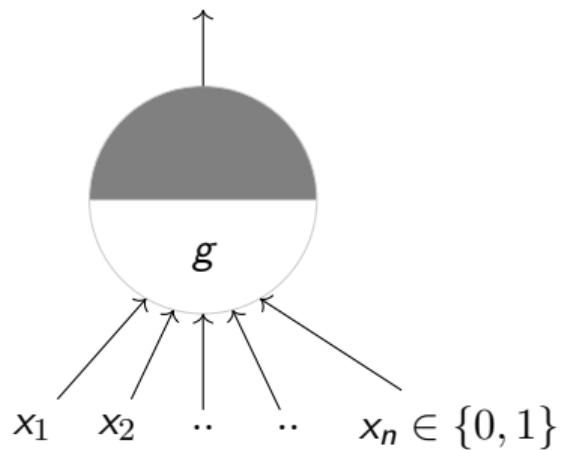
- 1943 年，神经科学家 McCulloch 和逻辑学家 Pitts 提出了一个非常简化的神经元计算模型



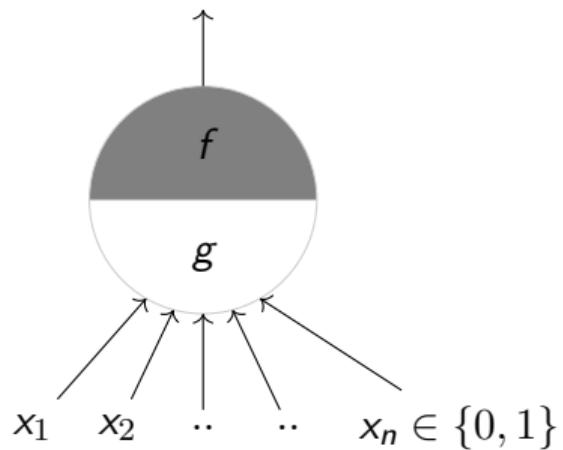


- 1943 年，神经科学家 McCulloch 和逻辑学家 Pitts 提出了一个非常简化的神经元计算模型



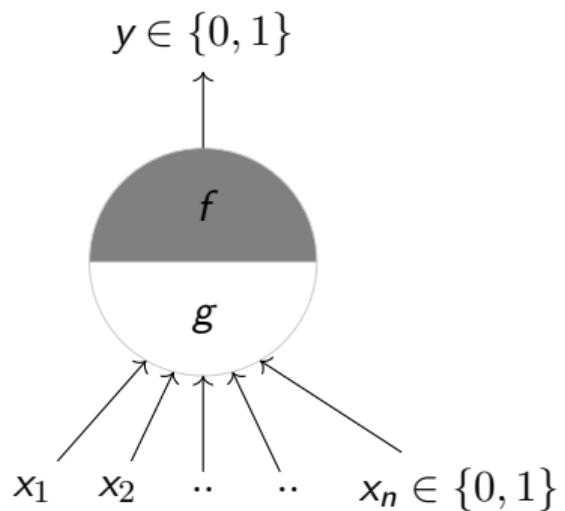


- 1943 年, 神经科学家 McCulloch 和逻辑学家 Pitts 提出了一个非常简化的神经元计算模型
- 函数 g 对输入进行聚合,



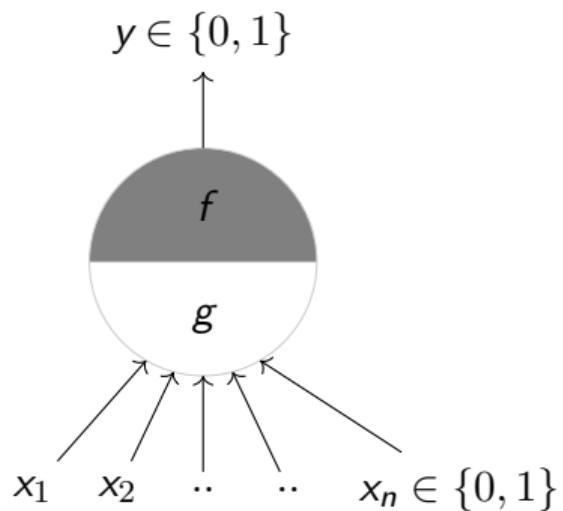
- 1943 年，神经科学家 McCulloch 和逻辑学家 Pitts 提出了一个非常简化的神经元计算模型
- 函数 g 对输入进行聚合，

$$g(x_1, x_2, \dots, x_n) = g(\mathbf{x}) = \sum_{i=1}^n x_i$$



- 1943 年，神经科学家 McCulloch 和逻辑学家 Pitts 提出了一个非常简化的神经元计算模型
- 函数 g 对输入进行聚合，函数 f 基于聚合的值做出一个决策

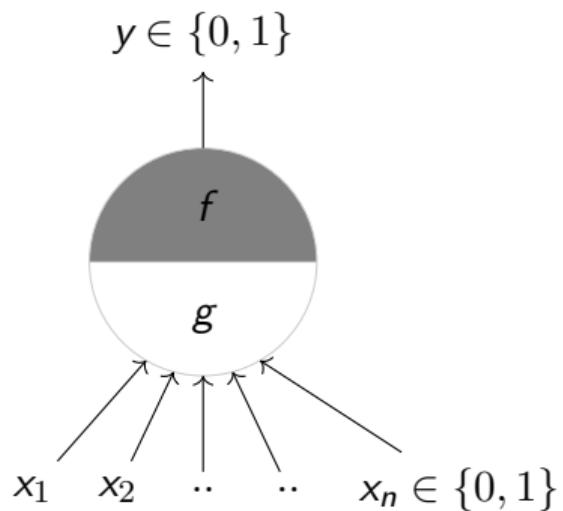
$$g(x_1, x_2, \dots, x_n) = g(\mathbf{x}) = \sum_{i=1}^n x_i$$



- 1943 年，神经科学家 McCulloch 和逻辑学家 Pitts 提出了一个非常简化的神经元计算模型
- 函数 g 对输入进行聚合，函数 f 基于聚合的值做出一个决策

$$g(x_1, x_2, \dots, x_n) = g(\mathbf{x}) = \sum_{i=1}^n x_i$$

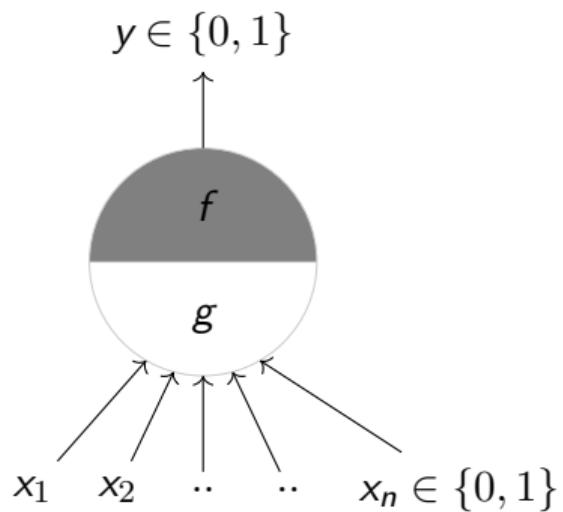
$$y = f(g(\mathbf{x})) = 1 \quad \text{if} \quad g(\mathbf{x}) \geq \theta$$



- 1943 年，神经科学家 McCulloch 和逻辑学家 Pitts 提出了一个非常简化的神经元计算模型
- 函数 g 对输入进行聚合，函数 f 基于聚合的值做出一个决策

$$g(x_1, x_2, \dots, x_n) = g(\mathbf{x}) = \sum_{i=1}^n x_i$$

$$\begin{aligned} y = f(g(\mathbf{x})) &= 1 & \text{if } g(\mathbf{x}) \geq \theta \\ &= 0 & \text{if } g(\mathbf{x}) < \theta \end{aligned}$$



- 1943 年，神经科学家 McCulloch 和逻辑学家 Pitts 提出了一个非常简化的神经元计算模型
- 函数 g 对输入进行聚合，函数 f 基于聚合的值做出一个决策

$$g(x_1, x_2, \dots, x_n) = g(\mathbf{x}) = \sum_{i=1}^n x_i$$

$$\begin{aligned} y &= f(g(\mathbf{x})) = 1 && \text{if } g(\mathbf{x}) \geq \theta \\ &= 0 && \text{if } g(\mathbf{x}) < \theta \end{aligned}$$

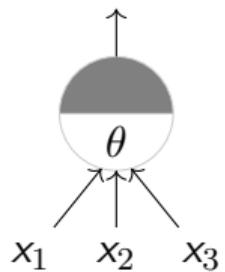
- θ 是阈值参数



我们可以使用 M-P 神经元来实现一些布尔函数...



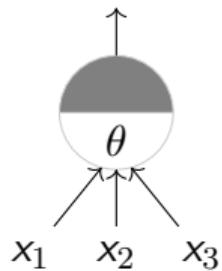
$$y \in \{0, 1\}$$



A McCulloch Pitts unit

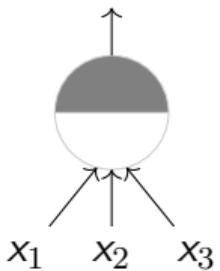


$$y \in \{0, 1\}$$



A McCulloch Pitts unit

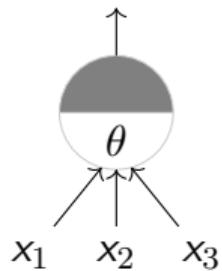
$$y \in \{0, 1\}$$



AND function

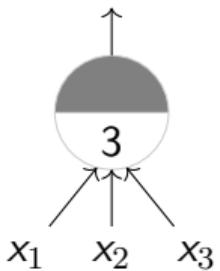


$$y \in \{0, 1\}$$



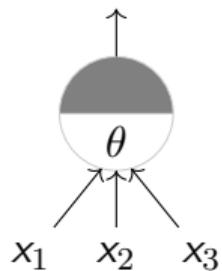
A McCulloch Pitts unit

$$y \in \{0, 1\}$$



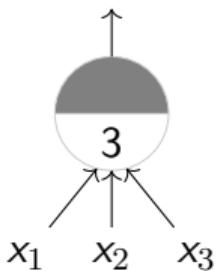
AND function

$$y \in \{0, 1\}$$



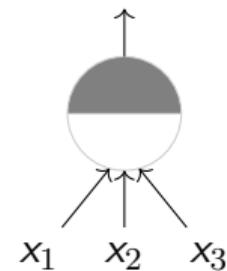
A McCulloch Pitts unit

$$y \in \{0, 1\}$$



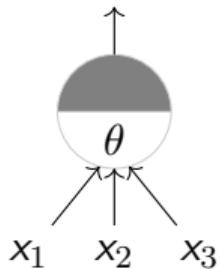
AND function

$$y \in \{0, 1\}$$



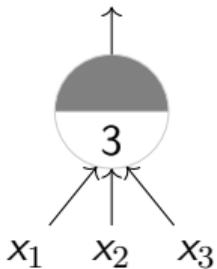
OR function

$$y \in \{0, 1\}$$



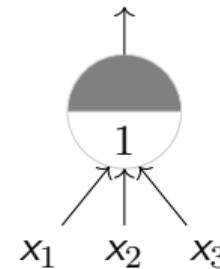
A McCulloch Pitts unit

$$y \in \{0, 1\}$$



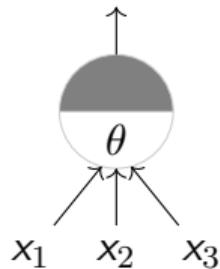
AND function

$$y \in \{0, 1\}$$



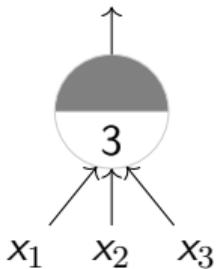
OR function

$$y \in \{0, 1\}$$



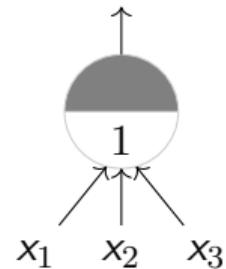
A McCulloch Pitts unit

$$y \in \{0, 1\}$$



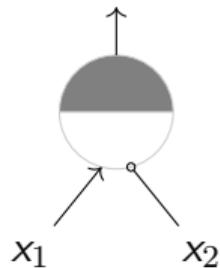
AND function

$$y \in \{0, 1\}$$



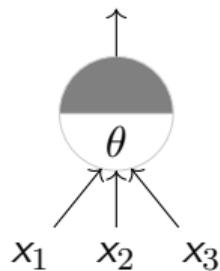
OR function

$$y \in \{0, 1\}$$



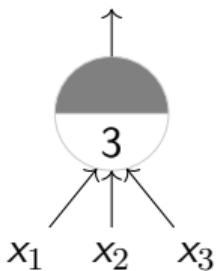
$$x_1 \text{ AND } !x_2^*$$

$$y \in \{0, 1\}$$



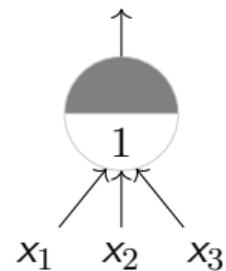
A McCulloch Pitts unit

$$y \in \{0, 1\}$$



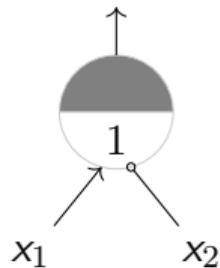
AND function

$$y \in \{0, 1\}$$



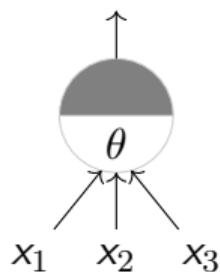
OR function

$$y \in \{0, 1\}$$



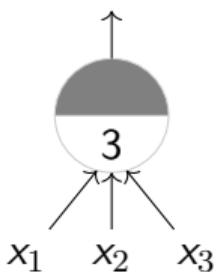
$$x_1 \text{ AND } !x_2^*$$

$$y \in \{0, 1\}$$



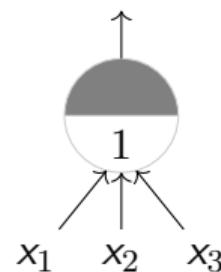
A McCulloch Pitts unit

$$y \in \{0, 1\}$$



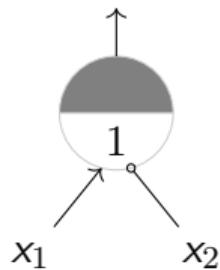
AND function

$$y \in \{0, 1\}$$



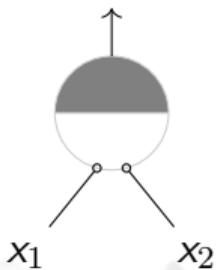
OR function

$$y \in \{0, 1\}$$



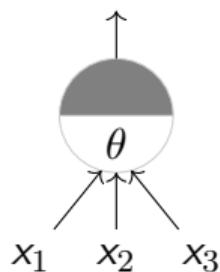
x_1 AND $\neg x_2^*$

$$y \in \{0, 1\}$$



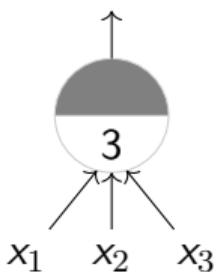
NOR function

$$y \in \{0, 1\}$$



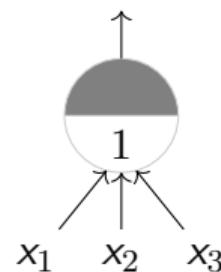
A McCulloch Pitts unit

$$y \in \{0, 1\}$$



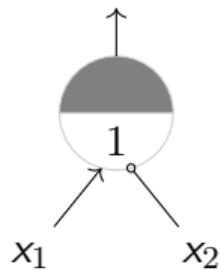
AND function

$$y \in \{0, 1\}$$



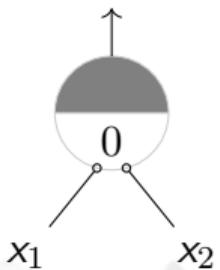
OR function

$$y \in \{0, 1\}$$



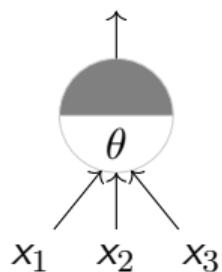
x_1 AND $\neg x_2^*$

$$y \in \{0, 1\}$$



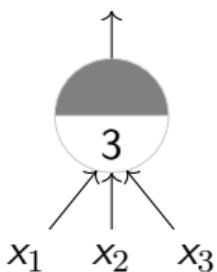
NOR function

$$y \in \{0, 1\}$$



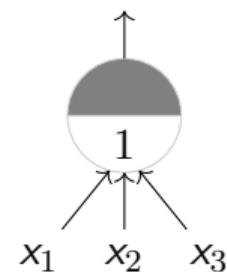
A McCulloch Pitts unit

$$y \in \{0, 1\}$$



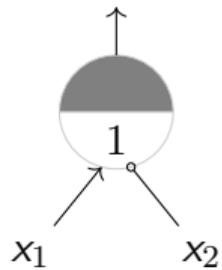
AND function

$$y \in \{0, 1\}$$



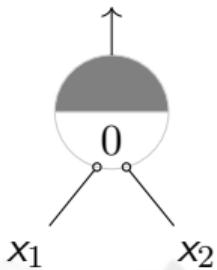
OR function

$$y \in \{0, 1\}$$



x_1 AND $\neg x_2^*$

$$y \in \{0, 1\}$$



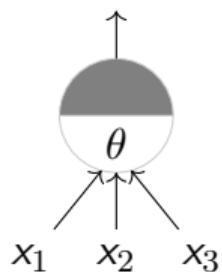
NOR function

$$y \in \{0, 1\}$$



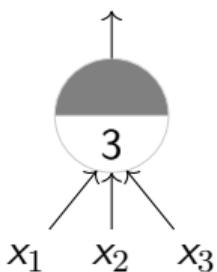
NOT function

$$y \in \{0, 1\}$$



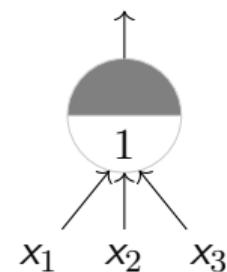
A McCulloch Pitts unit

$$y \in \{0, 1\}$$



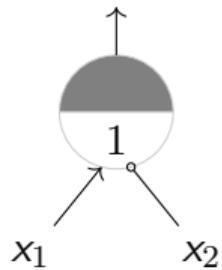
AND function

$$y \in \{0, 1\}$$



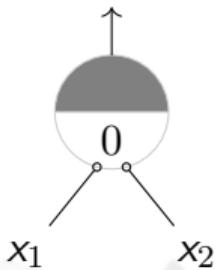
OR function

$$y \in \{0, 1\}$$



x_1 AND $\neg x_2^*$

$$y \in \{0, 1\}$$



NOR function

$$y \in \{0, 1\}$$



NOT function

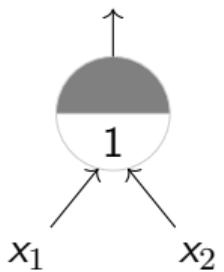


- M-P 神经元能实现任意的布尔函数吗?



- M-P 神经元能实现任意的布尔函数吗?
- 回答这个问题之前, 让我们首先看看 M-P 神经元的几何解释...

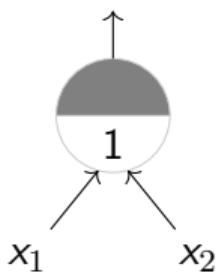
$$y \in \{0, 1\}$$



OR function

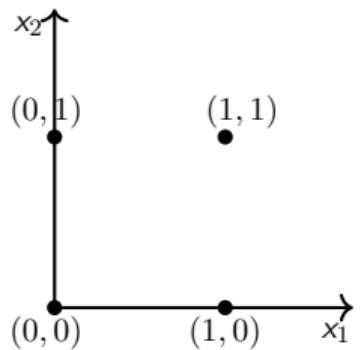
$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 1$$

$$y \in \{0, 1\}$$

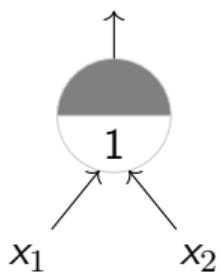


OR function

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 1$$

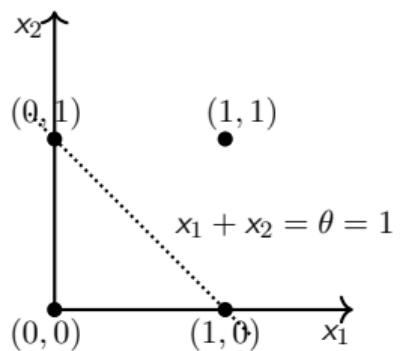


$$y \in \{0, 1\}$$

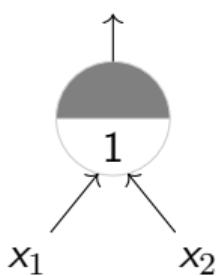


OR function

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 1$$



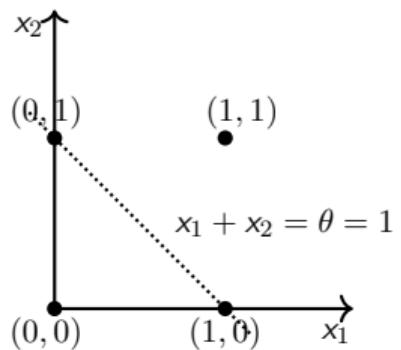
$$y \in \{0, 1\}$$



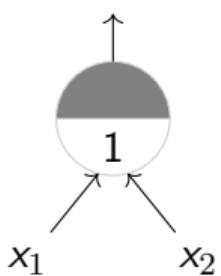
- 一个 M-P 神经元将两个二进制输入的 4 个点划分成两部分

OR function

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 1$$



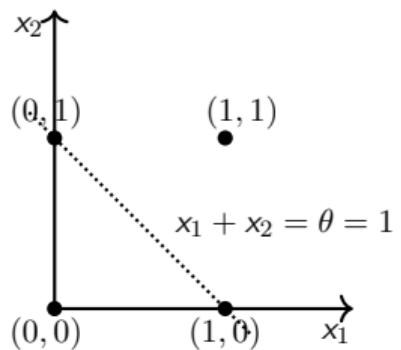
$$y \in \{0, 1\}$$



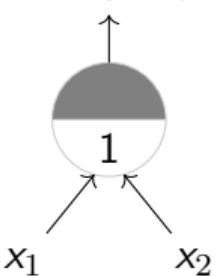
- 一个 M-P 神经元将两个二进制输入的 4 个点划分成两部分
- 3 个点存在于直线 $\sum_{i=1}^n x_i - \theta = 0$ 上或在其上方，另一个点在直线线下方

OR function

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 1$$

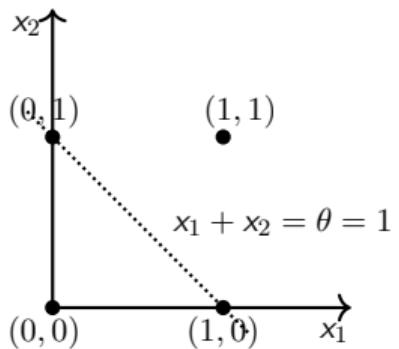


$$y \in \{0, 1\}$$



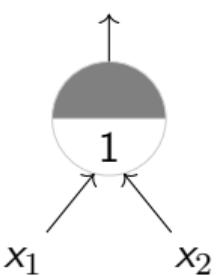
OR function

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 1$$



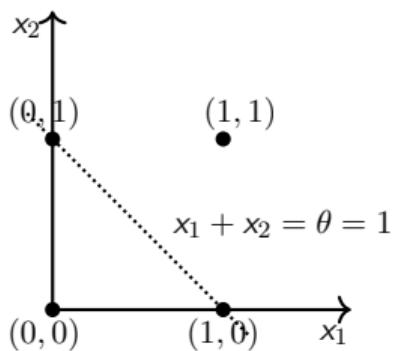
- 一个 M-P 神经元将两个二进制输入的 4 个点划分成两部分
- 3 个点存在于直线 $\sum_{i=1}^n x_i - \theta = 0$ 上或在其上方，另一个点在直线线下方
- 换句话说，所有对应输出为 0 的点在直线的一侧 ($\sum_{i=1}^n x_i < \theta$)，对应输出为 1 的点在直线的另一侧 ($\sum_{i=1}^n x_i \geq \theta$)

$$y \in \{0, 1\}$$



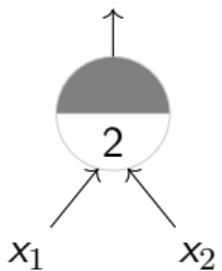
OR function

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 1$$



- 一个 M-P 神经元将两个二进制输入的 4 个点划分成两部分
- 3 个点存在于直线 $\sum_{i=1}^n x_i - \theta = 0$ 上或在其上方，另一个点在直线线下方
- 换句话说，所有对应输出为 0 的点在直线的一侧 ($\sum_{i=1}^n x_i < \theta$)，对应输出为 1 的点在直线的另一侧 ($\sum_{i=1}^n x_i \geq \theta$)
- 通过更多的例子来看看

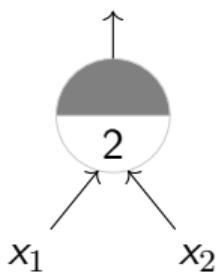
$$y \in \{0, 1\}$$



AND function

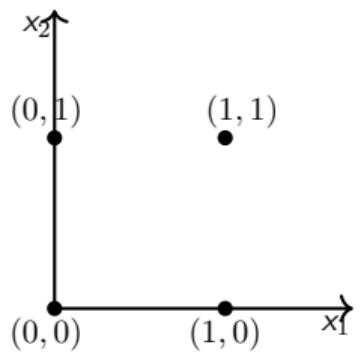
$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 2$$

$$y \in \{0, 1\}$$

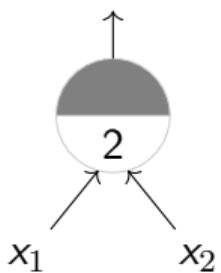


AND function

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 2$$

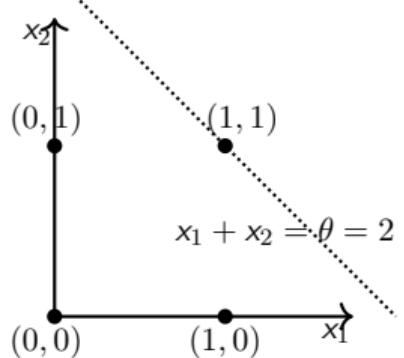


$$y \in \{0, 1\}$$

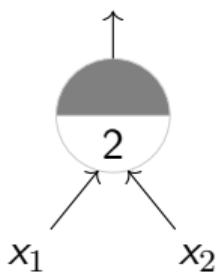


AND function

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 2$$

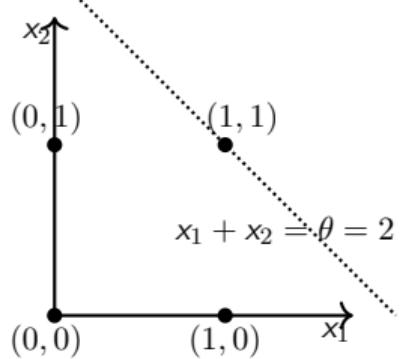


$$y \in \{0, 1\}$$

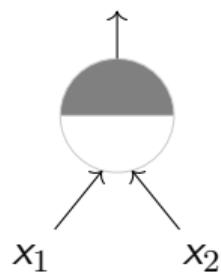


AND function

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 2$$

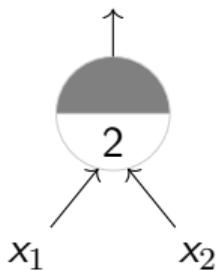


$$y \in \{0, 1\}$$



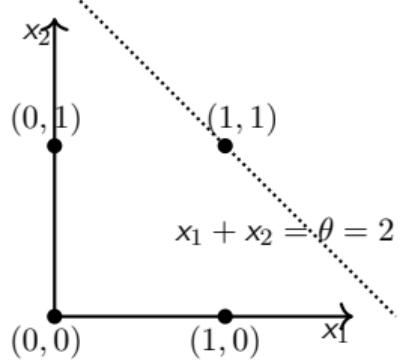
Tautology (always ON)

$$y \in \{0, 1\}$$

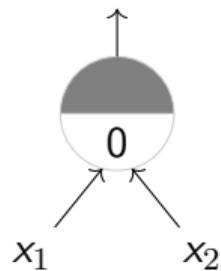


AND function

$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 2$$

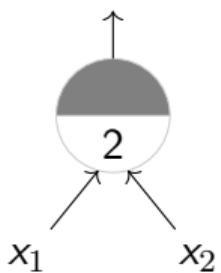


$$y \in \{0, 1\}$$



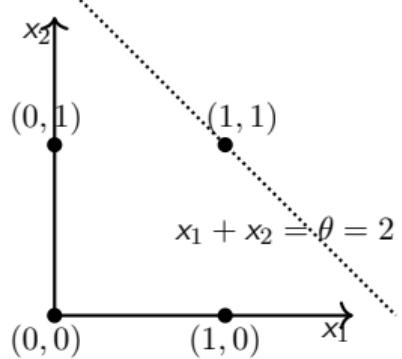
Tautology (always ON)

$$y \in \{0, 1\}$$

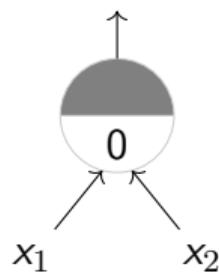


AND function

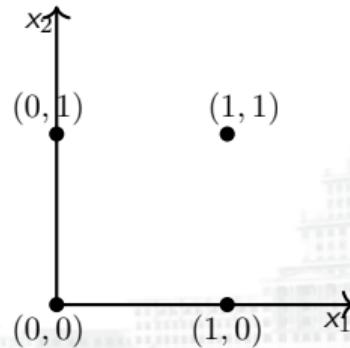
$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 2$$



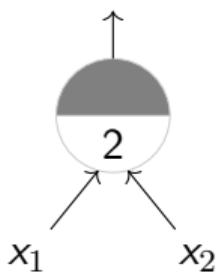
$$y \in \{0, 1\}$$



Tautology (always ON)

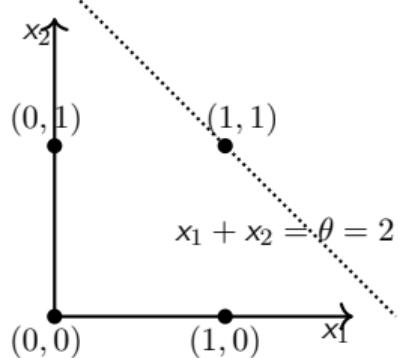


$$y \in \{0, 1\}$$

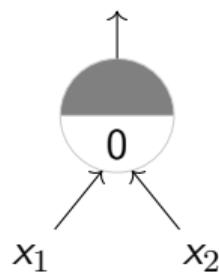


AND function

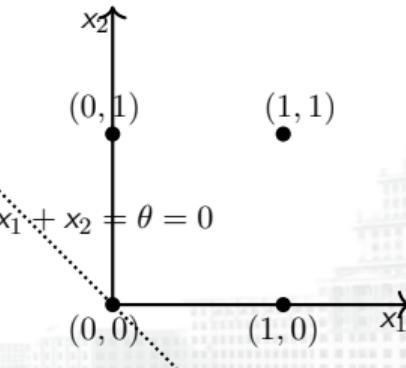
$$x_1 + x_2 = \sum_{i=1}^2 x_i \geq 2$$



$$y \in \{0, 1\}$$

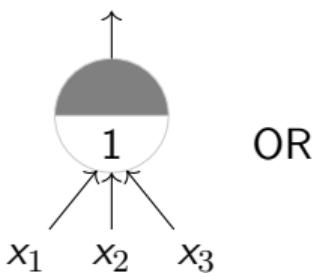


Tautology (always ON)



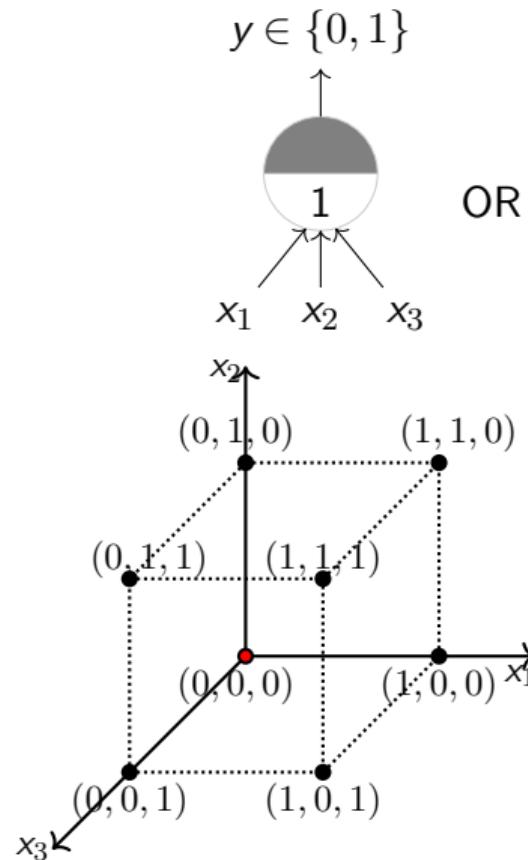


$$y \in \{0, 1\}$$

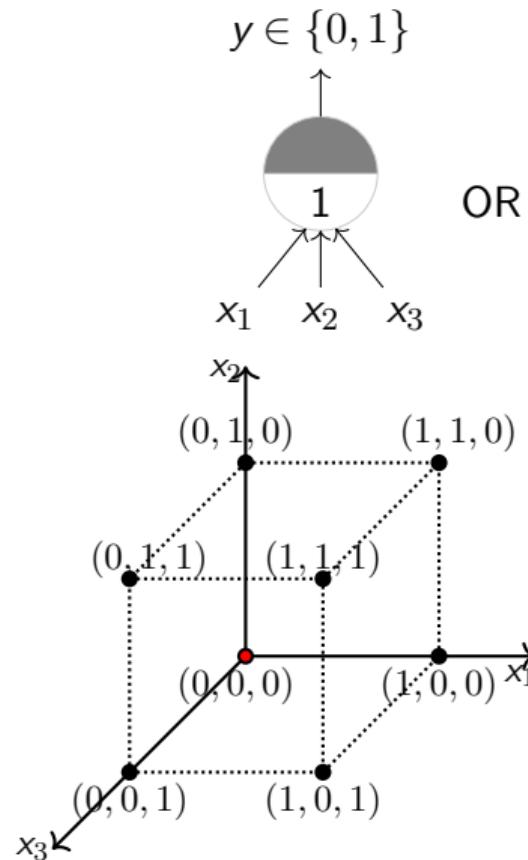


OR

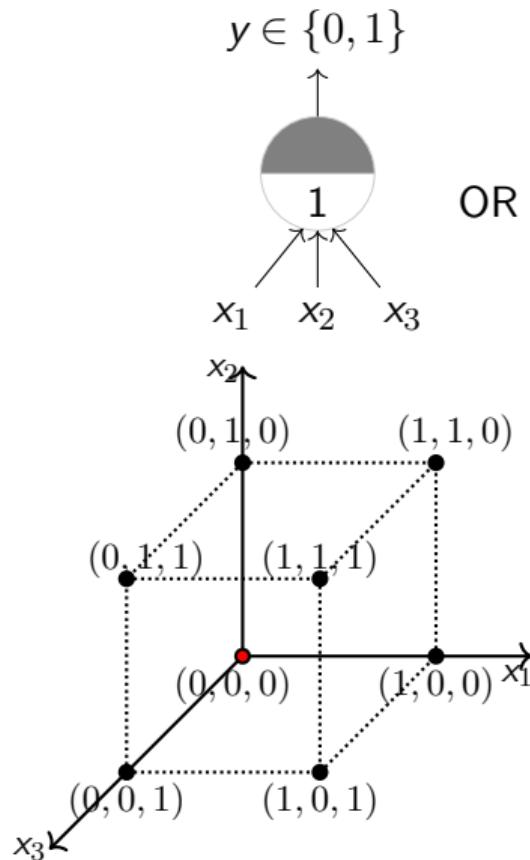
- 当有多于 2 个输入时会是什么情况?



