第四周 传统人工智能模型

李泽榉，复旦大学生物医学工程与技术创新学院

部分内容参考斯坦福CS221课程课件

任课老师介绍

李泽榉，生物医学工程与技术创新学院
个人主页：https://zerojumpline.github.io/

研究兴趣：
医学图像处理，机器学习，计算神经学

办公室：
江湾校区交叉学科二号楼C2008

邮箱：
zejuli@fudan.edu.cn

授课时间

Week	Date	Plan
5	10/10/2025	Traditional AI
6	10/17/2025	Artifical Neural Networks
11	11/21/2025	Transformer
13	12/5/2025	Reinforcement Learning
14	12/12/2025	Ethics and Safety
16	12/26/2025	Open Discussion

小心！

- 这是由我讲授的第一门面向中文专业本科生的课程，讲解上可能还有不清晰之处，敬请大家见谅。
- **课件内容包含了平行课程的所有信息**，并增加了一些我个人的理解。
- 课件中也可能存在一些不准确或疏漏，非常欢迎大家随时指出问题并与我讨论。

授课时间

课程网页

目录

K近邻算法（KNN）

近朱者赤，近墨者黑

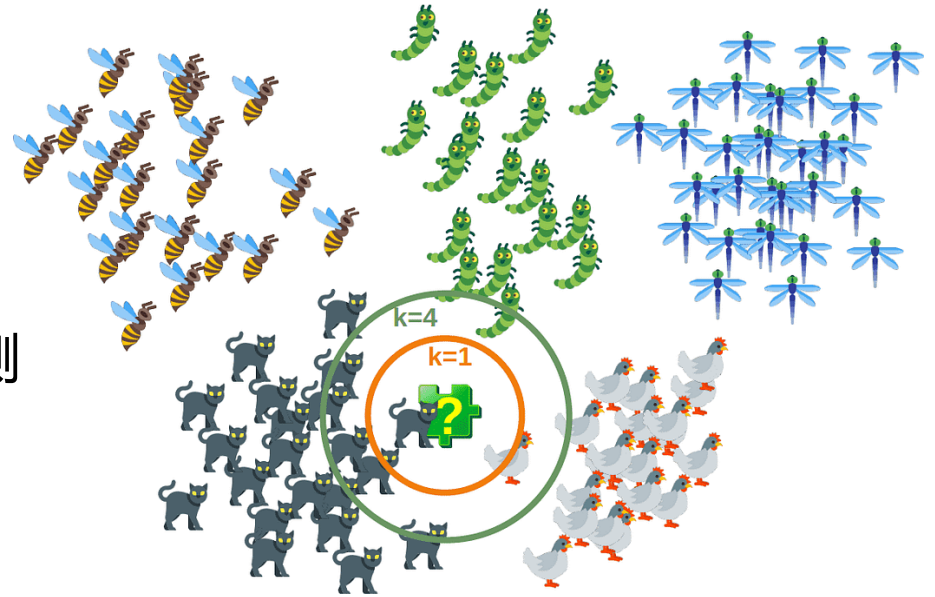
——晋·傅玄《太子少傅箴》

靠着朱砂的变红，靠着墨的变黑。比喻接近好人可以使人变好，接近坏人可以使人变坏；用于比喻客观环境对人有很大影响。我们在这里用来表示一种**古老而经典的分类思想。**

机器学习里面有一个非常经典的分类算法，K近邻算法，就是利用这种思想，**根据周围数据点的类别去判定当前数据点的类别**。这里面涉及一个重要的假设，即“同类的数据会靠的更近”，不管是在输入空间还是在表征空间。在深度学习的时代，表征学习的一个根本原则就是“在相关性比较强的数据点需要在表征空间考的更近”，与这样的假设一脉相承。所以**这个简单分类方法，可以基于深度表征做出很准确的分类准确率。**

K近邻算法（KNN）

- **定义**：K-近邻(KNN)是一种常用的监督学习算法
- **用途**：KNN算法可以用于解决**分类问题**和**回归问题**
- **核心思想**：通过计算**测试数据**与**训练数据**之间的**距离**进行预测

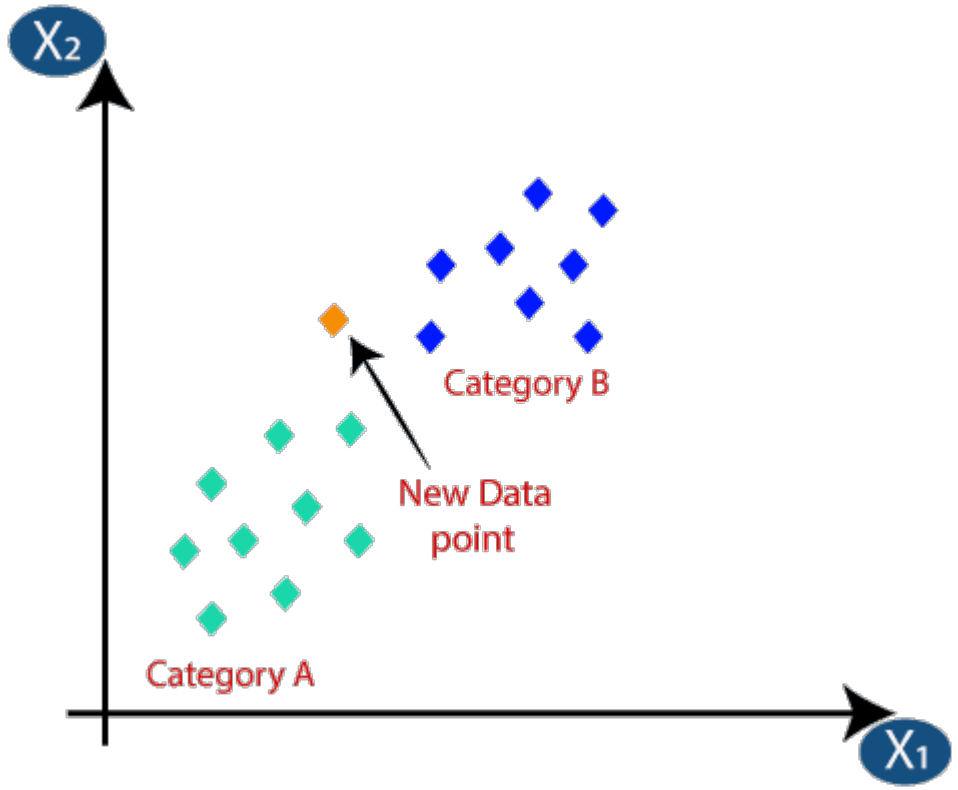


- **决策机制**：
  1. 算法首先选择k
  2. 然后，计算测试数据点与所有训练数据点的距离，选择距离测试数据点最近的K个训练数据点，称为K-近邻
  3. 分类问题：KNN算法统计这K个近邻中每个类别的训练数据点数量，并将测试数据点**预测为数量最多的类别**
  4. 回归问题：KNN算法将这K个近邻的目标值的**平均值作为测试数据点的预测值。**

K近邻算法（KNN）

- KNN算法步骤例子：

  步骤 1：选择K值（K=5）



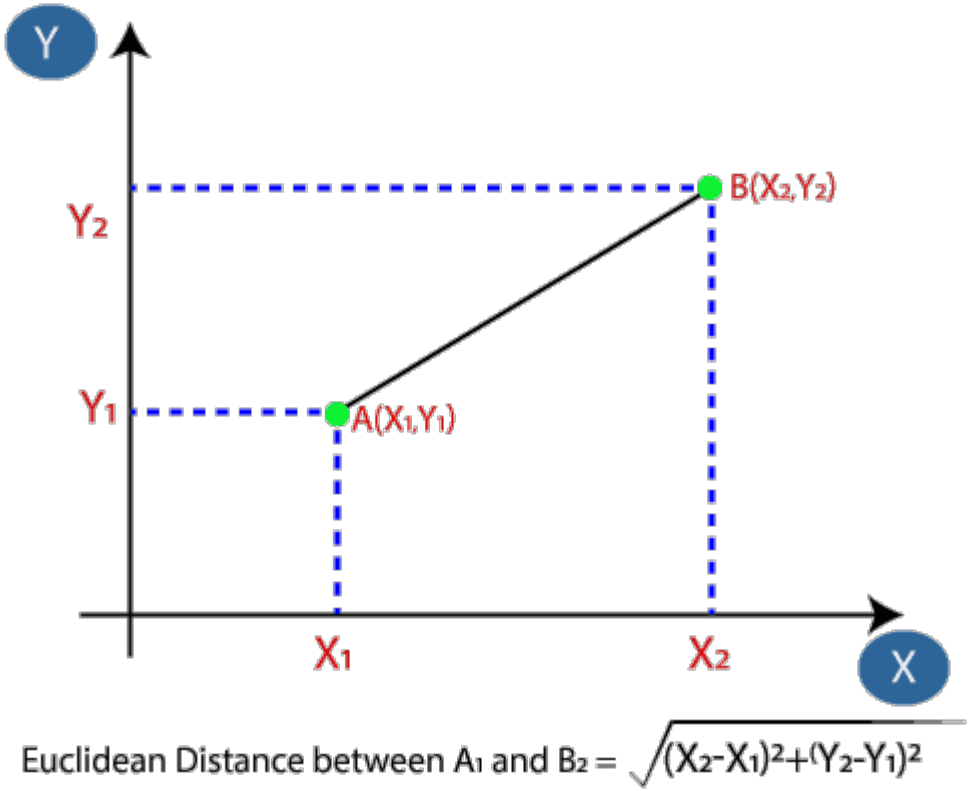
K近邻算法（KNN）

- KNN算法步骤例子：

步骤 1：选择K值（K=5）

步骤 2：计算欧式距离

① 计算不同点之间的距离



K近邻算法（KNN）

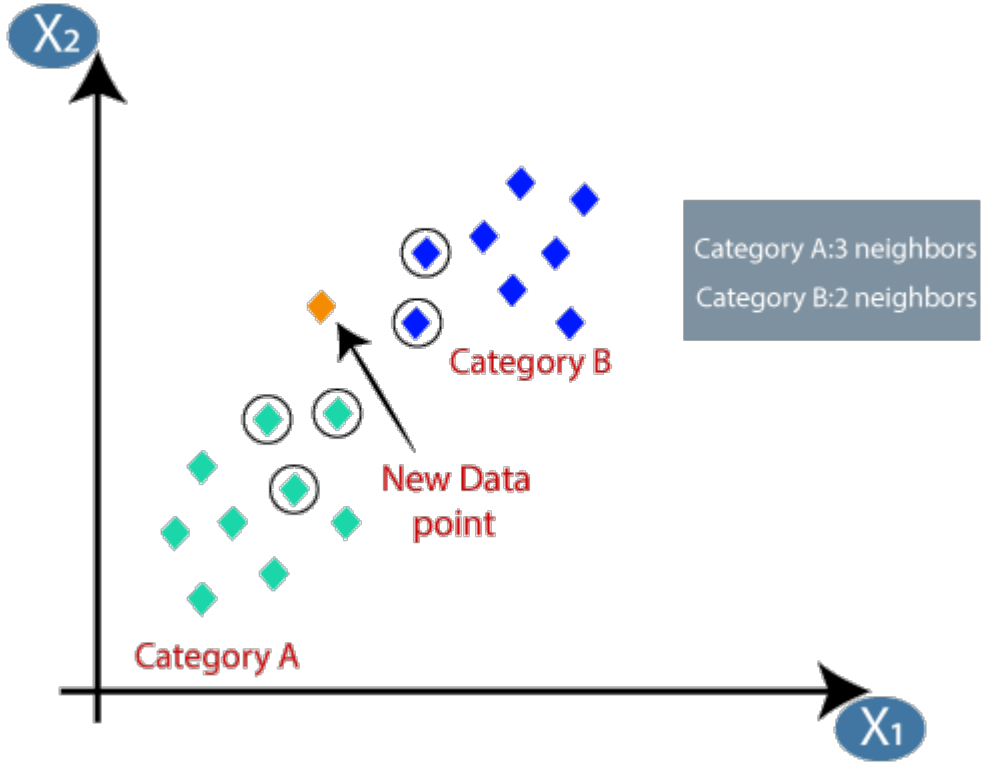
- KNN算法步骤例子：

步骤 1：选择K值（K=5）

步骤 2：计算欧式距离

① 计算不同点之间的距离

② 选择最近的5个数据点



K近邻算法（KNN）

- KNN算法步骤例子：

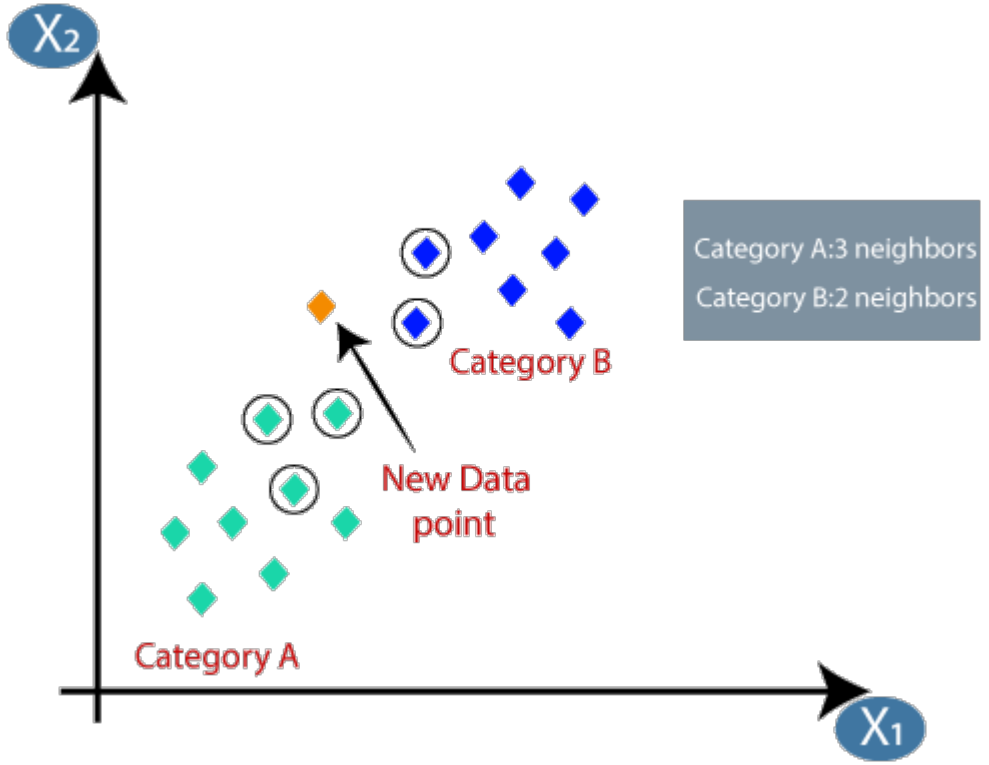
步骤 1：选择K值（K=5）

步骤 2：计算欧式距离

① 计算不同点之间的距离

② 选择最近的5个数据点

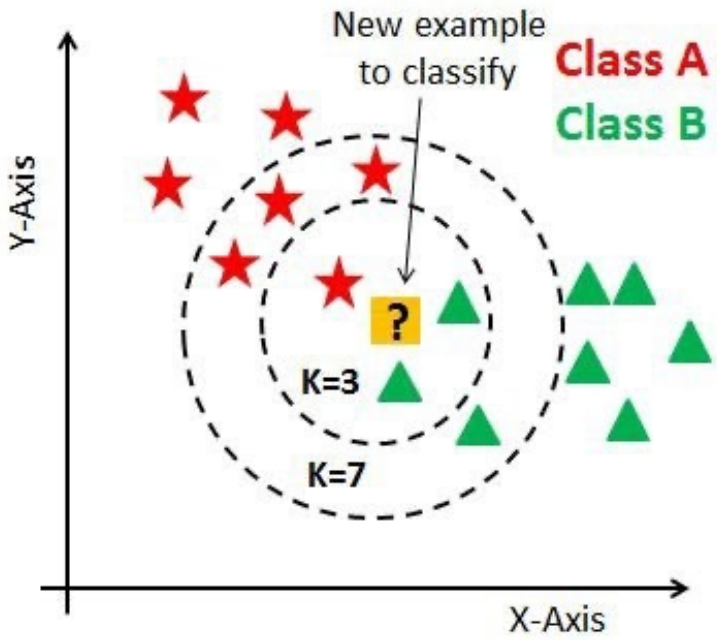
步骤 3：最近三个点是A，两个点是B，所以类别是A



K近邻算法（KNN）

- **如何选择最佳的K：**
  - ① 初始K值：先随便选一个K值开始尝试（例如K=3,5,或7）
  - ② 尝试不同K值：在一定范围内尝试多个不同的K值（例如K=1,2,3,...,20）
  - ③ 计算误差率：对于每个K值，计算误差率
  - ④ 选择最佳K值：在曲线上找到误差率最低的点，对应的K值就是近似最优的K值

- **K值大小的影响：**
  - ① 小K值（不稳定）：决策边界不稳定，容易受个别数据点影响
  - ② 大K值（更平滑）：决策边界更平滑，但可能忽略局部细节



举例：用KNN进行图片分类

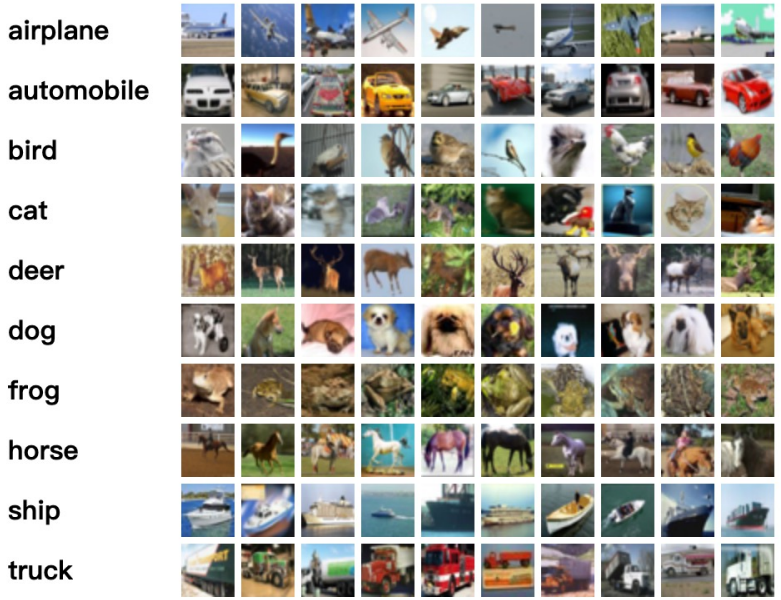
① **选定待分类图像:** 我们有一张待分类的 CIFAR-10 图像

② **计算距离:** 将待分类图像与训练集中的每张图像计算距离。

对于训练集图像 1，计算：

$$d_1 = ||\mathbf{x}_{待分类} - \mathbf{x}_{训练1}||_2 = \sqrt{\sum_{j=1}^{D}(x_{待分类,j} - x_{训练1,j})^2} = 1.44$$

以此类推，计算待分类图像与所有训练集图像的距离



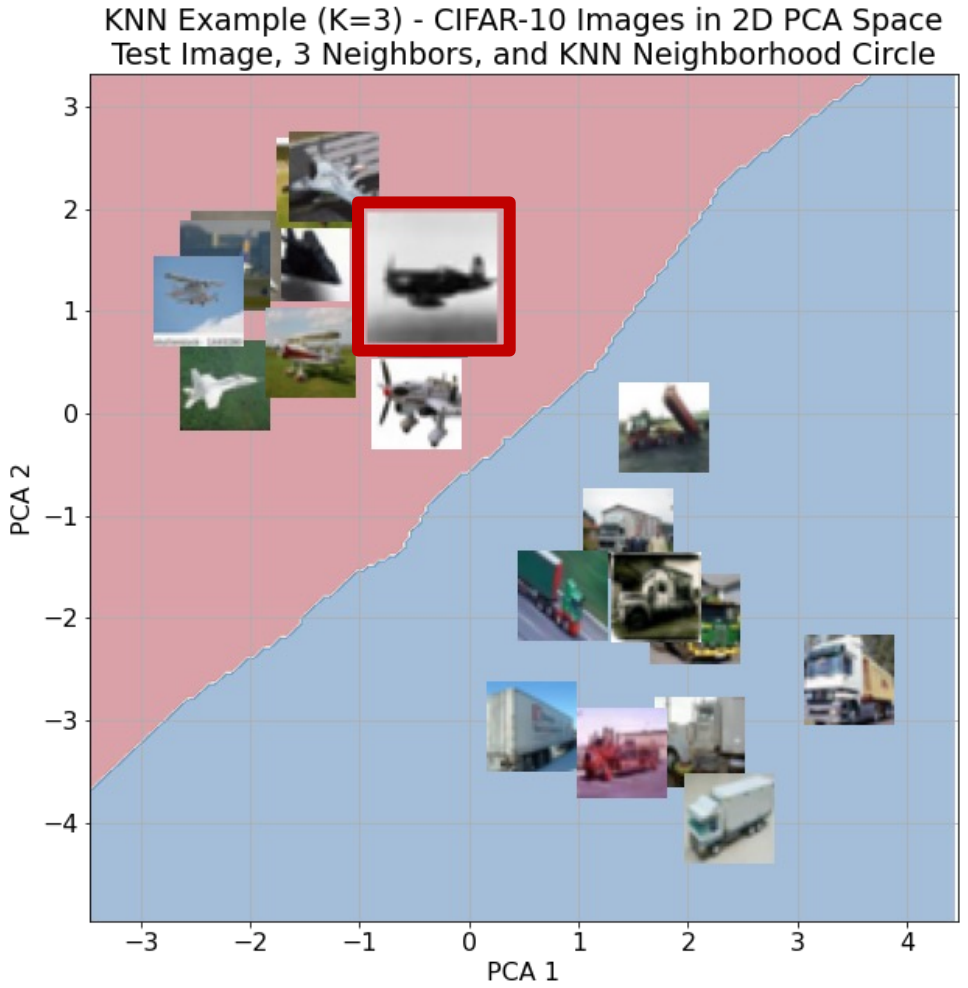
举例：用KNN进行图片分类

③ **选取 K 个最近邻:** 根据距离排序，选取距离**最小的K张图像**作为最近邻 。

- 最近邻 1: (距离: 0.84, 类别: 飞机)
- 最近邻 2: (距离: 0.96, 类别: 飞机)
- 最近邻 3: (距离: 1.01, 类别: 飞机)

