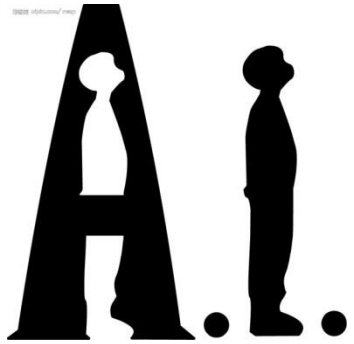
机器学习

讲师：曾江峰

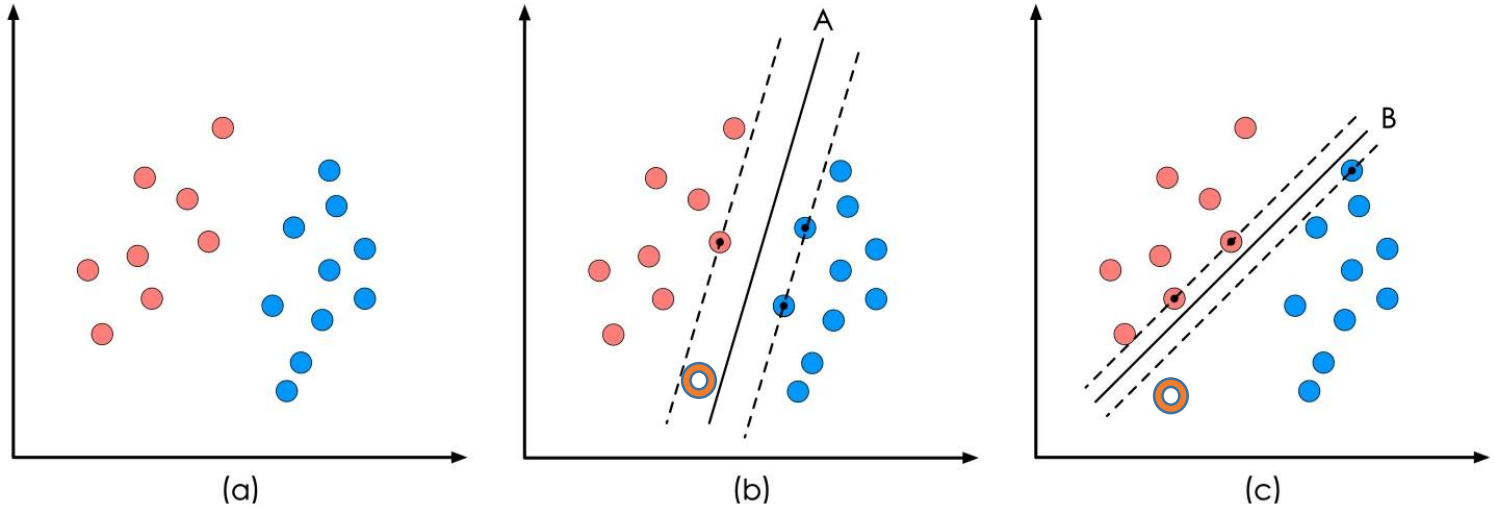
E-mail：jfzeng@ccnu.edu.cn



支持向量机

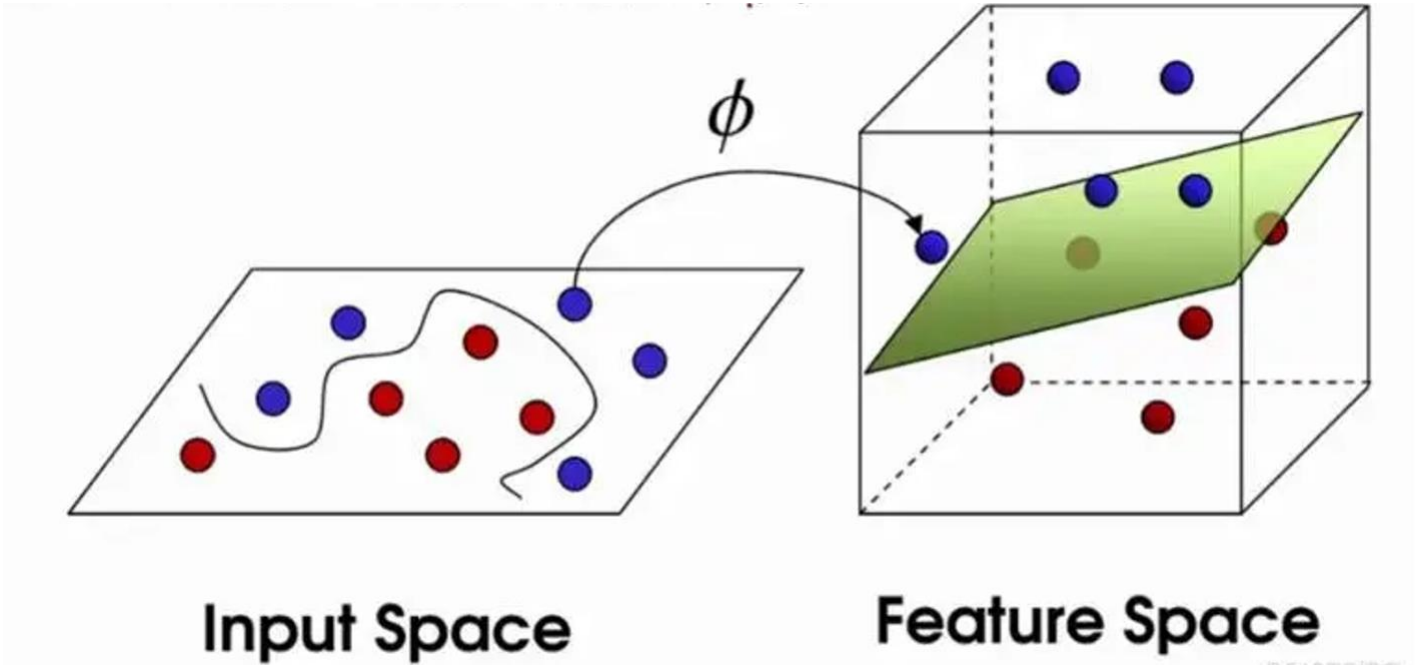
讲个故事——武林高手的思维

- （1）有规律放了两种颜色的球，用一根棍分开它们。要求：尽量保证在放更多新球之后，这个棍子的摆放位置仍然适用。



讲个故事——武林高手的思维

- （2）无规律放了两种颜色的球，不给棍子。



做个类比

- 球 → data
- 棍子（低维空间）、纸（高维空间） → classifier
- 几十年内力的一掌 → kernel

感知机

- 感知机（Perceptron）是一个线性二分类模型，旨在建立一个线性分离超平面对线性可分的数据集进行分类。
- 1957年由Rosenblatt提出，是神经网络与支持向量机的理论基础。
- 神经网络由多层感知机（MLP）发展而来。

