人工智能与自动化学院

模式识别

第二讲 感知器（*Perceptron for Pattern Recognition*）

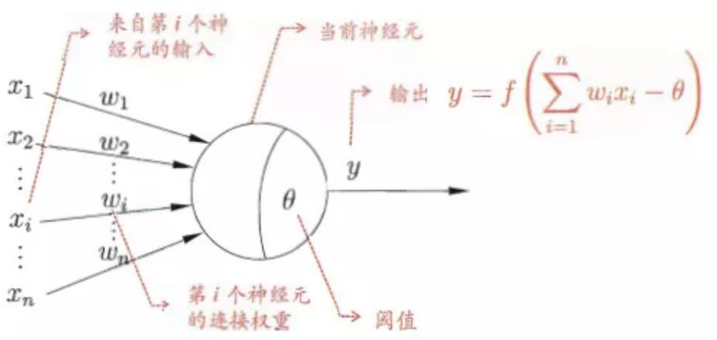
2.1 感知器模型参数空间 （*Perceptron Hypothesis Set*）

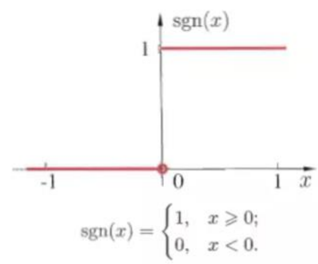
2.2 感知器算法 （*Perceptron Learning Algorithm: PLA*）

2.3 感知器算法的收敛性 （*Guarantee of PLA*）

2.4 线性不可分情况 （Non-separable Data）

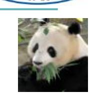
感知器Perceptron





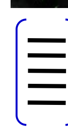
2.1 感知器模型参数空间

测试(Testing)



输入图像

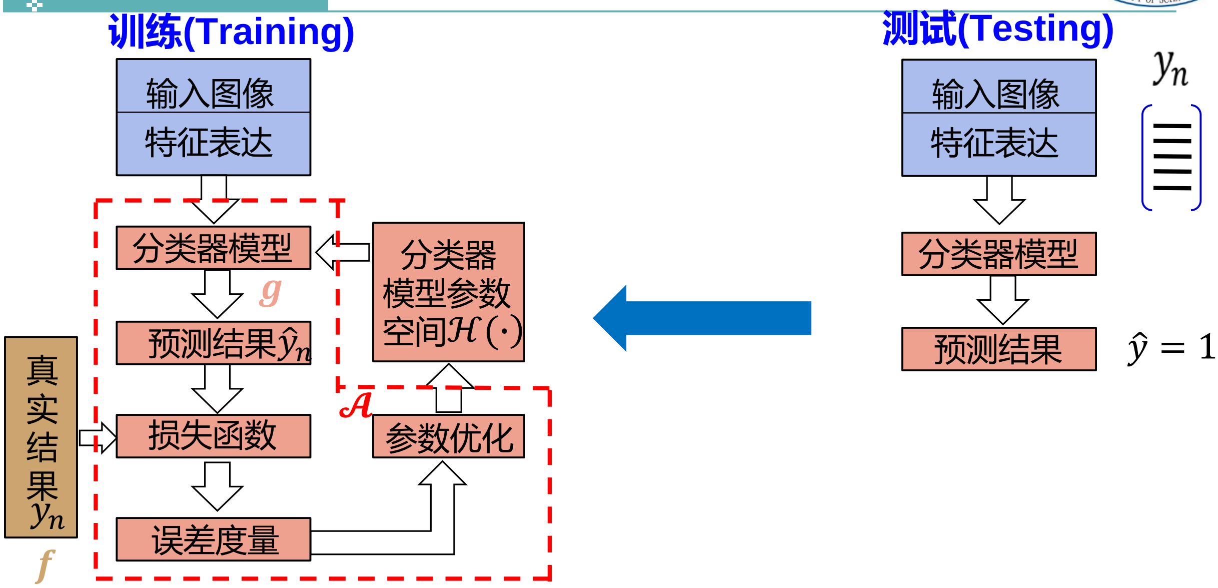
特征表达



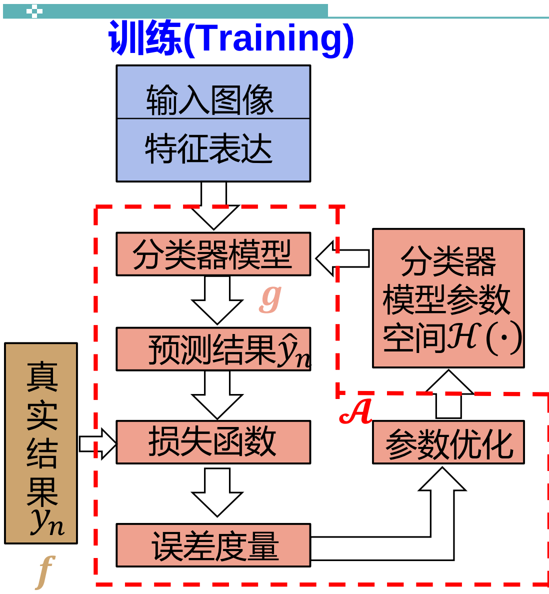
分类器模型

预测结果 $\hat{y} = 1$

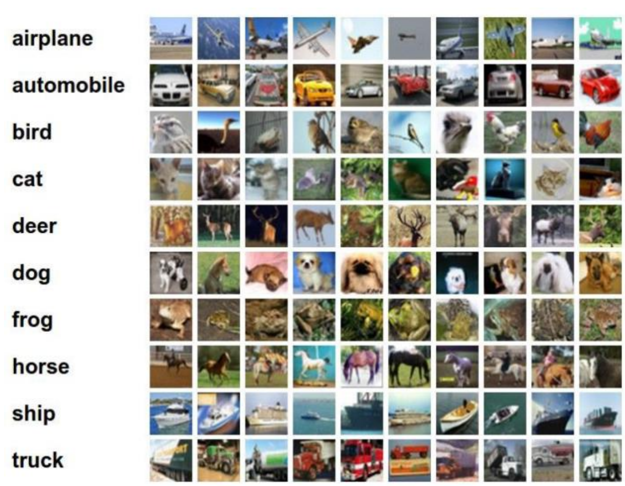
2. 1 感知器模型参数空间



2. 1 感知器模型参数空间

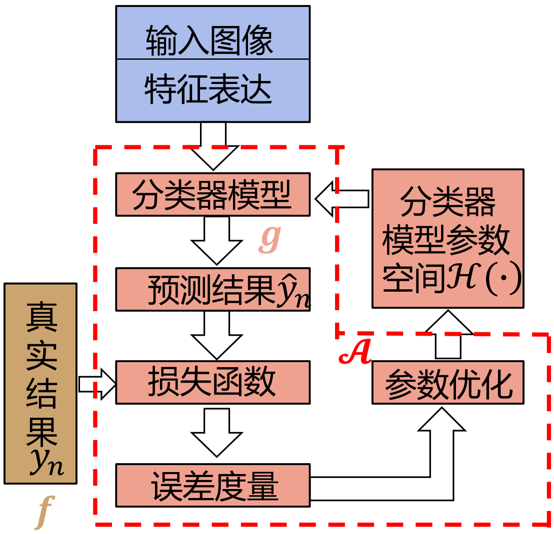


训练数据集



2.1 感知器模型参数空间

训练(Training)