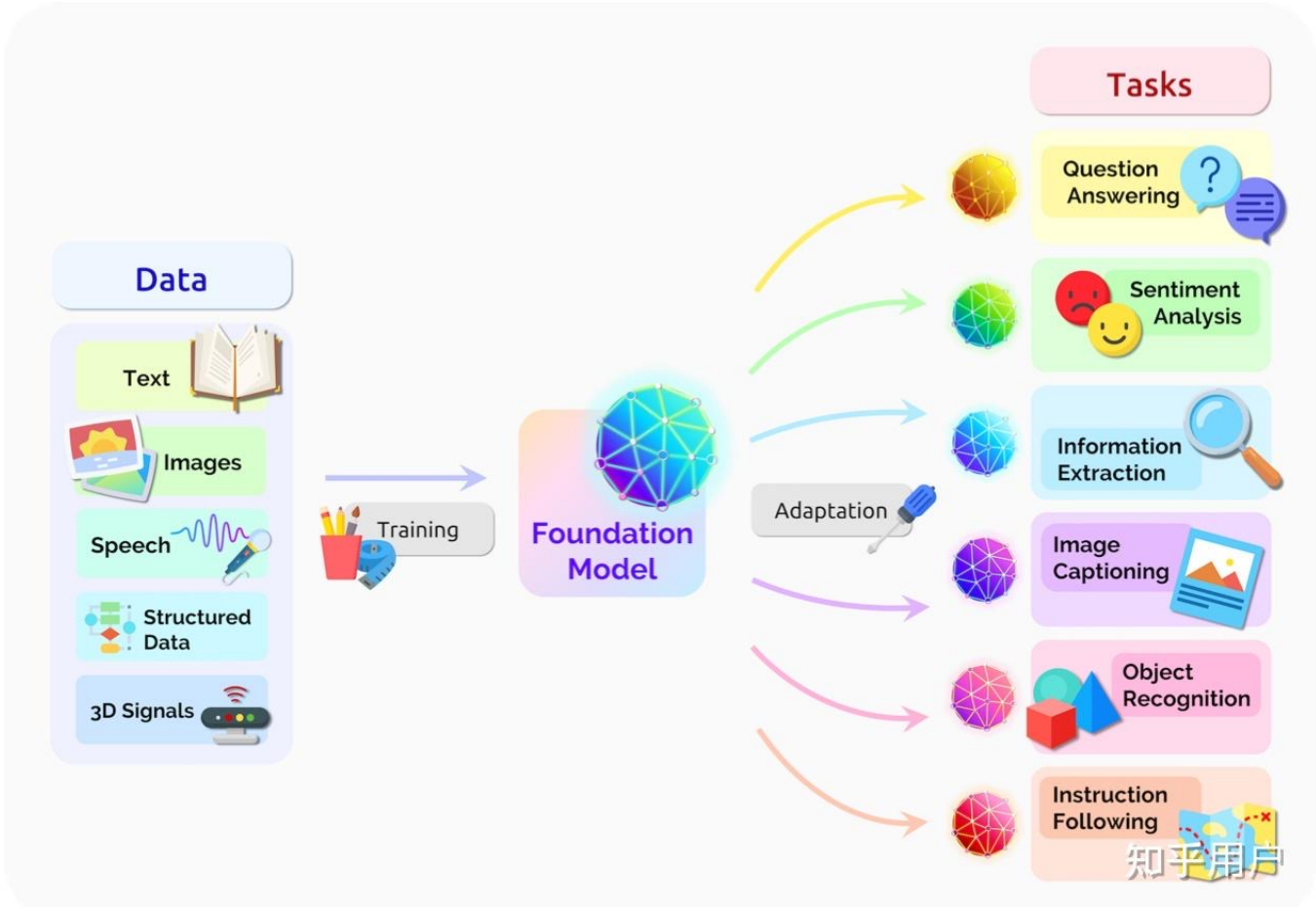
④ **类别投票:** 统计 K 个最近邻的类别，**票数最多的类别** 即为待分类图像的预测类别。 **示例:**

- “统计 3 个最近邻的类别: 飞机 (3票)”
- “**投票结果: 飞机 (票数最多)**”
- “**预测类别: 飞机**”



KNN真实应用举例

- 预训练模型
- Aka. 大模型



https://github.com/facebookresearch/dino

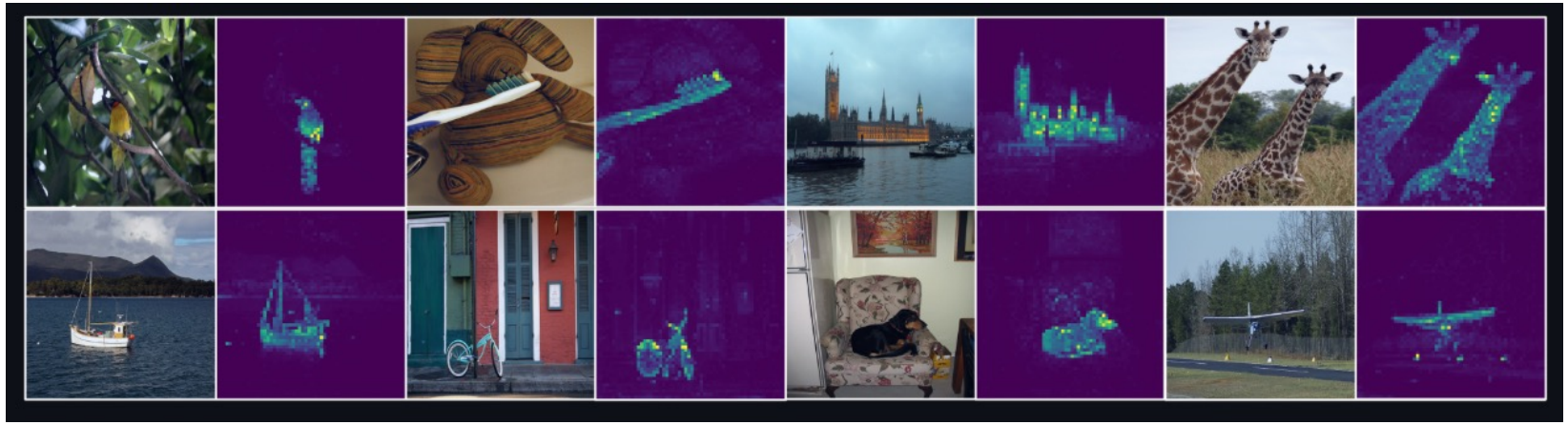
KNN真实应用举例

- 预训练不需要标签数据
- Aka. 自监督学习
- DINO

https://github.com/facebookresearch/dino

KNN真实应用举例

- 预训练不需要标签数据
- Aka. 自监督学习
- DINO



https://github.com/facebookresearch/dino

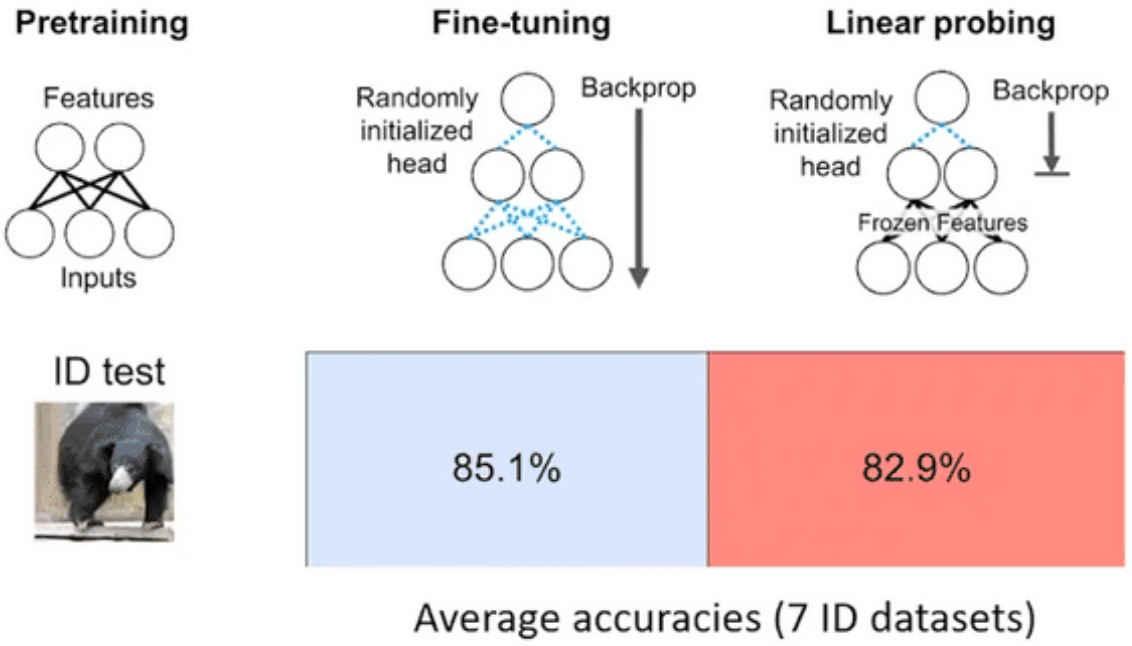
KNN真实应用举例

- 预训练不需要标签数据
- Aka. 自监督学习
- DINO

https://github.com/facebookresearch/dino

KNN真实应用举例

- 预训练不需要标签数据
- Aka. 自监督学习
- DINO
- 下游任务进行微调
- 全参数微调vs线性探测（回归）



https://github.com/facebookresearch/dino

KNN真实应用举例

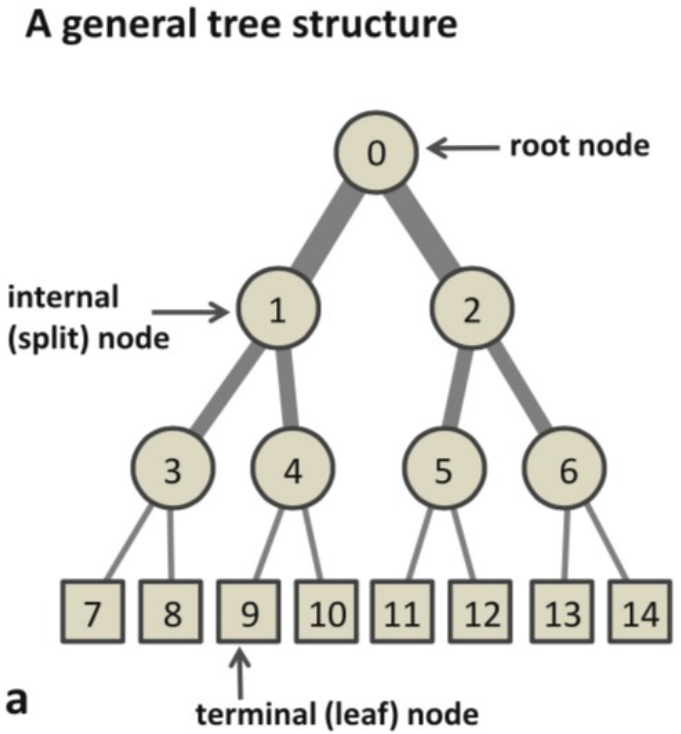
- 预训练不需要标签数据
- Aka. 自监督学习
- DINO
- 下游任务进行微调
- 下游任务微调使用KNN

arch	params	k-nn	linear
ViT-S/16	21M	74.5%	77.0%
ViT-S/8	21M	78.3%	79.7%
ViT-B/16	85M	76.1%	78.2%
ViT-B/8	85M	77.4%	80.1%
ResNet-50	23M	67.5%	75.3%

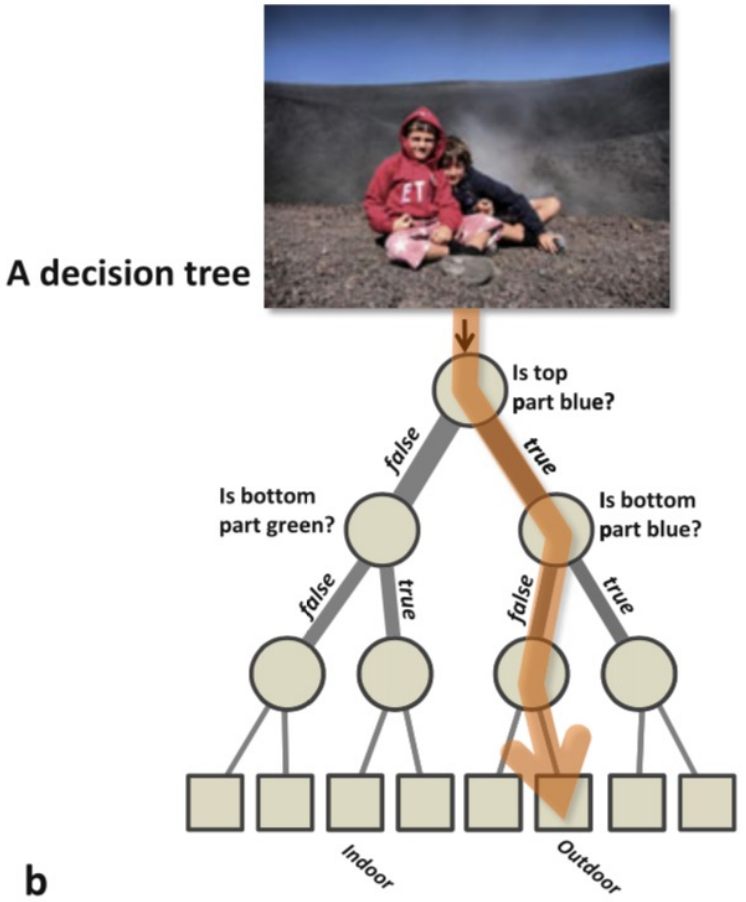
https://github.com/facebookresearch/dino

目录

决策树 （Decision Tree）



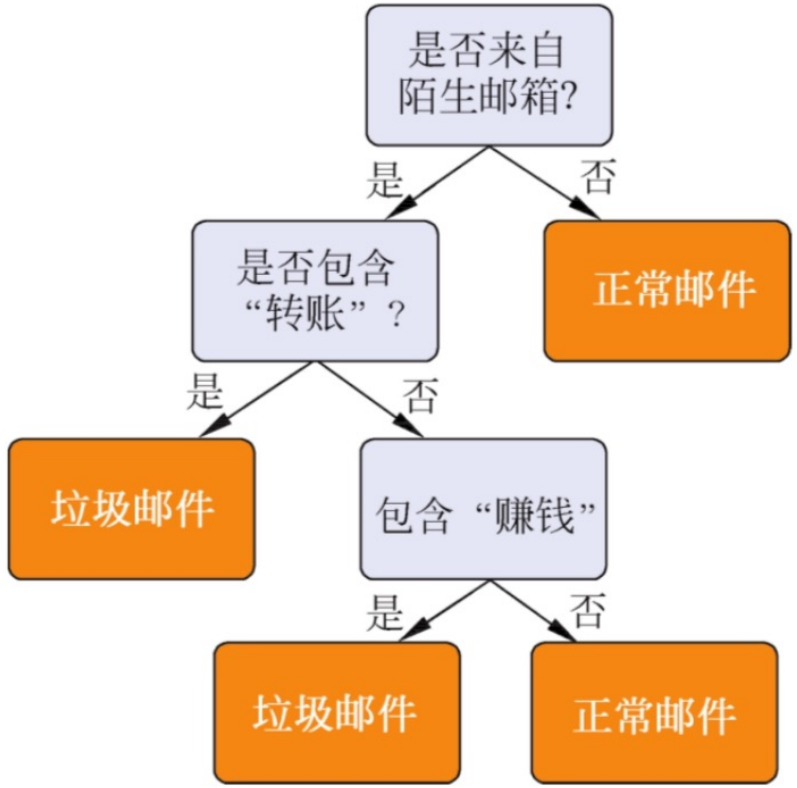
决策树 （Decision Tree）



决策树 （Decision Tree）

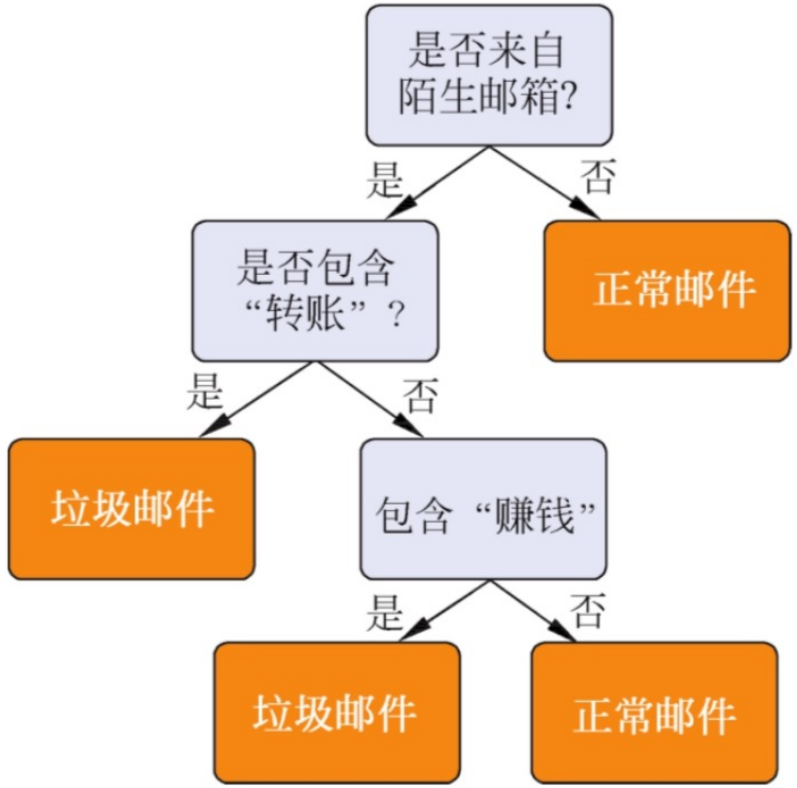
- 决策树是一种简单的**分类或回归模型**

- 决策树的基本结构：
  - 分类流程 -> **树状结构**
  - 最上方的灰色方框：**根节点**
  - 箭头：**树的分支**
  - 方框：**树的节点**
  - 灰色的方框：**中间节点，代表一个判断条件**
  - 橙色的方框：**叶子节点，记录判定结果**
  - 从根节点到每个叶子节点的路径：**决策路径**



决策树 （Decision Tree）

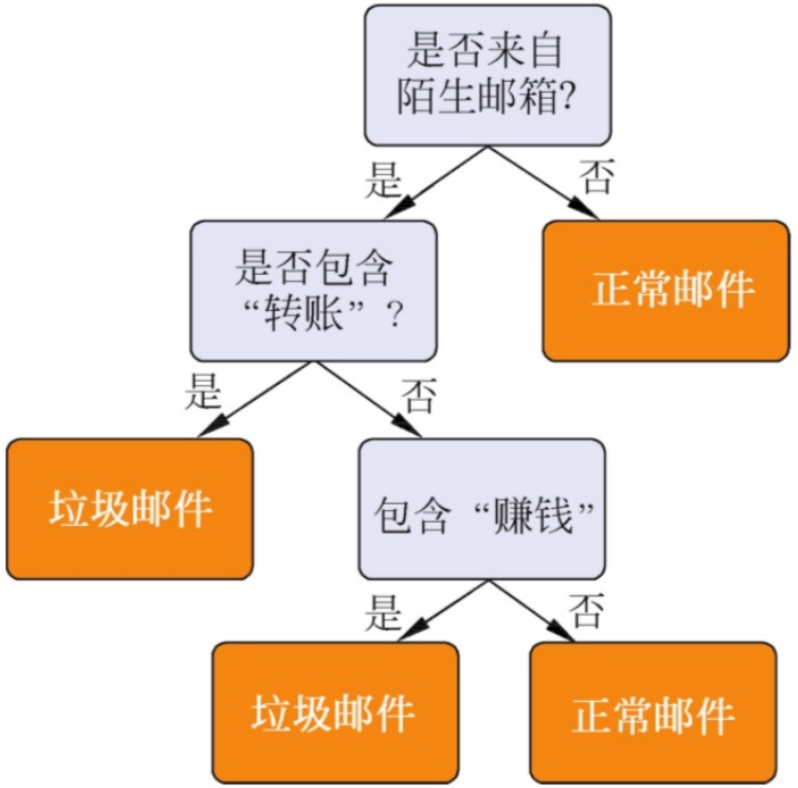
- 决策树是一种简单的**分类或回归模型**
- 当预测结果是数据所属的类别时，采用分类树分析
- 当预测结果可视为实数（如房屋价格、患者住院天
  时，则采用回归树分析



决策树 （Decision Tree）

□ 决策树的正式定义：

一个树结构的每个中间节点对数据的某一个特征进行判断，根据判断结果的不同指向相应的子节点。并且，该树结构的每个叶子节点，对符合所有根节点到该叶子节点路径上判断条件的数据给出一个预测值。这样的树结构称为决策树。



□ 决策树有两类：

针对不同的问题，决策树的叶子节点需要输出不同的预测。对于回归问题，叶子节点的预测值是一个实数；而对于分类问题，则是一个类别。前者称为**回归树**，后者称为**分类树**。

- 在垃圾邮件分类例子中，采用的是分类树

决策树 （Decision Tree）

口 如何训练得到一个决策树?

