人工智能：机器学习 II

饶洋辉
计算机学院，
中山大学
raoyangh@mail.sysu.edu.cn
http://cse.sysu.edu.cn/node/2471

课件来源：中山大学陈川副教授；台湾大学林轩田教授等

背景

- 人脑中的神经网络是一个非常复杂的组织，由很多具有适应性的简单同构单元组成的广泛并行互连的网络，它的组织能够模拟生物神经系统对真实世界所作出的交互反应。
- 这一简单的结构就是神经元。

- 成人的大脑中估计有 1000 亿个神经元。

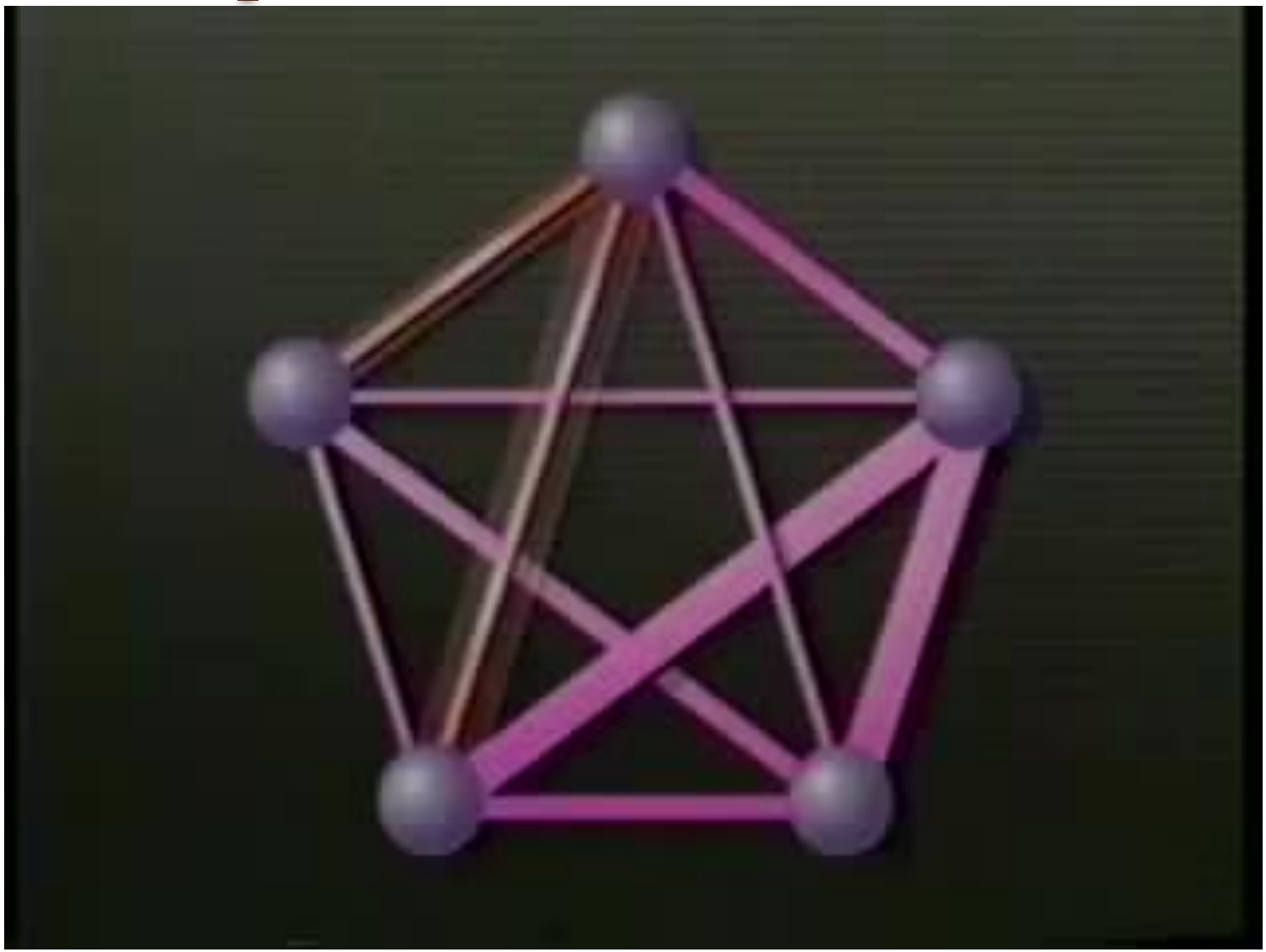
背景

- 联结主义（Connetionism）学派：认为人工智能源于仿生学。该学派的主要理论基础为神经网络及神经网络间的连接机制与学习算法。如果说符号主义是从宏观上模拟人的思维过程的话，那么联结主义则试图从微观上解决人类的认知功能，以探索认知过程的微观结构。
- 人的思维基元是神经元。该学派是把智能理解为相互联结的神经元竞争与协作的结果，以人工神经网络为代表。近年来深度神经网络的发展与应用，掀起了联结主义学派的研究热潮，其代表性领域是计算机视觉。

单层神经网络

- 1958年，计算科学家Rosenblatt提出了由两层神经元组成的神经网络。他给它起了一个名字——“感知机”（Perceptron，有的文献翻译成“感知器”）。
- 感知机是当时首个可以学习的人工神经网络。Rosenblatt现场演示了其学习识别简单图像的过程，在当时的社会引起了轰动。
- 人们认为已经发现了智能的奥秘，许多学者和科研机构纷纷投入到神经网络的研究中。美国军方大力资助了神经网络的研究，并认为神经网络比“原子弹工程”更重要。这段时间直到1969年才结束，这个时期可以看作神经网络的第一次高潮。

Perceptron in the 50's & 60's



感知机学习算法

- Perceptron Learning Algorithm
  - 适用于连续型属性，离散型属性需要转化为连续型随机变量。
- 对于拥有d个特征的 $\mathbf{x}=(x_1,x_2,\ldots,x_d)$，计算它的带权“分数”。

  如果 $\sum_{k=1}^{d} w_k x_k > threshold$，预测为+1(good)

  如果 $\sum_{k=1}^{d} w_k x_k < threshold$，预测为-1(bad)
- $\mathbf{y}$={+1(good), -1(bad)}

$$h(\mathbf{x}) = sign\left(\left(\sum_{k=1}^{d} w_k x_k\right) - threshold\right)$$

感知机学习算法

$$h(\mathbf{x}) = sign\left(\left(\sum_{k=1}^{d} w_k x_k\right) - threshold\right)$$

$$= sign\left(\left(\sum_{k=1}^{d} w_k x_k\right) + \underbrace{(-threshold)}_{w_0} \cdot \underbrace{(+1)}_{x_0}\right)$$

$$= sign\left(\sum_{j=0}^{d} w_j x_j\right)$$

$$= sign\left(\tilde{\mathbf{w}}^{\mathrm{T}} \tilde{\mathbf{x}}\right)$$

感知机学习算法

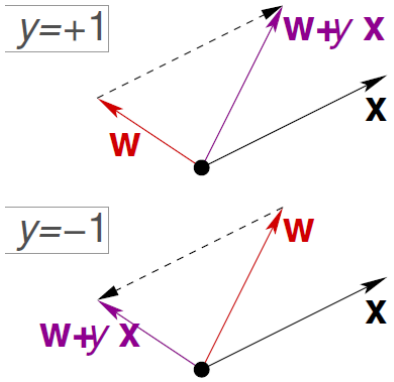
- 难点: 函数 $h(\mathbf{x})$ 有无限多种可能
- 想法: 先初始化$\mathbf{w}_{(0)}$, 然后根据D来修正$\mathbf{w}$。
- For t = 0, 1,…
  - **找到**$\mathbf{w}_{(t)}$ 预测错的数据 $(\mathbf{x}_{i(t)}, y_{i(t)})$

    $$sign\left(\tilde{\mathbf{w}}_{(t)}^{\mathrm{T}}\tilde{\mathbf{x}}_{i(t)}\right) \neq y_{i(t)}$$

