

机器学习与深度学习 ——支持向量机（SVM）算法



Personal Website: <https://www.miaopeng.info/>



Email: miaopeng@stu.scu.edu.cn



Github: <https://github.com/MMeowhite>



Youtube: <https://www.youtube.com/@pengmiao-bmm>

目录章节

CONTENTS

01 SVM的概念与原理

02 模型求解与评估

03 模型扩展与改进

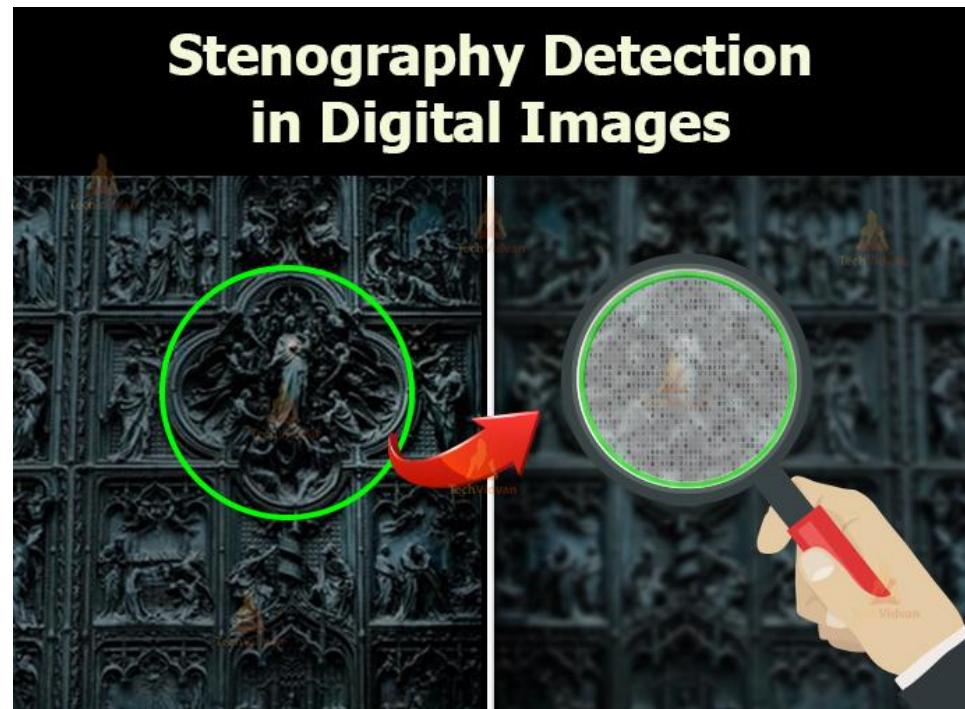
04 模型实现与实战

05 总结

▶ 为什么学SVM?

- SVM 是机器学习中最经典的分类算法之一，在图像识别、文本分类、医学诊断等场景都被广泛应用。如文本分类、图像识别、生物信息学、金融风控、医疗诊断、异常检测等。
- 它在小样本、高维特征的数据场景下表现优异，应用广泛，如文本分类、图像识别和医疗诊断。

Facial Expression Classification using SVM

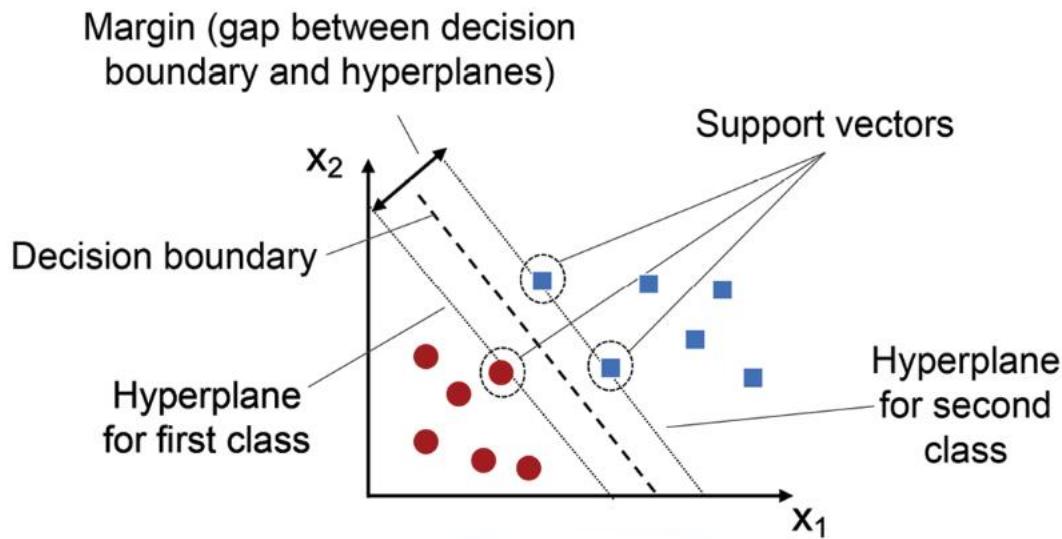


source: <https://techvidvan.com/tutorials/svm-applications/>

SVM 是机器学习的第四个台阶，代表了从“简单线性分类”到“核方法与最大间隔”的跃升。

► 什么是SVM?

- SVM (支持向量机, Support Vector Machine) 是一种常用的**监督学习**算法, 主要用于分类, 也可扩展到回归。
- 本质: 通过**最大化间隔**来寻找**最优分类边界**的算法, 并能借助**核函数**解决**非线性**问题。

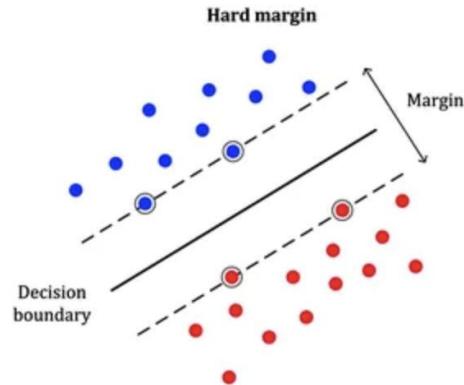


- **间隔:** 在二分类任务中, 我们用一个超平面 (直线/平面/高维平面) 把两类数据分开。间隔就是这个超平面到最近样本点 (支持向量) 的距离。
- **最大化间隔:** 在所有能把两类数据分开的超平面里, 选择间隔最大的那个超平面。优点: 边界离样本更远, 模型对噪声和新样本更稳健 (泛化能力更强)。
- **最优分类边界:** 即实现了最大间隔化的那条超平面。它由支持向量 (离边界最近的点) 决定。一旦确定支持向量, 边界的位置和方向也随之唯一确定。
- **核函数:** SVM可以通过核函数把线性不可分的问题映射到高维空间, 使其变得线性可分。

► SVM的前置知识

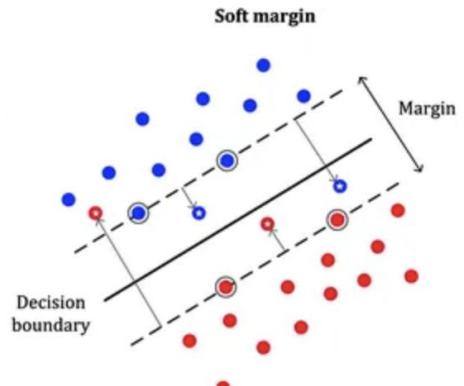
➤ 1) SVM的基本形式是一个有监督的线性二分类模型，它是间隔最大化的分类器。主要包括以下几种形式：

- 硬间隔SVM：数据完全线性可分；约束条件为所有样本必须被完全正确分类：



$$\min_{w,b} \frac{1}{2} \|w\|^2, \quad y_i(w^T x_i + b) \geq 1$$

- 软间隔SVM：数据近似线性可分，存在噪声或少量误分；可引入松弛变量允许部分样本违背约束：



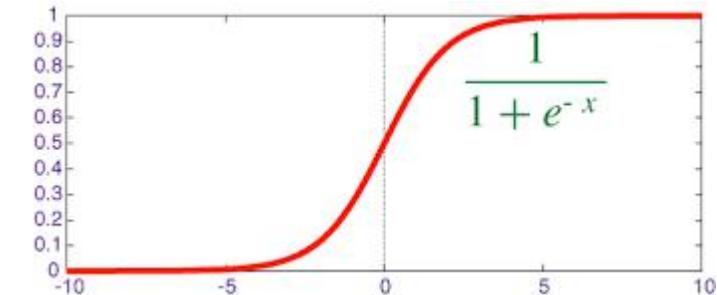
$$\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_i \xi_i, \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0$$

- 核SVM：数据线性不可分。核心思想为用核函数 $K(x,z) = \phi(x)^T \phi(z)$ 将数据隐式映射到高维特征空间，决策函数： $f(x) = \text{sign}(\sum_{i \in SV} \alpha_i y_i K(x_i, x) + b)$ 常用核函数：线性核、多项式核、RBF（高斯）核、Sigmoid 核。

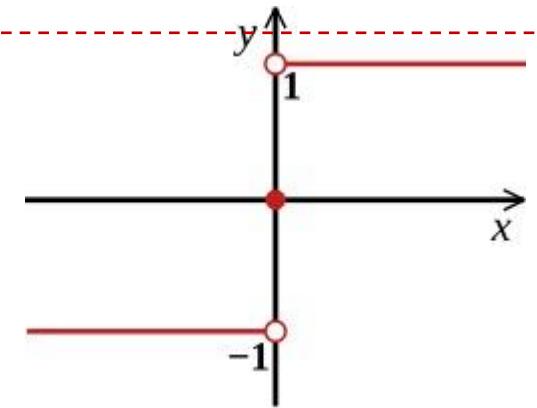
► SVM的前置知识

➤ 注意SVM的y的取值只能是-1（错误分类），+1（正确分类），不同于逻辑回归的y的取值为0（错误分类），+1（正确分类）

- 逻辑回归的 $y \in \{0, +1\}$:
- 逻辑回归是 概率模型，建模目标是: $P(y = 1|x) = \sigma(w^T x + b)$
- 判别函数为 $\sigma(\cdot)$ 是 Sigmoid，输出范围[0,1]。
- 所以自然把标签定义为 0（负类）和 1（正类），对应概率事件的真假。



- SVM的 $y \in \{-1, +1\}$:
- SVM是间隔最大化模型，不是概率模型。
- 判别函数为 $f(x) = \text{sign}(w^T x + b)$ 这里sign函数输出范围为-1或+1。
- 更重要的是，SVM 的约束写法需要: $y_i(w^T x_i + b) \geq 1$