训练目标： 从训练数据中找到最好的判断条件和树的结构，从而准确地预测新邮件是不是垃圾邮件

训练过程可以分为4步:

1. **从根节点开始**： 把所有训练邮件放在一起。
2. **选择最佳判定条件**： 从邮件的特征中，选择一个最好的条判定件来划分邮件
3. **划分数据**： 根据问题的答案，把邮件划分成不同的组 (即创建分支)
4. **重复执行2、3两步**： 对每一组邮件，重复步骤2和3，直到满足停止条件

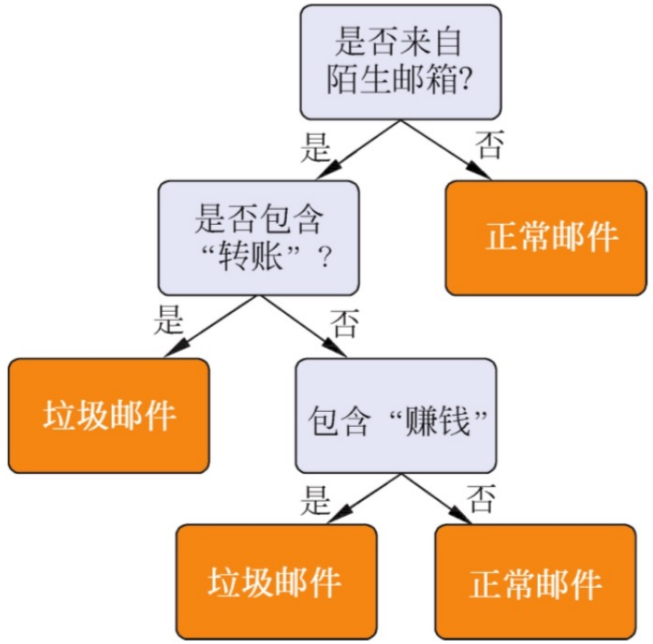
决策树 （Decision Tree）

- **决策树训练**：选择“最佳判定条件”

- **什么是“判断条件”？**
  针对特征提出的问题，用来分割数据。（比如“是否是陌生邮箱？”、“是否包含‘转账’两个字？”）

- **如何选择“判断条件”？**
  核心目标：选择一个问题，让划分后的节点更“**纯净**”

  - **纯净**：分割后的节点里，同类邮件 (垃圾或正常) 尽可能多
  - **更纯净**：分割后的节点里，大部分是垃圾邮件，或者大部分是正常邮件
  - **不纯净**：分割后的节点里，垃圾邮件和正常邮件都很多，混在一起

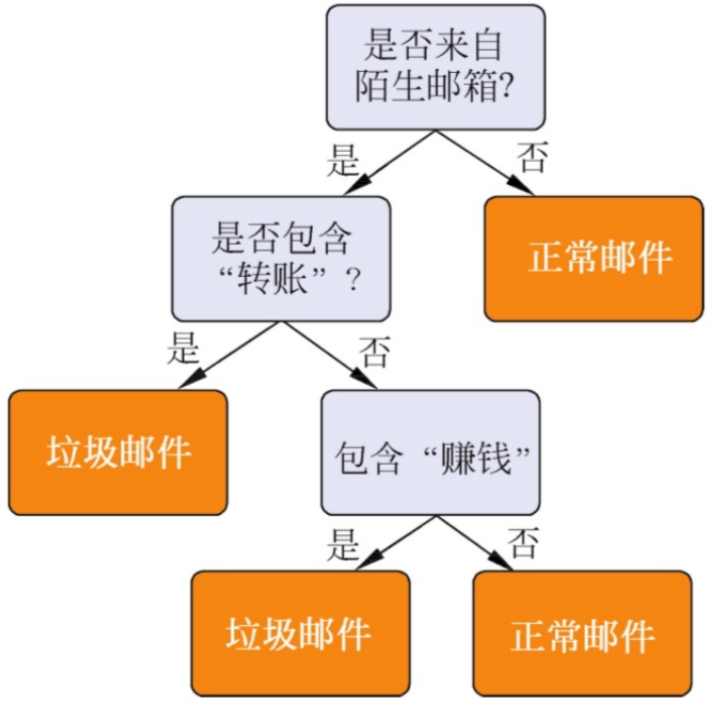


决策树 （Decision Tree）

➢ **决策树训练：**分支生长

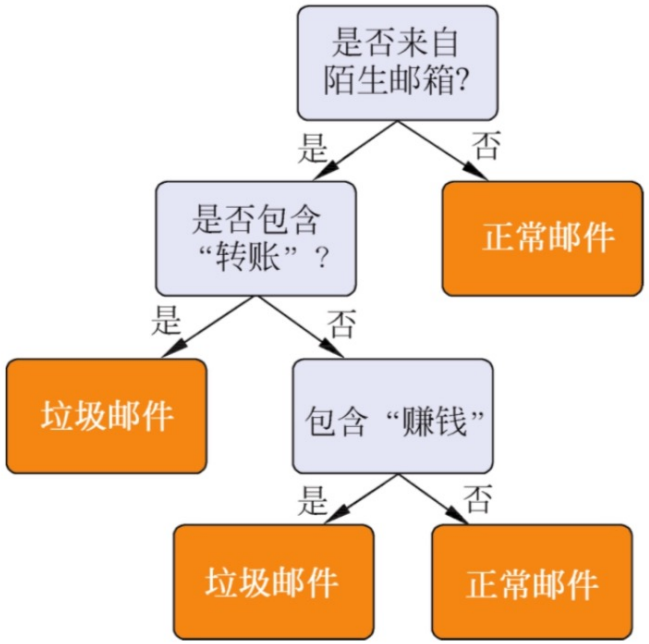
根据最佳判定条件，将根节点中的样本划分为两个或多个子节点，形成树的分支

- **每个分支代表一种答案**： 例如，如果问题是“是否来自陌生邮箱？”， 就分出“是” 分支和 “否” 分支。
- **数据划分**： 训练数据根据判定结果，被分配到不同的分支，进入不同的子节点
- **子节点变成新的“根”**： 每个子节点又可以看作是新的 “根节点”，等待进一步的分割



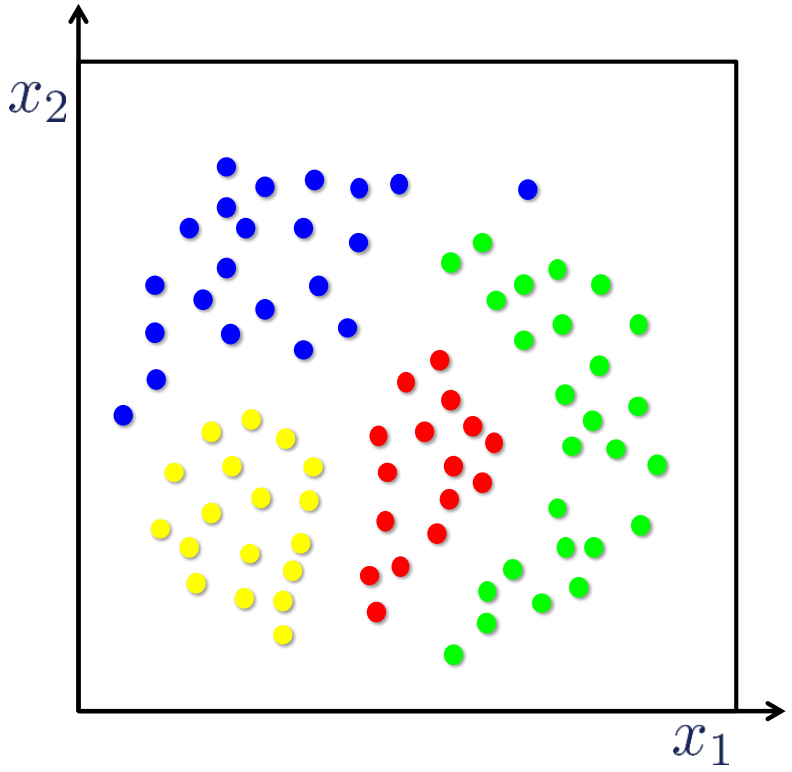
决策树 （Decision Tree）

- **决策树训练：**分支生长
- **递归分割：** 对每个子节点，重复步骤2和步骤3 ，逐步细化分类
- **停止条件：** 划分过程不会无限进行，需要设定停止条件。
  - ✓ 节点足够 “纯净”： 一个节点里的邮件，95%以上都是垃圾或正常邮件，就认为足够纯净，不再划分。
  - ✓ 达到树的最大深度： 限制树的层数，防止树过于复杂
  - ✓ 节点内数据太少： 一个节点里的邮件数量太少，停止划分



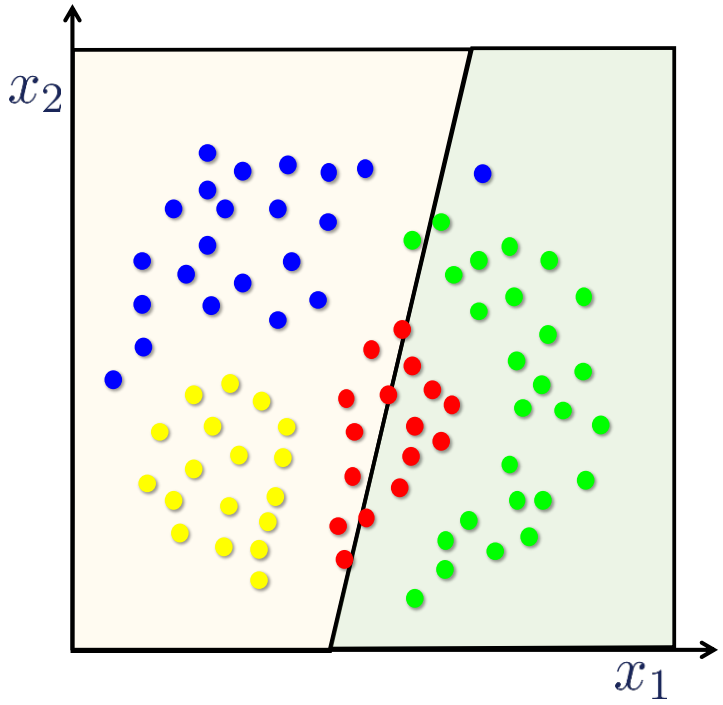
- **叶子节点诞生**： 当满足停止条件时，节点变成叶子节点。
- **叶子节点的预测值**： 根据节点中大多数样本的类别， “少数服从多数” (垃圾邮件 或 正常邮件)