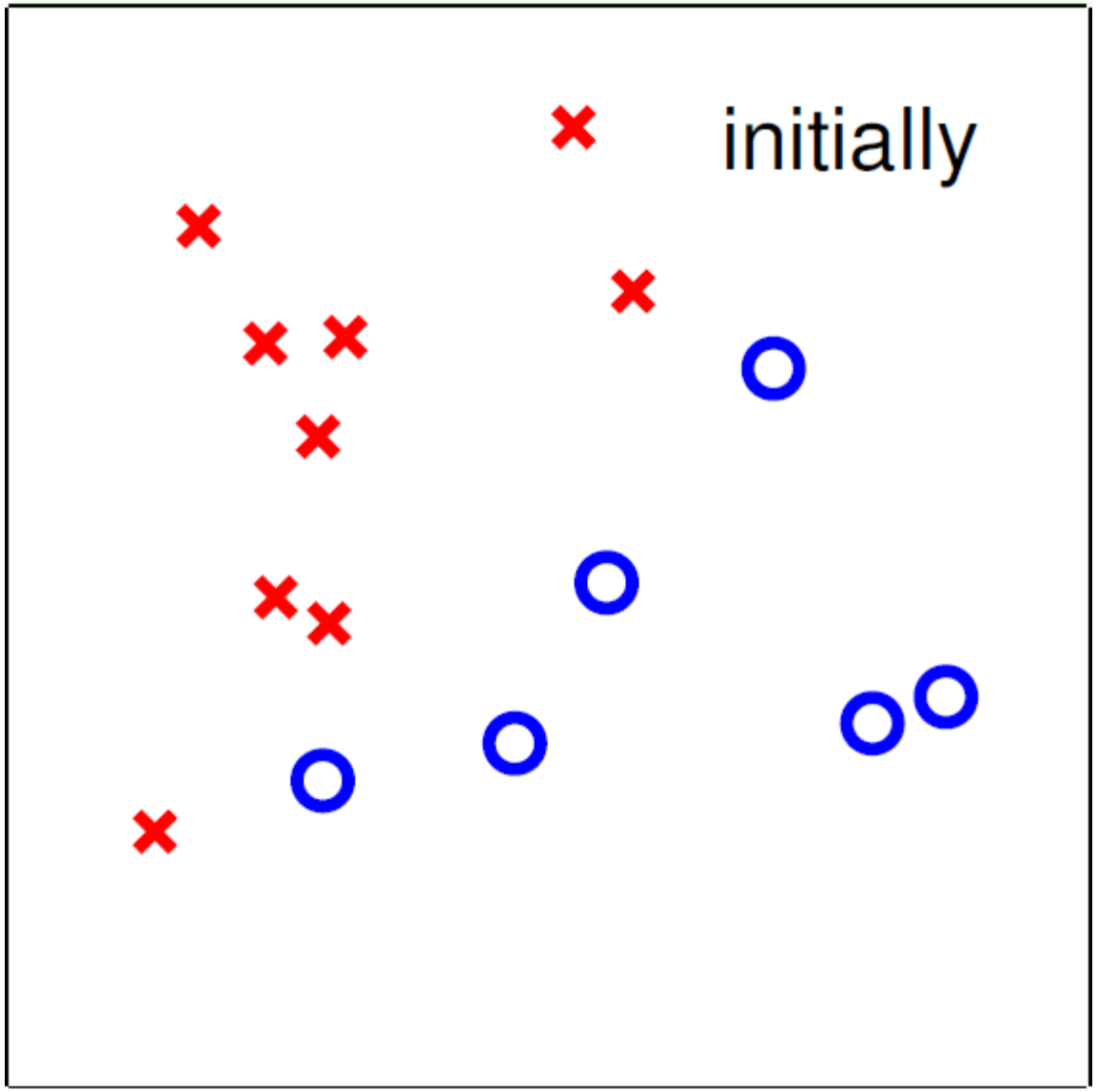
  - (尝试) 用下面的方法修正错误

    $$\tilde{\mathbf{w}}_{(t+1)} \leftarrow \tilde{\mathbf{w}}_{(t)} + y_{i(t)}\tilde{\mathbf{x}}_{i(t)}$$

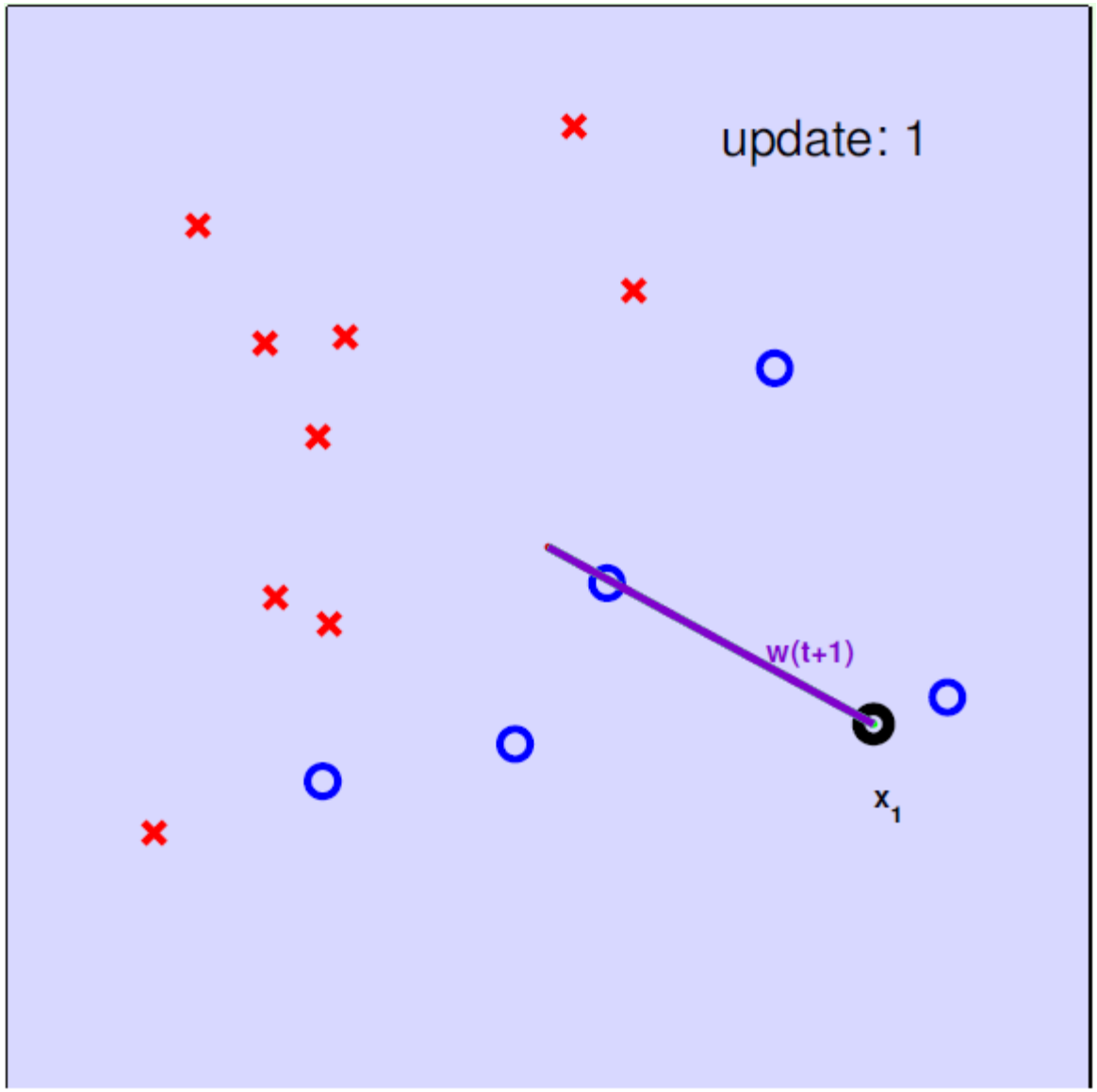
  - 直到没有错误
- 返回最终的 $\mathbf{W}$ (called $\mathbf{W}_{\mathrm{PLA}}$)