若 $y \in \{0,1\}$ ，这个约束就失去了对称性（负类为0会直接让不等式失效）。所以必须用-1,+1来保持对称的几何间隔定义。

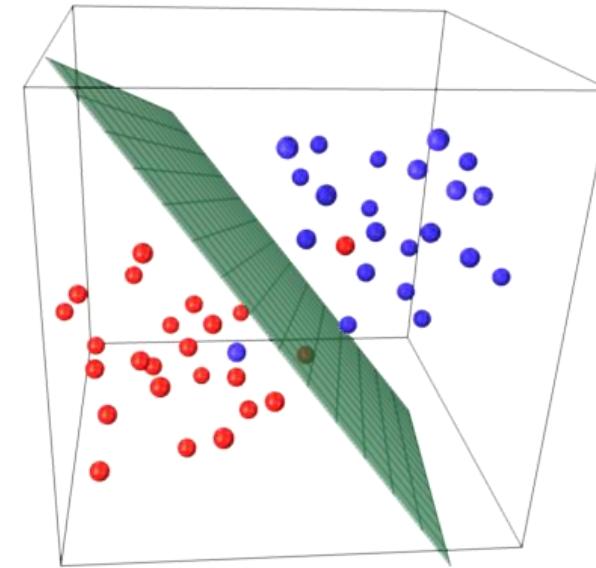
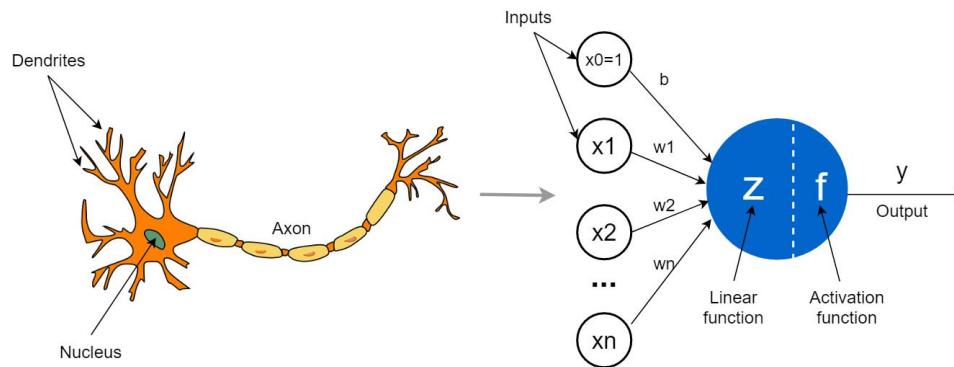
本质：SVM 用对称的“正负间隔”表达分类边界，而逻辑回归用的是概率的二元事件。

► SVM的前置知识

➤ 2) SVM的前身：感知机（Perceptron）模型——线性分类的起点

- 模型形式：感知机是最早的线性二分类器： $f(x) = \text{sign}(w^T x + b)$

$$\text{sign}(x) = \begin{cases} -1, & x < 0 \\ 0, & x = 0 \\ 1, & x > 0 \end{cases}$$



- 目标：找到一个能把正负样本分开的超平面。
- 特点：1) 只要数据线性可分，感知机一定会收敛；2) 但它可能找到“任意”一个可分超平面，而不是“最优”的；3) 对噪声极其敏感，数据稍微不可分就会发散。
- 判别条件： $y_i(w^T x_i + b) > 0$
- 意思是：样本 x_i 被正确分类。但是！这个条件**只要求“正负号正确”，并没有度量分类边界与样本点的“远近”**，所以可能找到很多不同的分割超平面。

► SVM的前置知识

➤ 2) SVM的前身：感知机（Perceptron）模型——线性分类的起点

- 感知机的目标是让所有样本满足：

$$y_i(w^T x_i + b) > 0$$

- 如果某个样本被错分，则：

$$y_i(w^T x_i + b) \leq 0$$

- 于是感知机的损失函数常定义为误分类样本的总“代价”：

$$L(w, b) = - \sum_{i \in M} y_i(w^T x_i + b)$$

- 其中M表示被误分类的样本集合。损失函数的直观含义：让错分点的“负值”尽量小，逼着模型修正方向。

- **基于所有样本的批量梯度下降法（BGD）是行不通的，原因在于损失函数里面只有误分类集合里面的样本才能参与损失函数的优化。由于每次仅需要使用一个误分类的点来更新梯度，于是采用随机梯度下降：**

- 对于某个误分类样本(x_i, y_i)，梯度为：

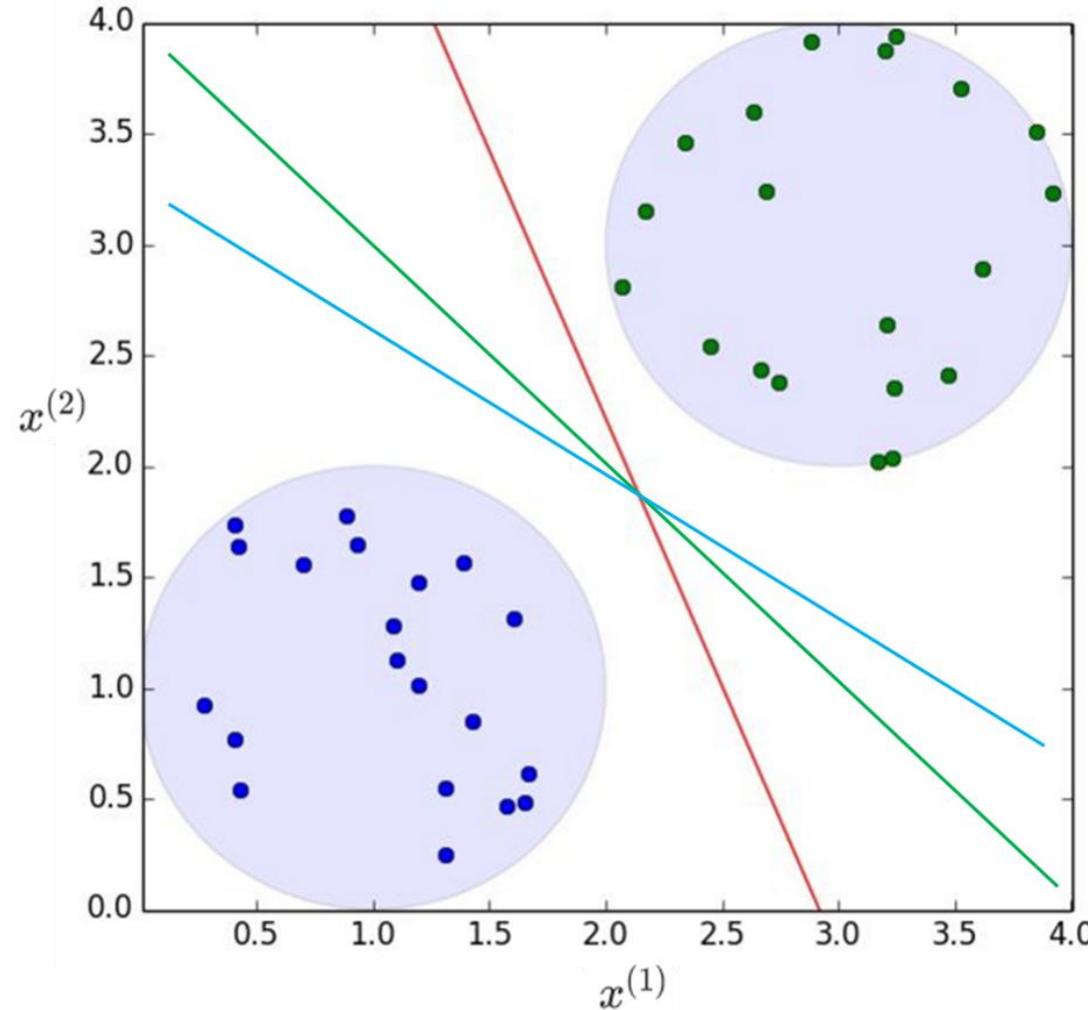
$$\nabla_w L = -y_i x_i, \quad \nabla_b L = -y_i$$

- 于是参数更新规则为：

$$\begin{aligned} w &\leftarrow w + \lambda y_i x_i \\ b &\leftarrow b + \lambda y_i \end{aligned}$$

► SVM的前置知识

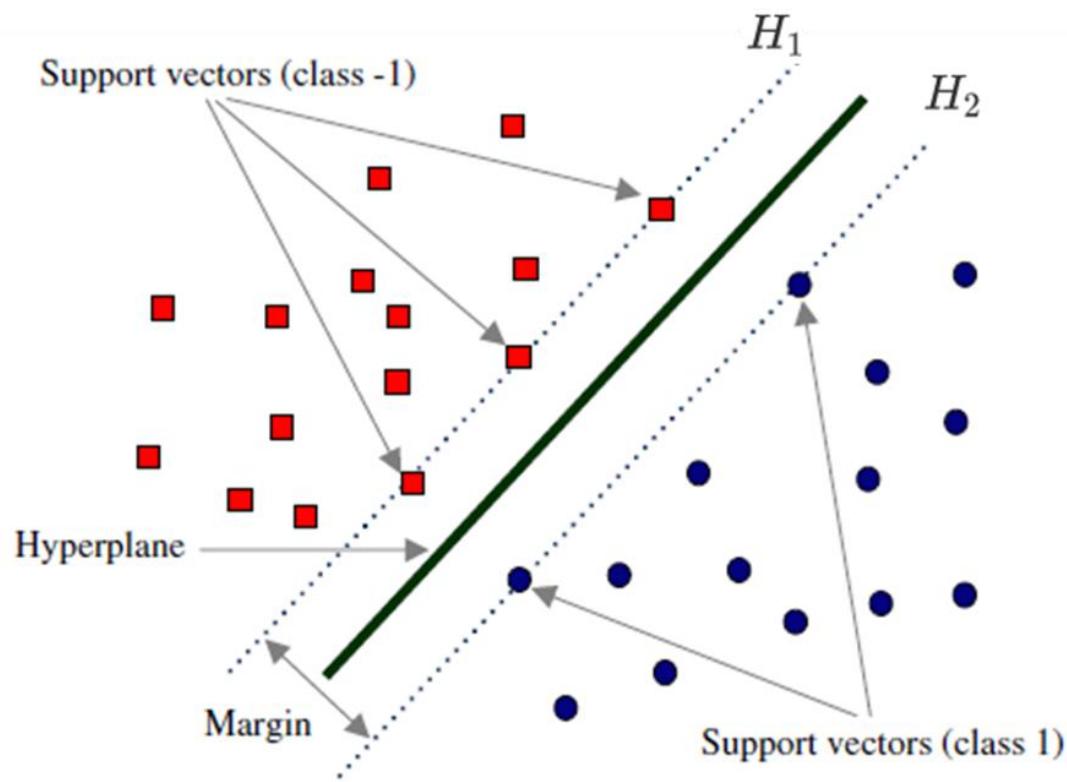
- 满足感知机分类的超平面并不止一个，不同的超平面依赖参数的初始值，感知机模型可以有多个解。问题：那个是最好的超平面呢？



- a.能够将两个类型的样本分开；
 - b.能够最大化决策边界附近的两类型之间的距离。
 - 虽然红色和蓝色的超平面能够分类，但是绿色的超平面显然更加合理。
- 感知机只要求把数据分开，而 SVM 在此基础上引入功能间隔、几何间隔的概念，并通过最大化间隔找到最优分割超平面，从“能分”进化为“分得最好”。

► SVM的前置知识

➤ 间隔是什么？间隔就是支持向量到超平面的间隔。那什么又是支持向量呢？间隔如何定义呢？



- 距离超平面最近的样本点，我们定义为支持向量。如左图所示，黑色实线为超平面(hyperplane)，在虚线 H_1 和虚线 H_2 上的点即为支持向量。
- 超平面两侧一定都有支持向量吗？对于初始化的超平面而言不一定。但是最终一定会求出一个合适的决策边界，既能保证 H_1 和 H_2 关于超平面对称，又能保证超平面两侧都有支持向量。
- H_1 和 H_2 关于超平面对称吗？ H_1 和 H_2 关于超平面对称， H_1 与 H_2 之间的距离就称为边界(margin)

