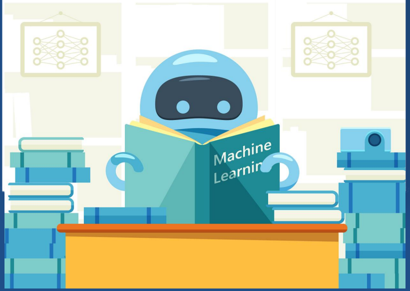
山东大学软件学院
SCHOOL OF SOFTWARE, SHANDONG UNIVERSITY

Machine Learning

机器学习



Chapter 6
Neural Networks

软件学院 罗昕

luoxin@sdu.edu.cn

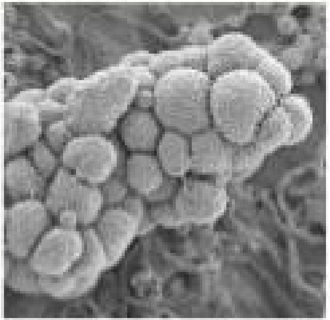
软件学院办公楼-425

DEEP LEARNING EVERYWHERE

MIMA

INTERNET & CLOUD

Image Classification
Speech Recognition
Language Translation
Sentiment Analysis
Recommendation



MEDICINE & BIOLOGY

Cancer Cell Detection
Diabetic Grading
Drug Discovery

MEDIA & ENTERTAINMENT

Video Captioning
Video Search
Real Time Translation

SECURITY & DEFENSE

Face Detaction
Video Surveilance
Satellite Imagery

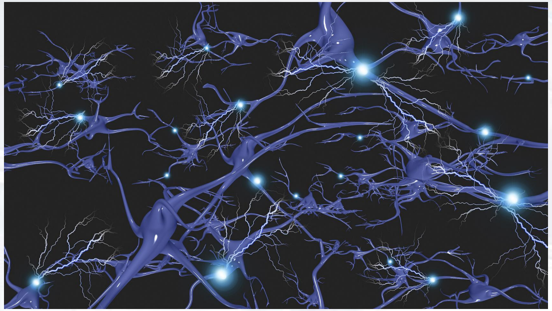
AUTONOMOUS & MACHINES

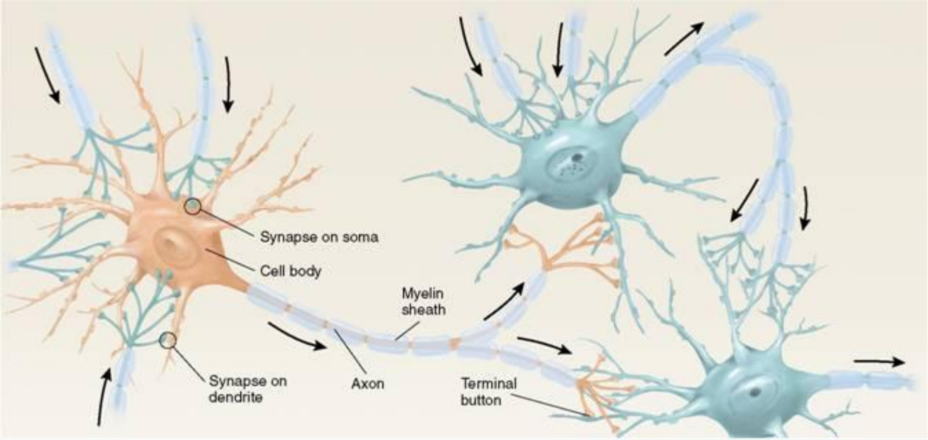
Pedestrian Detection
Lane Tracking
Recognize Traffic Sign

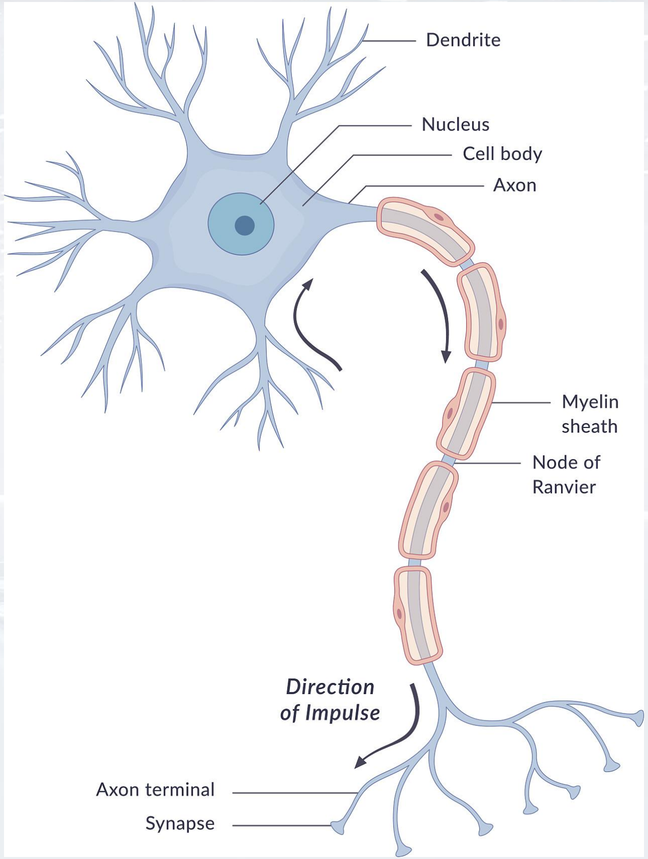
Contents

MIMA

Biological Neural Systems



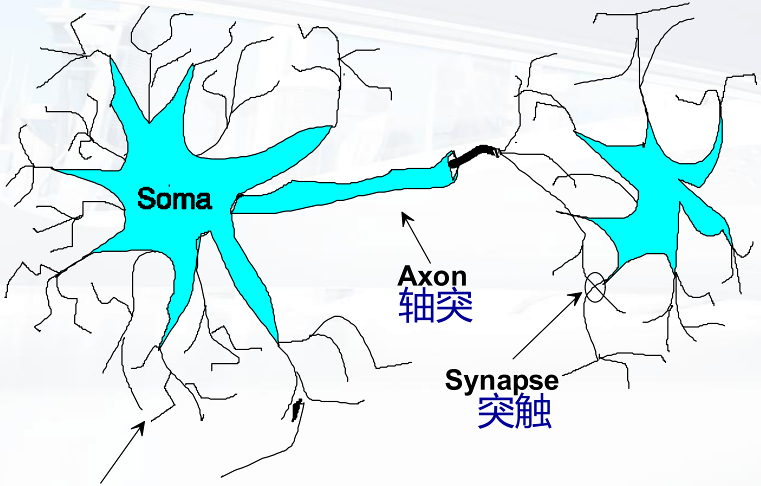




Biological Neural Systems

MIMA

- The brain is composed of approximately 100 billion (10^{11}) neurons

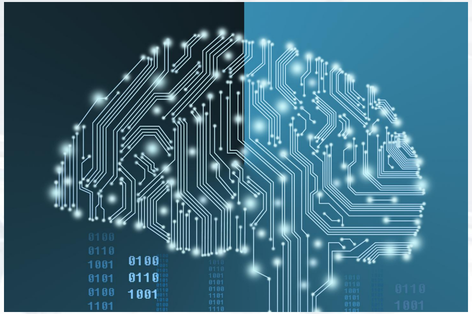


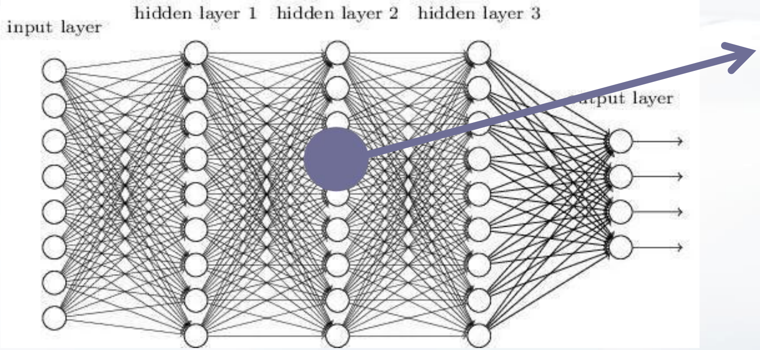
Schematic drawing of two biological neurons connected by synapses

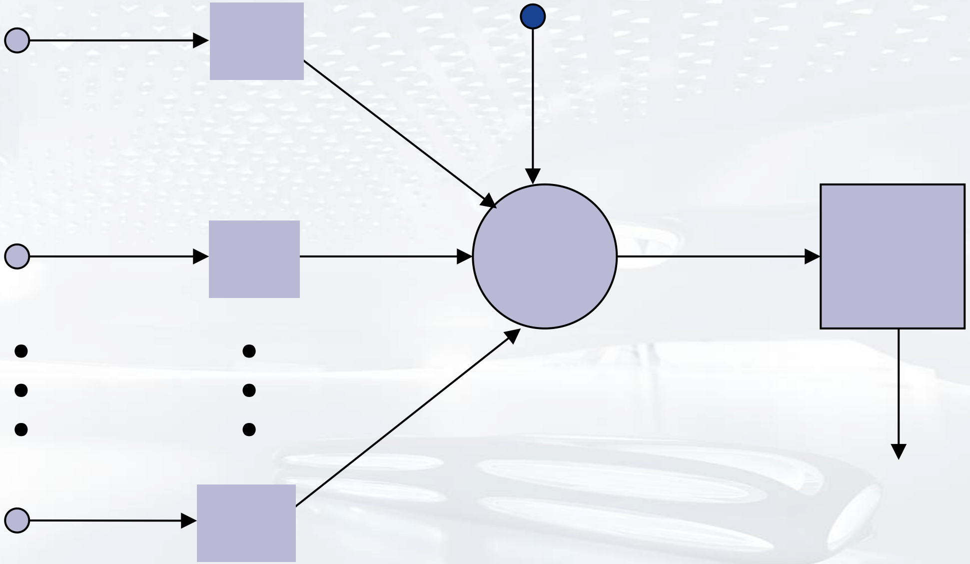
- A typical neuron collects signals from other neurons through a host of fine structures called **dendrites**（树突）.
- The neuron sends out spikes of electrical activity through a long, thin strand known as an **axon**（轴突）, which splits into thousands of branches.
- At the end of the branch, a structure called a **synapse**（突触）converts the activity from the axon into electrical effects that inhibit or excite activity in the connected neurons.
- When a neuron receives excitatory input that is sufficiently large compared with its inhibitory input, it sends a spike of electrical activity down its axon.

Learning occurs by changing the effectiveness of the synapses so that the influence of one neuron on the other changes

人工神经网络







Introduction

MIMA

- (Artificial) Neural Networks are
  - Computational models which mimic the brain's learning processes.
  - They have the essential features of neurons and their interconnections as found in the brain.
  - Typically, a computer is programmed to simulate these features.
- Other definitions …
  - A neural network is a massively parallel distributed processor made up of simple processing units, which has a natural propensity for storing experimental knowledge and making it available for use. It resembles the brain in two respects:
    - Knowledge is acquired by the network from its environment through a learning process.
    - Interneuron connection strengths, known as synaptic weights, are used to store the acquired knowledge.

Introduction

MIMA

- A neural network is a machine learning approach inspired by the way in which the brain performs a particular learning task:
  - Knowledge about the learning task is given in the form of examples.
  - Inter neuron connection strengths (weights) are used to store the acquired information (the training examples) .
  - During the learning process the weights are modified in order to model the particular learning task correctly on the training examples.

Neural Networks

- A NN is a machine learning approach inspired by the way in which the brain performs a particular learning task
- Various types of neurons
- Various network architectures
- Various learning algorithms
- Various applications

Characteristics of NN' s

- Characteristics of Neural Networks
  - Large scale and parallel processing
  - Robust
  - Self-adaptive and organizing
  - Good enough to simulate non-linear relations
  - Hardware

Applications

- Combinatorial Optimization
- Pattern Recognition
- Bioinformatics
- Text processing
- Natural language processing
- Data Mining
- …

Types

- Structure
  - Feed-forward
  - Feed-back
- Learning method
  - Supervised
  - Unsupervised
- Signal type
  - Continuous
  - Discrete

Historical Background

- 1943 McCulloch and Pitts proposed the first computational models of neuron.
- 1949 Hebb proposed the first learning rule.
- 1958 Rosenblatt’ s work in perceptrons（感知器）.
- 1969 Minsky and Papert exposed limitation of the theory.
- 1970s Decade of dormancy for neural networks.
- 1980-90s Neural network return (self-organization, back-propagation algorithms, etc)

Historical Background

- 弗兰克·罗森布拉特

(Frank Rosenblatt，康奈尔大学的心理学家)

- 1958年，在《纽约时报 (New York Times)》上发表文章《Electronic ‘Brain’ Teaches Itself.》，正式把算法取名为“**感知器**”

- 它有400个光传感器，它们一起充当视网膜，将信息传递给大约1000个“神经元”，这些神经元进行处理并输出单一信息。

图片来源https://baijiahao.baidu.com/s?id=1713433120837173170&wfr=spider&for=pc

Historical Background

MIMA

- 马文·明斯基

  "人工智能之父" (Marvin Minsky) 1970图灵奖获得者

- 1969年，Minsky 和Papert所著的《Perceptron》一书出版，从数学角度证明了关于单层感知器的计算具有根本的局限性，指出感知器的处理能力有限，甚至连XOR这样的问题也不能解决

- 神经网络进入了萧条期
- 第一个**人工智能冬天**

<u>计算机有限的内存和处理速度不足以解决任何实际的人工智能问题</u>

——《人工智能发展简史》

中共中央网络安全和信息化委员会办公室
中华人民共和国国家互联网信息办公室
http://www.cac.gov.cn/2017-01/23/c_1120366748.htm

图片来源https://baijiahao.baidu.com/s?id=1713433120837173170&wfr=spider&for=pc

Historical Background

- 杰弗里·辛顿

  “神经网络之父”(Geoffrey Hinton) 2019图灵奖获得者

- 多伦多大学的辛顿实现了一种叫做**反向传播**的原理来让神经网络从他们的错误中学习 1986

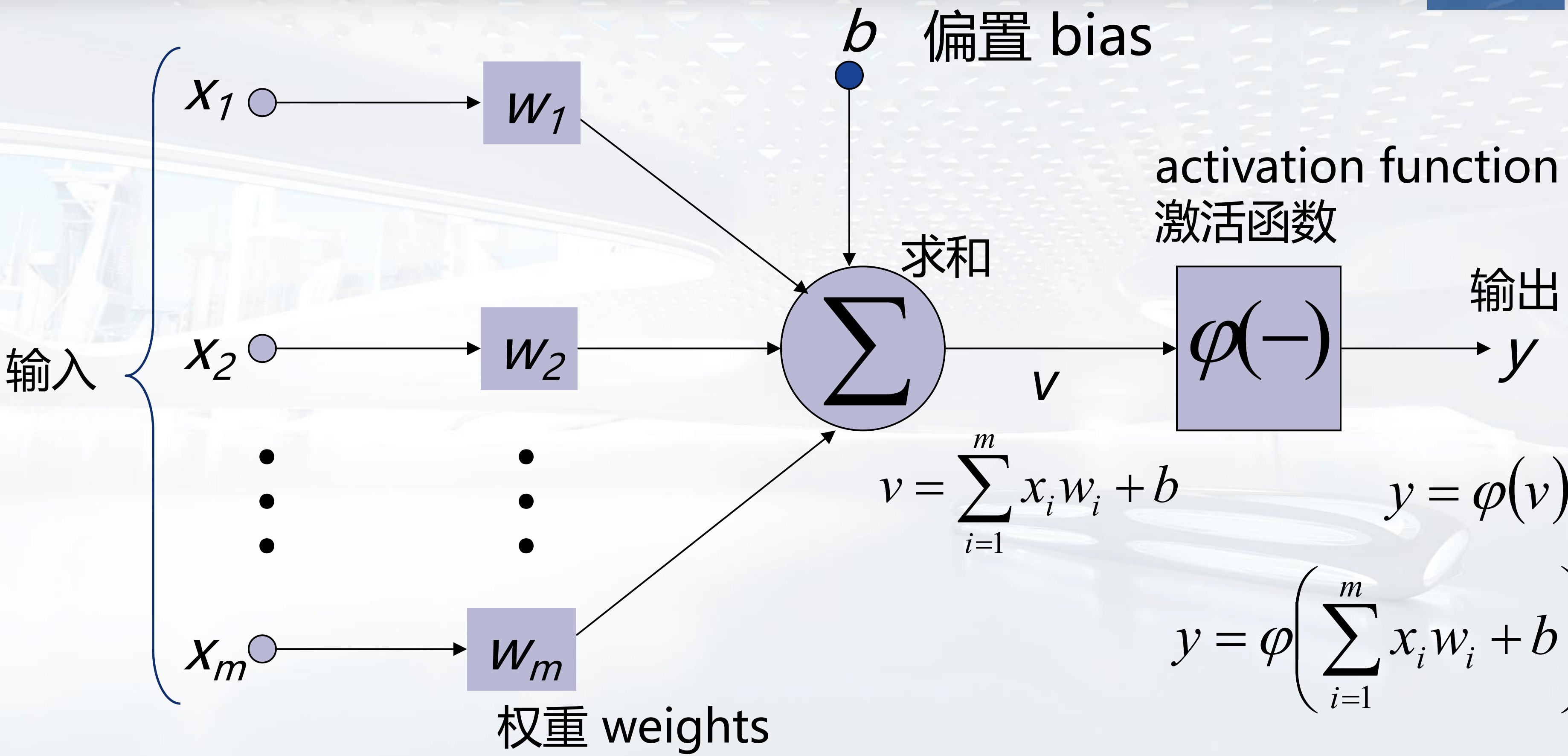
- 为人工智能的发展奠定了基础
- **“数据、算法、算力”**人工智能三要素

- 2004年IEEE Frank Rosenblatt Award成立，Frank Rosenblatt被尊称为神经网络的创立者

图片来源https://baijiahao.baidu.com/s?id=1713433120837173170&wfr=spider&for=pc

Contents

感知机



$$v=\sum_{i=1}^{m} x_i w_i + b$$

$$y=\varphi(v)$$

$$y=\varphi\left(\sum_{i=1}^{m} x_i w_i + b\right)$$

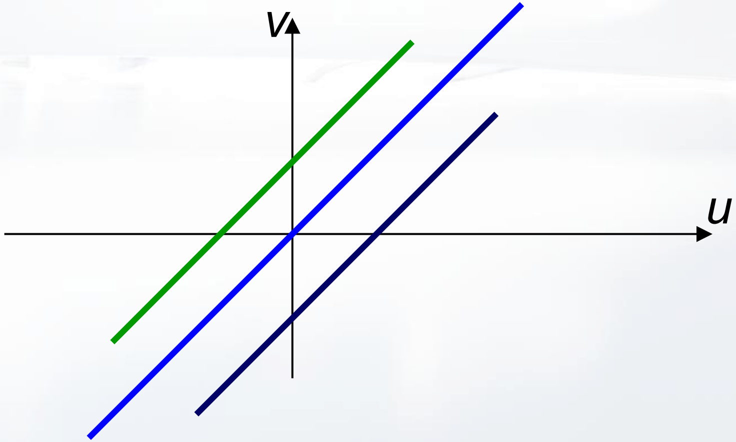
感知机 - Bias of a Neuron

MIMA

- Bias b has the effect of applying an affine transformation to u

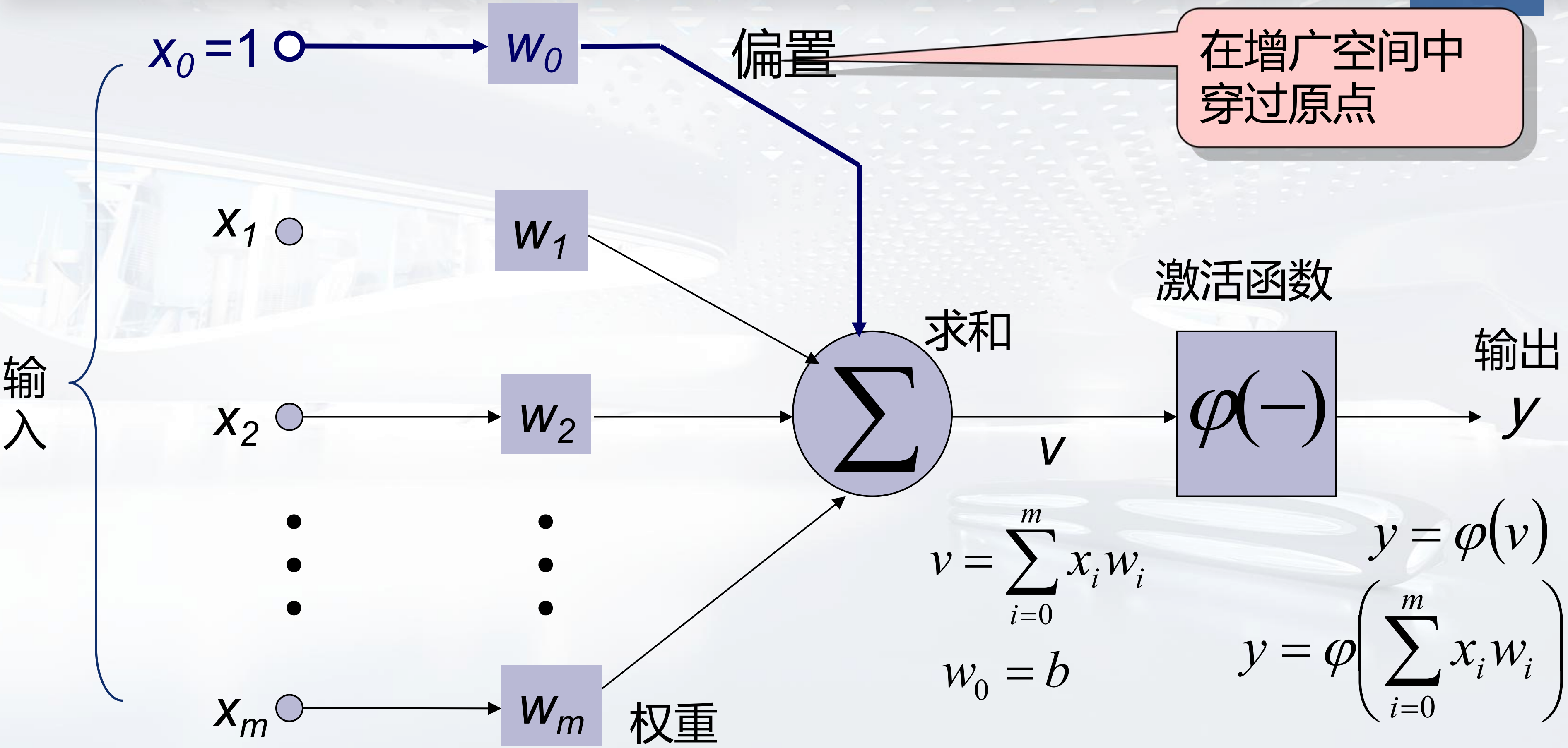
$$v = u + b$$

- v is the **induced field** of the neuron



$$u = \sum_{j=1}^{m} w_j x_j$$

感知机 - Bias as Extra Input



$$v = \sum_{i=0}^{m} x_i w_i$$

$$w_0 = b$$

$$y = \varphi(v)$$

$$y = \varphi\left(\sum_{i=0}^{m} x_i w_i\right)$$

The Neuron

MIMA

- The neuron is the basic information processing unit of a NN. It consists of:

  1 A set of synapses or connecting links, each link characterized by a weight:

  $$W_1,\ W_2,\ \ldots,\ W_m$$

  2 An adder function (linear combiner) which computes the weighted sum of the inputs:

  $$v = \sum_{i=1}^{m} x_i w_i + b \qquad\qquad v = \sum_{i=0}^{m} x_i w_i$$

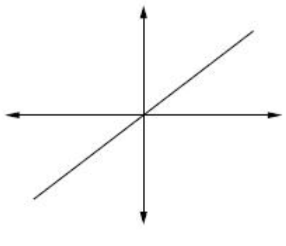
  3 Activation function (squashing function) for limiting the amplitude of the output of the neuron.

  $$y = \varphi(v) \qquad\qquad y = \varphi\left(\sum_{i=0}^{m} x_i w_i\right)$$

感知机 - 激活函数

- 1.线性函数 linear function

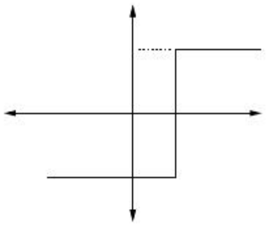
$$f(x) = ax$$



Linear function

- 2. 阶梯函数 step function

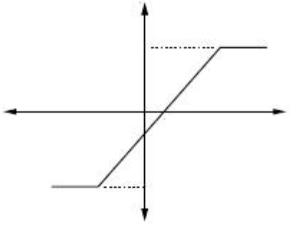
$$f(x) = \begin{cases} a_1 & if\ x \geq \theta \\ a_2 & if\ x < \theta \end{cases}$$



Step function

- 3. 斜坡函数 ramp function

$$f(x) = \begin{cases} \alpha & if\ x \geq \theta \\ x & if\ -\theta < x < \theta \\ -\alpha & if\ x \leq \theta \end{cases}$$



Ramp function

感知机 - 激活函数

MIMA

- 4. 逻辑函数 logistic function

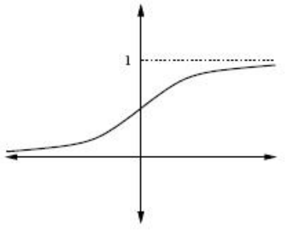
$$f(x) = \frac{1}{1+e^{-\lambda x}}$$

- 5. 双曲正切函数 hyperbolic tangent

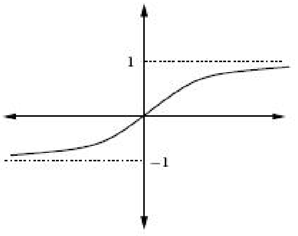
$$f(x) = \frac{e^{\lambda x} - e^{-\lambda x}}{e^{\lambda x} + e^{-\lambda x}}$$

- 6. 高斯函数 Gaussian function

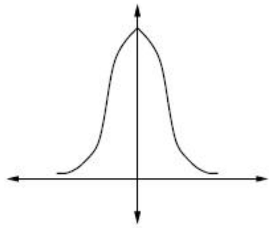
$$f(x) = e^{-x^2/\sigma^2}$$



Sigmoid function



Hyperbolic tangent function



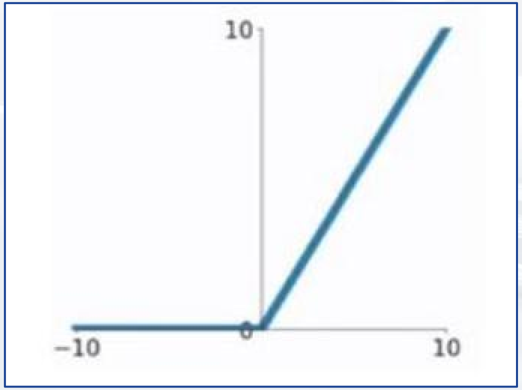
Gaussian function

Activation Function

- 7. ReLU function

修正线性单元 (Rectified Linear Unit)

$$g(z) = \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{if } z < 0 \end{cases}$$



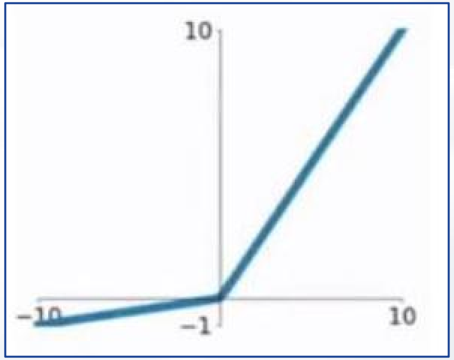
Activation Function

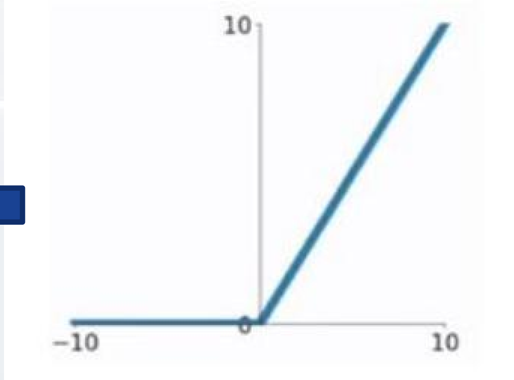
- 8. Leaky ReLU function

又称为PReLU函数

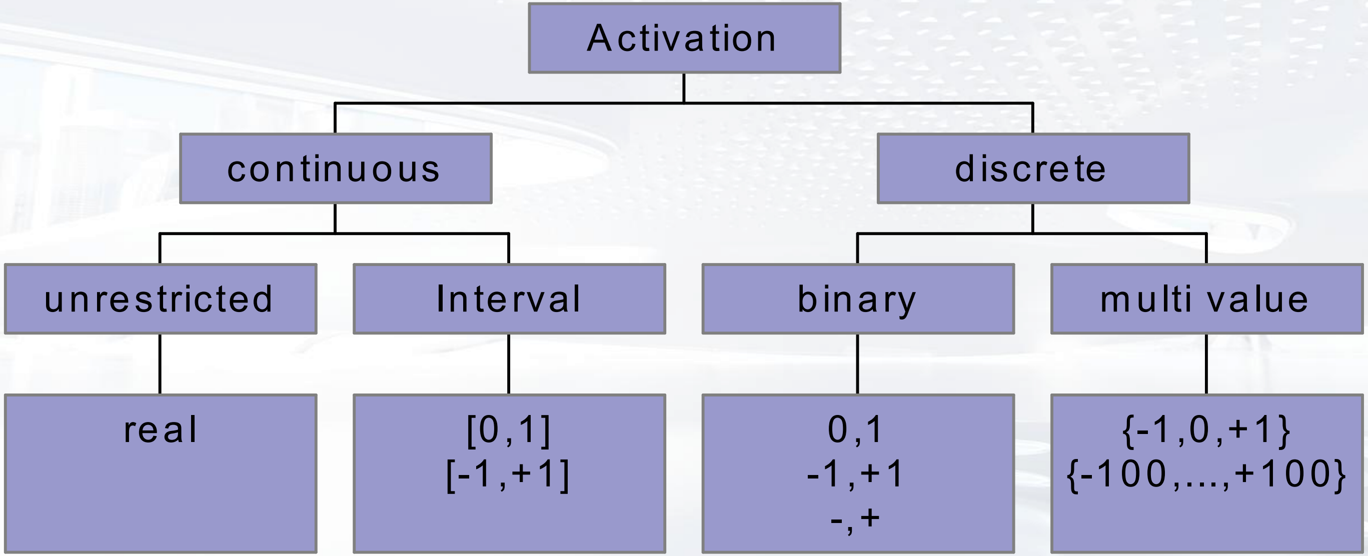
$$g(z) = \begin{cases} z, & \text{if } z > 0 \\ az, & \text{if } z < 0 \end{cases}$$

$$g(z) = \begin{cases} z, & \text{if } z > 0 \\ 0, & \text{if } z < 0 \end{cases}$$





Activation function

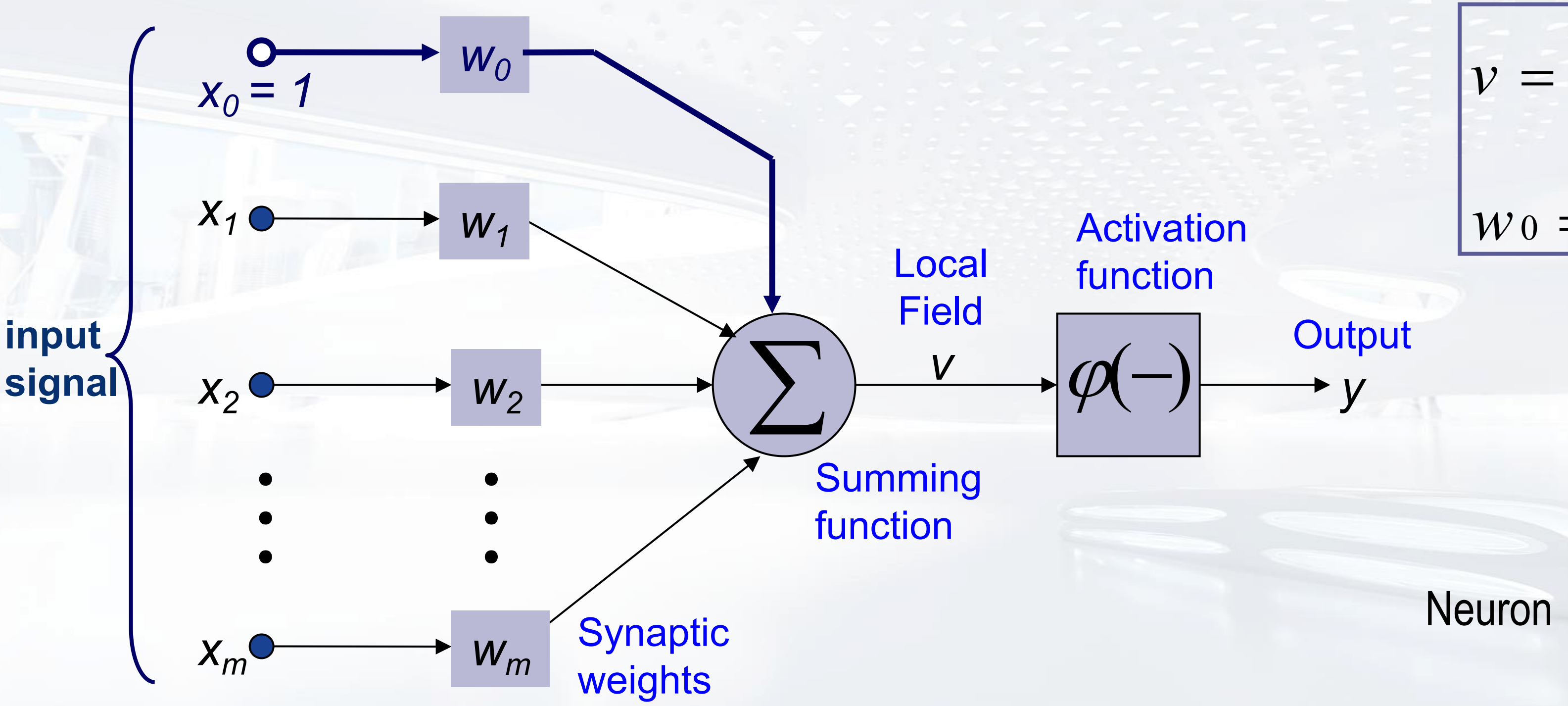


Perceptron

MIMA

- In 1943, McCulloch and Pitts proposed the first single neuron model.
- Hebb proposed the theory that the learning process is generated from the change of weights between synapses.
- In 1958, Rosenblatt combined them together, and proposed "Perceptron".
- Perceptron is just a single neural model, and is composed of synaptic weights and threshold.
- It is the simplest and earliest neural network model, used for **classification**.

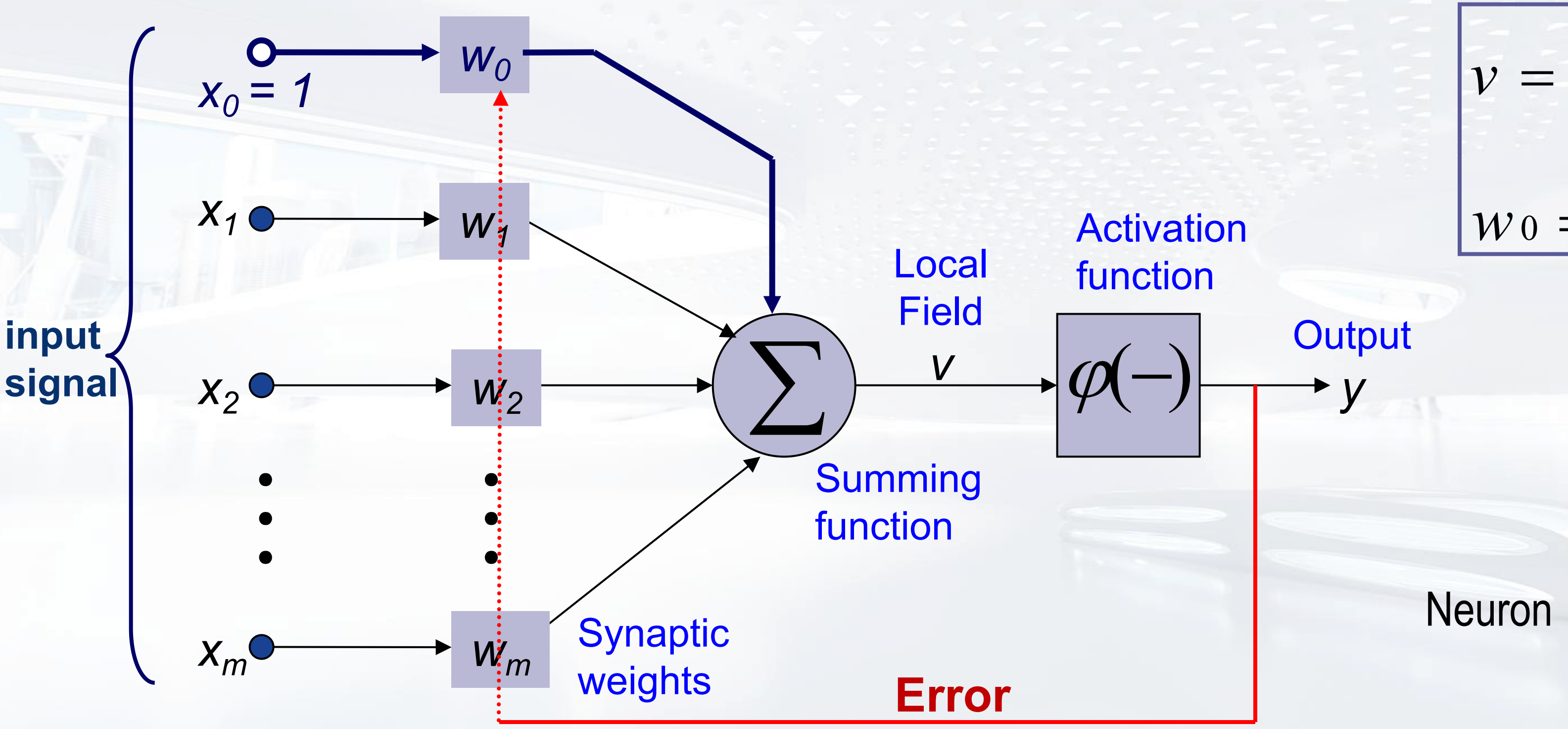
Perceptron



$$v = \sum_{j=0}^{m} w_j x_j$$

$$w_0 = b$$

Perceptron



$$v = \sum_{j=0}^{m} w_j x_j$$

$$w_0 = b$$

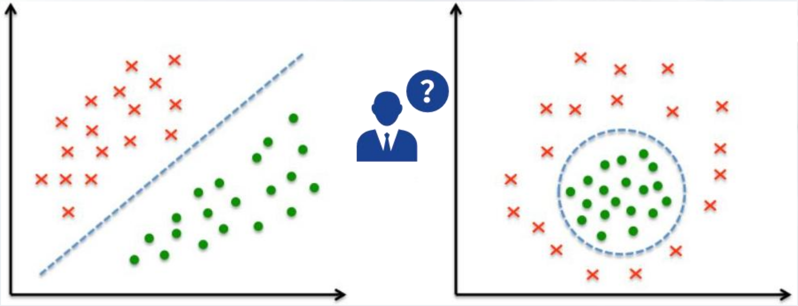
感知机

MIMA

- 我们有训练集 $T_1 \in C_1$ 和 $T_2 \in C_2$ 。其中，样本表示为 $\mathbf{x}=(x_0, x_1, x_2, \dots, x_m)^T$，且 $x_1, x_2, \dots, x_m \in R$ ， $x_0 = 1$。
- 假设 T_1 和 T_2 是 **线性可分的** linearly separable。
- 能否给出一个感知机将数据正确划分？ $\mathbf{w}=(w_0, w_1, w_2, \dots, w_m)^T$

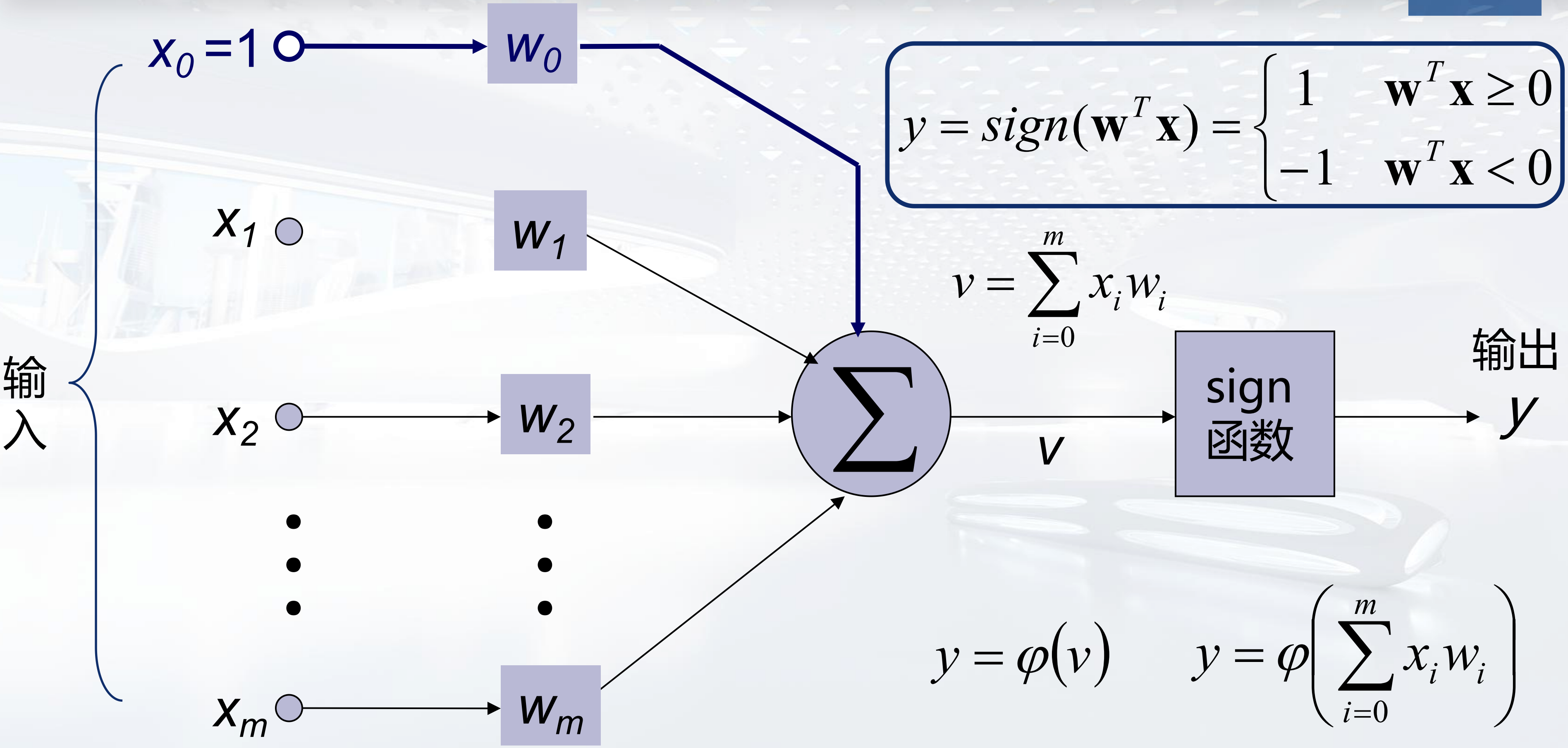
$d = +1$

$d = -1$



图片来源https://blog.csdn.net/pxhdky/article/details/85248575

感知机

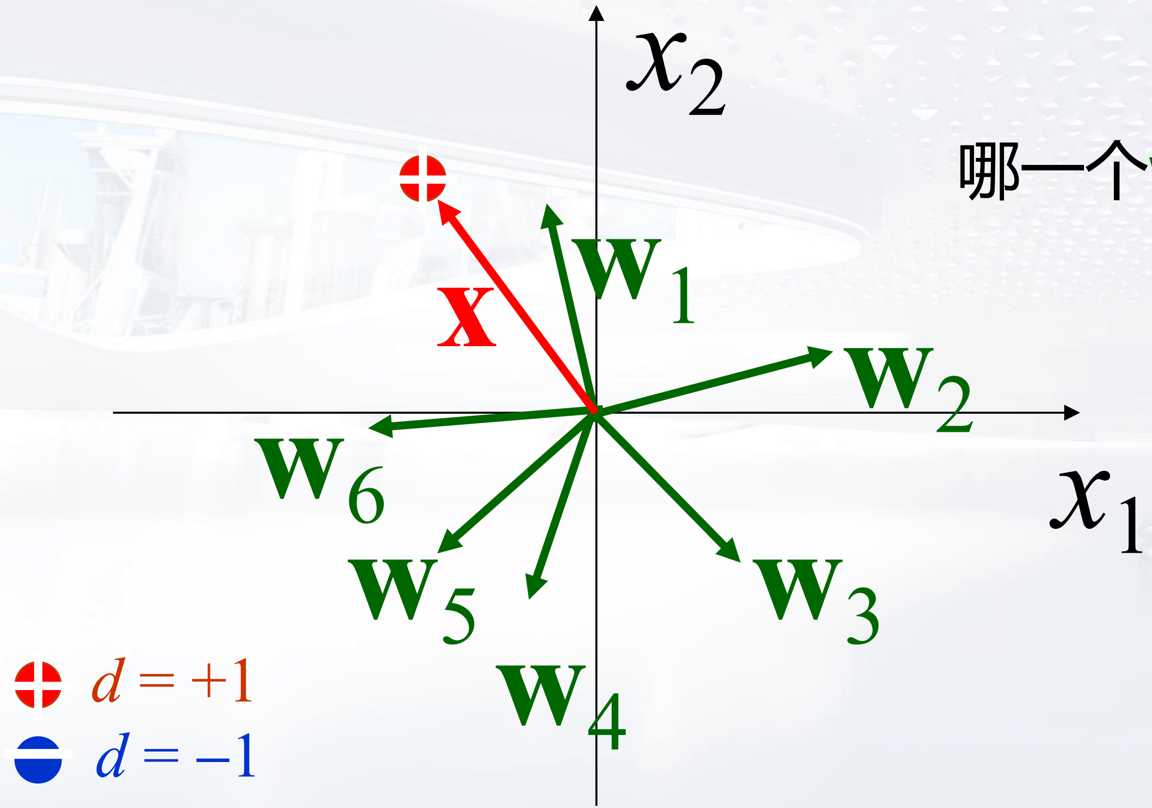


$$y = sign(\mathbf{w}^T\mathbf{x}) = \begin{cases} 1 & \mathbf{w}^T\mathbf{x} \geq 0 \\ -1 & \mathbf{w}^T\mathbf{x} < 0 \end{cases}$$