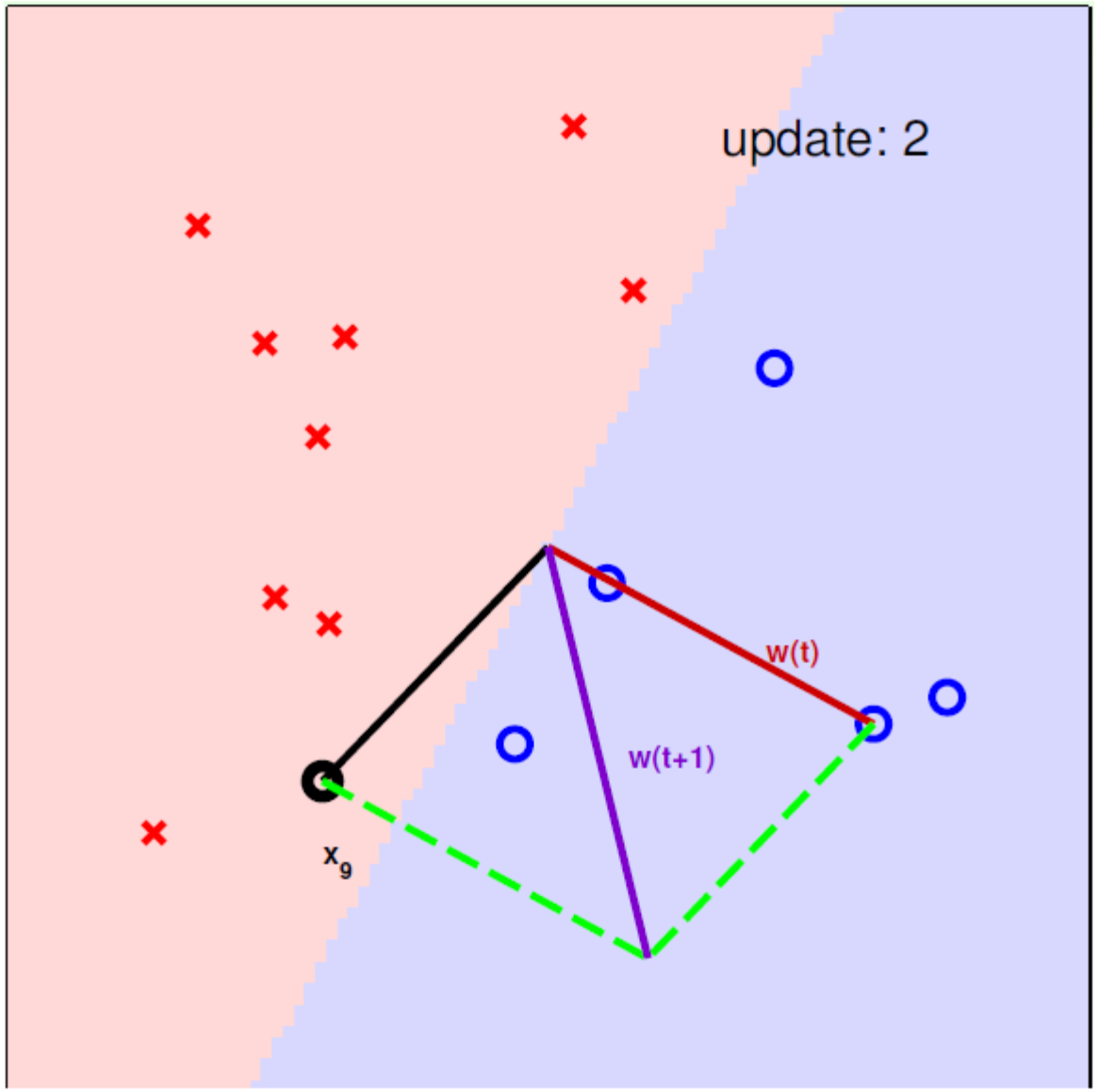
感知机学习算法



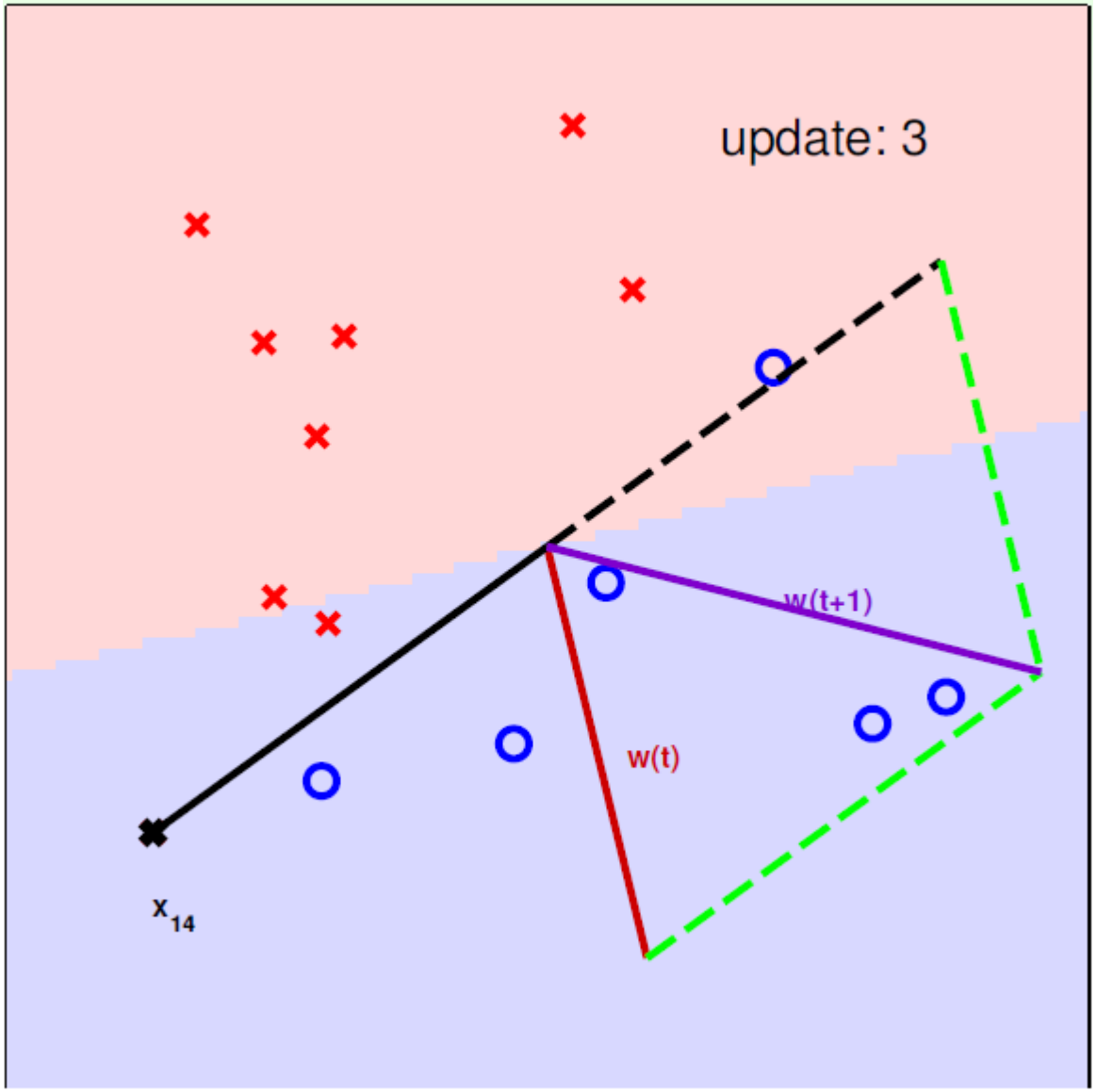
感知机学习算法



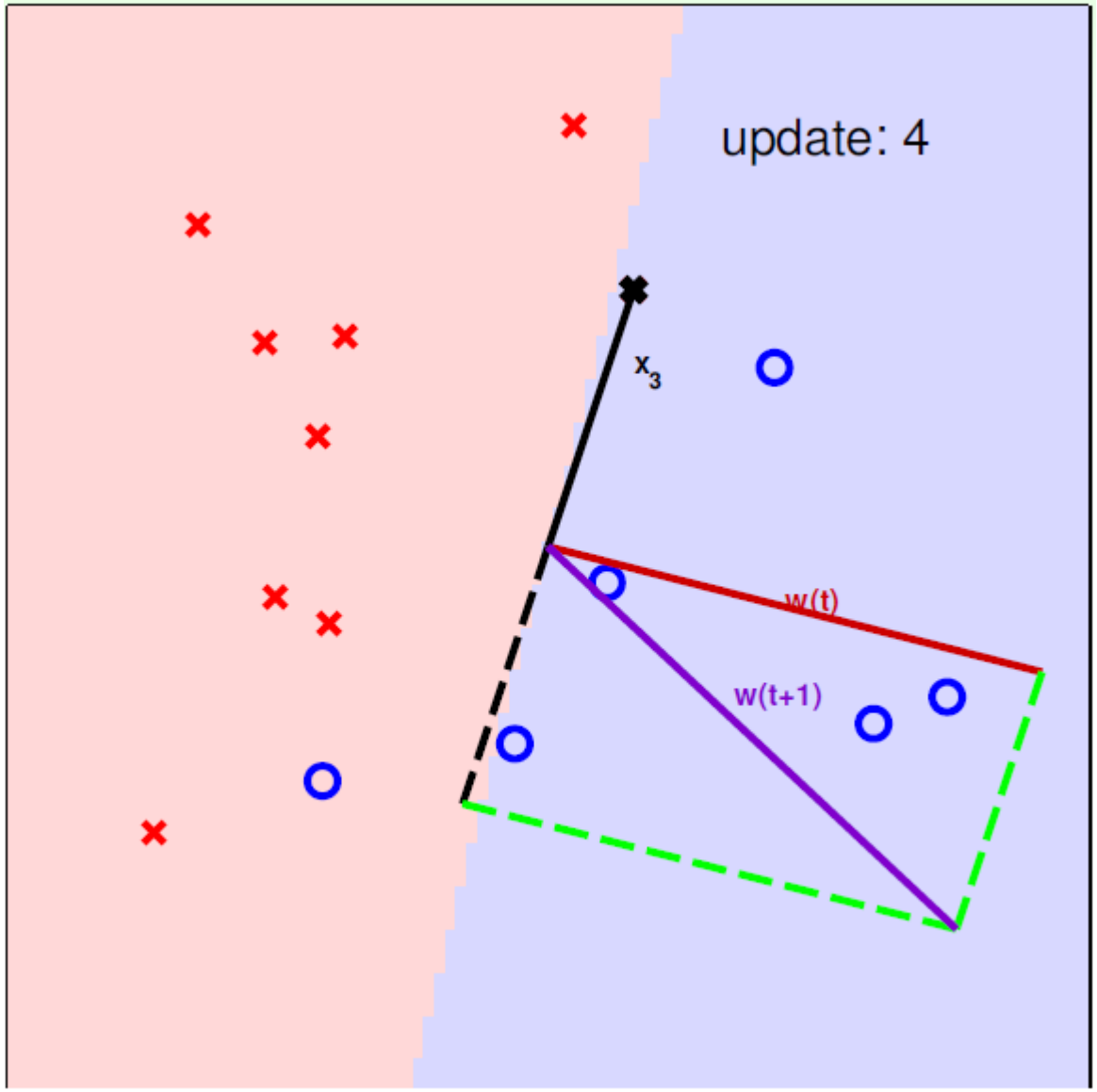
感知机学习算法



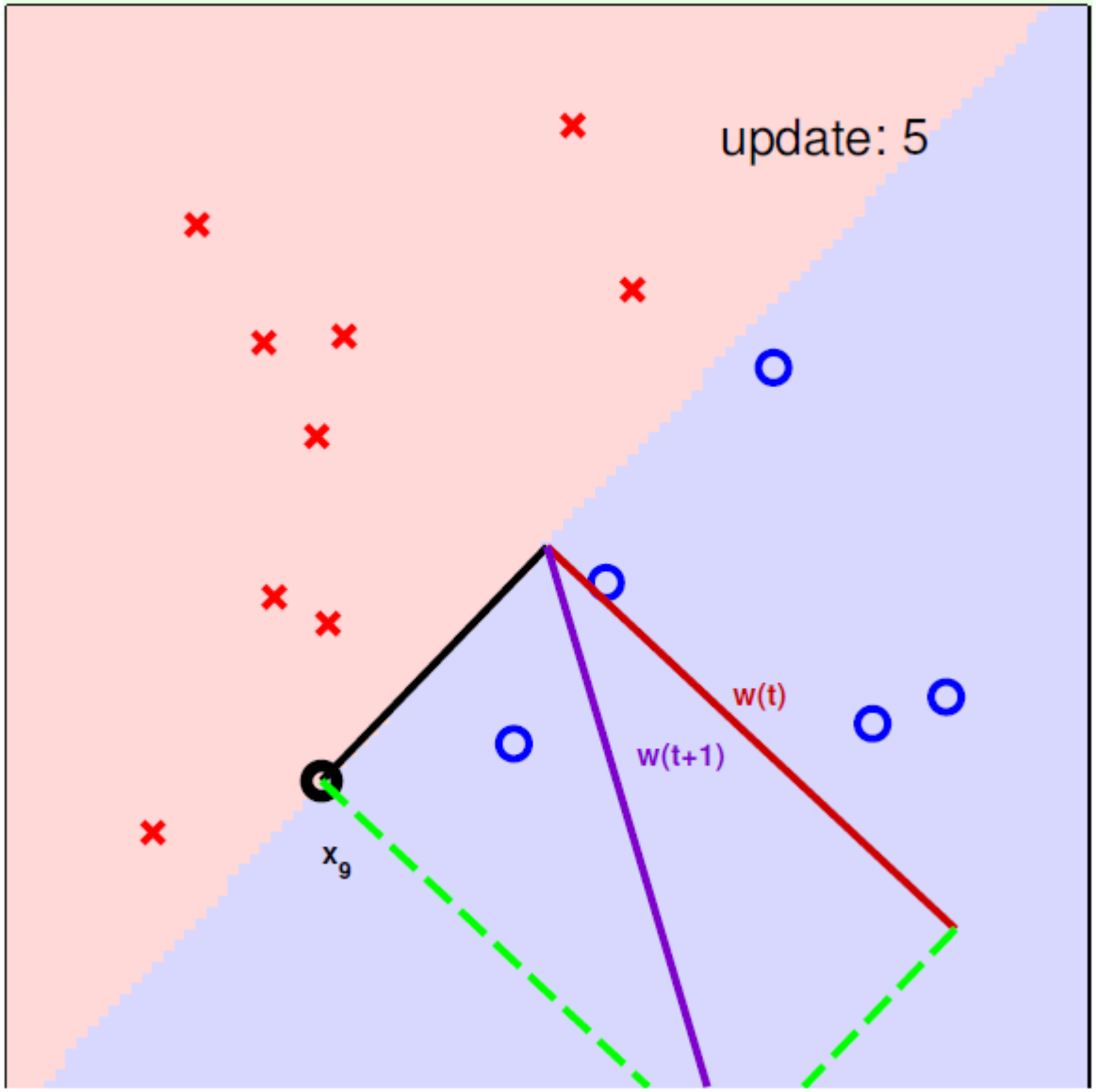
感知机学习算法



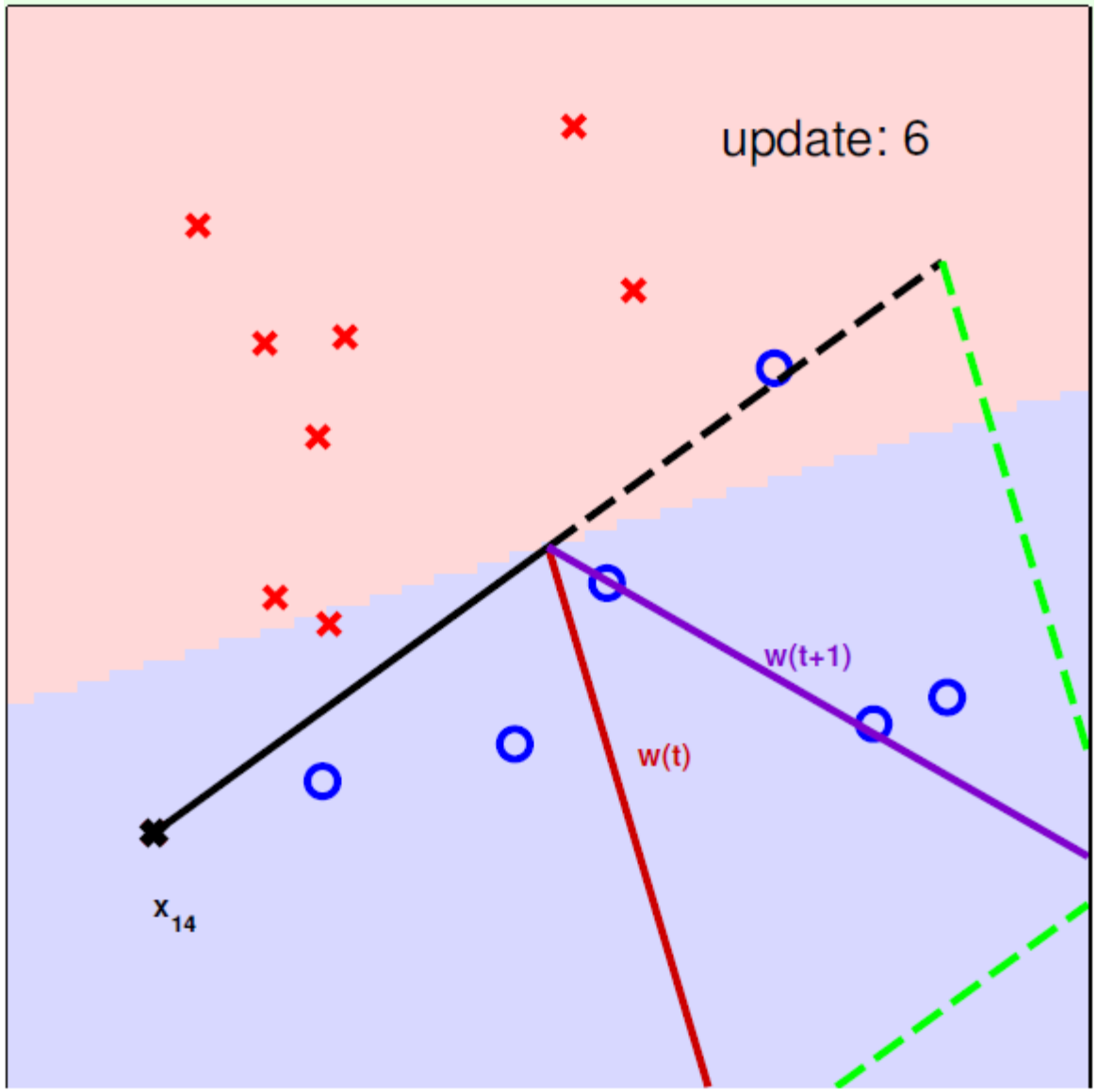
感知机学习算法



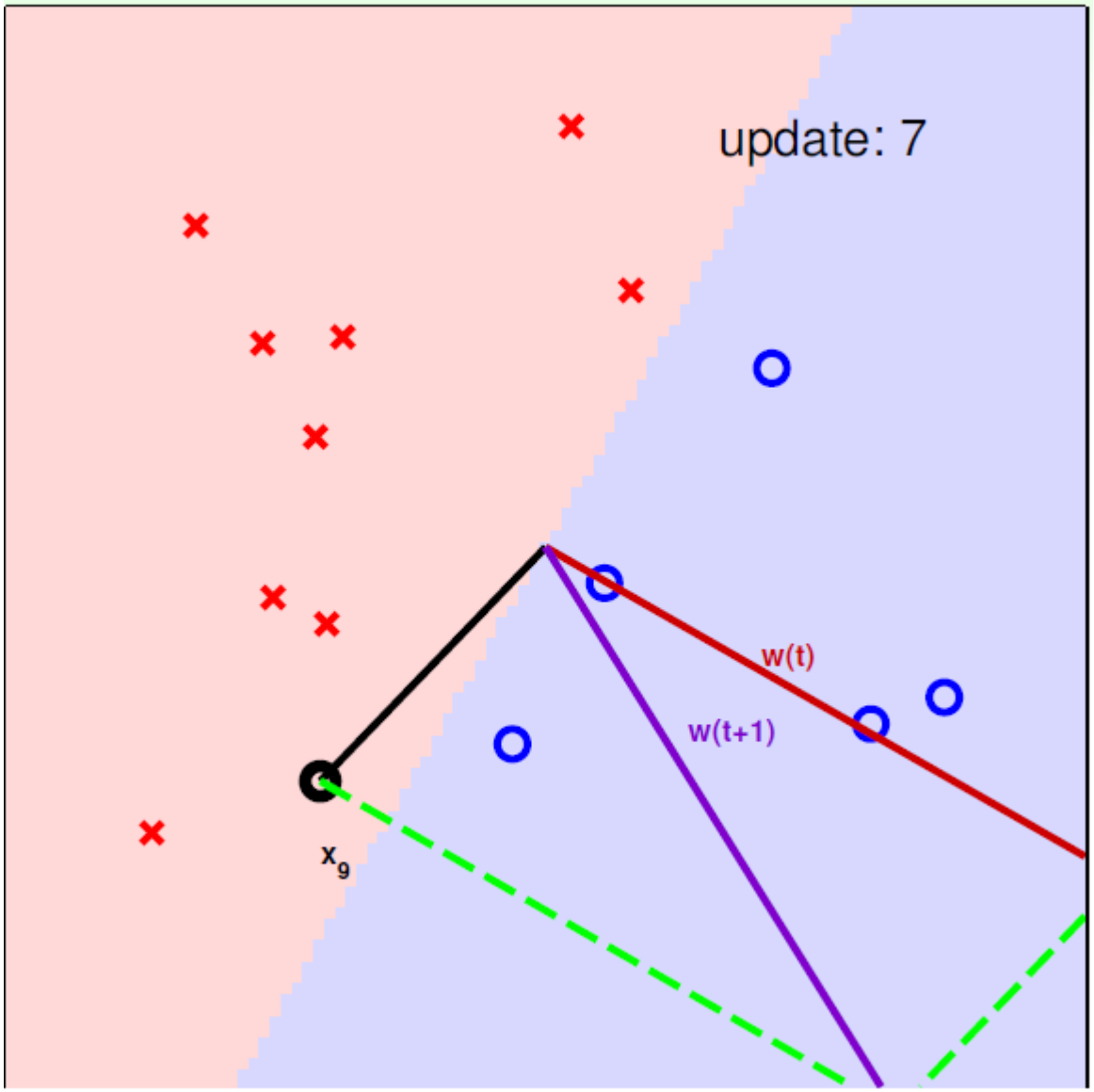
感知机学习算法



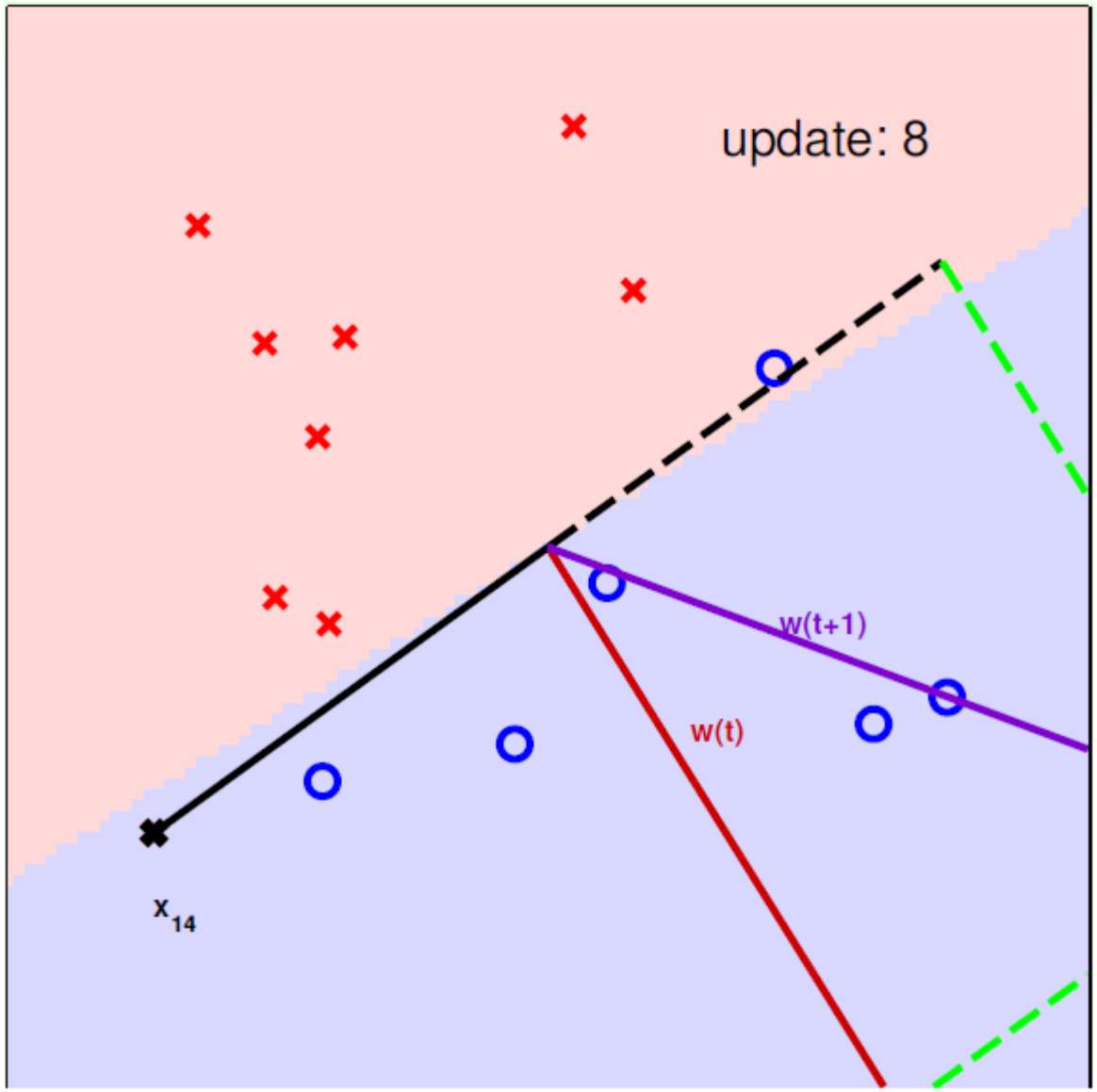
感知机学习算法



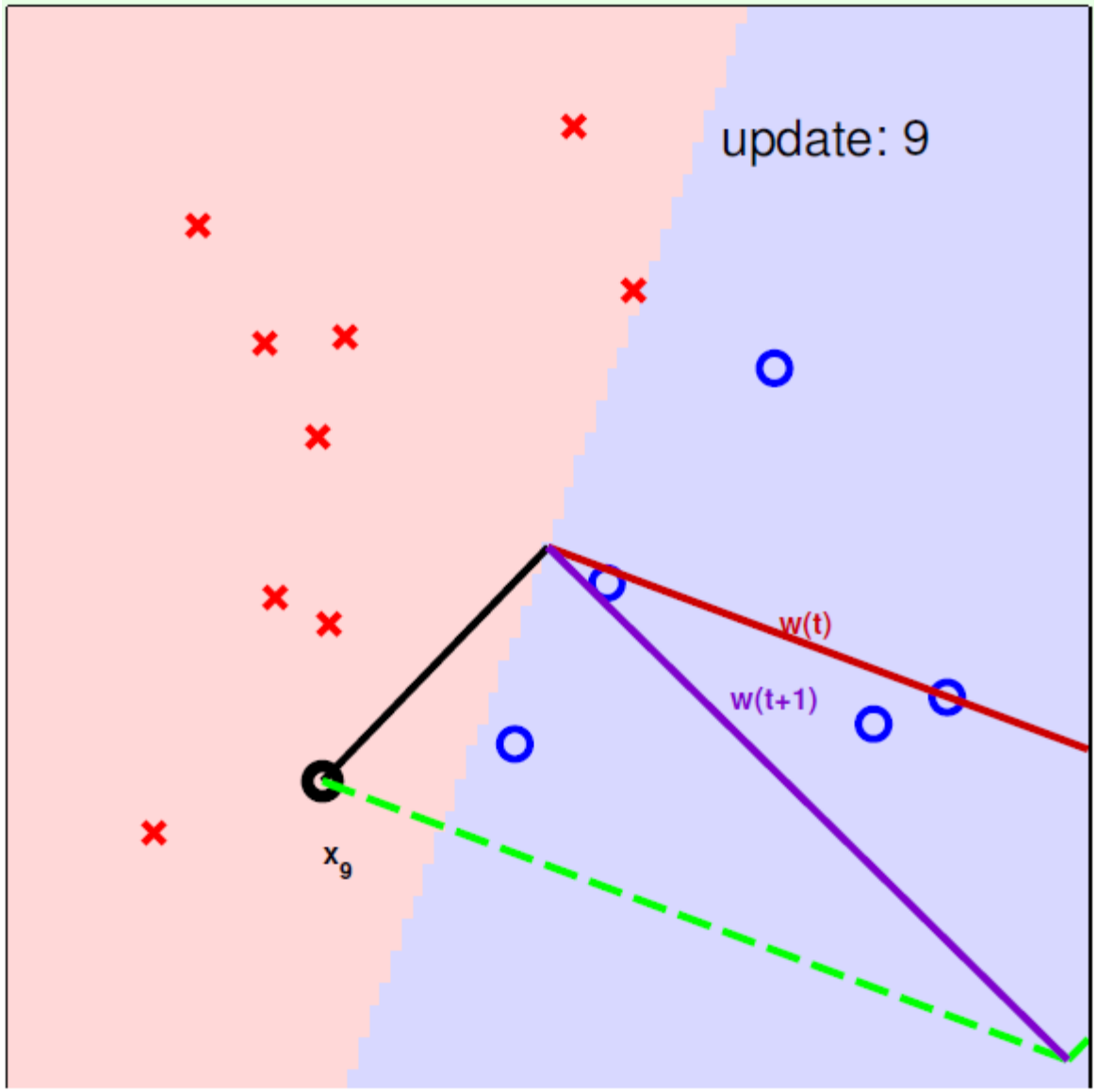
感知机学习算法



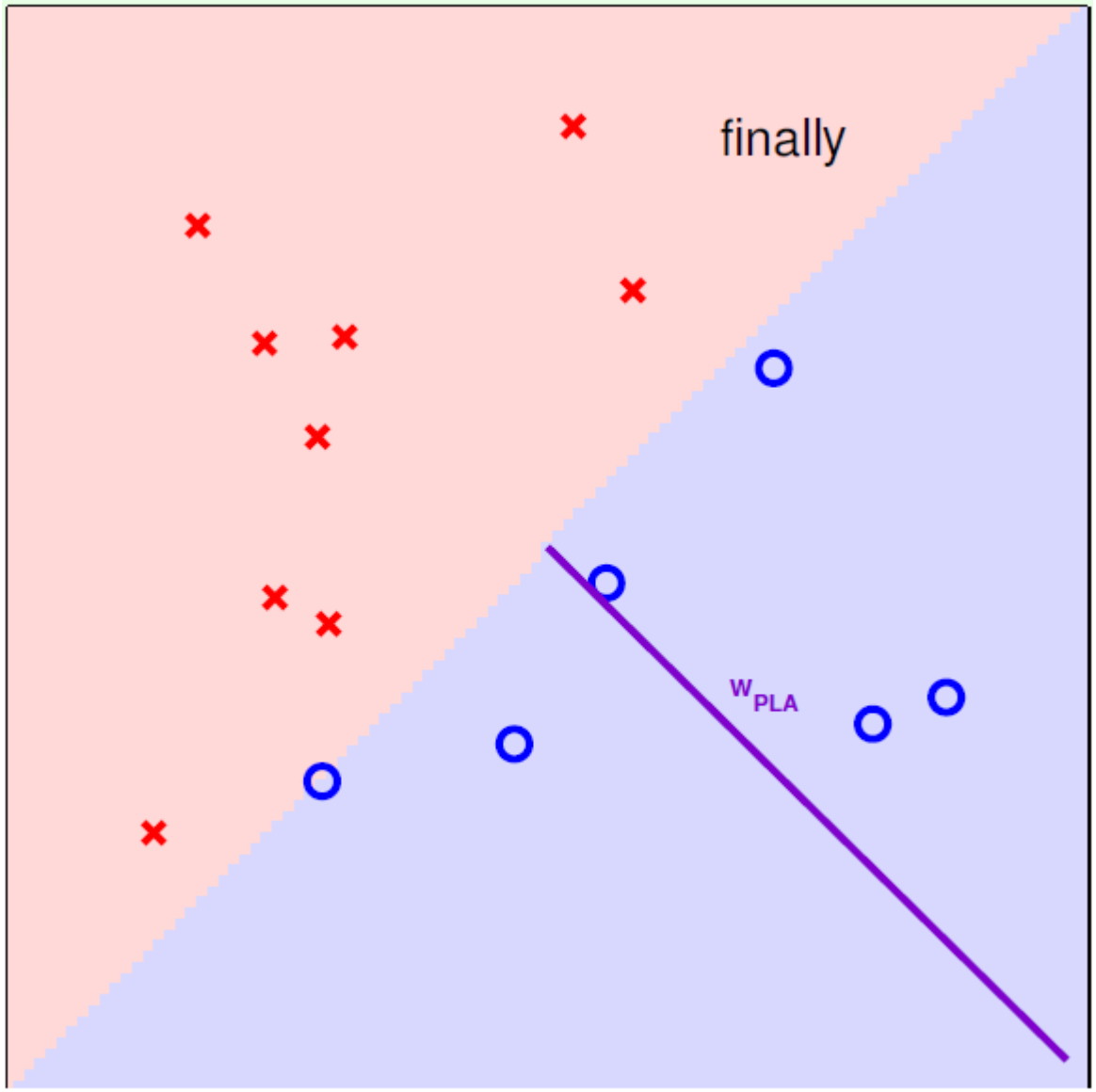
感知机学习算法



感知机学习算法

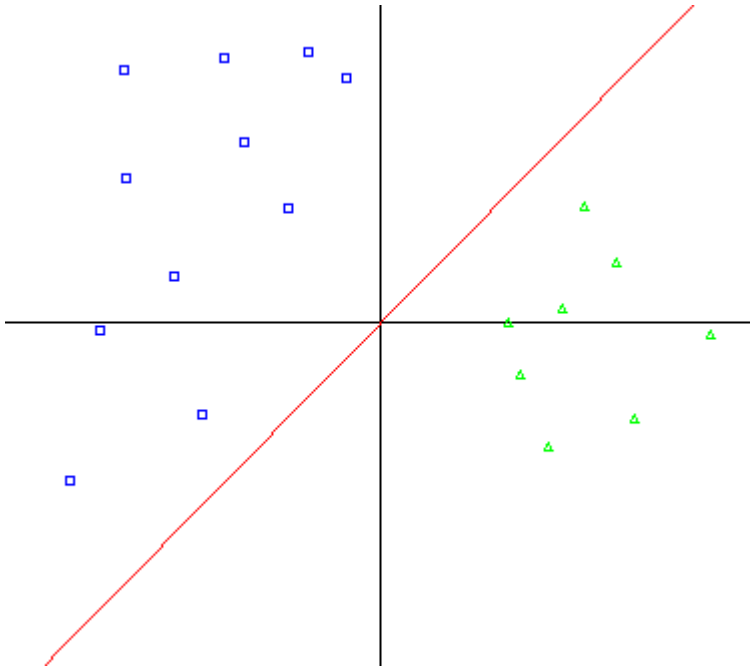