训练数据集

Example dataset: CIFAR-10

labels: 10

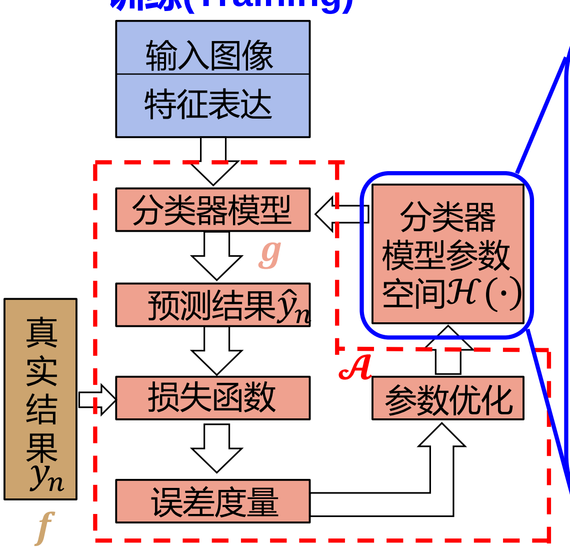
training images: 50,000

each image is 32x32x3

test images:

2.1 感知器模型参数空间

训练(Training)



$\mathcal{H}(\cdot)$ 像什么?

考虑二元分类 (Binary Classification)情况

$$\mathbf{x} = (x_1, x_2, \ldots x_d)^T$$

$$Yes, \quad if \quad \sum_{i=1}^{d} w_i x_i > threshold$$

$$No, \quad if \quad \sum_{i=1}^{d} w_i x_i < threshold$$

$\mathcal{Y} = \{+1(Yes), \ -1(No)\}$, $if\ \mathcal{Y} = 0$ 拒识

$h(\quad) \in \mathcal{H}(\cdot)$

$$h(\mathbf{x}) = sign(\left(\sum_{i=1}^{d} w_i x_i\right) - threshold)$$

2.1 感知器模型参数空间

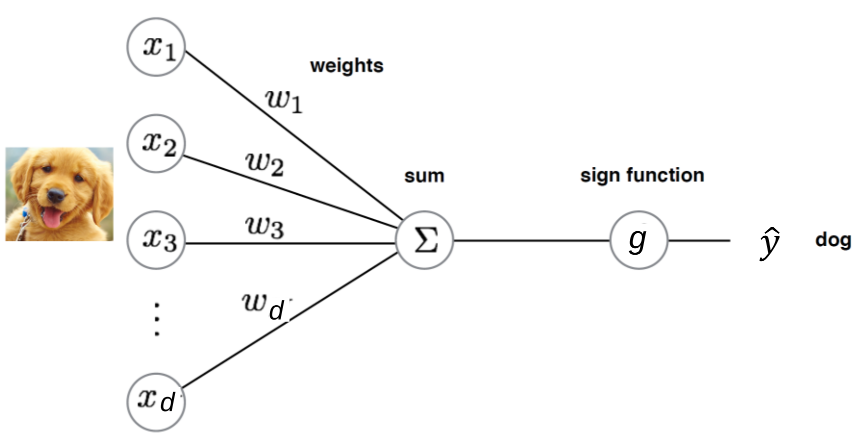
用向量形式(*Vector Form*)来表示感知器模型：

$$
\begin{aligned}
h(\mathbf{x}) &= \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) - \text{threshold}\right) \\
&= \text{sign}\left(\left(\sum_{i=1}^{d} w_i x_i\right) + \underbrace{(-\text{threshold})}_{w_0} \cdot \underbrace{(+1)}_{x_0}\right) \\
&= \text{sign}\left(\boxed{\sum_{i=0}^{d} w_i x_i}\right) \\
&= \text{sign}\left(\mathbf{w}^T \mathbf{x}\right)
\end{aligned}
$$

对样本的特征向量**x**和权向量***w*** 增广化

2.1 感知器模型参数空间

将感知器算法用于图像分类(*image classification*)示例：

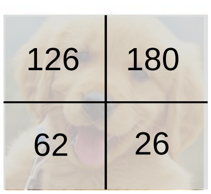


2. 1 感知器模型参数空间

将感知器算法用于图像分类(*image classification*)示例：

2. 1 感知器模型参数空间

将感知器算法用于图像分类(*image classification*)示例：



$$\mathbf{x}=\begin{pmatrix}126\\180\\62\\26\end{pmatrix} \quad \mathbf{x}=\begin{pmatrix}1\\126\\180\\62\\26\end{pmatrix}$$

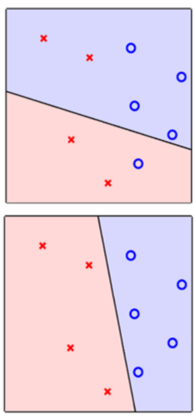
$$if: \mathbf{w}^T=(3.2 \quad 1.5 \quad 1.3 \quad 2.1 \quad 0.8)$$

$$\hat{y}=h(\mathbf{x})=sign(\mathbf{w}^T\mathbf{x})=sign((3.2 \quad 0.5 \quad 1.3 \quad 2.1 \quad 0.8)\begin{pmatrix}1\\126\\180\\62\\26\end{pmatrix})$$
$$=sign(451.2)=1$$

2.1 感知器模型参数空间

在二维空间观察感知器的分类面(*候选的*) $\mathcal{H}(\cdot)$

$h(\mathbf{x}) = sign(w_0 + w_1x_1 + w_2x_2)$



样本 $\mathbf{x}$ ➡ 用平面(或者 $\mathbf{R}^d$ 超平面)上的点表示

标签 y ➡ ○ (+1), × (-1)

$h(\mathbf{x})$ ➡ 平面上的线(或者 $\mathbf{R}^d$ 上的超平面)

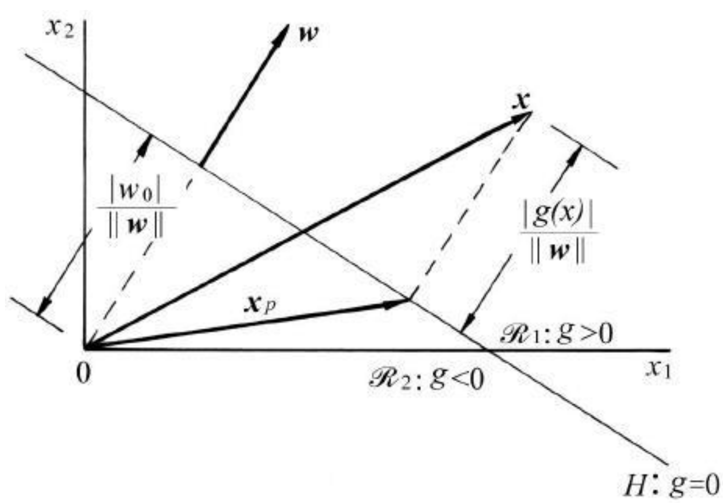
平面被分成两个区域，分别表示+1类和-1类所在的区域

不同的线有可能将同一样本分到不同的类别中去

感知器也被称作二元线性分类器(*binary linear classifiers*)