$$v = \sum_{i=0}^{m} x_i w_i$$

$$y = \varphi(v) \qquad y = \varphi\left(\sum_{i=0}^{m} x_i w_i\right)$$

感知机 - 求解

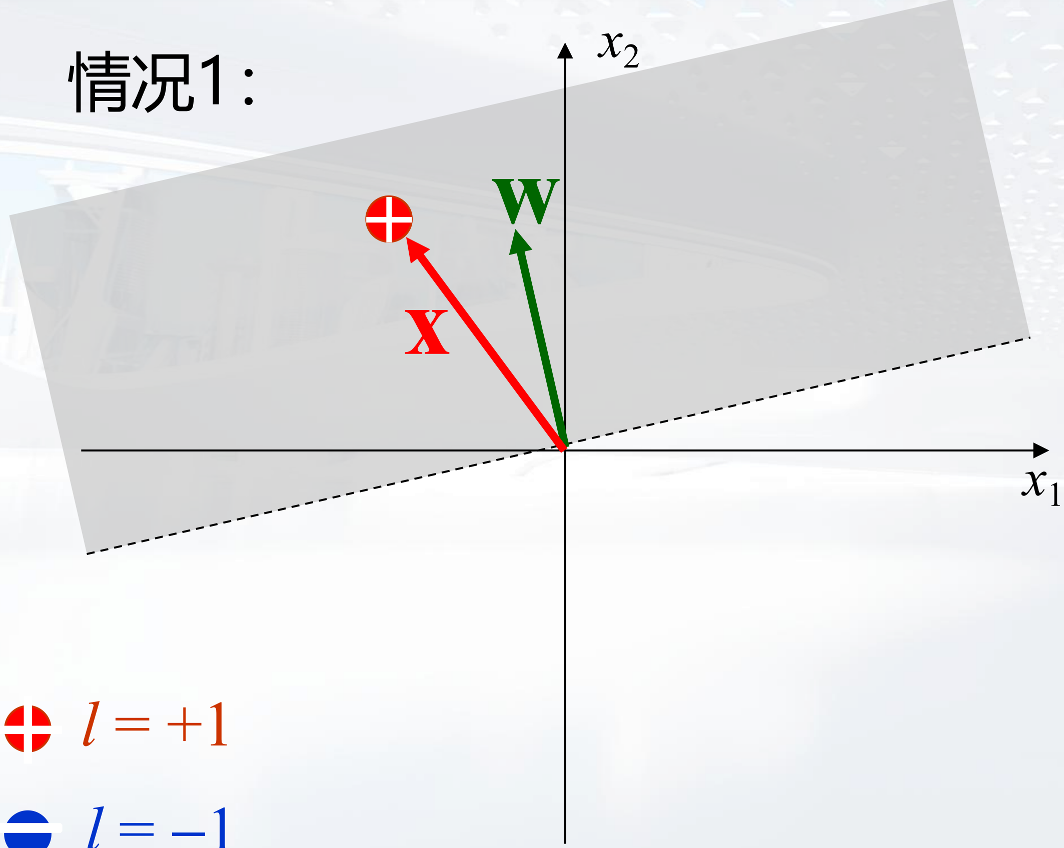


哪一个**w**可以正确的分类**x**?

感知机的学习机制

MIMA

情况1：



这个w代表的感知机可行吗?

可行 不行

$$\mathbf{w}^T\mathbf{x} > 0$$

$$\boxed{y = sign(\mathbf{w}^T\mathbf{x}) = 1}$$

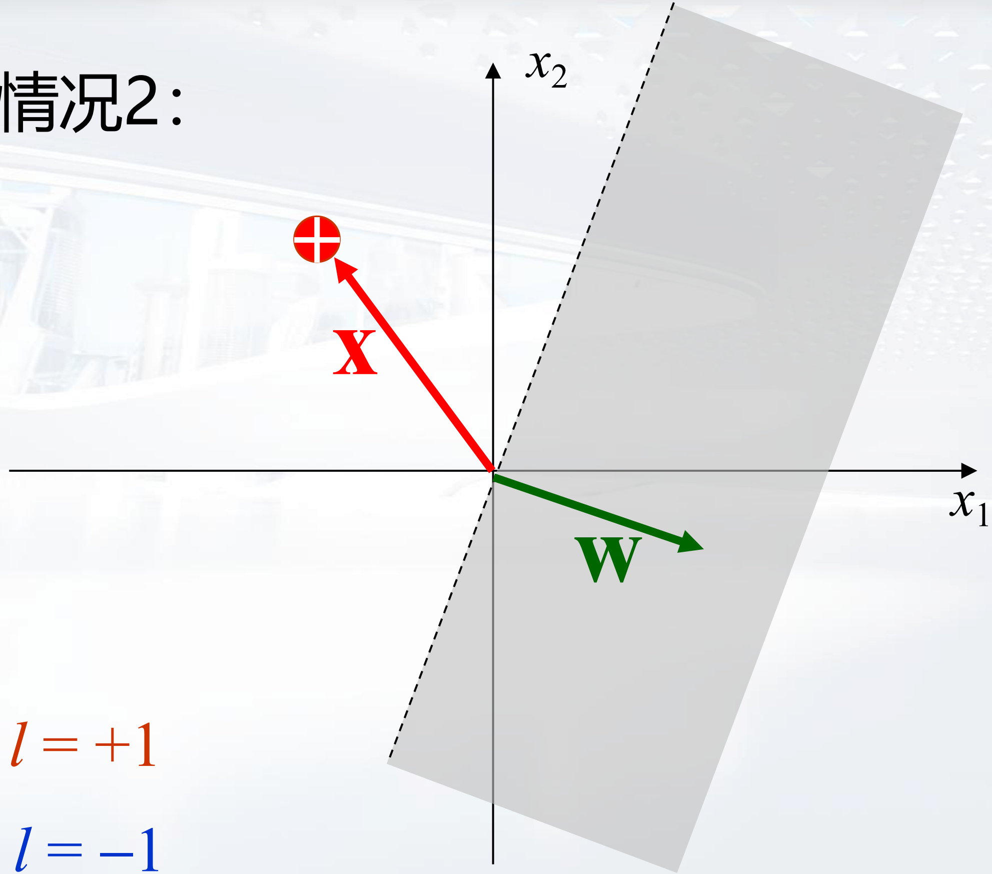
w 不需要更新（学习）

$l = +1$

$l = -1$

感知机的学习机制

情况2：



$l = +1$

$l = -1$

这个w代表的感知机可行吗？

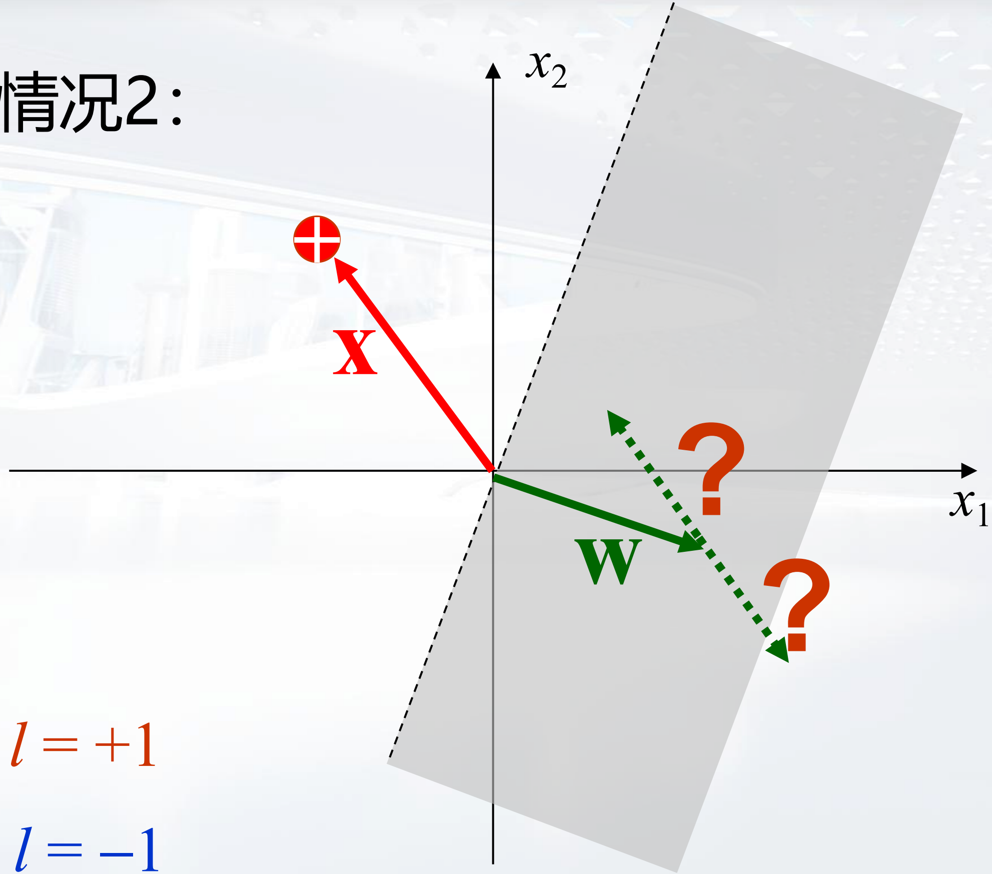
可行 不行

$\mathbf{w}^T\mathbf{x} < 0$ ✖

$$\boxed{y = sign(\mathbf{w}^T\mathbf{x}) = -1}$$

感知机的学习机制

情况2：



如何更新w让感知机变得可行?

$\mathbf{w}^T\mathbf{x} < 0$ 当前

$\mathbf{w}^T\mathbf{x} > 0$ 目标

$\Delta\mathbf{w} = ?$

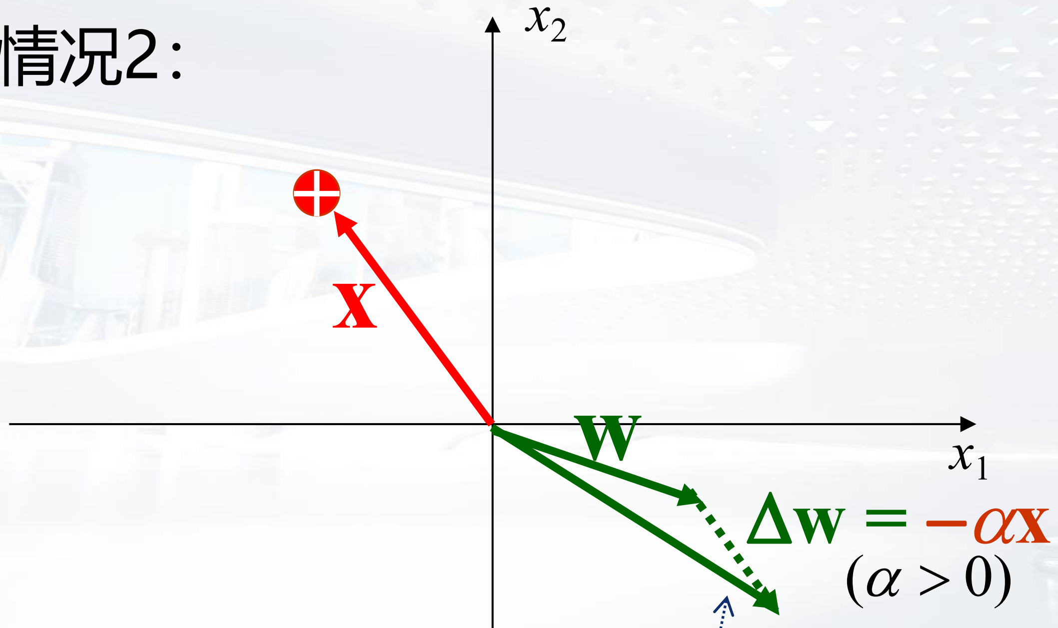
$l = +1$

$l = -1$

感知机的学习机制

MIMA

情况2：



如何更新**w**让感知机变得可行?

$\mathbf{w}^T\mathbf{x} < 0$ 当前

↓

$\mathbf{w}^T\mathbf{x} > 0$ 目标

$$(\mathbf{w}+\Delta\mathbf{w})^T\mathbf{x} = \underbrace{\mathbf{w}^T\mathbf{x}}_{<0} \underbrace{-\,\alpha\mathbf{x}^T\mathbf{x}}_{<0}$$

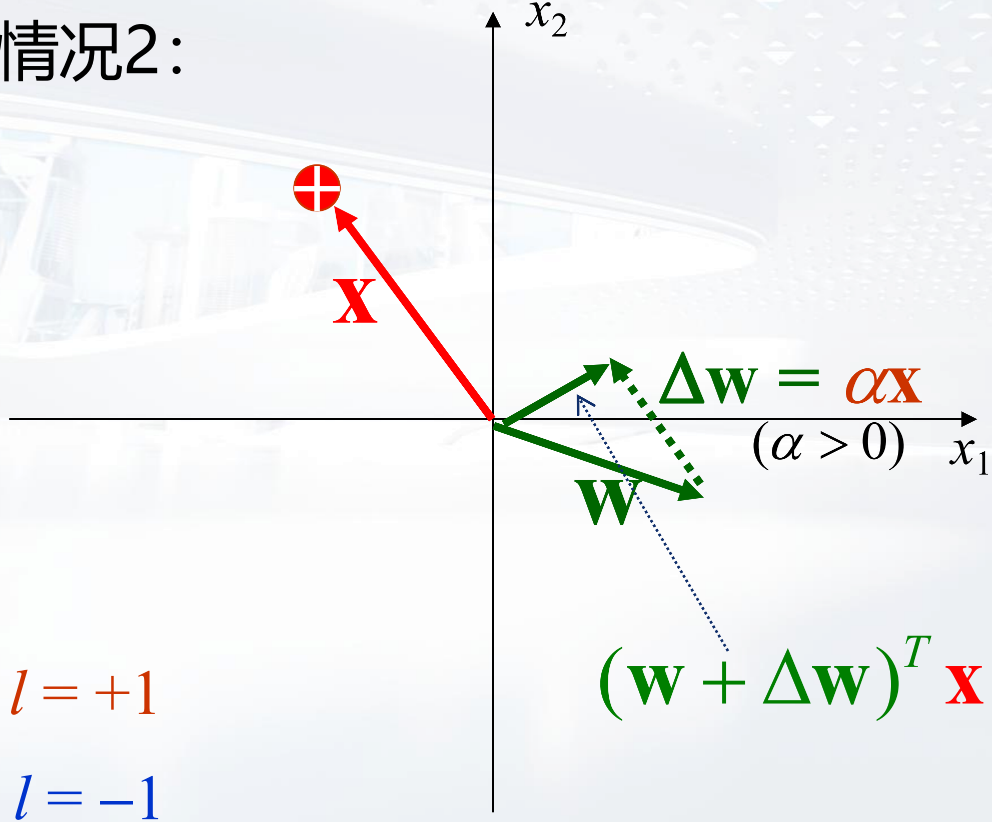
$l = +1$

$l = -1$

感知机的学习机制

MIMA

情况2：



如何更新**w**让感知机变得可行?

$\mathbf{w}^T\mathbf{x} < 0$ 当前

$\mathbf{w}^T\mathbf{x} > 0$ 目标

$$(\mathbf{w}+\Delta\mathbf{w})^T\mathbf{x} = \underbrace{\mathbf{w}^T\mathbf{x}}_{<0} + \underbrace{\alpha\mathbf{x}^T\mathbf{x}}_{>0}$$

$l = +1$

$l = -1$

感知机的学习机制

- 如果分类正确的话（情况1）　$l = y = 1$　$\mathbf{W}$ 不变

- 如果分类错误的话（情况2）
  - $l = +1$　$y = -1$　$\Delta\mathbf{w} = \alpha\mathbf{x}$
  - $l = -1$　$y = +1$　$\Delta\mathbf{w} = -\alpha\mathbf{x}$

给出一种设计：

$$r = l - y = \begin{cases} +2 \\ -2 \\ 0 \end{cases}$$

真实类别 - 预测类别

真实类别-预测类别

$+2$	1 - (-1)	把正的错分成负的，假负例
-2	(-1) - 1	把负的错分成正的，假正例
0 No error	1 - 1 (-1) - (-1)	预测正确

感知机的学习机制

- 如果分类正确的话（情况1） $l = y = 1$ $\mathbf{w}$ 不变
- 如果分类错误的话（情况2）

$l = +1 \quad y = -1 \quad \Delta\mathbf{w} = \alpha\mathbf{x}$

$l = -1 \quad y = +1 \quad \Delta\mathbf{w} = -\alpha\mathbf{x}$

给出一种设计：

真实类别-预测类别

$$r = l - y = \begin{cases} +2 \\ -2 \\ 0 \end{cases}$$

真实类别 - 预测类别

$+2$	⊕ → ⊖	1 - (-1)
-2	⊖ → ⊕	(-1) - 1
0	No error	1 - 1 (-1) - (-1)

$$\Delta\mathbf{w} = \eta r \mathbf{x}$$

学习率 $\eta > 0$

误差

输入

把正的错分成负的，假负例

把负的错分成正的，假正例

预测正确

感知机 - 求解

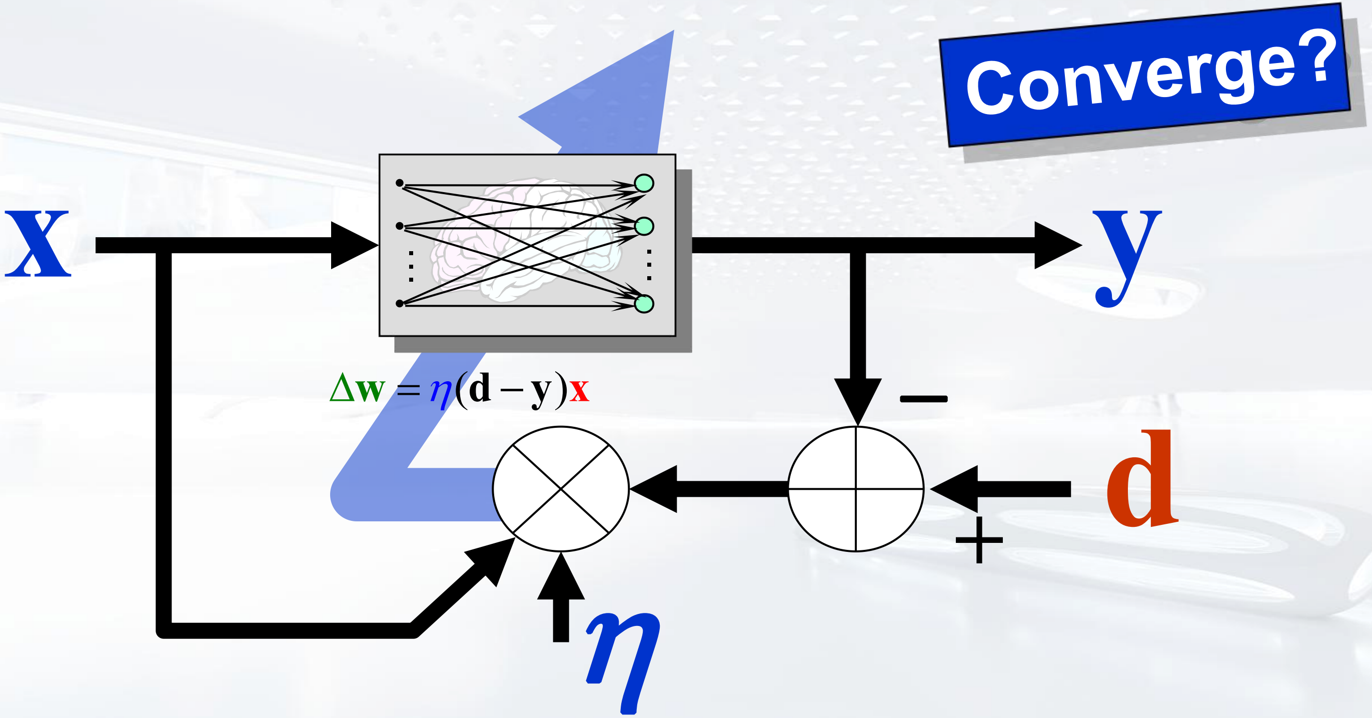
多点 情况如何更新**w**?