► SVM的前置知识

➤ 3) 功能间隔

- 定义：给定训练样本 (x_i, y_i) 和分类超平面 $f(x) = w^T x + b$ ，样本的功能间隔定义为：

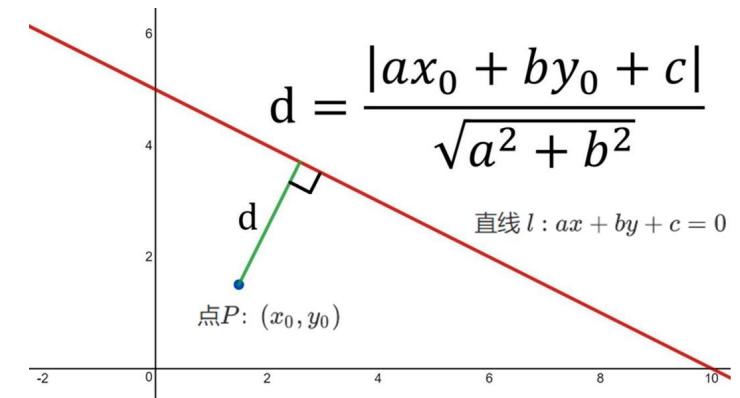
$$\hat{\gamma}_i = y_i(w^T x_i + b)$$

- 作用：衡量超平面对样本分类的“信心”大小。
- 注意：功能间隔会随着 w 的缩放而成比例缩放（即如果 w 变成 $2w$ ，功能间隔也会变成原来的2倍）。功能间隔本身不能直接衡量模型泛化能力，因为可以无限放大 w 来增大

➤ 4) 几何间隔

- 定义：是样本到超平面的实际距离，即将功能间隔除以权重的范数：

$$\gamma_i = \frac{\hat{\gamma}_i}{\|w\|} = \frac{y_i(w^T x_i + b)}{\|w\|}$$

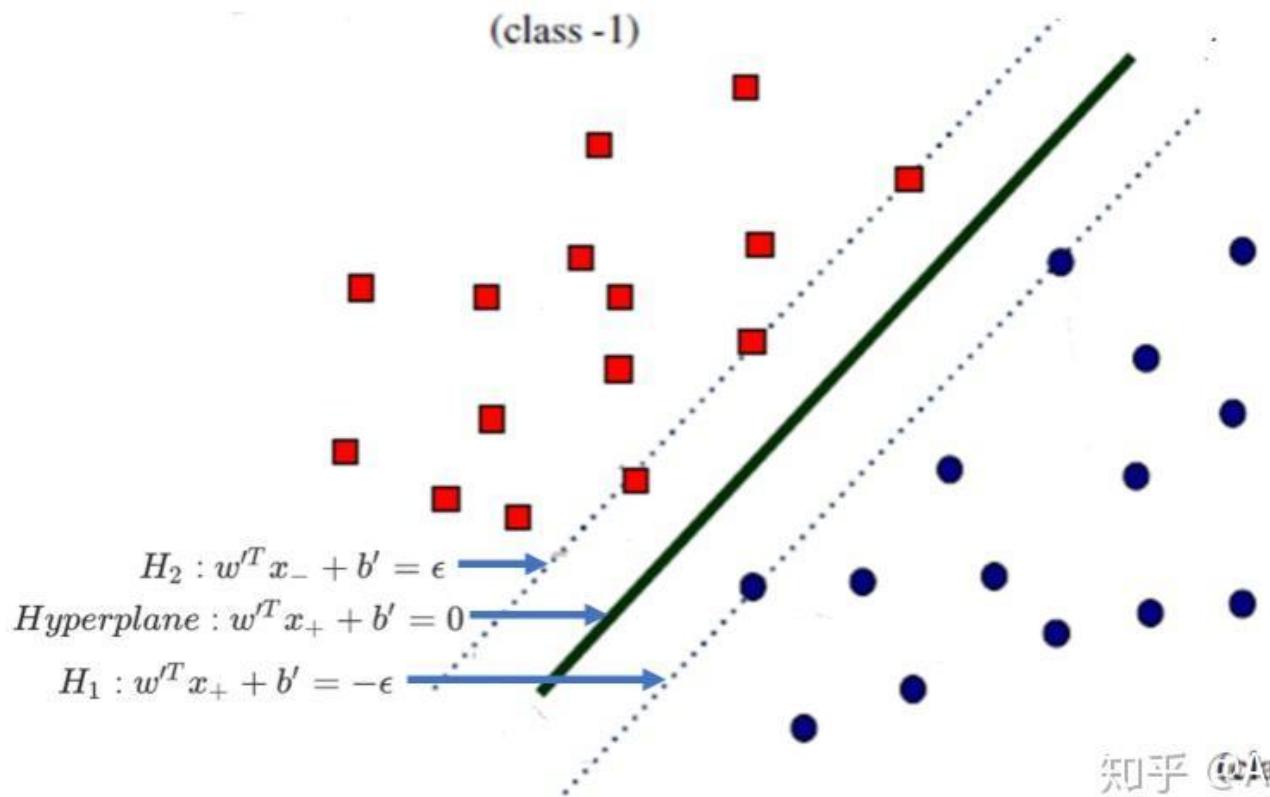


- 作用：衡量样本到分类边界的真实距离，是衡量分类器泛化能力的指标。
- 特点：对 w 的缩放不变，表示“真实间隔”。

功能间隔是代数上的分类置信度，几何间隔才是衡量分类边界稳健性的距离，SVM的目标是最大化最小几何间隔。

► SVM的数学表达形式

- SVM的核心目标即使在一定约束的条件下找到最大间隔（硬间隔，线性可分）。
 - SVM 的模型是让所有点到超平面的距离大于一定的距离，也就是所有的分类点要在各自类别的支持向量两边。我们只需要定义支持向量到超平面的距离最大化即可。



source:<https://zhuanlan.zhihu.com/p/36332083>

◆ 我们定义超平面、H1、H2的数学表达式为：

$$\text{Hyperplane} : w'^T x_+ + b' = 0$$

$$H_1 : w'^T x_+ + b' = -\epsilon$$

$$H_2 : w'^T x_- + b' = \epsilon$$

◆ 两边同时除以，并简化方程组得到：

$$\text{Hyperplane} : w^T x_+ + b = 0$$

$$H_1 : w^T x_+ + b = -1$$

$$H_2 : w^T x_- + b = 1$$

$$w = \frac{w'}{\epsilon}, \quad b = \frac{b'}{\epsilon}$$

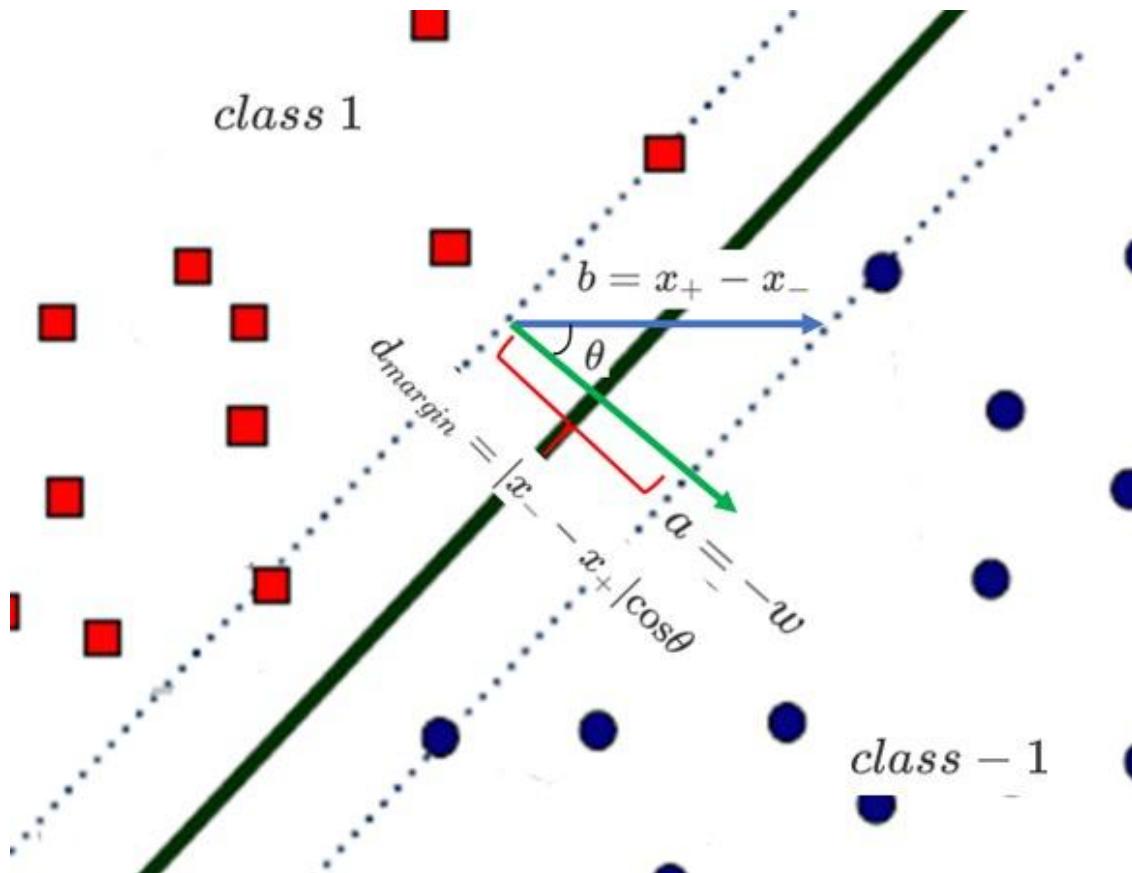
◆ $H_1 - H_2$ ，得到：

$$(-w)^T (x_+ - x_-) = 2$$

知乎 @Aug1st

► SVM的数学表达形式

- SVM的核心目标即使在一定约束的条件到找到最大间隔（硬间隔，线性可分）。
 - SVM 的模型是让所有点到超平面的距离大于一定的距离，也就是所有的分类点要在各自类别的支持向量两边。我们只需要定义支持向量到超平面的距离最大化即可。



source:<https://zhuanlan.zhihu.com/p/36332083>

- ◆ w 表示超平面的法向量， $x_+ - x_-$ 表示沿x轴方向的向量。从其几何意义可以推断出：

$$(-w)^T(x_+ - x_-) = 2 \rightarrow |w||x_+ - x_-| \cos \theta = 2$$

- ◆ 令： $d_{margin} = |x_+ - x_-| \cos \theta$

- ◆ 则： $|w|d_{margin} = 2 \rightarrow d_{margin} = \frac{2}{\|w\|_2}$

- ◆ 由此，优化函数定义为：

$$\max \frac{2}{\|w\|_2} \quad s.t. \quad y_i(w^T x_i + b) \geq 1 \quad (i = 1, 2, \dots, m)$$

- ◆ SVM的代价函数等价于：

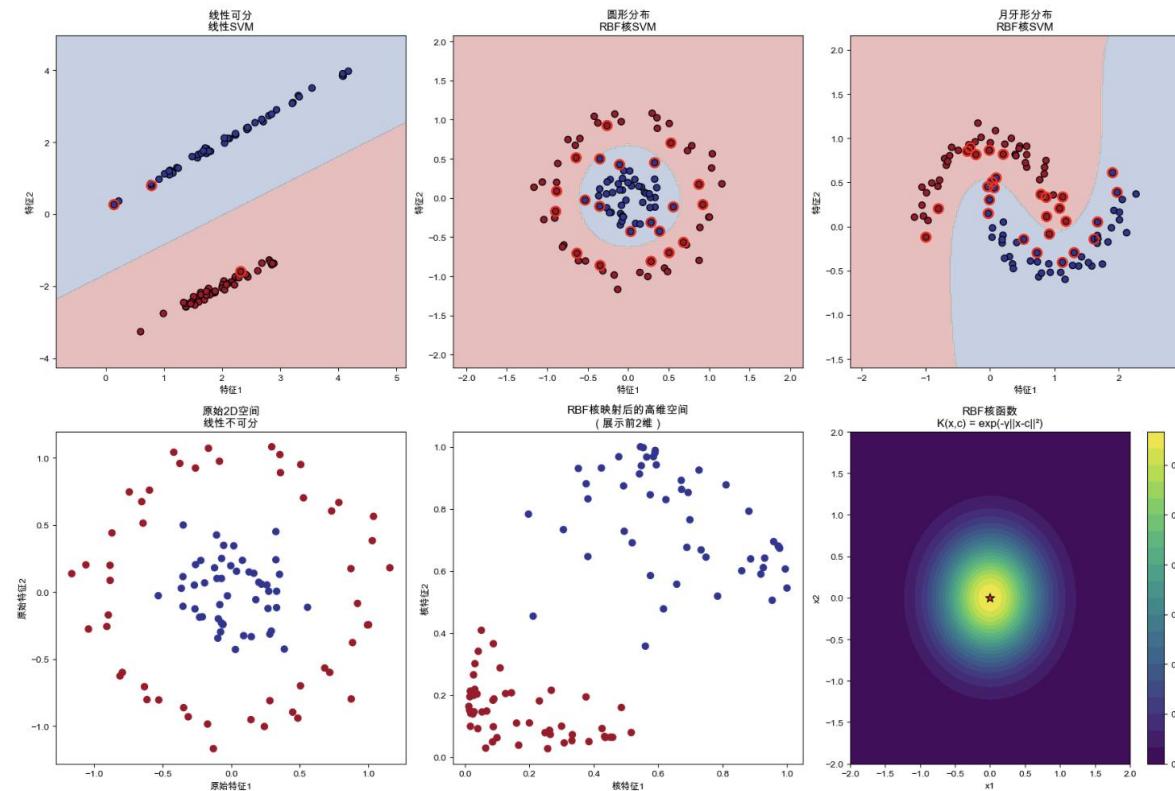
$$\min \frac{1}{2} \|w\|_2^2 \quad s.t. \quad y_i(w^T x_i + b) \geq 1 \quad (i = 1, 2, \dots, m)$$