2.1 感知器模型参数空间

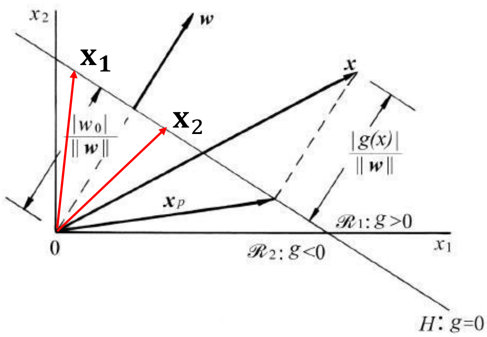
用向量几何知识来分析感知器模型



2.1 感知器模型参数空间

用向量几何知识来分析感知器模型

假定 $\mathbf{x}_1$ 和 $\mathbf{x}_2$ 在分类面上:



$$g(\mathbf{x}_1) = \boldsymbol{w}^T\mathbf{x}_1 + w_0 = 0$$

$$g(\mathbf{x}_2) = \boldsymbol{w}^T\mathbf{x}_2 + w_0 = 0$$

$$\boldsymbol{w}^T(\mathbf{x}_1 - \mathbf{x}_2) = 0$$

$$\boldsymbol{w} \perp (\mathbf{x}_1 - \mathbf{x}_2)$$

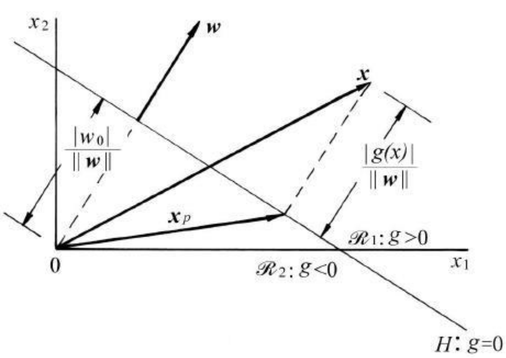
权向量垂直于分类面

2.1 感知器模型参数空间

用向量几何知识来分析感知器模型

如果 $\mathbf{x} \in D$ 在特征平面上， $\mathbf{x}_p$ 在分类面上

r 表示 $\mathbf{x}$ 与分类面 $\left(g(\mathbf{x}_p) = 0\right)$ 之间的距离



$$\mathbf{x} = \mathbf{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}$$

$$\begin{aligned} g(\mathbf{x}) &= \boldsymbol{w}^T\mathbf{x} + w_0 \\ &= \boldsymbol{w}^T\left(\mathbf{x}_p + r\frac{\boldsymbol{w}}{\|\boldsymbol{w}\|}\right) + w_0 \\ &= \boldsymbol{w}^T\mathbf{x}_p + w_0 + r\frac{\boldsymbol{w}^T\boldsymbol{w}}{\|\boldsymbol{w}\|} \\ &= r\|\boldsymbol{w}\| \end{aligned}$$

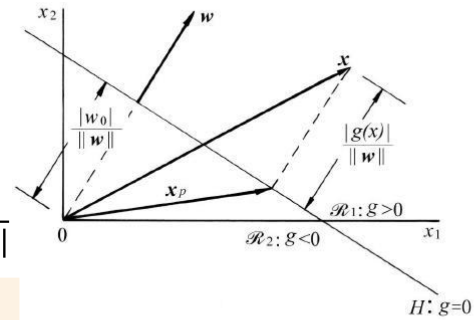
2. 1 感知器模型参数空间

用向量几何知识来分析感知器模型

$$r = \frac{g(\mathbf{x})}{\|\boldsymbol{w}\|}$$

当 $\mathbf{x} = \mathbf{0}$， $g(\mathbf{x}) = w_0$ ⇨ $r = \dfrac{w_0}{\|\mathbf{w}\|}$

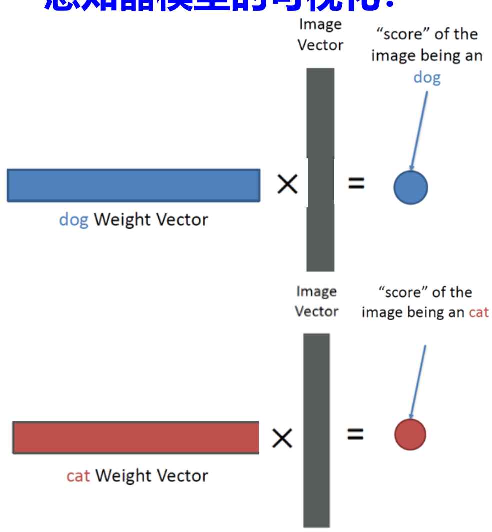
即：原点到分类面的距离为： $r = \dfrac{w_0}{\|\boldsymbol{w}\|}$