$$\Delta w_i(t) = \eta r_i x_i(t)$$

$$r_i = d_i - y_i = \begin{cases} 0 & d_i = y_i \\ +2 & d_i = 1, y_i = -1 \\ -2 & d_i = -1, y_i = 1 \end{cases}$$

$$\Delta w_i(t) = \eta (d_i - y_i) x_i(t)$$

Learning Rule

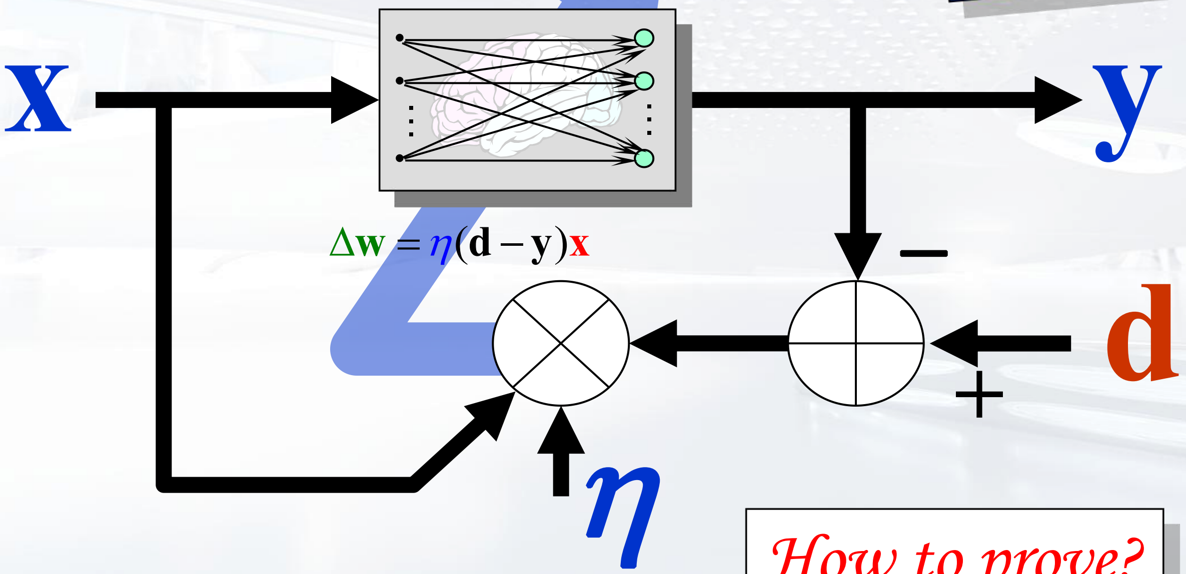


Learning Rule

MIMA

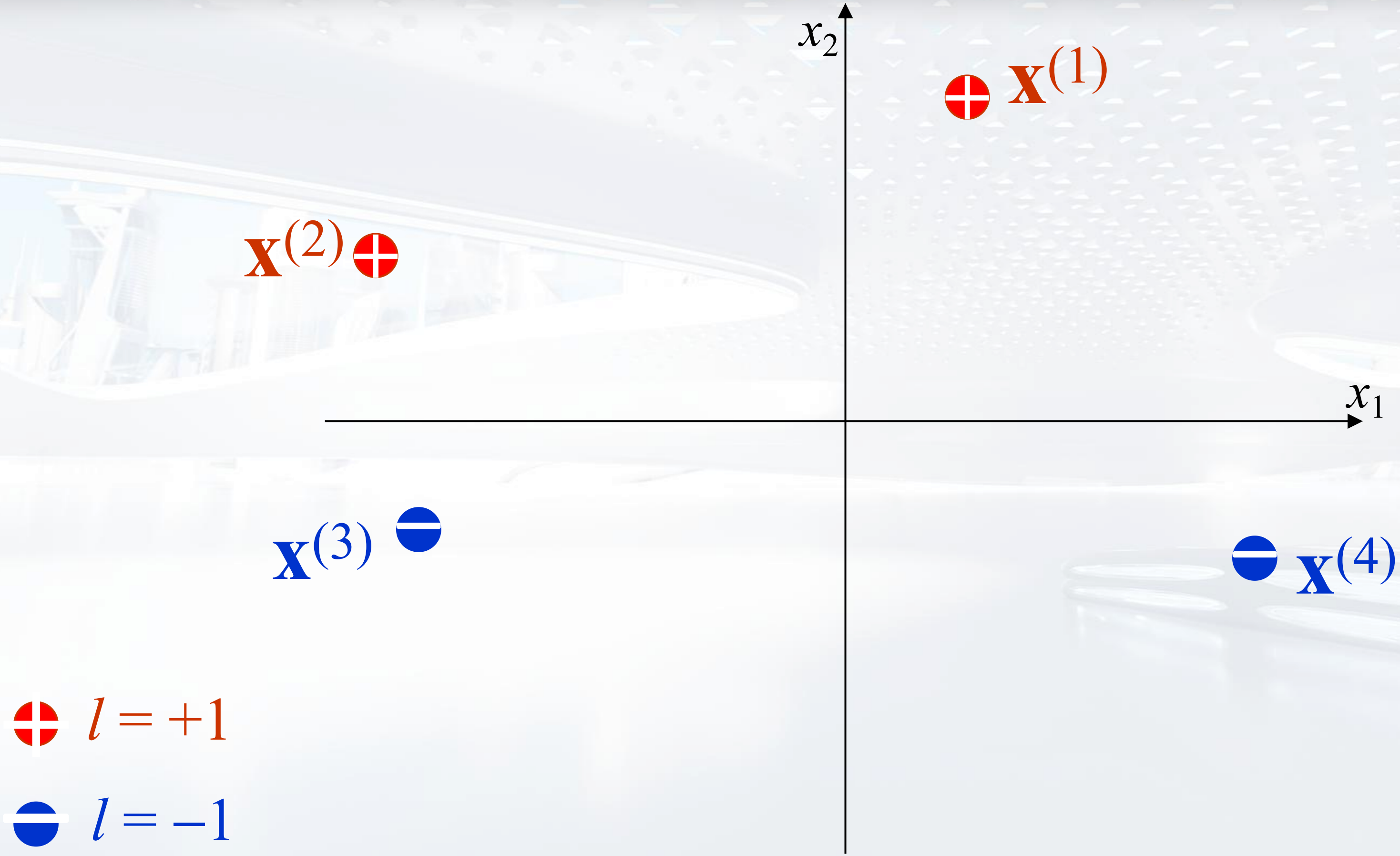
If the given training set is linearly separable, the learning process will converge in a finite number of steps.

Converge?

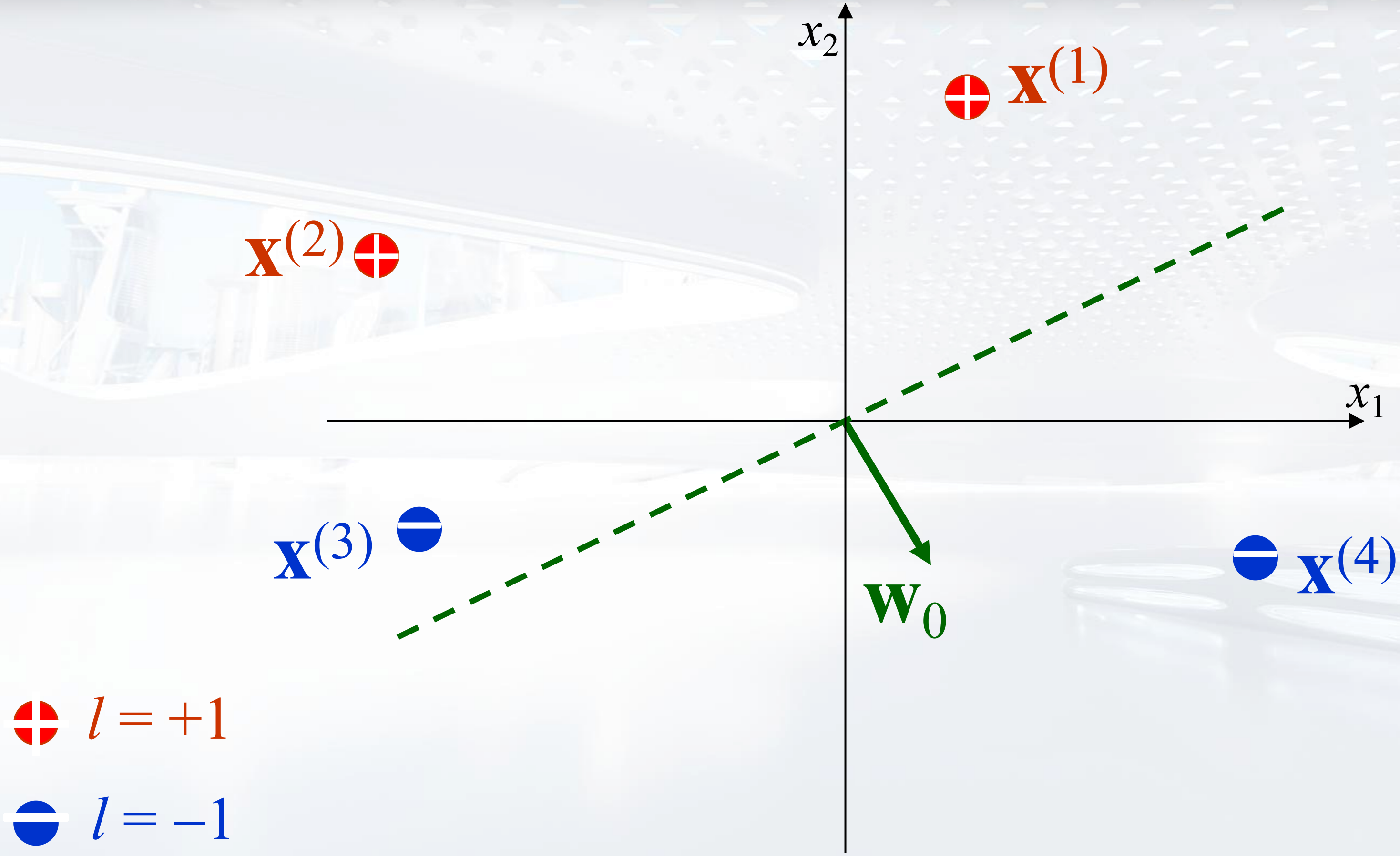


How to prove?

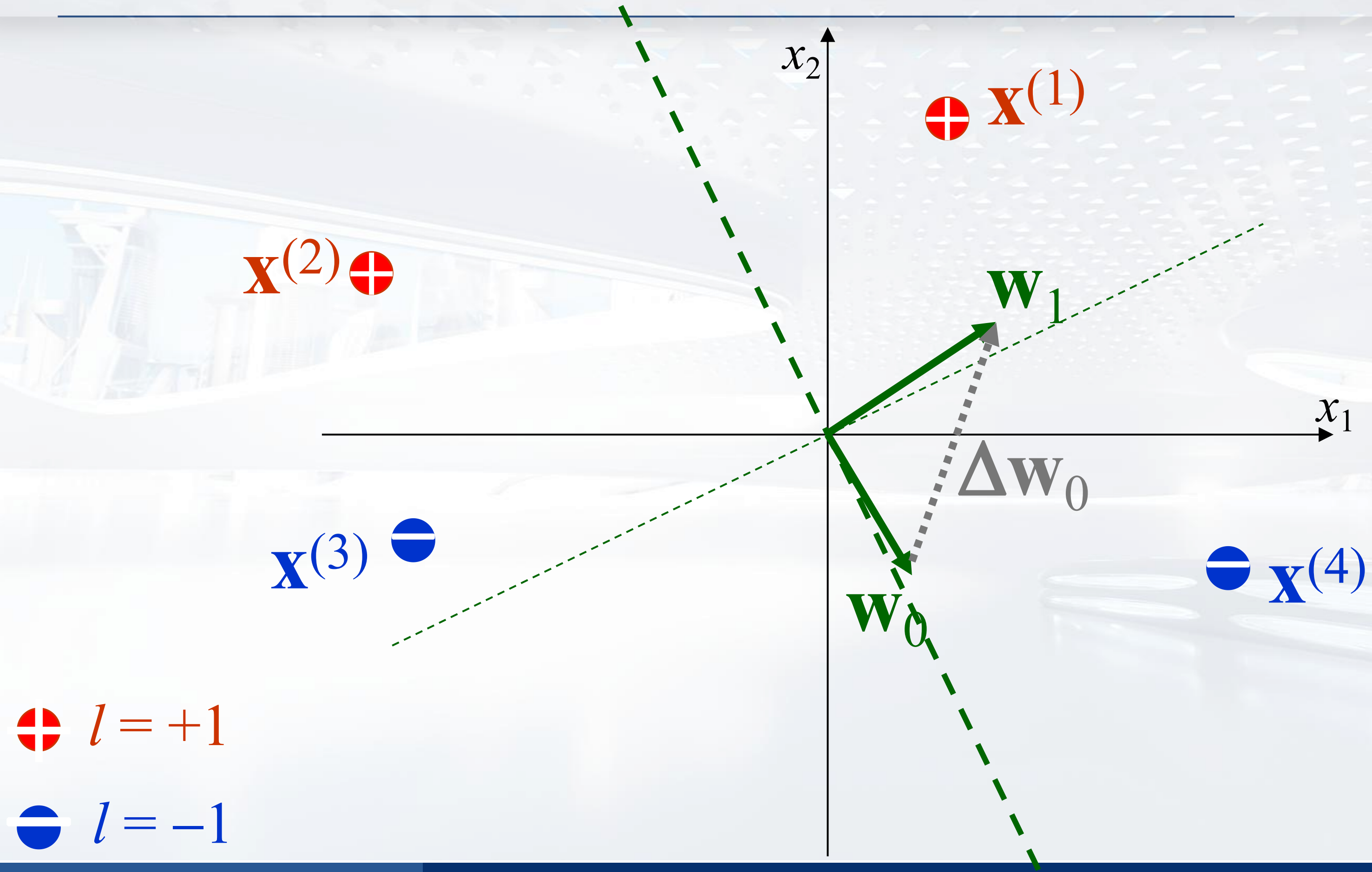
感知机的学习机制



感知机的学习机制

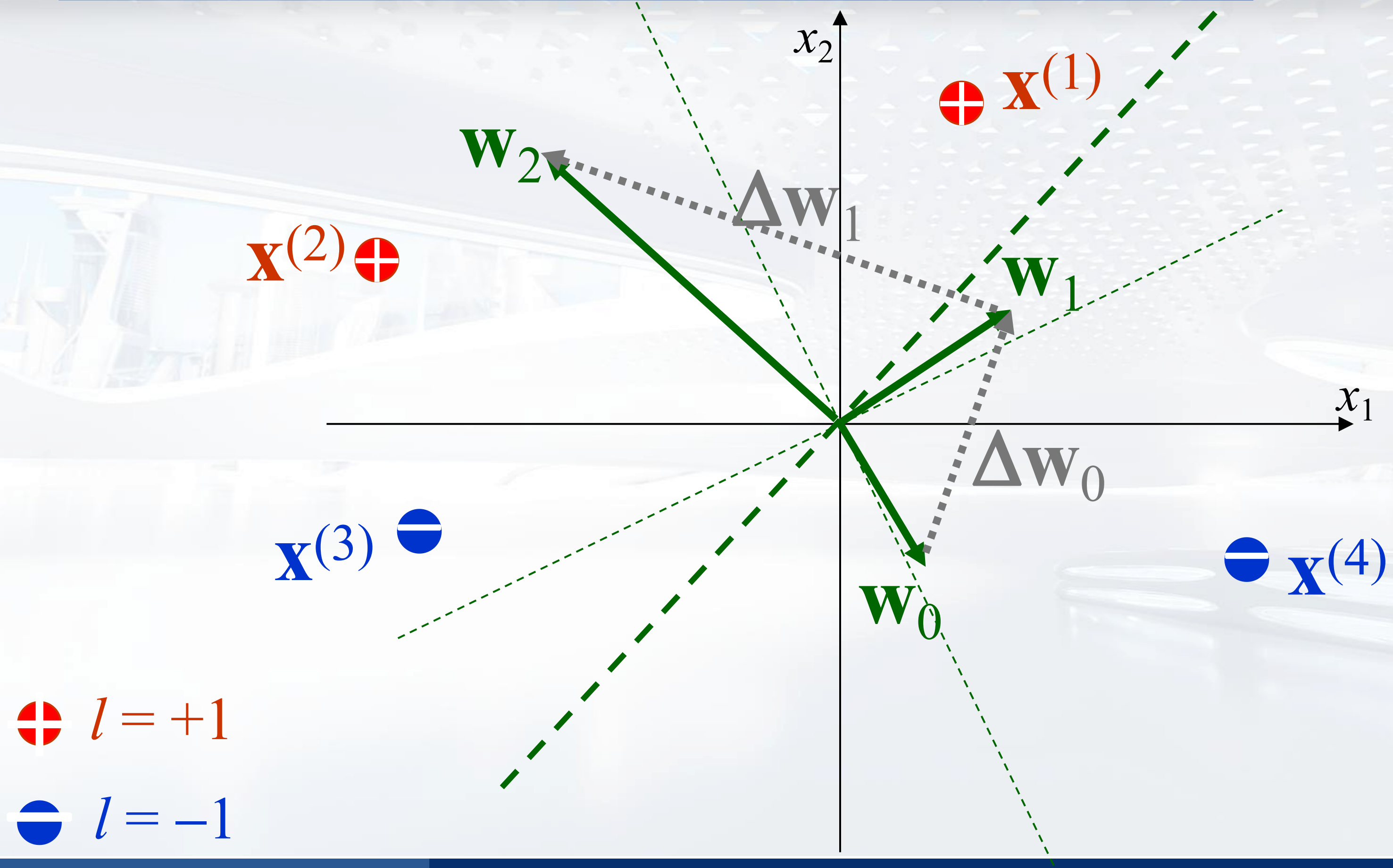


感知机的学习机制



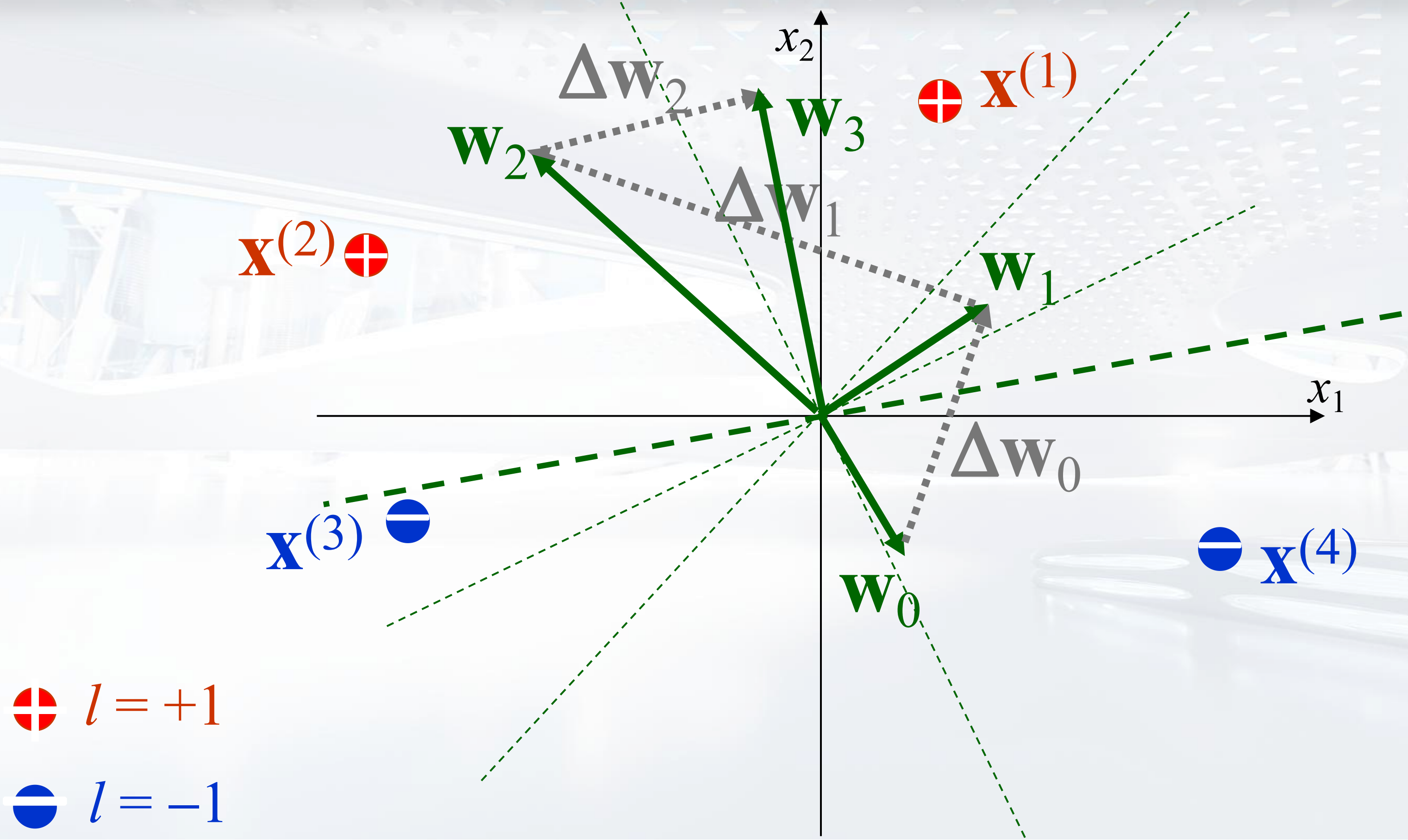
x_2
$\mathbf{x}^{(1)}$
$\mathbf{x}^{(2)}$
$\mathbf{w}_1$
x_1
$\Delta\mathbf{w}_0$
$\mathbf{x}^{(3)}$
$\mathbf{x}^{(4)}$
$\mathbf{w}_0$
$l = +1$
$l = -1$