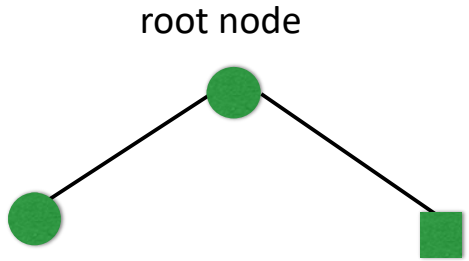
决策树 （Decision Tree）



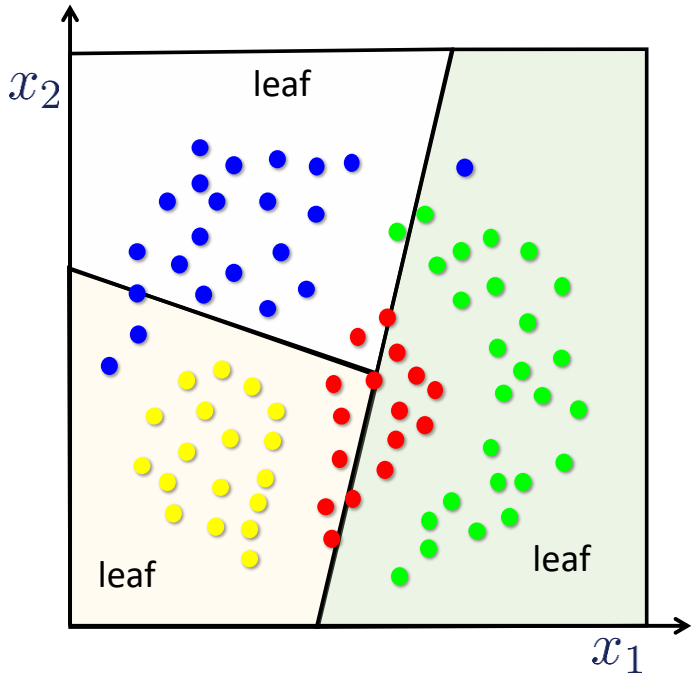
root node

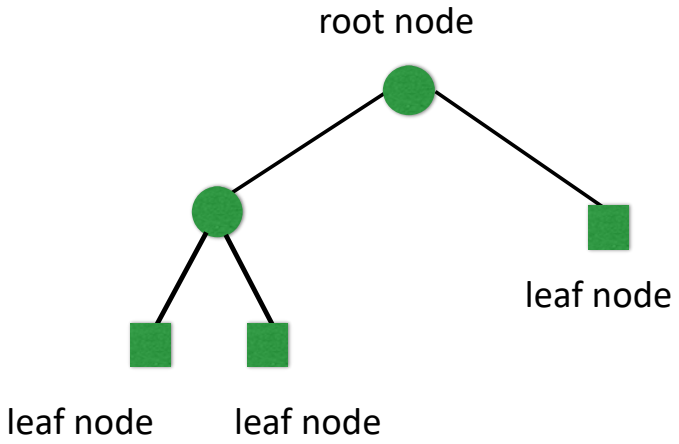
决策树 （Decision Tree）



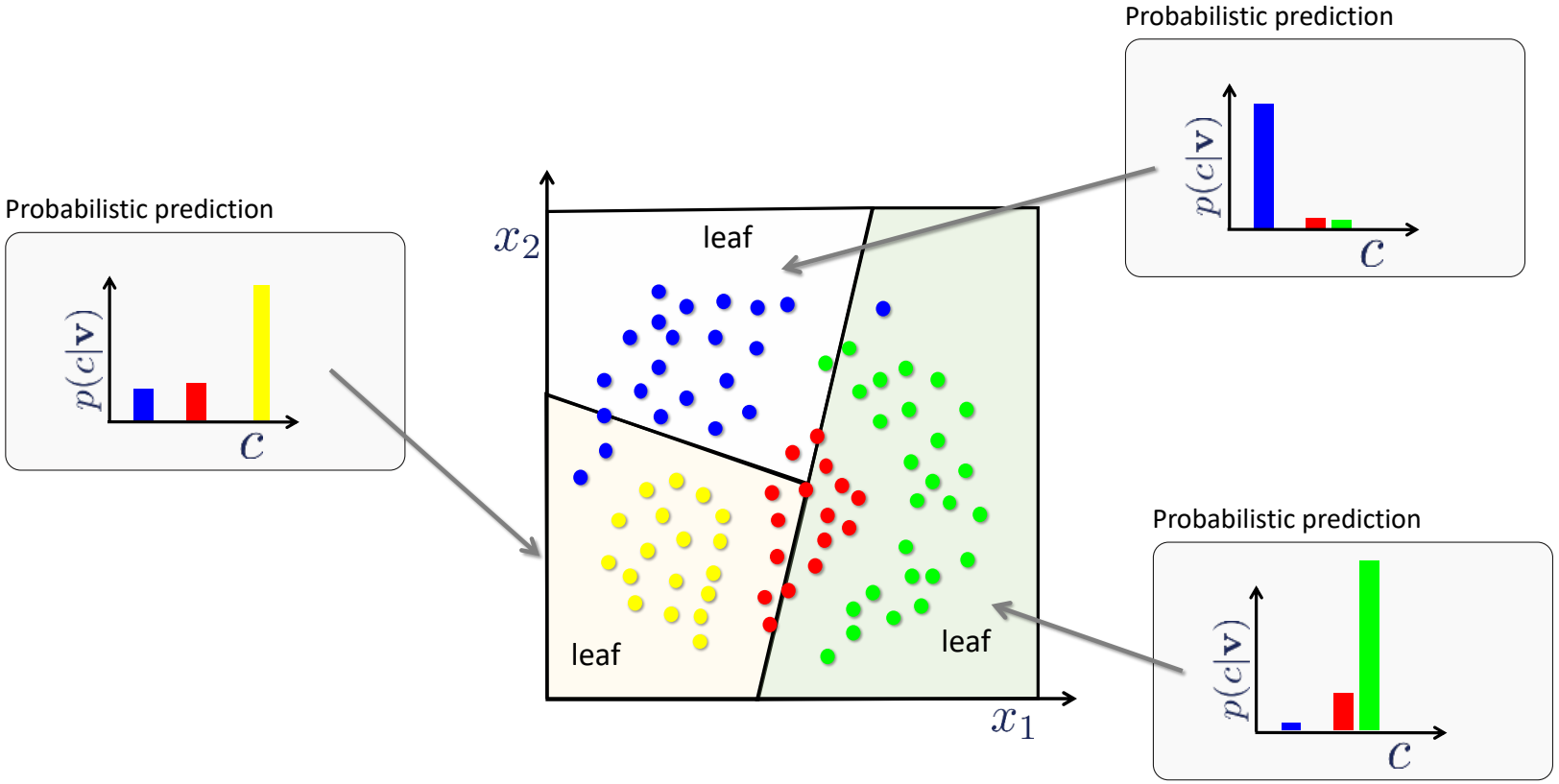


决策树 （Decision Tree）



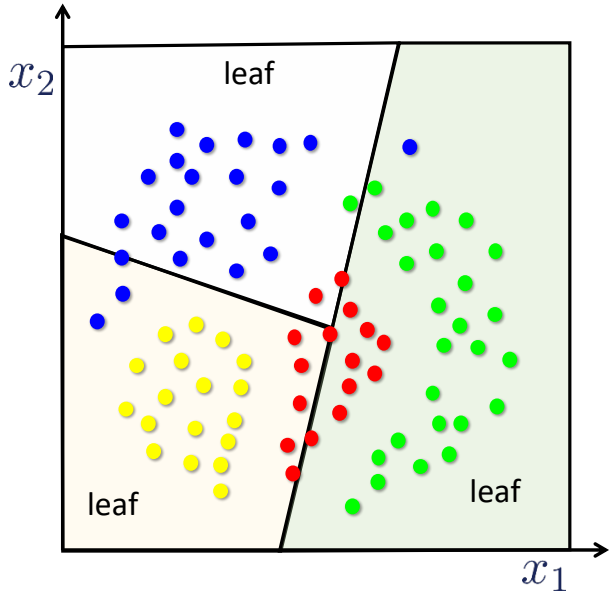


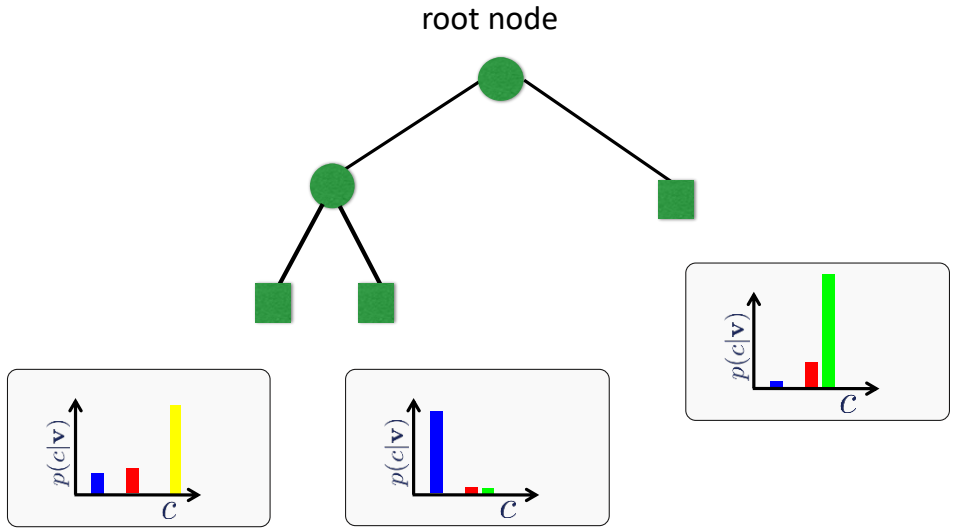
决策树 （Decision Tree）



p_i是该样本中属于类别i的数据比例

决策树 （Decision Tree）



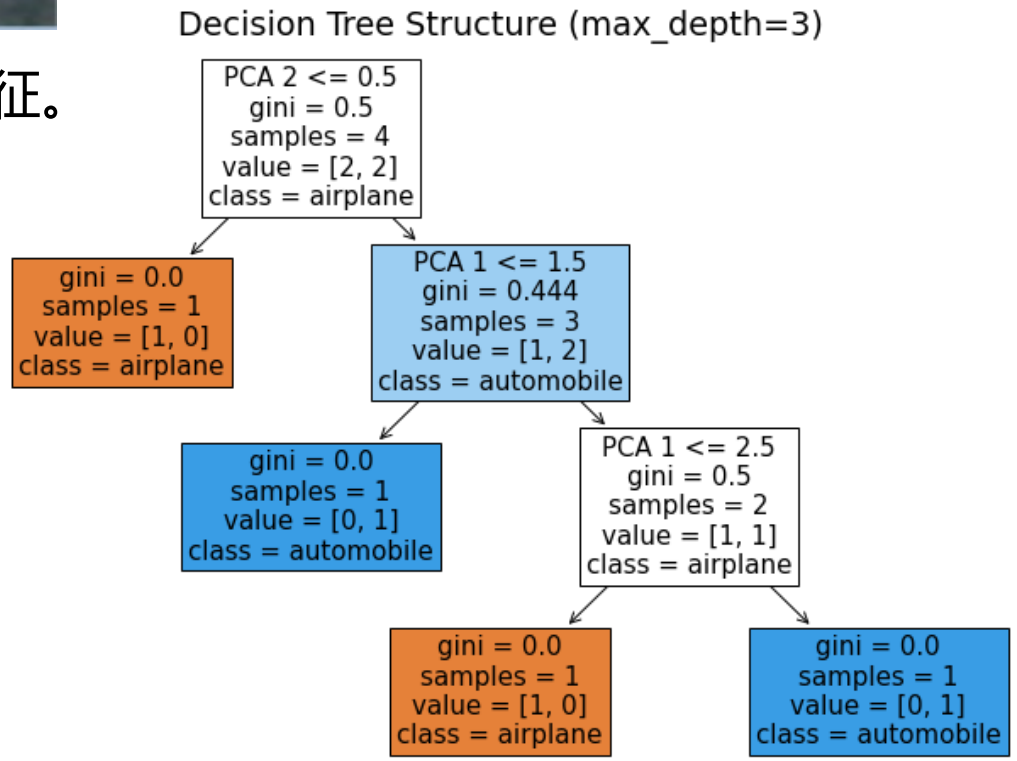


举例：用决策树进行图片分类

① **选定待分类图像:** 我们有一张待分类的 CIFAR-10 图像

② **决策过程:** 将待分类图像输入决策树进行决策，逐步判断特征。

- 对于 **决策树的根节点**
  - **判断条件:** PCA 2 ≤ 0.5
  - **数值示例: 待分类图像的 PCA 1 值为 1.0**
  - **判断结果:** 否
- **决策路径:** 进入**决策树的是分支**，到达下一个节点。
- **对于下一个节点**
  - **判断条件:** PCA 1 ≤ 1.5
  - **数值示例: 待分类图像的 PCA 1 值为 1.0**
  - **判断结果:** 是汽车



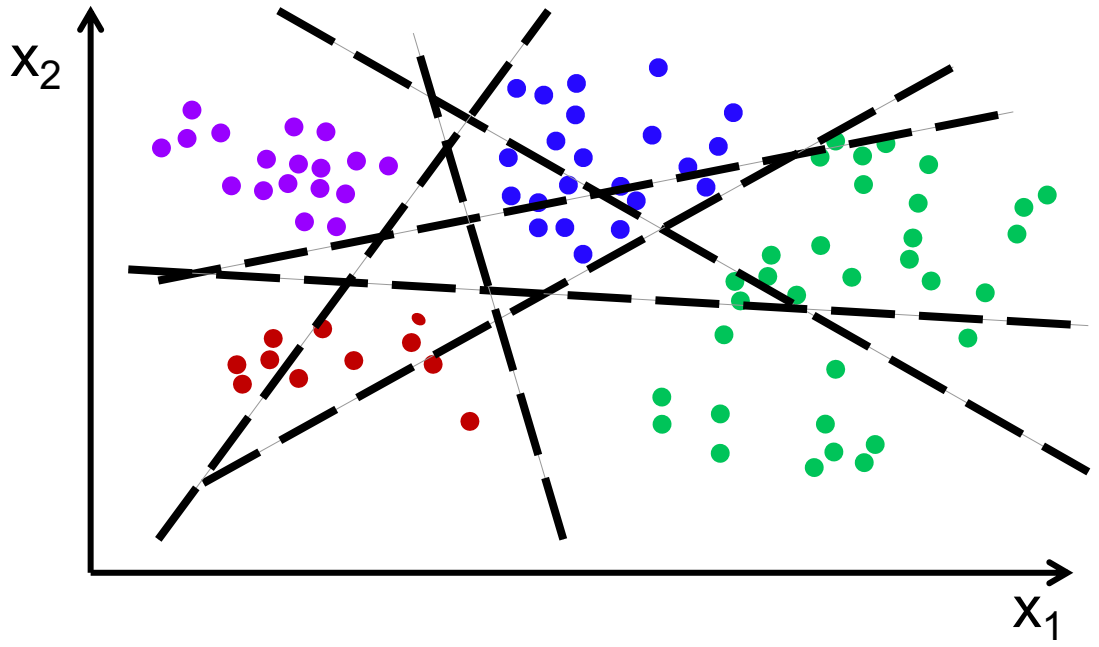
$$Gini = 1 - \sum_{i=1}^{n} (p_i)^2$$

p_i是该样本中属于类别i的数据比例

决策树的优化

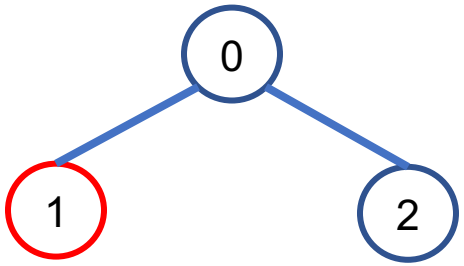
贪婪随机优化

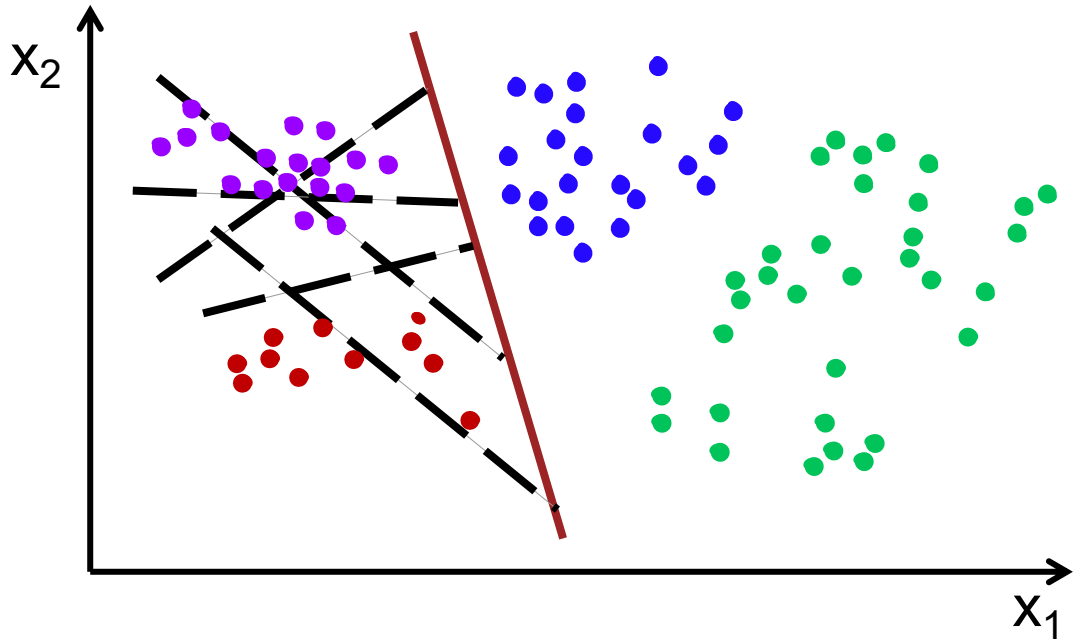
0



决策树的优化

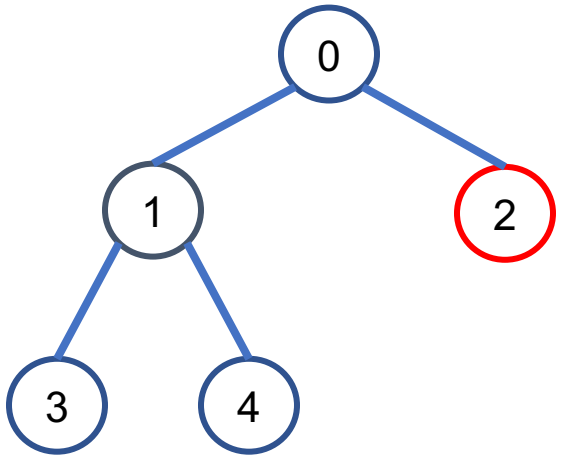
贪婪随机优化

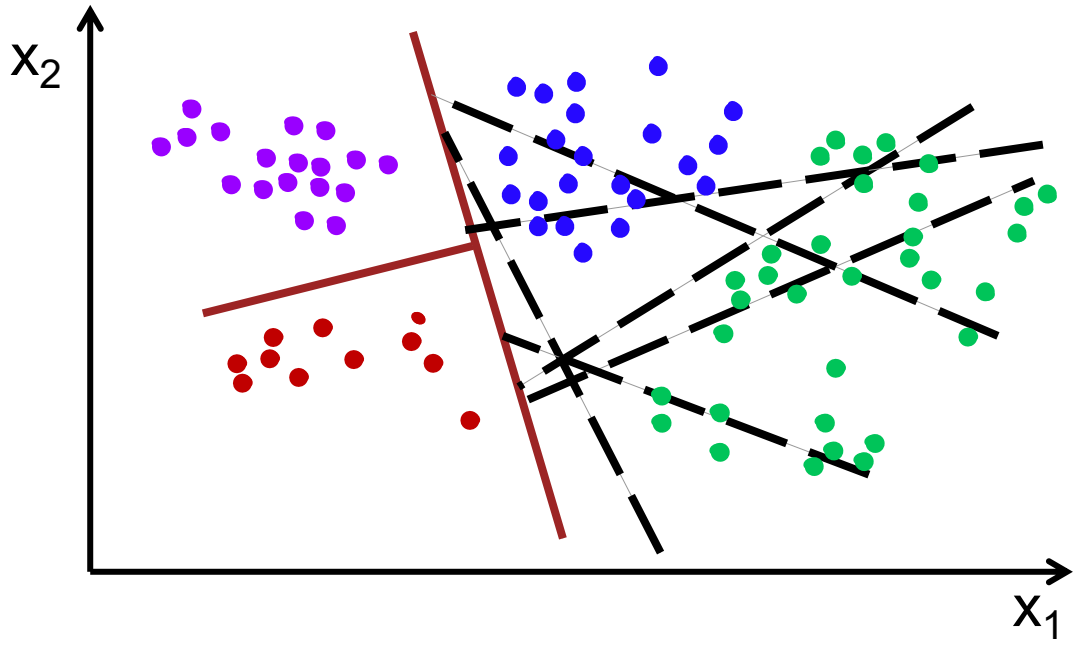




决策树的优化

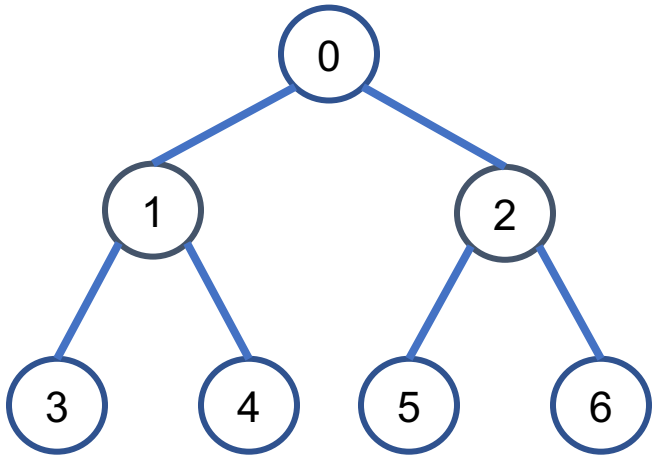
贪婪随机优化

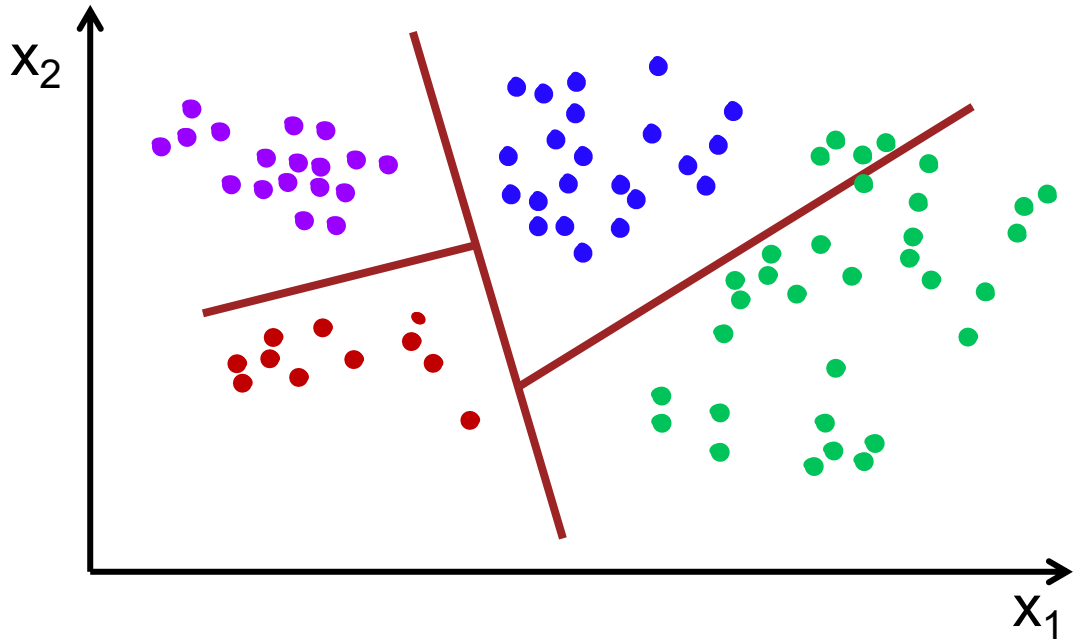




决策树的优化

贪婪随机优化





决策树的优化

At each node S_j, given a data attribute A_j

REPEAT

For each possible feature:

For each value of this feature in the data set:

Evaluate function $I(S_j, A_j)$ → This estimates the cost of splitting training data into left/right* components (S_j^1, S_j^2)

*: leaf 1, leaf 2

↓

Chose the best feature and value for splitting on

↓

Split data according to this function

until termination criterion is met, e.g.

- no more data to split
- maximum depth is reached

决策树的优化

At each node S_j, given a data attribute A_j

a random subset of

For ~~each possible~~ feature:

a random subset of values

For ~~each value~~ of this feature in the data set:

Evaluate function $I(S_j, A_j)$ → This estimates the cost of splitting training data into left/right* components (S_j^1, S_j^2)

*: leaf 1, leaf 2

Chose the best feature and value for splitting on

Split data according to this function

REPEAT

until termination criterion is met, e.g.

- no more data to split
- maximum depth is reached

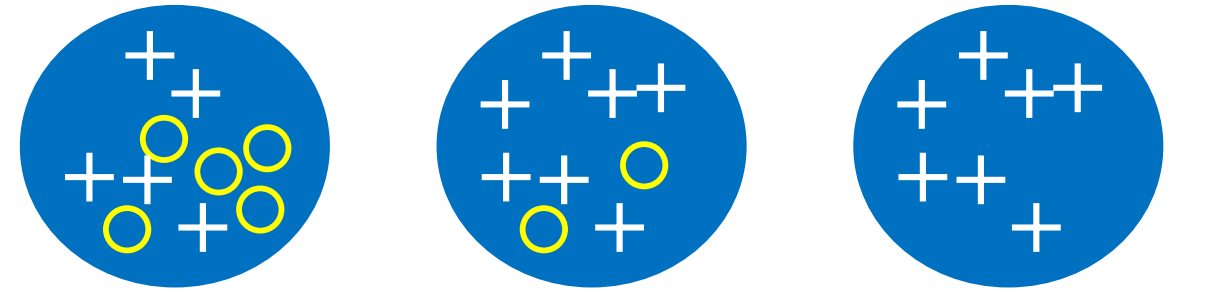
决策树的优化

- 有很多中损失函数的选择$I(S_j, A_j)$

- 最常见的一种是信息增益（Information gain）
  - 根据属性拆分数据集后熵的减少

熵（Entropy）

- 衡量一组样本中的杂质水平：



- 熵值越高，不确定性越大。

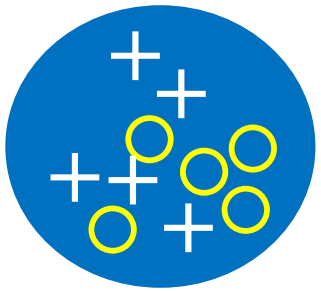
熵（Entropy）

Entropy is defined as

$$H = -\sum_{y_k \in Y} p(y_k) \log_2 p(y_k)$$

where $p(y_k) = p(y = k)$ is the probability of class k

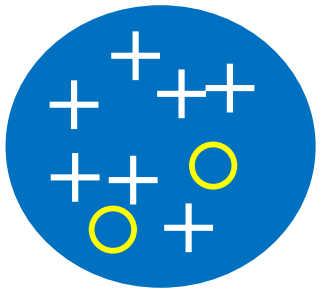
$p(y_1) = p(y_2) = 5/10 = 1/2$



Very impure group
-> maximum entropy

$H = -2 * \frac{1}{2} \log_2 \frac{1}{2} = 1$

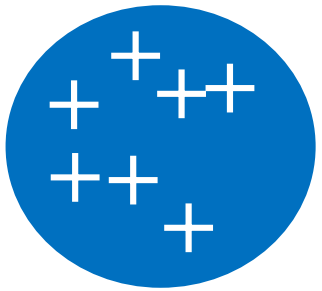
$p(y_1) = 7/9$
$p(y_2) = 2/9$



Less impure

$H = -(7/9 \log_2 7/9 + 2/9 \log_2 2/9) = 0.7642$

$p(y_1) = 7/7 = 1$
$p(y_2) = 0$



Maximum purity
-> minimum entropy

$H = -1 \log_2 1 = 0$

信息增益（Information Gain）

- 期望信息增益是指熵H从先验状态到考虑某些给定信息后的状态所发生的变化：

$$I(S_j, A_j) = H(S_j) - \sum_i \frac{|S_j^i|}{|S_j|} H(S_j^i)$$

- 也就是说

Information gain = Entropy(parent) – [weighted average entropy(children)]