- 当 $w_0 > 0$，原点处于分类面的正区域
- 当 $w_0 < 0$，原点处于分类面的负区域
- 当 $w_0 = 0$，分类面穿过原点

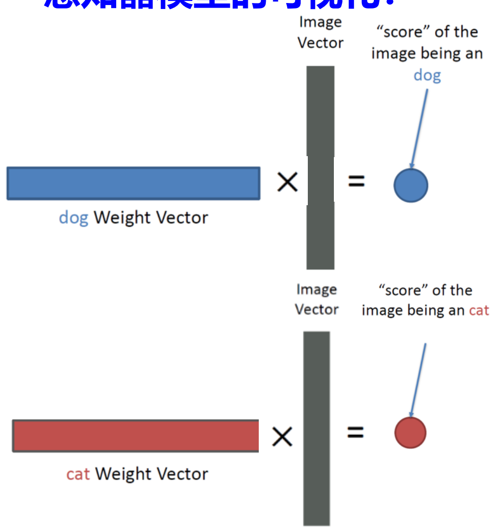
2. 1 感知器模型参数空间

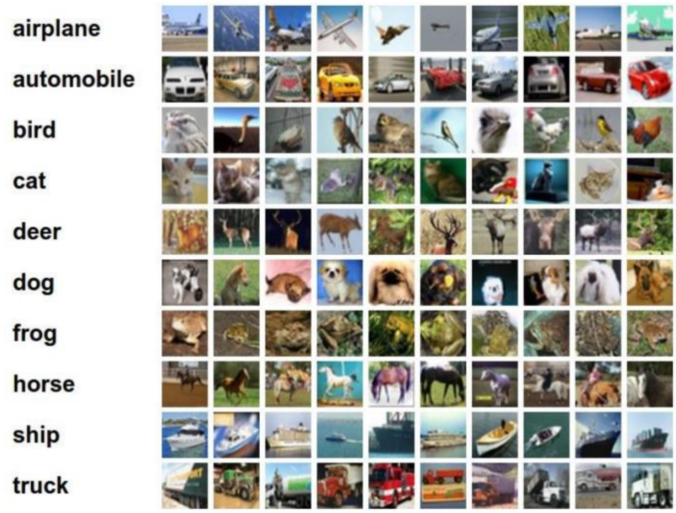
感知器模型的可视化：



2.1 感知器模型参数空间

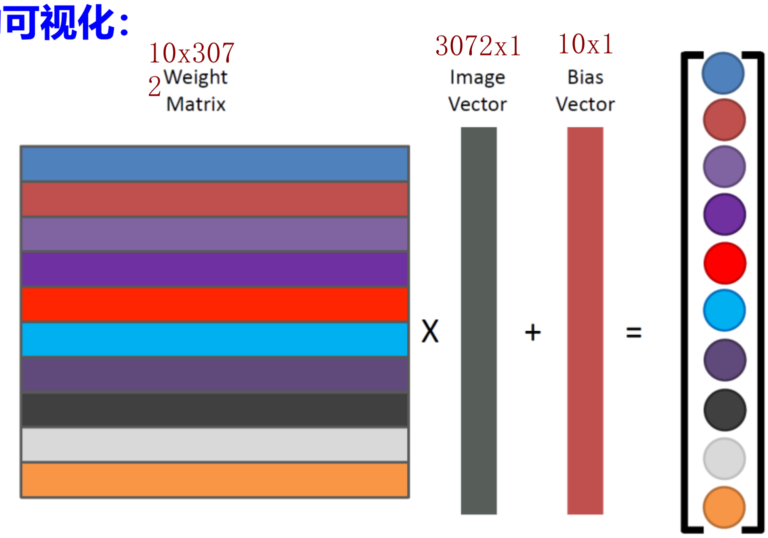
感知器模型的可视化：





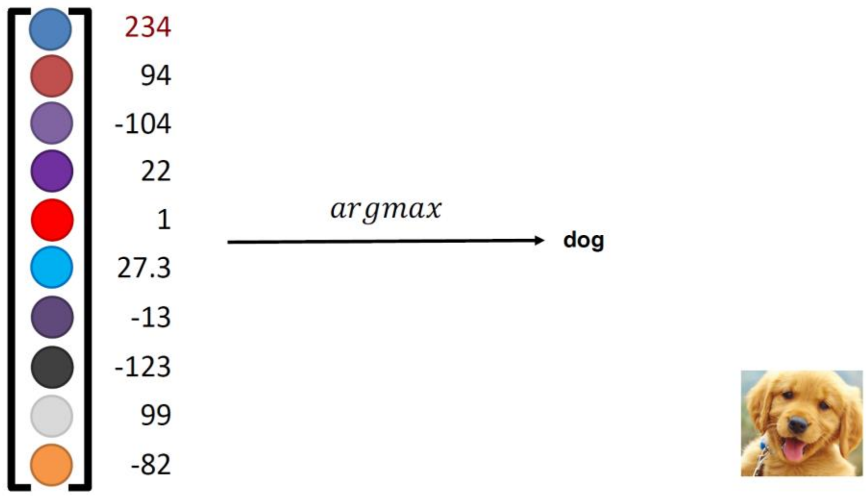
2.1 感知器模型参数空间

感知器模型的可视化：



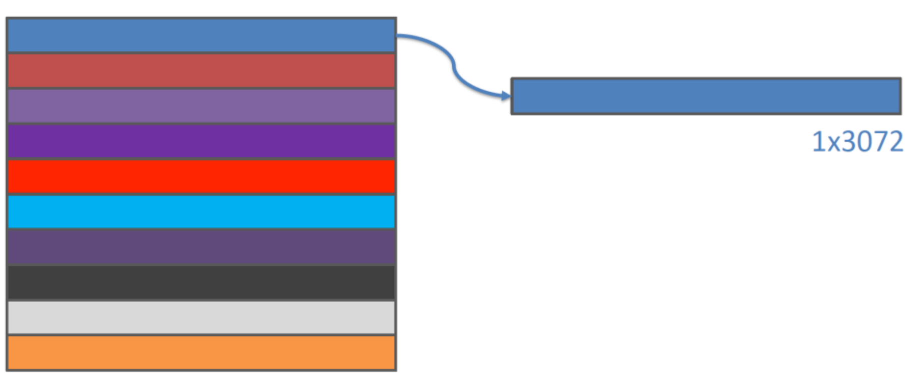
2.1 感知器模型参数空间

感知器模型的可视化：



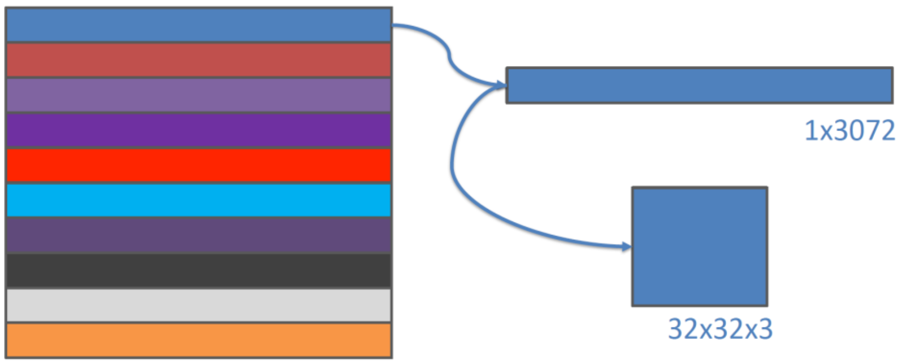
2.1 感知器模型参数空间

感知器模型的可视化：



2.1 感知器模型参数空间

感知器模型的可视化：

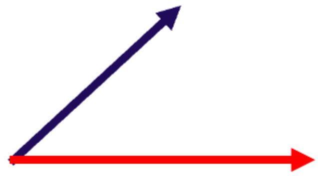


2.1 感知器模型参数空间

感知器模型的可视化：

2.1 感知器模型参数空间

感知器模型的可视化：



$$\mathbf{W}^T\mathbf{X}=\|\mathbf{W}\|\|\mathbf{X}\|cos\theta$$

第二讲　感知器（*Perceptron for Pattern Recognition*）

2.1 感知器模型参数空间　（*Perceptron Hypothesis Set*）

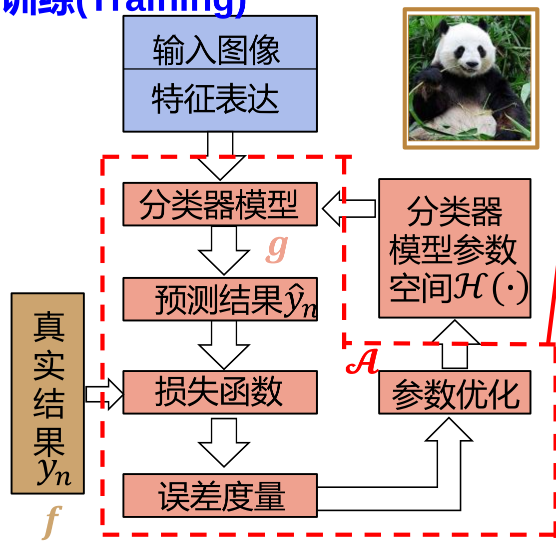
2.2 感知器算法　（*Perceptron Learning Algorithm: PLA*）

2.3 感知器算法的收敛性　（*Guarantee of PLA*）

2.4 线性不可分情况　（Non-separable Data）

2.2 感知器算法（*PLA*）

训练(Training)

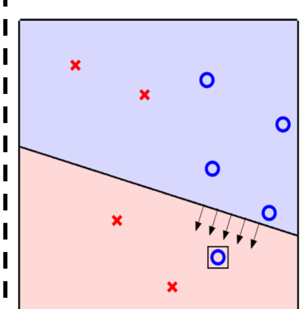


算法$\mathcal{A}$的目的是在$\mathcal{H}(h(\cdot))$中找到最优结果作为分类器的模型 g

- 最优结果： $g \approx f$
- 挑战： f 未知
- 学习的资源：在训练集$\mathcal{D}$上，如果每一个样本都有：$g(\mathbf{x}_n) = y_n = f(\mathbf{x}_n)$，则在训练集$\mathcal{D}$上做到了$g \approx f$
- 困难：$\mathcal{H}(h(\cdot))$ 的候选模型无穷多