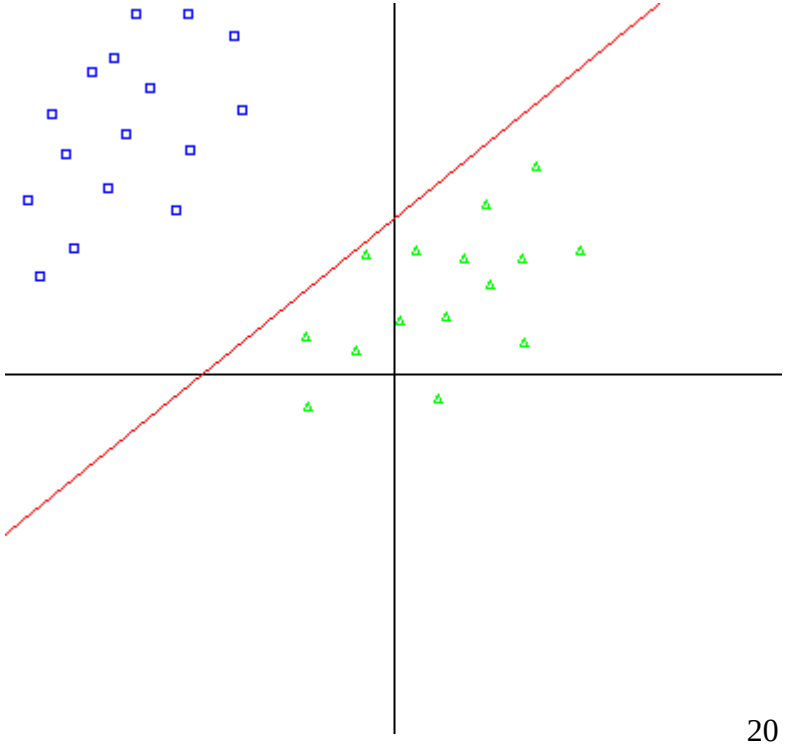
感知机学习算法



感知机学习算法

- 遍历数据的顺序不同，可能会导致结果不同，因此有多个解存在。





逻辑回归模型

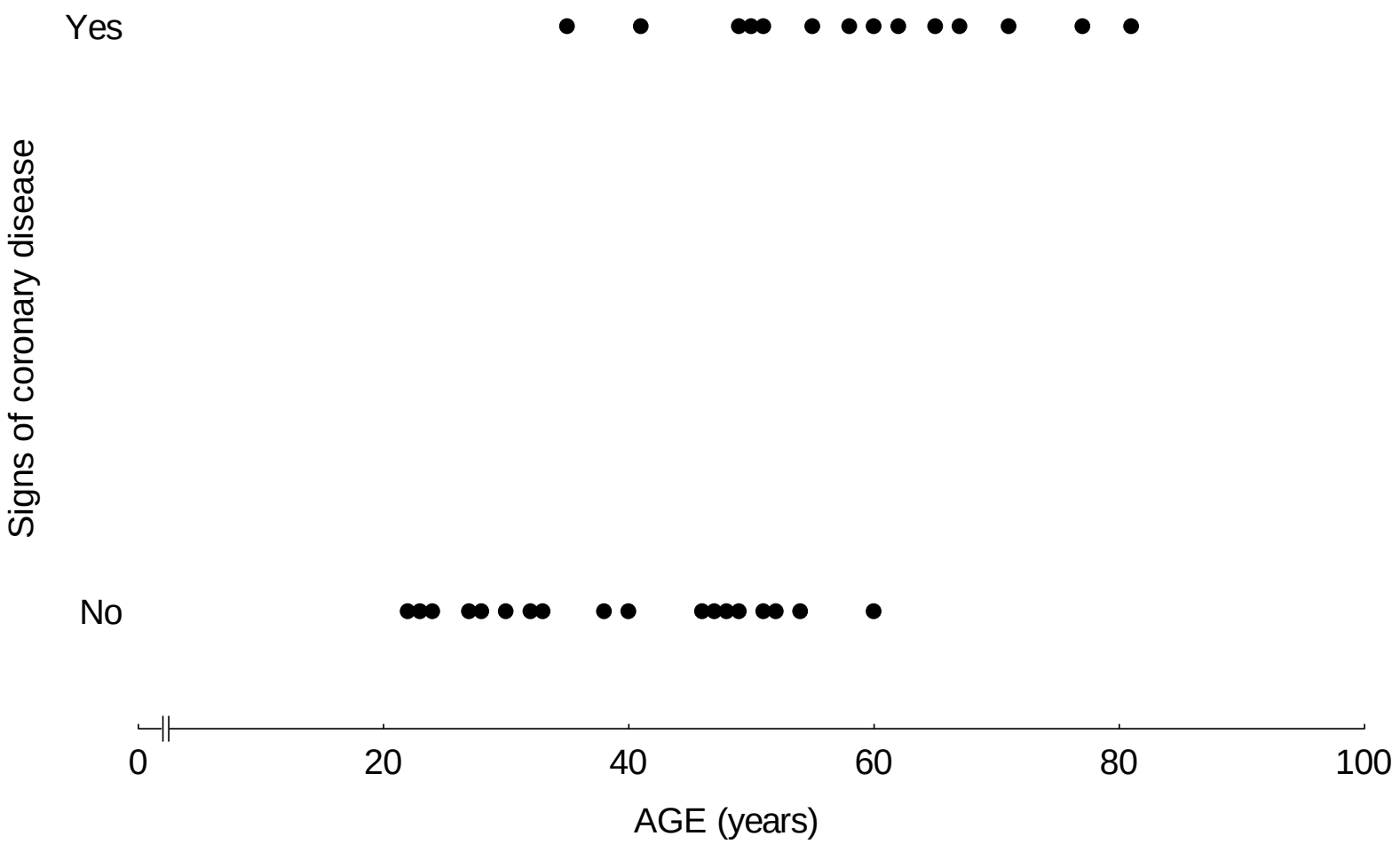
- 如果使用最小二乘法的回归模型来做二分类任务：

$$y = w_0 + \sum_{j=1}^{d} w_j x_j + u$$

$$= \tilde{\mathbf{w}}^{\mathrm{T}} \tilde{\mathbf{x}}$$

- 基于上述模型预测的y值，即样本属于某个类的概率，会超出0到1的范围。

逻辑回归模型



逻辑回归模型

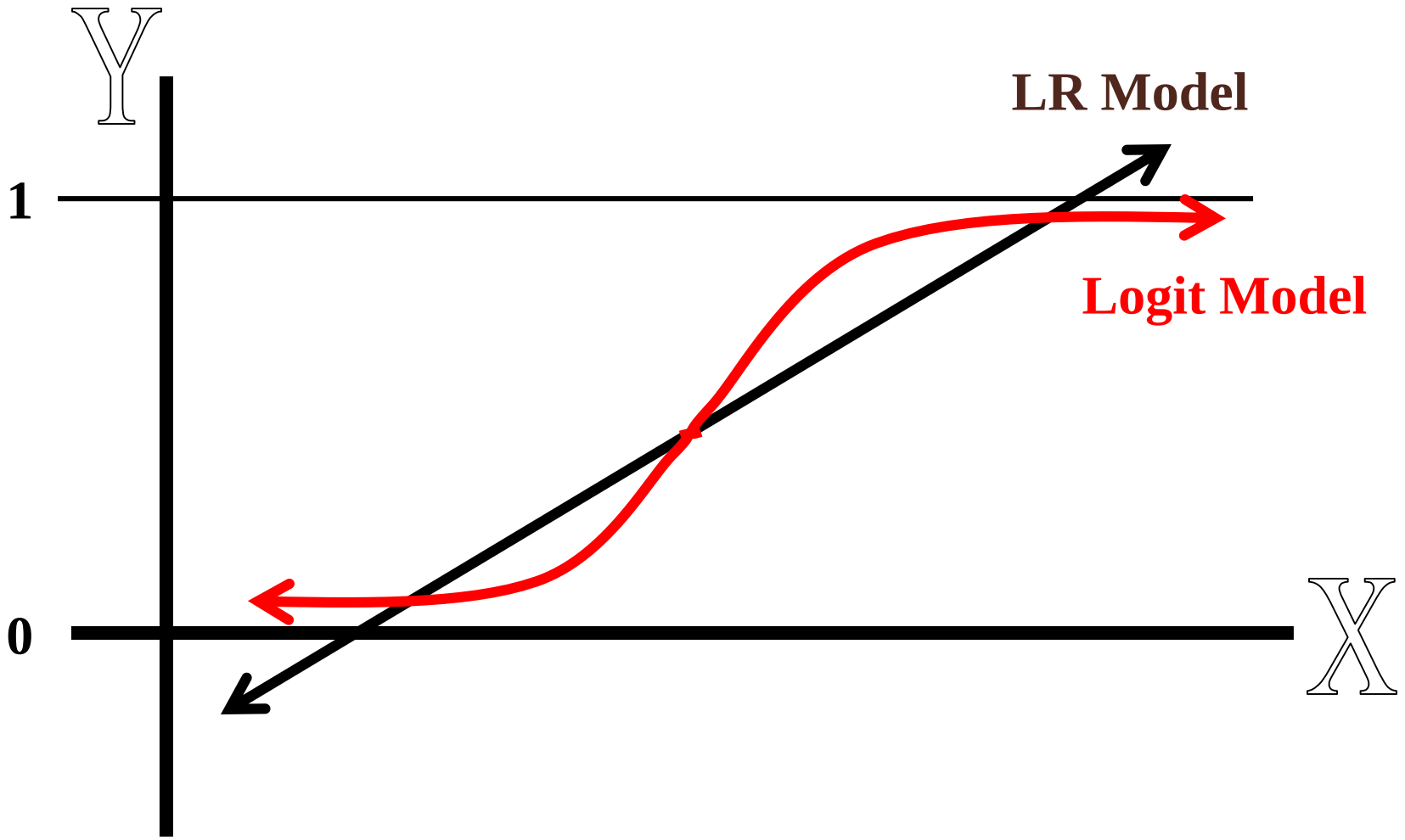
- “logit” 变换可以解决上述问题：

$$\log\left(\frac{p}{1-p}\right) = w_0 + \sum_{j=1}^{d} w_j x_j + u$$

$$= \tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}$$