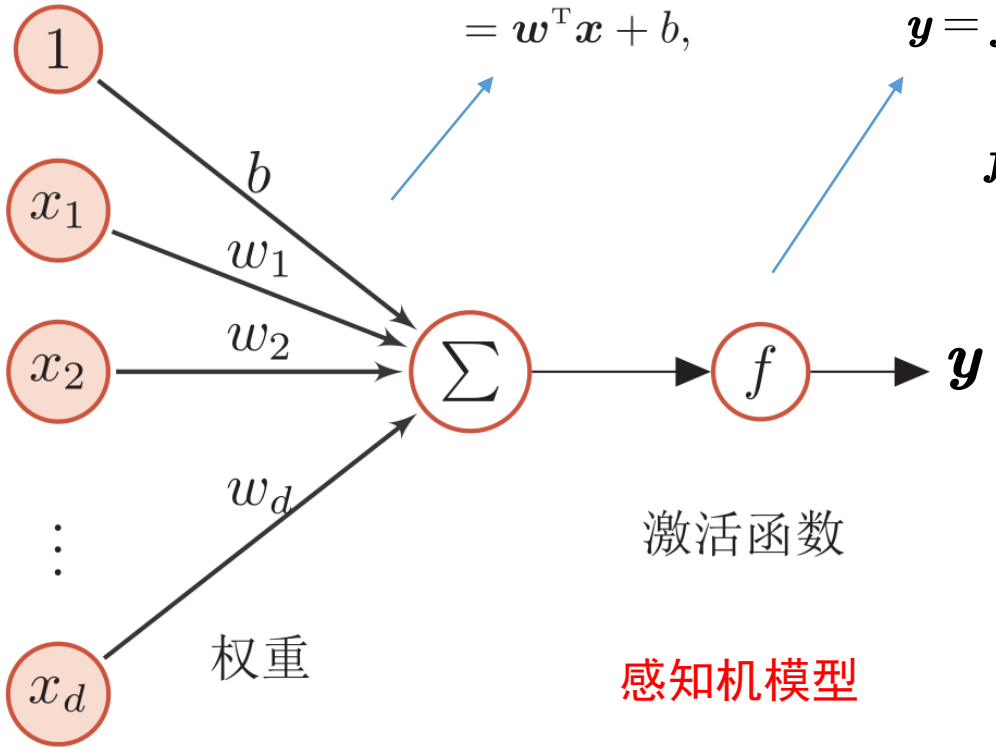
感知机模型

$$z = \sum_{i=1}^{d} w_i x_i + b$$

$$= \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x} + b,$$

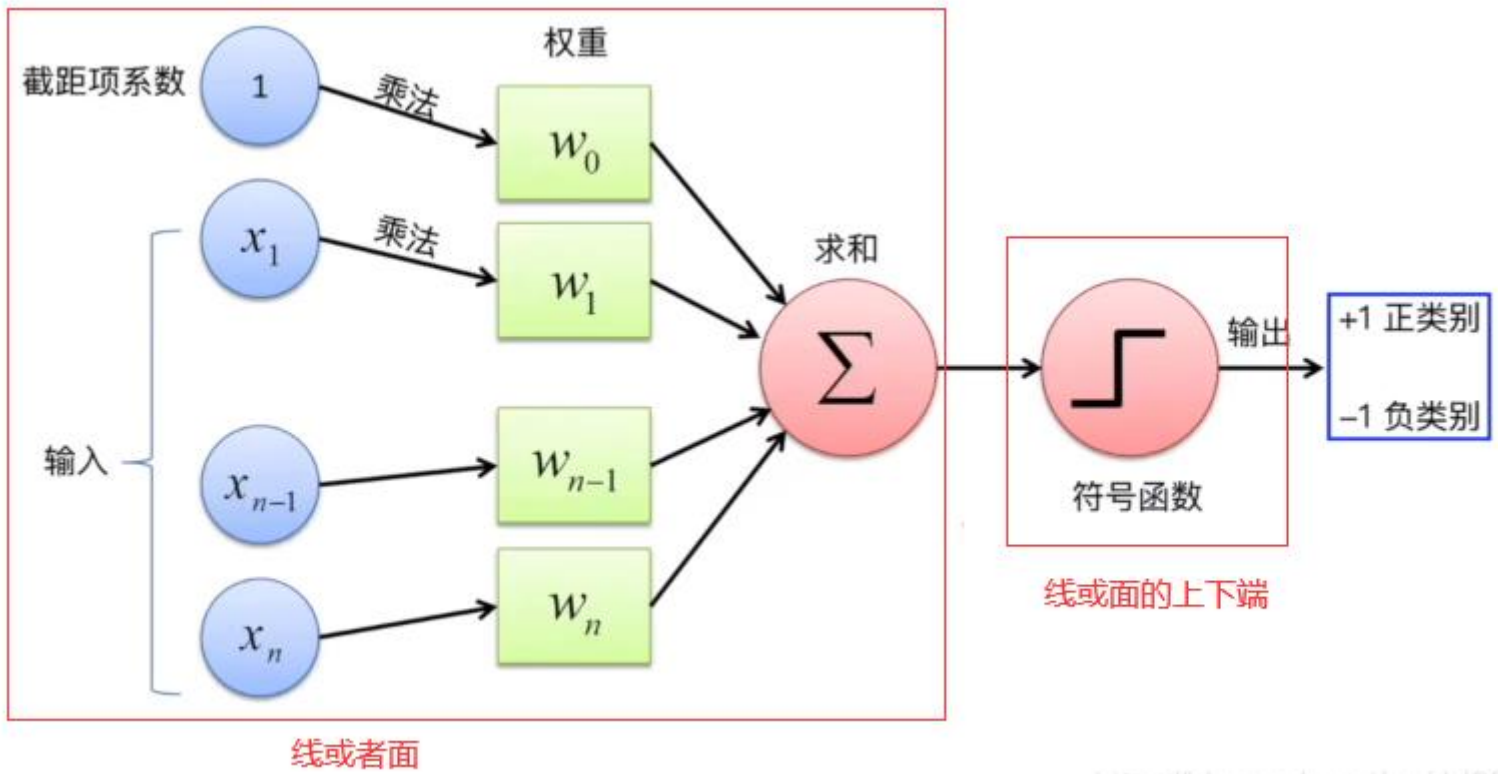
$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{z})$$

$$\boldsymbol{f}(\boldsymbol{z}) = \boldsymbol{sign}(\boldsymbol{z}) = \begin{cases} +1, & \boldsymbol{z} \geqslant 0 \\ -1, & \boldsymbol{z} < 0 \end{cases}$$