2.2 感知器算法（*PLA*）

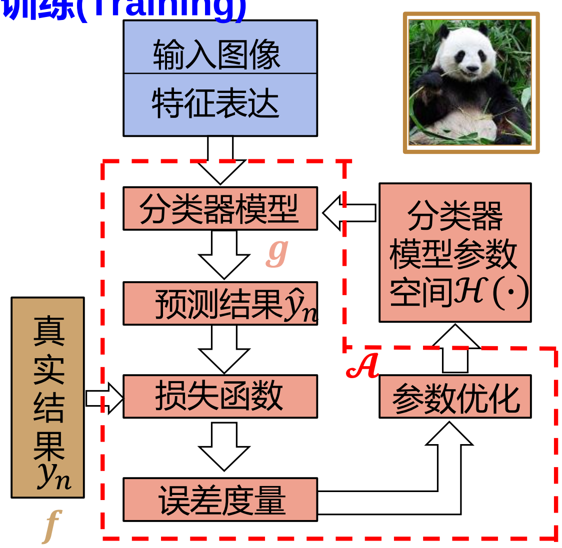
算法思路



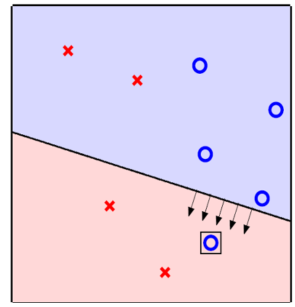
- 设置初始分类面(权重)$\boldsymbol{w}_0$
- 如果有样本分错，就修正权重

2.2 感知器算法（*PLA*）

训练(Training)



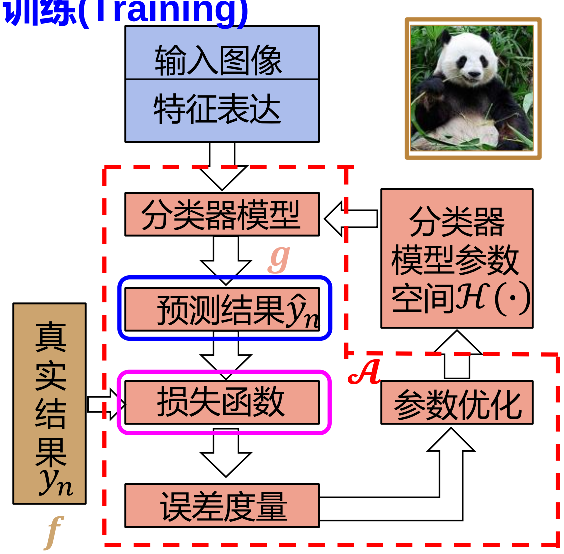
算法思路



- 设置初始分类面(权重)$\boldsymbol{w}_0$
- 如果有样本分错，就修正权重

2.2 感知器算法（*PLA*）

训练(Training)

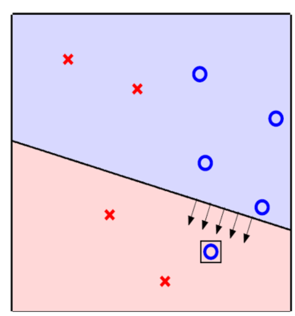


$$\hat{y}_{n(t)} = \mathrm{sign}(\boldsymbol{w}_t^T \mathbf{x}_{n(t)})$$

$$L_{in} = [\![y_n \neq \hat{y}_{n(t)}]\!]$$

L_{in}训练阶段

算法思路



- 设置初始分类面(权重)$\boldsymbol{w}_0$
- 如果有样本分错，就修正权重

2.2 感知器算法（*PLA*）

训练(Training)

输入图像

特征表达

分类器模型

g

预测结果$\hat{y}_n$

损失函数

误差度量

分类器模型参数空间$\mathcal{H}(\cdot)$

$\mathcal{A}$

参数优化

真实结果y_n

f

$\hat{y}_{n(t)} = \mathrm{sign}(\boldsymbol{w}_t^T \mathbf{x}_{n(t)})$

$L_{out} = [\![y_n \neq \hat{y}_{n(t)}]\!]$

L_{out}测试阶段

测试(Testing)

输入图像

特征表达

分类器模型

预测结果 $\hat{y} = 1$

损失函数

2.2 感知器算法（*PLA*）

训练(Training)

输入图像

特征表达

分类器模型

g

预测结果$\hat{y}_n$

损失函数

误差度量

分类器模型参数空间$\mathcal{H}(\cdot)$

$\mathcal{A}$

参数优化

真实结果y_n

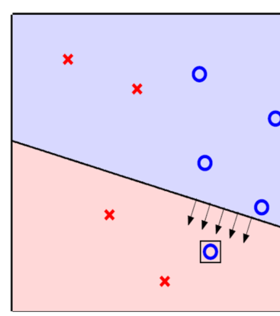
f

$\hat{y}_{n(t)} = \mathrm{sign}(\boldsymbol{w}_t^T \mathbf{x}_{n(t)})$

$L_{in} = [\![y_n \neq \hat{y}_{n(t)}]\!]$

$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_n \mathbf{x}_{n(t)}$

算法思路



- 设置初始分类面(权重)$\boldsymbol{w}_0$
- 如果有样本分错，就修正权重

2.2 感知器算法（*PLA*）

- 对样本的特征向量x和权向量$\boldsymbol{w}$ 增广化
- 初始化权向量$\boldsymbol{w}_0$ (例如： $\boldsymbol{w}_0 = \mathbf{0}$)
- $\boldsymbol{for}\ t = 0,1,2,\ldots$ （t 代表迭代次数)

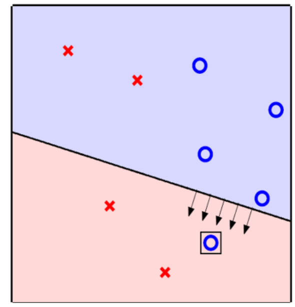
  ① 进行到第t 次迭代时权向量为$\boldsymbol{w}_t$，它对样本$(\mathbf{x}_{n(t)}, y_{n(t)})$错分

$$\mathrm{sign}(\boldsymbol{w}_t^T \mathbf{x}_{n(t)}) \neq y_n$$

  ② 通过下式对权向量$\boldsymbol{w}_t$进行更新： $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_n \mathbf{x}_{n(t)}$