- p 是事件y发生的概率，比如：$p=p(y=1|\mathbf{X})$
- $p/(1-p)$ 称为机率比或优势比 (odds ratio)
- $\log[p/(1-p)]$ 是机率比的对数，或称为 "logit"

逻辑回归模型



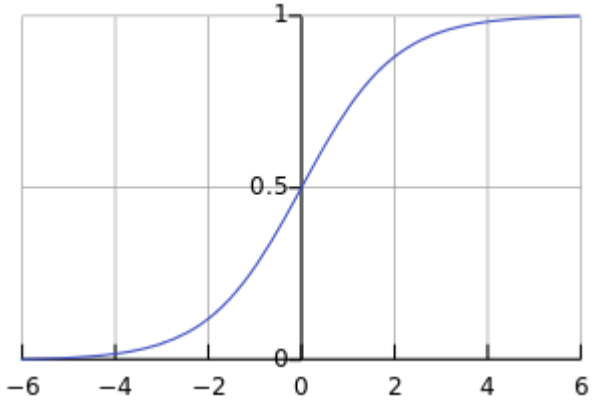
逻辑回归模型

- logistic 函数使得输出的概率值在0到1的范围内.

- 样本 **X** 标签为正的概率 $p(y=1|\mathbf{X})$ 是:

$$p=\frac{1}{1+e^{-w_0-\sum_{j=1}^{d}w_jx_j}}=\frac{e^{w_0+\sum_{j=1}^{d}w_jx_j}}{1+e^{w_0+\sum_{j=1}^{d}w_jx_j}}$$

$$=\frac{1}{1+e^{-\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}}}=\frac{e^{\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}}}{1+e^{\tilde{\mathbf{w}}^{\mathrm{T}}\tilde{\mathbf{x}}}}$$