感知机的学习机制

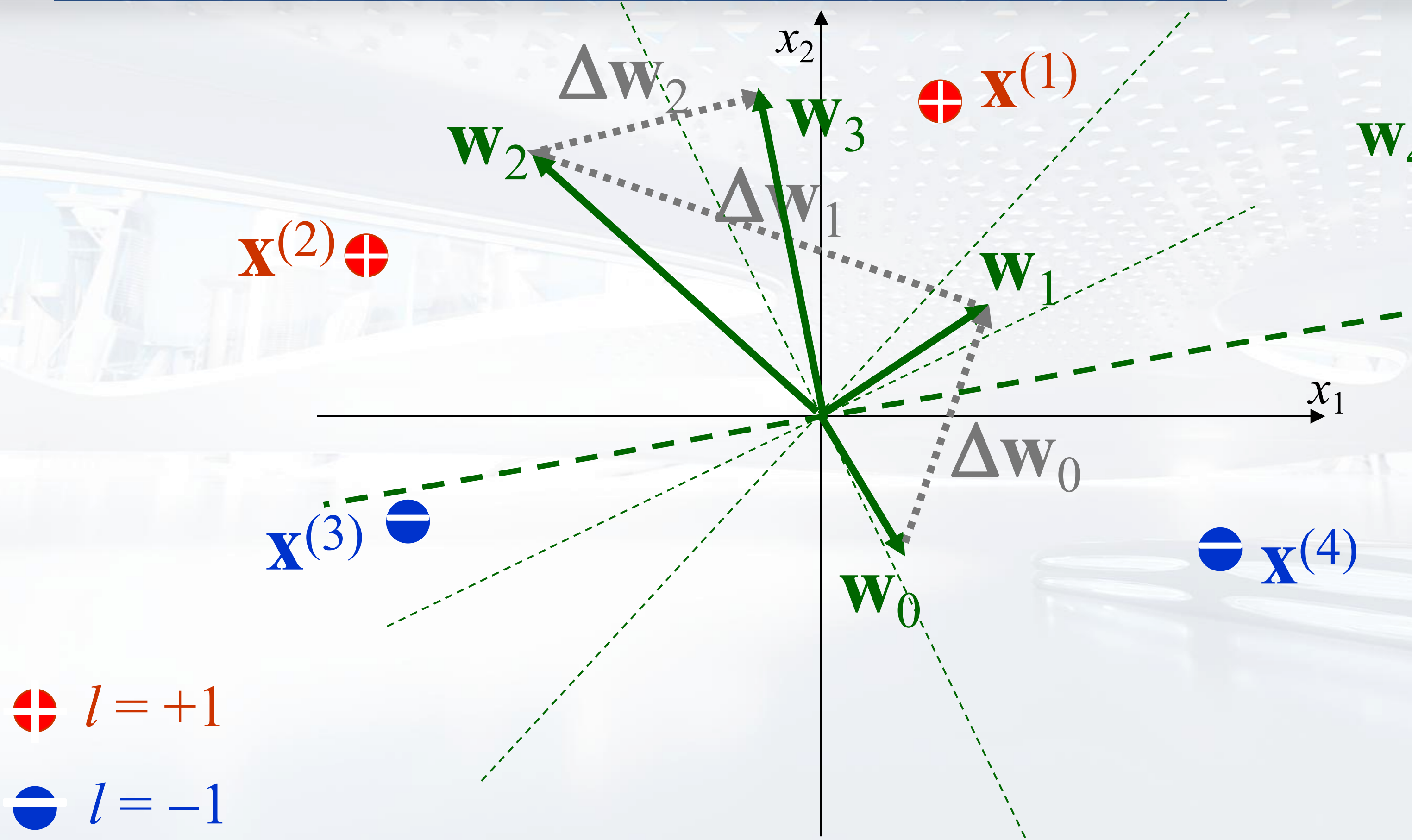


x_2
x_1
$\mathbf{x}^{(1)}$
$\mathbf{x}^{(2)}$
$\mathbf{x}^{(3)}$
$\mathbf{x}^{(4)}$
$\mathbf{w}_2$
$\Delta\mathbf{w}_1$
$\mathbf{w}_1$
$\Delta\mathbf{w}_0$
$\mathbf{w}_0$
$l = +1$
$l = -1$

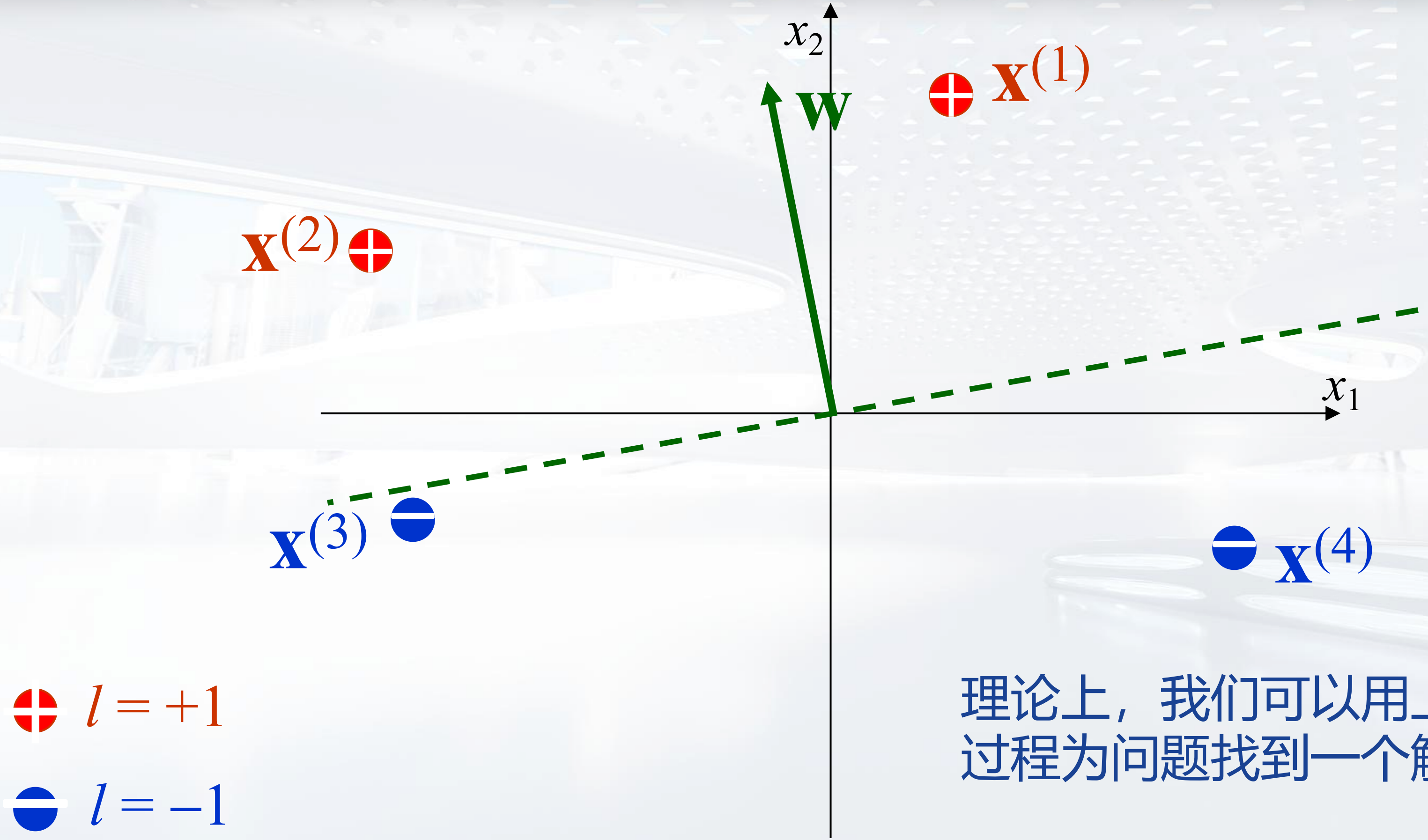
感知机的学习机制


x_2
x_1
$\Delta\mathbf{w}_2$
$\mathbf{w}_3$
$\mathbf{x}^{(1)}$
$\mathbf{w}_2$
$\Delta\mathbf{w}_1$
$\mathbf{x}^{(2)}$
$\mathbf{w}_1$
$\Delta\mathbf{w}_0$
$\mathbf{x}^{(3)}$
$\mathbf{x}^{(4)}$
$\mathbf{w}_0$
$l = +1$
$l = -1$

感知机的学习机制


x_2
x_1
$\Delta \mathbf{w}_2$
$\mathbf{w}_3$
$\mathbf{x}^{(1)}$
$\mathbf{w}_4 = \mathbf{w}_3$
$\mathbf{w}_2$
$\Delta \mathbf{w}_1$
$\mathbf{x}^{(2)}$
$\mathbf{w}_1$
$\Delta \mathbf{w}_0$
$\mathbf{x}^{(3)}$
$\mathbf{x}^{(4)}$
$\mathbf{w}_0$
$l = +1$
$l = -1$

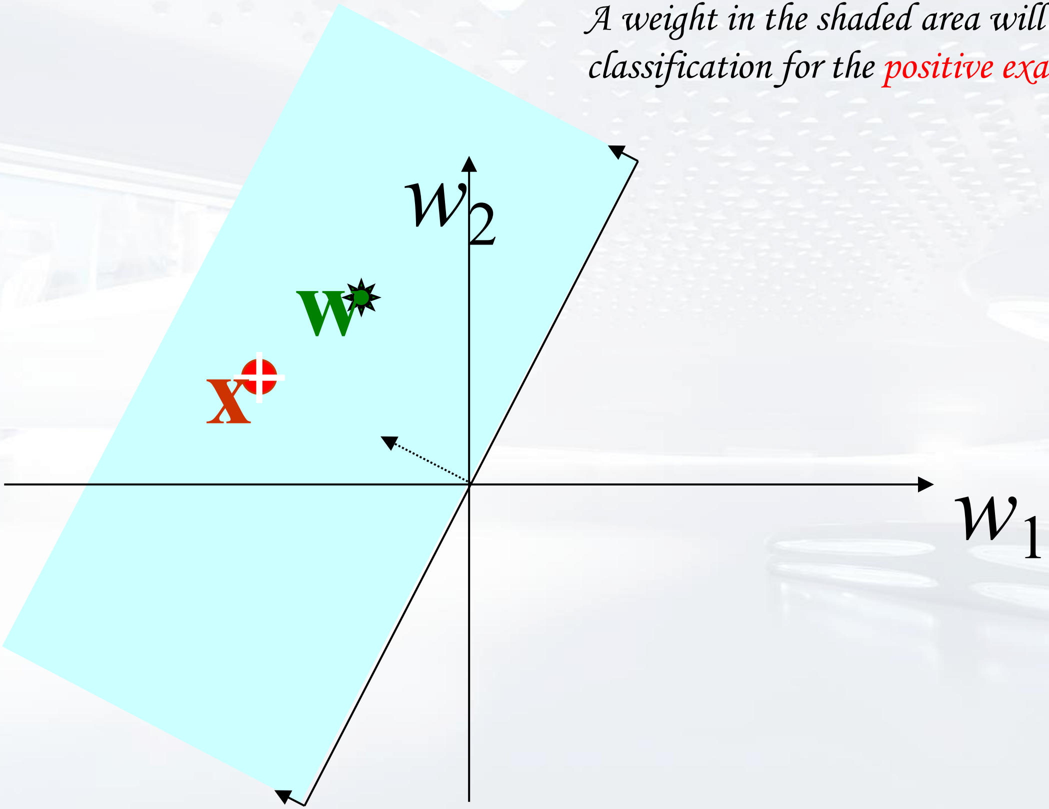
感知机的学习机制



理论上，我们可以用上述过程为问题找到一个解。

Weight Space

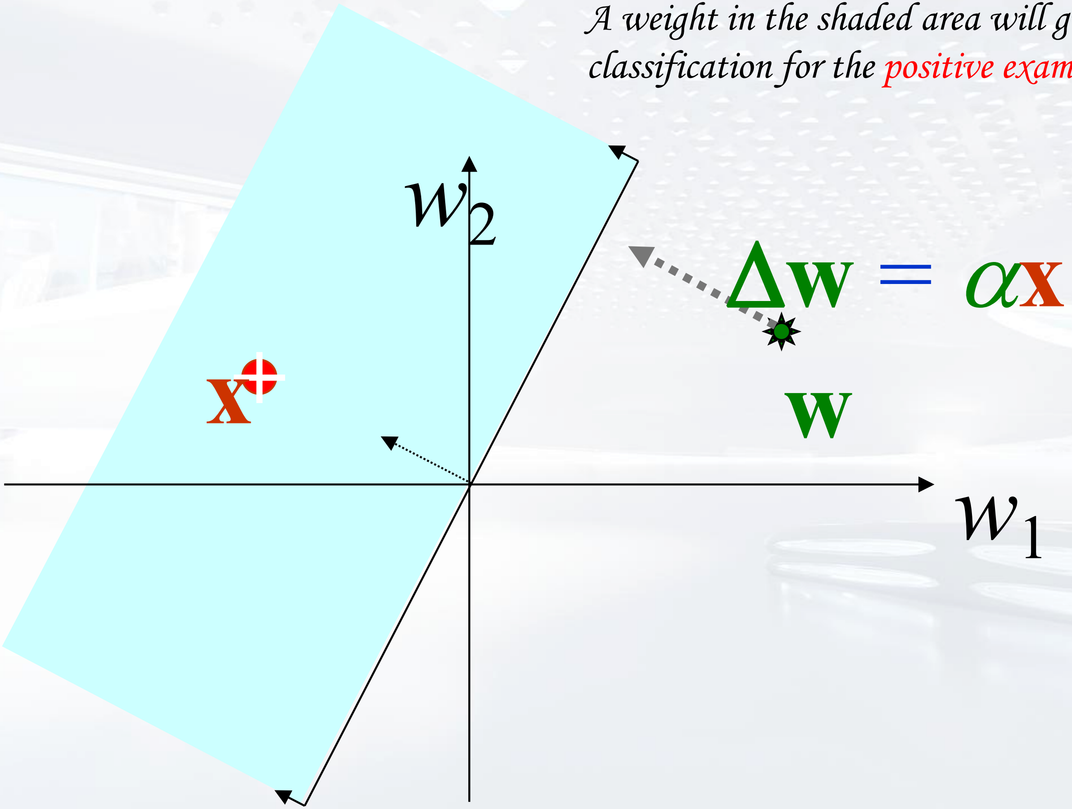
A weight in the shaded area will give correct classification for the positive example.



Weight Space

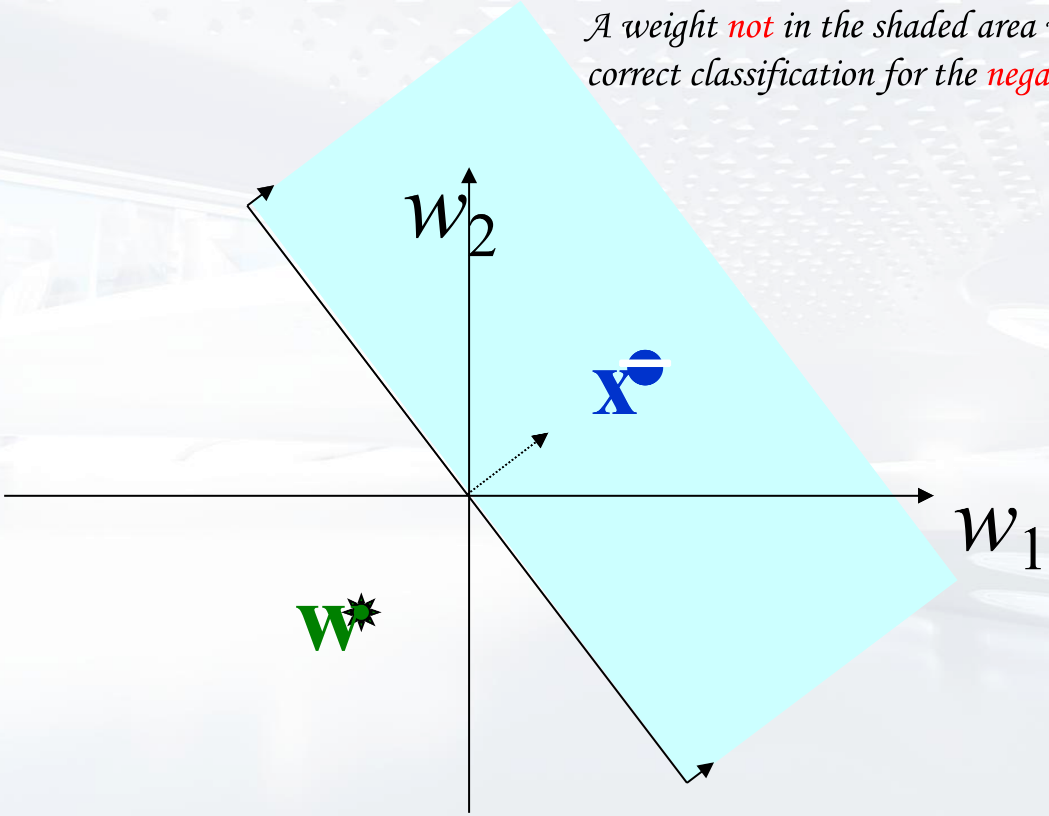
MIMA

A weight in the shaded area will give correct classification for the positive example.



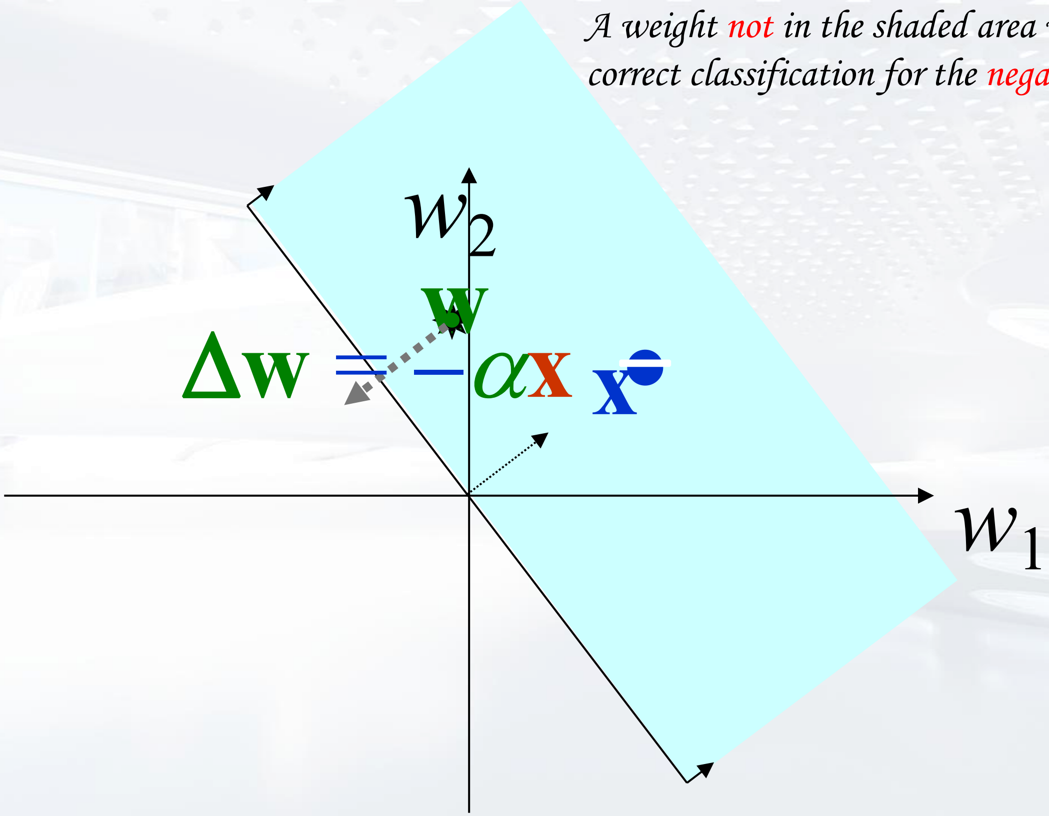
Weight Space

*A weight **not** in the shaded area will give correct classification for the **negative example.***