…直到所有样本均能被正确分类，此时的$\boldsymbol{w}_{t+1}$作为学到的g

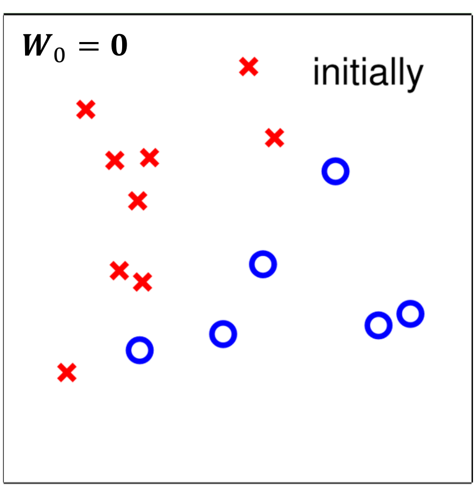
算法思路

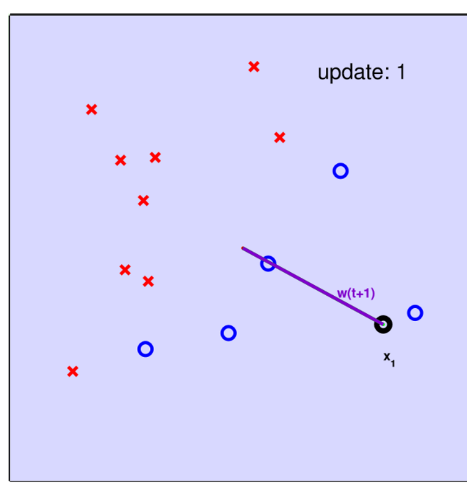


- 设置初始分类面(权重)$\boldsymbol{w}_0$
- 如果有样本分错，就修正权重

2.2 感知器算法（*PLA*）

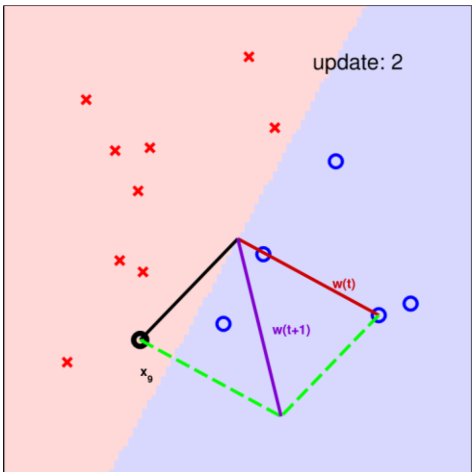
算法迭代示例

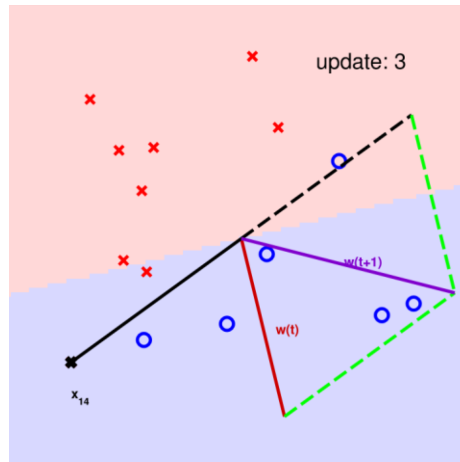




2.2 感知器算法（*PLA*）

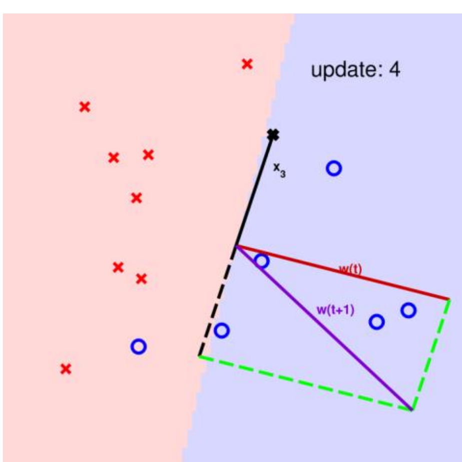
算法迭代示例

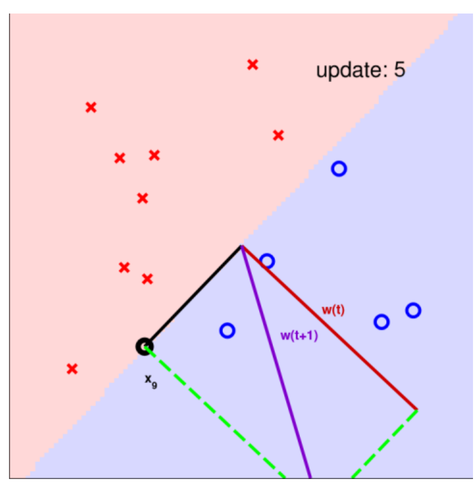




2.2 感知器算法（*PLA*）

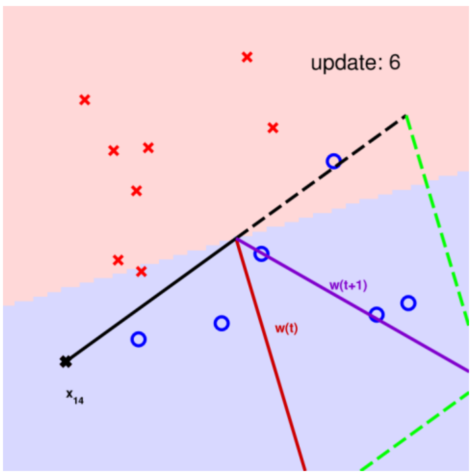
算法迭代示例

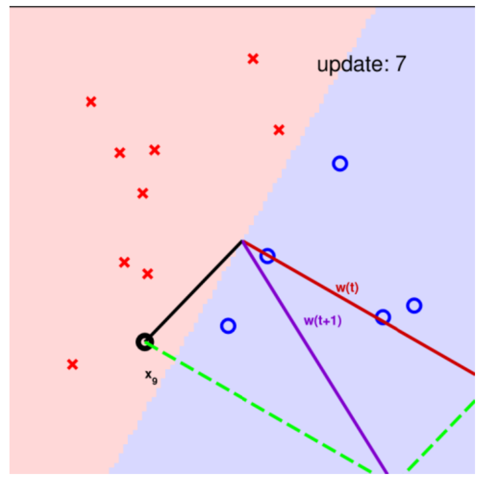




2.2 感知器算法（*PLA*）

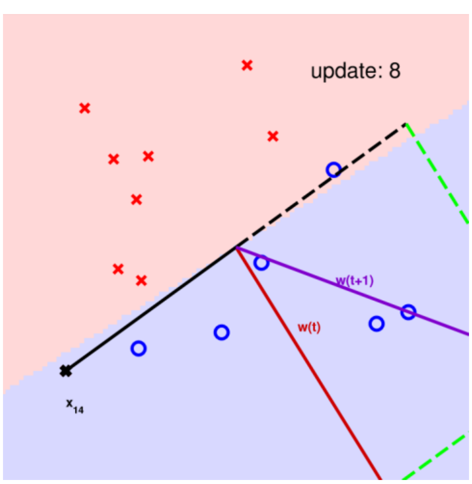
算法迭代示例

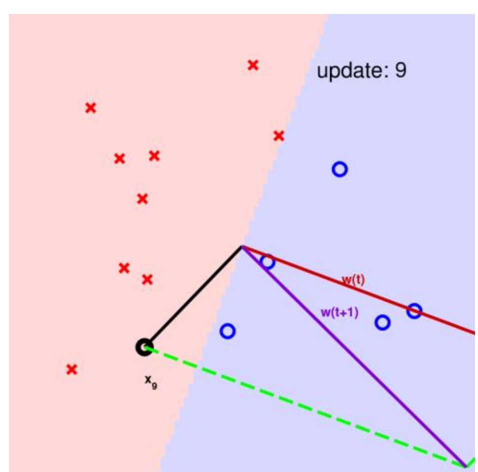




2.2 感知器算法（*PLA*）

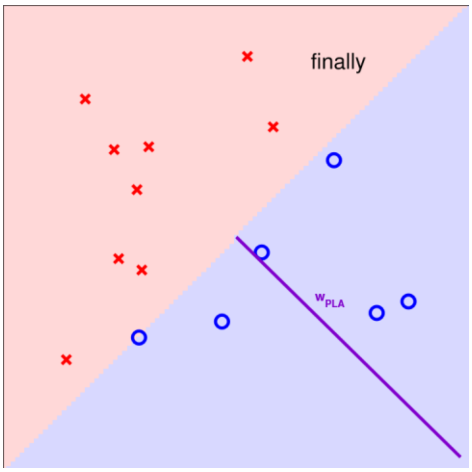
算法迭代示例





2.2 感知器算法（*PLA*）

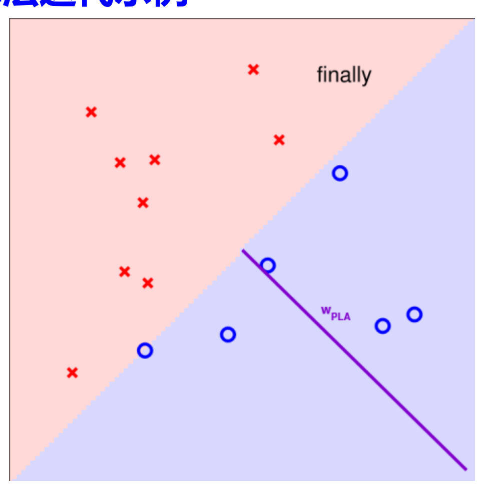
算法迭代示例



训练样本集中所有样本都分类正确，算法停止

2.2 感知器算法（*PLA*）

算法迭代示例



训练样本集中所有样本都分类正确，算法停止

问题1：算法会收敛吗?

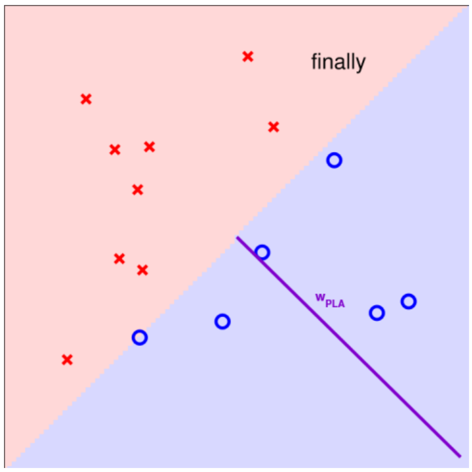
结果与输入样本顺序是否有关?

问题2：能学到 $g \approx f$ 吗?

- *在$\mathcal{D}$上，如果$L_{in}=0$，$g \approx f$?*
- *不在$\mathcal{D}$上时，$L_{out}=0$? $g \approx f$?*
- *算法不能收敛时，$g \approx f$?*

2.2 感知器算法（*PLA*）

算法迭代示例



训练样本集中所有样本都分类正确，算法停止

问题1：算法会收敛吗?

结果与输入样本顺序是否有关?

问题2：能学到 $g \approx f$ 吗?

- *在 $\mathcal{D}$ 上，如果 $L_{in} = 0$， $g \approx f$?*
- *不在 $\mathcal{D}$ 上时， $L_{out} = 0$? $g \approx f$?*
- *算法不能收敛时， $g \approx f$?*

能否证明在满足什么样的条件下，感知器算法经过足够次迭代，一定会收敛

第二讲　感知器 （*Perceptron for Pattern Recognition*）

2.3 感知器算法的收敛性（*Guarantee of PLA*）

线性可分性 (*linear separable*)

*PLA*算法收敛 ⟷ 算法停止 ⟷ $\boldsymbol{w}$ 对 $\mathcal{D}$ 上所有样本正确分类

⬇

$\mathcal{D}$ 上所有样本是线性可分的