- 如果 $w_0+\sum_{j=1}^{d}w_jx_j=0$, 那么 $p=0.5$
- 当 $w_0+\sum_{j=1}^{d}w_jx_j$ 很大时, p 趋近于 1
- 当 $w_0+\sum_{j=1}^{d}w_jx_j$ 很小时, p 趋近于 0

逻辑回归模型

- 最小二乘法回归模型，使用了最小二乘的公式，直接得到了最终的模型。
- 对于逻辑回归，可以使用极大似然估计，配合以一种迭代式的方法，计算出最终的模型。
- 算法：
  - 首先，随机初始化权重，并对某个样本进行预测；
  - 接着，计算这个模型在这次预测上的误差，改变权重，以提高模型在这个样本上的似然度；
  - 重复这个过程，直到模型收敛，即当前模型和上一步的模型的表现相差无几。
- 该想法的本质：找到一个最有可能产生你观察到的数据的参数。

逻辑回归模型

- 似然函数：$\prod_{i=1}^{n}(p_i)^{y_i}(1-p_i)^{1-y_i}$
- 极大似然法：

$$L(\tilde{\mathbf{W}})=\sum_{i=1}^{n}\left(y_i\log p_i+(1-y_i)\log(1-p_i)\right)$$

$$=\sum_{i=1}^{n}\left(y_i\log\frac{p_i}{1-p_i}+\log(1-p_i)\right)$$

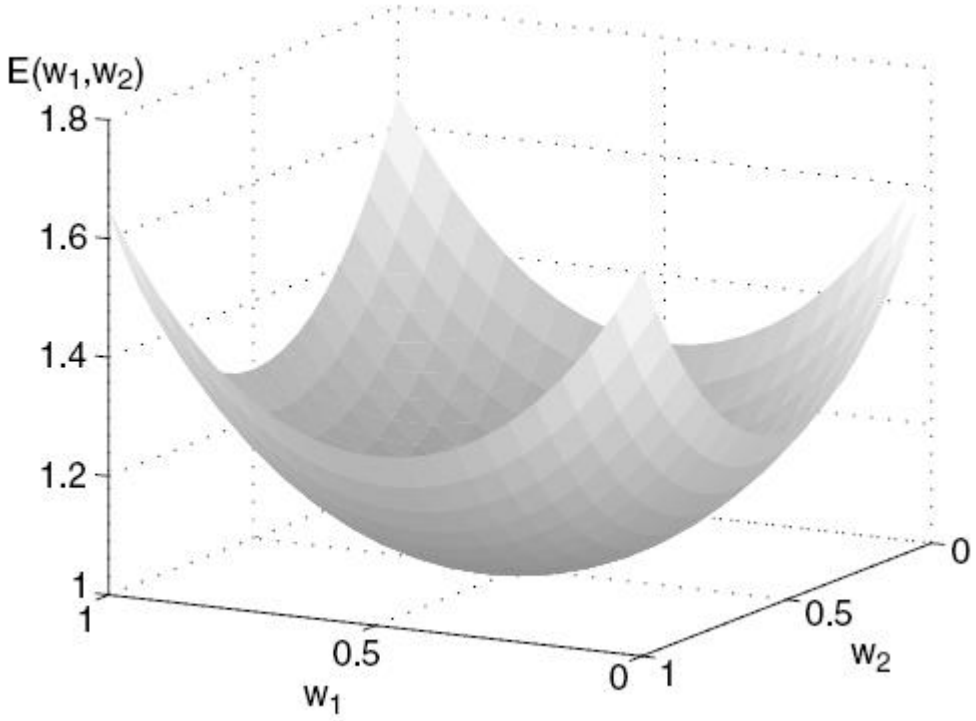
$$=\sum_{i=1}^{n}\left(y_i\tilde{\mathbf{W}}^{\mathrm{T}}\tilde{\mathbf{X}}_i-\log(1+e^{\tilde{\mathbf{W}}^{\mathrm{T}}\tilde{\mathbf{X}}_i})\right)$$

$$\frac{\partial L(\tilde{\mathbf{W}})}{\partial\tilde{\mathbf{W}}}=\sum_{i=1}^{n}\left[\left(y_i-\frac{e^{\tilde{\mathbf{W}}^{\mathrm{T}}\tilde{\mathbf{X}}_i}}{1+e^{\tilde{\mathbf{W}}^{\mathrm{T}}\tilde{\mathbf{X}}_i}}\right)\tilde{\mathbf{X}}_i\right]$$