信息增益（Information Gain）

$$I(S_j, A_j) = H(S_j) - \sum_i \frac{|S_j^i|}{|S_j|} H(S_j^i)$$

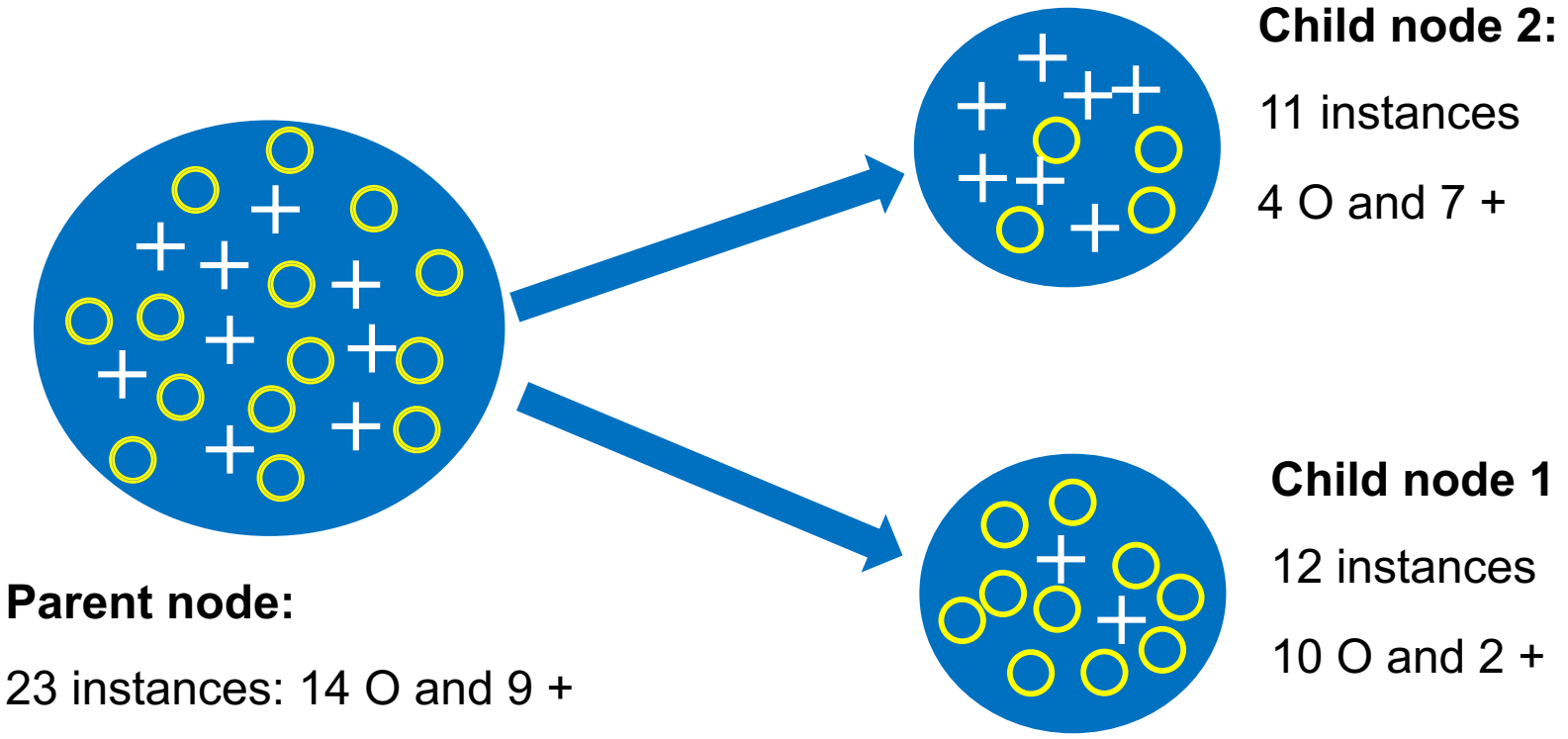
with

$$H(S) = -\sum_{y_k \in Y} p(y_k) \log_2 p(y_k)$$

- y = class labels i.e. for binary class $y = \{0, 1\}$
- $p(y_k)$ is probability of class k at given node/leaf;
- $|S_j|$= total number of data points reaching node j;
- $|S_j^i|$ = total number of data points reaching leaf i of node j;
- A_j = data attribute

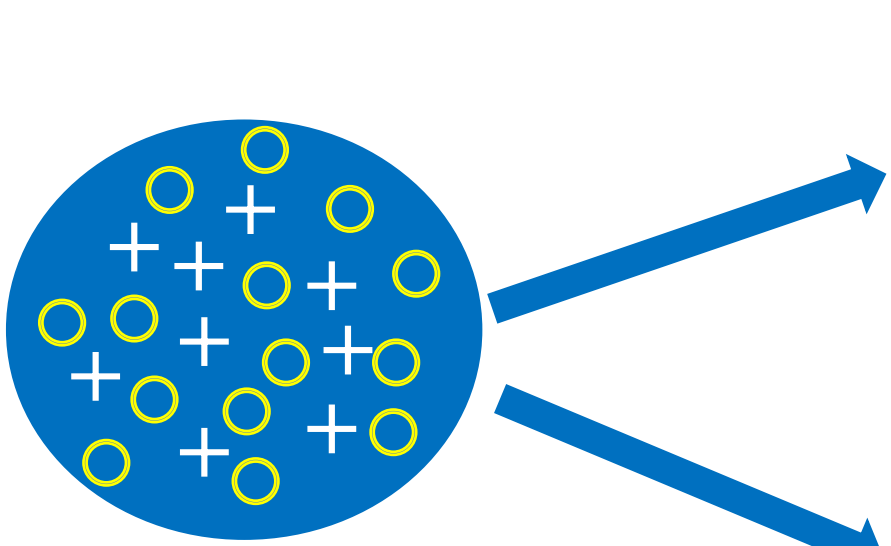
信息增益（Information Gain）

- 二进制分类问题的信息增益示例



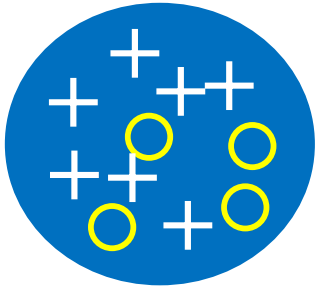
信息增益（Information Gain）

- 二进制分类问题的信息增益示例



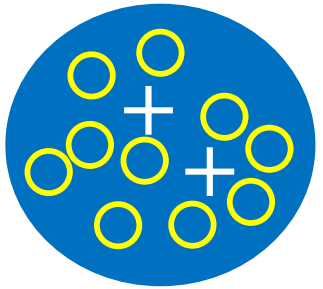
Parent node:

$$H(S_j) = -(\frac{14}{23}\log\left(\frac{14}{23}\right)) - (\frac{9}{23}\log\left(\frac{9}{23}\right)) = 0.966$$



Child node 2

$$H(S_j^2) = -(\frac{4}{11}\log\left(\frac{4}{11}\right)) - (\frac{7}{11}\log\left(\frac{7}{11}\right)) = 0.946$$



Child node 1

$$H(S_j^1) = -(\frac{10}{12}\log\left(\frac{10}{12}\right)) - (\frac{2}{12}\log\left(\frac{2}{12}\right)) = 0.650$$

信息增益（Information Gain）

- 子节点熵的加权和估计值：

$$\frac{12}{23} \times 0.650 + \frac{11}{23} \times 0.946 = 0.792$$

信息增益（Information Gain）

- 子节点熵的加权和估计值：

$$\frac{12}{23} \times 0.650 + \frac{11}{23} \times 0.946 = 0.792$$

$$I(S_j, A_j) = H(S_j) - \sum_i \frac{|S_j^i|}{|S_j|} H(S_j^i)$$

- 因此，这样分类的信息增益是

$$I(S_j, A_j) = 0.966 - 0.792 = 0.163$$

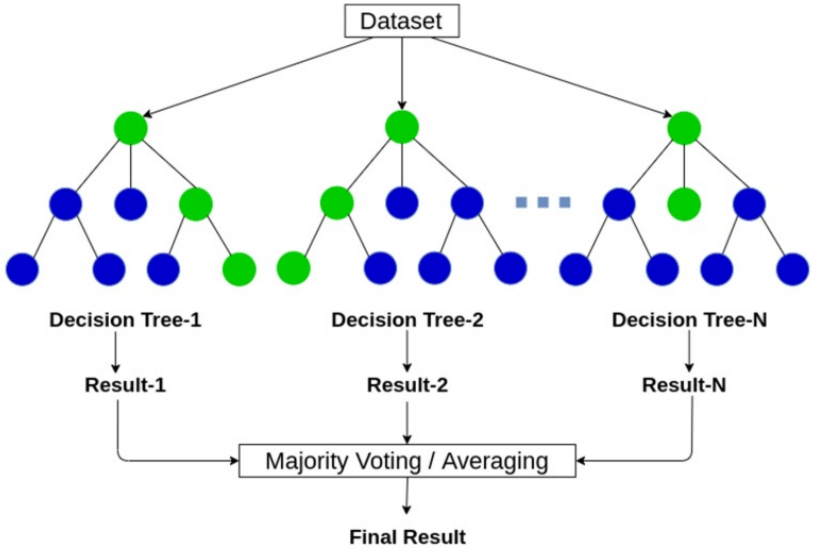
决策树作为弱学习器

- 单一决策树容易**过拟合**，尤其在未控制树深度的情况下，但通过集成多棵树能显著提升模型的鲁棒性。
- 决策树是同质集成学习中最受欢迎的**弱学习器**之一。
- 这类分类器的预测准确率通常难以与逻辑回归、K近邻等其他分类器相媲美

目录

随机森林 （Random Forest）

- **决策树的问题：** 单个决策树的判断规则可能**过于简单**
- **怎么解决：** **随机森林 = 很多个决策树一起预测**
  - **集成学习思想： 三个臭皮匠赛过诸葛亮**
  - **基础模型**： 每个“树”都是一个决策树 (简单模型)。
  - **最终预测**： 由所有决策树共同决定，更可靠！
- **预测方式 (综合决策):**
  - **回归问题 (预测数值):** 取所有决策树预测值的平均值
  - **分类问题 (预测类别):** 所有决策树**投票决定**，得票最多的类别胜出

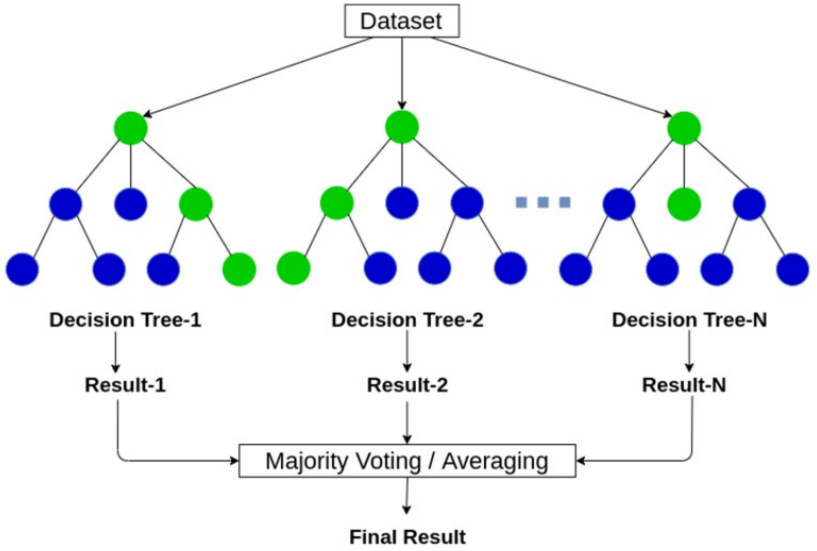


随机森林 （Random Forest）

- **随机森林强大的关键：集成学习**
- **为何要“组团”？** (差异性是关键):
  - **单棵树能力有限**： 容易片面，预测规则简单。
  - **森林更智慧**： 多棵树 “集思广益”，避免“一叶障目”
  - **前提条件：** **决策树之间要有“差异性”**
- **如何保证“差异性”？ ：引入随机性)**

  (1) 数据随机选取： 每棵树 只用一部分训练数据 来学习

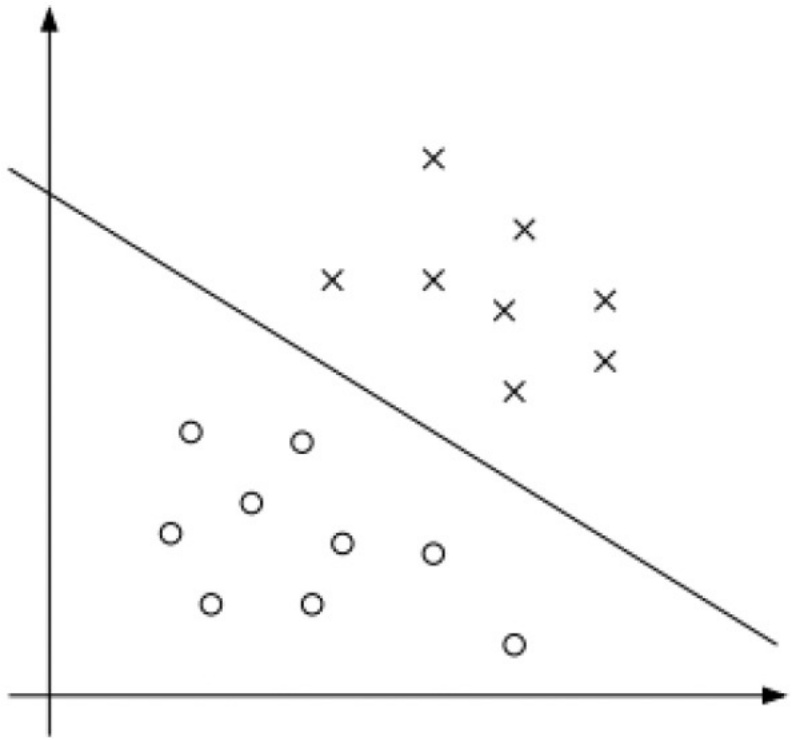
  (2) 特征随机选取： 每次分裂节点，只使用看一部分特征



目录

支持向量机 （SVM）

- **SVM的目标是：**发现一条线（或者超平面）可以将两个类别分开



支持向量机 （SVM）

- **SVM**与逻辑回归的对比：只关注锚点，对异常点不敏感

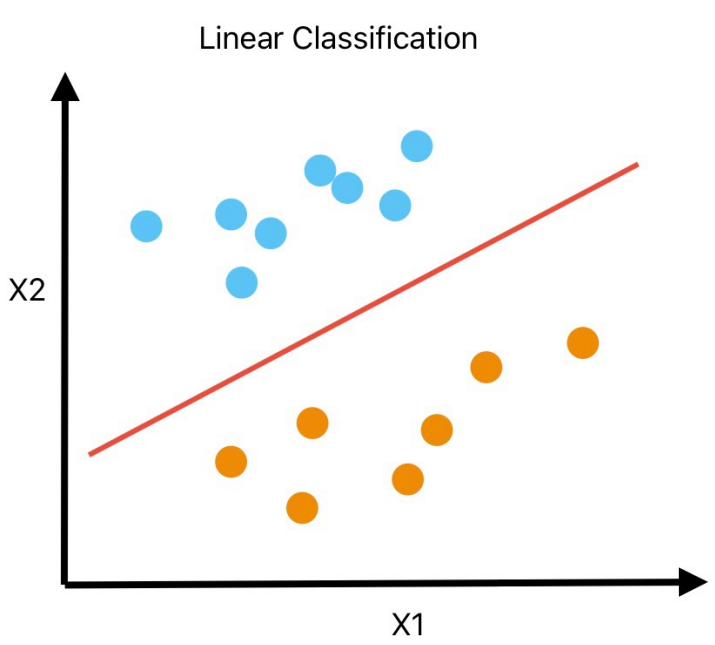
$$J_m(\boldsymbol{a}) = (\boldsymbol{y} - X^T\boldsymbol{a})^T(\boldsymbol{y} - X^T\boldsymbol{a})$$