- 总结：感知机的代价函数是分母为1的几何间隔，SVM 的优化函数是分子为2几何间隔。之所以分子为2是因为我们度量的是正负两条支持向量超平面之间的总宽度，而不是单侧距离。

► 模型假设（非常重要！）

- SVM 的模型同样隐含了一些重要假设，如果这些假设在数据中不成立，那么模型的拟合效果往往显著下降。

- 1) 数据近似线性可分（或通过核函数在高维空间可分）

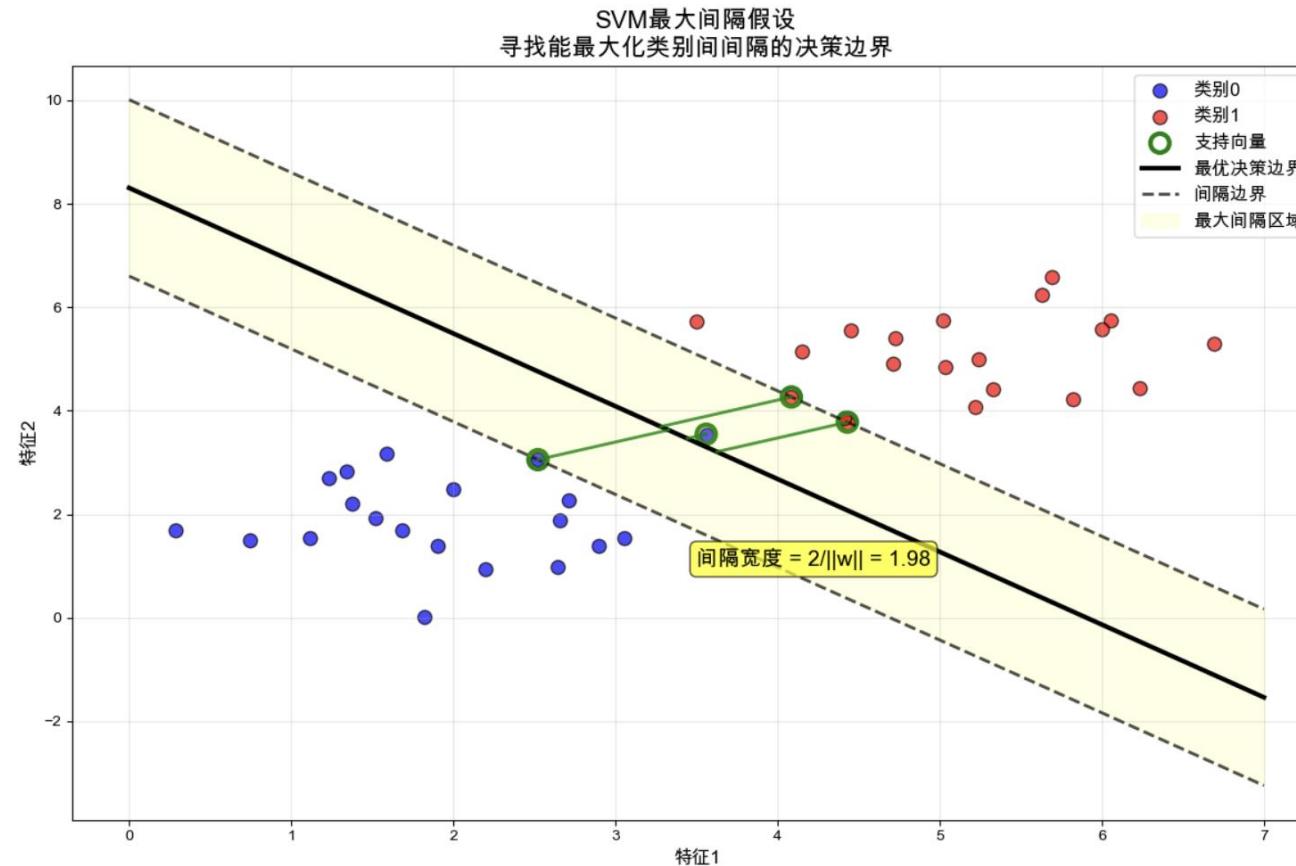


- 基本假设是类别之间存在一个“间隔最大化”的超平面，能较好地区分数据，如果在原空间不可分，可以通过核技巧映射到高维特征空间。**

► 模型假设（非常重要！）

- SVM 的模型同样隐含了一些重要假设，如果这些假设在数据中不成立，那么模型的拟合效果往往显著下降。

- 2) 分类的鲁棒性来自最大间隔假设



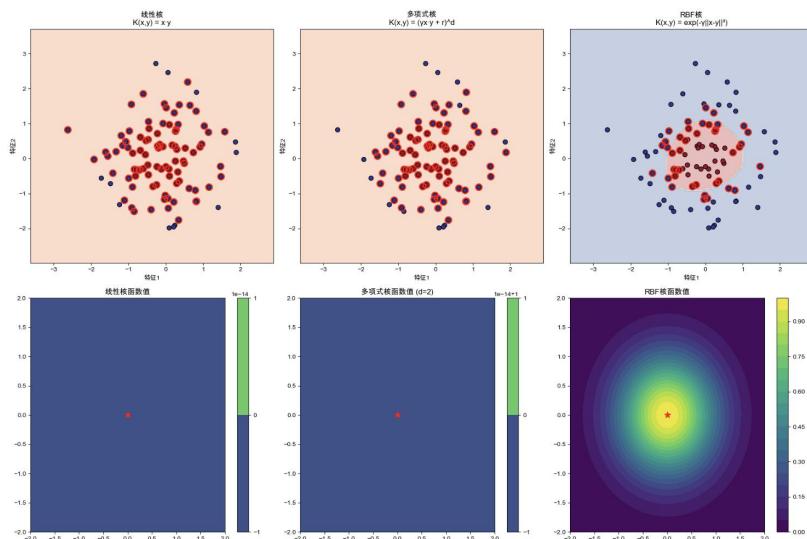
- 假设最优的分类超平面应该让支持向量到超平面的间隔最大，从而提升泛化能力。

► 模型假设（非常重要！）

- SVM 的模型同样隐含了一些重要假设，如果这些假设在数据中不成立，那么模型的拟合效果往往显著下降。

- 3) 样本之间的关系可用内积/核函数度量

- SVM的优化问题决策函数最终可以写成 $f(x) = \text{sign}(\sum_i \alpha_i y_i \langle x, x_i \rangle + b)$
 - 所以，最后的判决结果只依赖于样本点之间的内积 $\langle x, x_i \rangle$ ：



- 在线性 SVM 里， $\langle x, x_i \rangle$ 表示点和点之间的“相似度”。
 - 在核 SVM 里，直接用核函数 $K(x, x_i)$ 代替内积，比如：RBF核假设相似度由欧式距离决定；多项式核假设特征组合能捕捉非线性关系。

- 所以 SVM 的隐含假设就是：样本的可分性与相似性可以通过某种核函数（或内积）来正确衡量，如果核函数选错了，SVM 在那个空间里可能就不可分了。

► 小结

➤ 为什么需要学习SVM?

- 学习 SVM 能帮助我们理解最大间隔原理与核方法，掌握一种在小样本、高维度下仍具强大泛化能力的经典机器学习模型。

➤ 什么是KNN（K-近邻算法）？

- SVM（支持向量机）是一种通过寻找最大化分类间隔的超平面来实现二分类的机器学习模型。它可通过核函数将数据映射到高维空间，从而解决线性不可分问题，兼具良好的泛化能力。

- SVM的代价函数：
$$\min \frac{1}{2} \|w\|_2^2 \quad s.t. \quad y_i(w^T x_i + b) \geq 1 \quad (i = 1, 2, \dots, m)$$

➤ 模型假设：

- 数据近似线性可分（或通过核函数在高维空间可分）。
- 分类的鲁棒性来自最大间隔假设。
- 样本之间的关系可用内积/核函数度量。

学习 SVM 可以掌握一种基于最大间隔的判别式算法，它通过最优超平面与核函数实现分类预测，假设数据在合适空间中近似可分。

目录章节

CONTENTS

01

KNN的概念和原理

02

模型求解与评估

03

模型扩展与改进

04

模型实现与实战

05

总结

► 模型求解

➤ 从上一节可知，SVM的目标是在约束条件下找到最大间隔的超平面。

- 给定 (x_i, y_i) , $x_i \in \mathbb{R}^d$, $y_i \in \{-1, +1\}$ 。分类超平面 $w^T x + b = 0$ 。几何间隔最小值：

$$\gamma = \min_i \frac{y_i(w^T x_i + b)}{\|w\|}$$

- 由于几何间隔缩放保持不变，所以通过对 w , b 做缩放归一化，**把离超平面最近的支持向量的功能间隔固定为1**，最大化几何间隔问题就转换为：

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad s.t. \quad y_i(w^T x_i + b) \geq 1$$

➤ 求解步骤：

- 根据凸优化理论，代价函数满足KKT条件，我们可以通过拉格朗日函数将我们的优化目标转化为无约束的优化函数：

$$L(w, b, \alpha) = \frac{1}{2} \|w\|_2^2 - \sum_{i=1}^m \alpha_i [y_i(w^T x_i + b) - 1] \quad s.t. \quad \alpha_i \geq 0$$

- 由于引入了拉格朗日函数，我们的优化目标变成：

$$\min_{w,b} \max_{\alpha_i \geq 0} L(w, b, \alpha)$$

- 注：支持向量功能间隔为什么固定为1？由于可以缩放 w, b 使得最小功能间隔等于任意正数，选1这是为了方便数学处理（方便归一化），几何间隔变成 $\gamma = 1/\|w\|$ ，最大化几何间隔 $\max \gamma$ 就等价于最小化 $\|w\|^2/2$ ，最终优化问题变为标准的凸二次规划问题。

► 模型求解

- 转换为对偶问题：

$$\max_{\alpha_i \geq 0} \min_{w,b} L(w, b, \alpha)$$

- 首先先求w和b的最小值，分别对w和b求偏导，得到：

$$\frac{\partial L}{\partial w} \rightarrow w = \sum_{i=1}^m \alpha_i y_i x_i, \quad \frac{\partial L}{\partial b} \rightarrow \sum_{i=1}^m \alpha_i y_i = 0$$