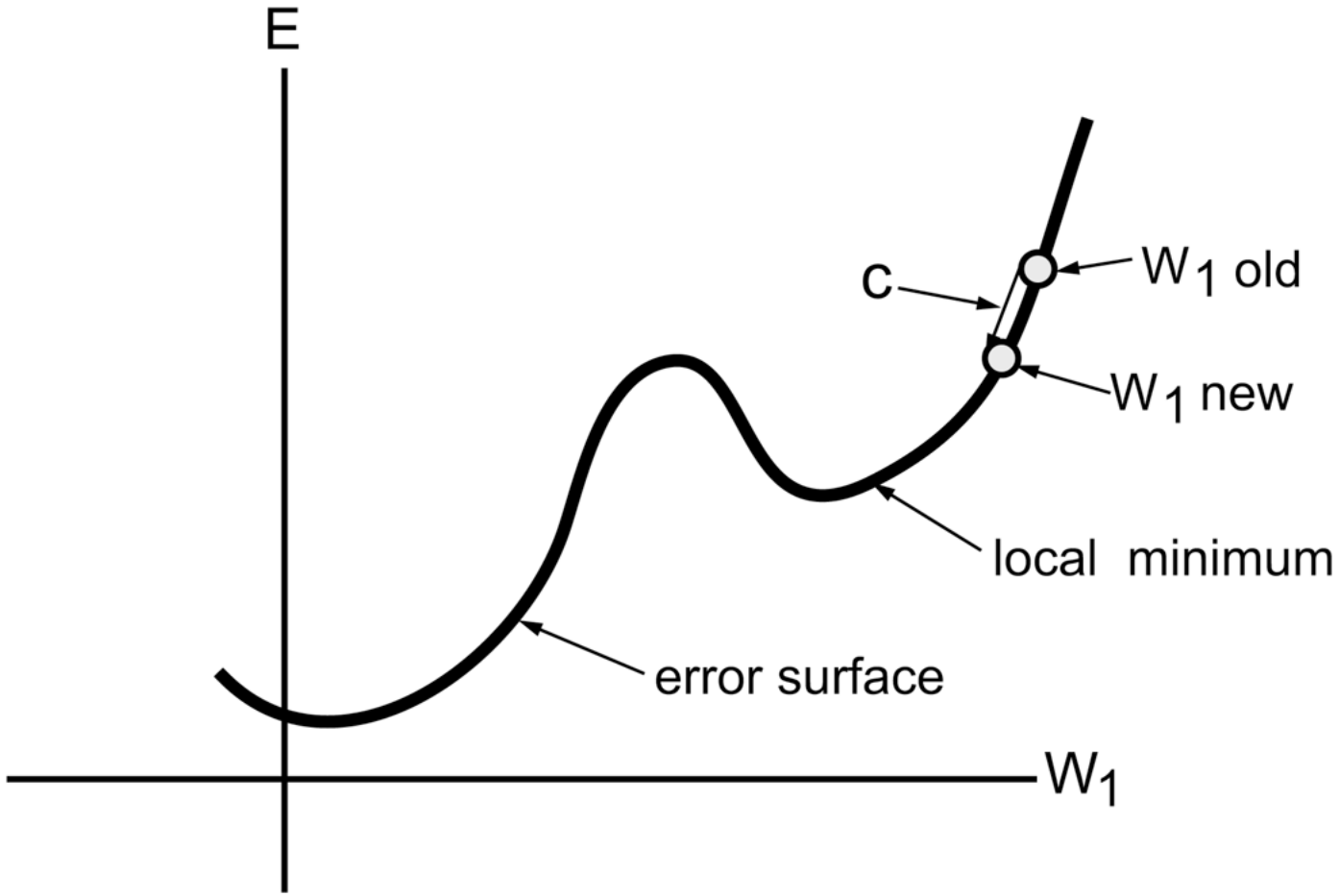
- 等价于最小化代价函数：

$$C(\tilde{\mathbf{W}})=-L(\tilde{\mathbf{W}})=-\sum_{i=1}^{n}\left(y_i\log p_i+(1-y_i)\log(1-p_i)\right)$$ 交叉熵

梯度下降



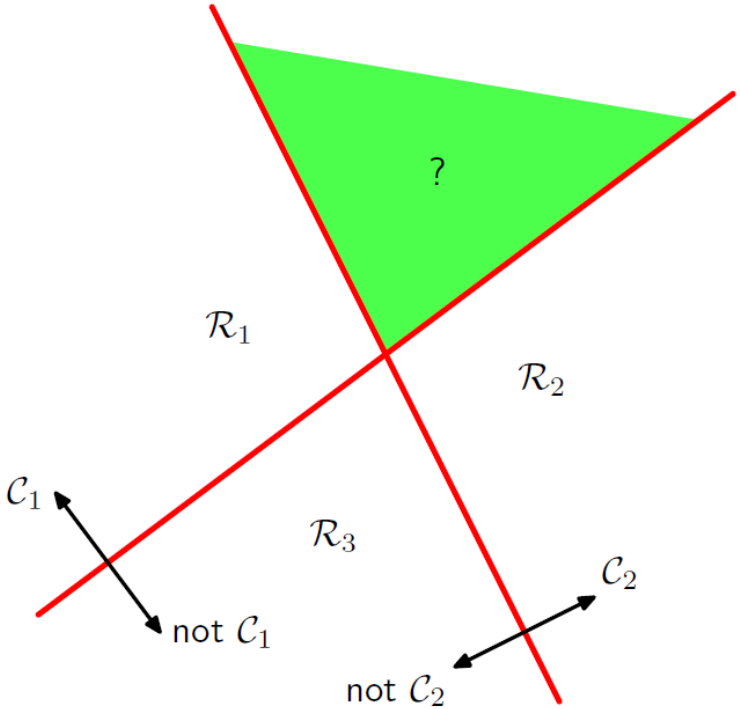
梯度下降

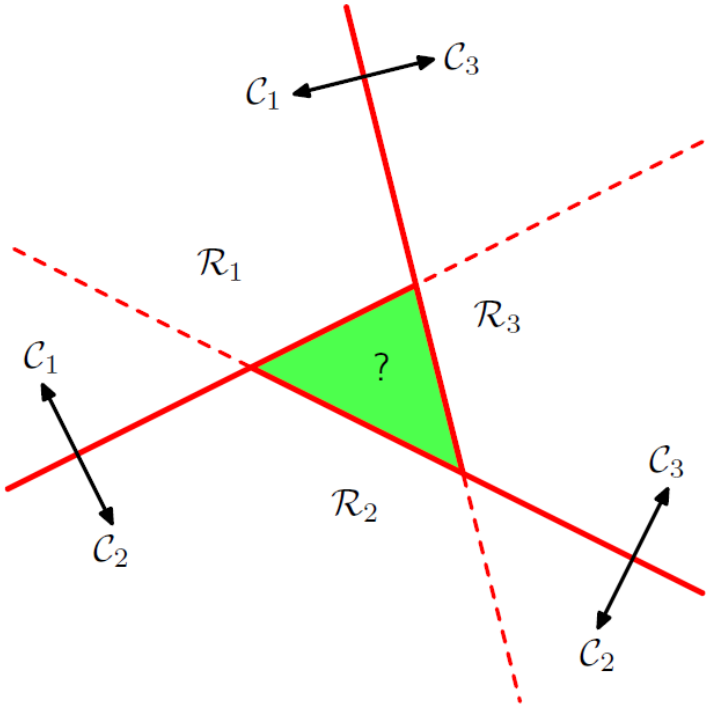


逻辑回归模型

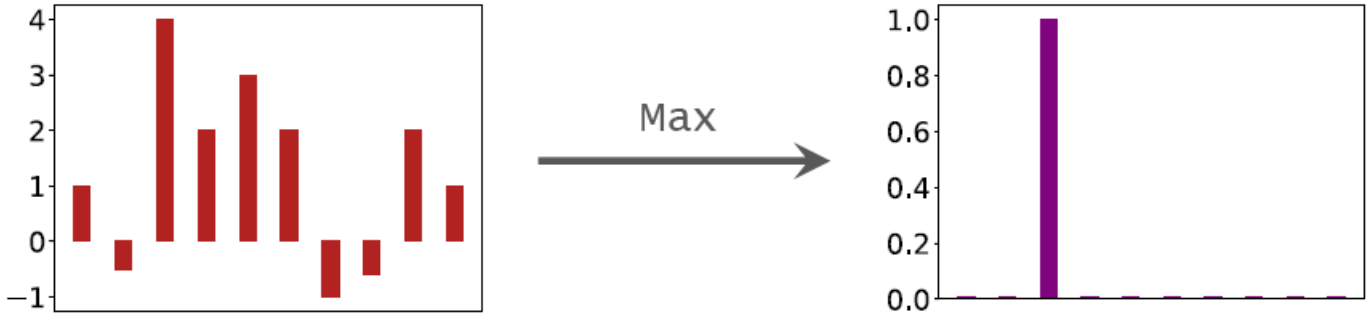
- 梯度下降
  - 计算梯度向量
  - 每次用梯度向量的反方向来更新权重
- 重复:
$$\tilde{\mathbf{w}}_{new}^{(j)} = \tilde{\mathbf{w}}^{(j)} - \eta \frac{\partial C(\tilde{\mathbf{w}})}{\partial \tilde{\mathbf{w}}^{(j)}}$$
$$= \tilde{\mathbf{w}}^{(j)} - \eta \sum_{i=1}^{n} \left[\left(\frac{e^{\tilde{\mathbf{w}}^{\mathrm{T}} \tilde{\mathbf{x}}_i}}{1 + e^{\tilde{\mathbf{w}}^{\mathrm{T}} \tilde{\mathbf{x}}_i}} - y_i \right) \tilde{\mathbf{x}}_i^{(j)} \right]$$
- 直至收敛。

多类别分类


?
$\mathcal{R}_1$
$\mathcal{R}_2$
$\mathcal{R}_3$
$\mathcal{C}_1$
not $\mathcal{C}_1$
$\mathcal{C}_2$
not $\mathcal{C}_2$


$\mathcal{C}_1$
$\mathcal{C}_3$
$\mathcal{R}_1$
$\mathcal{R}_3$
$\mathcal{C}_1$
?
$\mathcal{C}_3$
$\mathcal{R}_2$
$\mathcal{C}_2$
$\mathcal{C}_2$

Softmax



非线性可分