感知机模型

感知机模型



感知机模型的学习目标

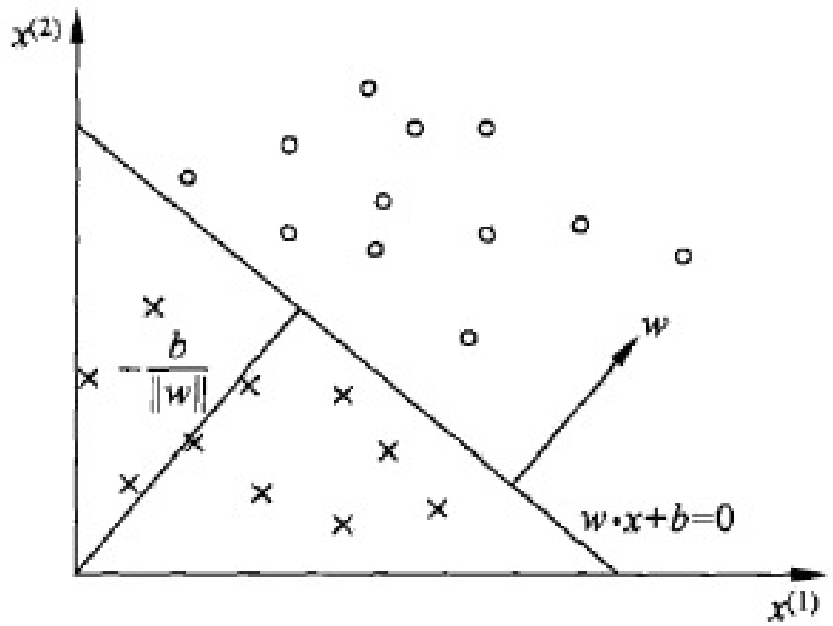
- 感知机的学习目标是建立一个线性分离超平面，以将训练数据正例和负例完全分开。
- 优化对象 → 分离超平面
- 目标函数 → 如何定义损失函数？
  - 大致思路：基于误分类的损失函数
  - 也就是说设计损失函数时，不需要考虑正确分类的样本的影响，只使用误分类的样本参与损失的计算。
  - 自然选择：误分类点的数目，但损失函数对w,b不是连续可导，不宜优化。
  - 另一选择：误分类点到超平面的总距离。

感知机模型损失函数的直观解释

- 当一个实例被误分类时，即实例位于线性分离超平面的错误一侧时，我们需要调整参数w和b，使得线性分离超平面向该误分类点的一侧移动，以缩短该误分类点与线性分离超平面的距离，指导线性分离超平面越过该误分类点使其能够被正确分类。

感知机模型的几何解释

- 线性方程：$$w^T x + b = 0$$
- 对应于超平面S，w为法向量，b截距，分隔正、负类。
- 分离超平面：



分离超平面方程的数学推导

- 从2维空间中的直线方程开始：

$$y = ax + \gamma$$

- 根据数据样本特征变量改写：

$$x_2 = ax_1 + \gamma$$

$$ax_1 + (-1) \cdot x_2 + \gamma = 0$$

$$[a, -1] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \gamma = 0$$

a 是斜率，控制着直线的方向，
γ 是截距，控制着直线的位置。

- 扩展到n维空间中的超平面方程：

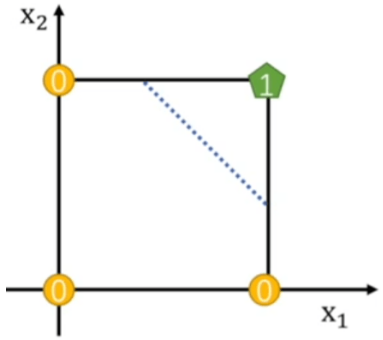
$$wx + b = 0$$

感知机模型的数学推导

- 点 x_0 到线性分离超平面的距离为：$\frac{1}{\|\boldsymbol{w}\|_2}(\boldsymbol{w}\cdot\boldsymbol{x}_0+\boldsymbol{b})$
- 对于任一误分类点（xi, yi)，其到超平面的距离为：$-\frac{1}{\|\boldsymbol{w}\|_2}\hat{\boldsymbol{y}}_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+\boldsymbol{b})$
- 假设共有M个误分类点，总距离为：$-\frac{1}{\|\boldsymbol{w}\|_2}\sum_{\boldsymbol{x}_i\in\boldsymbol{M}}\hat{\boldsymbol{y}}_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+\boldsymbol{b})$
- 感知机的损失函数：$\boldsymbol{L}(\boldsymbol{w},\boldsymbol{b})=-\sum_{\boldsymbol{x}_i\in\boldsymbol{M}}\hat{\boldsymbol{y}}_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+\boldsymbol{b})$
- 梯度下降法求解最优参数：$\frac{\partial\boldsymbol{L}}{\partial\boldsymbol{w}}=-\sum_{\boldsymbol{x}_i\in\boldsymbol{M}}\hat{\boldsymbol{y}}_i\boldsymbol{x}_i$　　$\frac{\partial\boldsymbol{L}}{\partial\boldsymbol{b}}=-\sum_{\boldsymbol{x}_i\in\boldsymbol{M}}\hat{\boldsymbol{y}}_i$

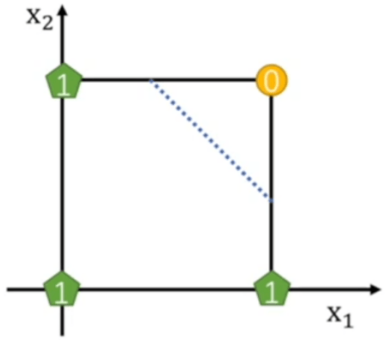
感知机不能处理非线性可分情形

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



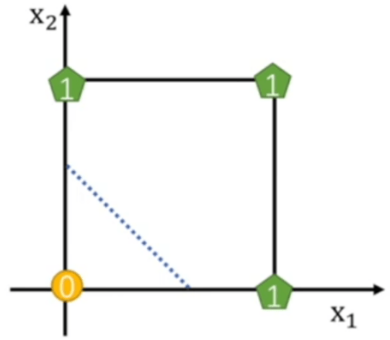
$x_1 \wedge x_2$

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0



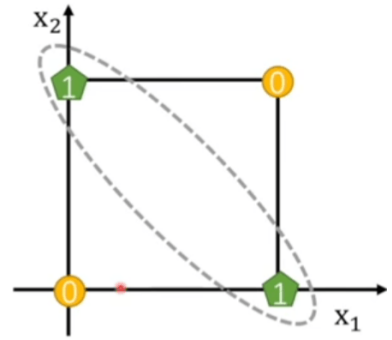
$\neg(x_1 \wedge x_2)$

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



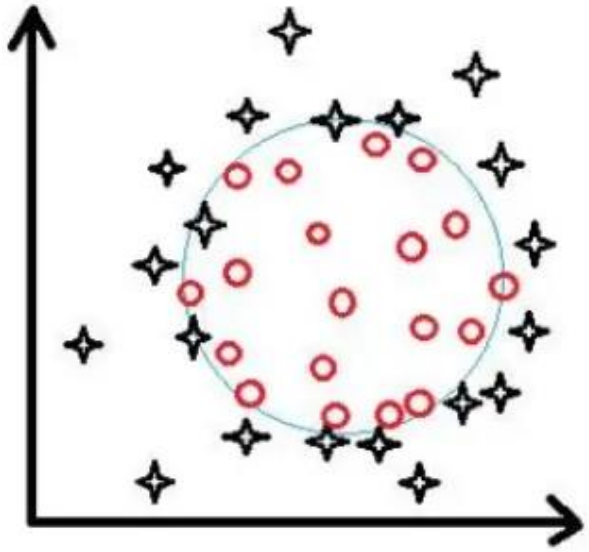
$x_1 \vee x_2$

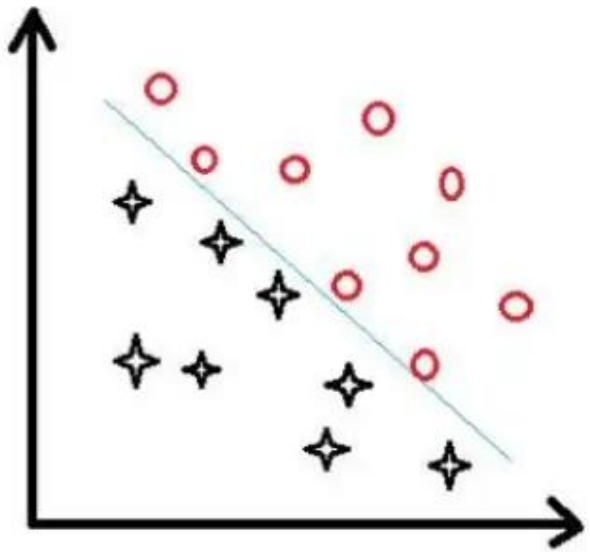
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0