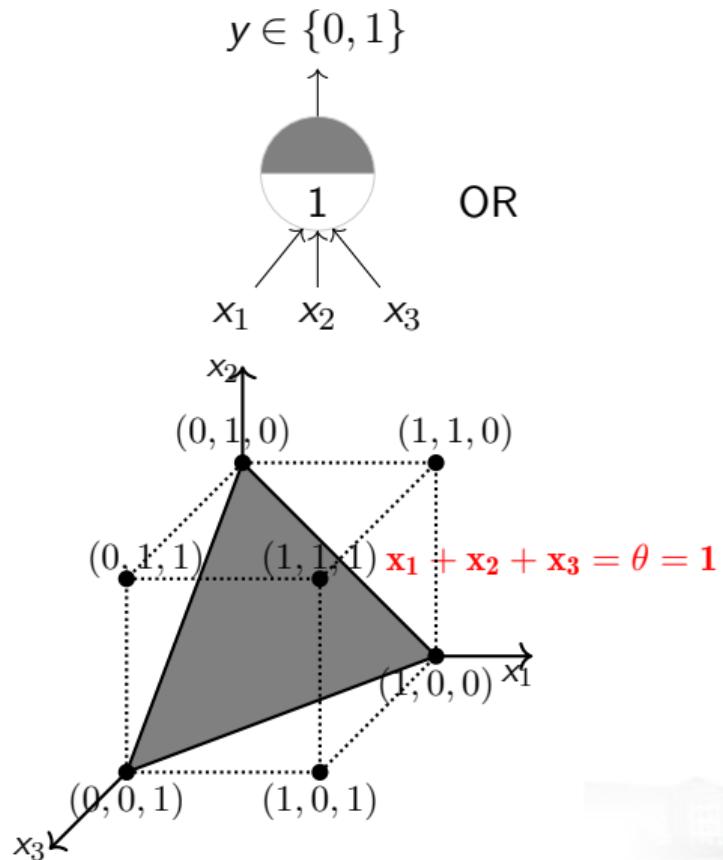
- 当有多于 2 个输入时会是什么情况?



- 当有多于 2 个输入时会是什么情况?
- 不同于 2 个输入时平面上的直线, 当多于 2 个输入时会存在一个(超)平面



- 当有多于 2 个输入时会是什么情况?
- 不同于 2 个输入时平面上的直线, 当多于 2 个输入时会存在一个(超)平面
- 对于 OR 函数, 存在一个平面使得点 $(0,0,0)$ 在平面的一侧, 剩下的 7 个点在平面的另一侧



- 当有多于 2 个输入时会是什么情况?
- 不同于 2 个输入时平面上的直线, 当多于 2 个输入时会存在一个(超)平面
- 对于 OR 函数, 存在一个平面使得点 $(0,0,0)$ 在平面的一侧, 剩下的 7 个点在平面的另一侧

总结

- 一个单个的 M-P 神经元能够用来实现线性可分的布尔函数

总结

- 一个单个的 M-P 神经元能够用来实现线性可分的布尔函数
- 对布尔函数而言，线性可分意味着：存在一条直线或一个平面使得所有产生输出 1 的输入位于线（或平面）的一侧，所有产生输出 0 的输入位于线（或平面）的另一侧

感知机 (Perceptron)



思考：

- 如何处理非布尔的输入（实数输入）？

思考：

- 如何处理非布尔的输入（实数输入）？
- 必须使用固定的阈值吗？

思考：

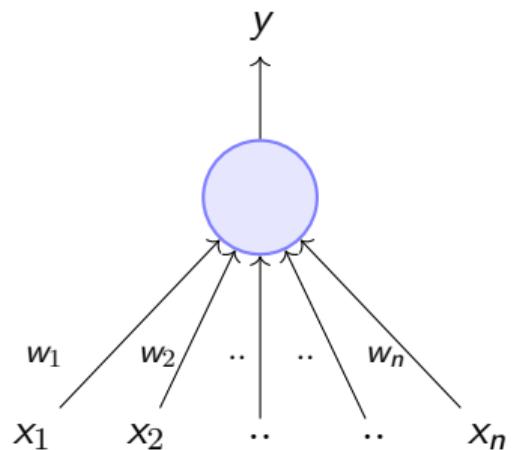
- 如何处理非布尔的输入（实数输入）？
- 必须使用固定的阈值吗？
- 所有的输入都是同等重要的吗？如果我们需要对某些输入赋予更多的权重（重要性）呢？



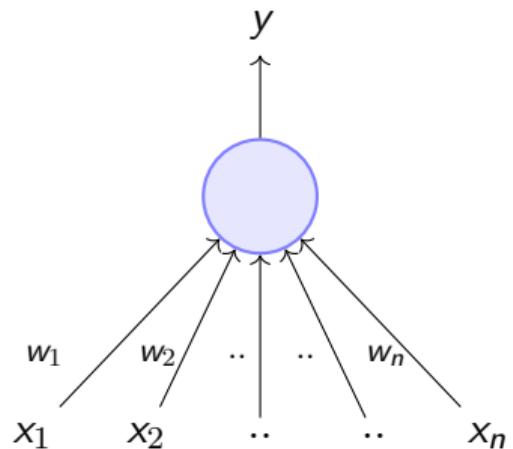
思考：

- 如何处理非布尔的输入（实数输入）？
- 必须使用固定的阈值吗？
- 所有的输入都是同等重要的吗？如果我们需要对某些输入赋予更多的权重（重要性）呢？
- 如何实现非线性可分的函数？

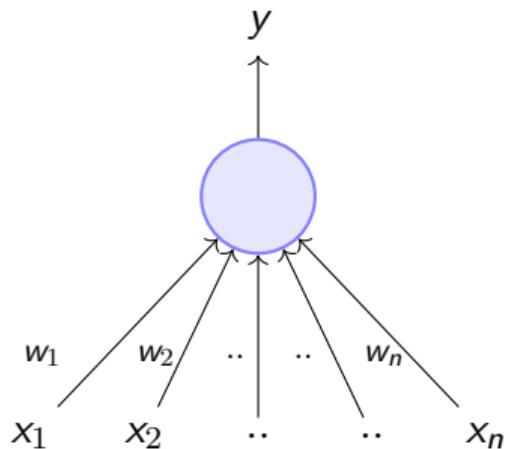
- 1958 年，美国心理学家 Frank Rosenblatt 提出了经典的感知机（perceptron）模型



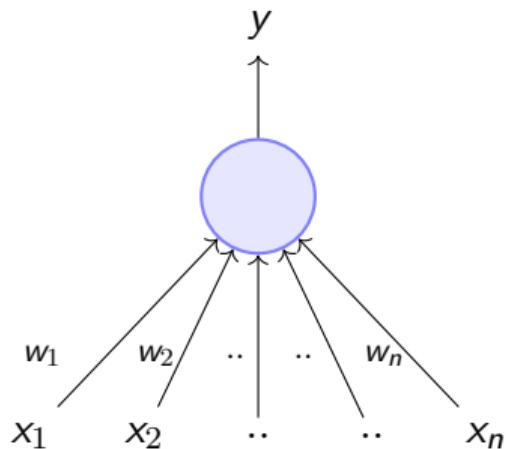
- 1958 年, 美国心理学家 Frank Rosenblatt 提出了经典的感知机 (perceptron) 模型



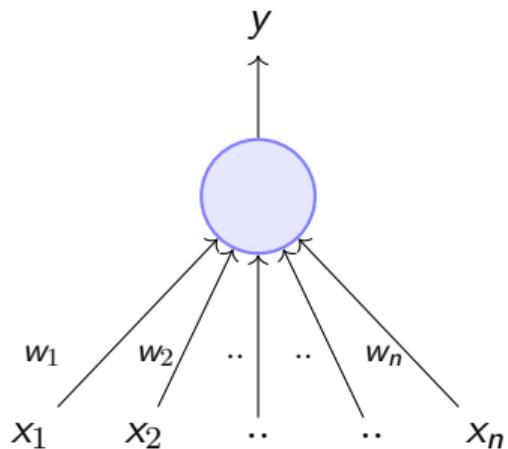
- 1958 年, 美国心理学家 Frank Rosenblatt 提出了经典的感知机 (perceptron) 模型
- 感知机是一个比 M-P 神经元更一般化的计算模型



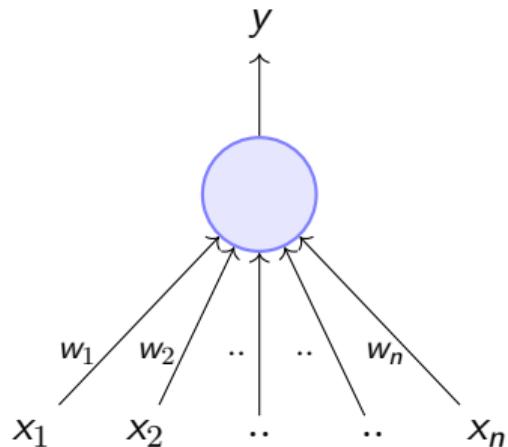
- 1958 年，美国心理学家 Frank Rosenblatt 提出了经典的感知机（perceptron）模型
- 感知机是一个比 M-P 神经元更一般化的计算模型
- **主要差别：**可以给输入赋值不同的权重，并且提供了学习权重的机制

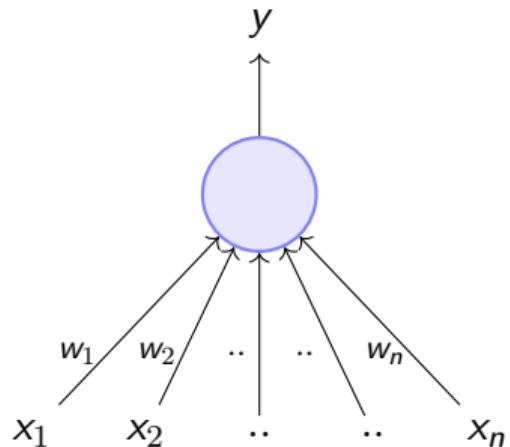


- 1958 年，美国心理学家 Frank Rosenblatt 提出了经典的感知机（perceptron）模型
- 感知机是一个比 M-P 神经元更一般化的计算模型
- **主要差别：**可以给输入赋值不同的权重，并且提供了学习权重的机制
- 输入不再局限于布尔值

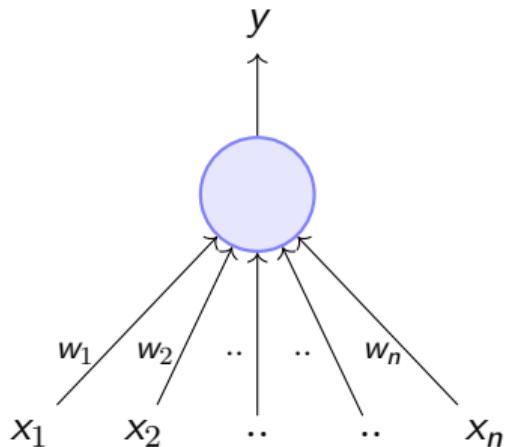


- 1958 年，美国心理学家 Frank Rosenblatt 提出了经典的感知机（perceptron）模型
- 感知机是一个比 M-P 神经元更一般化的计算模型
- **主要差别：**可以给输入赋值不同的权重，并且提供了学习权重的机制
- 输入不再局限于布尔值
- 1969 年经过 Minsky and Papert 的优化和详细分析 — 感知机（perceptron）模型指的就是 Minsky and Papert 改进后的模型

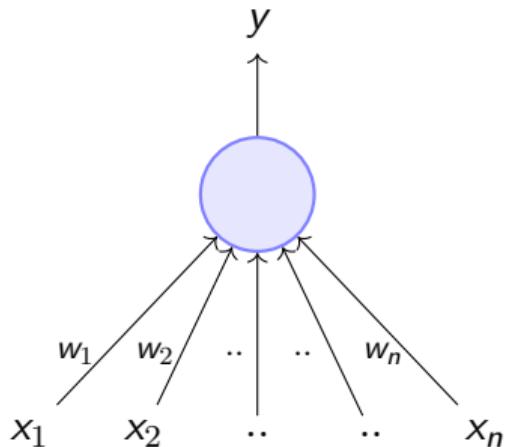




$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i \geq \theta$$

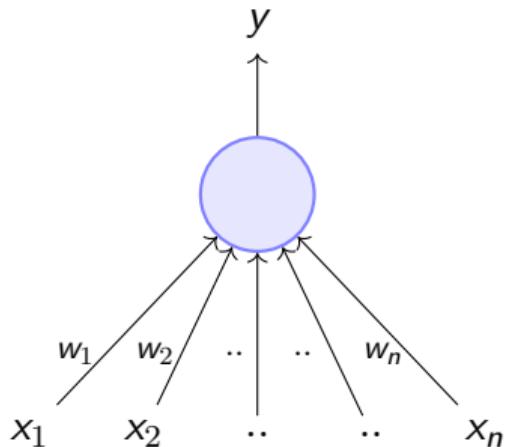


$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i \geq \theta$$
$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i < \theta$$



$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i \geq \theta$$
$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i < \theta$$

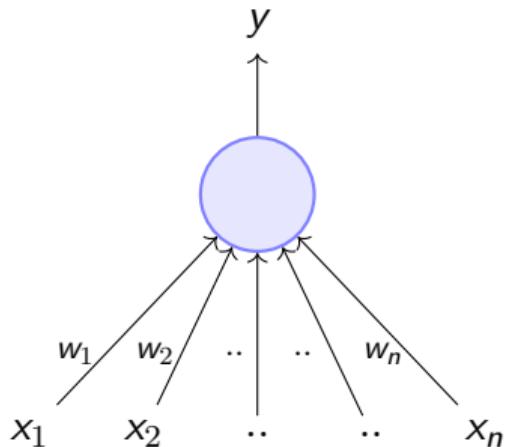
重写上面的式子,



$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i \geq \theta$$
$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i < \theta$$

重写上面的式子,

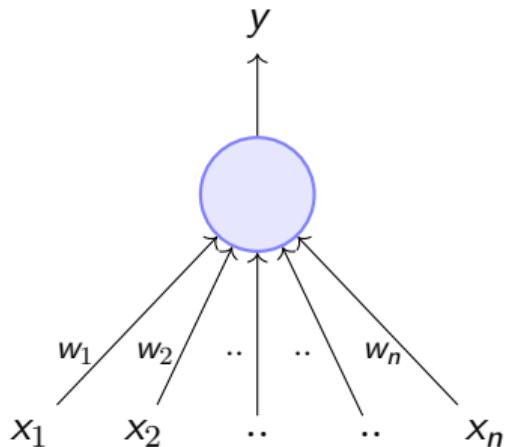
$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta \geq 0$$



$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i \geq \theta$$
$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i < \theta$$

重写上面的式子,

$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta \geq 0$$
$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta < 0$$

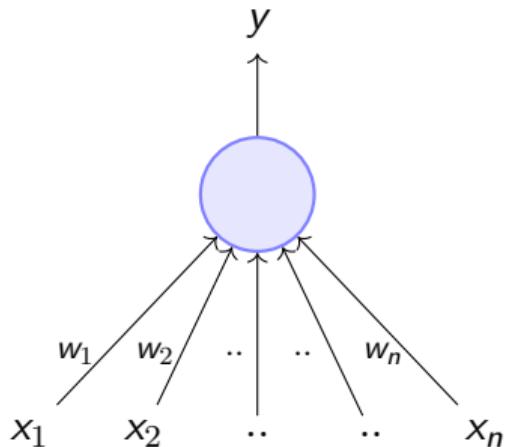


$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i \geq \theta$$
$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i < \theta$$

重写上面的式子,

一个更简洁的表示:

$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta \geq 0$$
$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta < 0$$



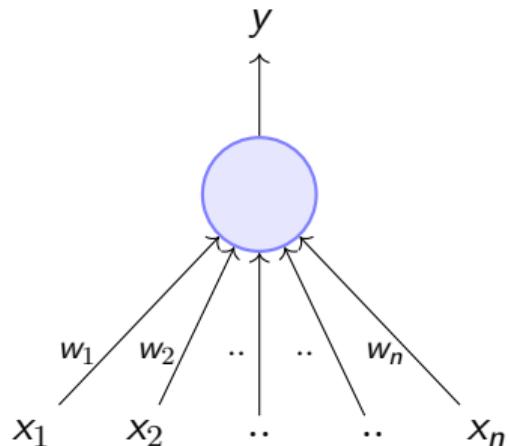
$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^n w_i * x_i \geq \theta \\ 0 & \text{if } \sum_{i=1}^n w_i * x_i < \theta \end{cases}$$

重写上面的式子,

一个更简洁的表示:

$$y = 1 \quad \text{if } \sum_{i=0}^n w_i * x_i \geq 0$$

$$y = \begin{cases} 1 & \text{if } \sum_{i=1}^n w_i * x_i - \theta \geq 0 \\ 0 & \text{if } \sum_{i=1}^n w_i * x_i - \theta < 0 \end{cases}$$



$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i \geq \theta$$

$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i < \theta$$

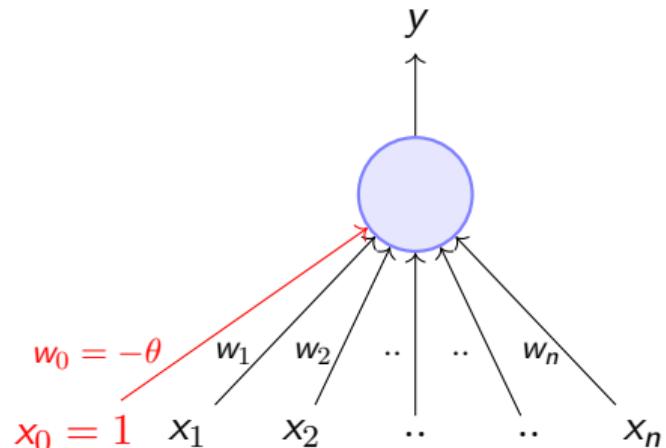
重写上面的式子,

一个更简洁的表示:

$$y = 1 \quad \text{if} \sum_{i=0}^n w_i * x_i \geq 0$$

$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta \geq 0$$

$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta < 0$$



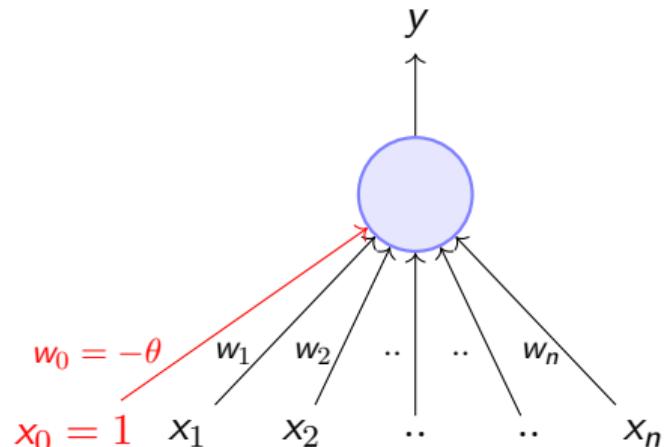
一个更简洁的表示：

$$y = 1 \quad \text{if} \sum_{i=0}^n w_i * x_i \geq 0$$

重写上面的式子,

$$\begin{aligned} y &= 1 \quad \text{if} \sum_{i=1}^n w_i * x_i \geq \theta \\ &= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i < \theta \end{aligned}$$

$$\begin{aligned} y &= 1 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta \geq 0 \\ &= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta < 0 \end{aligned}$$



一个更简洁的表示：

$$y = 1 \quad \text{if} \sum_{i=0}^n w_i * x_i \geq 0$$

$$= 0 \quad \text{if} \sum_{i=0}^n w_i * x_i < 0$$

$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i \geq \theta$$

$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i < \theta$$

重写上面的式子,

$$y = 1 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta \geq 0$$

$$= 0 \quad \text{if} \sum_{i=1}^n w_i * x_i - \theta < 0$$

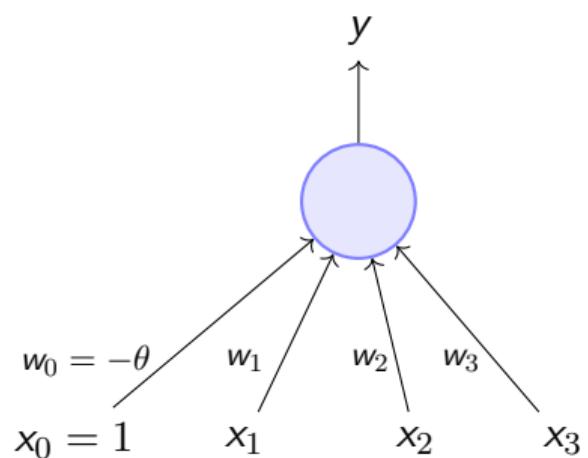


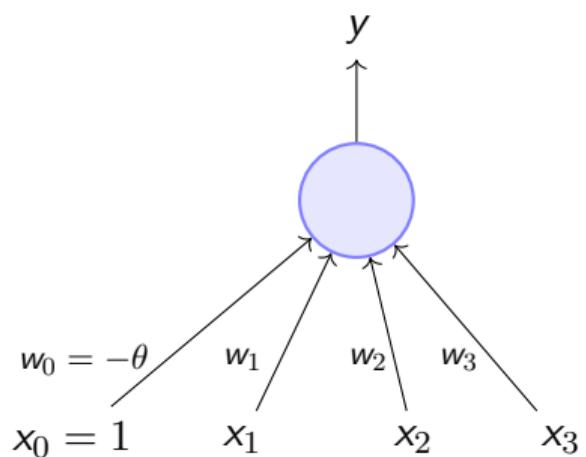
思考:

- 为什么我们要尝试实现布尔函数?
- 为什么我们需要权重?
- 为什么称 $w_0 = -\theta$ 为偏置 (bias) ?

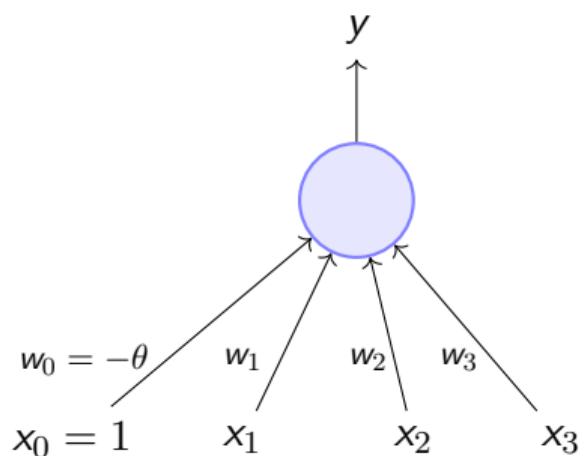


- 考虑一个任务：预测我们是否会喜欢看一部电影





- 考虑一个任务：预测我们是否会喜欢看一部电影
- 假设我们的预测是基于三个输入（为简化，每个输入是二值的）

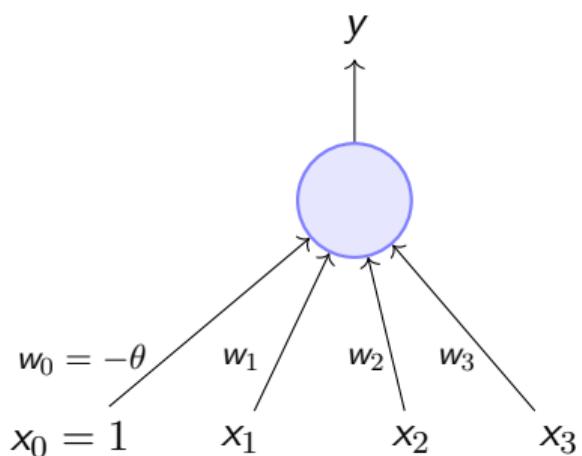


- 考虑一个任务：预测我们是否会喜欢看一部电影
- 假设我们的预测是基于三个输入（为简化，每个输入是二值的）

$x_1 = \text{isActorDamon}$

$x_2 = \text{isGenreThriller}$

$x_3 = \text{isDirectorNolan}$

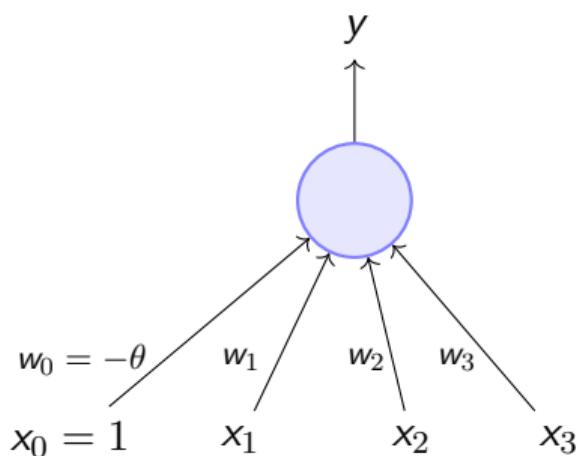


- 考虑一个任务：预测我们是否会喜欢看一部电影
- 假设我们的预测是基于三个输入（为简化，每个输入是二值的）
- 基于我们过去观看电影的经历 (**data**)，我们可能会给 *isDirectorNolan* 一个相对更高的权重

$x_1 = \text{isActorDamon}$

$x_2 = \text{isGenreThriller}$

$x_3 = \text{isDirectorNolan}$



- 考虑一个任务：预测我们是否会喜欢看一部电影
- 假设我们的预测是基于三个输入（为简化，每个输入是二值的）
- 基于我们过去观看电影的经历 (**data**)，我们可能会给 *isDirectorNolan* 一个相对更高的权重
- 这意味着，即使演员不是 *Matt Damon* 类型不是 *thriller*，我们仍然希望通过阈值 θ

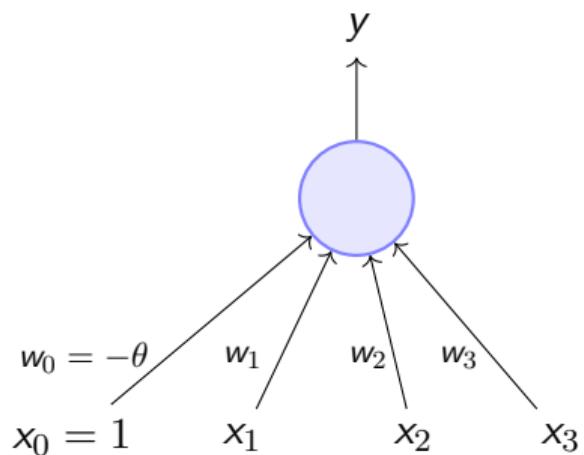
$x_1 = \text{isActorDamon}$

$x_2 = \text{isGenreThriller}$

$x_3 = \text{isDirectorNolan}$



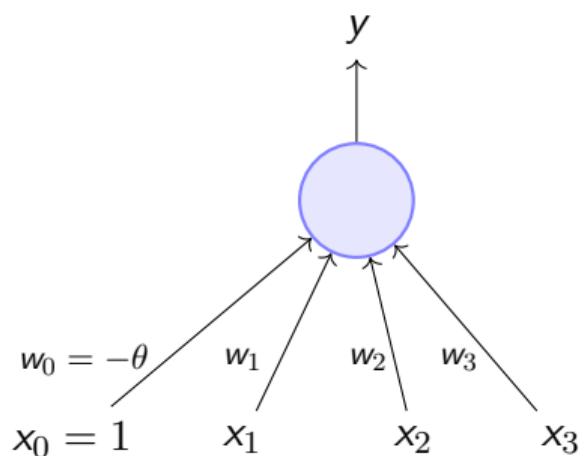
- w_0 被称为偏置, 因为它表示经验 (偏见 prejudice)