- 令

$$\begin{aligned}\psi(\alpha) &= \min_{w,b} L(w, b, \alpha) \\ &= \frac{1}{2} \|w\|_2^2 - \sum_{i=1}^m \alpha_i [y_i(w^T x_i + b) - 1]\end{aligned}$$

- 代入w偏导的等式：

$$\psi(\alpha) = \frac{1}{2} \left\| \sum_{i=1}^m \alpha_i y_i x_i \right\|^2 - \sum_{i=1}^m \alpha_i \left(y_i \left(\left(\sum_{j=1}^m \alpha_j y_j x_j \right)^T x_i + b \right) - 1 \right)$$

- 展开第二项：

$$\sum_{i=1}^m \alpha_i y_i \left(\left(\sum_{j=1}^m \alpha_j y_j x_j \right)^T x_i \right) = \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_j^T x_i$$

- 进一步代入b偏导的等式： $\sum_i \alpha_i y_i b = b \sum_i \alpha_i y_i = 0$ ， 所以对偶函数简化为：

$$\psi(\alpha) = \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j x_i^T x_j$$

- 对偶问题的最终形式：

$$\boxed{\max_{\alpha} \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i,j=1}^m \alpha_i \alpha_j y_i y_j x_i^T x_j \quad s.t. \quad \sum_{i=1}^m \alpha_i y_i = 0, \alpha_i \geq 0, i = 1, \dots, m}$$

- 这其实就是 **凸二次规划 (QP)** 问题：目标函数是二次函数（凸），约束是线性等式+不等式。

► 扩展：模型求解的四种方法

➤ A. 内点法 (Interior Point Method) :

- 思想：属于凸优化的通用算法，通过把约束条件引入到目标函数里，加上一个障碍函数 (barrier function)，保证迭代过程中始终在可行域的“内部”。在迭代过程中，逐渐让障碍函数的影响减小，最终收敛到边界处的最优解。
- 简化步骤：
 - 把不等式约束 $\alpha_i \geq 0$ 转换为障碍项 $-\mu \sum \log(\alpha_i)$ 。
 - 构造一个新的目标函数（原始目标 + 障碍项）。
 - 用牛顿法在可行域内部迭代更新 α 。
 - 随着迭代逐渐减小 $\mu \rightarrow 0$ ，收敛到最优解。
- 特点：
 - 优点：收敛速度快（多项式时间复杂度），数值稳定，适合中等规模问题。
 - 缺点：需要解大型线性方程组，复杂度一般是 $O(m^3)$ ，样本量一大就不行。

► 扩展：模型求解的四种方法

➤ B. 活动集法 (Active Set Method)

- 思想：在最优解上，只有一部分约束是“紧的”（即等号成立，例如 $\alpha_i=0$ 或 $\alpha_i=C$ ），其他不等式约束是“松的”。活动集法就是 动态维护一组“活跃约束”，把它们当作等式处理，然后在这个子空间上解更小的 QP 问题。
- 简化步骤：
 - 从一个初始可行解开始（满足约束）。
 - 找出当前解下最可能“起作用”的约束（比如某个 α_i 已经贴近0或C）。
 - 把这些约束加入活动集，解约化后的 QP 子问题。
 - 迭代更新活动集，直到收敛到 KKT 条件满足的点。
- 特点：
 - 优点：比内点法更高效，尤其在解的稀疏性强时（很多 $\alpha_i=0$ ）。
 - 缺点：更新活动集需要反复调整，可能收敛慢，不适合超大规模数据。

► 扩展：模型求解的四种方法

- C.坐标下降法 (Coordinate Descent)
 - 思想：每次只更新一个（或一对）变量，固定其余变量不变。在一维上更新有封闭解，问题大大简化。
 - 简化步骤：
 - 从初始解开始。
 - 选择某个坐标 (α_i)，在该维度上解优化问题。
 - 更新 α_i 保持其他变量不动。
 - 按顺序或按启发式规则选择下一个坐标，迭代直到收敛。
 - 特点：
 - 优点：内存占用少，计算快，适合超大规模问题（比如上百万样本）。
 - 缺点：收敛速度可能慢，尤其在维度之间相关性强时。
- **SMO算法**就是坐标下降法的特例：每次只优化一对变量 (α_i, α_j)，因为SVM的等式约束保持成立。

► 扩展：模型求解的四种方法

- D. 梯度下降法 (Gradient Descent)
 - 思想：是基于 原始形式的铰链损失 + 正则化项 构建目标函数，然后利用（随机）梯度下降迭代优化。
 - 简化步骤：
 - 构建目标函数。
 - 计算梯度。
 - 更新参数。
 - 迭代直到收敛。
 - 特点：
 - 优点：梯度下降实现简单、可扩展到大规模数据（尤其是用随机梯度下降时），避免对偶问题的复杂计算。
 - 缺点：收敛速度依赖学习率选择，可能震荡或收敛慢，并且对非线性核 SVM 不如对偶方法高效。

► 扩展：SMO算法

- 根据决策函数，发现我们只需要求出所有 α 和偏置值 b 即可，利用**SMO算法**求解最终对偶形式问题：
 - SMO 是什么？全称：Sequential Minimal Optimization（序列最小优化），由Platt在1998年在《Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines》提出。背景：传统二次规划（QP）方法求解 SVM 对偶问题，复杂度高，内存消耗大，SMO将问题拆分为最小可解问题（两个变量），极大提升训练效率。
 - 核心思想：每次只选择两个拉格朗日乘子(α_i, α_j)进行优化，利用解析解直接更新，避免使用复杂的数值优化方法（如梯度下降、二次规划求解器）
 - 特点：每次只优化两个变量，满足线性约束；有解析解，不需要迭代求解整个高维二次规划问题；高效，适合大规模 SVM。
 - SMO算法步骤：初始化 $a_i=0, b=0$ -> 选择违反KKT条件以及最大化步长的两个优化变量(α_i, α_j) -> 通过线性约束条件计算约束的上下界限 -> 更新 α_j 的解析解 -> 更新 α_i -> 更新偏置 b -> 检查所有 α 是否满足KKT条件，如果满足条件停止否则继续选择违反KKT条件的样本进行更新。

► 模型求解：KKT条件

➤ 对于一个凸优化+不等式约束的问题，最优解必须满足KKT条件，即：

- 1) 原始可行性 (Primal feasibility) :

$$y_i(w^T x_i + b) - 1 \geq 0, \quad i = 1, \dots, m$$

- 2) 对偶可行性 (Dual feasibility) :

$$\alpha_i \geq 0, \quad i = 1, \dots, m$$

- 3) 站立性条件 (Stationarity) :

$$\frac{\partial L}{\partial w} \rightarrow w = \sum_{i=1}^m \alpha_i y_i x_i, \quad \frac{\partial L}{\partial b} \rightarrow \sum_{i=1}^m \alpha_i y_i = 0$$

- 4) 互补松弛条件 (Complementary slackness) :

$$\alpha_i(y_i(w^T x_i + b) - 1) = 0, \quad i = 1, \dots, m$$

- 根据互补松弛条件：可以推断两种情况（在软间隔条件下可以分为三种）

$$\alpha_i = 0 \rightarrow y_i(w^T x_i + b) \geq 1$$

$$\alpha_i \geq 0 \rightarrow y_i(w^T x_i + b) = 1$$

- 直观理解：a.KKT条件最终可以表示**只有支持向量 ($\alpha_i > 0$) 决定分类超平面**；b.非支持向量 ($\alpha_i = 0$) 没有贡献；c.可以写成支持向量的线性组合。

► 模型求解：KKT条件

- 根据对偶问题推导+KKT条件能得到什么结论呢？
- 1) 根据互补松弛条件：可以推断两种种情况（在软间隔条件下可以分为三种）

$$\begin{aligned}\alpha_i = 0 \rightarrow y_i(w^T x_i + b) &\geq 1 \\ \alpha_i \geq 0 \rightarrow y_i(w^T x_i + b) &= 1\end{aligned}$$

- 直观理解：a.KKT条件最终可以表示只有支持向量 ($\alpha_i > 0$) 决定分类超平面；b.非支持向量 ($\alpha_i = 0$) 没有贡献；c.可以写成支持向量的线性组合。
- 2) 根据站立性条件：可以推断出决策函数的形式
- 分类的超平面为：

$$f(x) = w^T x + b$$

- 代入站立性条件：

$$w = \sum_{i=1}^m \alpha_i y_i x_i$$

- 得到：

$$f(x) = (\sum_{i=1}^m \alpha_i y_i x_i)^T x + b = \sum_{i=1}^m \alpha_i y_i (x_i^T x) + b$$

- 如果使用和函数 $K(x_i, x) = x_i^T x$ ， 推广成：

$$f(x) = \sum_{i=1}^m \alpha_i y_i K(x_i, x) + b$$

► 模型求解：SMO算法

➤ 第一步：初始化 α_i 和 b

- 将所有 $\alpha_i=0$ ， 初始化偏置 $b=0$ 。
- 还可以设置容差 ϵ 判断KKT条件是否满足。

➤ 第二步：选择两个优化变量 (α_i, α_j)

- 首先选择第一个变量 α_i ，也称为外循环，这个变量需要选择对目标函数影响最大且距离真实值最离谱的变量，也就是违反 KKT 约束条件最严重的样本点，即违反

$$\alpha_i = 0 \rightarrow y_i(w^T x_i + b) \geq 1$$

$$\alpha_i \geq 0 \rightarrow y_i(w^T x_i + b) = 1$$

- 对于样本种的任意一个点，都有可能违反上诉条件中的任意一个条件，那首先选择哪一个呢？由于支持向量可以决定超平面，所以我们应该优先选择 $\alpha_i > 0$ 的点，即第二个。
- 然后选择第二个变量 α_i ，也称为内循环。 α_i 的选择在于满足 α_j 的变化最大，即 $|E_i - E_j|$ 最大值对应的 α_j 。其中 E_i 表示 α_i 的误差， E_j 表示 α_j 的误差。

► 模型求解：SMO算法

➤ 第三步：我们已经选择了两个拉格朗日乘子 α_i, α_j ，下一步就是最核心的一步，根据约束条件求得 α_i, α_j 的取值范围：

- 这是每次只优化两个拉格朗日乘子 α_i, α_j 【因为两个变量的子问题是二次函数，带边界约束，可以直接用解析公式求解】，这样可以保持线性约束不变。然后迭代所有样本，直到所有 α 满足 KKT 条件 \rightarrow SVM 收敛。
- 那么在更新这两个时，唯一需要保持的约束就是：

$$y_i\alpha_i + y_j\alpha_j + \sum_{k \neq i,j} y_k\alpha_k = 0$$

- 由于其余项是常数，等式约束可以简化为：

$$y_i\alpha_i^{\text{new}} + y_j\alpha_j^{\text{new}} = \text{const} = \zeta \rightarrow \alpha_i^{\text{new}} = \frac{\zeta - y_j\alpha_j^{\text{new}}}{y_i}$$

- 因为 $y_i \in \{-1, +1\}$ ，所以这就是一条直线；可行域还必须满足 $\alpha_i \geq 0, \alpha_j \geq 0$ 。
- a. 当 $y_i \neq y_j$ ，即 $y_i = -y_j$ （注意 $|y|=1$ ），由于

$$\zeta = y_i\alpha_i^{\text{old}} + y_j\alpha_j^{\text{old}} = y_i(\alpha_i^{\text{old}} - \alpha_j^{\text{old}})$$

- 带回可得：

$$\alpha_i^{\text{new}} = \frac{y_i(\alpha_i^{\text{old}} - \alpha_j^{\text{old}}) - (-y_i)\alpha_j^{\text{new}}}{y_i} = \alpha_i^{\text{old}} - \alpha_j^{\text{old}} + \alpha_j^{\text{new}} = \alpha_i^{\text{old}} + (\alpha_j^{\text{new}} - \alpha_j^{\text{old}})$$