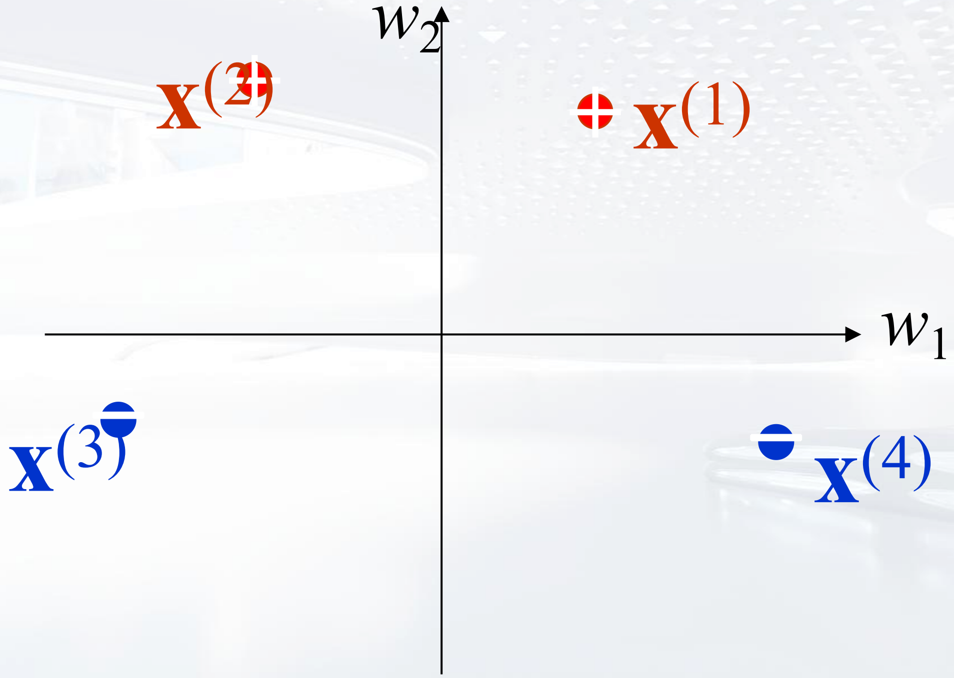


Weight Space

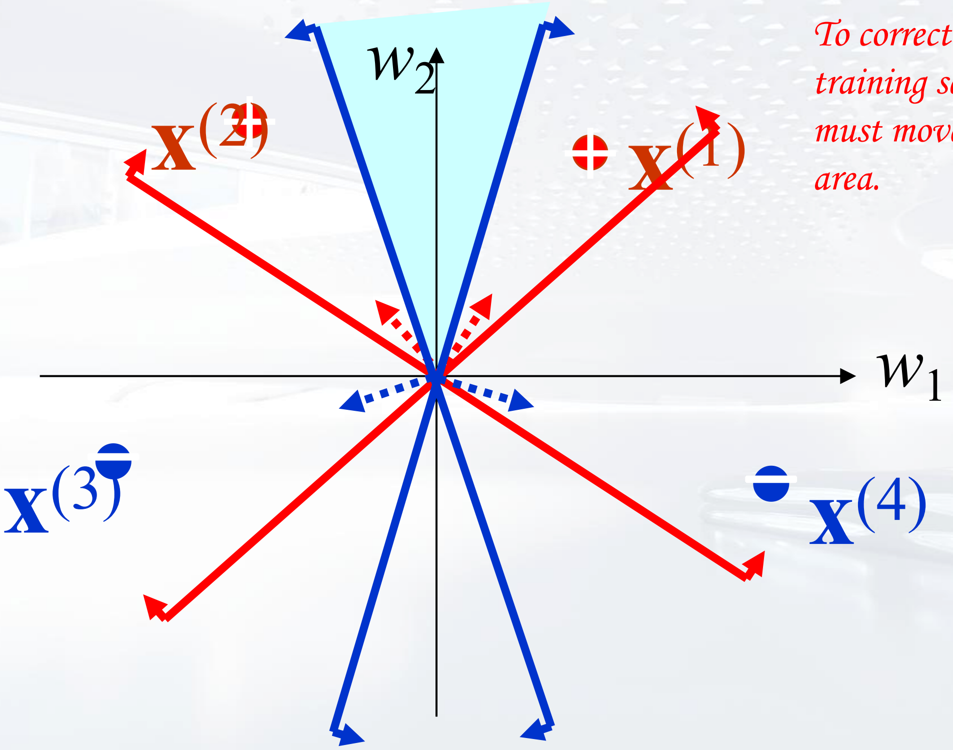
*A weight **not** in the shaded area will give correct classification for the **negative example.***



Learning Scenario in Weight Space

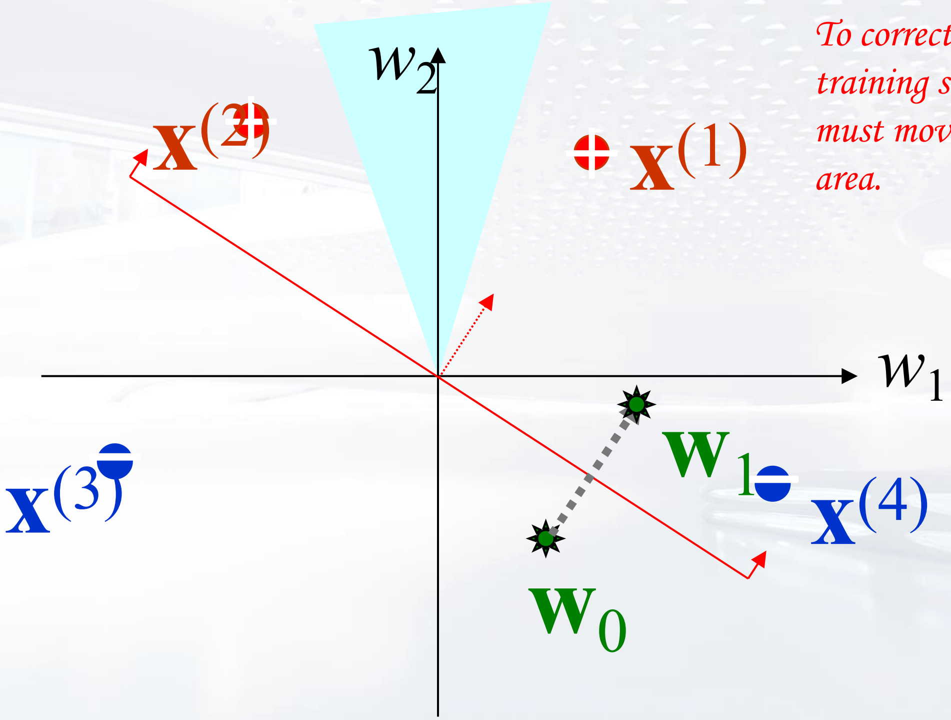


Learning Scenario in Weight Space



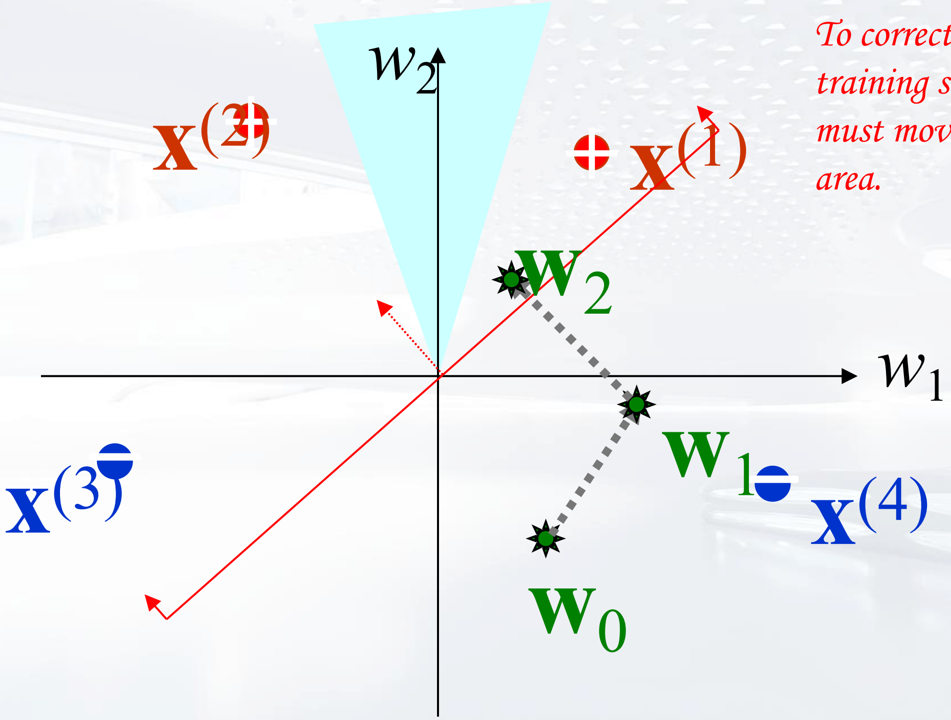
To correctly classify the training set, the weight must move into the shaded area.

Learning Scenario in Weight Space



To correctly classify the training set, the weight must move into the shaded area.

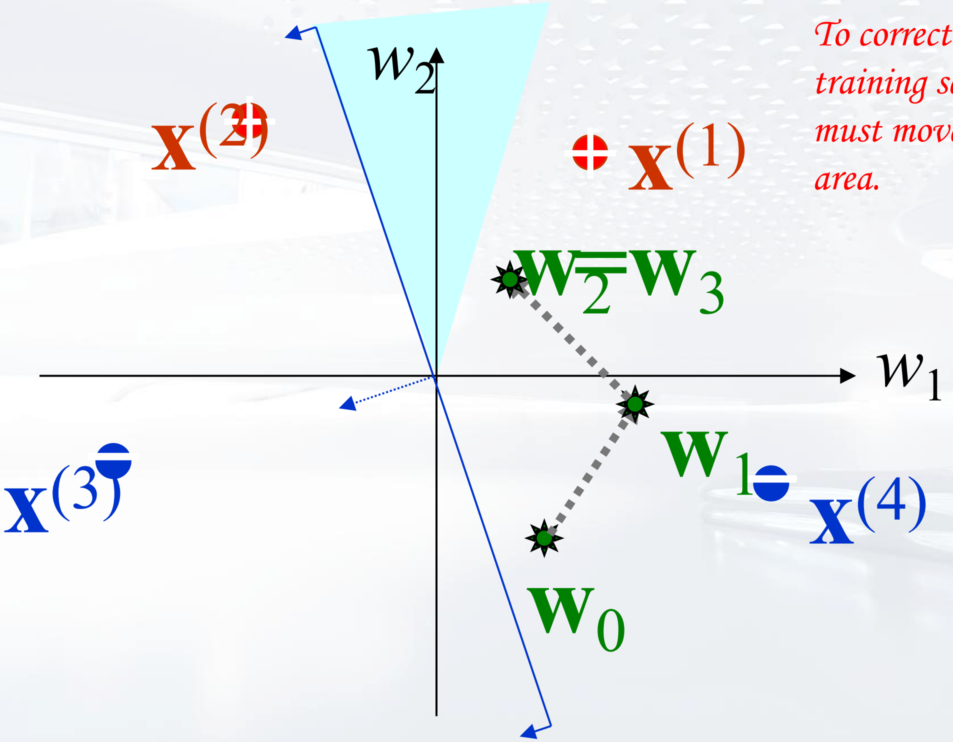
Learning Scenario in Weight Space



To correctly classify the training set, the weight must move into the shaded area.

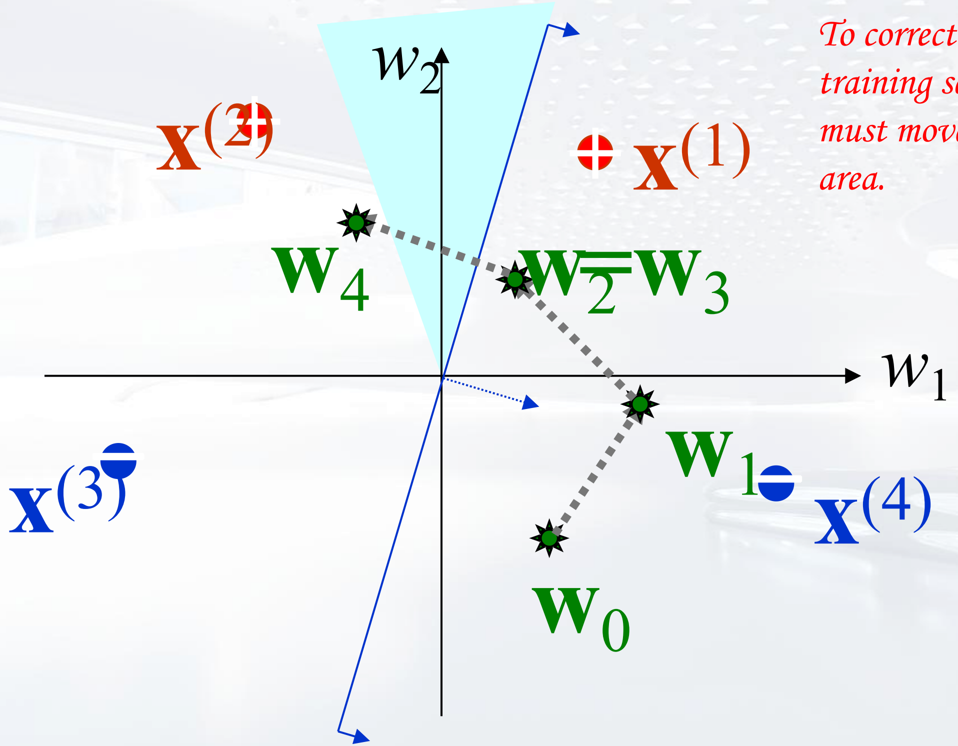
Learning Scenario in Weight Space

MIMA



To correctly classify the training set, the weight must move into the shaded area.

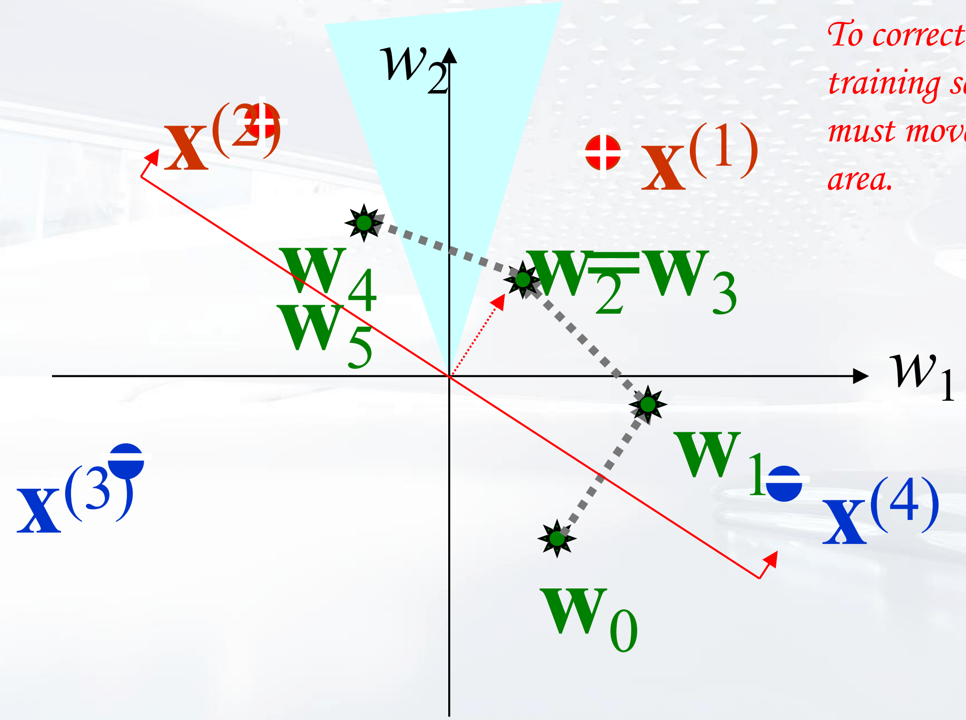
Learning Scenario in Weight Space



To correctly classify the training set, the weight must move into the shaded area.

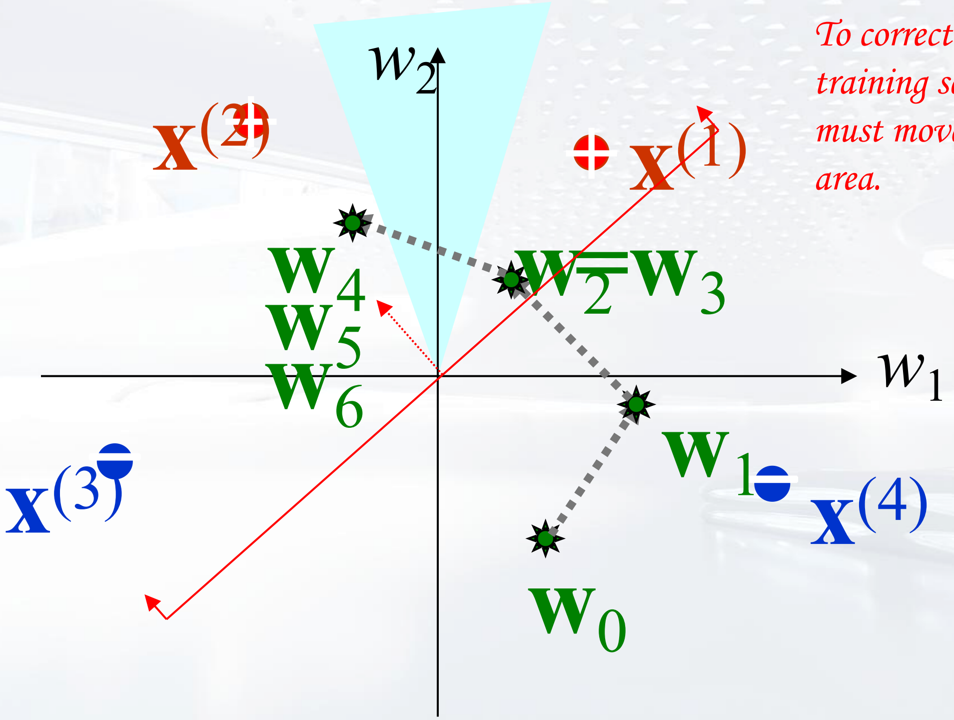
Learning Scenario in Weight Space

MIMA



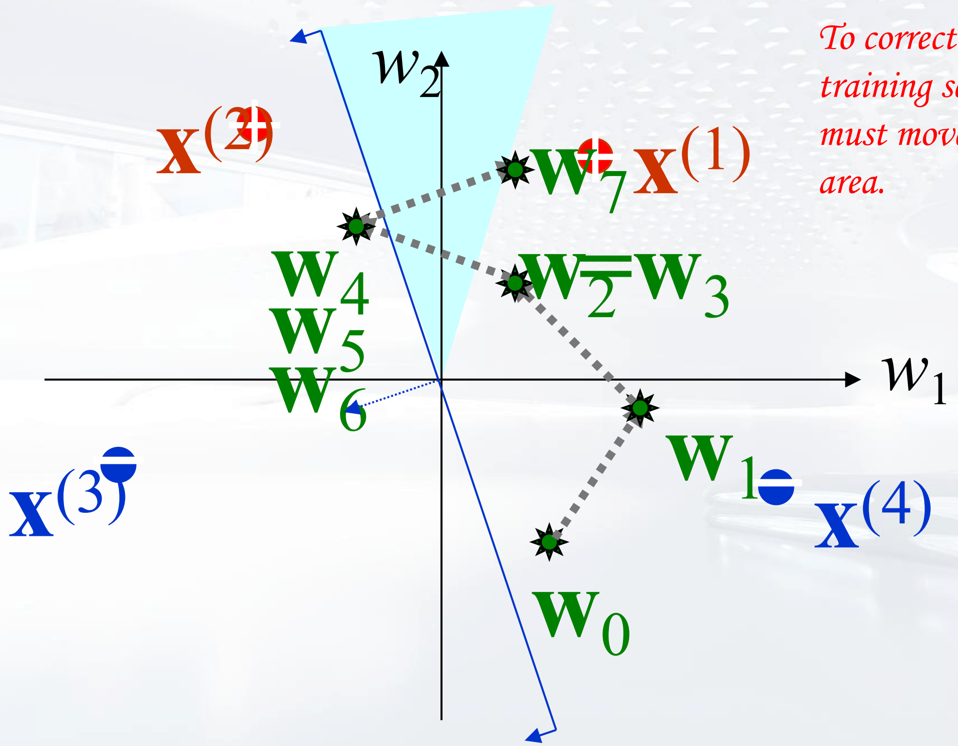
To correctly classify the training set, the weight must move into the shaded area.

Learning Scenario in Weight Space



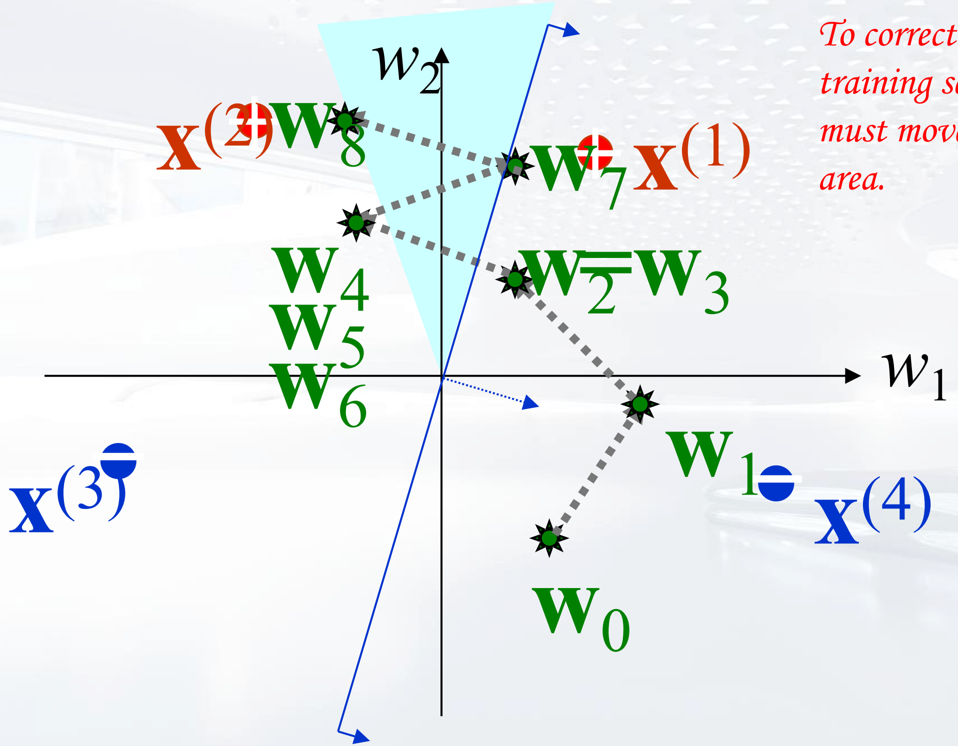
To correctly classify the training set, the weight must move into the shaded area.