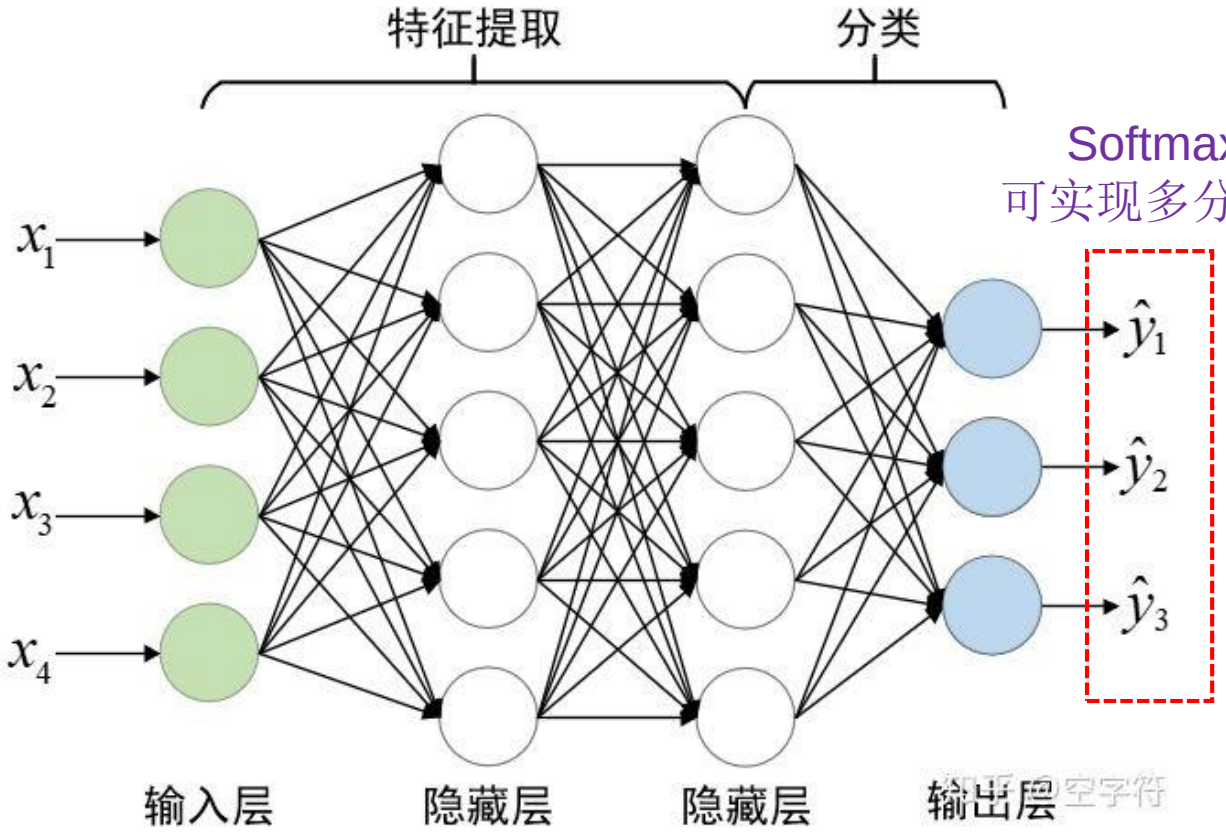
$x_1 \oplus x_2$

线性可分 **vs** 非线性可分





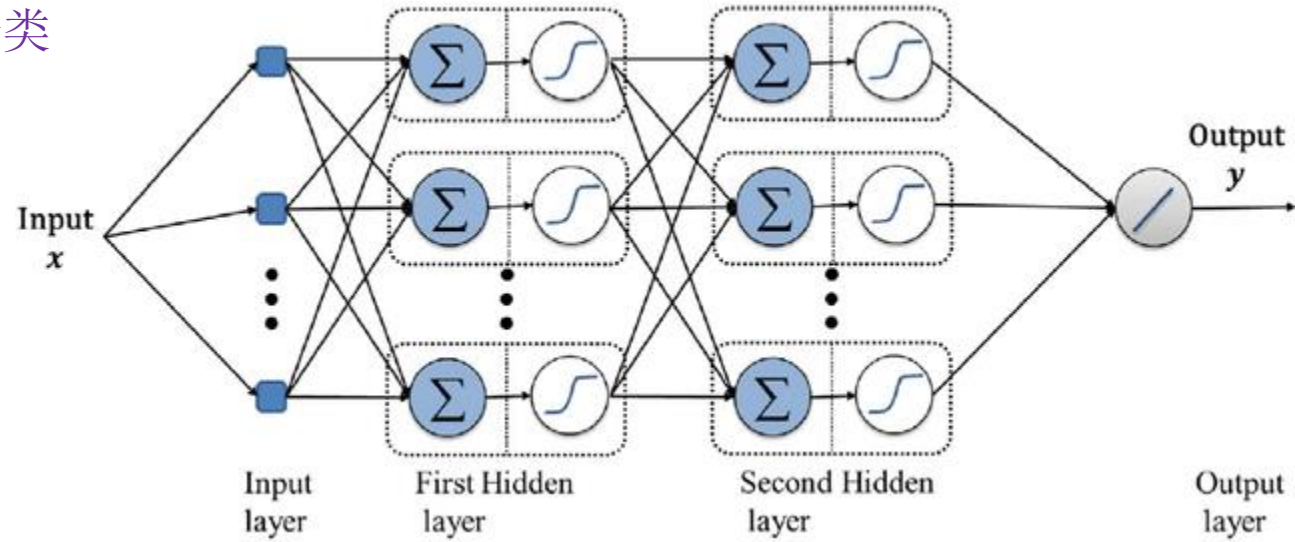
感知机 → 神经网络



Softmax
可实现多分类

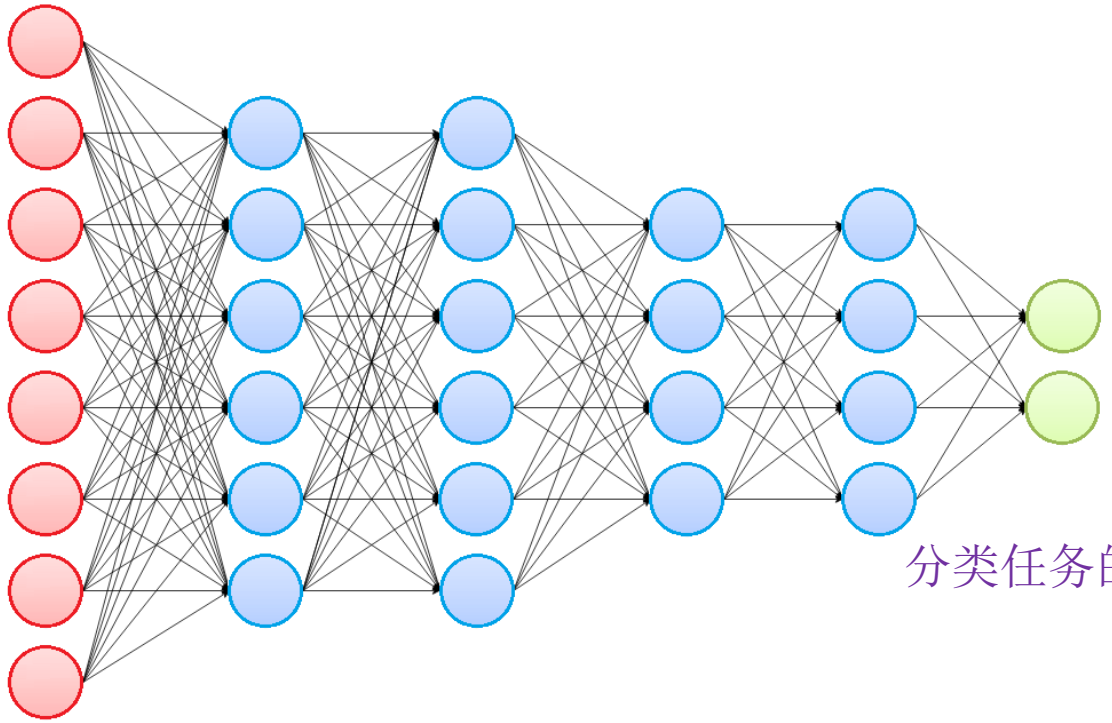
神经元之间采用全连接

Sigmoid、tanh、relu等
隐藏层的激活函数对参数连续可导



神经网络

神经网络通过对感知机添加隐藏层来实现非线性化



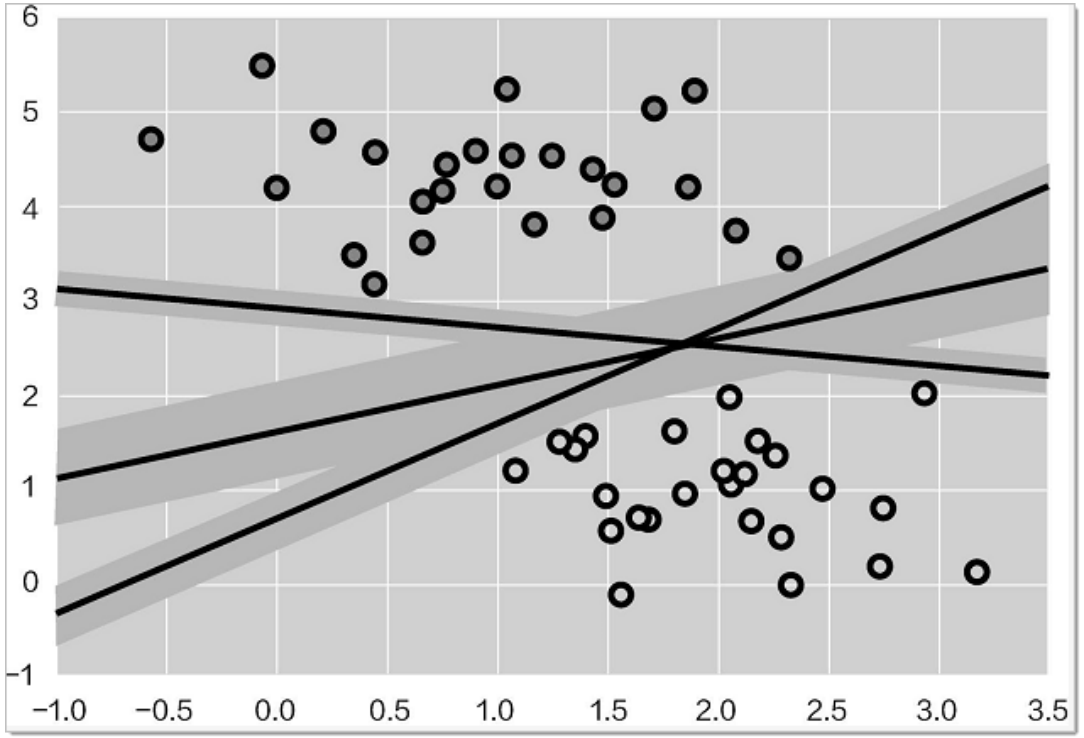
Input Layer | Hidden Layer 1 | Hidden Layer 2 | Hidden Layer 3 | Hidden Layer 4 | Output Layer

$$\boldsymbol{L} = -\ln \boldsymbol{p}(\boldsymbol{y}|\boldsymbol{x}) = -\frac{1}{\boldsymbol{n}} \sum_{\boldsymbol{i}=1}^{\boldsymbol{n}} \sum_{\boldsymbol{c}=1}^{\boldsymbol{m}} \boldsymbol{y}_{\boldsymbol{c}} \ln \hat{\boldsymbol{y}}_{\boldsymbol{c}}$$