► 模型求解：SMO算法

- 根据 $\alpha_i \geq 0, \alpha_j \geq 0$:

$$\alpha_i^{\text{new}} \geq 0 \rightarrow \alpha_j^{\text{new}} \geq \alpha_j^{\text{old}} - \alpha_i^{\text{old}}$$

$$\alpha_j^{\text{new}} \geq 0 \rightarrow \alpha_j^{\text{new}} \geq 0$$

- 合并得到下界:

$$L = \max(0, \alpha_j^{\text{old}} - \alpha_j^{\text{old}})$$

- 由于硬间隔SVM没有上界，因此上界:

$$H = +\infty$$

- 直观：可行直线在第一象限内从某个点起可以“无限向右上”走；唯一限制只是别把 α_i 变成负的，所以给 α_j 一个下界即可。
- b. 当 $y_i=y_j$ （注意 $|y|=1$ ），这时：

$$y_i(\alpha_i^{\text{new}} + \alpha_j^{\text{new}}) = \zeta \rightarrow \alpha_i^{\text{new}} + \alpha_j^{\text{new}} = \zeta' = \alpha_i^{\text{old}} + \alpha_j^{\text{old}}$$

- 即：

$$\alpha_i^{\text{new}} = \alpha_i^{\text{old}} + \alpha_j^{\text{old}} - \alpha_j^{\text{new}}$$

► 模型求解：SMO算法

- 根据 $\alpha_i \geq 0, \alpha_j \geq 0$:

$$\begin{aligned}\alpha_i^{\text{new}} \geq 0 &\rightarrow \alpha_j^{\text{new}} \leq \alpha_i^{\text{old}} + \alpha_j^{\text{old}} \\ \alpha_j^{\text{new}} \geq 0 &\rightarrow \alpha_j^{\text{new}} \geq 0\end{aligned}$$

- 因此上下界为

$$L = 0$$

$$H = \alpha_i^{\text{old}} + \alpha_j^{\text{old}}$$

- 直观：可行直线是“和为常数”的线段；落在第一象限里的那段从 0 到 $\alpha_i + \alpha_j$ 。
- 总结：

$$\begin{cases} y_i \neq y_j : & L = \max(0, \alpha_j^{\text{old}} - \alpha_i^{\text{old}}), \quad H = +\infty, \\ y_i = y_j : & L = 0, \quad H = \alpha_i^{\text{old}} + \alpha_j^{\text{old}}. \end{cases}$$

- 直观理解：当我们打算更新 α_i 时， α_j 也必须跟着进行调整，保证线性约束条件依然成立，这时就会出现一个区间， α_j 的新值只能落在 $[L, H]$ 之间，公式其实就是把约束+边界两个条件综合起来推断的。

► 模型求解：SMO算法

- 第四步：现在我们已经有了 α 的上下界限，下一步根据牛顿迭代法快速求得 α
 - 之前根据目前关于 α 的函数（对偶问题的最终形式，将 i 用 p, q 代替便于后续的推导）：

$$W(\alpha) = \sum_{k=1}^m \alpha_k - \frac{1}{2} \sum_{p=1}^m \sum_{q=1}^m \alpha_p \alpha_q y_p y_q K_{pq}.$$

◆ SMO算法的想法是根据牛顿迭代法进行快速迭代更新（二次型），即

$$\alpha_j^{\text{new}} = \alpha_j^{\text{old}} - \frac{\frac{dW}{d\alpha_j}}{\frac{d^2W}{d\alpha_j^2}}$$

- 结合约束条件+边界条件（之前推导的 α 的上下界限），这就意味着这意味着 α_i 可以用 α_j 表示，因此 W 最终只依赖于 α_j 。所以目前的任务是求得 $W(\alpha)$ 的一阶导数和二阶导数：
- ◆ 先求二阶导数：我们首先把 W 中与 α_i, α_j 有关的二次项挑出来：

$$W(\alpha) = \sum_{k=1}^m \alpha_k + \boxed{\frac{1}{2} \sum_{p=1}^m \sum_{q=1}^m \alpha_p \alpha_q y_p y_q K_{pq}}.$$

- 再次观察对偶函数，这里重点在第二项，它是一个双重求和的二次项，包含所有 α 两两之间的乘积。
- 所以我们需要把所有和 α_i, α_j 表示有关的项挑出来，其他项作为常数放到一边。

► 模型求解：SMO算法

■ 展开求和的时候，涉及 α_i, α_j 的二次项有：

◆ 1) α_i^2 的项目，当 $p=i, q=i$ 的时候：

$$-\frac{1}{2}\alpha_i^2 y_i^2 K(x_i, x_i) = -\frac{1}{2}K_{ii}\alpha_i^2$$

◆ 2) α_j^2 的项目，当 $p=j, q=j$ 的时候：

$$-\frac{1}{2}\alpha_j^2 y_j^2 K(x_j, x_j) = -\frac{1}{2}K_{jj}\alpha_j^2$$

◆ 3) $\alpha_i\alpha_j$ 的交叉项目，这里分为两种情况：

● a.p=i, q=j的时候：

$$-\frac{1}{2}\alpha_i\alpha_j y_i y_j K(x_i, x_j)$$

● a.p=j, q=i的时候：

$$-\frac{1}{2}\alpha_j\alpha_i y_j y_i K(x_j, x_i)$$

● 由于 $K(x_i, x_j)=K(x_j, x_i)$ ，所以两个是一样的，加起来得到：

$$-\alpha_i\alpha_j y_i y_j K(x_i, x_j)$$

■ 最后合并起来，所以 α_i, α_j 相关的二次项就是：

$$Q(\alpha_i, \alpha_j) = -\frac{1}{2}[K_{ii}\alpha_i^2 + K_{jj}\alpha_j^2 + 2y_i y_j K_{ij}\alpha_i\alpha_j], \quad K_{ii} = K(x_i, x_i), K_{jj} = K(x_j, x_j), K_{ij} = K(x_i, x_j)$$

► 模型求解

■ 下一步就是用约束关系 $\alpha_i^{\text{new}} = \frac{\zeta - y_j \alpha_j^{\text{new}}}{y_i} = \zeta' - y_i y_j \alpha_j^{\text{new}}$ 替换其中的一个变量，将其变为

$$Q(\alpha_i, \alpha_j) = -\frac{1}{2}[K_{ii}\alpha_i^2 + K_{jj}\alpha_j^2 + 2y_i y_j K_{ij}\alpha_i\alpha_j] \rightarrow Q(\alpha_j) = -\frac{1}{2}[K_{ii}(\zeta' - y_i y_j \alpha_j)^2 + K_{jj}(\alpha_j)^2 + 2y_i y_j K_{ij}\alpha_j(\zeta' - y_i y_j \alpha_j)]$$

- 展开二次项，关注 α_j^2 的系数

$$Q(\alpha_j) = \dots - \frac{1}{2}[(K_{ii} + K_{jj} - 2K_{ij})\alpha_j^2 + (\text{first-order term}) + \text{const}]$$

- 最终，二阶导数为：

$$\boxed{\frac{d^2W}{d\alpha_j^2} = -(K_{ii} + K_{jj} - 2K_{ij})}$$

- ◆ 为什么这么写？

- 因为核函数是个内积形式，代表两个变量在特征空间的相似程度：

$$K(x_p, x_q) = \langle \phi(x_p), \phi(x_q) \rangle$$

- 于是：

$$K_{ii} + K_{jj} - 2K_{ij} = \|\phi(x_i) - \phi(x_j)\|^2 \geq 0$$

- 这就是几何上的来源：实际上就是两个样本在特征空间里的“平方距离”
【核空间中两个点的距离平方】。

► 模型求解：SMO算法

- ◆ 二阶导数目前已经求出，还差一阶导数：

$$W(\alpha) = \sum_{k=1}^m \alpha_k - \boxed{\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j K(x_i, x_j)}$$

- 直接进行求一阶导数，第一项直接：

$$\frac{\partial}{\partial \alpha_k} \left(\sum_{t=1}^m \alpha_t \right) = 1$$

- 计求和项目为S，乘以系数1/2为第二项T：

$$S := \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j K_{ij} \rightarrow T := \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j K_{ij}$$

- 首先对乘积进行求导，使用克罗内克：

$$\frac{\partial(\alpha_i \alpha_j)}{\partial \alpha_k} = \frac{\partial \alpha_i}{\partial \alpha_k} \alpha_j + \alpha_i \frac{\partial \alpha_j}{\partial \alpha_k} = \delta_{ik} \alpha_j + \alpha_i \delta_{jk}$$

$$\delta_{ik} = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{if } i \neq k. \end{cases}$$

- 因此：

$$\frac{\partial S}{\partial \alpha_k} = \sum_{i=1}^m \sum_{j=1}^m (\delta_{ik} \alpha_j + \alpha_i \delta_{jk}) y_i y_j K_{ij}$$

- 把两项拆开：

$$\frac{\partial S}{\partial \alpha_k} = \underbrace{\sum_{j=1}^m \alpha_j y_k y_j K_{kj}}_{(A)} + \underbrace{\sum_{i=1}^m \alpha_i y_i y_k K_{ik}}_{(B)}$$

► 模型求解：SMO算法

- 利用核函数的对称性合并两项，把A和B的求和索引重命名为l，可以看到两个是相同的

$$(A) = \sum_{l=1}^m \alpha_l y_k y_l K_{kl}, \quad (B) = \sum_{l=1}^m \alpha_l y_k y_l K_{lk} = \sum_{l=1}^m \alpha_l y_k y_l K_{kl}$$

- 因此

$$\frac{\partial S}{\partial \alpha_k} = 2 \sum_{l=1}^m \alpha_l y_k y_l K_{kl}$$

- 于是

$$\frac{\partial T}{\partial \alpha_k} = \frac{1}{2} \frac{\partial S}{\partial \alpha_k} = \sum_{l=1}^m \alpha_l y_k y_l K_{kl}$$

- 合并第一项和第二项的导数，最终得到：

$$\frac{\partial W}{\partial \alpha_k} = 1 - \sum_{l=1}^m \alpha_l y_k y_l K_{kl}$$

- 通常把 y_k 提出来写成更醒目的形式：

$$\boxed{\frac{\partial W}{\partial \alpha_k} = 1 - y_k \sum_{l=1}^m \alpha_l y_l K_{kl}}$$

► 模型求解：SMO算法

- 用j替换k后注意到后面求和的部分和决策函数的求和部分一致，即：

$$\frac{\partial W}{\partial \alpha_j} = 1 - y_k \sum_{l=1}^m \alpha_l y_l K_{jl}$$
$$f(x_j) = \sum_{l=1}^m \alpha_l y_l K_{lj} + b \rightarrow \sum_{l=1}^m \alpha_l y_l K_{lj} = f(x_j) - b$$

- 将上述公式合并后，可得：

$$\frac{\partial W}{\partial \alpha_j} = 1 - y_j(f(x_j) - b)$$

- 将误差定义为预测值-真实值：

$$E_j = f(x_j) - y_j$$

- 代入偏导中，得到：

$$\frac{\partial W}{\partial \alpha_j} = 1 - y_j((E_j + y_j) - b) = 1 - y_j E_j - y_j^2 + y_j b = -y_j E_j + y_j b$$

- 同理，对于ak有：

$$\frac{\partial W}{\partial \alpha_i} = -y_i E_i + y_i b$$

- 由于约束条件：

$$y_i \alpha_i + y_j \alpha_j = \text{const} \rightarrow \Delta \alpha_i = -\frac{y_j}{y_i} \Delta \alpha_j$$