Learning Scenario in Weight Space



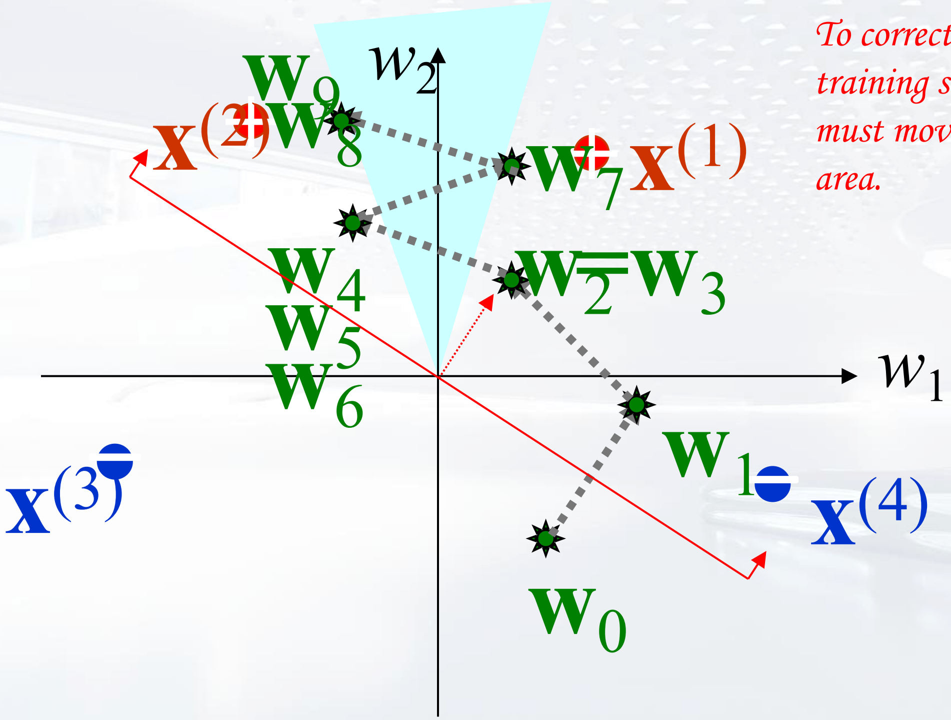
To correctly classify the training set, the weight must move into the shaded area.

Learning Scenario in Weight Space



To correctly classify the training set, the weight must move into the shaded area.

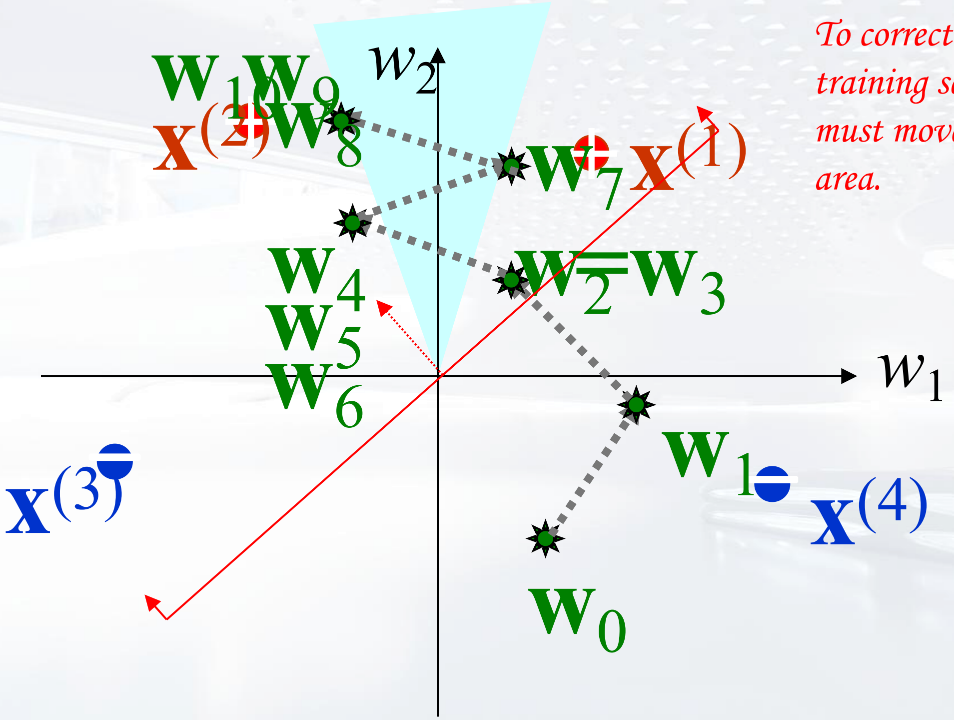
Learning Scenario in Weight Space



To correctly classify the training set, the weight must move into the shaded area.

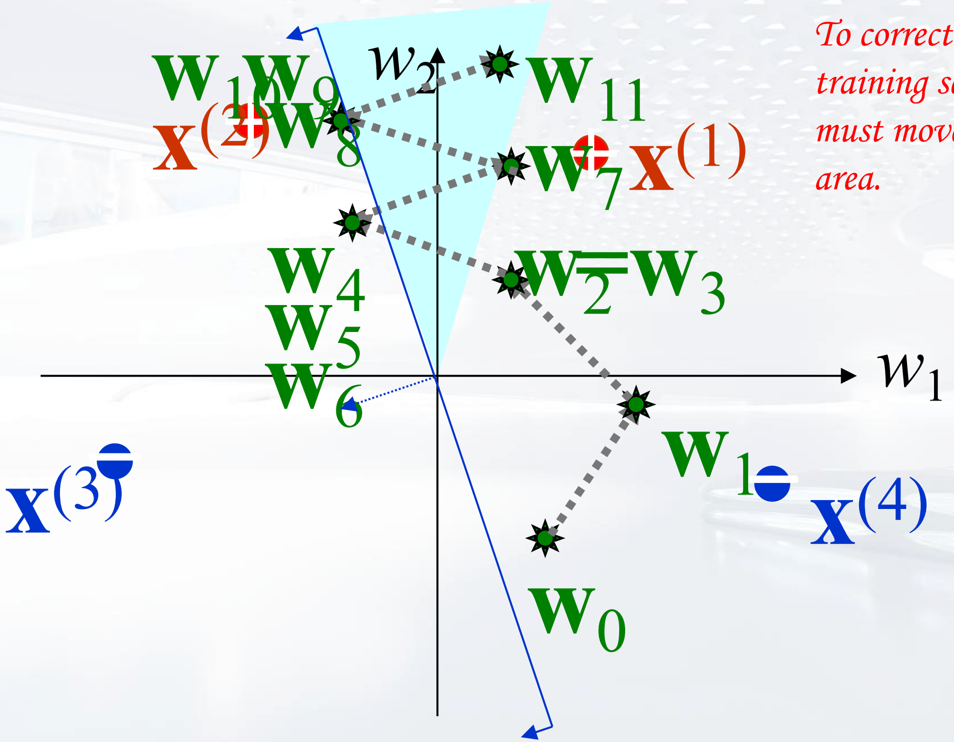
Learning Scenario in Weight Space

MIMA



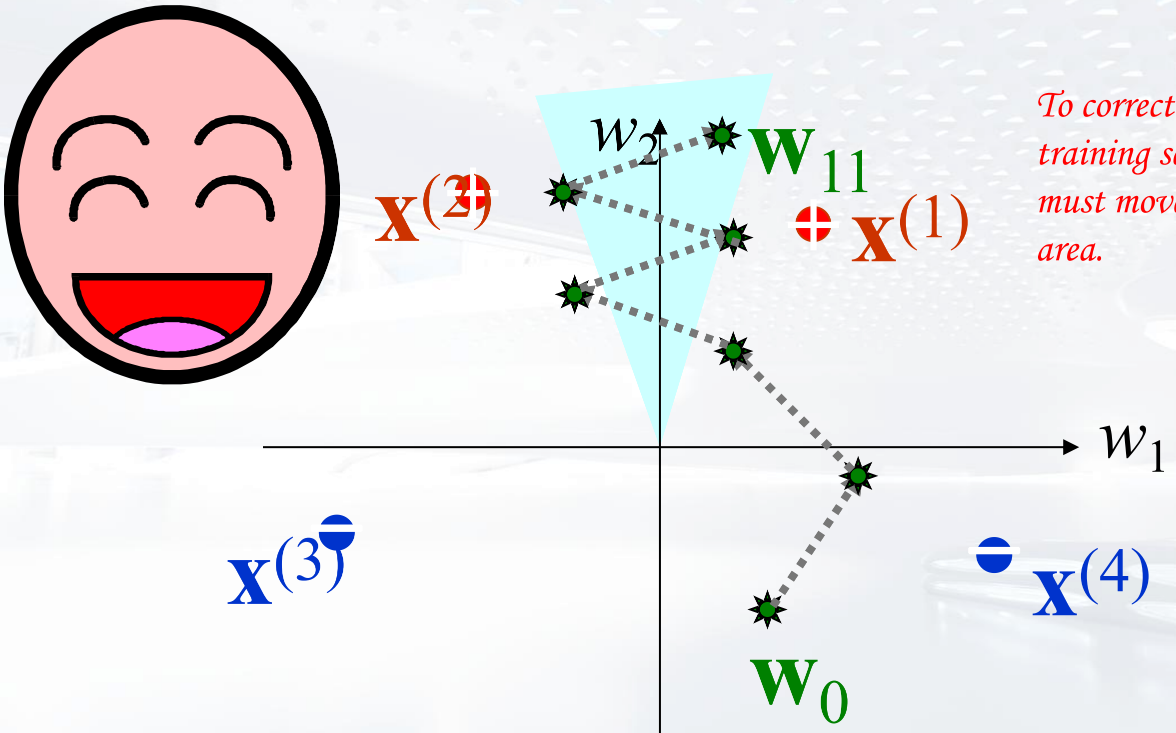
To correctly classify the training set, the weight must move into the shaded area.

Learning Scenario in Weight Space



To correctly classify the training set, the weight must move into the shaded area.

Learning Scenario in Weight Space



To correctly classify the training set, the weight must move into the shaded area.

Conceptually, in weight space, we move the weight into the feasible region.

Least Mean Square Learning

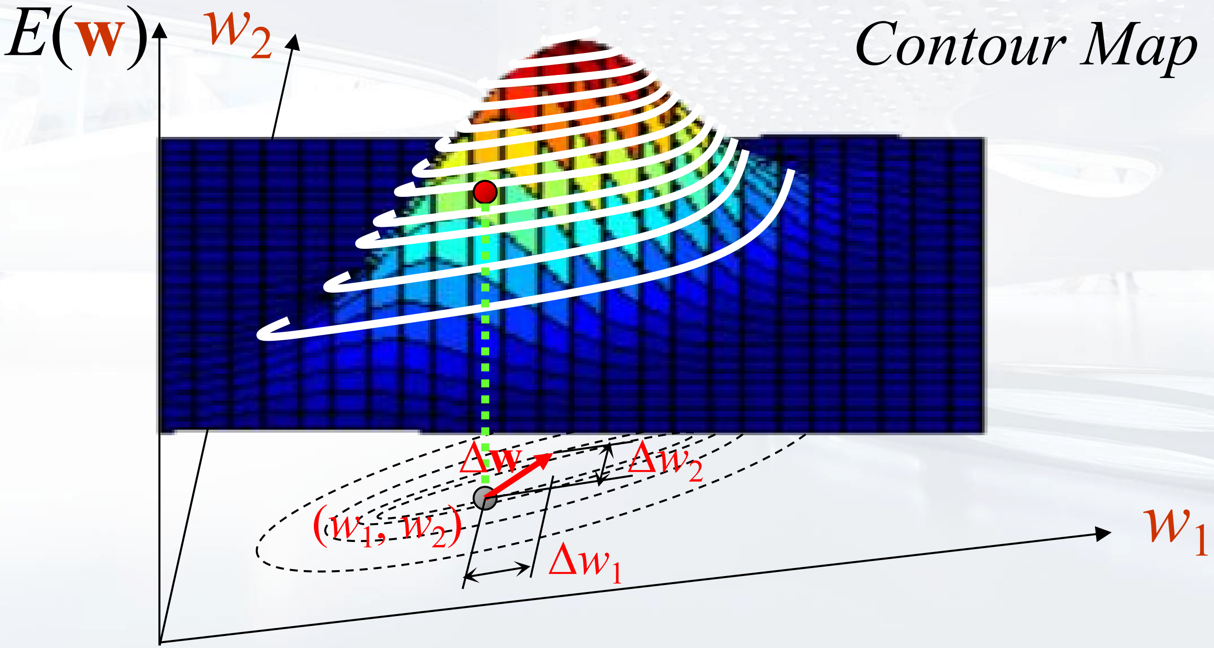
- Minimize the cost function (error function):

$$E(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{p}(d^{(k)} - y^{(k)})^2$$
$$= \frac{1}{2}\sum_{k=1}^{p}(d^{(k)} - \mathbf{w}^T\mathbf{x}^{(k)})^2$$
$$= \frac{1}{2}\sum_{k=1}^{p}\left(d^{(k)} - \sum_{l=1}^{m} w_l x_l^{(k)}\right)^2$$

d是真实的“正负”
y是预测的结果

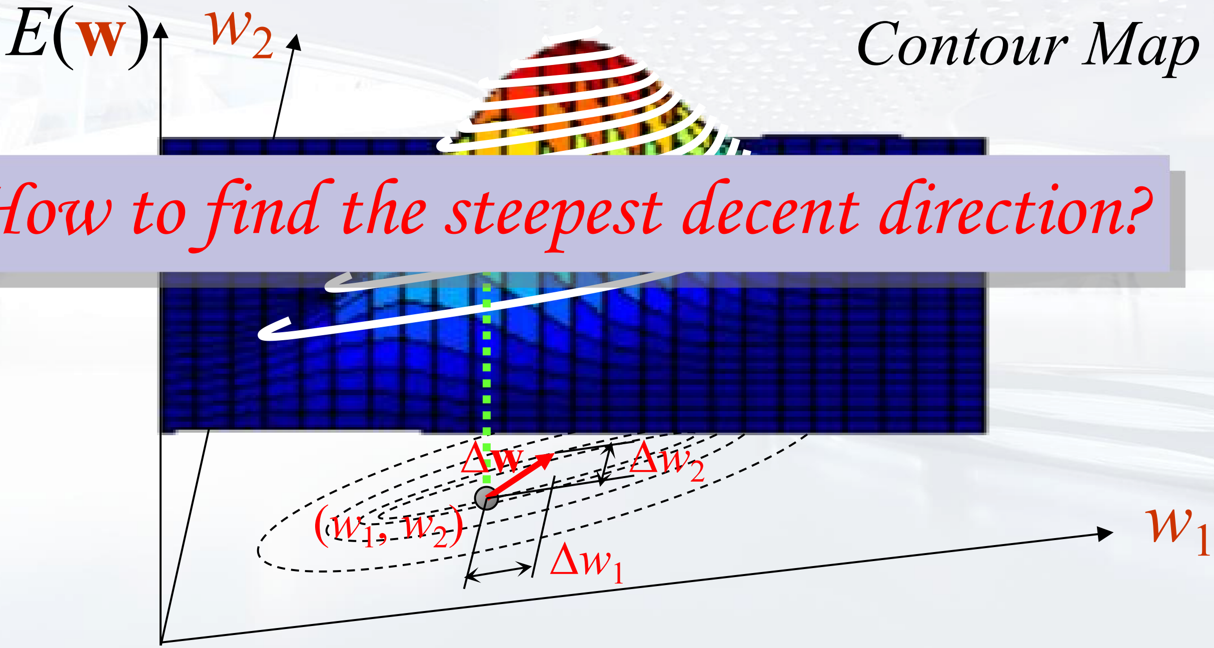
Least Mean Square Learning

- Our goal is to go *downhill*.



Least Mean Square Learning

- Our goal is to go *downhill*.



How to find the steepest decent direction?

Least Mean Square Learning

MIMA

- Gradient Operator

Let $f(\mathbf{w}) = f(w_1, w_2, \ldots, w_m)$ be a function over R^m.

$$df = \frac{\partial f}{\partial w_1} dw_1 + \frac{\partial f}{\partial w_2} dw_2 + \cdots + \frac{\partial f}{\partial w_m} dw_m$$

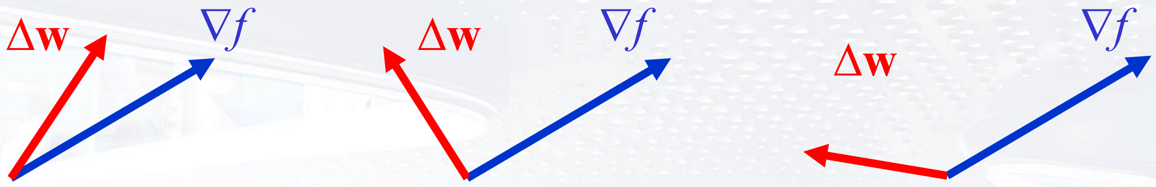
Define

$$\nabla f = \left(\frac{\partial f}{\partial w_1}, \frac{\partial f}{\partial w_2}, \cdots, \frac{\partial f}{\partial w_m} \right)^T$$

$$\Delta \mathbf{w} = \left(dw_1, dw_2, \cdots, dw_m \right)^T$$

$$\Rightarrow \quad df = \langle \nabla f, \Delta \mathbf{w} \rangle = \nabla f \bullet \Delta \mathbf{w}$$

Least Mean Square Learning



df : positive — Go uphill

df : zero — Plain

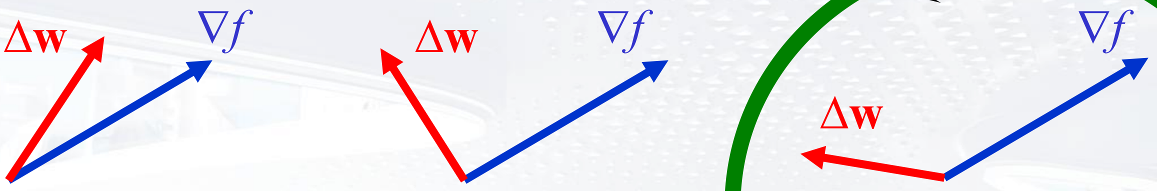
df : negative — Go downhill

$$df = \langle \nabla f, \Delta\mathbf{w} \rangle = \nabla f \bullet \Delta\mathbf{w}$$

Least Mean Square Learning

To minimize f, we choose

$$\Delta \mathbf{w} = -\eta \nabla f$$



df : positive	df : zero	df : negative
Go uphill	Plain	Go downhill

$$df = \langle \nabla f, \Delta \mathbf{w} \rangle = \nabla f \bullet \Delta \mathbf{w}$$

Least Mean Square Learning

- Minimize the cost function (error function):

$$E(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{p}\left(d^{(k)} - \sum_{l=1}^{m} w_l x_l^{(k)}\right)^2$$

$$\frac{\partial E(\mathbf{w})}{\partial w_j} = -\sum_{k=1}^{p}\left(d^{(k)} - \sum_{l=1}^{m} w_l x_l^{(k)}\right)x_j^{(k)}$$

$$= -\sum_{k=1}^{p}\left(d^{(k)} - \mathbf{w}^T\mathbf{x}^{(k)}\right)x_j^{(k)} = -\sum_{k=1}^{p}\overbrace{\left(d^{(k)} - y^{(k)}\right)}^{\delta^{(k)}}x_j^{(k)}$$

$$\boxed{\frac{\partial E(\mathbf{w})}{\partial w_j} = -\sum_{k=1}^{p}\delta^{(k)}x_j^{(k)} \qquad \delta^{(k)} = d^{(k)} - y^{(k)}}$$

Least Mean Square Learning

MIMA

- Minimize the cost function (error function):

$$E(\mathbf{w}) = \frac{1}{2}\sum_{k=1}^{p}\left(d^{(k)} - \sum_{l=1}^{m} w_l x_l^{(k)}\right)^2$$

$$\nabla_w E(\mathbf{w}) = \left(\frac{\partial E(\mathbf{w})}{\partial w_1}, \frac{\partial E(\mathbf{w})}{\partial w_2}, \ldots, \frac{\partial E(\mathbf{w})}{\partial w_m}\right)^T$$

$$\Delta \mathbf{w} = -\eta \nabla_{\mathbf{w}} E(\mathbf{w})$$ — Weight Modification Rule

$$\boxed{\frac{\partial E(\mathbf{w})}{\partial w_j} = -\sum_{k=1}^{p} \delta^{(k)} x_j^{(k)} \qquad \delta^{(k)} = d^{(k)} - y^{(k)}}$$

Least Mean Square Learning

- Learning Modes
  - Batch Learning Mode

$$\Delta w_j = \eta \sum_{k=1}^{p} \delta^{(k)} x_j^{(k)}$$

  - Incremental Learning Mode

$$\Delta w_j = \eta \delta^{(k)} x_j^{(k)}$$

$$\frac{\partial E(\mathbf{w})}{\partial w_j} = -\sum_{k=1}^{p} \delta^{(k)} x_j^{(k)} \qquad \delta^{(k)} = d^{(k)} - y^{(k)}$$

Perceptron

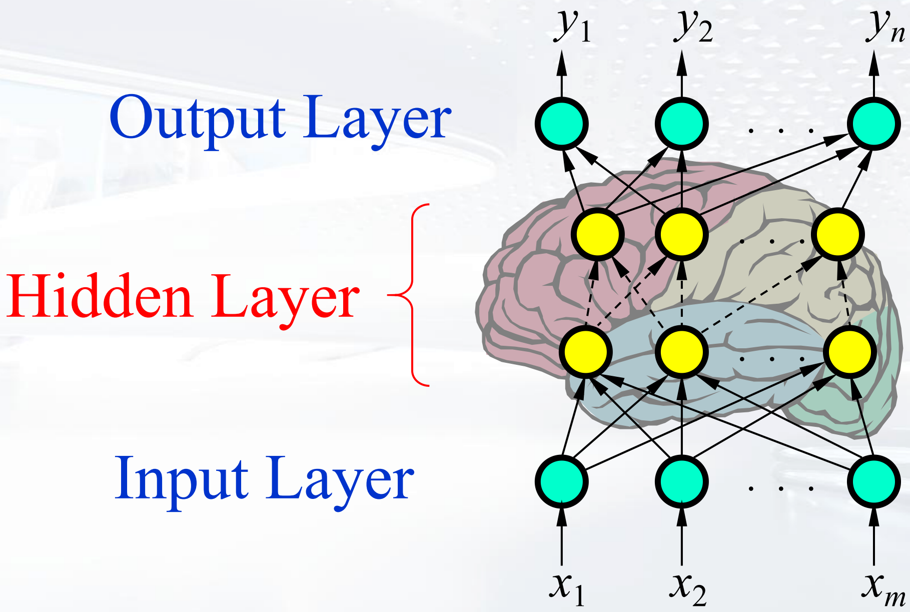
- Summary
  - Separability: some parameters get the training set perfectly correct.
  - Convergence: if the training is separable, perceptron will eventually converge (binary case)?