$x_1 = \text{isActorDamon}$

$x_2 = \text{isGenreThriller}$

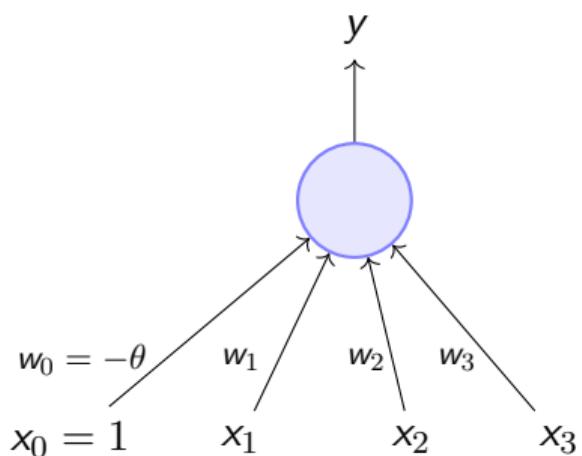
$x_3 = \text{isDirectorNolan}$



$x_1 = \text{isActorDamon}$

$x_2 = \text{isGenreThriller}$

$x_3 = \text{isDirectorNolan}$

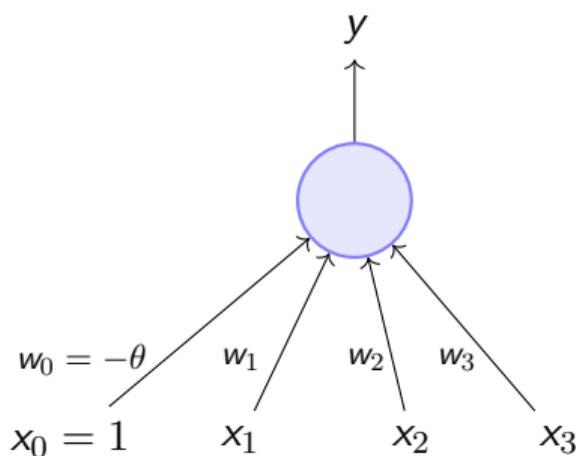


- w_0 被称为偏置, 因为它表示经验 (偏见 prejudice)
- 一个电影迷有一个非常低的阈值, 这样无论什么样的电影, 他都会喜欢看 [$\theta = 0$]
- 另一方面, 一个有主见的观影人可能只会看 Matt Damon 主演 Nolan 导演的惊悚片 [$\theta = 3$]

$x_1 = \text{isActorDamon}$

$x_2 = \text{isGenreThriller}$

$x_3 = \text{isDirectorNolan}$



- w_0 被称为偏置, 因为它表示经验 (偏见 prejudice)
- 一个电影迷有一个非常低的阈值, 这样无论什么样的电影, 他都会喜欢看 [$\theta = 0$]
- 另一方面, 一个有主见的观影人可能只会看 Matt Damon 主演 Nolan 导演的惊悚片 [$\theta = 3$]
- 权重 (w_1, w_2, \dots, w_n) 和偏置 (w_0) 依赖于过去的观影经历

$x_1 = \text{isActorDamon}$

$x_2 = \text{isGenreThriller}$

$x_3 = \text{isDirectorNolan}$

感知机能实现什么样的函数？感知机和 M-P 神经元有什么区别？

M-P 神经元

$$y = 1 \quad if \sum_{i=0}^n x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^n x_i < 0$$

感知机

$$y = 1 \quad if \sum_{i=0}^n \textcolor{red}{w_i} * x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^n \textcolor{red}{w_i} * x_i < 0$$

M-P 神经元

- 感知机也是将输入空间划分成两部分

$$y = 1 \quad if \sum_{i=0}^n x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^n x_i < 0$$

感知机

$$y = 1 \quad if \sum_{i=0}^n \textcolor{red}{w_i} * x_i \geq 0$$

$$= 0 \quad if \sum_{i=0}^n \textcolor{red}{w_i} * x_i < 0$$

M-P 神经元

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^n x_i \geq 0 \\ 0 & \text{if } \sum_{i=0}^n x_i < 0 \end{cases}$$

- 感知机也是将输入空间划分成两部分
- 对应输出为 1 的所有输入位于一侧，对应输出为 0 的所有输入位于另一侧

感知机

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^n \mathbf{w}_i * \mathbf{x}_i \geq 0 \\ 0 & \text{if } \sum_{i=0}^n \mathbf{w}_i * \mathbf{x}_i < 0 \end{cases}$$

M-P 神经元

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^n x_i \geq 0 \\ 0 & \text{if } \sum_{i=0}^n x_i < 0 \end{cases}$$

- 感知机也是将输入空间划分成两部分
- 对应输出为 1 的所有输入位于一侧，对应输出为 0 的所有输入位于另一侧
- 换句话说，单独一个感知机只能用于实现线性可分的函数

感知机

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^n \mathbf{w}_i * \mathbf{x}_i \geq 0 \\ 0 & \text{if } \sum_{i=0}^n \mathbf{w}_i * \mathbf{x}_i < 0 \end{cases}$$

M-P 神经元

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^n x_i \geq 0 \\ 0 & \text{if } \sum_{i=0}^n x_i < 0 \end{cases}$$

- 感知机也是将输入空间划分成两部分
- 对应输出为 1 的所有输入位于一侧，对应输出为 0 的所有输入位于另一侧
- 换句话说，单独一个感知机只能用于实现线性可分的函数
- 感知机和 M-P 神经元的差别是什么？

感知机

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^n \mathbf{w}_i * x_i \geq 0 \\ 0 & \text{if } \sum_{i=0}^n \mathbf{w}_i * x_i < 0 \end{cases}$$

M-P 神经元

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^n x_i \geq 0 \\ 0 & \text{if } \sum_{i=0}^n x_i < 0 \end{cases}$$

- 感知机也是将输入空间划分成两部分
- 对应输出为 1 的所有输入位于一侧，对应输出为 0 的所有输入位于另一侧
- 换句话说，单独一个感知机只能用于实现线性可分的函数
- 感知机和 M-P 神经元的差别是什么？感知机的权重包括阈值能通过学习得到；感知机的输入可以是实数

感知机

$$y = \begin{cases} 1 & \text{if } \sum_{i=0}^n w_i * x_i \geq 0 \\ 0 & \text{if } \sum_{i=0}^n w_i * x_i < 0 \end{cases}$$

M-P 神经元

$$y = 1 \quad \text{if} \sum_{i=0}^n x_i \geq 0$$
$$= 0 \quad \text{if} \sum_{i=0}^n x_i < 0$$

- 感知机也是将输入空间划分成两部分
- 对应输出为 1 的所有输入位于一侧，对应输出为 0 的所有输入位于另一侧
- 换句话说，单独一个感知机只能用于实现线性可分的函数
- 感知机和 M-P 神经元的差别是什么？感知机的权重包括阈值能通过学习得到；感知机的输入可以是实数
- 我们将首先回顾一些布尔函数，然后看看感知机学习算法（用于学习权重）

感知机

$$y = 1 \quad \text{if} \sum_{i=0}^n \textcolor{red}{w_i} * x_i \geq 0$$
$$= 0 \quad \text{if} \sum_{i=0}^n \textcolor{red}{w_i} * x_i < 0$$

x_1	x_2	OR
0	0	

x_1	x_2	OR
0	0	0

$$\begin{array}{ccc|c} x_1 & x_2 & \text{OR} \\ \hline 0 & 0 & 0 & w_0 + \sum_{i=1}^2 w_i x_i \end{array}$$

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 \geq -w_0$$



x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 \geq -w_0$$

- 一个可能解是 $w_0 = -1, w_1 = 1.1, w_2 = 1.1$
(还存在其他可能的解)

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$

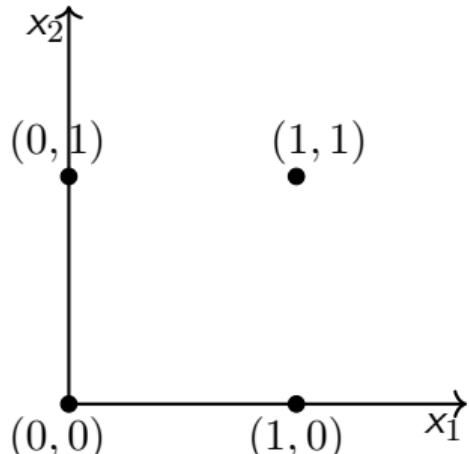
$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

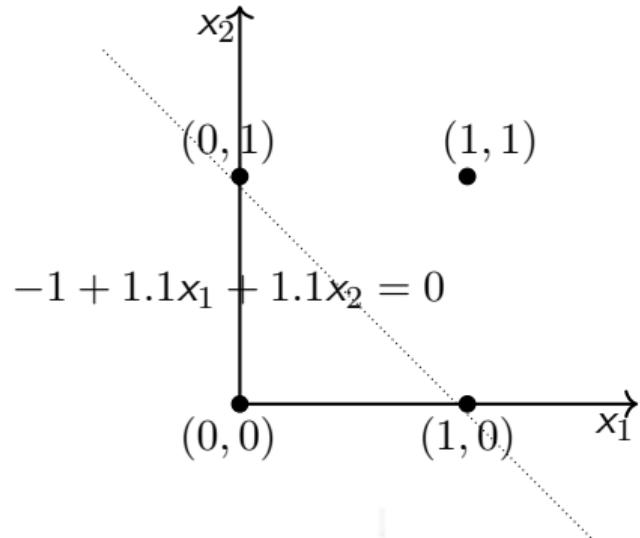
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 \geq -w_0$$

- 一个可能解是 $w_0 = -1, w_1 = 1.1, w_2 = 1.1$
(还存在其他可能的解)



x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$



$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

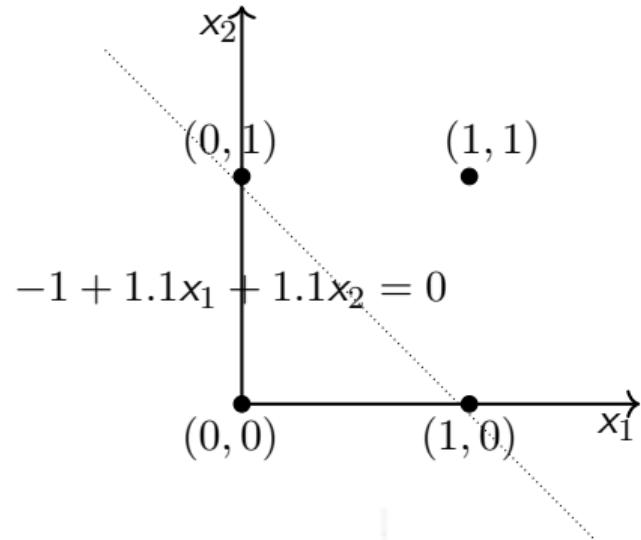
$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 \geq -w_0$$

- 一个可能解是 $w_0 = -1, w_1 = 1.1, w_2 = 1.1$
(还存在其他可能的解)

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$



$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

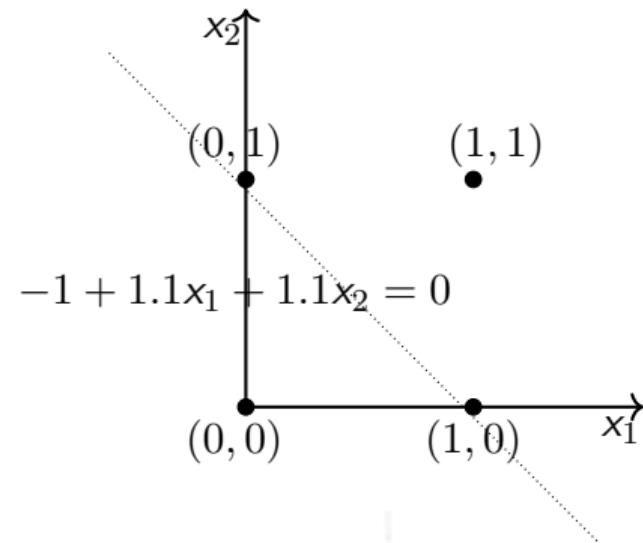
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 \geq -w_0$$

- 一个可能解是 $w_0 = -1, w_1 = 1.1, w_2 = 1.1$
(还存在其他可能的解)

- 注意：M-P 神经元也能够得到相似的不等式方程并且找到一个值 θ

x_1	x_2	OR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$



$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$

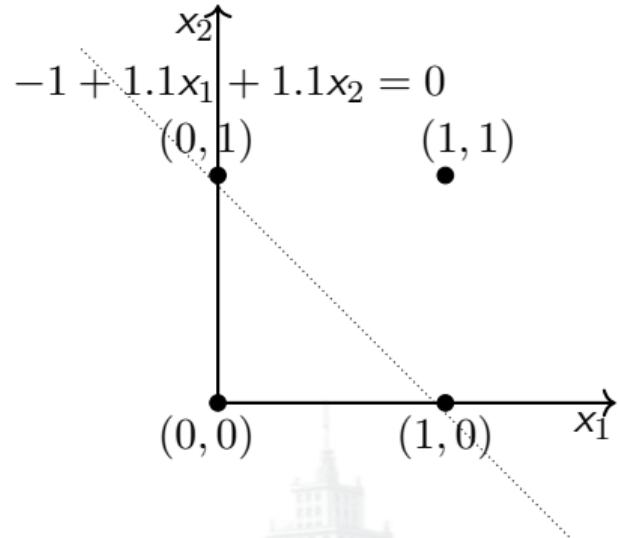
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 \geq 0 \implies w_1 + w_2 \geq -w_0$$

- 一个可能解是 $w_0 = -1, w_1 = 1.1, w_2 = 1.1$
(还存在其他可能的解)

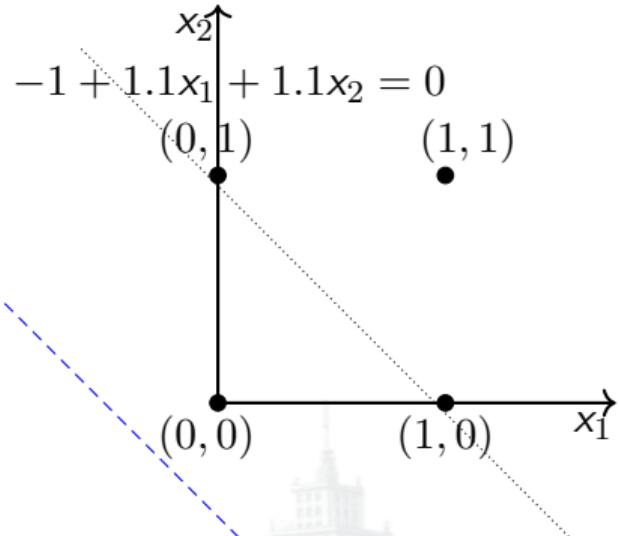
- 注意：M-P 神经元也能够得到相似的不等式方程并且找到一个值 θ (试试看!)

误差和误差曲面

- 固定阈值 ($-w_0 = 1$)，尝试不同的 w_1 和 w_2

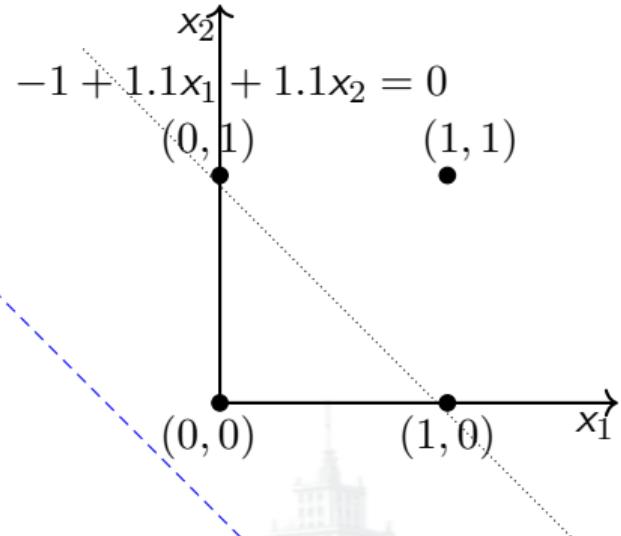


- 固定阈值 ($-w_0 = 1$)，尝试不同的 w_1 和 w_2
- 例如, $w_1 = -1$, $w_2 = -1$



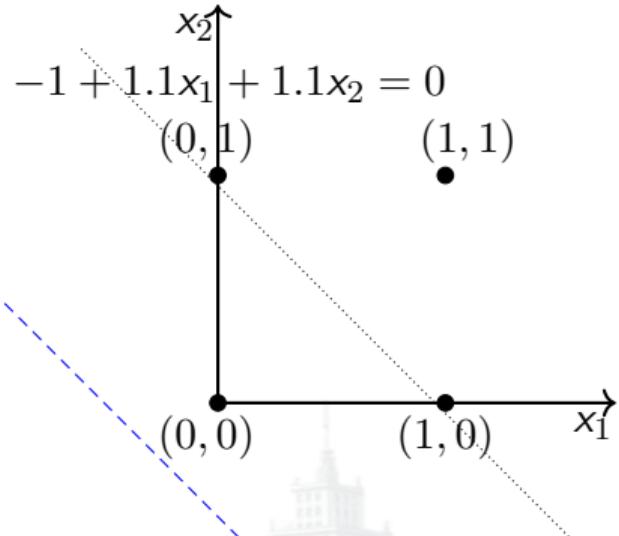


- 固定阈值 ($-w_0 = 1$)，尝试不同的 w_1 和 w_2
- 例如, $w_1 = -1$, $w_2 = -1$
- 这条直线怎么样?

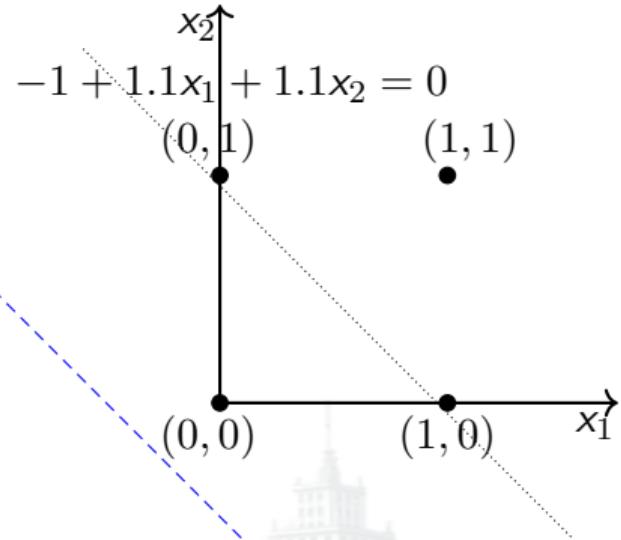


$$-1 + (-1)x_1 + (-1)x_2 = 0$$

- 固定阈值 ($-w_0 = 1$)，尝试不同的 w_1 和 w_2
- 例如, $w_1 = -1$, $w_2 = -1$
- 这条直线怎么样? 有 1 个错误

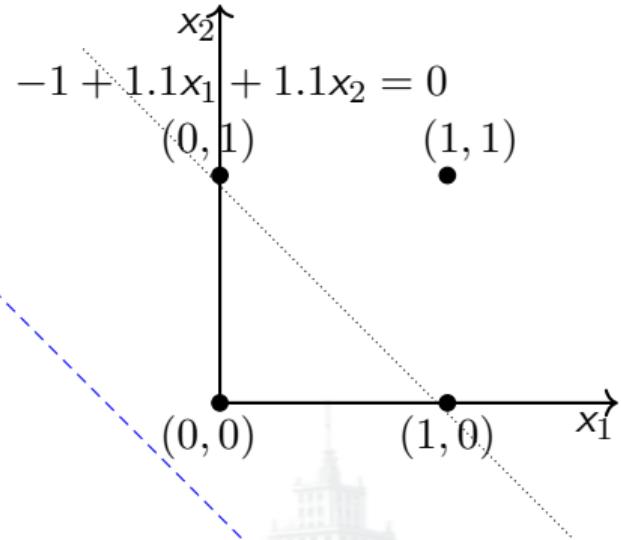


- 固定阈值 ($-w_0 = 1$)，尝试不同的 w_1 和 w_2
- 例如, $w_1 = -1$, $w_2 = -1$
- 这条直线怎么样? 有 1 个错误
- 尝试更多不同的 w_1 和 w_2 , 计算错误



- 固定阈值 ($-w_0 = 1$)，尝试不同的 w_1 和 w_2
- 例如, $w_1 = -1$, $w_2 = -1$
- 这条直线怎么样? 有 1 个错误
- 尝试更多不同的 w_1 和 w_2 , 计算错误

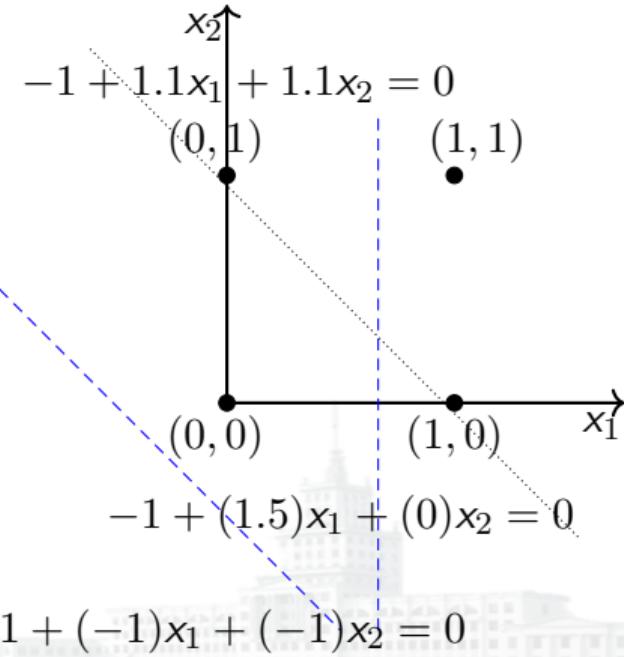
w_1	w_2	errors
-1	-1	3



$$-1 + (-1)x_1 + (-1)x_2 = 0$$

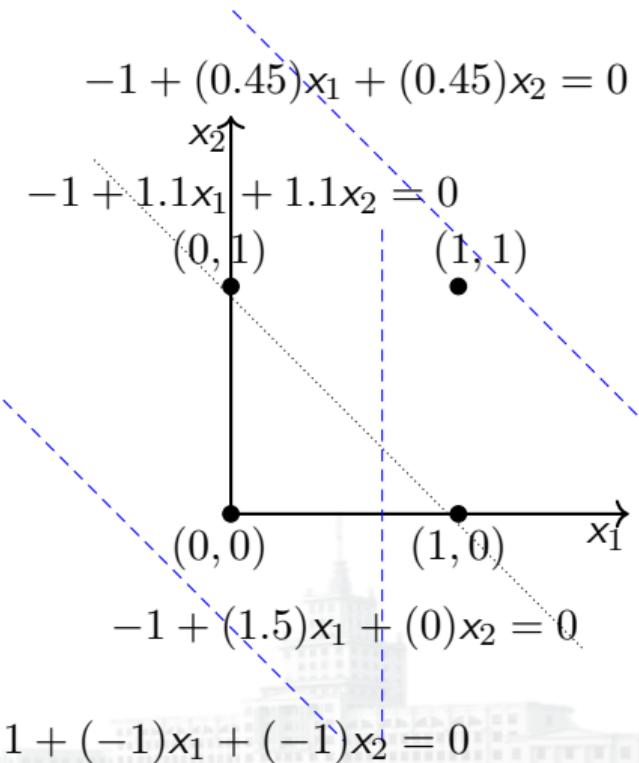
- 固定阈值 ($-w_0 = 1$)，尝试不同的 w_1 和 w_2
- 例如, $w_1 = -1$, $w_2 = -1$
- 这条直线怎么样? 有 1 个错误
- 尝试更多不同的 w_1 和 w_2 , 计算错误

w_1	w_2	errors
-1	-1	3
1.5	0	1



- 固定阈值 ($-w_0 = 1$)，尝试不同的 w_1 和 w_2
- 例如, $w_1 = -1$, $w_2 = -1$
- 这条直线怎么样? 有 1 个错误
- 尝试更多不同的 w_1 和 w_2 , 计算错误

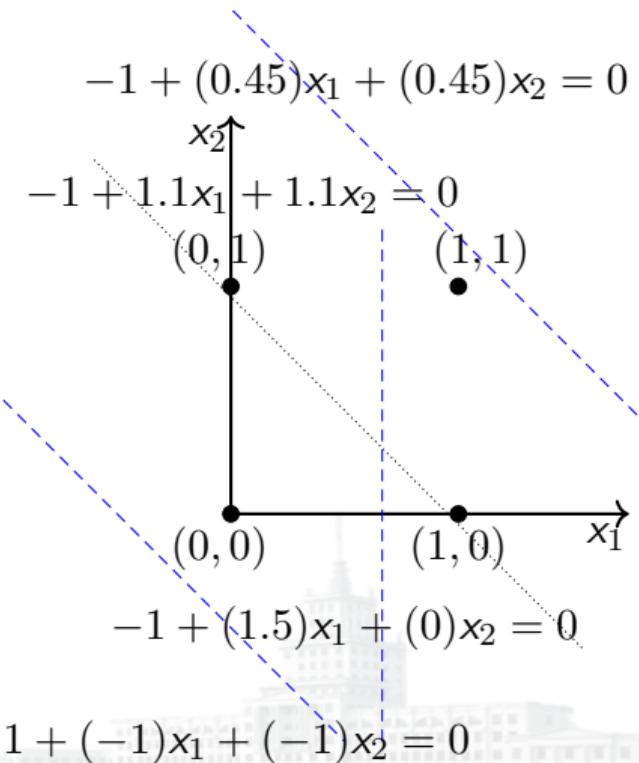
w_1	w_2	errors
-1	-1	3
1.5	0	1
0.45	0.45	3



- 固定阈值 ($-w_0 = 1$)，尝试不同的 w_1 和 w_2
- 例如, $w_1 = -1$, $w_2 = -1$
- 这条直线怎么样? 有 1 个错误
- 尝试更多不同的 w_1 和 w_2 , 计算错误

w_1	w_2	errors
-1	-1	3
1.5	0	1
0.45	0.45	3

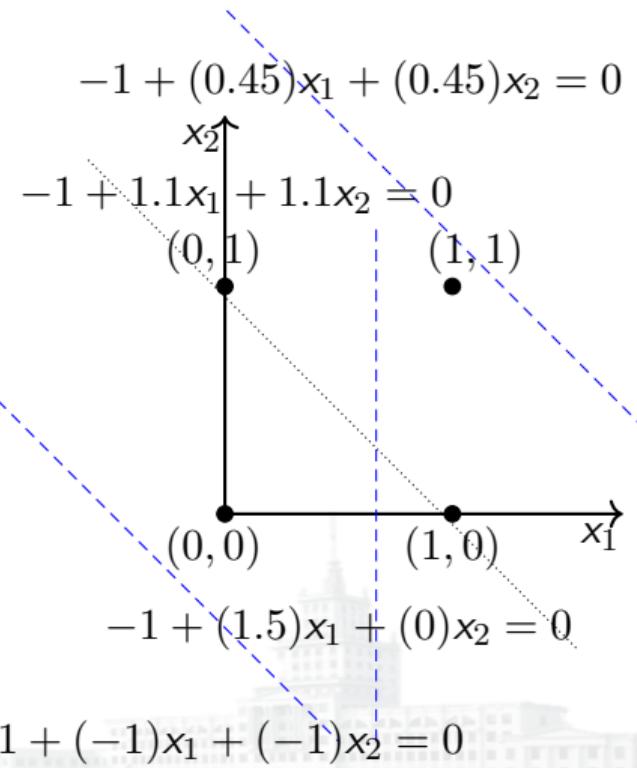
- 目标: 找到能够产生 0 个错误的 w_0 , w_1 和 w_2



- 固定阈值 ($-w_0 = 1$)，尝试不同的 w_1 和 w_2
- 例如, $w_1 = -1, w_2 = -1$
- 这条直线怎么样? 有 1 个错误
- 尝试更多不同的 w_1 和 w_2 , 计算错误

w_1	w_2	errors
-1	-1	3
1.5	0	1
0.45	0.45	3

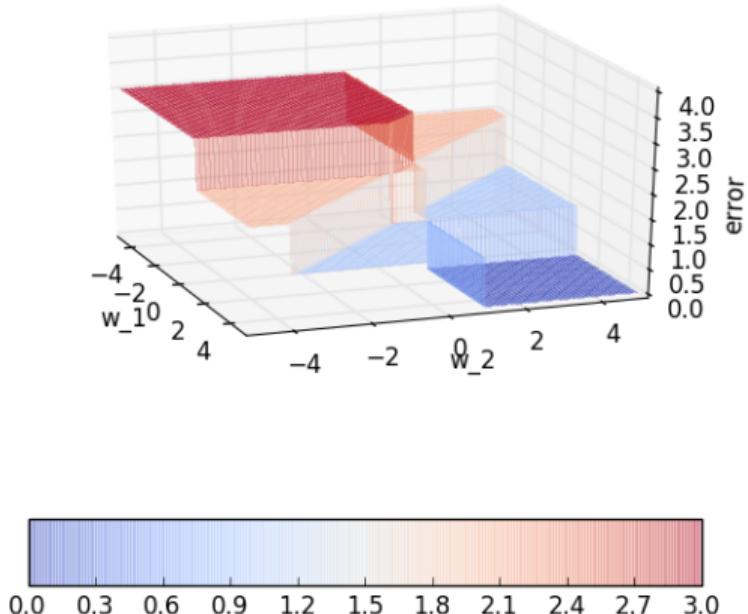
- 目标: 找到能够产生 0 个错误的 w_0, w_1 和 w_2
- 画出错误随 w_0, w_1 和 w_2 变化的图



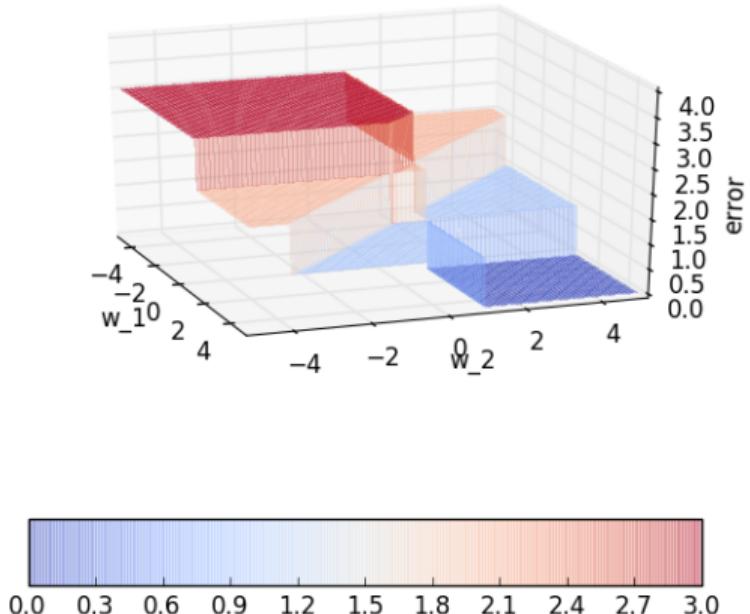
- 方便起见，将 w_0 固定为 (-1)，画出不同 w_1 和 w_2 产生的错误

- 方便起见，将 w_0 固定为 (-1)，画出不同 w_1 和 w_2 产生的错误
- 给定 w_0 , w_1 和 w_2 , 对 (x_1 和 x_2) 的所有组合计算 $-w_0 + w_1 * x_1 + w_2 * x_2$, 从而计算错误

- 方便起见，将 w_0 固定为 (-1)，画出不同 w_1 和 w_2 产生的错误
- 给定 w_0 , w_1 和 w_2 , 对 $(x_1$ 和 $x_2)$ 的所有组合计算 $-w_0 + w_1 * x_1 + w_2 * x_2$, 从而计算错误
- 对于 OR 函数, 当 $(x_1, x_2) = (0, 0)$ 但是 $-w_0 + w_1 * x_1 + w_2 * x_2 \geq 0$ 或当 $(x_1, x_2) \neq (0, 0)$ 但是 $-w_0 + w_1 * x_1 + w_2 * x_2 < 0$ 时, 发生错误



- 方便起见，将 w_0 固定为 (-1)，画出不同 w_1 和 w_2 产生的错误
- 给定 w_0 , w_1 和 w_2 , 对 $(x_1$ 和 $x_2)$ 的所有组合计算 $-w_0 + w_1 * x_1 + w_2 * x_2$, 从而计算错误
- 对于 OR 函数, 当 $(x_1, x_2) = (0, 0)$ 但是 $-w_0 + w_1 * x_1 + w_2 * x_2 \geq 0$ 或当 $(x_1, x_2) \neq (0, 0)$ 但是 $-w_0 + w_1 * x_1 + w_2 * x_2 < 0$ 时, 发生错误



- 方便起见，将 w_0 固定为 (-1)，画出不同 w_1 和 w_2 产生的错误
- 给定 w_0 , w_1 和 w_2 , 对 $(x_1$ 和 $x_2)$ 的所有组合计算 $-w_0 + w_1 * x_1 + w_2 * x_2$, 从而计算错误
- 对于 OR 函数, 当 $(x_1, x_2) = (0, 0)$ 但是 $-w_0 + w_1 * x_1 + w_2 * x_2 \geq 0$ 或当 $(x_1, x_2) \neq (0, 0)$ 但是 $-w_0 + w_1 * x_1 + w_2 * x_2 < 0$ 时, 发生错误
- 我们希望有一个算法, 能找到最小化这个错误的 w_1 和 w_2

感知机学习算法 (Perceptron Learning Algorithm)



- 在了解学习这些权重和阈值的方法之前，先回答这个问题...