► 模型求解：SMO 算法

- 当 α_j 变化时， α_i 必须跟着联动，所以沿着约束方向的总导数：

$$\frac{dW}{d\alpha_j} = \frac{\partial W}{\partial \alpha_j} \frac{d\alpha_j}{d\alpha_j} + \frac{\partial W}{\partial \alpha_i} \frac{d\alpha_i}{d\alpha_j} = \frac{\partial W}{\partial \alpha_j} + \frac{\partial W}{\partial \alpha_i} \frac{d\alpha_i}{d\alpha_j} = \frac{\partial W}{\partial \alpha_j} + \frac{\partial W}{\partial \alpha_i} \left(-\frac{y_j}{y_i}\right)$$

- 根据之前推导的：

$$\frac{\partial W}{\partial \alpha_i} = -y_i E_i + y_i b$$

$$\frac{\partial W}{\partial \alpha_j} = -y_j E_j + y_j b$$

- 代入上式，得到：

$$\begin{aligned}\frac{dW}{d\alpha_j} &= (-y_j E_j + y_j b) + (-y_i E_i + y_i b) \left(-\frac{y_j}{y_i}\right) \\ &= y_j (E_i - E_j)\end{aligned}$$

- 将之前求得的二阶导数和上述的一阶导数代入之前牛顿迭代公式中，得到：

$$\alpha_j^{\text{new}} = \alpha_j^{\text{old}} - \frac{\frac{dW}{d\alpha_j}}{\frac{d^2W}{d\alpha_j^2}} = \alpha_j^{\text{old}} + \frac{y_j (E_i - E_j)}{K_{ii} + K_{jj} - 2K_{ij}}$$

- 需要注意的是，更新后的 α_j 必须落在之前我们推导的合法区间 $[L, H]$ 以内。

这条公式就是 SMO 的核心更新公式，把牛顿迭代和 SVM 的约束、误差信息完美结合在一起。

► 模型求解：SMO算法

➤ 第五步：首先将更新的 α_j 投影到区间 $[L, H]$ ，然后再利用线性约束关系进行求解 α_i ：

- 之前我们已经求得区间两端的值 L, H 是：

$$\begin{cases} y_i \neq y_j : & L = \max(0, \alpha_j^{\text{old}} - \alpha_i^{\text{old}}), \quad H = +\infty, \\ y_i = y_j : & L = 0, \quad H = \alpha_i^{\text{old}} + \alpha_j^{\text{old}}. \end{cases}$$

- 如果 α_j 落在区间内，则不变，反之则将其变为区间的两端的值

$$\alpha_j^{\text{new,clipped}} = \begin{cases} H, & \alpha_j^{\text{new}} > H, \\ \alpha_j^{\text{new}}, & L \leq \alpha_j^{\text{new}} \leq H, \\ L, & \alpha_j^{\text{new}} < L. \end{cases}$$

- 然后根据线性约束关系：

$$y_i \alpha_i + y_j \alpha_j = \text{const} \rightarrow \Delta \alpha_i = -\frac{y_j}{y_i} \Delta \alpha_j \rightarrow \Delta \alpha_i = -\frac{y_i y_j}{y_i^2} \Delta \alpha_j \rightarrow \Delta \alpha_i = -y_i y_j \Delta \alpha_j$$

- 可以推断：

$$\alpha_i^{\text{new}} = \alpha_i^{\text{old}} + y_i y_j (\alpha_j^{\text{old}} - \alpha_j^{\text{new}})$$

► 模型求解：SMO算法

- 第六步：更新偏置b。
 - 为什么更新偏置b？更新了 α_i, α_j （其余 α 不变）后，要求新的偏置 b_{new} 使得支持向量仍满足 KKT 条件（落在间隔边界上）常用的两个候选来自对 x_i 与 x_j 分别强制 $y_k f_{\text{new}}(x_k) = 1$ 求得。
 - 首先写出更新后的决策函数（旧的决策函数+增量部分）

$$f_{\text{new}} = \sum_k \alpha_k^{\text{new}} y_k K(x_k, x) + b_{\text{new}} = f_{\text{old}} + \Delta \alpha_i y_i K(x_i, x) + \Delta \alpha_j y_j K(x_j, x) + (b_{\text{new}} - b_{\text{old}})$$

- 在 $x=x_i$ 处代入并用KKT条件（若把 x_i 视为支持向量，要求落在边界）

$$y_i f_{\text{new}}(x_i) = 1$$

- 代入上式：

$$y_i [f_{\text{old}}(x) + \Delta \alpha_i y_i K(x_i, x) + \Delta \alpha_j y_j K(x_j, x) + (b_{\text{new}} - b_{\text{old}})] = 1$$

- 把 $f_{\text{old}}(x_i) = y_i + E_i$ 代入（因为 $E_i = f_{\text{old}}(x_i) - y_i$ ）并注意到 $y_i^2 = 1$ ，整理得到

$$\begin{aligned} b_{\text{new}} &= b_{\text{old}} - E_i - y_i \Delta \alpha_i K_{ii} - y_j \Delta \alpha_j K_{ij} \rightarrow \\ &= b_{\text{old}} - E_i - y_i (\alpha_i^{\text{new}} - \alpha_i^{\text{old}}) K_{ii} - y_j (\alpha_j^{\text{new}} - \alpha_j^{\text{old}}) K_{ij} \end{aligned}$$

- 同理，在 $x=x_j$ 处代入KKT条件并进行计算，最后整理得到：

$$b'_{\text{new}} = b_{\text{old}} - E_j - y_i (\alpha_i^{\text{new}} - \alpha_i^{\text{old}}) K_{ij} - y_j (\alpha_j^{\text{new}} - \alpha_j^{\text{old}}) K_{jj}$$

► 模型求解：SMO算法

■ 问题来了，根据 $x=x_i$ 以及 $x=x_j$ 得到两个候选的更新偏置 b ，到底选择哪一个呢？

- 回顾硬间隔SVM（无松弛、无上界【后面扩展可知】）下，KKT条件为：

$$\alpha_i = 0 \rightarrow y_i(w^T x_i + b) \geq 1$$

$$\alpha_i > 0 \rightarrow y_i(w^T x_i + b) = 1$$

- 因此如果 $\alpha_i > 0$ ($x=x_i$)，取：

$$b_1 = b_{\text{old}} - E_i - y_i(\alpha_i^{\text{new}} - \alpha_i^{\text{old}})K_{ii} - y_j(\alpha_j^{\text{new}} - \alpha_j^{\text{old}})K_{ij}$$

- 因此如果 $\alpha_j > 0$ ($x=x_j$)，取：

$$b_2 = b_{\text{old}} - E_j - y_i(\alpha_i^{\text{new}} - \alpha_i^{\text{old}})K_{ij} - y_j(\alpha_j^{\text{new}} - \alpha_j^{\text{old}})K_{jj}$$

- 若二者都不大于 0（少见的退化情形）或都大于0，取

$$b = \frac{b_1 + b_2}{2}$$

- 总结：

$$b_{\text{new}} = \begin{cases} b_1, & \alpha_i^{\text{new}} > 0, \\ b_2, & \alpha_j^{\text{new}} > 0, \\ \frac{b_1 + b_2}{2}, & \text{otherwise.} \end{cases}$$

注：硬间隔没有上界 C ，KKT给出“ $\alpha_k > 0 \Rightarrow y_k f(x_k) = 1$ ”，因此谁是支持向量 $(\alpha > 0)$ ，就用谁的边界等式解 b ；若两者都不稳或都不大于 0，就取 $(b_1+b_2)/2$ 更稳健。

► 模型求解：SMO算法

➤ 第七步：重复之前的步骤一直迭代直到收敛：

- 1) 检查所有 α 是否满足 KKT 条件，如果满足则算法停止，否则将继续选择违反KKT条件的样本进行更新。
- 2) 或者迭代次数达到上限。

➤ 总结：SMO算法步骤（硬间隔，下一章将扩展为软间隔）：

● 1. 初始化 $\alpha=0, b=0$,以及设置容差 ϵ 。

● 2.选择两个优化变量 (α_i, α_j) 。

● 3.根据线性约束条件求得 α_i, α_j 的取值范围：

$$\begin{cases} y_i \neq y_j : & L = \max(0, \alpha_j^{\text{old}} - \alpha_i^{\text{old}}), \quad H = +\infty, \\ y_i = y_j : & L = 0, \quad H = \alpha_i^{\text{old}} + \alpha_j^{\text{old}}. \end{cases}$$

● 4.根据牛顿迭代法快速求得 α_j

$$\alpha_j^{\text{new}} = \alpha_j^{\text{old}} - \frac{\frac{dW}{d\alpha_j}}{\frac{d^2W}{d\alpha_j^2}} = \alpha_j^{\text{old}} + \frac{y_j(E_i - E_j)}{K_{ii} + K_{jj} - 2K_{ij}}$$

● 5.然后将更新的 α_j 投影到区间 $[L, H]$ ，再利用线性约束关系进行求解 α_i :

$$\alpha_i^{\text{new}} = \alpha_i^{\text{old}} + y_i y_j (\alpha_j^{\text{old}} - \alpha_j^{\text{new}})$$

● 6.更新偏置b:

$$b_{\text{new}} = \begin{cases} b_1, & \alpha_i^{\text{new}} > 0, \\ b_2, & \alpha_j^{\text{new}} > 0, \\ \frac{b_1 + b_2}{2}, & \text{otherwise.} \end{cases}$$

● 7.重复步骤2~6，直到收敛。

► 模型求解：梯度下降法

➤ 梯度下降法：构建目标函数 → 计算梯度 → 参数更新 → 迭代收敛。

- 严格来说，硬间隔 SVM 是一个带不等式约束的二次规划问题：

$$\min \frac{1}{2} \|w\|_2^2 \quad s.t. \quad y_i(w^T x_i + b) \geq 1 \quad (i = 1, 2, \dots, m)$$

- 由于存在约束，梯度下降法不能直接在原始目标上无约束迭代，原因是梯度下降只适合无约束优化：它沿负梯度方向更新参数，但可能会越过约束边界，使 $y_i(w^T x_i + b) < 1$ 的约束被违反。

➤ 所以梯度下降法一般不适用于硬间隔SVM的问题，但是可以采用梯度下降的变体：投影梯度下降/惩罚方法。

- 投影梯度法：每次梯度更新后， w, b 投影回可行域，使所有约束满足
- 惩罚法 / Barrier 方法：把约束转换为目标的惩罚项，例如：

$$\min_{w,b} \frac{1}{2} \|w\|^2 + \sum_i \phi(1 - y_i(w^T x_i + b))$$

- 注意：这样就类似软间隔，原始硬间隔不再纯粹。（下一节将详细推导软间隔SVM的梯度下降法）

▶ 模型评估

- 训练好SVM模型后，我们需要知道它在**未见过的数据**上的表现，即是否能很好地泛化，而不是仅仅记住训练数据，由于SVM是分类模型，所以主要用**分类性能指标**进行评估：

| 类别 | 指标名称 | 公式 | 说明 |
|--------|---------|--|---------------------------|
| 分类性能指标 | 准确率 | $Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$ | 正确预测样本数 / 总样本数，整体正确性。 |
| | 精确率 | $Precision = \frac{TP}{TP + FP}$ | 在预测为正类的样本中，真正为正类的比例。 |
| | 召回率/灵敏度 | $Recall = \frac{TP}{TP + FN}$ | 在实际正类样本中，被正确预测为正类的比例。 |
| | F1分数 | $F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$ | 精确率与召回率的调和平均，综合考量模型性能。 |
| | ROC-AUC | $ROC : t \mapsto (FPR(t), TPR(t)), \quad t \in [0, 1]$ | 反映模型区分正负样本的能力，曲线下的面积越大越好。 |