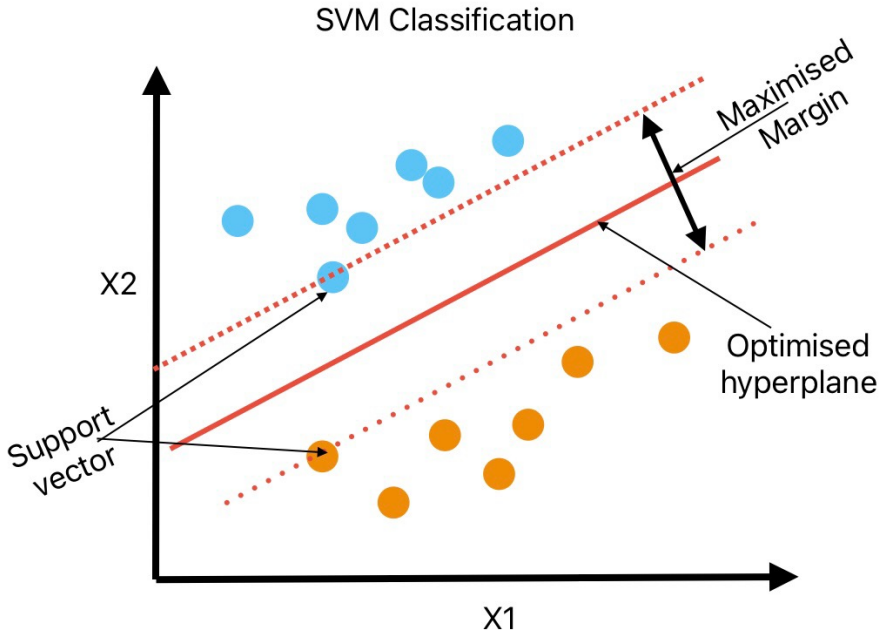
逻辑回归损失函数导数为0

$$XX^T\boldsymbol{a} = X\boldsymbol{y}$$

$$\boldsymbol{a} = (XX^T)^{-1}X\boldsymbol{y}$$

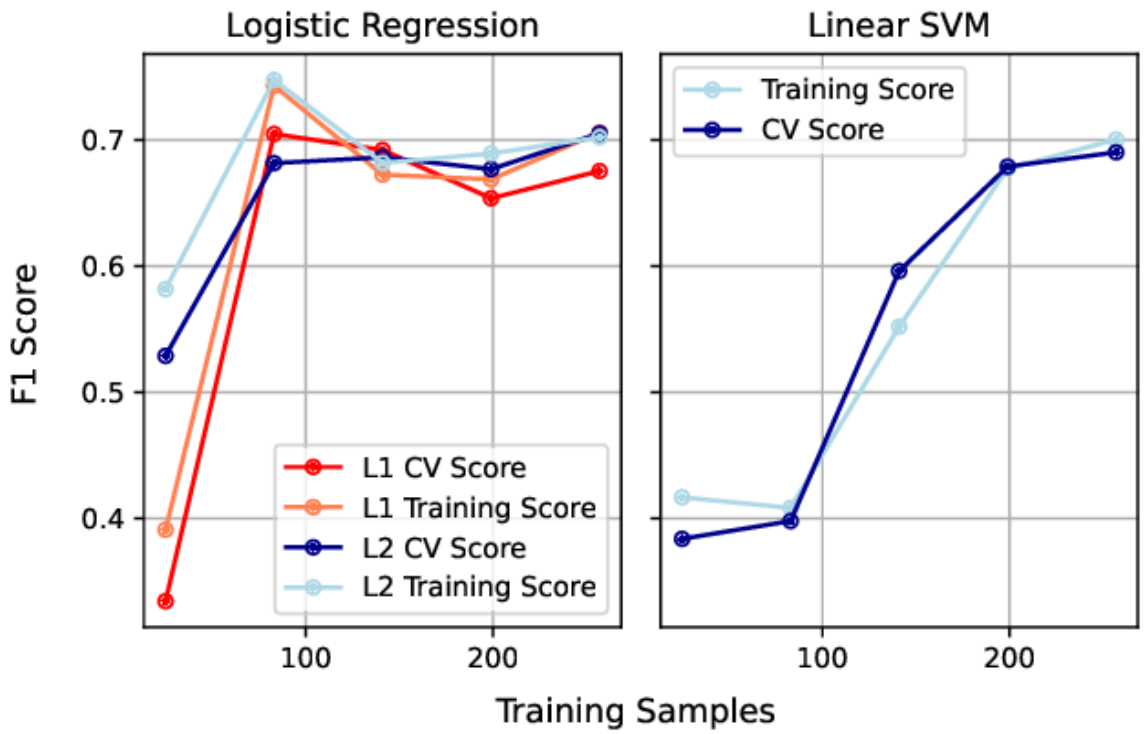
决策与所有样本（X）有关





支持向量机 （SVM）

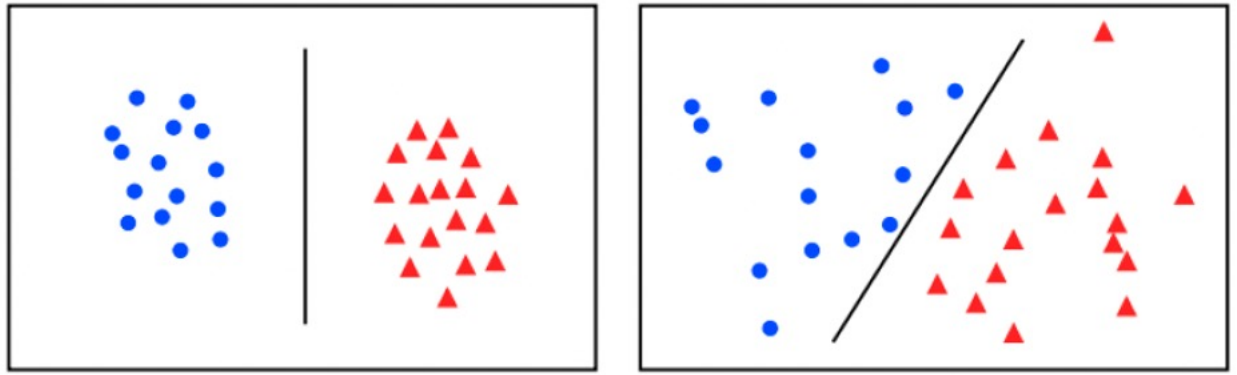
- **SVM**与线形回归的对比：只关注锚点，对异常点不敏感



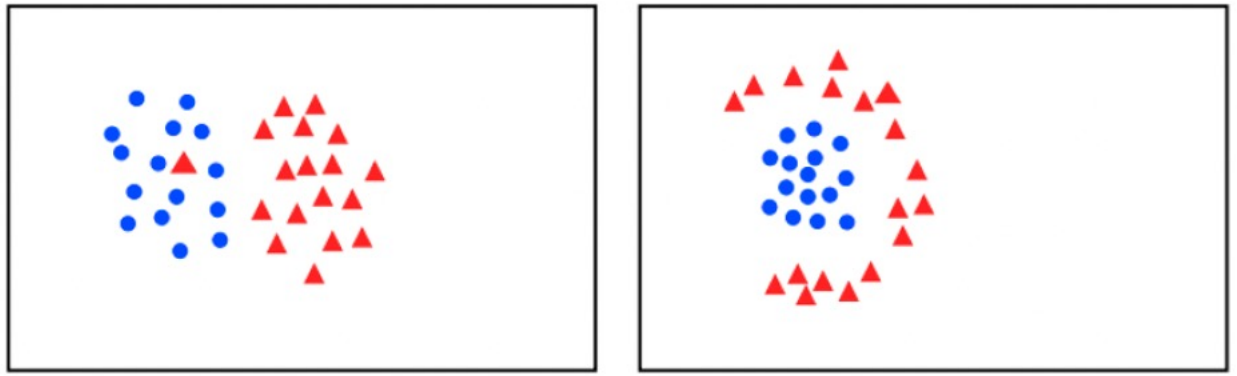
SVM：线性可分性

□ 线性可分 vs. 线性不可分

线性可分



线性不可分



SVM：决策超平面

- 在 D 维度中，一个分离超平面可以由一个法向量$\mathbf{w}$ 和一个截距 b 来定义

$$\mathbf{w}^T\mathbf{x}+b=0$$

$$w_1x_1+\ldots+w_Dx_D+b=0$$

- 经过点 $\mathbf{x}=\begin{bmatrix}x_1\\x_2\\\cdots\\x_D\end{bmatrix}$ 的超平面是：$\mathbf{w}=\begin{bmatrix}w_1\\w_2\\\cdots\\w_D\end{bmatrix}$

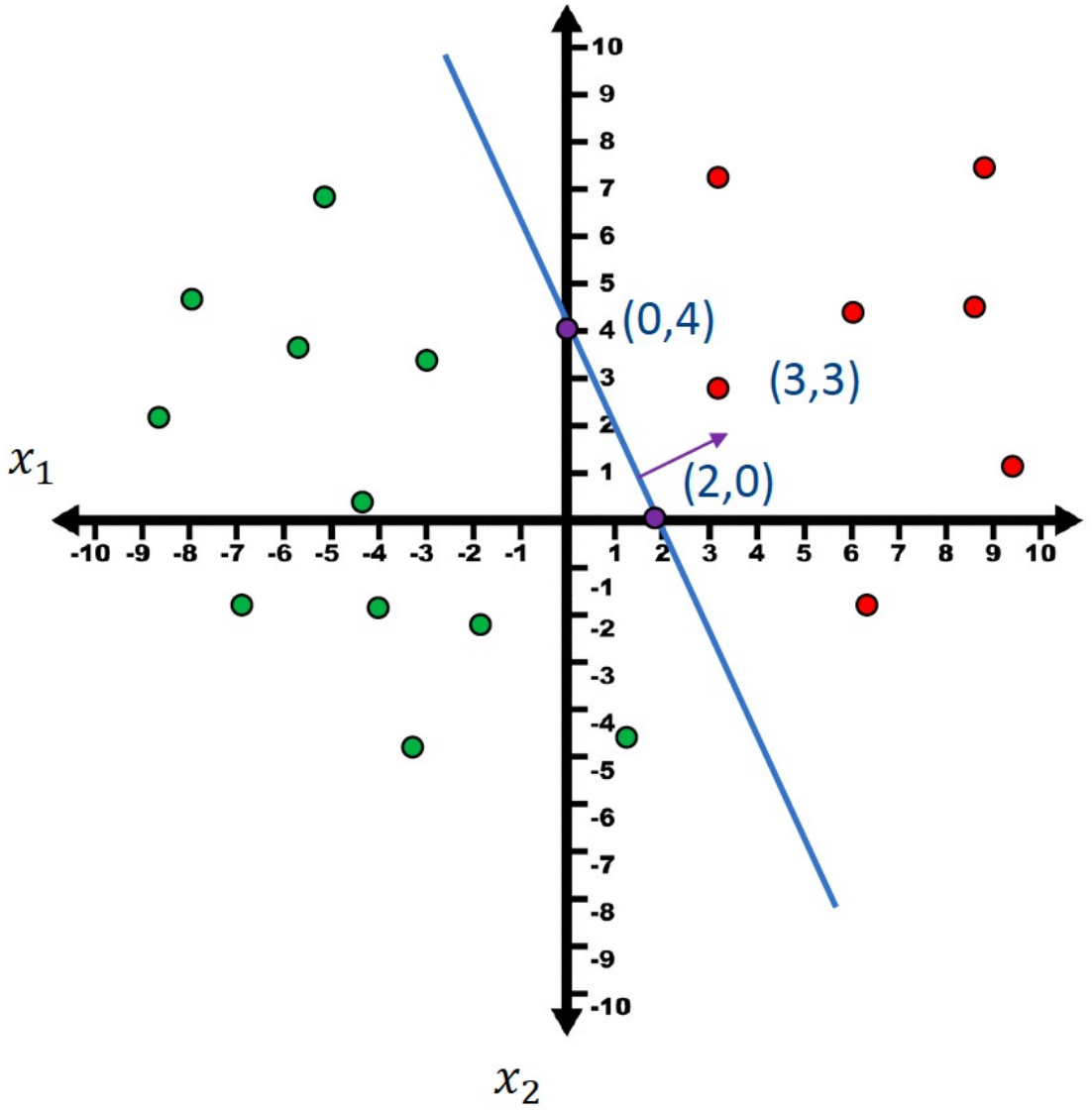
SVM: 线性模型的决策超平面

$$2x_1 + x_2 - 4 = 0$$

$$[2 \quad 1]\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - 4 = 0$$

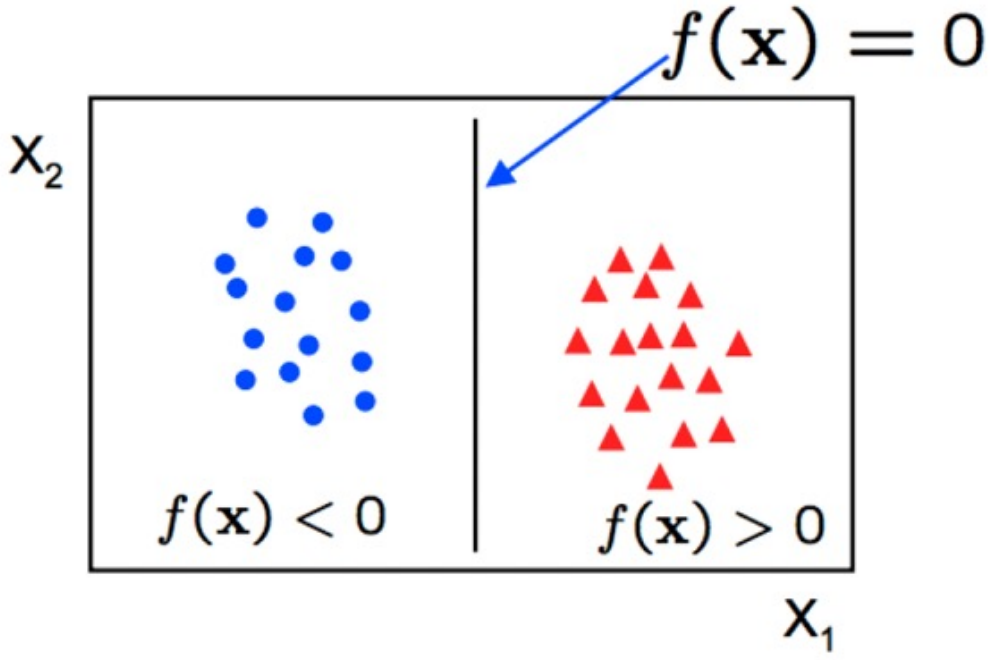
Normal vector

Intercept



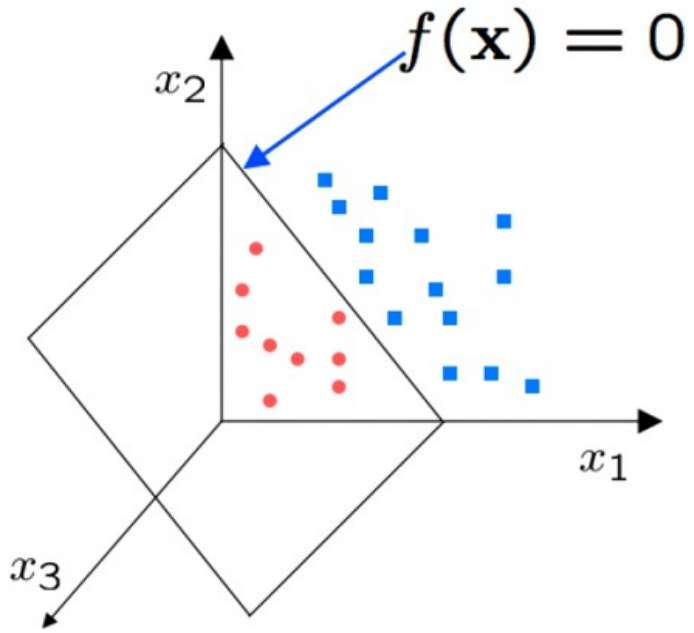
SVM：线性模型的决策超平面

- 线性分类器的形式为 $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$
- 在 2D 空间中，这是一条直线：



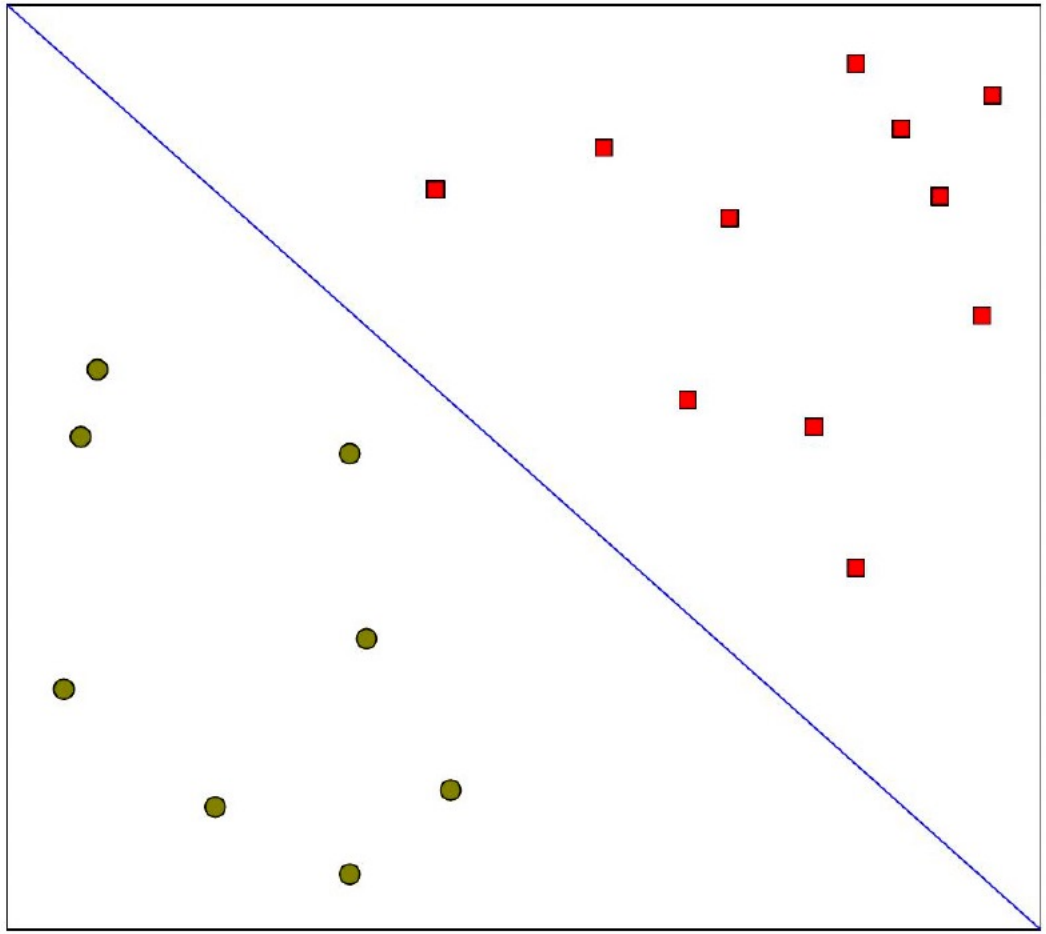
SVM：线性模型的决策超平面

- 线性分类器的形式为 $f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$
- 在 3D 空间中，这是一个平面：



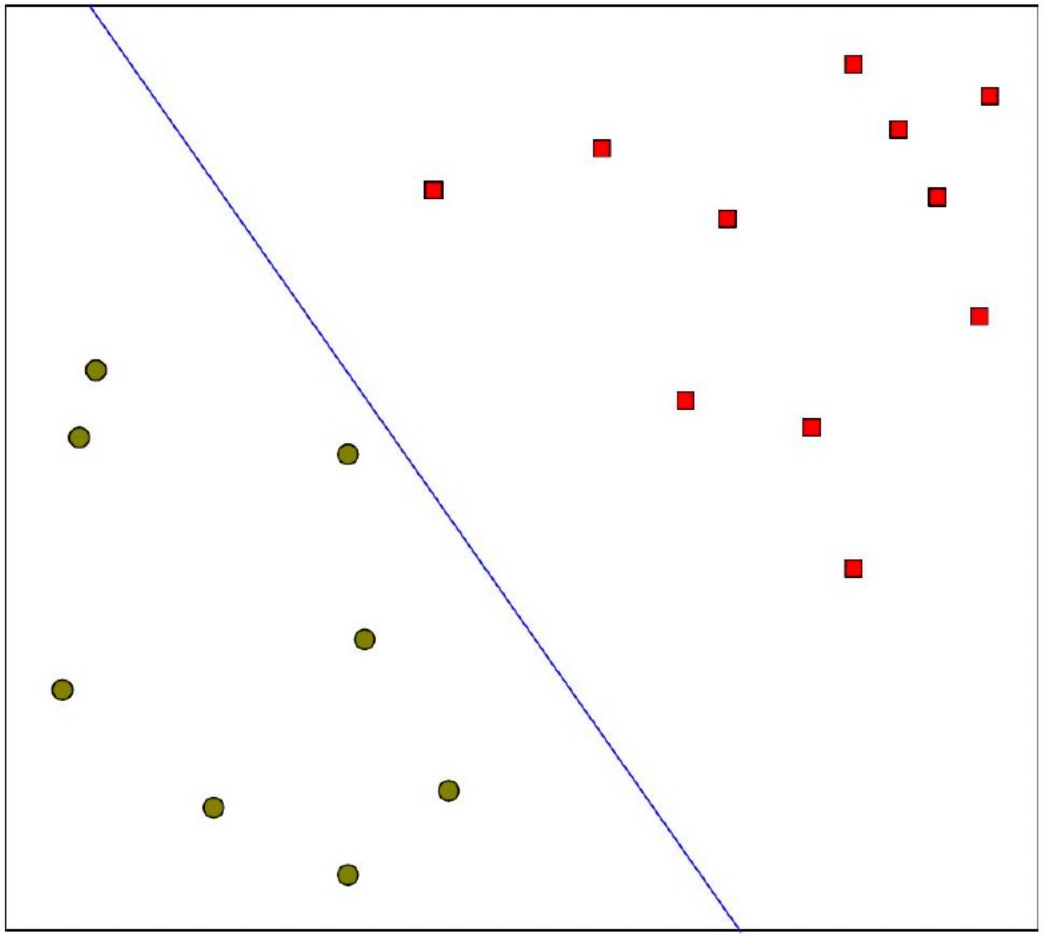
SVM的决策超平面

超平面1：



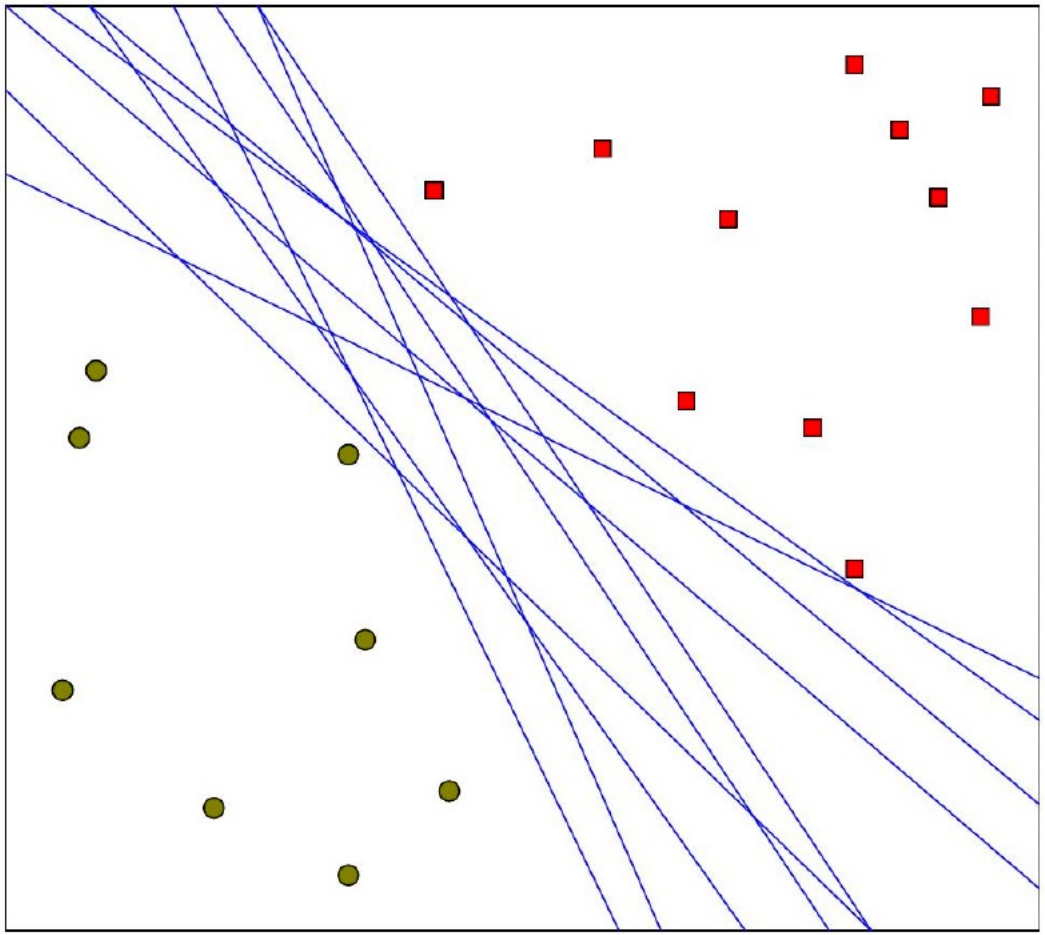
SVM的决策超平面

超平面2:



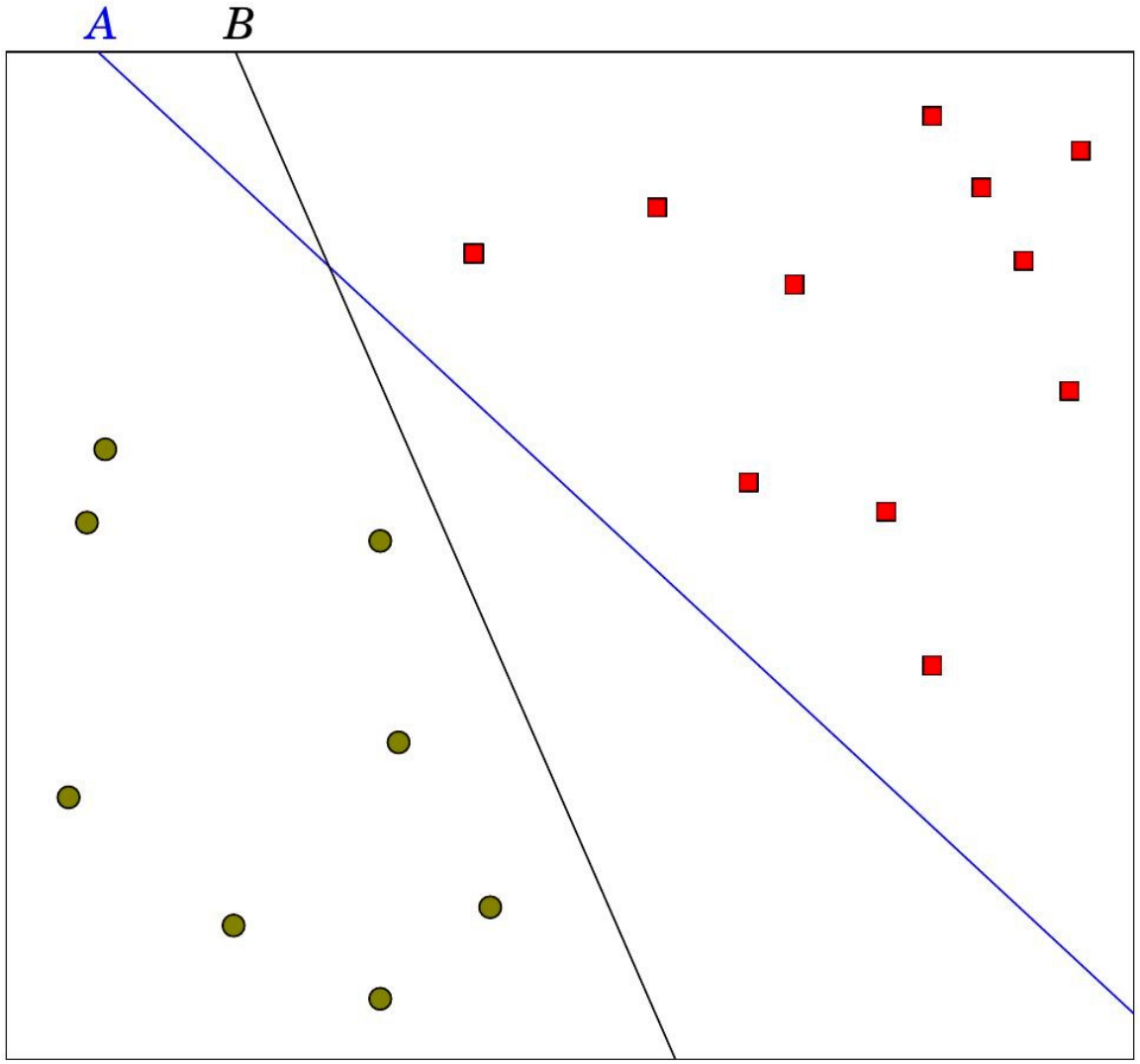
SVM的决策超平面

超平面n：



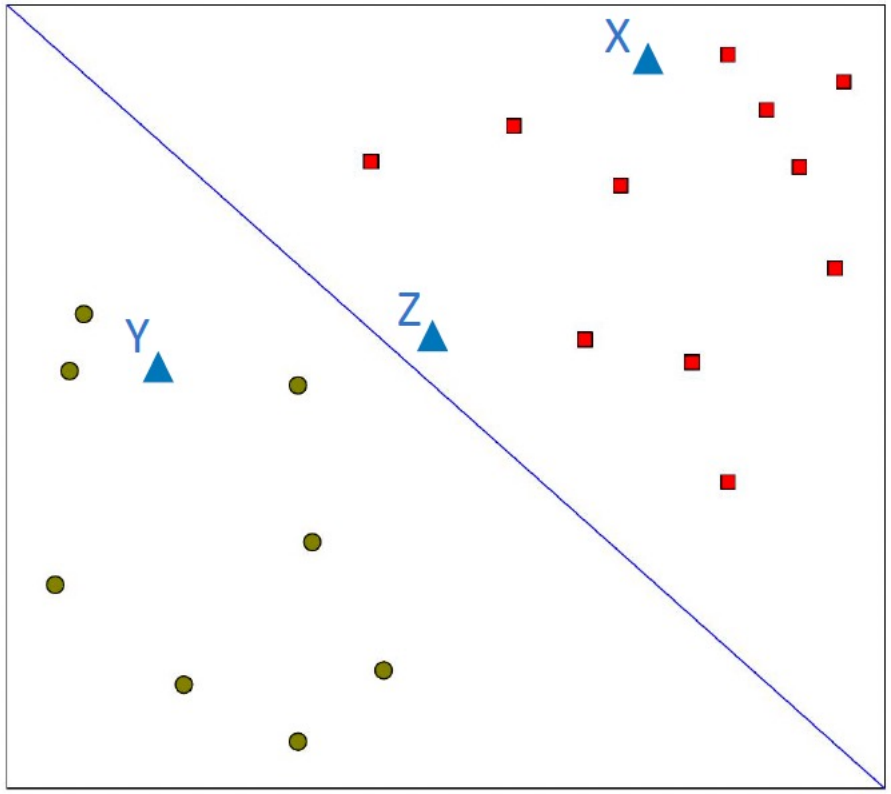
SVM的决策超平面

哪个更好?



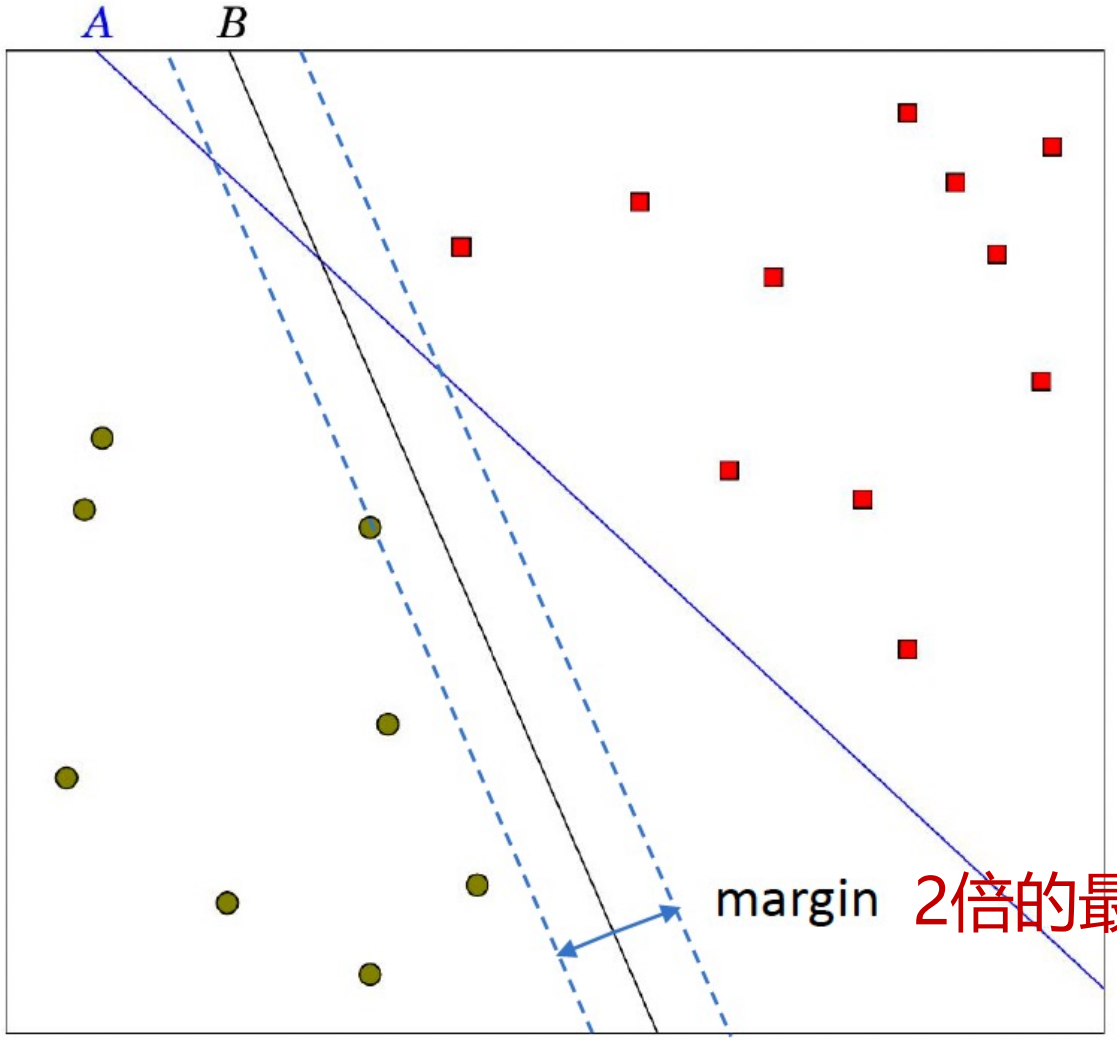
SVM的决策超平面

- 考虑一个数据点到决策超平面的距离



- X数据点离超平面很远，所以置信度更高
- Z数据点离超平面很近，小的变化就导致类别翻转，置信度低

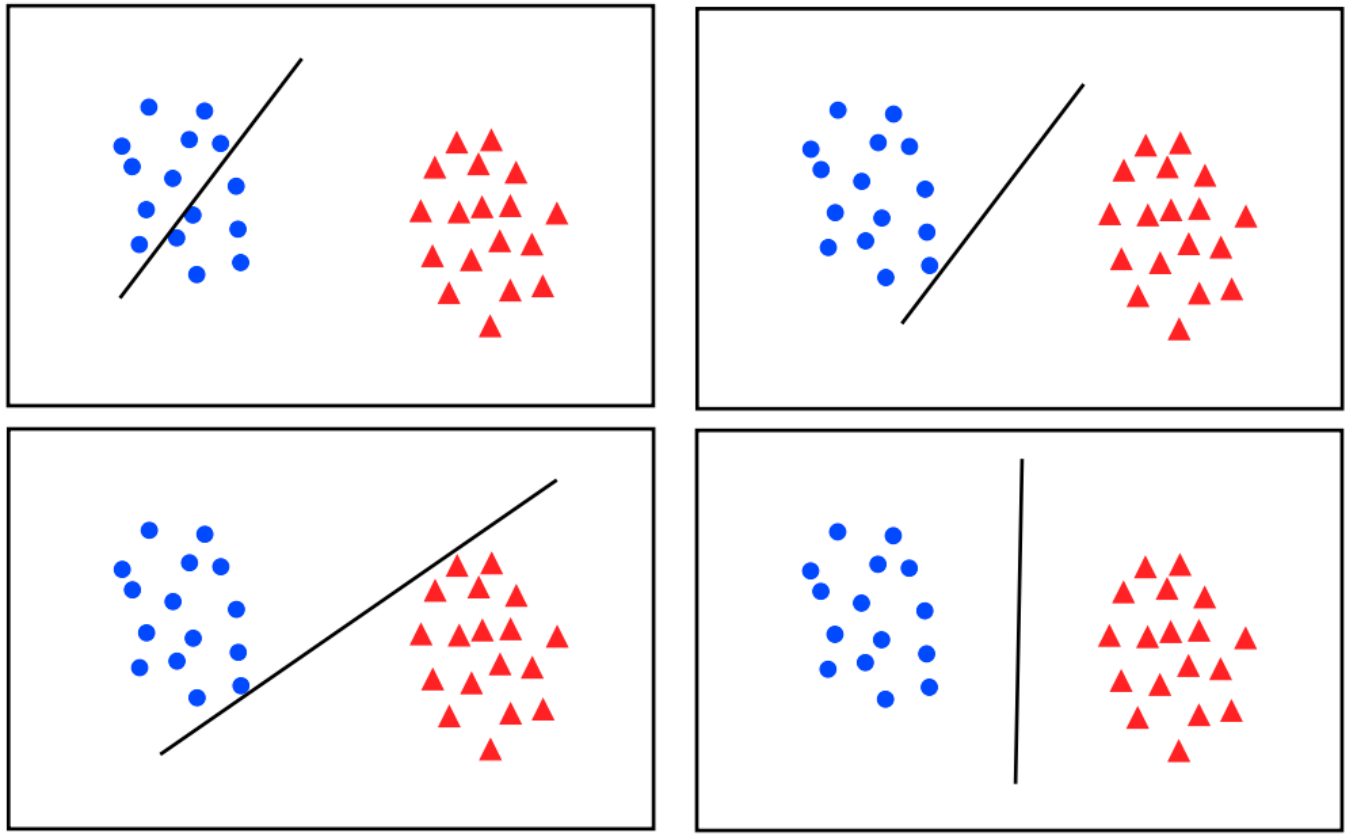
SVM： 决策边界最大化



2倍的最近的数据点到决策边界的距离

SVM：最优决策超平面

哪个决策超平面更优?



SVM: 公式定义

- **训练集**：N 个样本 $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$ 其中 $\mathbf{x}_i \in \mathbb{R}^D$ 且 $y_i \in \{-1, 1\}$。假设两个类别是线性可分的。

- **分隔两个类别的超平面**可以表示为：$\mathbf{w}^T \mathbf{x} + b = 0$

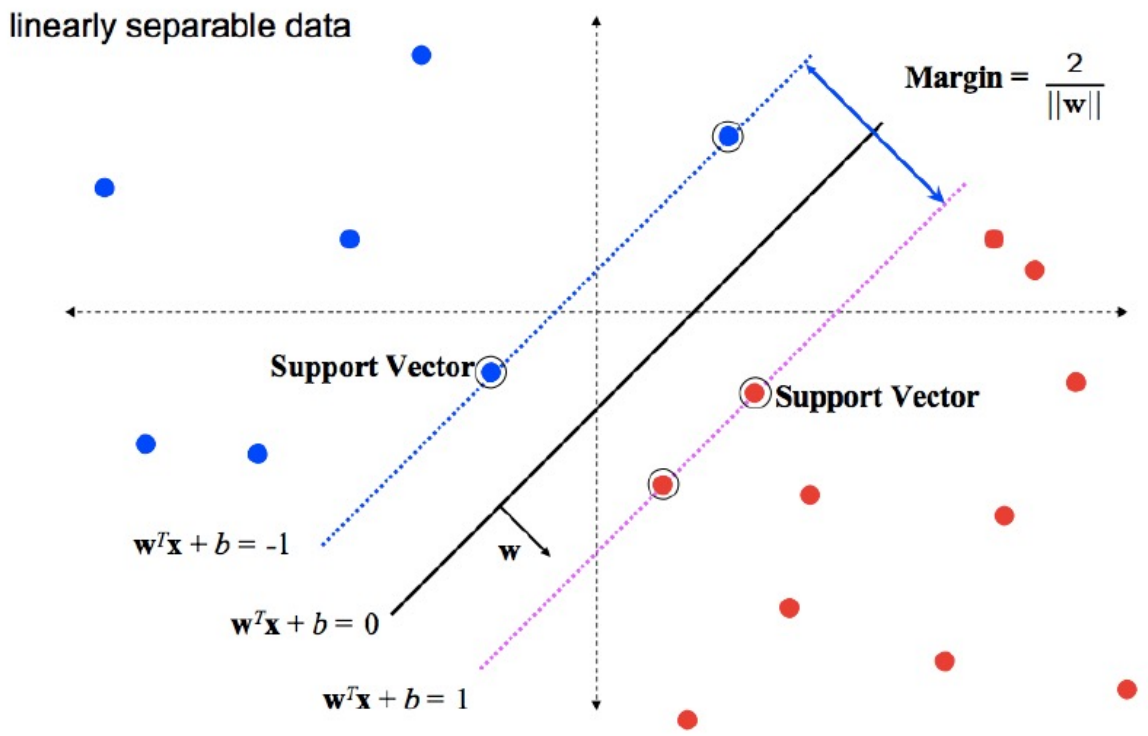
$$\mathbf{w}^T \mathbf{x}_i + b \geq 1 \text{ 对于 } y_i = +1$$

并且对于训练样本：$\mathbf{w}^T \mathbf{x}_i + b \leq -1 \text{ 对于 } y_i = -1$

$$\Longrightarrow y_i(\mathbf{w}^T \mathbf{x}_i + b) - 1 \geq 0$$

SVM: 支持向量

- **目标**：找到充当两个类别**边界**的数据点（即“**支持向量**”）
- 它们**约束**两个类别之间的**间隔**



SVM: 支持向量机优化目标

- SVM 的目标是找到一个能够**最大化两个类别之间间隔的分离超平面**
- 间隔 (Margin) 的大小定义为：$$\text{间隔} = \frac{2}{||\mathbf{w}||}$$

- **优化目标：**

  为了最大化间隔，等价于最小化 $\frac{1}{2}||\mathbf{w}||^2$

- SVM 优化问题公式化：

间隔最大化（注意分子分母调换了）

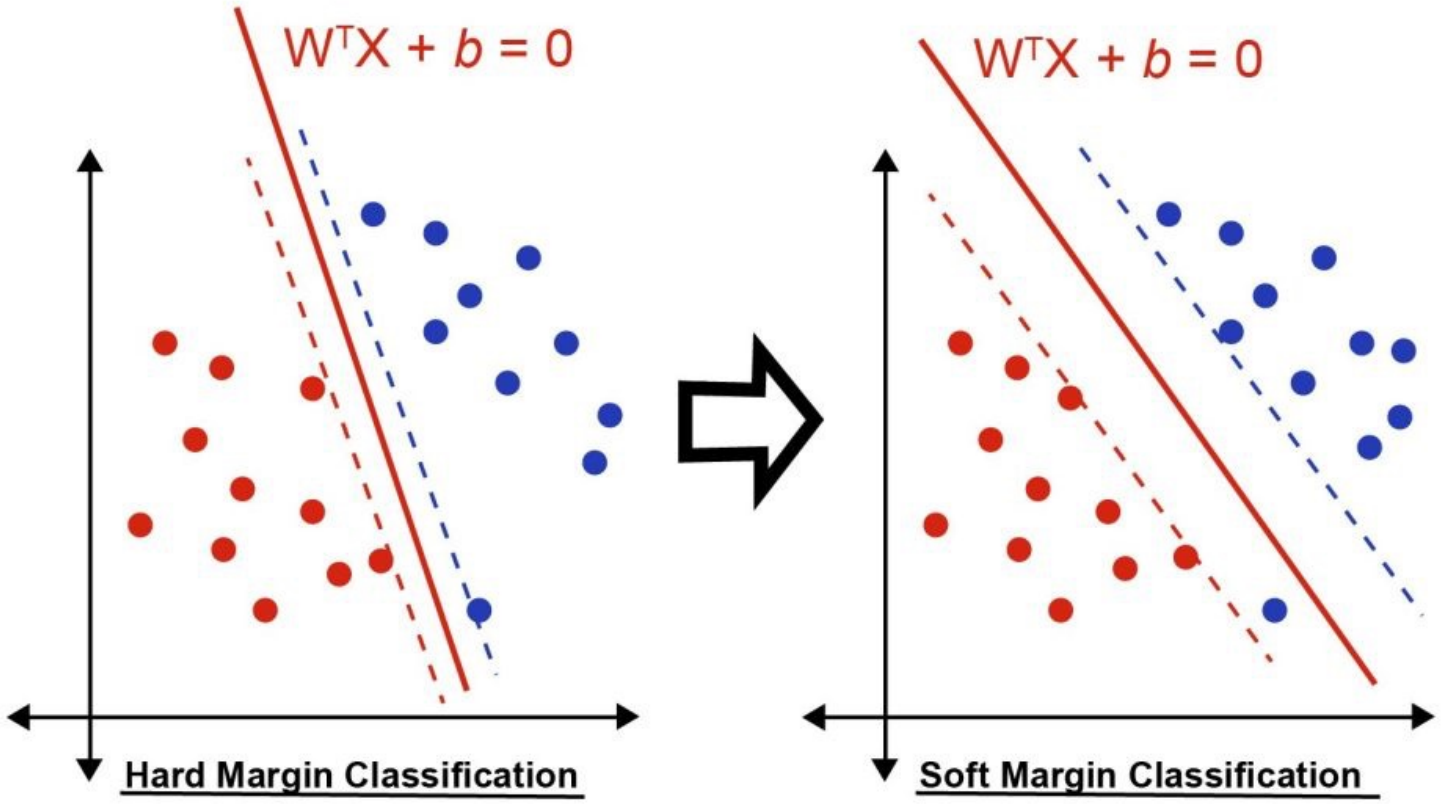
保证分类正确

$$\underset{\mathbf{w},b}{\text{minimize}} \quad \frac{1}{2}||\mathbf{w}||^2$$

$$s.t. \quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, N$$

SVM: 支持向量机优化目标

- Hard Margin to Soft Margin SVM



SVM: 支持向量机优化目标

- 增加阈值的调节项

The Soft Margin SVM

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1-\xi_i,$$

$$\xi_i \geq 0$$

for $i = 1, \ldots, N$.