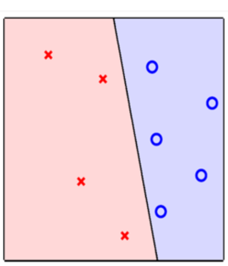
2.3 感知器算法的收敛性（*Guarantee of PLA*）

线性可分性 (*linear separable*)

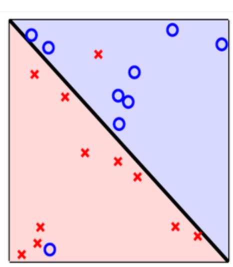
*PLA*算法收敛 ⟷ 算法停止 ⟷ $\boldsymbol{w}$ 对 $\mathcal{D}$ 上所有样本正确分类

√ ⬇

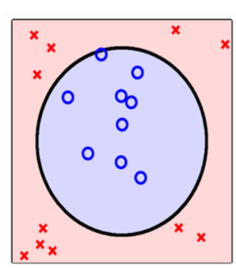
$\mathcal{D}$ 上所有样本是线性可分的



线性可分



线性不可分



线性不可分

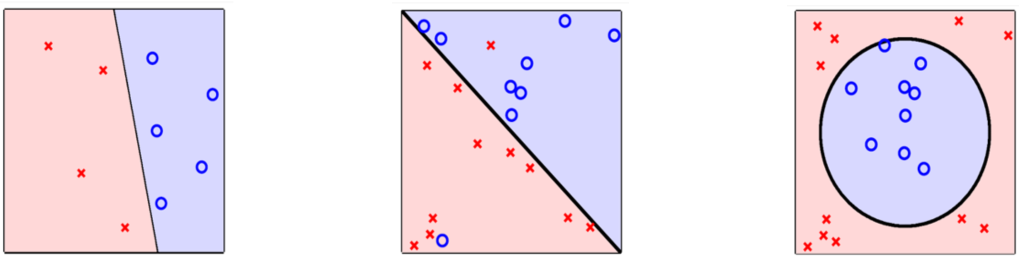
2.3 感知器算法的收敛性（*Guarantee of PLA*）

线性可分性 (*linear separable*)

$\boxed{PLA\text{算法收敛} \Longleftrightarrow \text{算法停止} \Longleftrightarrow \boldsymbol{w} \text{ 对 } \mathcal{D} \text{ 上所有样本正确分类}}$

√ ⬇ ⬆ ?

$\mathcal{D}$ 上所有样本是线性可分的



线性可分　　　　线性不可分　　　　线性不可分

2.3 感知器算法的收敛性（*Guarantee of PLA*）

假设 $\boldsymbol{w}_f$ 是理想的分类面：

$\boxed{\mathcal{D}\text{ 是线性可分的} \Longleftrightarrow y_n = sign(\boldsymbol{w}_f^T \mathbf{x}_n)}$

⬇ 第t次迭代时，任意一个样本$\mathbf{x}_{n(t)}$满足

$$y_{n(t)} \boldsymbol{w}_f^T \mathbf{x}_{n(t)} \geq \min_n y_n \boldsymbol{w}_f^T \mathbf{x}_n > 0$$

⬇ 迭代次数增加

$$\boldsymbol{w}_f^T \boldsymbol{w}_{t+1} = \boldsymbol{w}_f^T \left(\boldsymbol{w}_t + y_{n(t)} \mathbf{x}_{n(t)}\right) \geq \boldsymbol{w}_f^T \boldsymbol{w}_t + \min_n y_n \boldsymbol{w}_f^T \mathbf{x}_n > \boldsymbol{w}_f^T \boldsymbol{w}_t + 0$$

结论：随着迭代次数增加， $\boldsymbol{w}_f^T \boldsymbol{w}_t$随之增加，意味着$\boldsymbol{w}_t$与 $\boldsymbol{w}_f$越来越接近

2.3 感知器算法的收敛性（*Guarantee of PLA*）

假设 $\boldsymbol{w}_t$ 是第 t 次迭代得到的分类面：

当 $\mathbf{x}_n$被分类错误时，才更新 $\boldsymbol{w}_t \Longleftrightarrow sign(\boldsymbol{w}_t^T\mathbf{x}_n) \neq y_n \Longleftrightarrow y_{n(t)}\boldsymbol{w}_t^T\mathbf{x}_{n(t)} \leq 0$