- 在了解学习这些权重和阈值的方法之前，先回答这个问题...
- 除了实现布尔函数外，感知机还能用来干什么？



- 在了解学习这些权重和阈值的方法之前，先回答这个问题...
- 除了实现布尔函数外，感知机还能用来干什么？
- 感知机能够用来作为二分类器...

- 回到前面举过的看电影的例子

- 回到前面举过的看电影的例子
- 给定 m 部电影和它们的标签（表明是否喜欢这部电影），这就是一个二分类问题

- 回到前面举过的看电影的例子
- 给定 m 部电影和它们的标签（表明是否喜欢这部电影），这就是一个二分类问题
- 每部电影有 n 个特征（有可能是布尔值，也有可能是实数值）

- 回到前面举过的看电影的例子
- 给定 m 部电影和它们的标签（表明是否喜欢这部电影），这就是一个二分类问题
- 每部电影有 n 个特征（有可能是布尔值，也有可能是实数值）

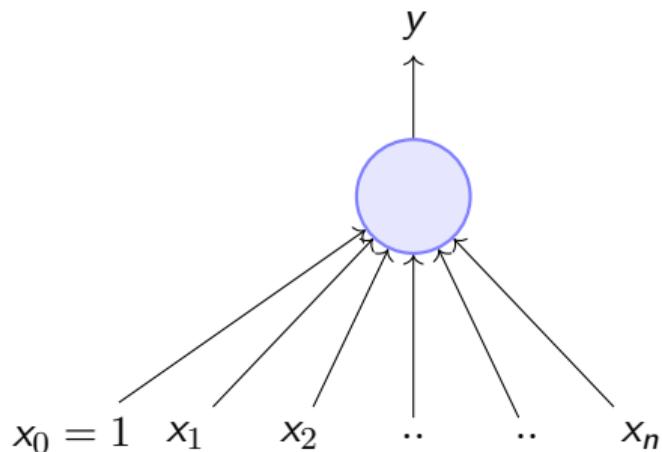
$x_1 = \text{isActorDamon}$

$x_2 = \text{isGenreThriller}$

$x_3 = \text{isDirectorNolan}$

... ...

$x_n = \text{criticsRating}(\text{scaled to } 0 \text{ to } 1)$



$x_1 = \text{isActorDamon}$

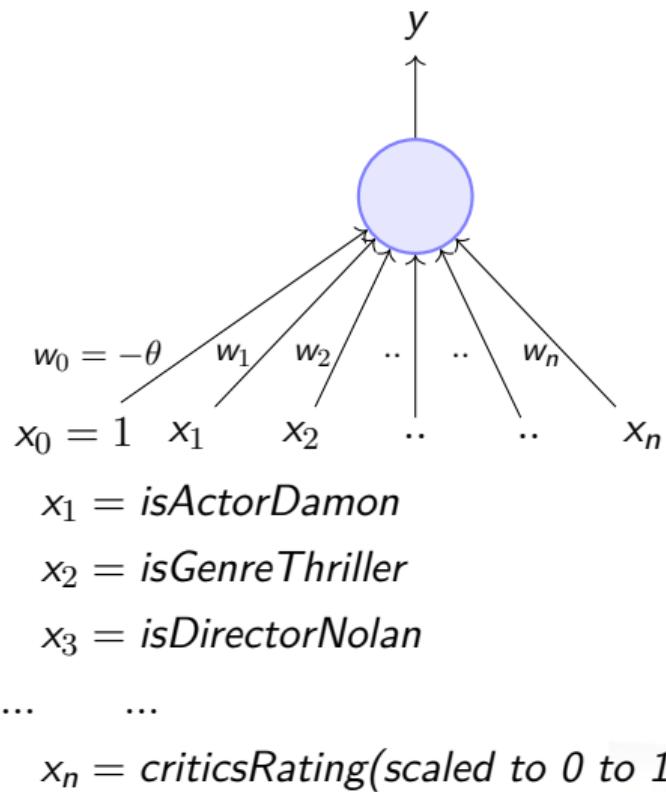
$x_2 = \text{isGenreThriller}$

$x_3 = \text{isDirectorNolan}$

\dots \dots

$x_n = \text{criticsRating}(\text{scaled to } 0 \text{ to } 1)$

- 回到前面举过的看电影的例子
- 给定 m 部电影和它们的标签（表明是否喜欢这部电影），这就是一个二分类问题
- 每部电影有 n 个特征（有可能是布尔值，也有可能是实数值）
- 假定数据是线性可分的，我们希望使用一个感知机来进行二分类



- 回到前面举过的看电影的例子
- 给定 m 部电影和它们的标签（表明是否喜欢这部电影），这就是一个二分类问题
- 每部电影有 n 个特征（有可能是布尔值，也有可能是实数值）
- 假定数据是线性可分的，我们希望使用一个感知机来进行二分类
- 换句话说，我们希望找到一个超平面或找到 $w_0, w_1, w_2, \dots, w_n$ ，使得这个超平面能将训练数据尽可能的分对



Algorithm 1: 感知机学习算法 (Perceptron Learning Algorithm)



Algorithm 2: 感知机学习算法 (Perceptron Learning Algorithm)

$P \leftarrow \text{inputs with label } 1;$



Algorithm 3: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;
```



Algorithm 4: 感知机学习算法 (Perceptron Learning Algorithm)

$P \leftarrow \text{inputs with label } 1;$

$N \leftarrow \text{inputs with label } 0;$

Initialize w randomly;



Algorithm 5: 感知机学习算法 (Perceptron Learning Algorithm)

$P \leftarrow$ inputs with label 1;

$N \leftarrow$ inputs with label 0;

Initialize w randomly;

while !convergence **do**

end

Algorithm 6: 感知机学习算法 (Perceptron Learning Algorithm)

$P \leftarrow \text{inputs with label } 1;$

$N \leftarrow \text{inputs with label } 0;$

Initialize w randomly;

while !convergence **do**

end

//当所有的输入被正确分类时，算法收敛

Algorithm 7: 感知机学习算法 (Perceptron Learning Algorithm)

$P \leftarrow \text{inputs with label } 1;$

$N \leftarrow \text{inputs with label } 0;$

Initialize w randomly;

while !convergence **do**

Pick random $x \in P \cup N$;

end

//当所有的输入被正确分类时，算法收敛

Algorithm 8: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and  $\sum_{i=0}^n w_i * x_i < 0$  then  
        |  
        |  
        end  
    |  
end  
//当所有的输入被正确分类时， 算法收敛
```

Algorithm 9: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and  $\sum_{i=0}^n w_i * x_i < 0$  then  
        | w = w + x ;  
    end  
  
end  
//当所有的输入被正确分类时， 算法收敛
```

Algorithm 10: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and  $\sum_{i=0}^n w_i * x_i < 0$  then  
        | w = w + x ;  
    end  
    if x ∈ N and  $\sum_{i=0}^n w_i * x_i \geq 0$  then  
        |  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

Algorithm 11: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and  $\sum_{i=0}^n w_i * x_i < 0$  then  
        | w = w + x ;  
    end  
    if x ∈ N and  $\sum_{i=0}^n w_i * x_i \geq 0$  then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

Algorithm 12: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and  $\sum_{i=0}^n w_i * x_i < 0$  then  
        | w = w + x ;  
    end  
    if x ∈ N and  $\sum_{i=0}^n w_i * x_i \geq 0$  then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 为什么这个算法能有效?

Algorithm 13: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and  $\sum_{i=0}^n w_i * x_i < 0$  then  
        | w = w + x ;  
    end  
    if x ∈ N and  $\sum_{i=0}^n w_i * x_i \geq 0$  then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 为什么这个算法能有效?
- 回答这个问题，先回顾一些线性代数和几何的知识...



- 考虑两个向量 w 和 x



- 考虑两个向量 w 和 x

$$w = [w_0, w_1, w_2, \dots, w_n]$$

$$x = [1, x_1, x_2, \dots, x_n]$$



- 考虑两个向量 w 和 x

$$w = [w_0, w_1, w_2, \dots, w_n]$$

$$x = [1, x_1, x_2, \dots, x_n]$$

$$w \cdot x = w^T x = \sum_{i=0}^n w_i * x_i$$



- 考虑两个向量 w 和 x

$$w = [w_0, w_1, w_2, \dots, w_n]$$

$$x = [1, x_1, x_2, \dots, x_n]$$

$$w \cdot x = w^T x = \sum_{i=0}^n w_i * x_i$$

- 感知机规则可以重写为



- 考虑两个向量 w 和 x

$$w = [w_0, w_1, w_2, \dots, w_n]$$

$$x = [1, x_1, x_2, \dots, x_n]$$

$$w \cdot x = w^T x = \sum_{i=0}^n w_i * x_i$$

- 感知机规则可以重写为

$$y = 1 \quad if \quad w^T x \geq 0$$

$$= 0 \quad if \quad w^T x < 0$$



- 考虑两个向量 w 和 x

$$w = [w_0, w_1, w_2, \dots, w_n]$$

$$x = [1, x_1, x_2, \dots, x_n]$$

$$w \cdot x = w^T x = \sum_{i=0}^n w_i * x_i$$

- 我们希望找到一条直线 $w^T x = 0$ 能够将输入空间划分成两部分

- 感知机规则可以重写为

$$y = 1 \quad if \quad w^T x \geq 0$$

$$= 0 \quad if \quad w^T x < 0$$

- 考虑两个向量 w 和 x

$$w = [w_0, w_1, w_2, \dots, w_n]$$

$$x = [1, x_1, x_2, \dots, x_n]$$

$$w \cdot x = w^T x = \sum_{i=0}^n w_i * x_i$$

- 我们希望找到一条直线 $w^T x = 0$ 能够将输入空间划分成两部分
- 直线上的每一个点 x 满足等式 $w^T x = 0$

- 感知机规则可以重写为

$$y = 1 \quad if \quad w^T x \geq 0$$

$$= 0 \quad if \quad w^T x < 0$$

- 考虑两个向量 w 和 x

$$w = [w_0, w_1, w_2, \dots, w_n]$$

$$x = [1, x_1, x_2, \dots, x_n]$$

$$w \cdot x = w^T x = \sum_{i=0}^n w_i * x_i$$

- 我们希望找到一条直线 $w^T x = 0$ 能够将输入空间划分成两部分
- 直线上的每一个点 x 满足等式 $w^T x = 0$
- 想一想，直线上的任意一个点 x 和权重向量 w 之间的角度 α 是多少？

- 感知机规则可以重写为

$$\begin{aligned} y &= 1 \quad if \quad w^T x \geq 0 \\ &= 0 \quad if \quad w^T x < 0 \end{aligned}$$

- 考虑两个向量 w 和 x

$$w = [w_0, w_1, w_2, \dots, w_n]$$

$$x = [1, x_1, x_2, \dots, x_n]$$

$$w \cdot x = w^T x = \sum_{i=0}^n w_i * x_i$$

- 感知机规则可以重写为

$$\begin{aligned} y &= 1 \quad \text{if} \quad w^T x \geq 0 \\ &= 0 \quad \text{if} \quad w^T x < 0 \end{aligned}$$

- 我们希望找到一条直线 $w^T x = 0$ 能够将输入空间划分成两部分
- 直线上的每一个点 x 满足等式 $w^T x = 0$
- 想一想，直线上的任意一个点 x 和权重向量 w 之间的角度 α 是多少？
- α 是 90° ($\because \cos\alpha = \frac{w^T x}{\|w\|\|x\|} = 0$)

- 考虑两个向量 w 和 x

$$w = [w_0, w_1, w_2, \dots, w_n]$$

$$x = [1, x_1, x_2, \dots, x_n]$$

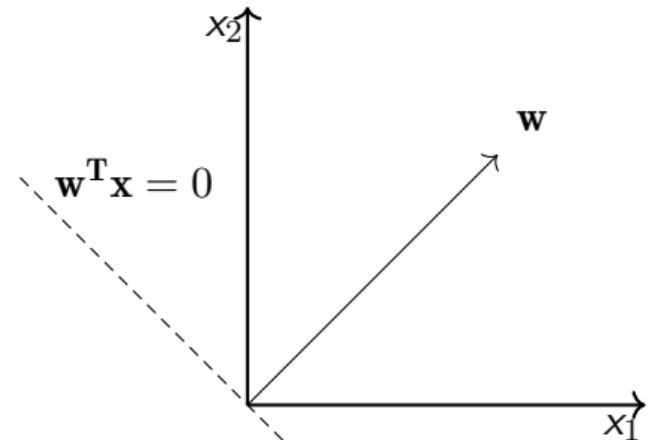
$$w \cdot x = w^T x = \sum_{i=0}^n w_i * x_i$$

- 感知机规则可以重写为

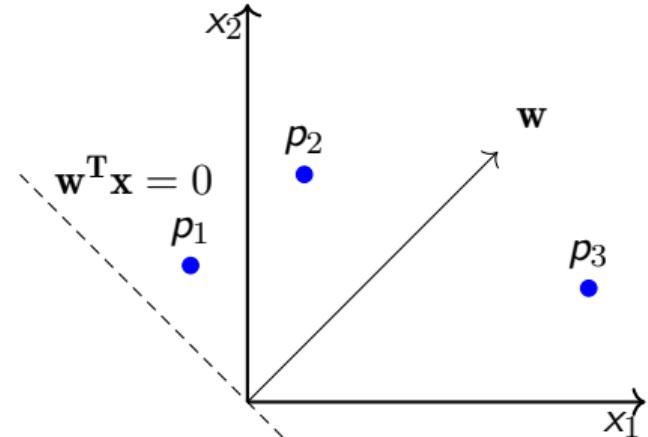
$$y = 1 \quad if \quad w^T x \geq 0$$

$$= 0 \quad if \quad w^T x < 0$$

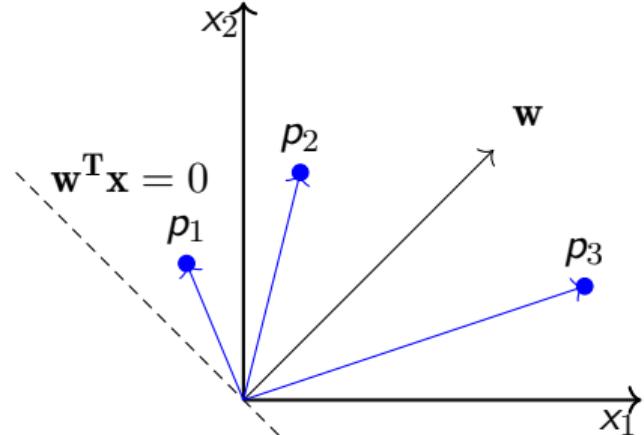
- 我们希望找到一条直线 $w^T x = 0$ 能够将输入空间划分成两部分
- 直线上的每一个点 x 满足等式 $w^T x = 0$
- 想一想，直线上的任意一个点 x 和权重向量 w 之间的角度 α 是多少？
- α 是 90° ($\because \cos\alpha = \frac{w^T x}{\|w\|\|x\|} = 0$)
- 因为权重向量 w 正交于直线上的每一个点， w 实际上正交于直线本身



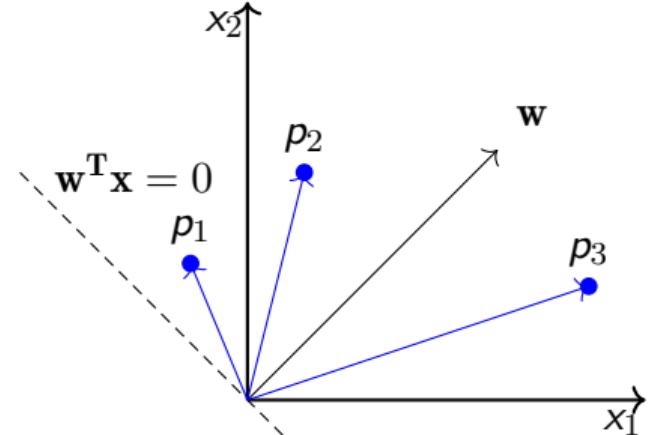
- 考虑那些标签为正的样本点 (i.e., $\mathbf{w}^T \mathbf{x} \geq 0$)



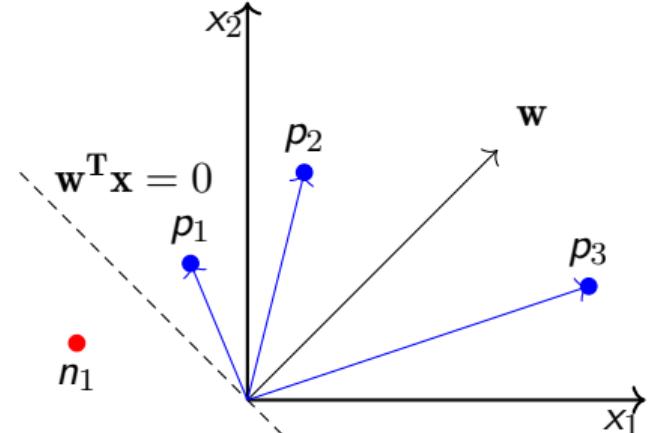
- 考虑那些标签为正的样本点 (i.e., $\mathbf{w}^T \mathbf{x} \geq 0$)
- 这些样本点和 \mathbf{w} 的角度是多少?



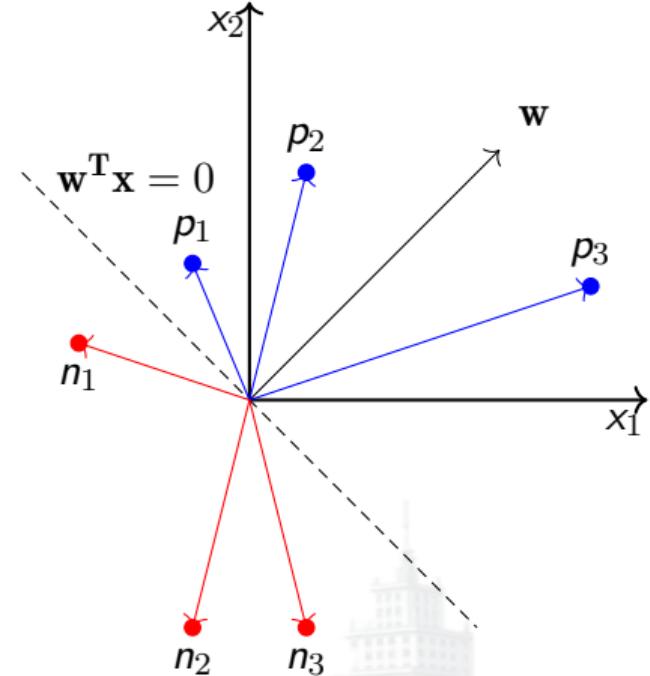
- 考虑那些标签为正的样本点 (*i.e.*, $\mathbf{w}^T \mathbf{x} \geq 0$)
- 这些样本点和 \mathbf{w} 的角度是多少? 显然, 小于 90°



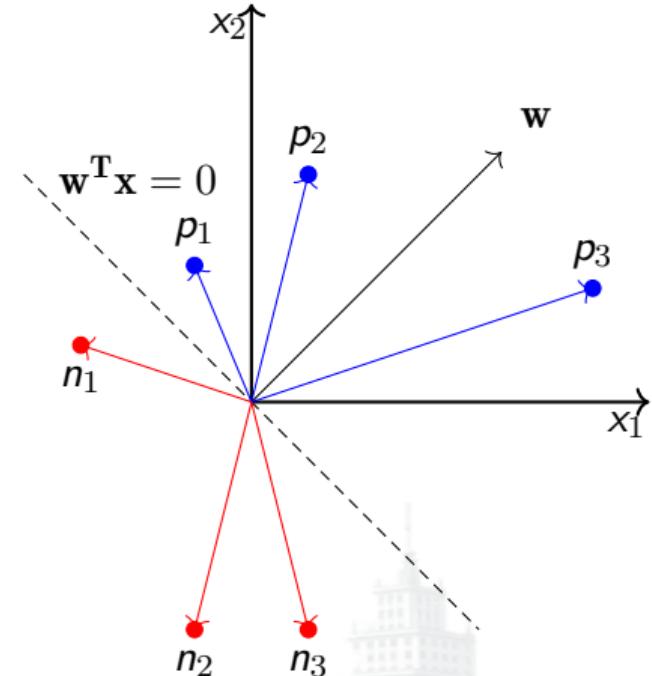
- 考虑那些标签为正的样本点 (*i.e.*, $\mathbf{w}^T \mathbf{x} \geq 0$)
- 这些样本点和 \mathbf{w} 的角度是多少? 显然, 小于 90°
- 对于那些标签为负的样本点 (*i.e.*, $\mathbf{w}^T \mathbf{x} < 0$)



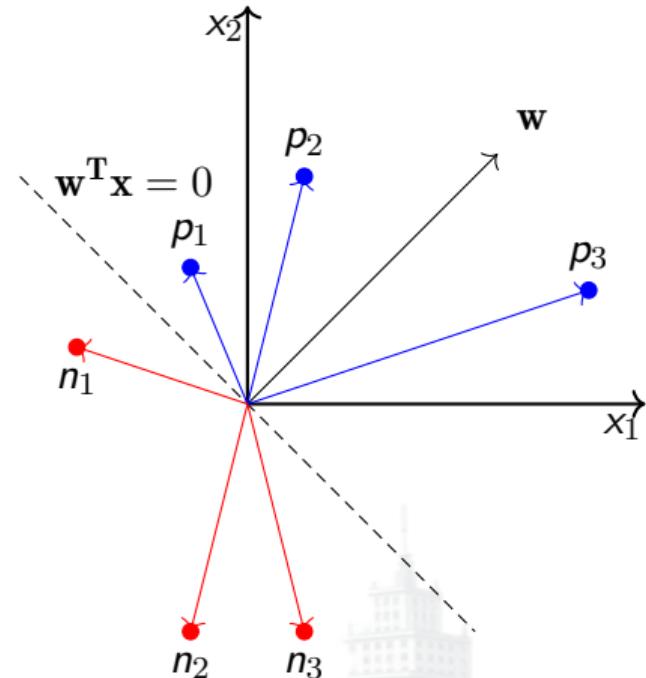
- 考虑那些标签为正的样本点 (*i.e.*, $\mathbf{w}^T \mathbf{x} \geq 0$)
- 这些样本点和 \mathbf{w} 的角度是多少? 显然, 小于 90°
- 对于那些标签为负的样本点 (*i.e.*, $\mathbf{w}^T \mathbf{x} < 0$)
- 这些样本点和 \mathbf{w} 的角度是多少?



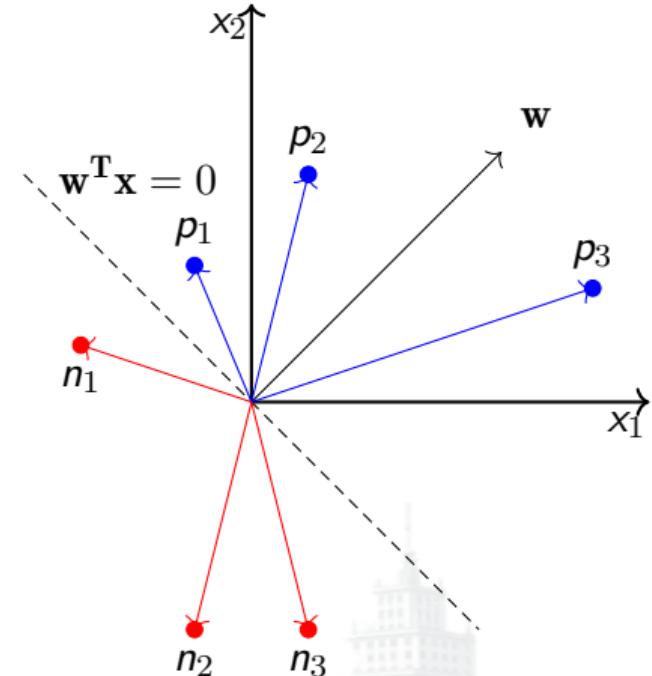
- 考虑那些标签为正的样本点 (*i.e.*, $\mathbf{w}^T \mathbf{x} \geq 0$)
- 这些样本点和 \mathbf{w} 的角度是多少? 显然, 小于 90°
- 对于那些标签为负的样本点 (*i.e.*, $\mathbf{w}^T \mathbf{x} < 0$)
- 这些样本点和 \mathbf{w} 的角度是多少? 显然, 大于 90°



- 考虑那些标签为正的样本点 (*i.e.*, $\mathbf{w}^T \mathbf{x} \geq 0$)
- 这些样本点和 \mathbf{w} 的角度是多少? 显然, 小于 90°
- 对于那些标签为负的样本点 (*i.e.*, $\mathbf{w}^T \mathbf{x} < 0$)
- 这些样本点和 \mathbf{w} 的角度是多少? 显然, 大于 90°
- 这些点与 \mathbf{w} 的角度也满足公式 $\cos\alpha = \frac{\mathbf{w}^T \mathbf{x}}{\|\mathbf{w}\| \|\mathbf{x}\|}$



- 考虑那些标签为正的样本点 (*i.e.*, $w^T x \geq 0$)
- 这些样本点和 w 的角度是多少? 显然, 小于 90°
- 对于那些标签为负的样本点 (*i.e.*, $w^T x < 0$)
- 这些样本点和 w 的角度是多少? 显然, 大于 90°
- 这些点与 w 的角度也满足公式 $\cos\alpha = \frac{w^T x}{\|w\|\|x\|}$
- 记住这些, 再回到感知机学习算法



Algorithm 14: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end
```

//当所有的输入被正确分类时， 算法收敛

Algorithm 15: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 对于 $x \in P$ 如果 $w \cdot x < 0$, 则 x 当前 w 的角度 α 大于 90°

Algorithm 16: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 对于 $x \in P$ 如果 $w \cdot x < 0$, 则 x 当前 w 的角度 α 大于 90° (但是我们希望 α 小于 90°)

Algorithm 17: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end
```

//当所有的输入被正确分类时， 算法收敛

- 对于 $x \in P$ 如果 $w \cdot x < 0$, 则 x 当前 w 的角度 α 大于 90° (但是我们希望 α 小于 90°)
- 当 $w_{new} = w + x$ 时, 新的 α_{new} 将会如何?

Algorithm 18: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 对于 $x \in P$ 如果 $w \cdot x < 0$, 则 x 当前 w 的角度 α 大于 90° (但是我们希望 α 小于 90°)
- 当 $w_{new} = w + x$ 时, 新的 α_{new} 将会如何?

$$\cos(\alpha_{new}) \propto w_{new}^T x$$

Algorithm 19: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 对于 $x \in P$ 如果 $w \cdot x < 0$, 则 x 当前 w 的角度 α 大于 90° (但是我们希望 α 小于 90°)
- 当 $w_{new} = w + x$ 时, 新的 α_{new} 将会如何?

$$\cos(\alpha_{new}) \propto w_{new}^T x \\ \propto (w + x)^T x$$

Algorithm 20: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 对于 $x \in P$ 如果 $w \cdot x < 0$, 则 x 当前 w 的角度 α 大于 90° (但是我们希望 α 小于 90°)
- 当 $w_{new} = w + x$ 时, 新的 α_{new} 将会如何?

$$\begin{aligned}\cos(\alpha_{new}) &\propto w_{new}^T x \\ &\propto (w + x)^T x \\ &\propto w^T x + x^T x\end{aligned}$$

Algorithm 21: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 对于 $x \in P$ 如果 $w \cdot x < 0$, 则 x 当前 w 的角度 α 大于 90° (但是我们希望 α 小于 90°)
- 当 $w_{new} = w + x$ 时, 新的 α_{new} 将会如何?

$$\begin{aligned}\cos(\alpha_{new}) &\propto w_{new}^T x \\ &\propto (w + x)^T x \\ &\propto w^T x + x^T x \\ &\propto \cos\alpha + x^T x\end{aligned}$$

Algorithm 22: 感知机学习算法 (Perceptron Learning Algorithm)

```

 $P \leftarrow$  inputs with label 1;
 $N \leftarrow$  inputs with label 0;
Initialize  $w$  randomly;
while !convergence do
    Pick random  $x \in P \cup N$  ;
    if  $x \in P$  and  $w \cdot x < 0$  then
        |  $w = w + x$  ;
    end
    if  $x \in N$  and  $w \cdot x \geq 0$  then
        |  $w = w - x$  ;
    end
end

```

//当所有的输入被正确分类时， 算法收敛

- 对于 $x \in P$ 如果 $w \cdot x < 0$, 则 x 当前 w 的角度 α 大于 90° (但是我们希望 α 小于 90°)
- 当 $w_{new} = w + x$ 时, 新的 α_{new} 将会如何?

$$\begin{aligned}
 \cos(\alpha_{new}) &\propto w_{new}^T x \\
 &\propto (w + x)^T x \\
 &\propto w^T x + x^T x \\
 &\propto \cos\alpha + x^T x
 \end{aligned}$$

$$\cos(\alpha_{new}) > \cos\alpha$$

Algorithm 23: 感知机学习算法 (Perceptron Learning Algorithm)

```

 $P \leftarrow \text{inputs with label } 1;$ 
 $N \leftarrow \text{inputs with label } 0;$ 
Initialize  $w$  randomly;
while !convergence do
    Pick random  $x \in P \cup N$  ;
    if  $x \in P$  and  $w \cdot x < 0$  then
        |  $w = w + x$  ;
    end
    if  $x \in N$  and  $w \cdot x \geq 0$  then
        |  $w = w - x$  ;
    end
end

```

//当所有的输入被正确分类时，算法收敛

- 对于 $x \in P$ 如果 $w \cdot x < 0$, 则 x 当前 w 的角度 α 大于 90° (但是我们希望 α 小于 90°)
- 当 $w_{\text{new}} = w + x$ 时, 新的 α_{new} 将会如何?

$$\begin{aligned}
 \cos(\alpha_{\text{new}}) &\propto w_{\text{new}}^T x \\
 &\propto (w + x)^T x \\
 &\propto w^T x + x^T x \\
 &\propto \cos\alpha + x^T x
 \end{aligned}$$

$$\cos(\alpha_{\text{new}}) > \cos\alpha$$

- 因此, α_{new} 将小于 α , 这正是我们期望的

Algorithm 24: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end
```

//当所有的输入被正确分类时， 算法收敛

Algorithm 25: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 对于 $x \in N$ 如果 $w \cdot x \geq 0$, x 与当前 w 的角度 α 小于 90°

Algorithm 26: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 对于 $x \in N$ 如果 $w \cdot x \geq 0$, x 与当前 w 的角度 α 小于 90° (但是我们期望 α 大于 90°)

Algorithm 27: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 对于 $x \in N$ 如果 $w \cdot x \geq 0$, x 与当前 w 的角度 α 小于 90° (但是我们期望 α 大于 90°)
- 当更新权重 $w_{new} = w - x$ 之后, 得到新的角度 (α_{new}) 又如何呢?