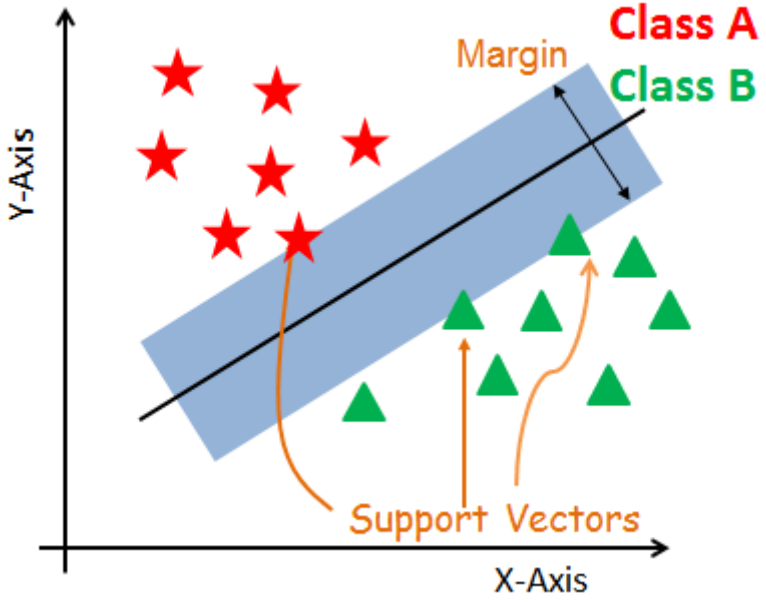
分类任务的损失函数通常采用交叉熵损失函数

感知机 → 支持向量机

- 对于线性可分情形，理论上感知机可以帮我们找到一个分离超平面实现无差错的二分类，但这样的分离超平面其实可以有很多个，哪个才是最优的？



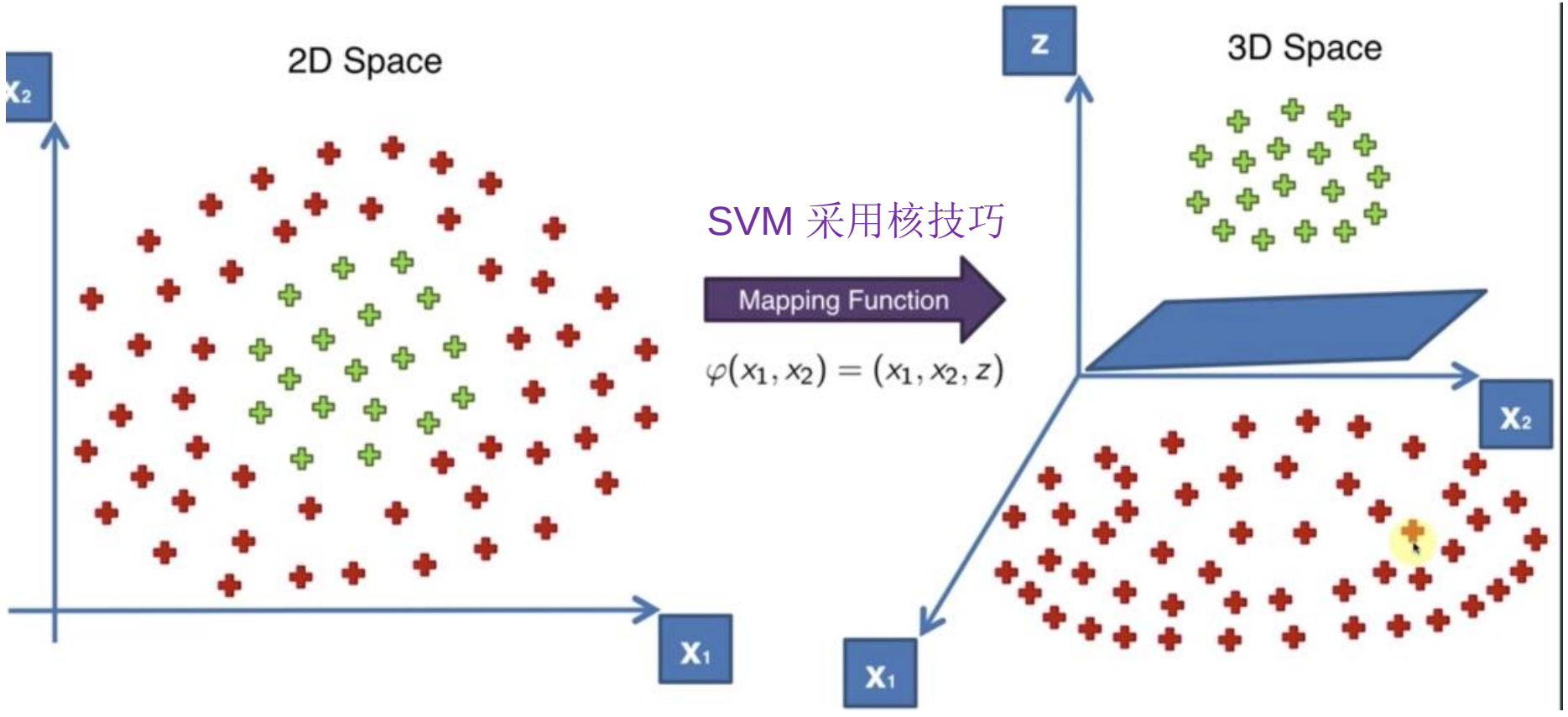
分类间隔最大化



感知机 → 支持向量机

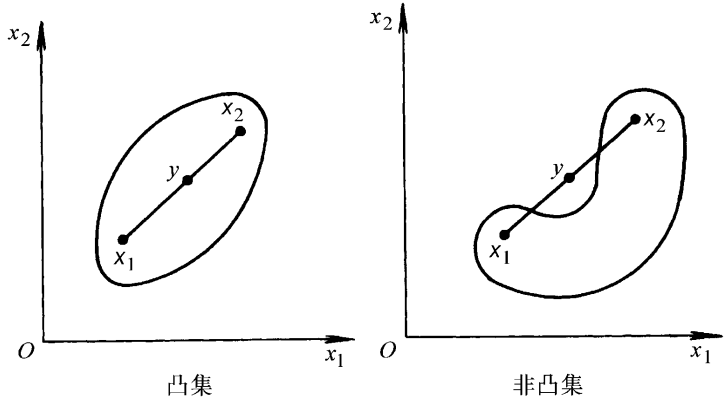
- 对于线性不可分情形，感知机模型无效。

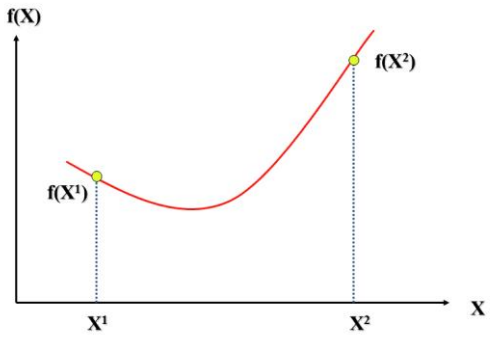
支持向量机通过核函数的方法来实现非线性化



支持向量机

- 支持向量机（Support Vector Machine）是统计学方法的代表，是一个功能强大且全面的机器学习模型，它能够执行线性或者非线性分类、回归以及异常值检测任务。
- SVM在解决小样本、非线性及高维度的模式识别问题中表现出明显的优势。在许多领域，如文本分类、图像识别、生物信息学等领域中得到了成功的应用。
- 有严密的数学依据，得到了严格的数学证明。