假设 $\mathbf{x}_n$在样本集中模值最大，随着迭代次数增加，$\|\boldsymbol{w}_t\|$?

$$
\begin{aligned}
\|\boldsymbol{w}_{t+1}\|^2 &= \left\|\boldsymbol{w}_t + y_{n(t)}\mathbf{x}_{n(t)}\right\|^2 \\
&= \|\boldsymbol{w}_t\|^2 + 2y_{n(t)}\boldsymbol{w}_t^T\mathbf{x}_{n(t)} + \left\|y_{n(t)}\mathbf{x}_{n(t)}\right\|^2 \\
&\leq \|\boldsymbol{w}_t\|^2 + 0 + \left\|y_{n(t)}\mathbf{x}_{n(t)}\right\|^2 \leq \|\boldsymbol{w}_t\|^2 + \max_n\|y_n\mathbf{x}_n\|^2
\end{aligned}
$$

结论：随着迭代次数增加， $\boldsymbol{w}_t$模值增长不会太快， 意味着$\boldsymbol{w}_t$与 $\boldsymbol{w}_f$的接近是方向上在靠近，而非模值的贡献

2.3 感知器算法的收敛性（*Guarantee of PLA*）

课后证明（作业）：

(1) 针对线性可分训练样本集，*PLA*算法中，当$\boldsymbol{w}_0 = \mathbf{0}$，在对分错样本进行了$T$次修正后，下式成立:$\frac{\mathbf{w}_f^T}{\|\mathbf{w}_f\|}\frac{\mathbf{w}_T}{\|\mathbf{w}_T\|} \geq \sqrt{T} \cdot constant$

(2) 针对线性可分训练样本集，*PLA*算法中，假设对分错样本进行了T次修正后，得到的分类面不再出现错分状况，定义：

$R^2 = \max_n\|\mathbf{x}_n\|^2$ ，$\rho = \min_n y_n\frac{\mathbf{w}_f^T}{\|\mathbf{w}_f\|}\mathbf{x}_n$，证明：$\mathrm{T} \leq \frac{R^2}{\rho^2}$

第二讲　感知器（*Perceptron for Pattern Recognition*）

2.4 线性不可分情况（*Non-separable Data*）

算法收敛性保证：

$\mathcal{D}$ 是线性可分的

↓

$\mathbf{w}_f^T\mathbf{w}_t$随$t$增加而增加

$\mathbf{x}_n$被分类错误时才更新 $\mathbf{w}_t$

↓

$\mathbf{w}_t$模值增长不会太快

算法学习的分类面与 $\mathbf{w}_f$越来越接近

↓

算法收敛

算法的长处：

- 实现简单
- 运行快速
- 适用于任意维数d

算法的不足：

- $\mathcal{D}$ 线性可分，算法收敛
  --是否线性可分，事先未知
- $\mathrm{T} \le \dfrac{\max\limits_n \|\mathbf{x}_n\|^2}{\left(\min\limits_n y_n \dfrac{\mathbf{w}_f^T}{\|\mathbf{w}_f\|}\mathbf{x}_n\right)^2}$，T有界
  --$\mathbf{W}_f$未知，实际无法得到T

如果 $\mathcal{D}$ 是线性不可分的，算法不收敛，如何处理？

2.4 线性不可分情况（*Non-separable Data*）

输入图像

特征表达

分类器模型

g

预测结果 $\hat{y}_n$

损失函数

误差度量

真实结果 y_n

f

分类器模型参数空间$\mathcal{H}(\cdot)$

$\mathcal{A}$

参数优化

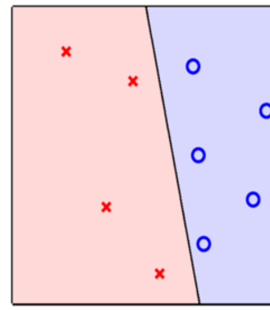
$$\hat{y}_{n(t)} = \mathrm{sign}(\boldsymbol{w}_t^T \mathbf{x}_{n(t)})$$

算法收敛：

$$L_{in} = \sum_{n=1}^{N} [\![y_n \neq \hat{y}_n]\!] = 0$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_n \mathbf{x}_{n(t)}$$

线性可分



- 设置初始分类面(权重)$\boldsymbol{w}_0$
- 如果有样本分错，就修正权重

2.4 线性不可分情况（*Non-separable Data*）

输入图像

特征表达

分类器模型

g

预测结果 $\hat{y}_n$

损失函数

误差度量

真实结果 y_n

f

分类器模型参数空间$\mathcal{H}(\cdot)$

$\mathcal{A}$

参数优化

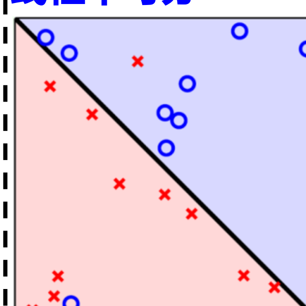
$$\hat{y}_{n(t)} = \mathrm{sign}(\boldsymbol{w}_t^T \mathbf{x}_{n(t)})$$

算法停止条件：

$$L_{in} = \underset{\boldsymbol{w}}{argmin} \sum_{n=1}^{N} [\![y_n \neq \hat{y}_n]\!]$$

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_n \mathbf{x}_{n(t)}$$

线性不可分



- NP难问题
- 求解相对最优解

2.4 线性不可分情况（*Non-separable Data*）

Pocket 算法 – *为处理线性不可分情况而对PLA算法的修正*

- 对样本的特征向量x和权向量$\boldsymbol{w}$ 增广化
- 初始化权向量$\boldsymbol{w}_0$ (例如： $\boldsymbol{w}_0 = \mathbf{0}$)，并任意选一个“Pocket”向量$\hat{\boldsymbol{w}}$
- $\boldsymbol{for}\ t = 0,1,2,\ldots$ (t 代表迭代次数)

  ① 进行到第t 次迭代时权向量为$\boldsymbol{w}_t$，它对样本$(\mathbf{x}_{n(t)}, y_{n(t)})$错分

$$\mathrm{sign}(\boldsymbol{w}_t^T \mathbf{x}_{n(t)}) \neq y_n$$

  ② 通过下式对权向量$\boldsymbol{w}_t$进行更新： $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t + y_n \mathbf{x}_{n(t)}$

  ③ 如果 $\boldsymbol{w}_{t+1}$在所有样本集上错分的样本少于$\hat{\boldsymbol{w}}$，则用$\boldsymbol{w}_{t+1}$代替$\hat{\boldsymbol{w}}$，并在错分样本中随机选一个对权向量进行更新

…达到指定的迭代次数

返回此时的 “Pocket”向量$\hat{\boldsymbol{w}}$ 作为算法学到的g

第二讲　感知器（*Perceptron for Pattern Recognition*）

2.1 感知器模型参数空间

在$\mathbf{R}^d$空间的超平面的线性分类面

2.2 感知器算法（*PLA*）

通过迭代的方式对错分样本的分类面进行修正

2.3 感知器算法的收敛性

如果训练样本集是线性可分的，算法能对所有的样本正确分类并停止

2.4 线性不可分情况

*通过“**Pocket**”算法在设置的迭代次数下寻找相对最佳的分类面*