Algorithm 28: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 对于 $x \in N$ 如果 $w \cdot x \geq 0$, x 与当前 w 的角度 α 小于 90° (但是我们期望 α 大于 90°)
- 当更新权重 $w_{\text{new}} = w - x$ 之后, 得到新的角度 (α_{new}) 又如何呢?

$$\cos(\alpha_{\text{new}}) \propto w_{\text{new}}^T x$$

Algorithm 29: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 对于 $x \in N$ 如果 $w \cdot x \geq 0$, x 与当前 w 的角度 α 小于 90° (但是我们期望 α 大于 90°)
- 当更新权重 $w_{\text{new}} = w - x$ 之后, 得到新的角度 (α_{new}) 又如何呢?

$$\begin{aligned} \cos(\alpha_{\text{new}}) &\propto w_{\text{new}}^T x \\ &\propto (w - x)^T x \end{aligned}$$

Algorithm 30: 感知机学习算法 (Perceptron Learning Algorithm)

```
P ← inputs with label 1;  
N ← inputs with label 0;  
Initialize w randomly;  
while !convergence do  
    Pick random x ∈ P ∪ N ;  
    if x ∈ P and w.x < 0 then  
        | w = w + x ;  
    end  
    if x ∈ N and w.x ≥ 0 then  
        | w = w - x ;  
    end  
end  
//当所有的输入被正确分类时， 算法收敛
```

- 对于 $x \in N$ 如果 $w \cdot x \geq 0$, x 与当前 w 的角度 α 小于 90° (但是我们期望 α 大于 90°)
- 当更新权重 $w_{\text{new}} = w - x$ 之后, 得到新的角度 (α_{new}) 又如何呢?

$$\begin{aligned} \cos(\alpha_{\text{new}}) &\propto w_{\text{new}}^T x \\ &\propto (w - x)^T x \\ &\propto w^T x - x^T x \end{aligned}$$

Algorithm 31: 感知机学习算法 (Perceptron Learning Algorithm)

```

 $P \leftarrow$  inputs with label 1;
 $N \leftarrow$  inputs with label 0;
Initialize  $w$  randomly;
while !convergence do
    Pick random  $x \in P \cup N$  ;
    if  $x \in P$  and  $w \cdot x < 0$  then
         $w = w + x$  ;
    end
    if  $x \in N$  and  $w \cdot x \geq 0$  then
         $w = w - x$  ;
    end
end
//当所有的输入被正确分类时, 算法收敛

```

- 对于 $x \in N$ 如果 $w \cdot x \geq 0$, x 与当前 w 的角度 α 小于 90° (但是我们期望 α 大于 90°)
- 当更新权重 $w_{\text{new}} = w - x$ 之后, 得到新的角度 (α_{new}) 又如何呢?

$$\begin{aligned} \cos(\alpha_{\text{new}}) &\propto w_{\text{new}}^T x \\ &\propto (w - x)^T x \\ &\propto w^T x - x^T x \\ &\propto \cos\alpha - x^T x \end{aligned}$$

Algorithm 32: 感知机学习算法 (Perceptron Learning Algorithm)

```

 $P \leftarrow \text{inputs with label } 1;$ 
 $N \leftarrow \text{inputs with label } 0;$ 
Initialize  $w$  randomly;
while !convergence do
    Pick random  $x \in P \cup N$  ;
    if  $x \in P$  and  $w \cdot x < 0$  then
         $w = w + x$  ;
    end
    if  $x \in N$  and  $w \cdot x \geq 0$  then
         $w = w - x$  ;
    end
end
//当所有的输入被正确分类时, 算法收敛

```

- 对于 $x \in N$ 如果 $w \cdot x \geq 0$, x 与当前 w 的角度 α 小于 90° (但是我们期望 α 大于 90°)
- 当更新权重 $w_{\text{new}} = w - x$ 之后, 得到新的角度 (α_{new}) 又如何呢?

$$\begin{aligned}
 \cos(\alpha_{\text{new}}) &\propto w_{\text{new}}^T x \\
 &\propto (w - x)^T x \\
 &\propto w^T x - x^T x \\
 &\propto \cos\alpha - x^T x
 \end{aligned}$$

$$\cos(\alpha_{\text{new}}) < \cos\alpha$$

Algorithm 33: 感知机学习算法 (Perceptron Learning Algorithm)

```

 $P \leftarrow$  inputs with label 1;
 $N \leftarrow$  inputs with label 0;
Initialize  $w$  randomly;
while !convergence do
    Pick random  $x \in P \cup N$  ;
    if  $x \in P$  and  $w \cdot x < 0$  then
        |  $w = w + x$  ;
    end
    if  $x \in N$  and  $w \cdot x \geq 0$  then
        |  $w = w - x$  ;
    end
end

```

//当所有的输入被正确分类时，算法收敛

- 对于 $x \in N$ 如果 $w \cdot x \geq 0$, x 与当前 w 的角度 α 小于 90° (但是我们期望 α 大于 90°)
- 当更新权重 $w_{\text{new}} = w - x$ 之后, 得到新的角度 (α_{new}) 又如何呢?

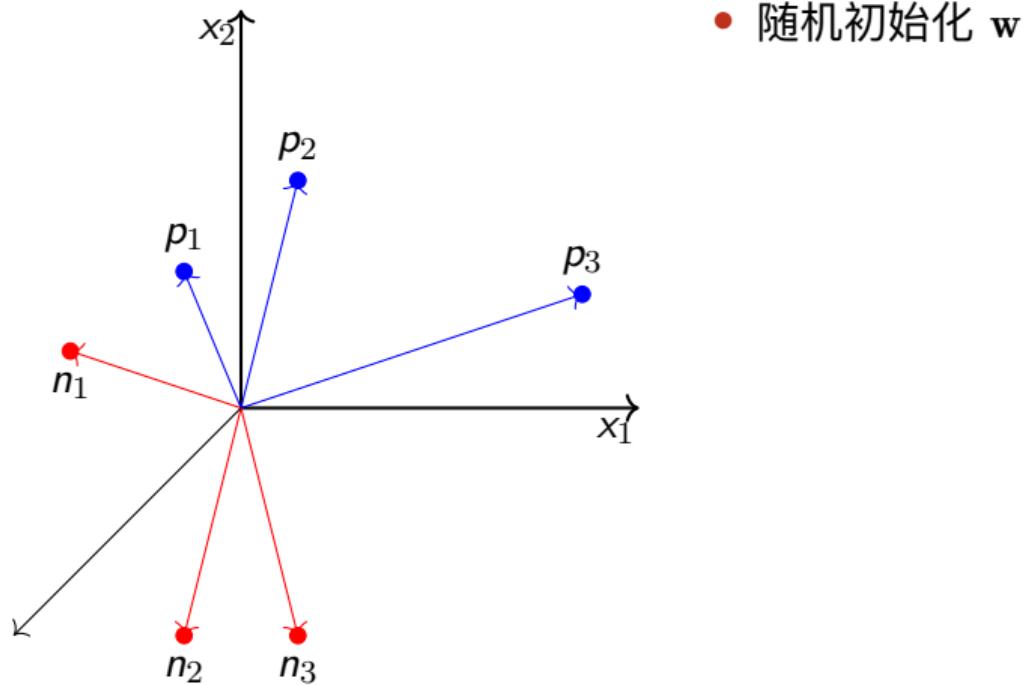
$$\begin{aligned}
 \cos(\alpha_{\text{new}}) &\propto w_{\text{new}}^T x \\
 &\propto (w - x)^T x \\
 &\propto w^T x - x^T x \\
 &\propto \cos\alpha - x^T x
 \end{aligned}$$

$$\cos(\alpha_{\text{new}}) < \cos\alpha$$

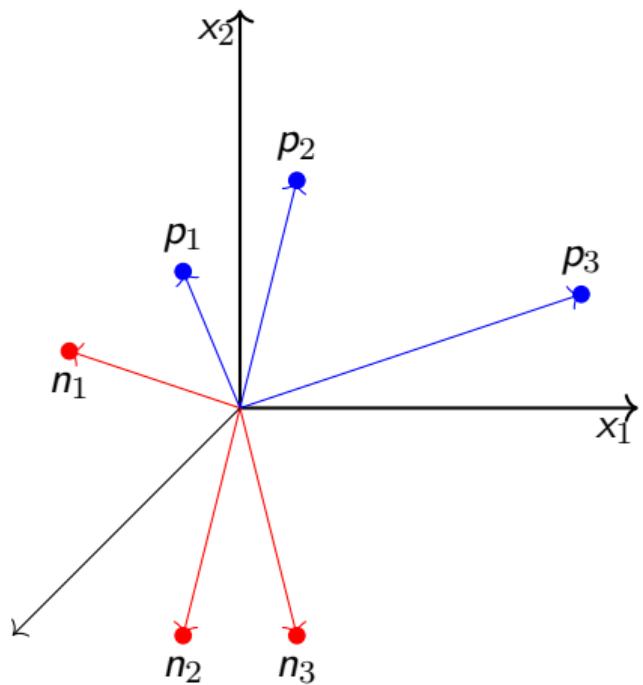
- 因此, α_{new} 将大于 α , 这正是我们期望的



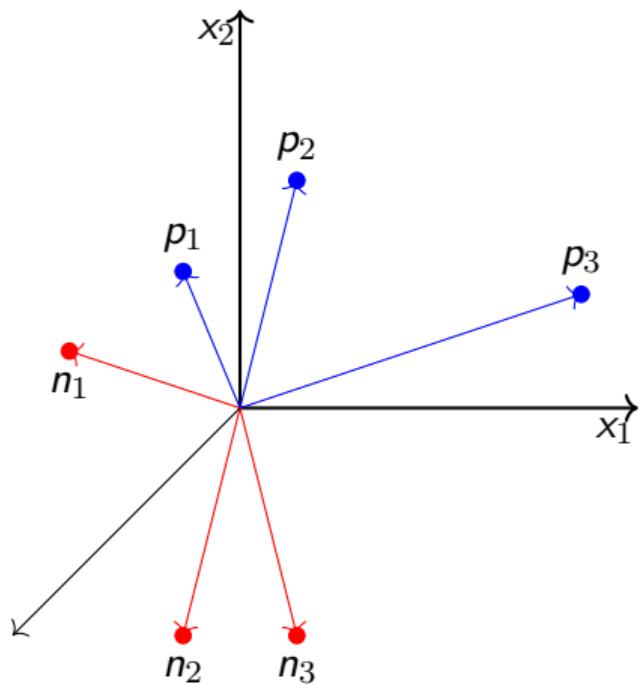
- 通过一个例子来看感知机学习算法是如何工作的



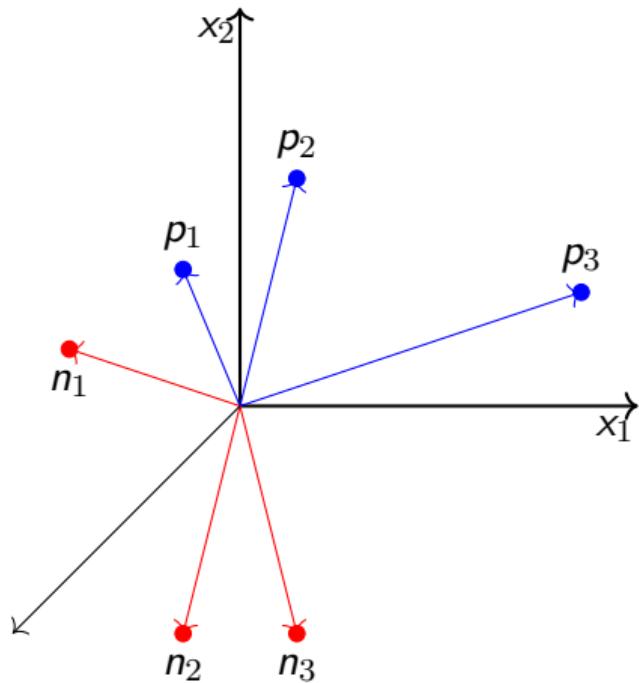
● 随机初始化 w



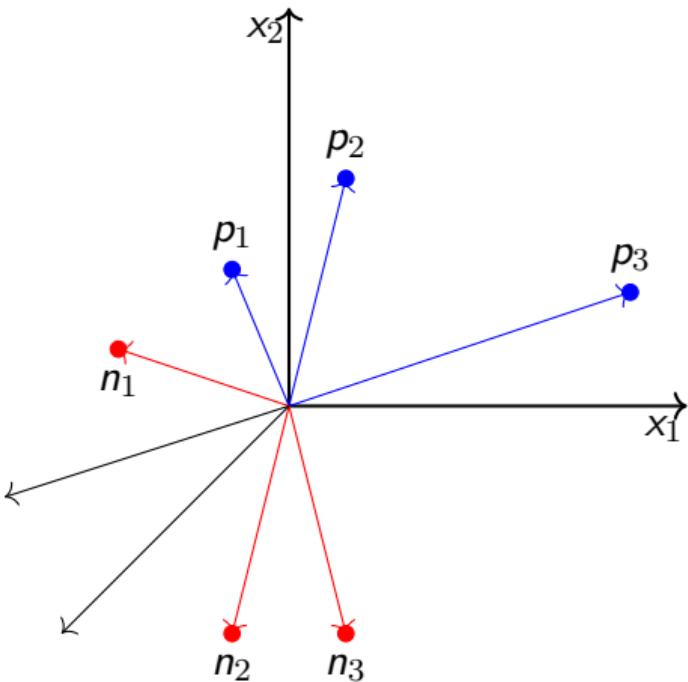
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)



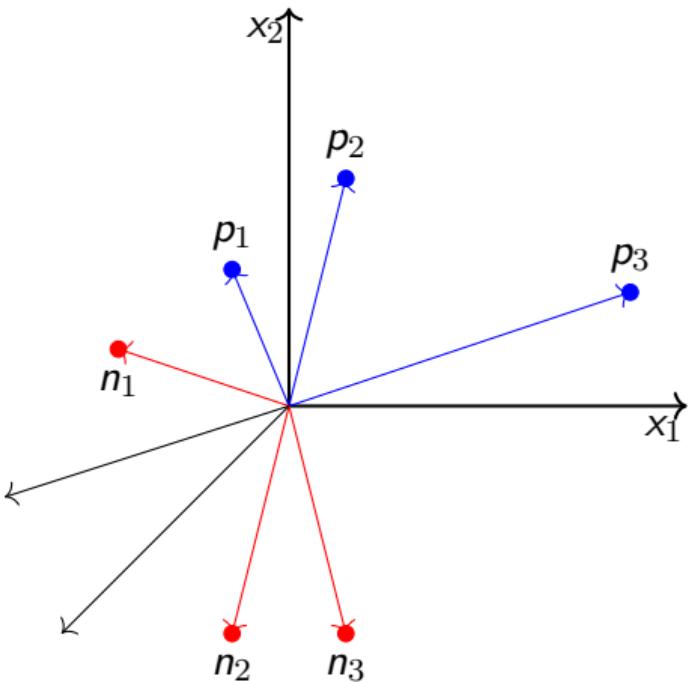
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点



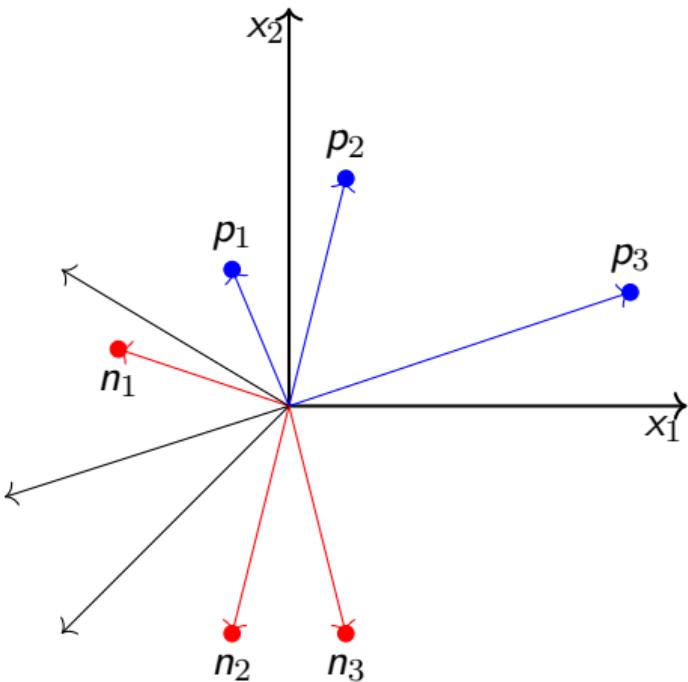
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, p_1), 对权重进行更新 $w = w + x \because w \cdot x < 0$ (看看角度如何变化)



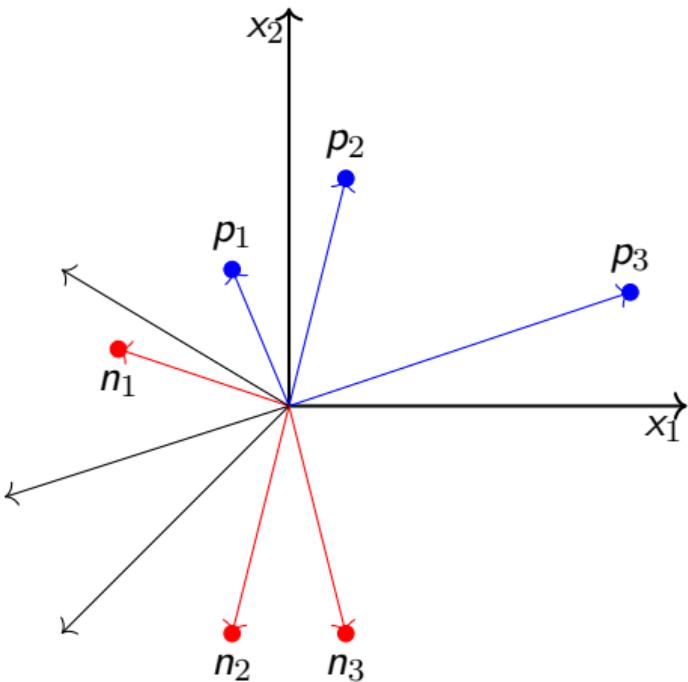
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, p_1), 对权重进行更新 $w = w + x \because w \cdot x < 0$ (看看角度如何变化)



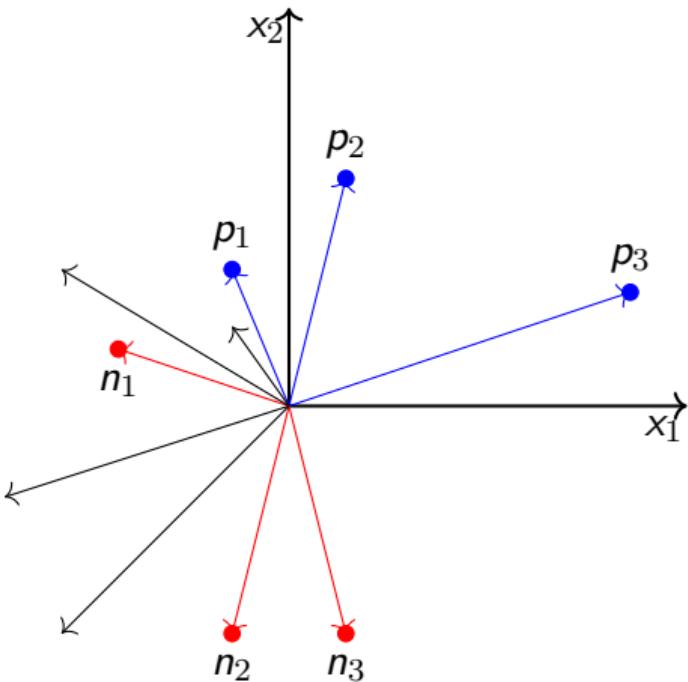
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, p_2), 对权重进行更新 $w = w + x \because w \cdot x < 0$ (看看角度如何变化)



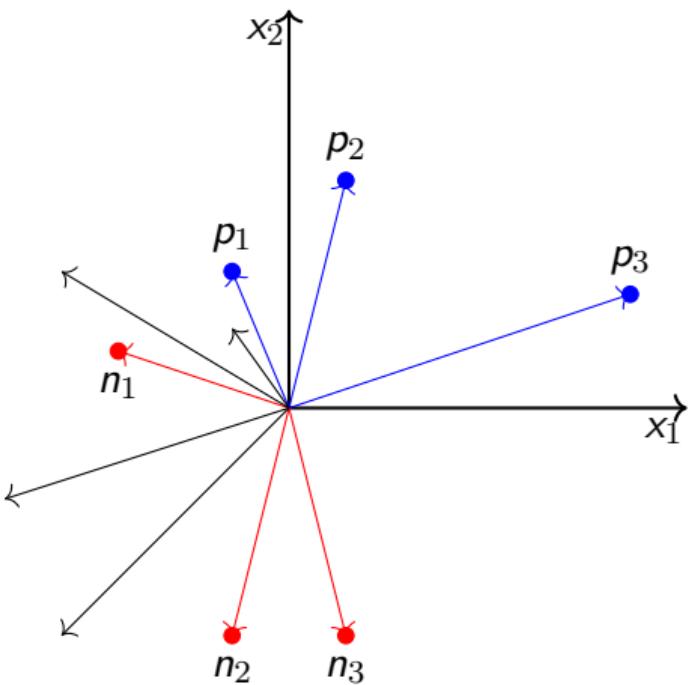
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, p_2), 对权重进行更新 $w = w + x \because w \cdot x < 0$ (看看角度如何变化)



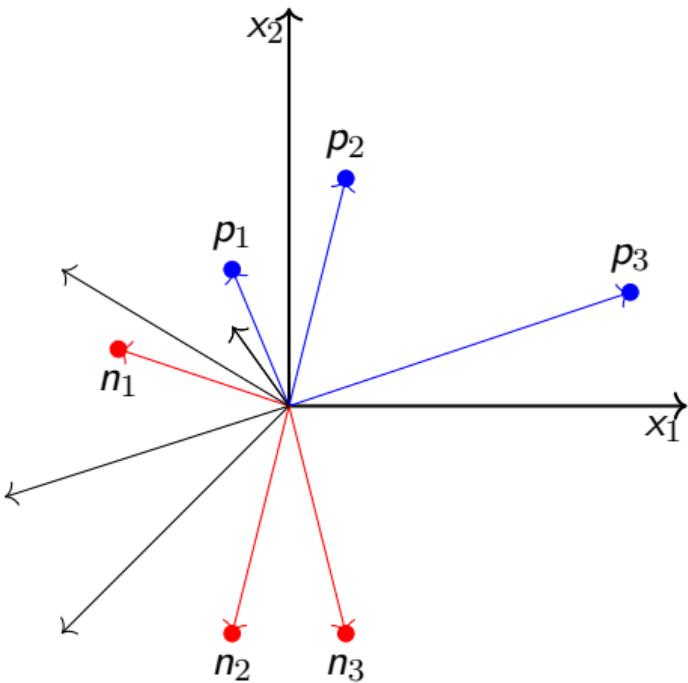
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, n_1), 对权重进行更新 $w = w - x \because w \cdot x \geq 0$ (看看角度如何变化)



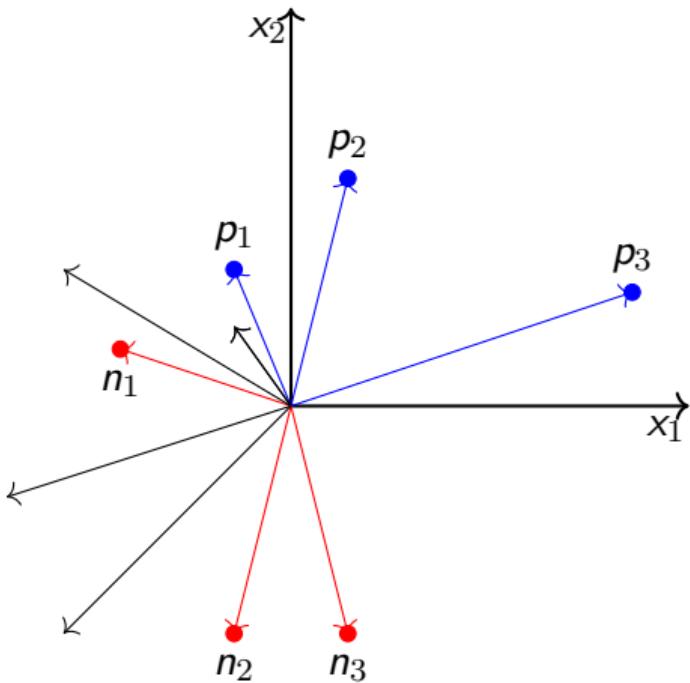
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, n_1), 对权重进行更新 $w = w - x \because w \cdot x \geq 0$ (看看角度如何变化)



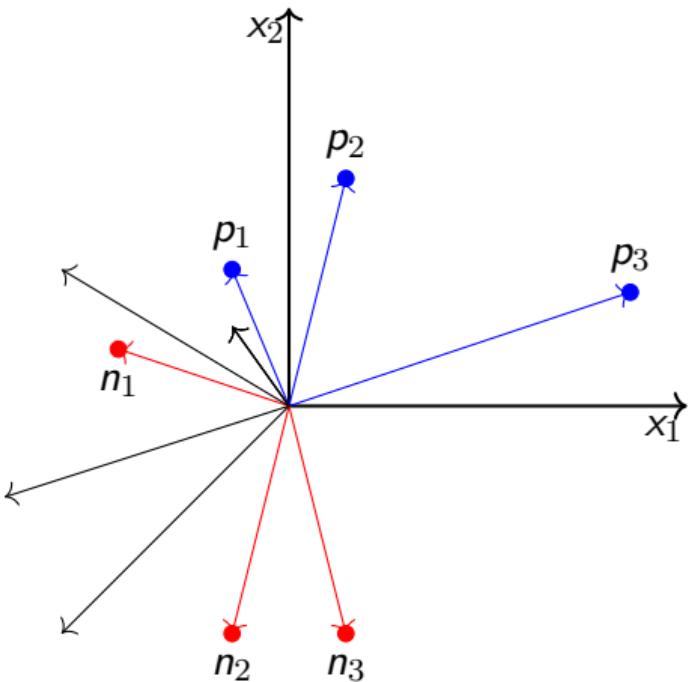
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, n_3), 无需对权重进行更新 $\because w \cdot x < 0$ (看看角度如何变化)



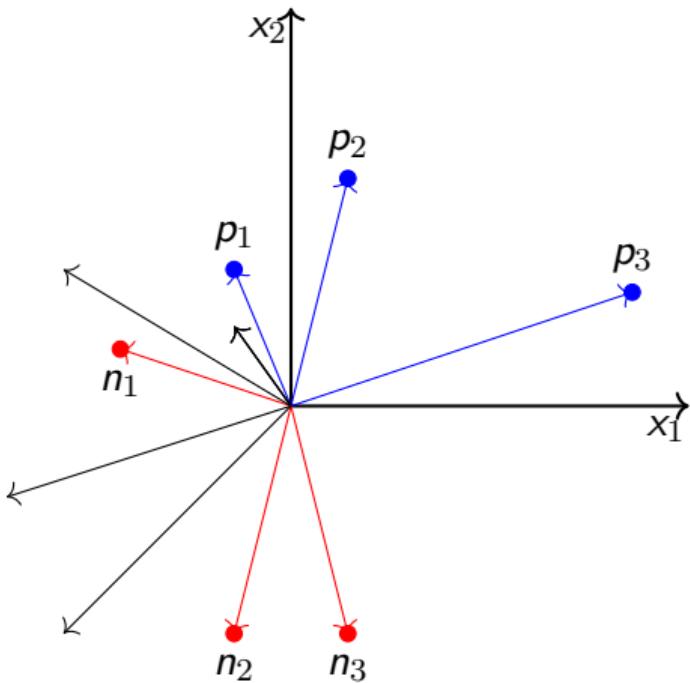
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\therefore \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\therefore \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, n_3), 无需对权重进行更新 $\because w \cdot x < 0$ (看看角度如何变化)



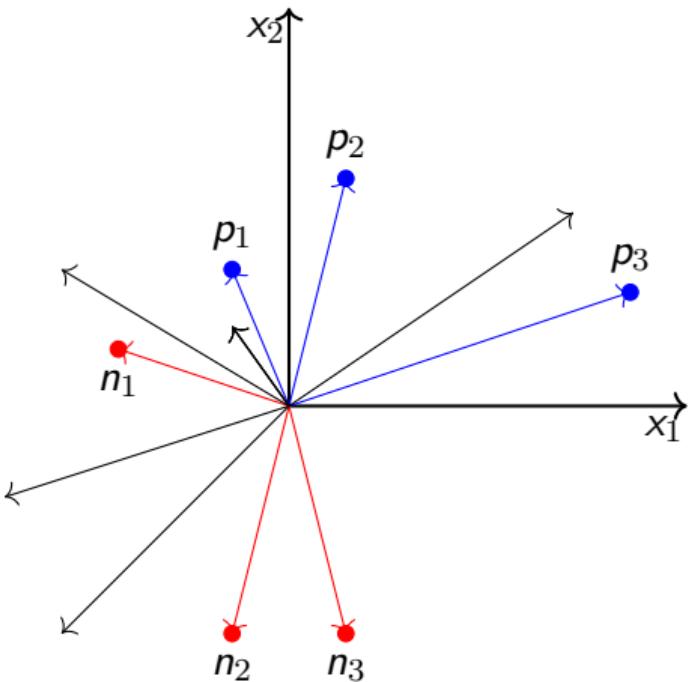
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, n_2), 无需对权重进行更新 $\because w \cdot x < 0$ (看看角度如何变化)



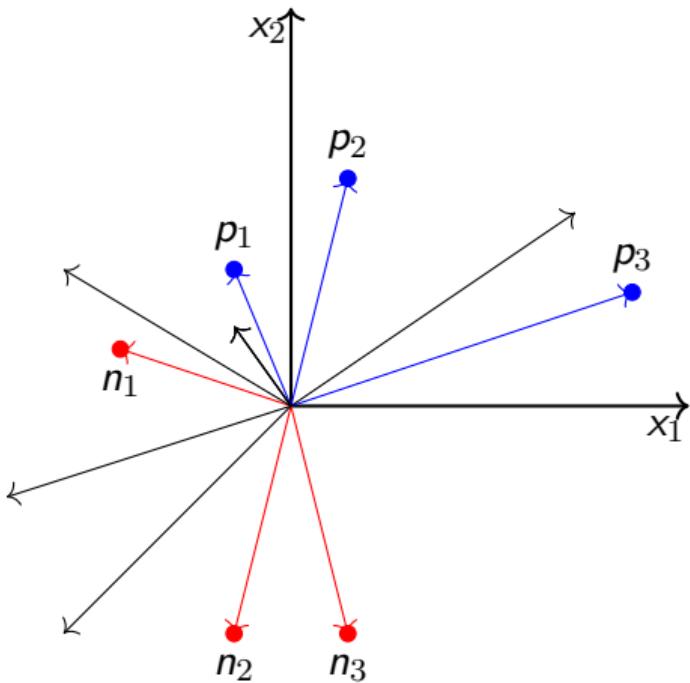
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, n_2), 无需对权重进行更新 $\because w \cdot x < 0$ (看看角度如何变化)



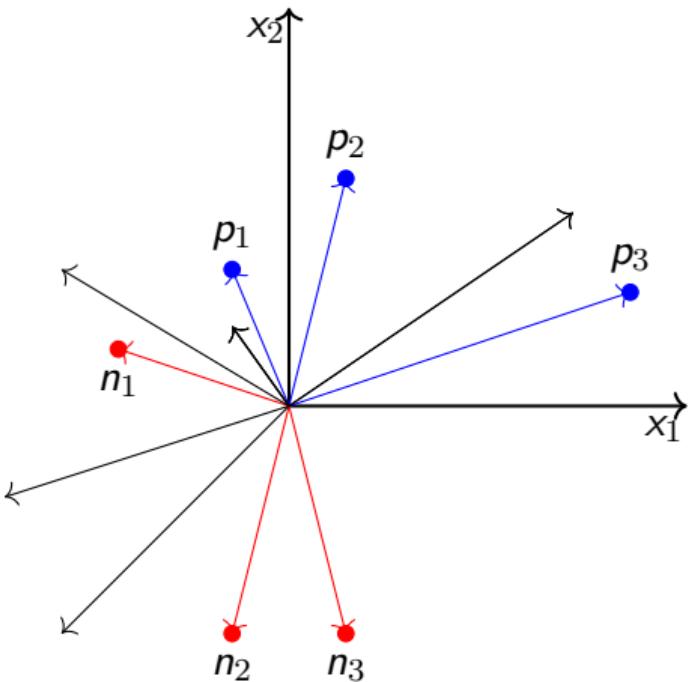
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\therefore \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\therefore \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, p_3), 对权重进行更新 $w = w + x \because w \cdot x < 0$ (看看角度如何变化)