求解凸二次规划问题

支持向量机的核心思想

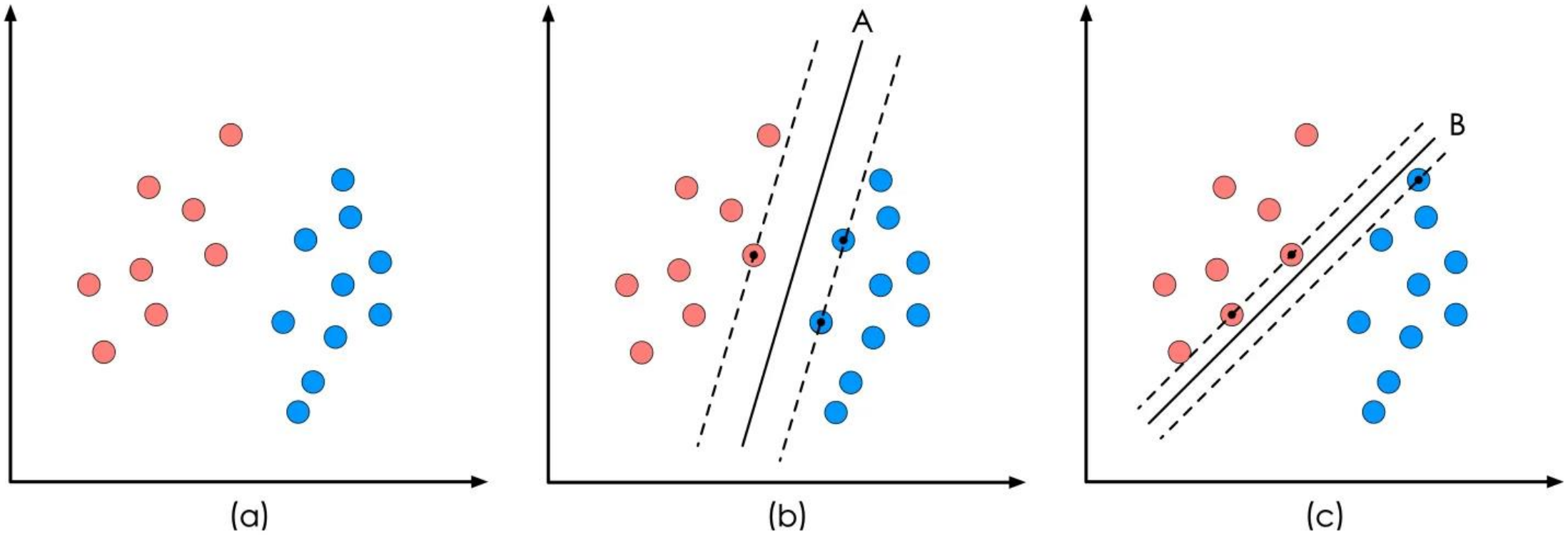
- 二分类模型
- 针对线性可分情形，SVM 选择在特征空间上分类间隔最大的线性分类器，分类间隔最大使它有别于感知机。
- 针对线性不可分情形，SVM 采用核技巧，这使它成为实质上的非线性分类器。
- 支持向量机的学习策略就是分类间隔最大化，可形式化为一个求解凸二次规划(convex quadratic programming)的问题，也等价于正则化的合页损失函数的最小化问题。支持向量机的学习算法是求解凸二次规划的最优化算法。

SVM 分类

- 线性可分支持向量机
  - 硬间隔最大化(hard margin maximization)；
- 近似线性可分支持向量机
  - 训练数据近似线性可分时，通过软间隔最大化(soft margin maximization)；
- 线性不可分支持向量机
  - 当训练数据线性不可分时，通过使用核技巧(kernel trick)及软间隔最大化。

线性可分支持向量机

Which one is better?



线性可分SVM的学习目标

- 线性可分SVM的学习目标是通过硬间隔最大化求解对应的凸二次规划问题得到最优线性分离超平面 $w^* \cdot x + b^* = 0$ 以及相应的分类决策函数 $f(x) = sign(w^* \cdot x + b^*)$ 。
- 优化对象 → 分离超平面
- 目标函数 → 分类间隔最大化
  - 每一个可能把数据集正确分开的方向都有一个最优决策面，而不同方向的最优决策面的分类间隔通常是不同的，那个具有“最大间隔”的决策面就是SVM要寻找的最优解。
  - 向上向下两个方向平移这个最优分离超平面，直至找到贯穿这两个超平面的样本点，就是SVM中的支持样本点，称为“支持向量”。

分类间隔

- 分离超平面方程为：$\boldsymbol{wx}+\boldsymbol{b}=0$
- 分离超平面关于任意样本点（x_i, y_i)的几何间隔可以表示为：

$$\boldsymbol{d}=\frac{|\boldsymbol{w}\cdot\boldsymbol{x}_i+\boldsymbol{b}|}{\|\boldsymbol{w}\|_2}$$

- 分类决策函数：

假设决策面正好处于间隔区域的中轴线上

找到能够实现二分类的分离超平面，存在多个

找到分类间隔最大的那个分离超平面

$$\boldsymbol{f}(\boldsymbol{x})=\boldsymbol{sign}(\boldsymbol{w}\cdot\boldsymbol{x}+\boldsymbol{b})=\begin{cases}+1, & \boldsymbol{w}\cdot\boldsymbol{x}+\boldsymbol{b}\geqslant 0\\ -1, & \boldsymbol{w}\cdot\boldsymbol{x}+\boldsymbol{b}<0\end{cases}$$

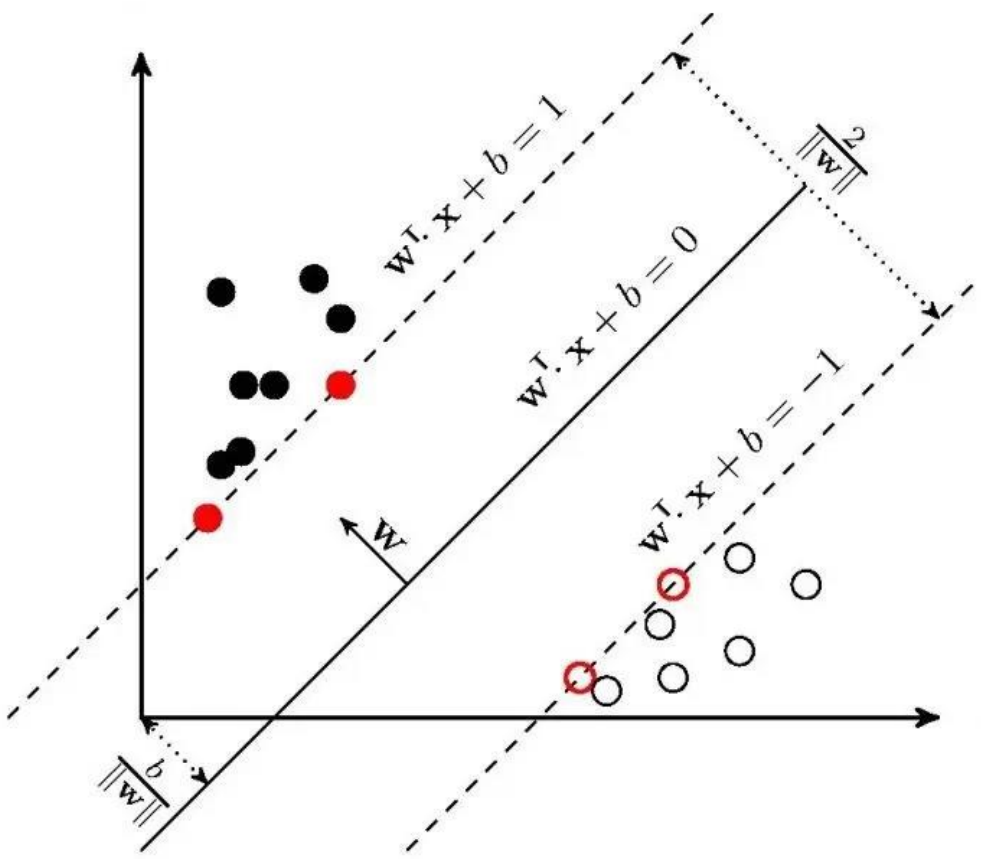
$$\boldsymbol{f}(\boldsymbol{x})=\boldsymbol{sign}(\boldsymbol{w}\cdot\boldsymbol{x}+\boldsymbol{b})=\begin{cases}+1, & (\boldsymbol{w}\cdot\boldsymbol{x}+\boldsymbol{b})/\|\boldsymbol{w}\|_2\geqslant \boldsymbol{d}\\ -1, & (\boldsymbol{w}\cdot\boldsymbol{x}+\boldsymbol{b})/\|\boldsymbol{w}\|_2<\boldsymbol{d}\end{cases}$$