- The Perceptron convergence theorem

- The relation between perceptron and Bayes classifier

Contents

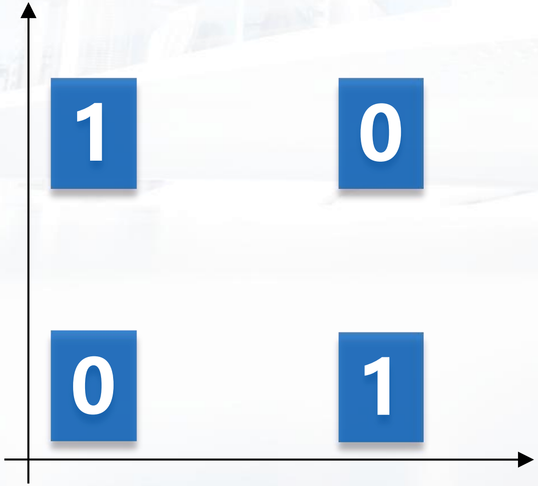
Multilayer Perceptron



How an MLP Works?

Example:

- Not linearly separable.
- Is a single layer perceptron workable?



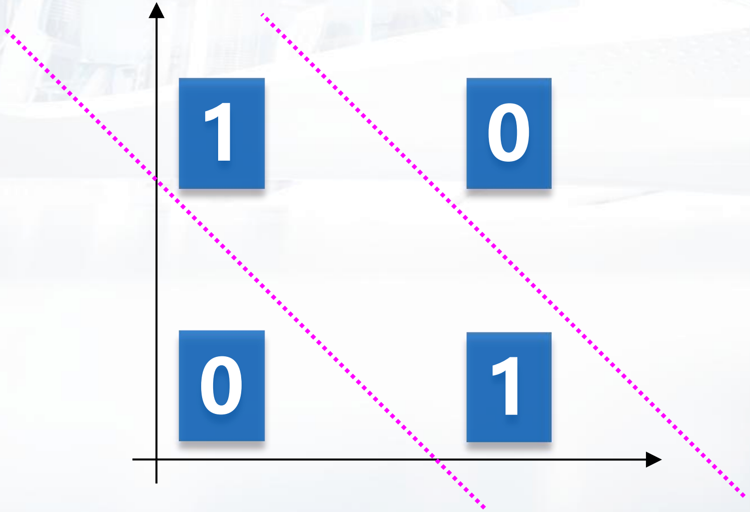
输入		异或 xor
0	0	0
0	1	1
1	0	1
1	1	0

How an MLP Works?

MIMA

Example:

- Not linearly separable.
- Is a single layer perceptron workable?



输入		异或 xor
0	0	0
0	1	1
1	0	1
1	1	0

How an MLP Works?

输入		或 or	与非 nand	与 and
0	0	0		
0	1	1		
1	0	1		
1	1	1		

输入		异或 xor
0	0	0
0	1	1
1	0	1
1	1	0

How an MLP Works?

输入		或 or	与非 nand	与 and
0	0		1	
0	1		1	
1	0		1	
1	1		0	

输入		异或 xor
0	0	0
0	1	1
1	0	1
1	1	0

How an MLP Works?

输入		或 or	与非 nand	与 and
		0	1	0
		1	1	1
		1	1	1
		1	0	0

输入		异或 xor
0	0	0
0	1	1
1	0	1
1	1	0

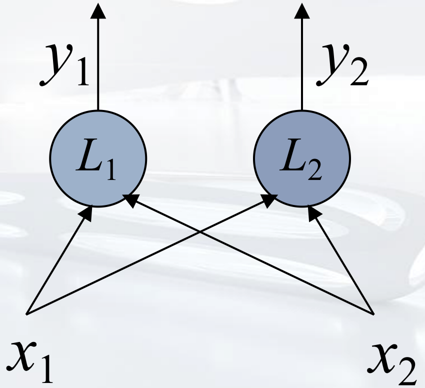
How an MLP Works?

输入		或 or	与非 nand	与 and
input		hidden		output
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0

输入		异或 xor
0	0	0
0	1	1
1	0	1
1	1	0

How an MLP Works?

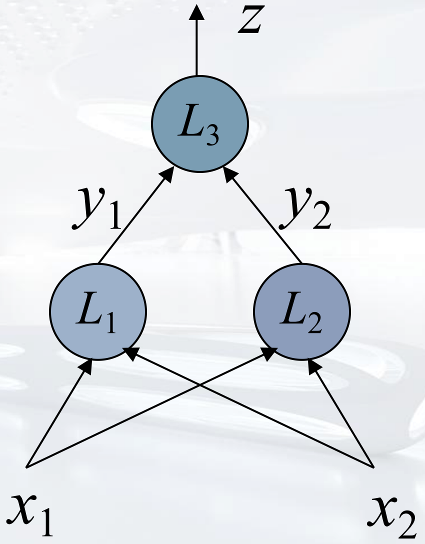
input		hidden		output
输入		或 or	与非 nand	与 and
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0



How an MLP Works?

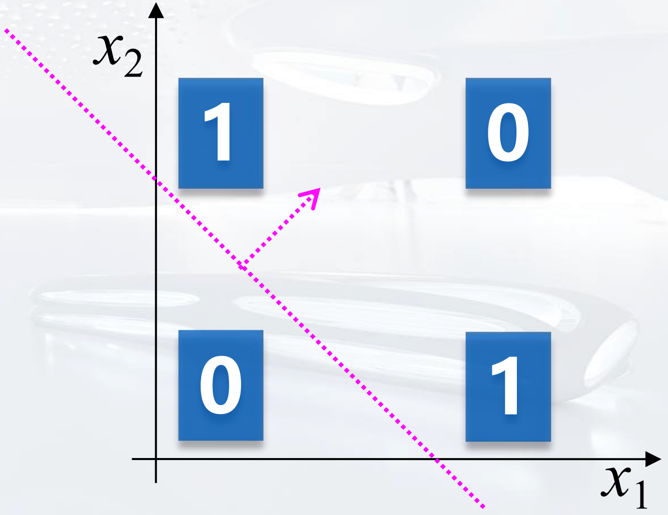
MIMA

input		hidden		output
输入		或 or	与非 nand	与 and
0	0	0	1	0
0	1	1	1	1
1	0	1	1	1
1	1	1	0	0



How an MLP Works?

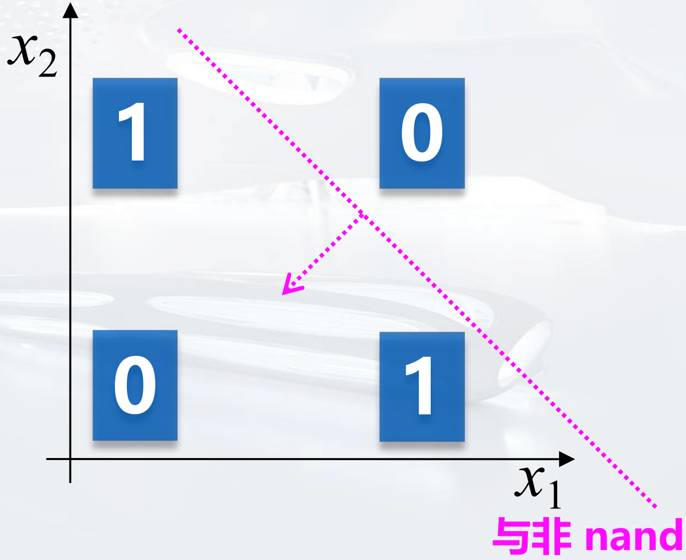
input		hidden		output
输入		或 or	与非 nand	与 and
0	0	0		
0	1	1		
1	0	1		
1	1	1		



How an MLP Works?

MIMA

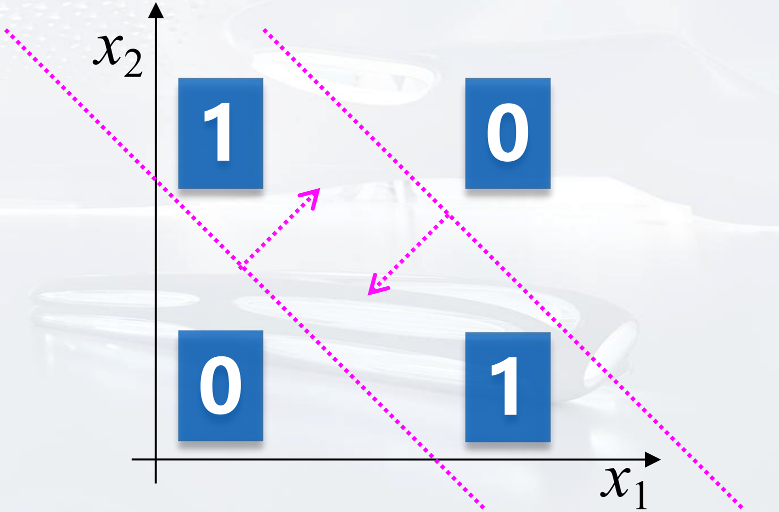
input		hidden		output
输入		或 or	与非 nand	与 and
0	0		1	
0	1		1	
1	0		1	
1	1		0	



How an MLP Works?

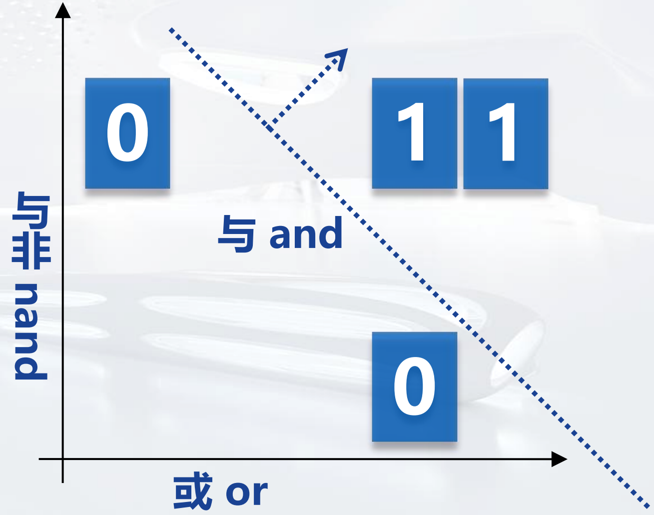
MIMA

input		hidden		output
输入		或 or	与非 nand	与 and
0	0	0	1	
0	1	1	1	
1	0	1	1	
1	1	1	0	



How an MLP Works?

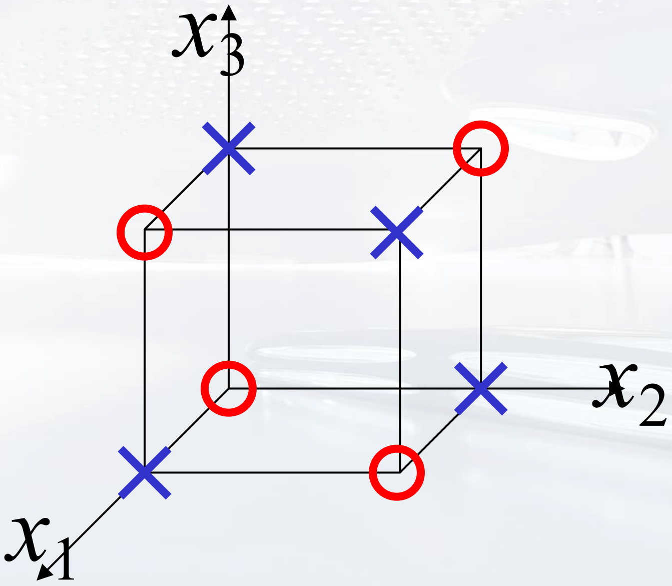
input		hidden		output
输入		或 or	与非 nand	与 and
		0	1	0
		1	1	1
		1	1	1
		1	0	0



Parity Problem

Is the problem linearly separable?

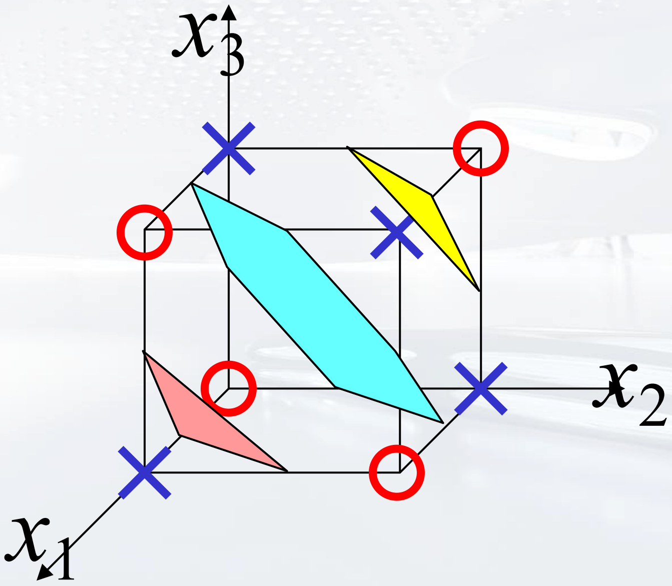
$x_1 x_2 x_3$	
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1



Parity Problem

Is the problem linearly separable?

$x_1 x_2 x_3$	
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1

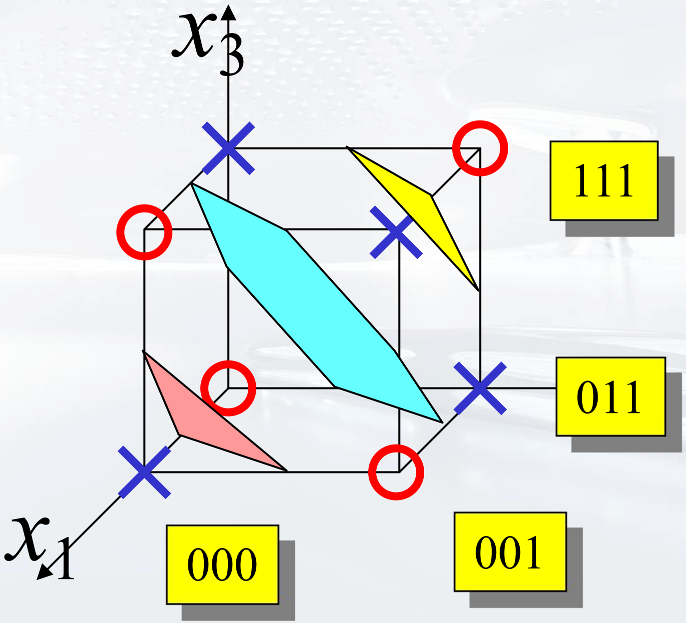


Parity Problem

MIMA

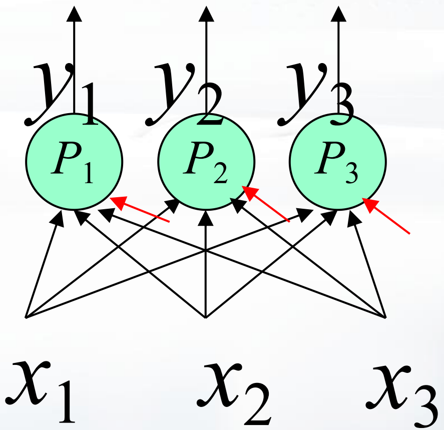
Is the problem linearly separable?

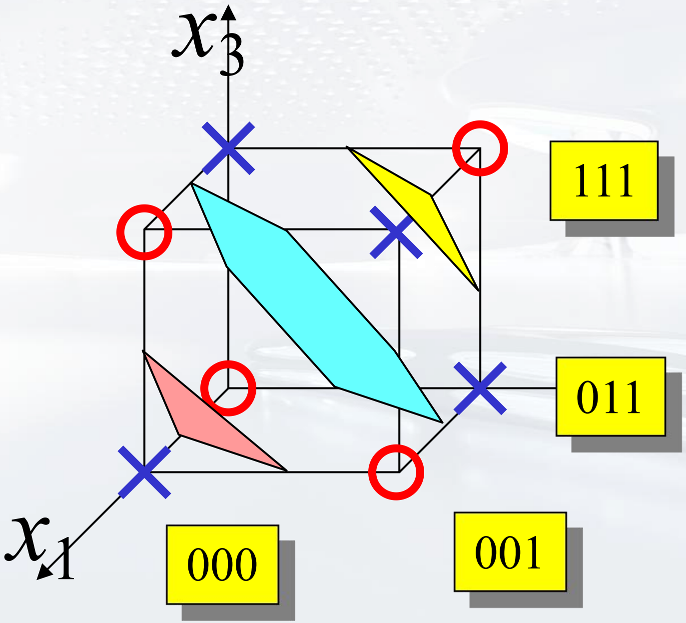
$x_1 x_2 x_3$	
000	0
001	1
010	1
011	0
100	1
101	0
110	0
111	1



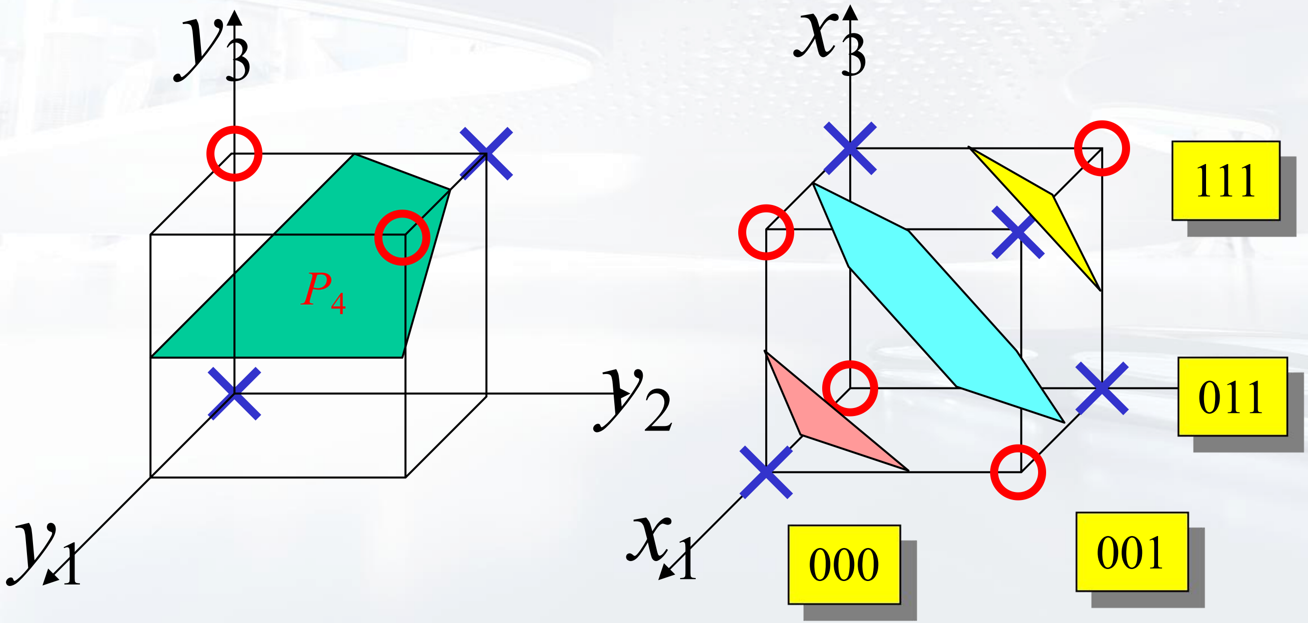
Parity Problem

Is the problem linearly separable?

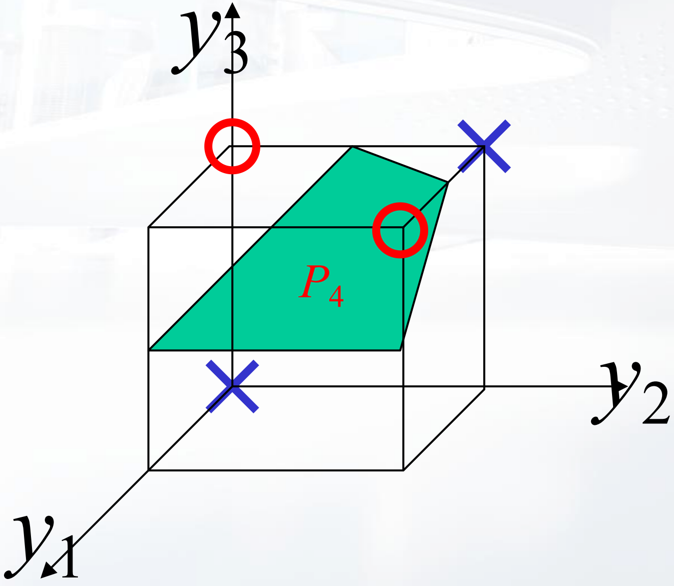


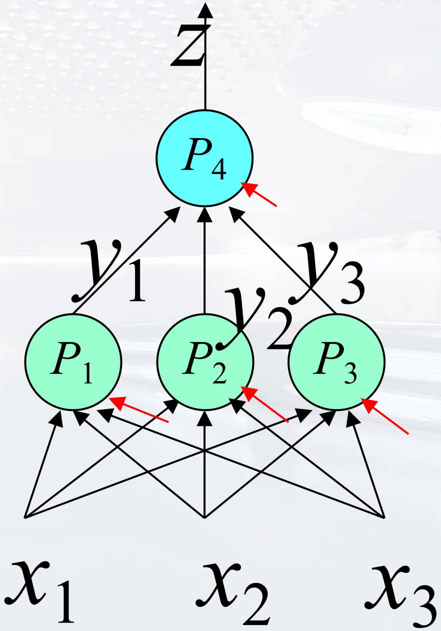


Parity Problem



Parity Problem





Contents

MIMA