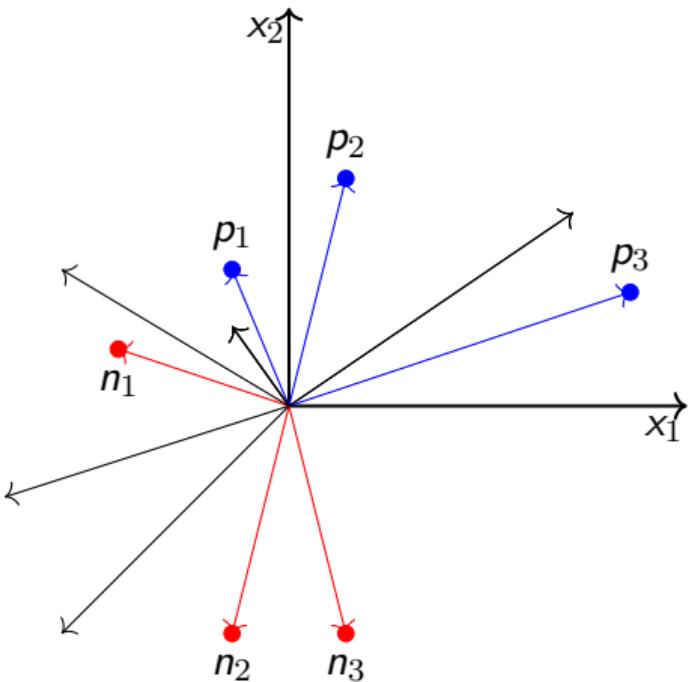
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, p_3), 对权重进行更新 $w = w + x \because w \cdot x < 0$ (看看角度如何变化)



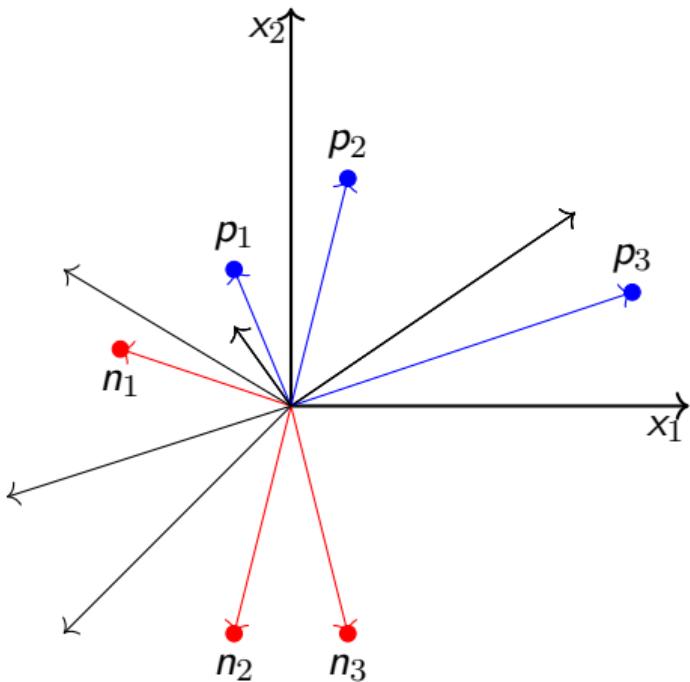
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, p_1), 无需对权重进行更新 $\because w \cdot x \geq 0$ (看看角度如何变化)



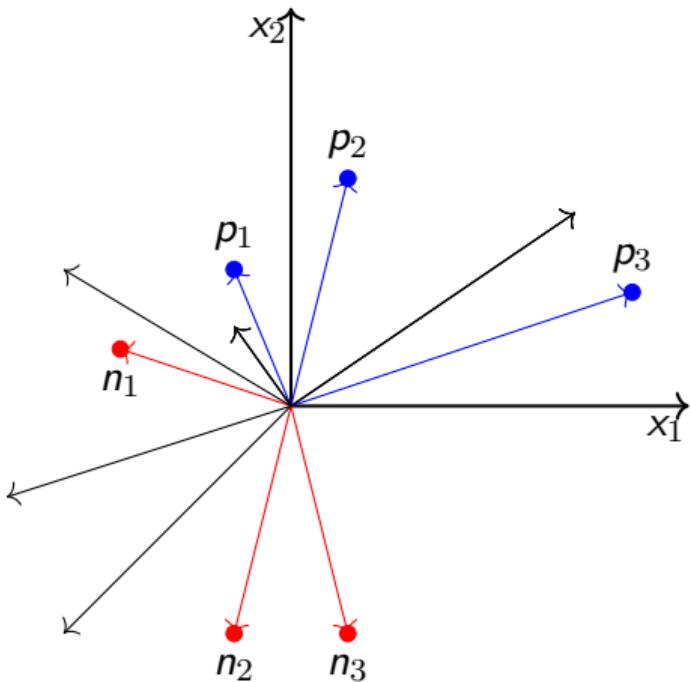
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, p_1), 无需对权重进行更新 $\because w \cdot x \geq 0$ (看看角度如何变化)



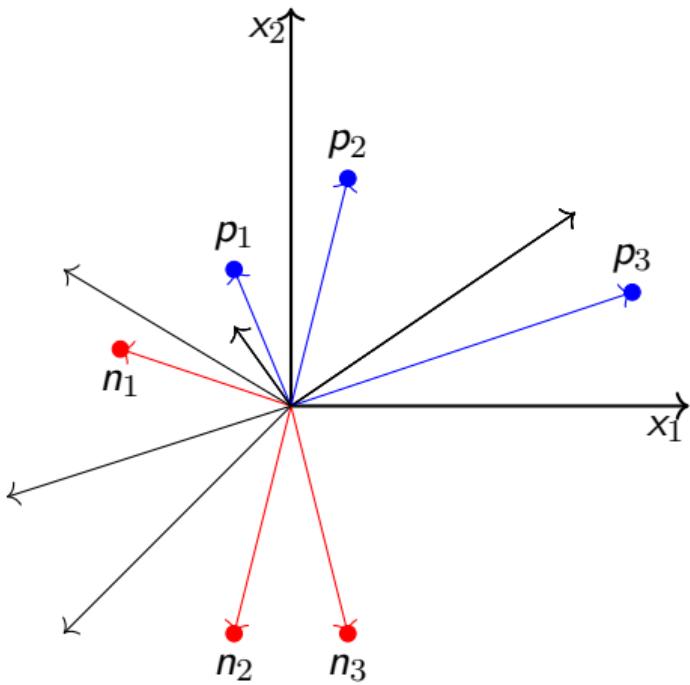
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, p_2), 无需对权重进行更新 $\because w \cdot x \geq 0$ (看看角度如何变化)



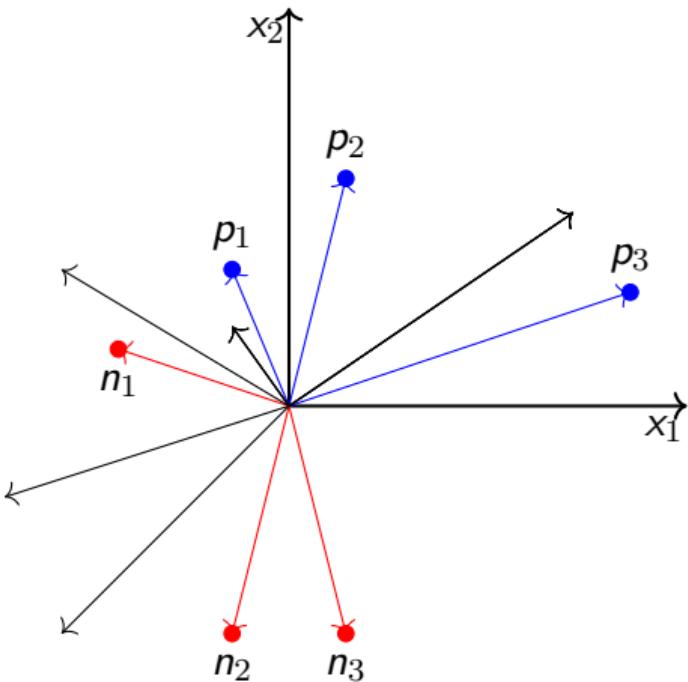
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, p_2), 无需对权重进行更新 $\because w \cdot x \geq 0$ (看看角度如何变化)



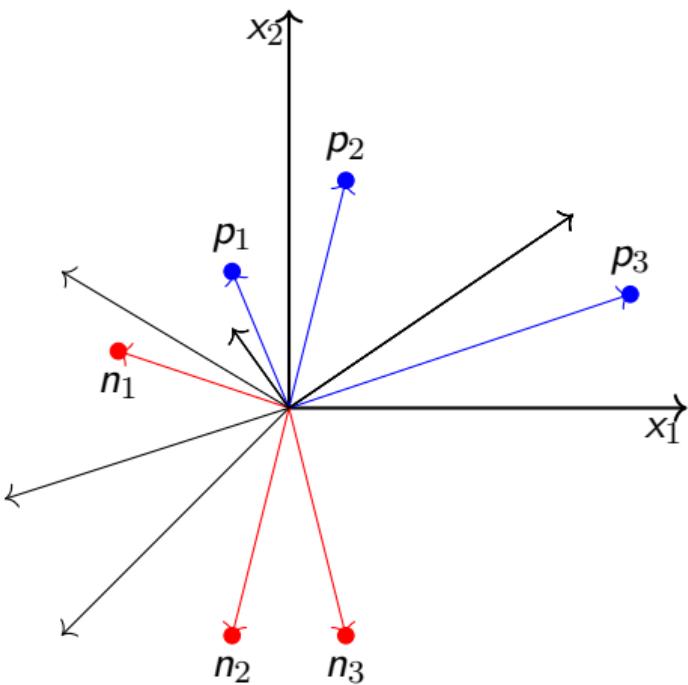
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, n_1), 无需对权重进行更新 $\because w \cdot x < 0$ (看看角度如何变化)



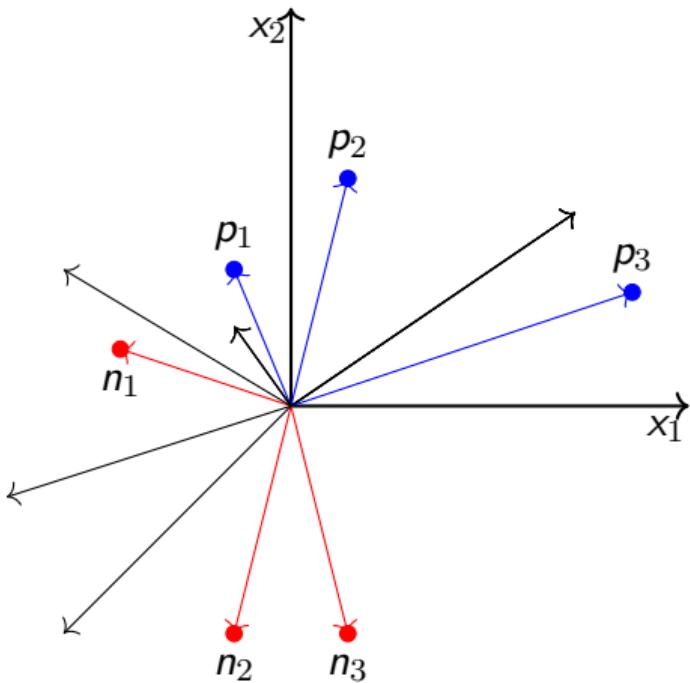
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, n_1), 无需对权重进行更新 $\because w \cdot x < 0$ (看看角度如何变化)



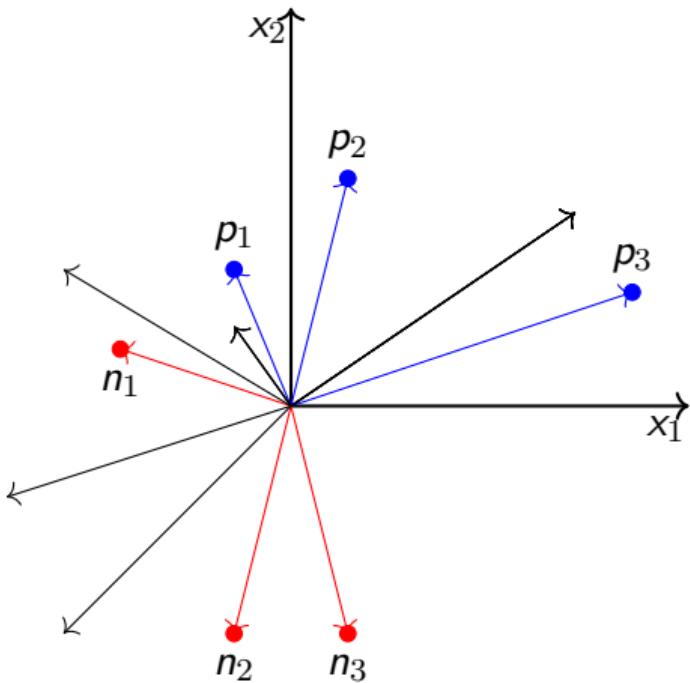
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, n_3), 无需对权重进行更新 $\because w \cdot x < 0$ (看看角度如何变化)



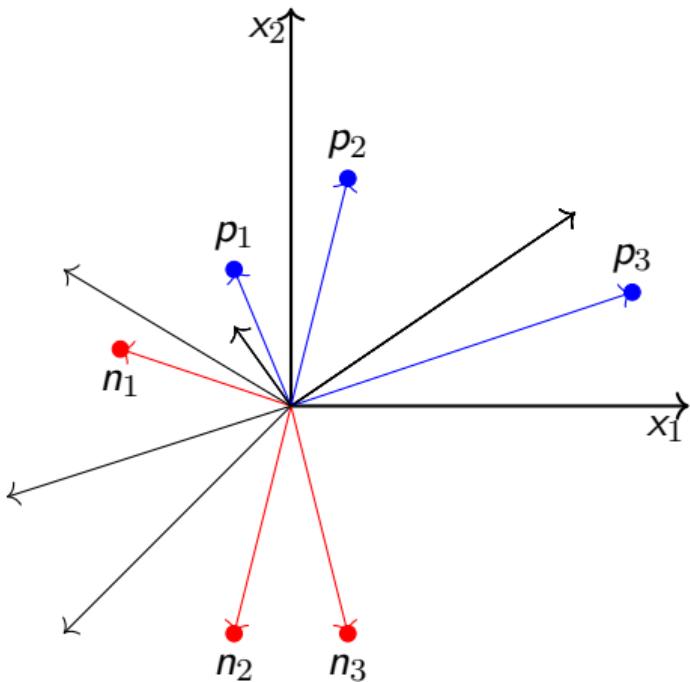
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\therefore \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\therefore \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, n_3), 无需对权重进行更新 $\because w \cdot x < 0$ (看看角度如何变化)



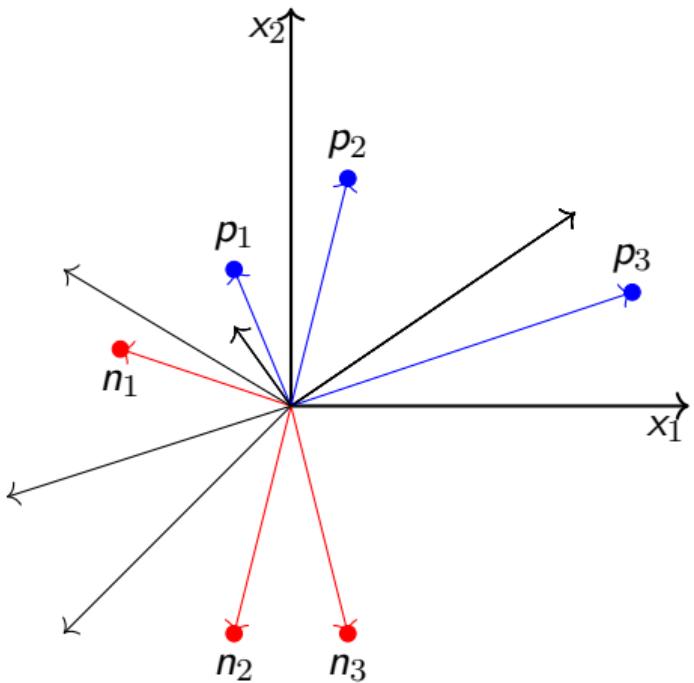
- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, n_2), 无需对权重进行更新 $\because w \cdot x < 0$ (看看角度如何变化)



- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, n_2), 无需对权重进行更新 $\because w \cdot x < 0$ (看看角度如何变化)



- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 随机选择一个样本 (如, p_3), 无需对权重进行更新 $\because w \cdot x \geq 0$ (看看角度如何变化)



- 随机初始化 w
- 当前, 对于所有标签为正的样本点 $w \cdot x < 0$ ($\because \text{angle} > 90^\circ$), 对于所有标签为负的样本点 $w \cdot x \geq 0$ ($\because \text{angle} < 90^\circ$) (当前形势和我们期望的正好相反)
- 运行算法, 随机地遍历样本点
- 算法收敛



线性不可分的布尔函数





- 如何实现线性不可分的函数?



- 如何实现线性不可分的函数?
- 看一个线性不可分的布尔函数

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 < 0 \implies w_1 + w_2 < -w_0$$

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 < 0 \implies w_1 + w_2 < -w_0$$

- 第 4 个不等式与不等式 2 和 3 冲突了

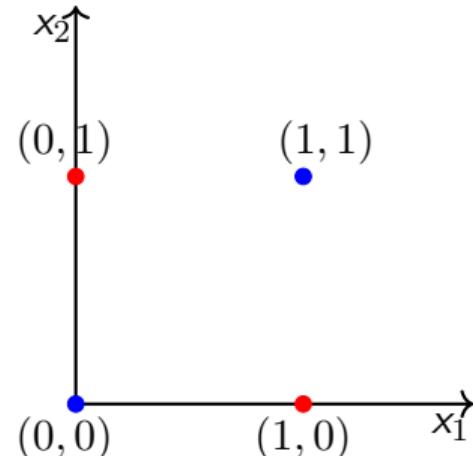
x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

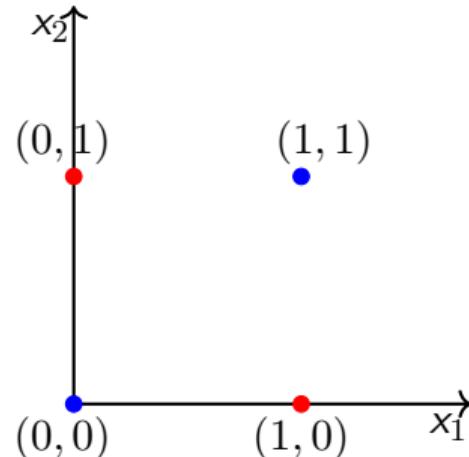
$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 < 0 \implies w_1 + w_2 < -w_0$$



- 第 4 个不等式与不等式 2 和 3 冲突了
- 因此，这些不等式没有解

x_1	x_2	XOR	
0	0	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$
1	0	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
0	1	1	$w_0 + \sum_{i=1}^2 w_i x_i \geq 0$
1	1	0	$w_0 + \sum_{i=1}^2 w_i x_i < 0$



$$w_0 + w_1 \cdot 0 + w_2 \cdot 0 < 0 \implies w_0 < 0$$

$$w_0 + w_1 \cdot 0 + w_2 \cdot 1 \geq 0 \implies w_2 \geq -w_0$$

$$w_0 + w_1 \cdot 1 + w_2 \cdot 0 \geq 0 \implies w_1 \geq -w_0$$

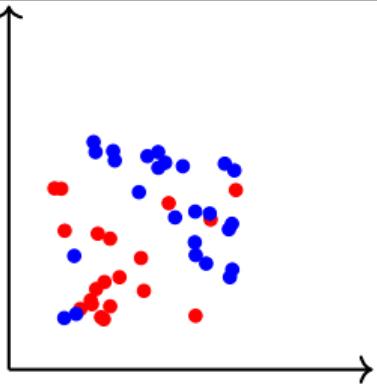
$$w_0 + w_1 \cdot 1 + w_2 \cdot 1 < 0 \implies w_1 + w_2 < -w_0$$

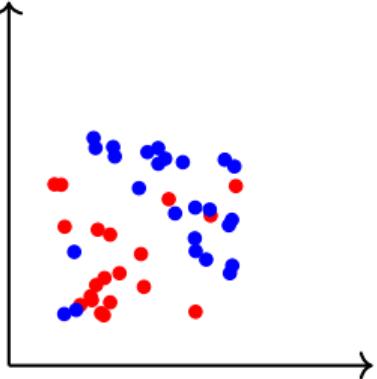
- 实际上, 不可能找到一条直线来区分红色的点和蓝色的点

- 第 4 个不等式与不等式 2 和 3 冲突了
- 因此, 这些不等式没有解

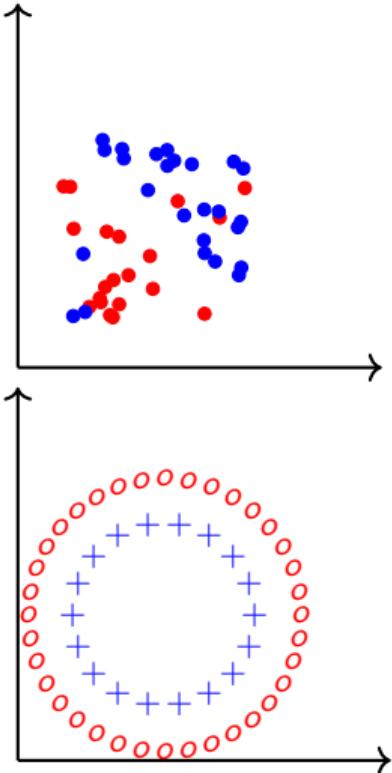


- 大多数真实的数据都是线性不可分的，总是存在一些 outliers

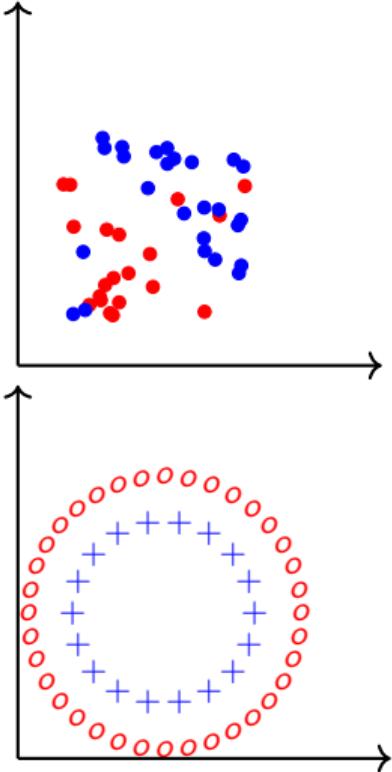




- 大多数真实的数据都是线性不可分的，总是存在一些 outliers
- 事实上，即使不存在 outliers，也可能是线性不可分的



- 大多数真实的数据都是线性不可分的，总是存在一些 outliers
- 事实上，即使不存在 outliers，也可能是线性不可分的
- 需要有能处理线性不可分数据的计算模型



- 大多数真实的数据都是线性不可分的，总是存在一些 outliers
- 事实上，即使不存在 outliers，也可能是线性不可分的
- 需要有能处理线性不可分数据的计算模型
- 尽管一个单个的感知机模型不能处理线性不可分的数据，但由感知机构成的一个网络可以处理线性不可分的数据



- 在看一个感知机网络如何处理线性不可分的数据之前，先看看布尔函数的更多细节...

- 给你两个输入，能设计多少个布尔函数？





- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..



- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2
0	0
0	1
1	0
1	1



- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1
0	0	0
0	1	0
1	0	0
1	1	0



- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_{16}
0	0	0	1
0	1	0	1
1	0	0	1
1	1	0	1



- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_{16}
0	0	0	0	1
0	1	0	0	1
1	0	0	0	1
1	1	0	1	1



- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_8	f_{16}
0	0	0	0	0	1
0	1	0	0	1	1
1	0	0	0	1	1
1	1	0	1	1	1



- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_8	f_{16}
0	0	0	0	0	0	1
0	1	0	0	0	1	1
1	0	0	0	1	1	1
1	1	0	1	0	1	1

- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_8	f_{16}
0	0	0	0	0	0	0	1
0	1	0	0	0	0	1	1
1	0	0	0	1	1	1	1
1	1	0	1	0	1	1	1



- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_8	f_{16}
0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	1	1	1
1	0	0	0	1	1	0	1	1
1	1	0	1	0	1	0	1	1



- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_8	f_{16}
0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	1	1

- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_{16}
0	0	0	0	0	0	0	0	0	0	1
0	1	0	0	0	0	1	1	1	1	1
1	0	0	0	1	1	0	0	1	1	1
1	1	0	1	0	1	0	1	0	1	1



- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1
0	1	0	0	0	0	1	1	1	1	0	1
1	0	0	0	1	1	0	0	1	1	0	1
1	1	0	1	0	1	0	1	0	1	0	1



- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	1
1	0	0	0	1	1	0	0	1	1	0	0	1
1	1	0	1	0	1	0	1	0	1	0	1	1



- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1

- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	1

- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	1

- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- 这些布尔函数中，哪些是线性可分的？

- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- 这些布尔函数中，哪些是线性可分的？(好像是除了 XOR and !XOR 外，- 可以验证一下)

- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- 这些布尔函数中，哪些是线性可分的？(好像是除了 XOR and !XOR 外，- 可以验证一下)
- 一般来说，对于 n 个输入，能够有多少个布尔函数？

- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- 这些布尔函数中，哪些是线性可分的？(好像是除了 XOR and !XOR 外，- 可以验证一下)
- 一般来说，对于 n 个输入，能够有多少个布尔函数？ 2^{2^n}

- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- 这些布尔函数中，哪些是线性可分的？(好像是除了 XOR and !XOR 外，- 可以验证一下)
- 一般来说，对于 n 个输入，能够有多少个布尔函数？ 2^{2^n}
- 所有 2^{2^n} 个布尔函数中有多少个是线性不可分的？

- 给你两个输入，能设计多少个布尔函数？
- 先看看一些已知的布尔函数..

x_1	x_2	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

- 这些布尔函数中，哪些是线性可分的？(好像是除了 XOR and !XOR 外，- 可以验证一下)
- 一般来说，对于 n 个输入，能够有多少个布尔函数？ 2^{2^n}
- 所有 2^{2^n} 个布尔函数中有多少个是线性不可分的？目前来看，确定肯定有线性不可分的布尔函数，至于精确的答案，大家可以想一想:-))



感知机网络的表示能力



- 我们将学习如何使用感知机网络来实现 任意布尔函数...

- 方便起见, 假设 True = +1, False = -1



- 方便起见, 假设 True = +1, False = -1
- 考虑 2 个输入, 4 个感知机



x_1

x_2



- 方便起见, 假设 True = +1, False = -1
- 考虑 2 个输入, 4 个感知机
- 每个输入都跟 4 个感知机连接并带有权重

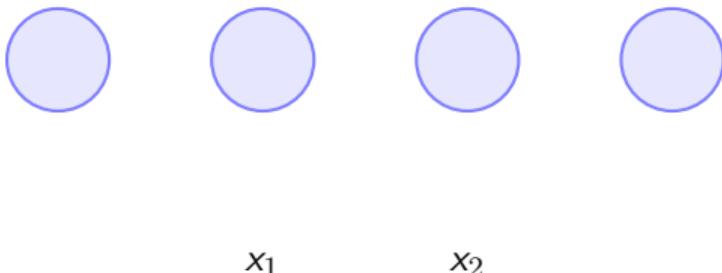


x_1

x_2



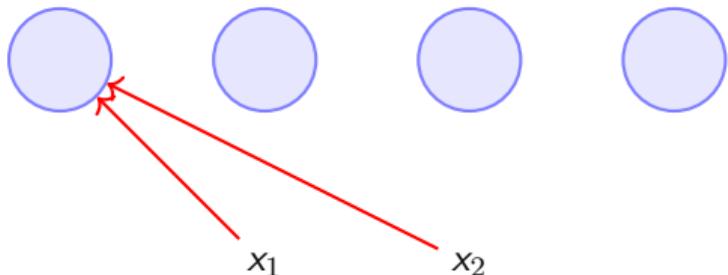
- 方便起见, 假设 True = +1, False = -1
- 考虑 2 个输入, 4 个感知机
- 每个输入都跟 4 个感知机连接并带有权重



red edge indicates $w = -1$
blue edge indicates $w = +1$



- 方便起见, 假设 True = +1, False = -1
- 考虑 2 个输入, 4 个感知机
- 每个输入都跟 4 个感知机连接并带有权重

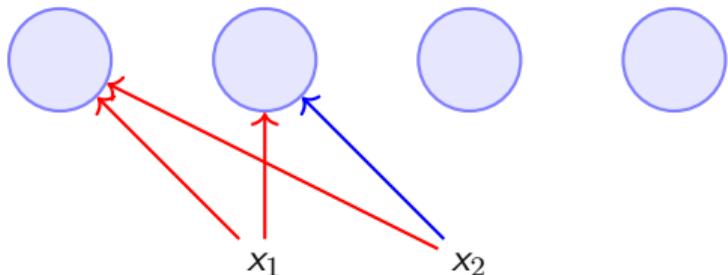


red edge indicates $w = -1$

blue edge indicates $w = +1$



- 方便起见, 假设 True = +1, False = -1
- 考虑 2 个输入, 4 个感知机
- 每个输入都跟 4 个感知机连接并带有权重

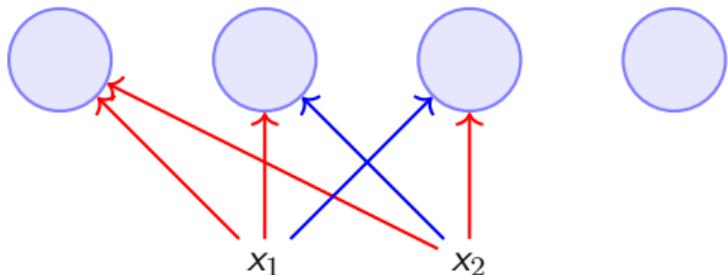


red edge indicates $w = -1$

blue edge indicates $w = +1$



- 方便起见, 假设 True = +1, False = -1
- 考虑 2 个输入, 4 个感知机
- 每个输入都跟 4 个感知机连接并带有权重

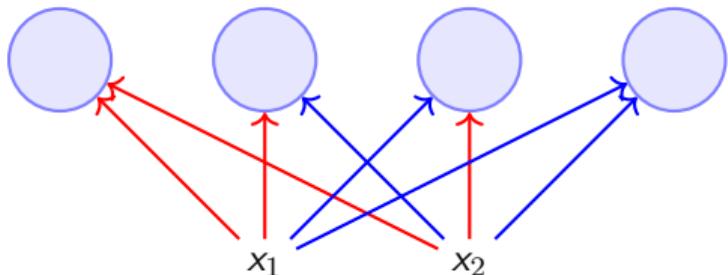


red edge indicates $w = -1$

blue edge indicates $w = +1$



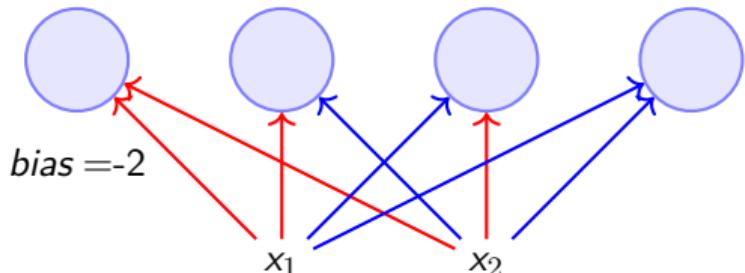
- 方便起见, 假设 True = +1, False = -1
- 考虑 2 个输入, 4 个感知机
- 每个输入都跟 4 个感知机连接并带有权重