扩展

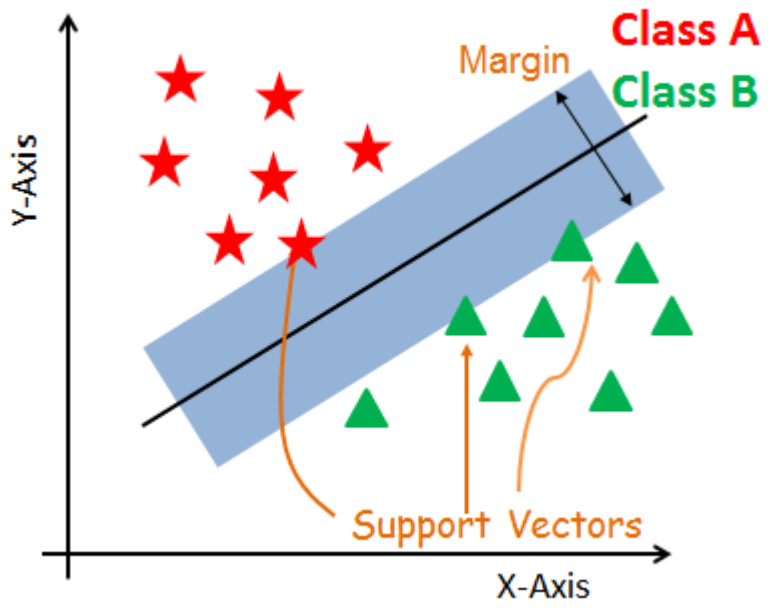
$$\boldsymbol{f}(\boldsymbol{x})=\boldsymbol{sign}(\boldsymbol{w}\cdot\boldsymbol{x}+\boldsymbol{b})=\begin{cases}+1, & \boldsymbol{w}\cdot\boldsymbol{x}+\boldsymbol{b}\geqslant 1\\ -1, & \boldsymbol{w}\cdot\boldsymbol{x}+\boldsymbol{b}<1\end{cases}$$

支持向量

$$d = \frac{|\boldsymbol{w} \cdot \boldsymbol{x}_i + \boldsymbol{b}|}{\|\boldsymbol{w}\|_2} = \frac{1}{\|\boldsymbol{w}\|_2}, \ \textit{if } \boldsymbol{x}_i \textit{ is a support vector}$$



分类间隔最大化



线性可分SVM的数学推导

- 考虑类别标签，分类决策函数可写为：

$$y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+\boldsymbol{b})\geqslant 1$$

- 损失函数为：

$$\max_{\boldsymbol{w},\boldsymbol{b}}\frac{2}{\|\boldsymbol{w}\|_2},\ \boldsymbol{s.t.}\ y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+\boldsymbol{b})\geqslant 1 \quad\Longrightarrow\quad \min_{\boldsymbol{w},\boldsymbol{b}}\frac{1}{2}\|\boldsymbol{w}\|_2,\ \boldsymbol{s.t.}\ y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i+\boldsymbol{b})\geqslant 1$$

如何求解这种有约束最优化问题？

SVM的优点

- 有严格的数学理论支持，可解释性强，不依靠统计方法，从而简化了通常的分类和回归问题；
- 能找出对任务至关重要的关键样本（即支持向量）；
- 采用核技巧之后，可以处理非线性分类/回归任务；
- 最终决策函数只由少数的支持向量所确定，计算的复杂性取决于支持向量的数目，而不是样本空间的维数，这在某种意义上避免了“维数灾难”。

SVM的缺点

- 训练时间长。当采用SMO算法时，由于每次都需要挑选一对参数，因此时间复杂度为O(N^2)，其中 N 为训练样本的数量；
- 当采用核技巧时，如果需要存储核矩阵，则空间复杂度为O(N^2);
- 模型预测时，预测时间与支持向量的个数成正比。当支持向量的数量较大时，预测计算复杂度较高。
- 支持向量机目前只适合小批量样本的任务，无法适应百万甚至上亿样本的任务。

LR vs Perceptron Vs SVM

- 1、目标函数不同(本质区别)
- 逻辑回归的目标函数是训练集上的极大似然函数，等价于交叉熵损失函数。SVM采用的是带L2正则化项的合页损失函数。感知机是最小化误分类点到分离超平面的距离的和。感知机损失函数非凸，但由于可以证明线性可分数据集中感知机模型具备收敛性，因此可以把零误差点当做最终学习目标，即损失函数等于0。逻辑回归的损失函数是凸函数，可以正常当作最优化问题去求解。

LR vs Perceptron Vs SVM

- 2、激活函数
- 逻辑回归比SVM（感知机）的优点之一在于对于激活函数的改进。前者为sigmoid，后者为符号函数。sigmoid使得最终结果有了概率解释的能力（将结果限制在0-1之间），而step为分段函数，对于分类的结果处理比较粗糙，非0即1，而不是返回一个分类的概率。

LR vs Perceptron Vs SVM

- 3、从寻找分离超平面的方式
- 逻辑回归找到的那个超平面，是尽量让所有点都远离它（交叉熵损失函数，所有点都要参与计算），而SVM寻找的那个超平面，是最大化几何间隔，即支持向量离超平面的距离最大。

LR vs Perceptron Vs SVM

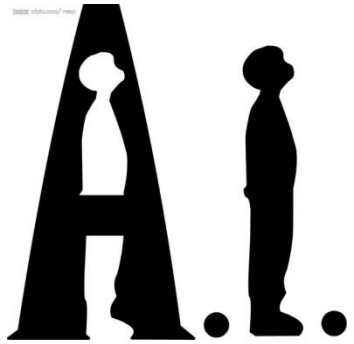
- 4、对异常值的敏感度
- LR中所有样本都对分类平面有影响，所以异常点的影响会被掩盖。但SVM的分类平面取决于支持向量，如果支持向量是异常点，则结果难以预测。

LR vs Perceptron Vs SVM

- 5、高维数据稳定性
- 在高维空间LR的表现比SVM更稳定，因为SVM是要最大化间隔，这个间隔依赖于距离表示方法（欧氏距离），在高维空间时这个距离难以度量，通常需要对向量做归一化。

LR vs Perceptron Vs SVM

- 6、训练难度
- SVM在训练过程只需要支持向量，依赖的训练样本数较小，而逻辑回归则是需要全部的训练样本数据（交叉熵损失函数，所有点都要参与计算），在训练时开销更大。



Thank you!