C: regularization parameter. $\|\mathbf{w}\|^2$: the regularizer.

SVM优化：对偶形式

$$\begin{aligned} \min_{\mathbf{w},b,\boldsymbol{\xi}} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ \text{subject to} \quad & y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) \geq 1-\xi_i, \\ & \xi_i \geq 0 \end{aligned}$$

- We use $\alpha_i \geq 0$ and $\gamma_i \geq 0$ as the Lagrange multipliers.
  - α_i: w.r.t. the constraint that examples are correctly classified.
  - γ_i: w.r.t. the non-negativity constraint of the slack variable.

$$\begin{aligned} \mathfrak{L}(\mathbf{w}, b, \xi, \alpha, \gamma) \quad := \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i \\ & - \sum_{i=1}^{N}\alpha_i(y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) - 1 + \xi_i) - \sum_{i=1}^{N}\gamma_i\xi_i \end{aligned}$$

SVM优化：对偶形式

$$\mathfrak{L} = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) - 1 + \xi_i) - \sum_{i=1}^{N}\gamma_i\xi_i$$

$$\begin{aligned}
\frac{\partial \mathfrak{L}}{\partial \mathbf{w}} &= \mathbf{w}^\top - \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i^\top \\
\frac{\partial \mathfrak{L}}{\partial b} &= -\sum_{i=1}^{N}\alpha_i y_i \\
\frac{\partial \mathfrak{L}}{\partial \xi_i} &= C - \alpha_i - \gamma_i
\end{aligned}$$

SVM优化：对偶形式

$$\mathfrak{L} = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N}\xi_i - \sum_{i=1}^{N}\alpha_i(y_i(\langle \mathbf{w}, \mathbf{x}_i\rangle + b) - 1 + \xi_i) - \sum_{i=1}^{N}\gamma_i\xi_i$$

$$\begin{aligned}
\frac{\partial \mathfrak{L}}{\partial \mathbf{w}} &= \mathbf{w}^\top - \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i^\top \\
\frac{\partial \mathfrak{L}}{\partial b} &= -\sum_{i=1}^{N}\alpha_i y_i \\
\frac{\partial \mathfrak{L}}{\partial \xi_i} &= C - \alpha_i - \gamma_i
\end{aligned}$$

- Maximizing the Lagrangian by setting $\frac{\partial \mathfrak{L}}{\partial \mathbf{w}} = \mathbf{0}^\top$,

$$\mathbf{w} = \sum_{i=1}^{N}\alpha_i y_i \mathbf{x}_i.$$

- The optimal weight vector is a linear combination of the examples $\mathbf{x}_i$'s.
- $\mathbf{x}_i$'s with $\alpha_i > 0$: support vectors.

SVM优化：对偶形式

Substituting the expression for $\mathbf{w}$ into the Lagrangian, we have

$$\begin{aligned}\mathfrak{D}(\xi,\alpha,\gamma) \;:=\; & \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^{N} y_i \alpha_i \left\langle \sum_{j=1}^{N} y_j \alpha_j \mathbf{x}_j,\, \mathbf{x}_i \right\rangle \\ & + C\sum_{i=1}^{N}\xi_i - b\sum_{i=1}^{N} y_i \alpha_i - \sum_{i=1}^{N}\alpha_i - \sum_{i=1}^{N}\alpha_i \xi_i - \sum_{i=1}^{N}\gamma_i \xi_i .\end{aligned}$$

- No terms involving the primal variable $\mathbf{w}$.

SVM优化：对偶形式

The Dual SVM

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^{N} \alpha_i$$

$$\text{subject to} \quad \sum_{i=1}^{N} y_i \alpha_i = 0,$$

$$0 \le \alpha_i \le C \quad \text{for all} \quad i = 1, \ldots, N.$$

- $\boldsymbol{\alpha} = [\alpha_1, \cdots, \alpha_N]^\top \in \mathbb{R}^N$: Lagrange multipliers.

SVM: 核函数

The Dual SVM

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^{N} \alpha_i$$

$$\text{subject to} \quad \sum_{i=1}^{N} y_i \alpha_i = 0,$$

$$0 \le \alpha_i \le C \quad \text{for all} \quad i = 1, \ldots, N.$$

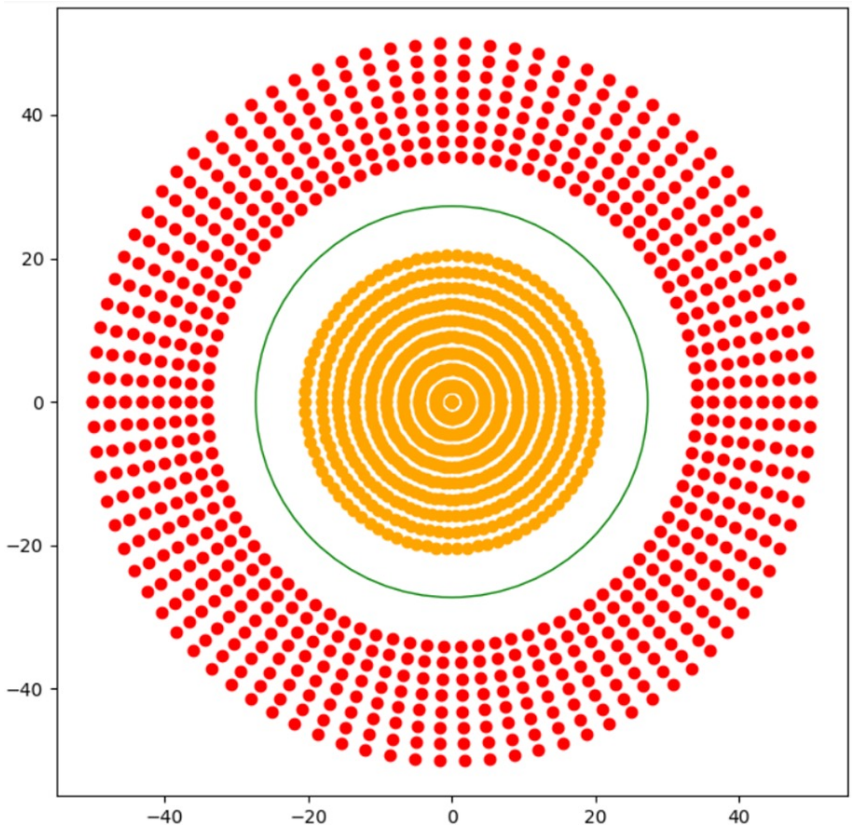
- We can see the inner product occurs only between examples. No inner products between examples and parameters!

Remark

- The primal SVM: # optimization variables: feature dimension D.
- The dual SVM: # optimization variables: the number N of examples.

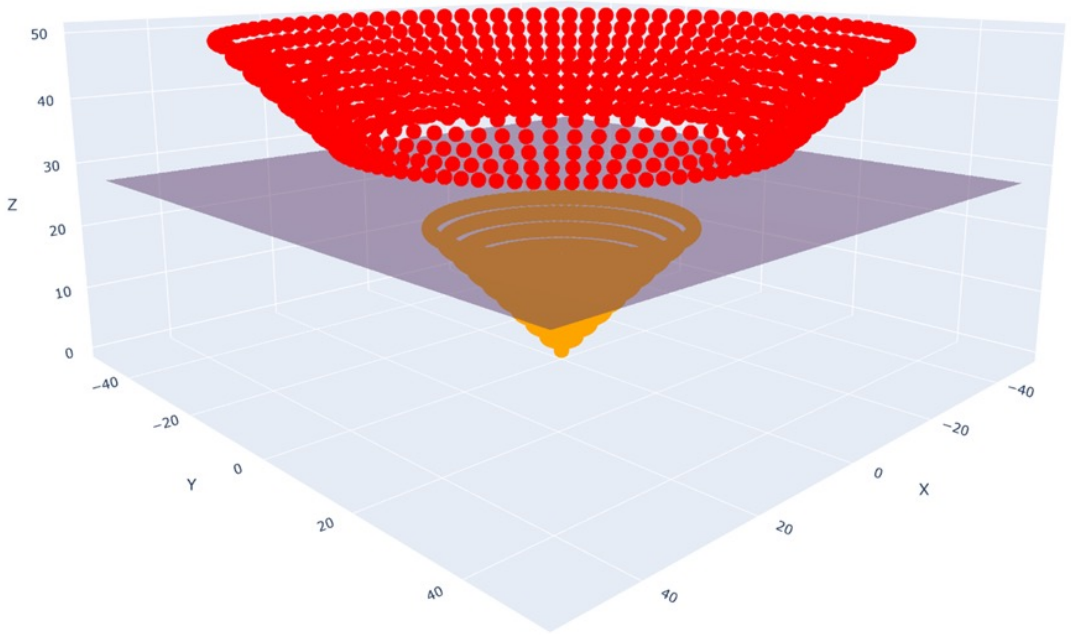
SVM: 核函数

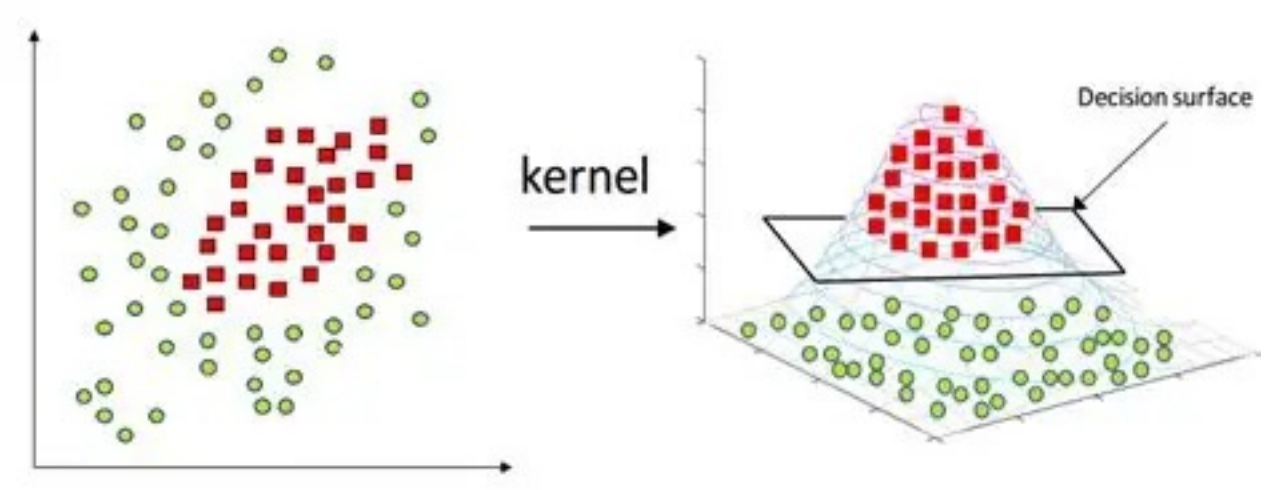
数据线性不可分怎么办?



SVM: 核函数

给数据增加新维度使其线性可分：





SVM: 核函数

The Dual SVM

$$\min_{\alpha} \quad \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} y_i y_j \alpha_i \alpha_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle - \sum_{i=1}^{N} \alpha_i$$

$$\text{subject to} \quad \sum_{i=1}^{N} y_i \alpha_i = 0,$$

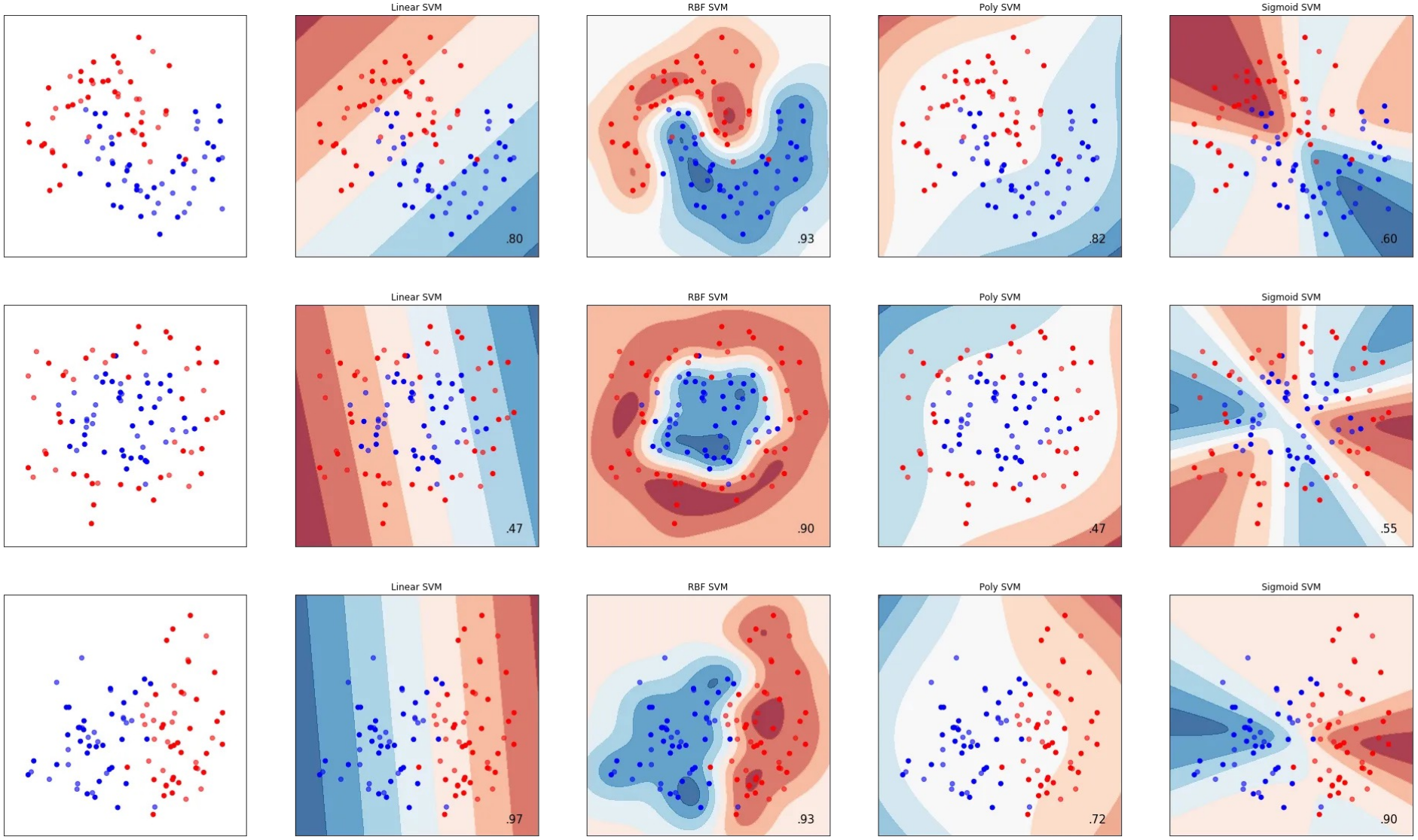
$$0 \le \alpha_i \le C \quad \text{for all} \quad i = 1, \ldots, N.$$

- We can see the inner product occurs only between examples. No inner products between examples and parameters!
- Kernel trick: consider $\phi(\mathbf{x}_i)$ to represent $\mathbf{x}_i$ ($\phi : \mathcal{X} \mapsto \mathcal{H}$).
- Consider a similarity function $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle_{\mathcal{H}}$ instead of defining $\phi(\cdot)$ and computing the resulting inner product.

SVM: 核函数

- 不同SVM核函数 （Kernels）是SVM相比逻辑回归的核心优势
- 核函数计算两个样本的距离，增加维度让分类更容易
  - linear $k(\mathbf{x}_1, \mathbf{x}_2) = \mathbf{x}_1 \cdot \mathbf{x}_2$
  - polynomial $k(\mathbf{x}_1, \mathbf{x}_2) = (\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)^d$
  - Gaussian or radial basis $k(\mathbf{x}_1, \mathbf{x}_2) = \exp\left(-\gamma \left\| \mathbf{x}_1 - \mathbf{x}_2 \right\|^2\right)$
  - sigmoid $k(\mathbf{x}_1, \mathbf{x}_2) = \tanh(\gamma\, \mathbf{x}_1 \cdot \mathbf{x}_2 + c)$

SVM: 核函数

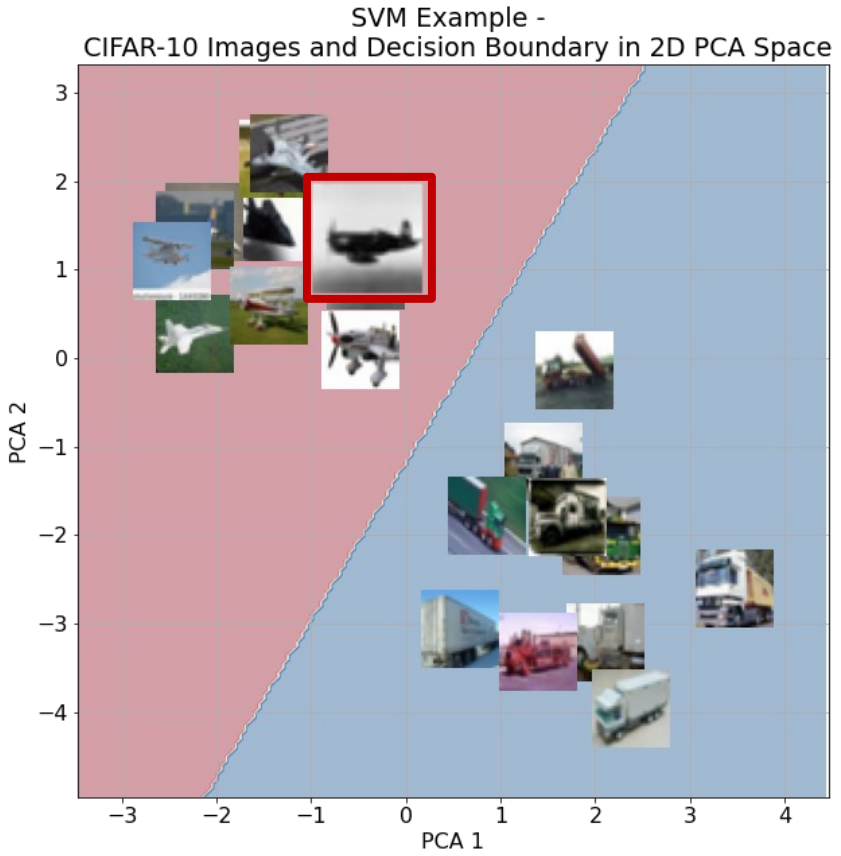

Linear SVM
RBF SVM
Poly SVM
Sigmoid SVM
.80
.93
.82
.60
Linear SVM
RBF SVM
Poly SVM
Sigmoid SVM
.47
.90
.47
.55
Linear SVM
RBF SVM
Poly SVM
Sigmoid SVM
.97
.93
.72
.90

举例：用SVM进行图片分类

① **选定待分类图像:** 一张待分类的 CIFAR-10 图像

② **SVM 决策:** 计算**决策值**并判断类别:

- **简化决策函数:** 决策值 = (w1 * 特征1) + (w2 * 特征2) + b
- **示例参数:** w1=0.5, w2=-0.8, b=0.2
- **待分类图像特征:** 特征1=-0.5, 特征2=1.2
- **计算:** 决策值 = (0.5 * -0.5) + (-0.8 * 1.2) + 0.2 = -1.11
- **分类规则:** 决策值 ≥ 0 => “卡车”, 决策值 < 0 => “飞机”
- **结果:** 决策值=-1.11 => 预测类别: **飞机**



谢谢！