red edge indicates $w = -1$

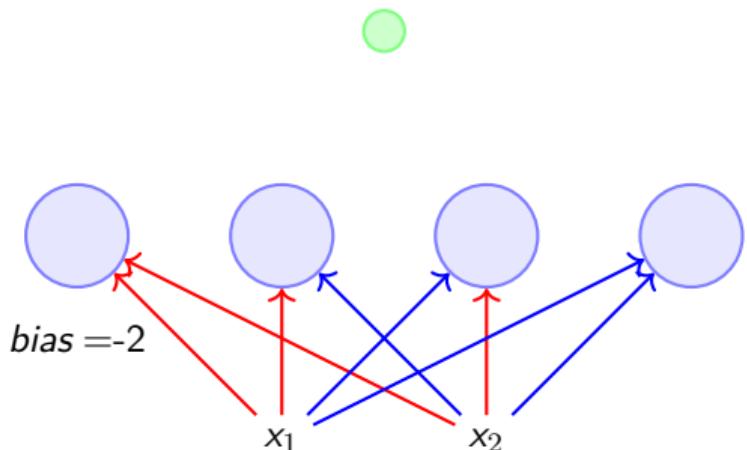
blue edge indicates $w = +1$

- 方便起见, 假设 True = +1, False = -1
- 考虑 2 个输入, 4 个感知机
- 每个输入都跟 4 个感知机连接并带有权重
- 每个感知机的偏置 (w_0) 是 -2 (例如, 每个感知机只要当它的输入的加权求和 ≥ 2 时才激活)



red edge indicates $w = -1$

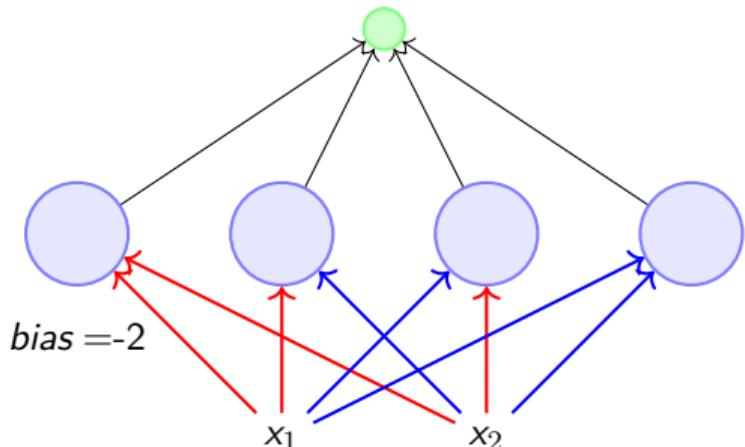
blue edge indicates $w = +1$



red edge indicates $w = -1$

blue edge indicates $w = +1$

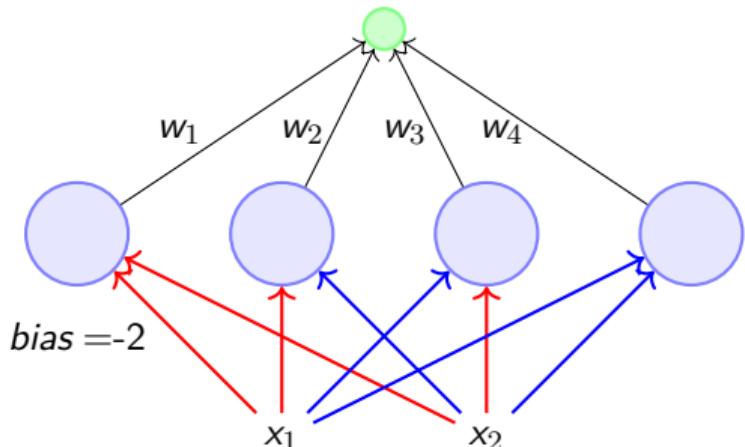
- 方便起见, 假设 True = +1, False = -1
- 考虑 2 个输入, 4 个感知机
- 每个输入都跟 4 个感知机连接并带有权重
- 每个感知机的偏置 (w_0) 是 -2 (例如, 每个感知机只要当它的输入的加权求和 ≥ 2 时才激活)
- 每个感知机又通过权重连接到一个输出感知机 (权重需要学习)



red edge indicates $w = -1$

blue edge indicates $w = +1$

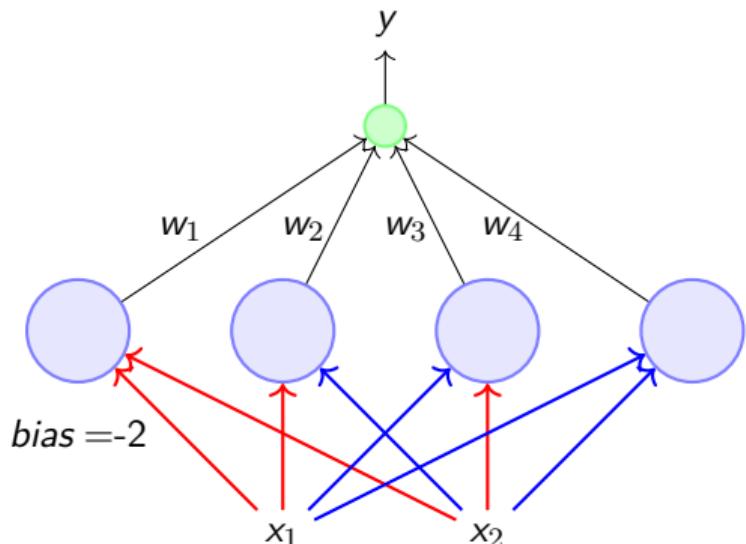
- 方便起见, 假设 True = +1, False = -1
- 考虑 2 个输入, 4 个感知机
- 每个输入都跟 4 个感知机连接并带有权重
- 每个感知机的偏置 (w_0) 是 -2 (例如, 每个感知机只要当它的输入的加权求和 ≥ 2 时才激活)
- 每个感知机又通过权重连接到一个输出感知机 (权重需要学习)



red edge indicates $w = -1$

blue edge indicates $w = +1$

- 方便起见, 假设 True = +1, False = -1
- 考虑 2 个输入, 4 个感知机
- 每个输入都跟 4 个感知机连接并带有权重
- 每个感知机的偏置 (w_0) 是 -2 (例如, 每个感知机只要当它的输入的加权求和 ≥ 2 时才激活)
- 每个感知机又通过权重连接到一个输出感知机 (权重需要学习)



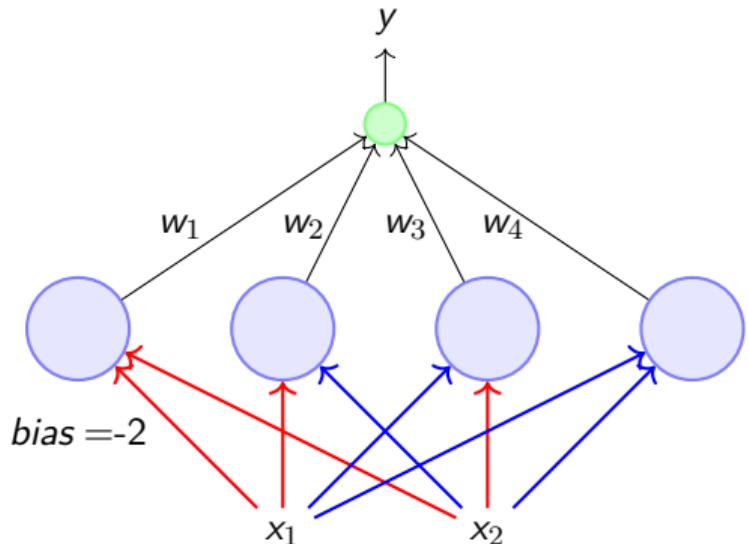
red edge indicates $w = -1$

blue edge indicates $w = +1$

- 方便起见, 假设 True = +1, False = -1
- 考虑 2 个输入, 4 个感知机
- 每个输入都跟 4 个感知机连接并带有权重
- 每个感知机的偏置 (w_0) 是 -2 (例如, 每个感知机只要当它的输入的加权求和 ≥ 2 时才激活)
- 每个感知机又通过权重连接到一个输出感知机 (权重需要学习)
- 输出感知机的输出 (y) 是整个网络的输出

术语:

- 这个网络包含 3 层

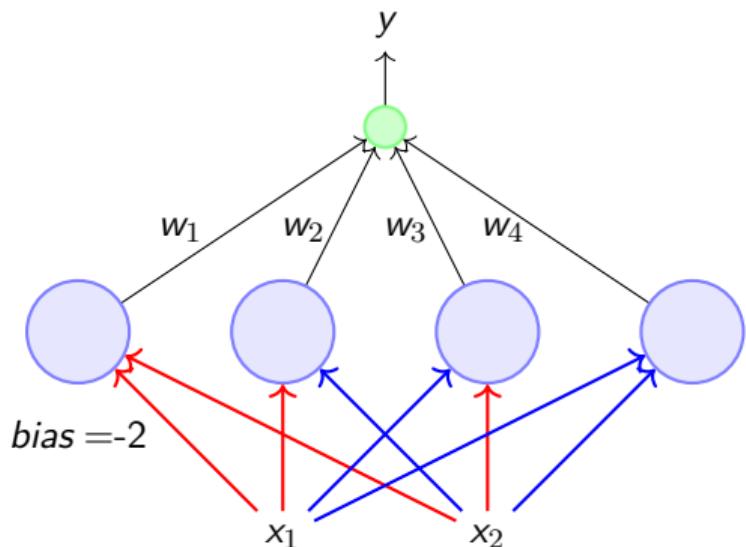


red edge indicates $w = -1$

blue edge indicates $w = +1$

术语:

- 这个网络包含 3 层
- 包含输入 (x_1, x_2) 的层称为 **input layer**

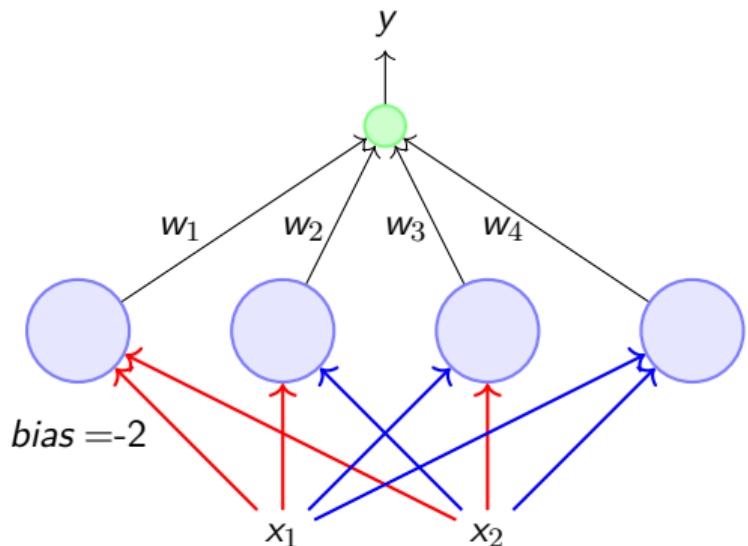


red edge indicates $w = -1$

blue edge indicates $w = +1$

术语:

- 这个网络包含 3 层
- 包含输入 (x_1, x_2) 的层称为 **input layer**
- 包含 4 个感知机的中间层称为 **hidden layer**

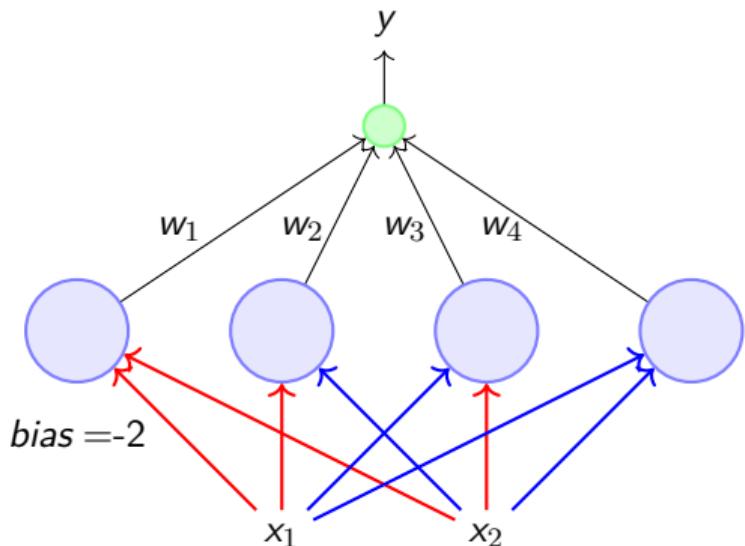


red edge indicates $w = -1$

blue edge indicates $w = +1$

术语:

- 这个网络包含 3 层
- 包含输入 (x_1, x_2) 的层称为 **input layer**
- 包含 4 个感知机的中间层称为 **hidden layer**
- 包含一个输出神经元的最后一层称为 **output layer**

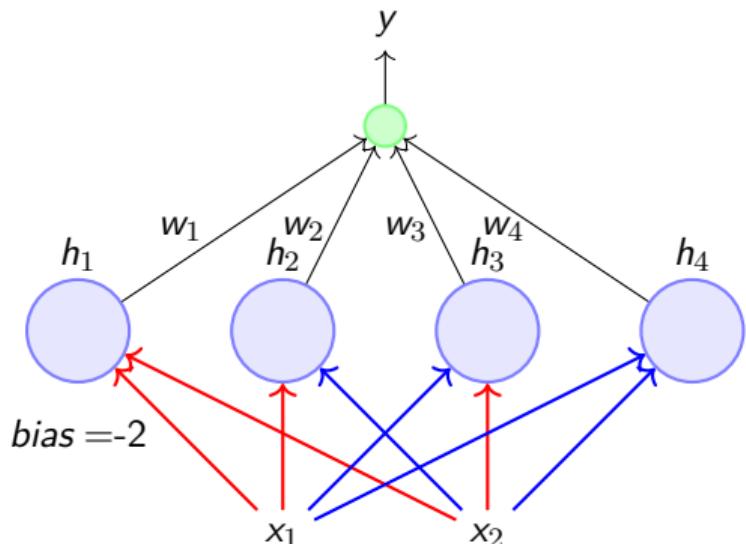


red edge indicates $w = -1$

blue edge indicates $w = +1$

术语:

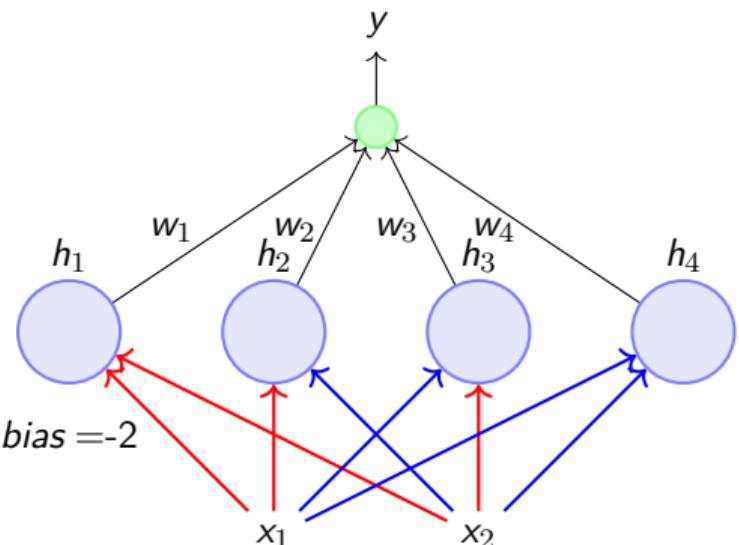
- 这个网络包含 3 层
- 包含输入 (x_1, x_2) 的层称为 **input layer**
- 包含 4 个感知机的中间层称为 **hidden layer**
- 包含一个输出神经元的最后一层称为 **output layer**
- 隐含层中 4 个感知机的输出用 h_1, h_2, h_3, h_4 表示



red edge indicates $w = -1$

blue edge indicates $w = +1$

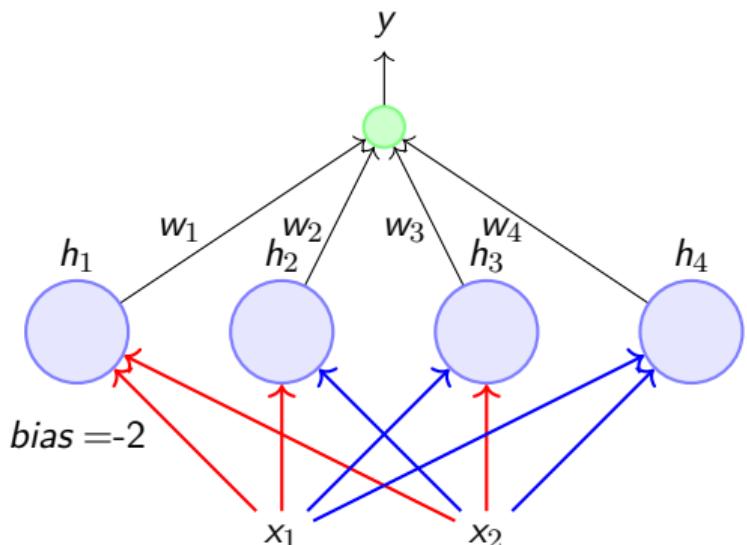
术语:



red edge indicates $w = -1$

blue edge indicates $w = +1$

术语:

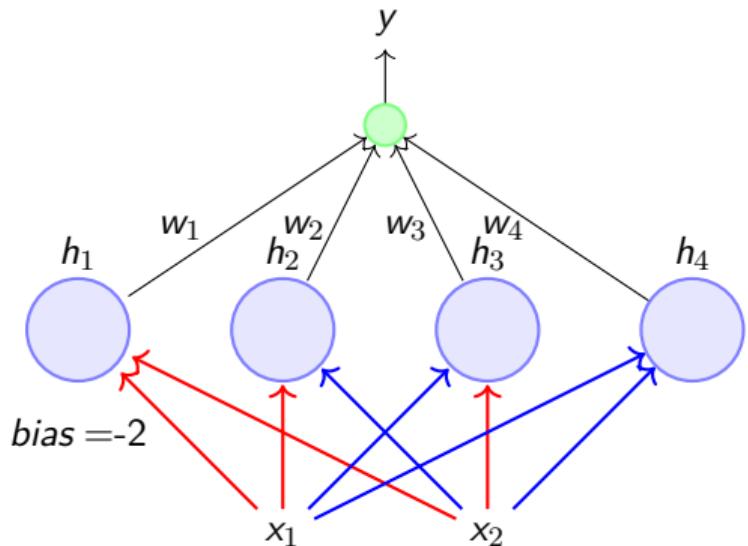


red edge indicates $w = -1$

blue edge indicates $w = +1$

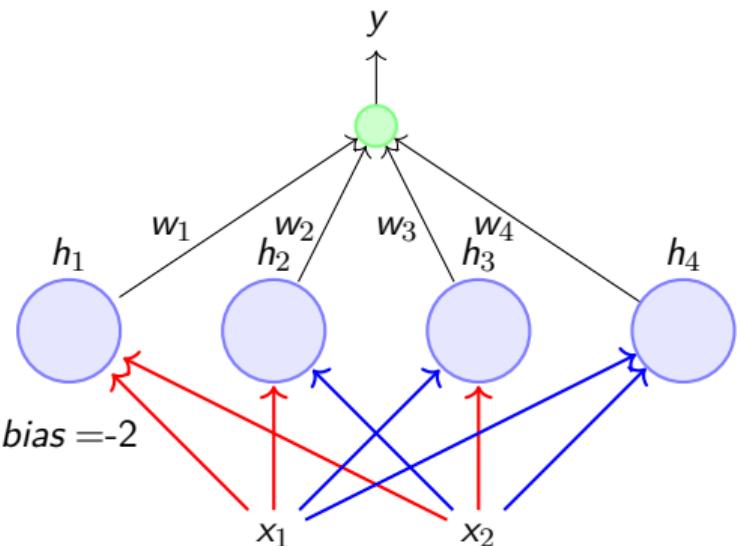
- 这个网络包含 3 层
- 包含输入 (x_1, x_2) 的层称为 **input layer**
- 包含 4 个感知机的中间层称为 **hidden layer**
- 包含一个输出神经元的最后一层称为 **output layer**
- 隐含层中 4 个感知机的输出用 h_1, h_2, h_3, h_4 表示
- 红色和蓝色的边称作为层 1 的权重
- w_1, w_2, w_3, w_4 称为层 2 的权重

- 这个网络能够实现 **任意**布尔函数 (线性可分和线性不可分)



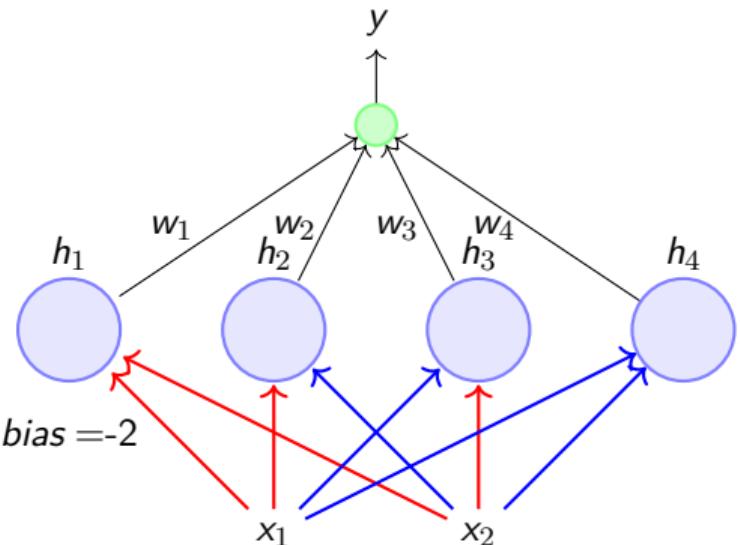
red edge indicates $w = -1$

blue edge indicates $w = +1$



red edge indicates $w = -1$

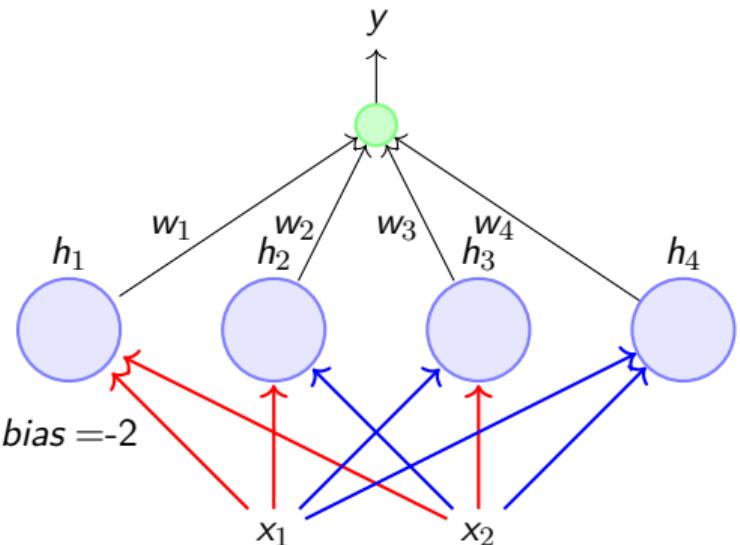
blue edge indicates $w = +1$



red edge indicates $w = -1$

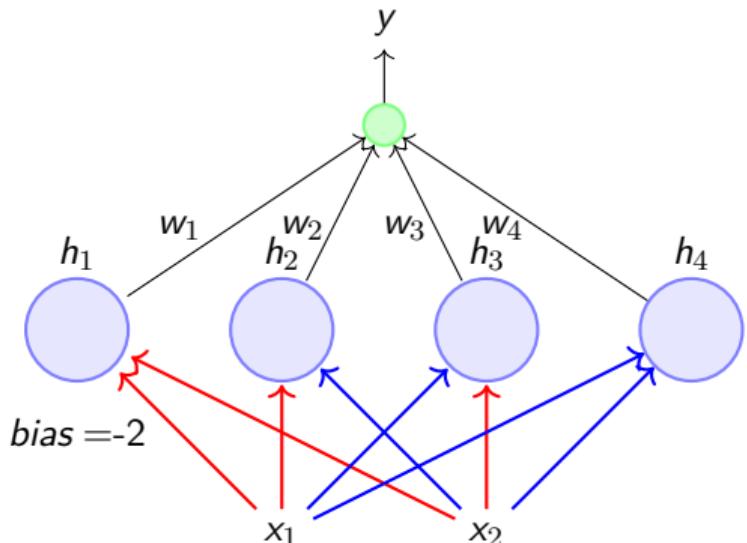
blue edge indicates $w = +1$

- 这个网络能够实现 **任意布尔函数** (线性可分和线性不可分)
- 换句话说, 我们能够找到 w_1, w_2, w_3, w_4 使得任意布尔函数都能由这个网络实现
- 很酷的结论!



red edge indicates $w = -1$

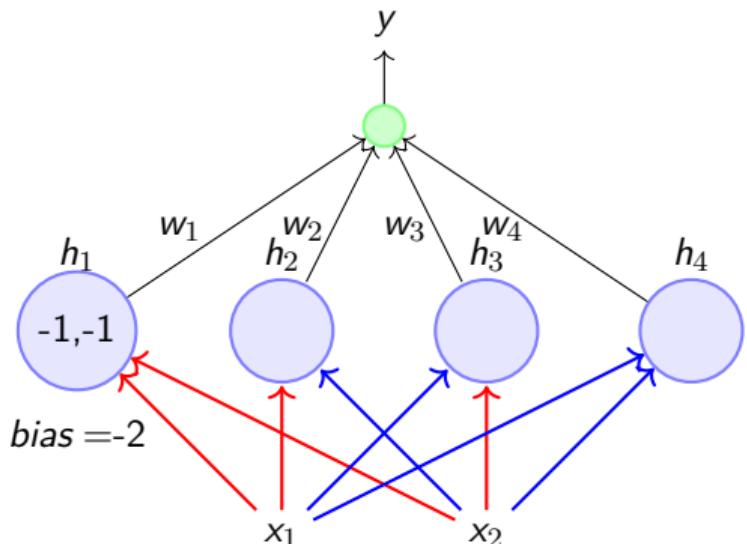
blue edge indicates $w = +1$



red edge indicates $w = -1$

blue edge indicates $w = +1$

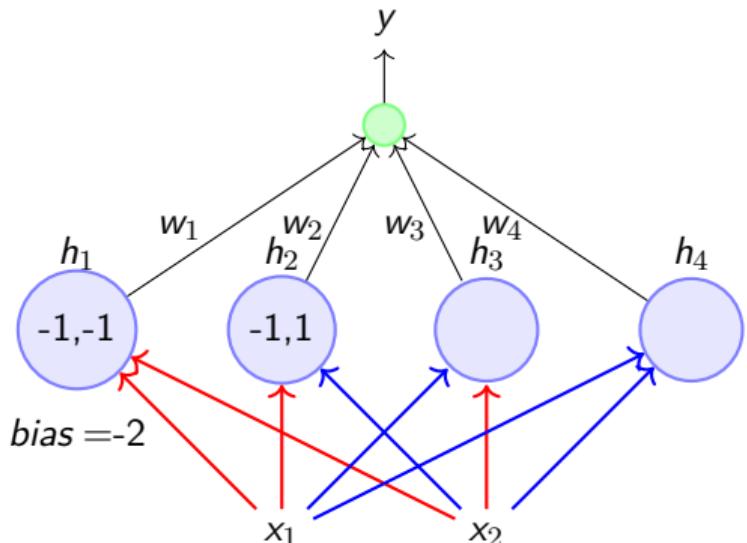
- 这个网络能够实现 **任意布尔函数** (线性可分和线性不可分)
- 换句话说, 我们能够找到 w_1, w_2, w_3, w_4 使得任意布尔函数都能由这个网络实现
- 很酷的结论! 看看具体的解释
- 中间层的每个感知机仅对特定的输入进行『开火』 (不存在两个感知机对相同的输入进行『开火』)



red edge indicates $w = -1$

blue edge indicates $w = +1$

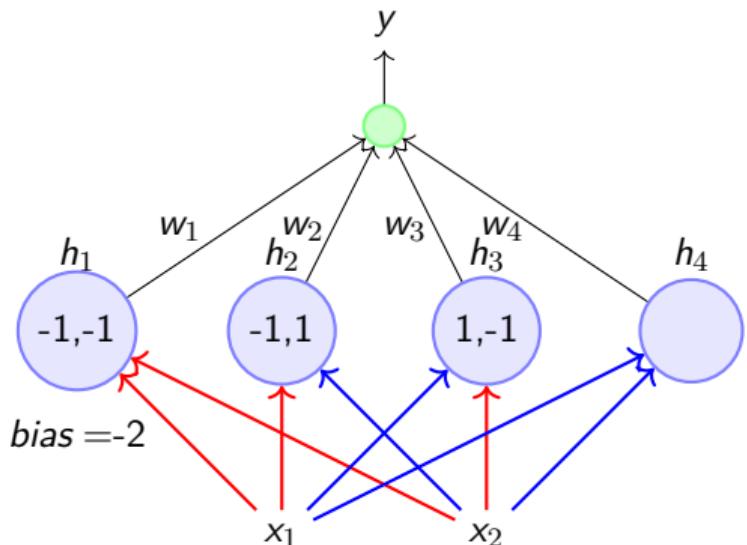
- 这个网络能够实现 **任意布尔函数** (线性可分和线性不可分)
- 换句话说, 我们能够找到 w_1, w_2, w_3, w_4 使得任意布尔函数都能由这个网络实现
- 很酷的结论! 看看具体的解释
- 中间层的每个感知机仅对特定的输入进行『开火』 (不存在两个感知机对相同的输入进行『开火』)
- 第一个感知机对 $\{-1, -1\}$ 『开火』



red edge indicates $w = -1$

blue edge indicates $w = +1$

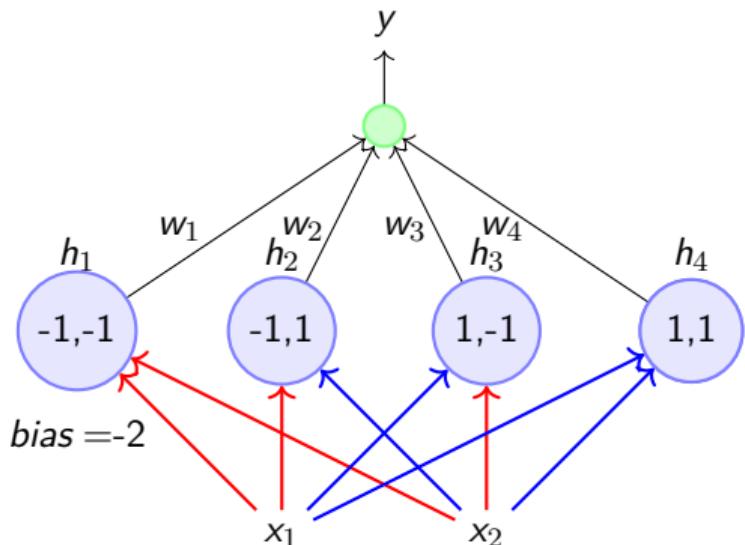
- 这个网络能够实现 **任意布尔函数** (线性可分和线性不可分)
- 换句话说, 我们能够找到 w_1, w_2, w_3, w_4 使得任意布尔函数都能由这个网络实现
- 很酷的结论! 看看具体的解释
- 中间层的每个感知机仅对特定的输入进行『开火』 (不存在两个感知机对相同的输入进行『开火』)
- 第二个感知机对 $\{-1, 1\}$ 『开火』



red edge indicates $w = -1$

blue edge indicates $w = +1$

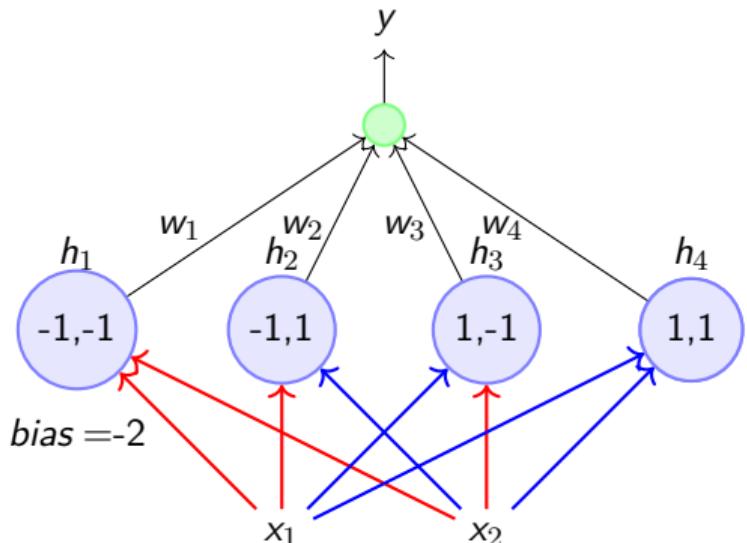
- 这个网络能够实现 **任意布尔函数** (线性可分和线性不可分)
- 换句话说, 我们能够找到 w_1, w_2, w_3, w_4 使得任意布尔函数都能由这个网络实现
- 很酷的结论! 看看具体的解释
- 中间层的每个感知机仅对特定的输入进行『开火』 (不存在两个感知机对相同的输入进行『开火』)
- 第三个感知机对 $\{1, -1\}$ 『开火』



red edge indicates $w = -1$

blue edge indicates $w = +1$

- 这个网络能够实现 **任意布尔函数** (线性可分和线性不可分)
- 换句话说, 我们能够找到 w_1, w_2, w_3, w_4 使得任意布尔函数都能由这个网络实现
- 很酷的结论! 看看具体的解释
- 中间层的每个感知机仅对特定的输入进行『开火』 (不存在两个感知机对相同的输入进行『开火』)
- 第四个感知机对 $\{1,1\}$ 『开火』



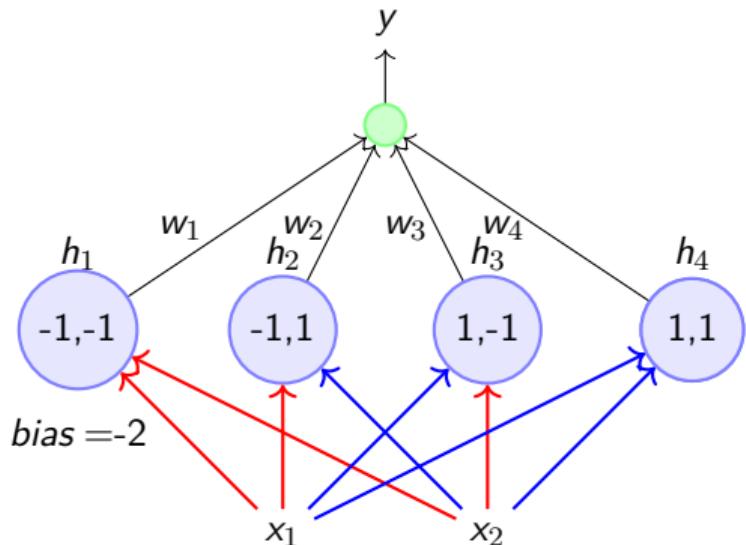
red edge indicates $w = -1$

blue edge indicates $w = +1$

- 这个网络能够实现 **任意**布尔函数 (线性可分和线性不可分)
- 换句话说, 我们能够找到 w_1, w_2, w_3, w_4 使得任意布尔函数都能由这个网络实现
- 很酷的结论! 看看具体的解释
- 中间层的每个感知机仅对特定的输入进行『开火』 (不存在两个感知机对相同的输入进行『开火』)
- 看看这个网络如何实现布尔函数 XOR



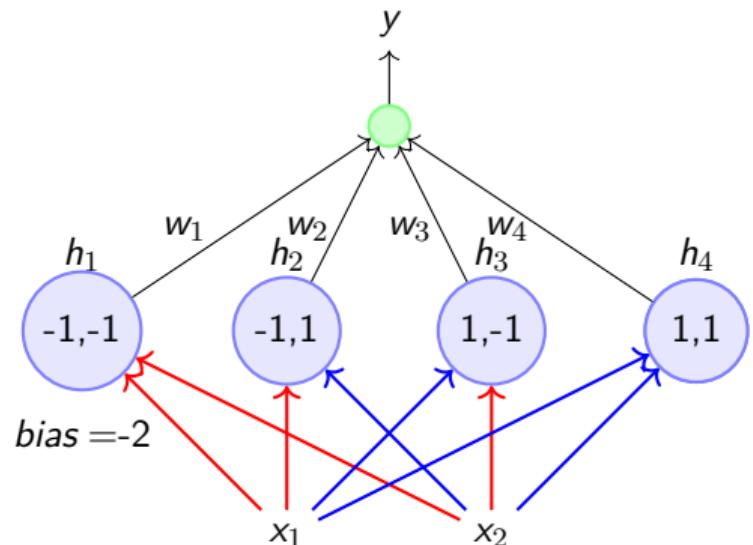
- 让 w_0 表示输出神经元的偏置 (i.e., 如果 $\sum_{i=1}^4 w_i h_i \geq w_0$, 输出神经元将激活)



red edge indicates $w = -1$

blue edge indicates $w = +1$

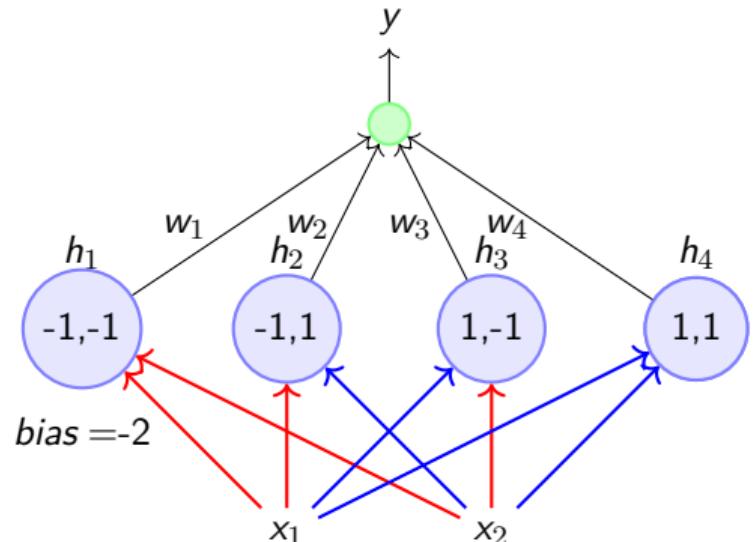
- 让 w_0 表示输出神经元的偏置 (i.e., 如果 $\sum_{i=1}^4 w_i h_i \geq w_0$, 输出神经元将激活)



red edge indicates $w = -1$
blue edge indicates $w = +1$

x_1	x_2	XOR	h_1	h_2	h_3	h_4	$\sum_{i=1}^4 w_i h_i$
0	0	0	1	0	0	0	w_1

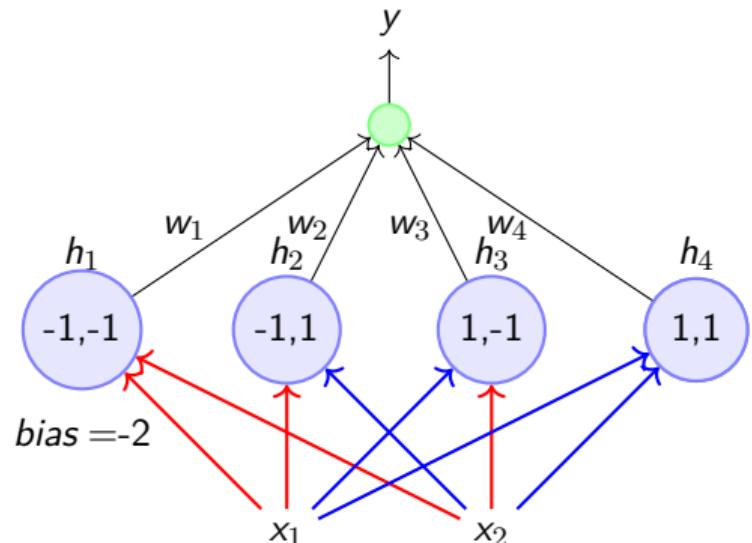
- 让 w_0 表示输出神经元的偏置 (i.e., 如果 $\sum_{i=1}^4 w_i h_i \geq w_0$, 输出神经元将激活)



red edge indicates $w = -1$
blue edge indicates $w = +1$

x_1	x_2	XOR	h_1	h_2	h_3	h_4	$\sum_{i=1}^4 w_i h_i$
0	0	0	1	0	0	0	w_1
0	1	1	0	1	0	0	w_2

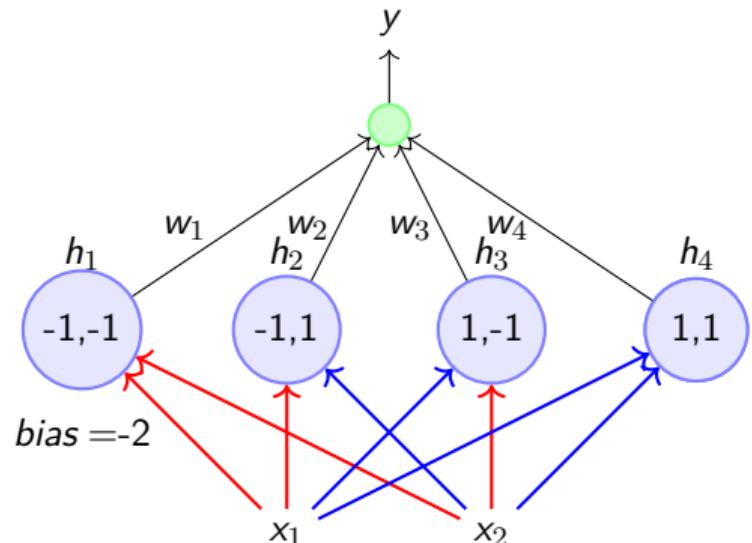
- 让 w_0 表示输出神经元的偏置 (i.e., 如果 $\sum_{i=1}^4 w_i h_i \geq w_0$, 输出神经元将激活)



red edge indicates $w = -1$
blue edge indicates $w = +1$

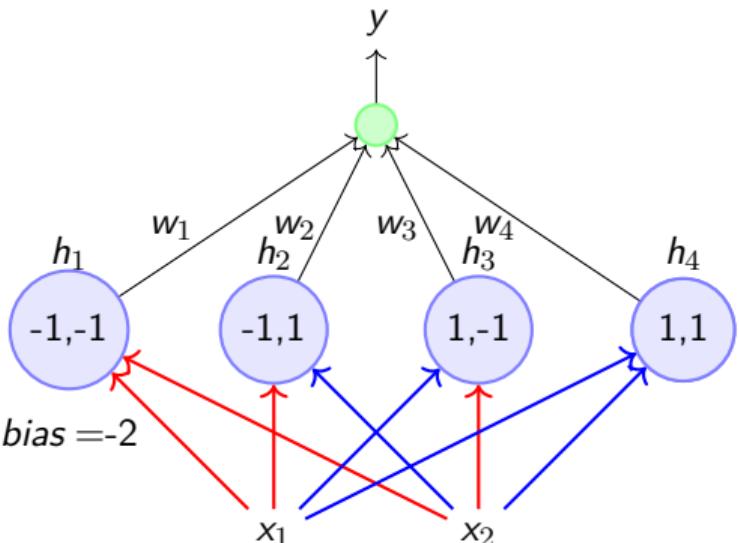
x_1	x_2	XOR	h_1	h_2	h_3	h_4	$\sum_{i=1}^4 w_i h_i$
0	0	0	1	0	0	0	w_1
0	1	1	0	1	0	0	w_2
1	0	1	0	0	1	0	w_3

- 让 w_0 表示输出神经元的偏置 (i.e., 如果 $\sum_{i=1}^4 w_i h_i \geq w_0$, 输出神经元将激活)



red edge indicates $w = -1$
blue edge indicates $w = +1$

x_1	x_2	XOR	h_1	h_2	h_3	h_4	$\sum_{i=1}^4 w_i h_i$
0	0	0	1	0	0	0	w_1
0	1	1	0	1	0	0	w_2
1	0	1	0	0	1	0	w_3
1	1	0	0	0	0	1	w_4

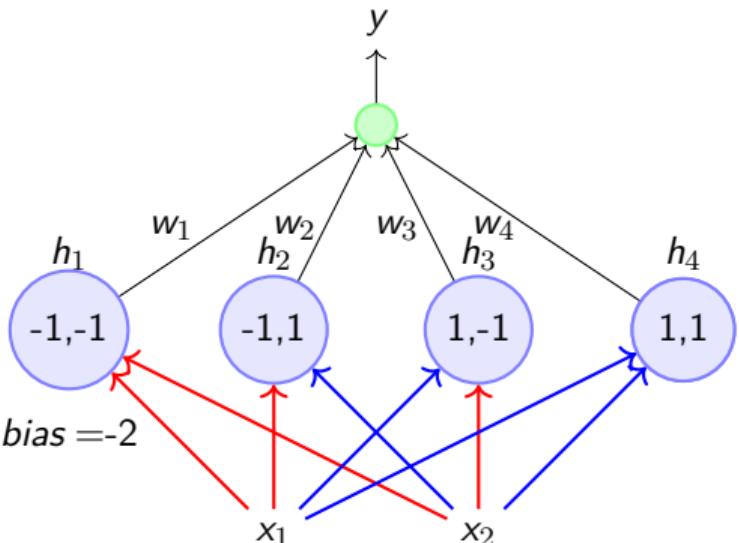


red edge indicates $w = -1$
blue edge indicates $w = +1$

- 让 w_0 表示输出神经元的偏置 (i.e., 如果 $\sum_{i=1}^4 w_i h_i \geq w_0$, 输出神经元将激活)

x_1	x_2	XOR	h_1	h_2	h_3	h_4	$\sum_{i=1}^4 w_i h_i$
0	0	0	1	0	0	0	w_1
0	1	1	0	1	0	0	w_2
1	0	1	0	0	1	0	w_3
1	1	0	0	0	0	1	w_4

- 实现 XOR 需要满足 4 个条件: $w_1 < w_0$, $w_2 \geq w_0$, $w_3 \geq w_0$, $w_4 < w_0$

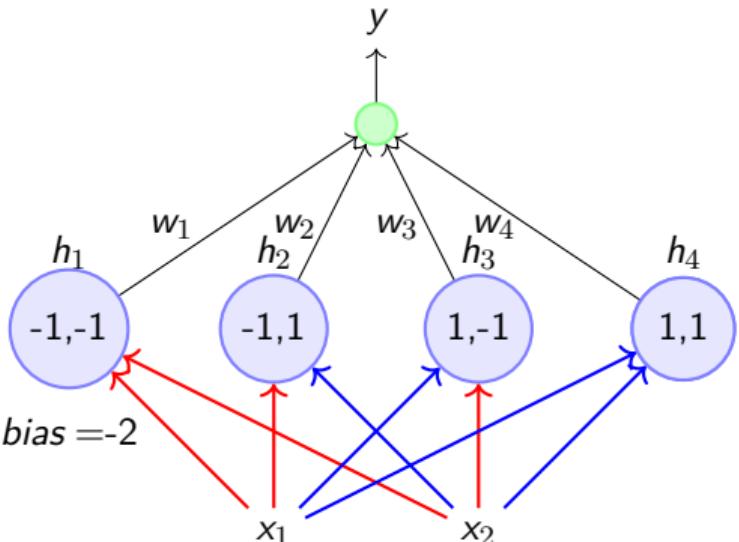


red edge indicates $w = -1$
blue edge indicates $w = +1$

- 让 w_0 表示输出神经元的偏置 (i.e., 如果 $\sum_{i=1}^4 w_i h_i \geq w_0$, 输出神经元将激活)

x_1	x_2	XOR	h_1	h_2	h_3	h_4	$\sum_{i=1}^4 w_i h_i$
0	0	0	1	0	0	0	w_1
0	1	1	0	1	0	0	w_2
1	0	1	0	0	1	0	w_3
1	1	0	0	0	0	1	w_4

- 实现 XOR 需要满足 4 个条件: $w_1 < w_0, w_2 \geq w_0, w_3 \geq w_0, w_4 < w_0$
- 不像之前, 这四个条件没有冲突, 能找到一个解来满足它们



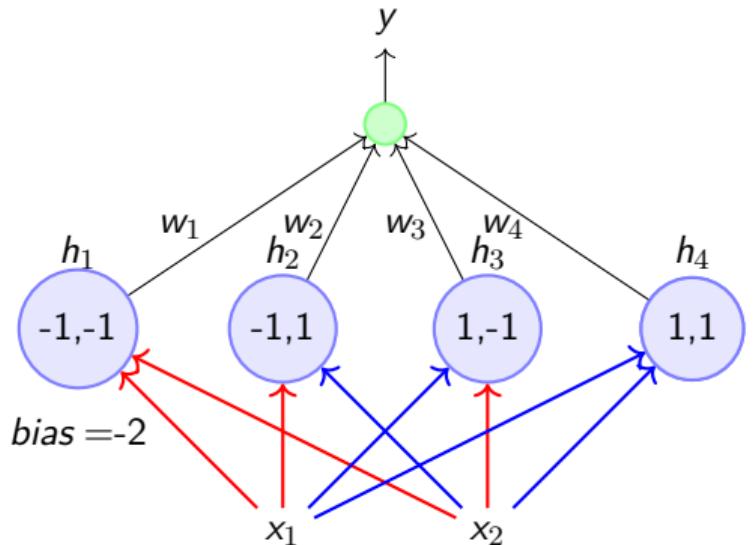
red edge indicates $w = -1$
blue edge indicates $w = +1$

- 让 w_0 表示输出神经元的偏置 (i.e., 如果 $\sum_{i=1}^4 w_i h_i \geq w_0$, 输出神经元将激活)

x_1	x_2	XOR	h_1	h_2	h_3	h_4	$\sum_{i=1}^4 w_i h_i$
0	0	0	1	0	0	0	w_1
0	1	1	0	1	0	0	w_2
1	0	1	0	0	1	0	w_3
1	1	0	0	0	0	1	w_4

- 实现 XOR 需要满足 4 个条件: $w_1 < w_0, w_2 \geq w_0, w_3 \geq w_0, w_4 < w_0$
- 不像之前, 这四个条件没有冲突, 能找到一个解来满足它们
- 本质上来说, 每个 w_i 负责处理 4 个可能的输入中的一个, 通过调整 w_i 可以得到对该输入的期望的输出

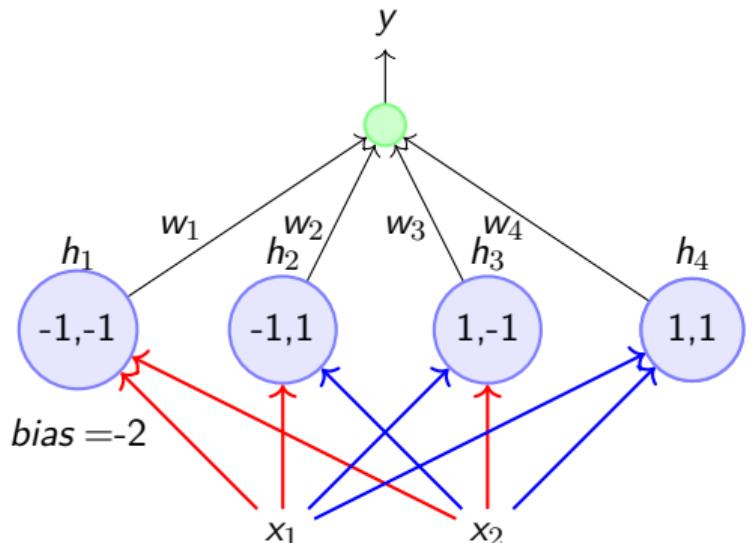
- 值得注意：同样的网络能够用来实现剩下的 15 个布尔函数



red edge indicates $w = -1$

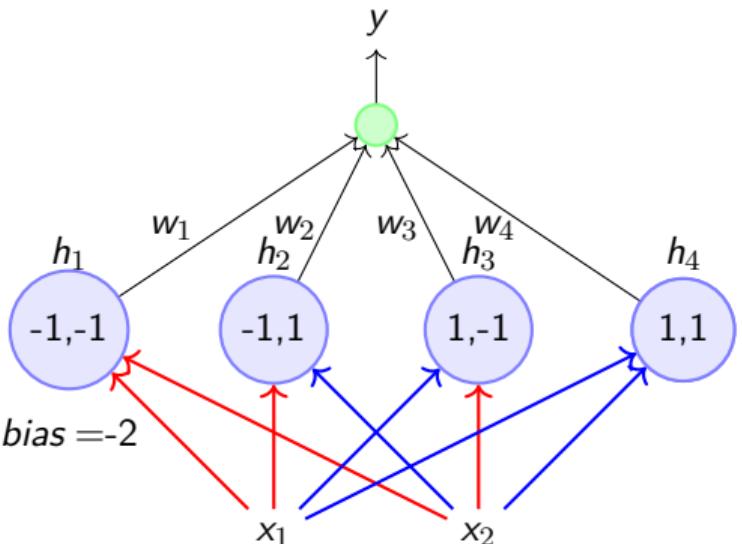
blue edge indicates $w = +1$

- 值得注意：同样的网络能够用来实现剩下的 15 个布尔函数
- 每个布尔函数会得到 4 个不相互冲突的不等式，能够求出 4 个不同的权重 w_1, w_2, w_3, w_4



red edge indicates $w = -1$

blue edge indicates $w = +1$



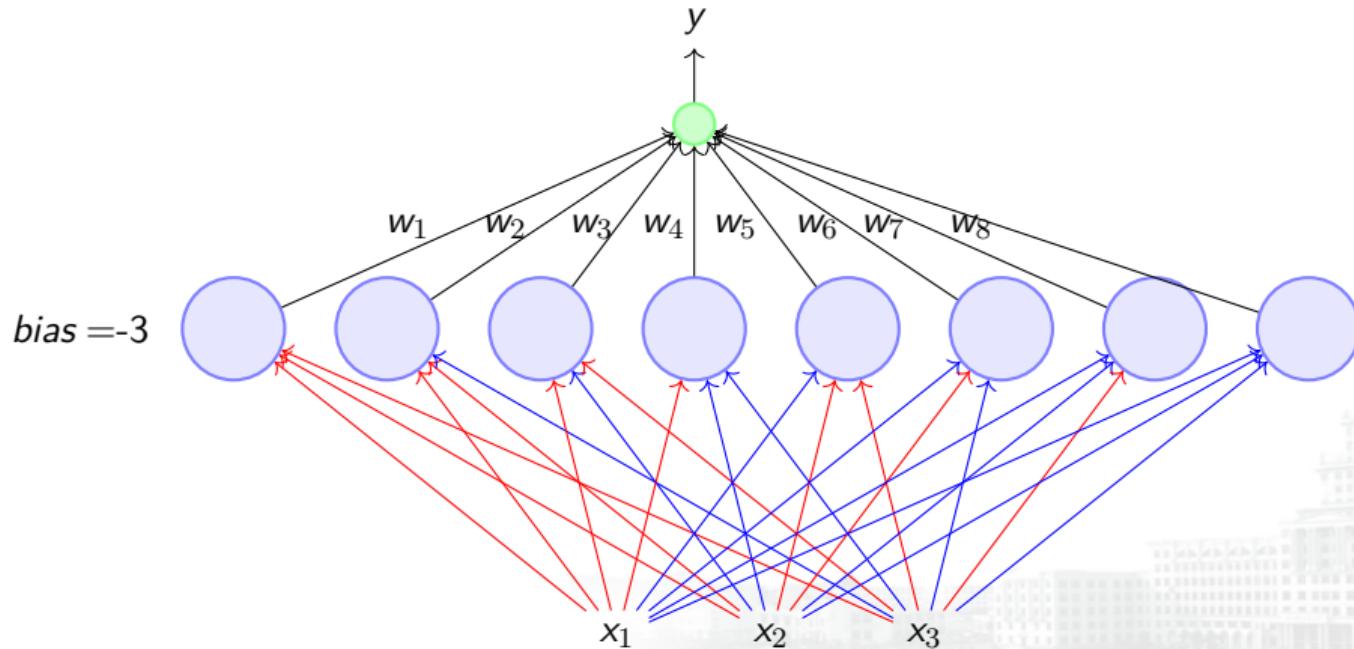
red edge indicates $w = -1$

blue edge indicates $w = +1$

- 值得注意：同样的网络能够用来实现剩下的 15 个布尔函数
- 每个布尔函数会得到 4 个不相互冲突的不等式，能够求出 4 个不同的权重 w_1, w_2, w_3, w_4
- Try it!

- 如果有 3 个输入，该如何处理？

- 八个感知机中的每一个将仅对八个输入中的一个进行『开火』
- 第二层的每一个权重负责一个输入，不同的权重决定整个网络对输入的不同响应



- 如果有 n 个输入，该如何处理？



Theorem

任何有 n 个输入的布尔函数能够由一个感知机网络实现，该网络包含一个有 2^n 个感知机的隐含层和一个有一个感知机的输出层

- 为什么我们对布尔函数感兴趣?



- 为什么我们对布尔函数感兴趣?
- 我们最初的问题: 预测是否喜欢看一部电影?

- 为什么我们对布尔函数感兴趣?
- 我们最初的问题: 预测是否喜欢看一部电影? Let us see!

- 过去观看电影的历史数据作为训练数据

$$\begin{array}{l} p_1 \left[\begin{matrix} x_{11} & x_{12} & \dots & x_{1n} & y_1 = 1 \end{matrix} \right] \\ p_2 \left[\begin{matrix} x_{21} & x_{22} & \dots & x_{2n} & y_2 = 1 \end{matrix} \right] \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ n_1 \left[\begin{matrix} x_{k1} & x_{k2} & \dots & x_{kn} & y_i = 0 \end{matrix} \right] \\ n_2 \left[\begin{matrix} x_{j1} & x_{j2} & \dots & x_{jn} & y_j = 0 \end{matrix} \right] \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \end{array}$$

- 过去观看电影的历史数据作为训练数据
- 对每一部电影，给定它的特征（也就是我们做决定所依据的因素） (x_1, x_2, \dots, x_n) 和标签 y （喜欢 =1/不喜欢 =0）

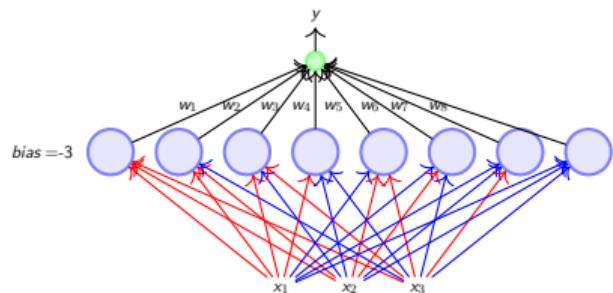
$$\begin{array}{l} p_1 \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} & y_1 = 1 \end{bmatrix} \\ p_2 \begin{bmatrix} x_{21} & x_{22} & \dots & x_{2n} & y_2 = 1 \end{bmatrix} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ n_1 \begin{bmatrix} x_{k1} & x_{k2} & \dots & x_{kn} & y_i = 0 \end{bmatrix} \\ n_2 \begin{bmatrix} x_{j1} & x_{j2} & \dots & x_{jn} & y_j = 0 \end{bmatrix} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \end{array}$$

- 过去观看电影的历史数据作为训练数据
- 对每一部电影，给定它的特征（也就是我们做决定所依据的因素） (x_1, x_2, \dots, x_n) 和标签 y （喜欢 = 1/不喜欢 = 0）
- p_i 's 是所有喜欢的电影， n_i 's 是所有不喜欢的电影

$$\begin{array}{ll} p_1 & \left[\begin{array}{ccccc} x_{11} & x_{12} & \dots & x_{1n} & y_1 = 1 \end{array} \right] \\ p_2 & \left[\begin{array}{ccccc} x_{21} & x_{22} & \dots & x_{2n} & y_2 = 1 \end{array} \right] \\ \vdots & \left[\begin{array}{ccccc} \vdots & \vdots & \vdots & \vdots & \vdots \end{array} \right] \\ n_1 & \left[\begin{array}{ccccc} x_{k1} & x_{k2} & \dots & x_{kn} & y_i = 0 \end{array} \right] \\ n_2 & \left[\begin{array}{ccccc} x_{j1} & x_{j2} & \dots & x_{jn} & y_j = 0 \end{array} \right] \\ \vdots & \left[\begin{array}{ccccc} \vdots & \vdots & \vdots & \vdots & \vdots \end{array} \right] \end{array}$$

- 过去观看电影的历史数据作为训练数据
- 对每一部电影，给定它的特征（也就是我们做决定所依据的因素） (x_1, x_2, \dots, x_n) 和标签 y （喜欢 = 1/不喜欢 = 0）
- p_i 's 是所有喜欢的电影， n_i 's 是所有不喜欢的电影
- 训练数据可能是线性可分的，也可能是线性不可分的

$$\begin{array}{ll} p_1 & \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1n} & y_1 = 1 \end{bmatrix} \\ p_2 & \begin{bmatrix} x_{21} & x_{22} & \dots & x_{2n} & y_2 = 1 \end{bmatrix} \\ \vdots & \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \\ n_1 & \begin{bmatrix} x_{k1} & x_{k2} & \dots & x_{kn} & y_i = 0 \end{bmatrix} \\ n_2 & \begin{bmatrix} x_{j1} & x_{j2} & \dots & x_{jn} & y_j = 0 \end{bmatrix} \\ \vdots & \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \end{array}$$



$$\begin{array}{l} p_1 \left[\begin{array}{ccccc} x_{11} & x_{12} & \dots & x_{1n} & y_1 = 1 \end{array} \right] \\ p_2 \left[\begin{array}{ccccc} x_{21} & x_{22} & \dots & x_{2n} & y_2 = 1 \end{array} \right] \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \\ n_1 \left[\begin{array}{ccccc} x_{k1} & x_{k2} & \dots & x_{kn} & y_i = 0 \end{array} \right] \\ n_2 \left[\begin{array}{ccccc} x_{j1} & x_{j2} & \dots & x_{jn} & y_j = 0 \end{array} \right] \\ \vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots \end{array}$$

- 过去观看电影的历史数据作为训练数据
- 对每一部电影，给定它的特征（也就是我们做决定所依据的因素） (x_1, x_2, \dots, x_n) 和标签 y （喜欢 = 1/不喜欢 = 0）
- p_i 's 是所有喜欢的电影， n_j 's 是所有不喜欢的电影
- 训练数据可能是线性可分的，也可能是线性不可分的
- 我们可以构造一个感知机网络，并且训练它的权重使得对任意给定的 p_i 或 n_j ，网络的输出与标签值 y_i 或 y_j 相同 (i.e., 我们能够区分正样本和负样本)



多层感知机

- 由一个输入层、一个输出层（一个感知机模型）、一个或多个隐含层（多个感知机模型）构成的网络称作多层感知机（Multilayer Perceptrons, MLP, in short）



多层感知机

- 由一个输入层、一个输出层（一个感知机模型）、一个或多个隐含层（多个感知机模型）构成的网络称作多层感知机（Multilayer Perceptrons, MLP, in short）
- 更准确的说法是“Multilayered Network of Perceptrons”，但 MLP 更常用

多层感知机

- 由一个输入层、一个输出层（一个感知机模型）、一个或多个隐含层（多个感知机模型）构成的网络称作多层感知机（Multilayer Perceptrons, MLP, in short）
- 更准确的说法是“Multilayered Network of Perceptrons”，但 MLP 更常用
- 具有一层隐含层的 MLP 能够表示 任意布尔函数