- SVM 的评估不仅依赖常规分类指标，还可结合**交叉验证**【不仅用一个训练/测试划分，而是反复随机划分数据，保证评估结果更稳健。】、**支持向量数量**【太多支持向量说明模型可能过拟合】及**间隔大小**【间隔越大，理论上泛化性能越好。】来综合判断模型的稳健性与泛化能力。

SVM 的评估核心是验证其泛化能力与间隔最大化效果，常用指标包括分类问题的准确率、精确率、召回率、F1、AUC，以及结合支持向量数量与间隔大小来辅助判断模型的表现。

► 小结

- SVM 模型的求解主要依赖于凸二次规划（QP）方法，求解主要包含三种方法：内点法、活动集法、坐标下降法，其中最常用的坐标下降法的高效实现是序列最小优化（SMO）算法，它通过分解为两个变量的子问题并迭代更新来快速收敛。
 - 内点法（Interior Point Method）：通过在可行域内部构造路径逼近最优解，以连续优化方式高效求解凸二次规划问题。
 - 活动集法（Active Set Method）：在每步迭代中假设一部分约束为等式约束，通过更新活动集来逐步逼近最优解。
 - **坐标下降法（Coordinate Descent）**：固定除一个变量外的其他变量，沿该坐标方向优化，通过轮换迭代快速逼近全局最优解。
- SVM 模型的评估主要依赖于分类的常见指标（如准确率、精确率、召回率、F1、AUC、等），并结合交叉验证、支持向量数量和间隔大小等特有指标来全面衡量模型的泛化性能与复杂度。

SVM 的核心在于“解”与“评”——通过凸二次规划及其高效实现（如 SMO）求解最优间隔超平面，再用分类指标结合支持向量数量与间隔大小评估其泛化能力。

目录章节

CONTENTS

01

KNN的概念和原理

02

模型求解与评估

03

模型扩展与改进

04

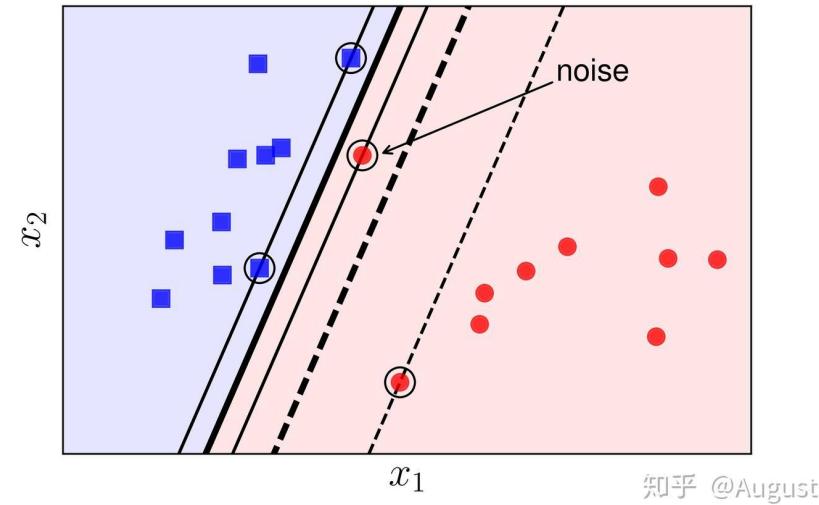
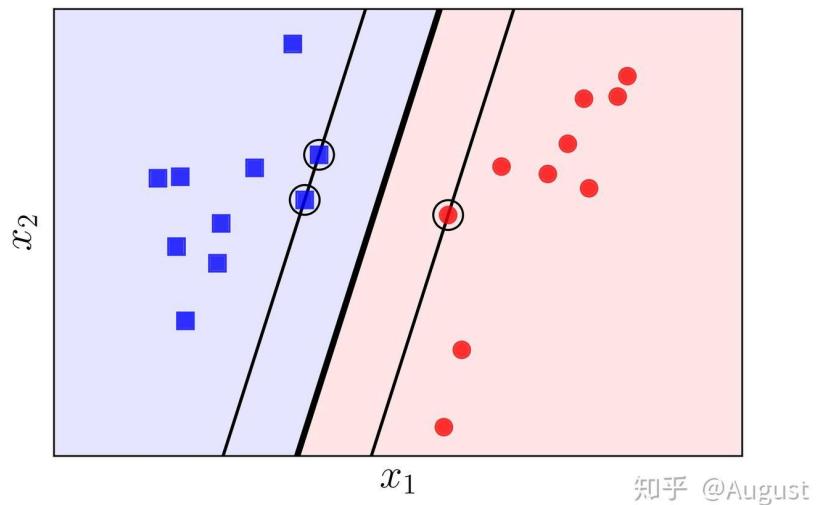
模型实现与实战

05

总结

► 模型扩展与改进——软间隔SVM

- 1) 软间隔 SVM: 允许部分样本违反间隔约束, 通过引入惩罚参数 C 控制误差, 提升对噪声数据的鲁棒性。
 - 怎么理解软间隔SVM? 当数据集中参杂了一些噪声, 如果继续用我们之间SVM模型(硬间隔)会导致决策从粗虚线变为粗实线。

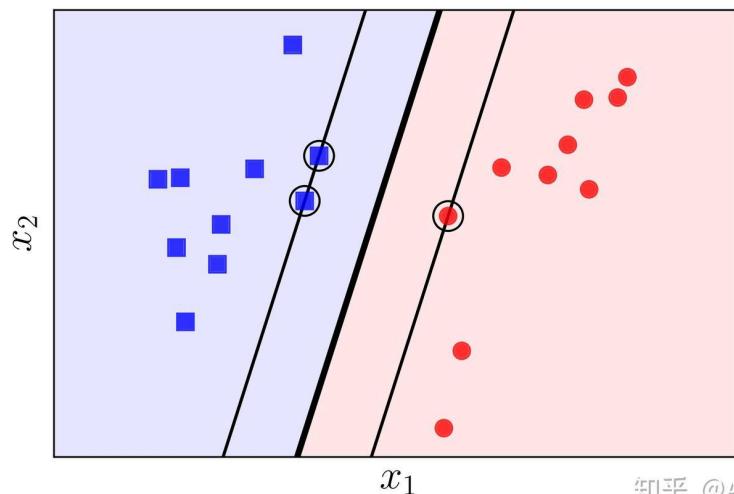


source: <https://zhuanlan.zhihu.com/p/36379394>

- 粗虚线的决策更好还是粗实现的决策更好? 综合来看, 答案是粗虚线的决策效果更好。之前我们提高模型的评估不仅需要看分类性能的指标, 也需要看SVM特有的评估指标即最大间隔如何, 明显粗虚线比粗实现的最大间隔更大。

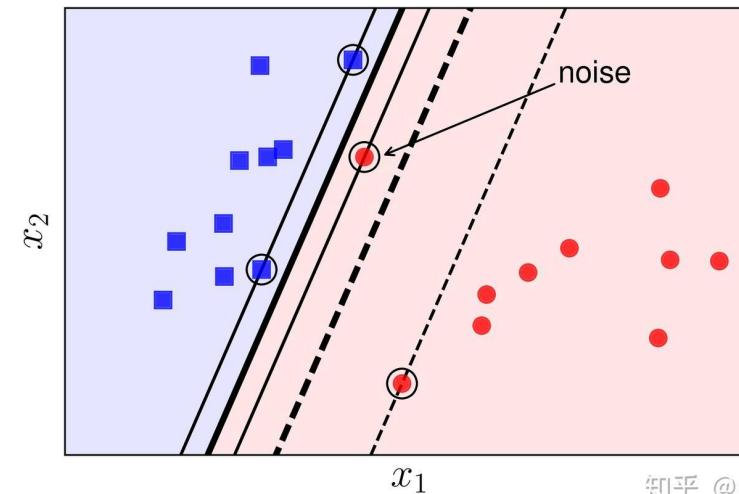
► 模型扩展与改进——软间隔SVM

- 1) 软间隔 SVM: 允许部分样本违反间隔约束, 通过引入惩罚参数 C 控制误差, 提升对噪声数据的鲁棒性。
 - 软间隔SVM的意义在哪里? 在于允许模型在噪声或不可完全线性分的数据上有一定容错, 通过引入惩罚参数C来平衡间隔最大化与误分类数量最小化。



知乎 @August

source: <https://zhuanlan.zhihu.com/p/36379394>

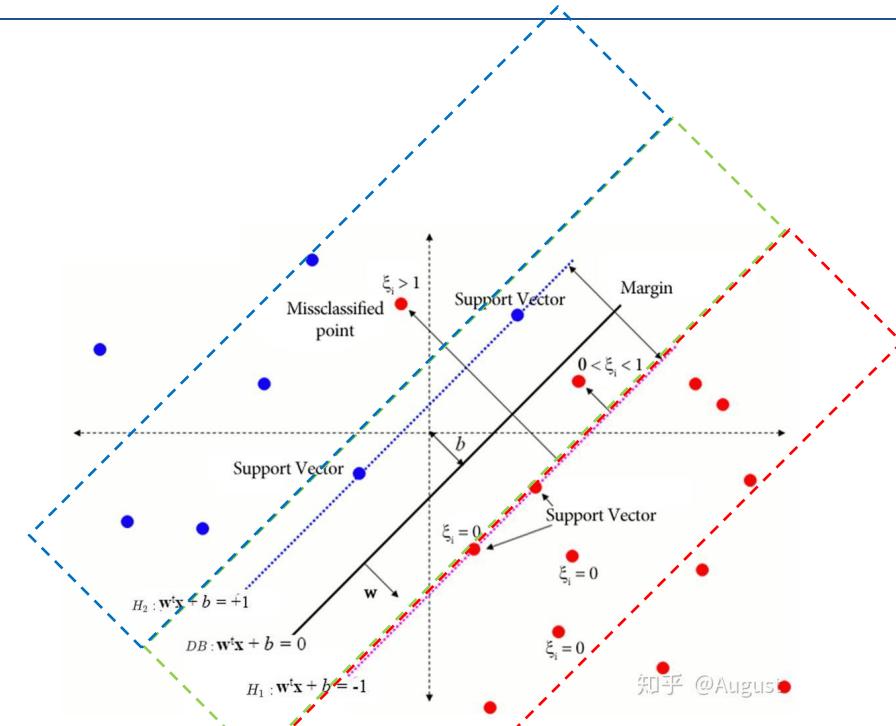
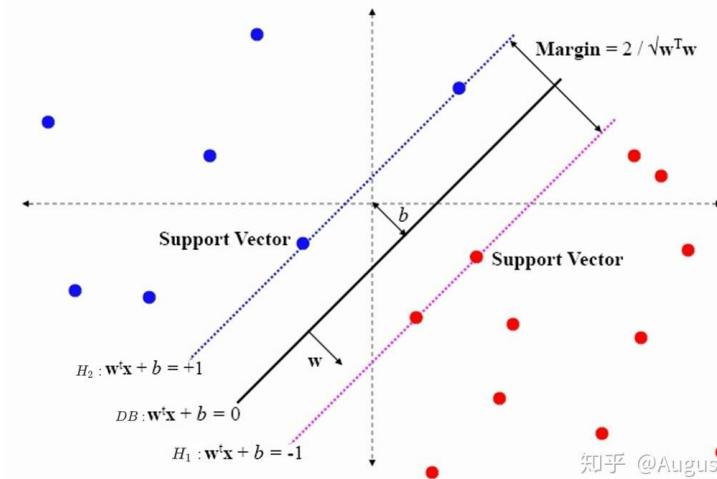


知乎 @August

- 软间隔SVM的作用? 提升模型的鲁棒性与泛化能力, 避免硬间隔 SVM 在实际数据中出现过拟合或无法求解的情况。即软间隔让 SVM “聪明地容忍错误”, 在噪声数据上仍能保持稳健的分类性能。

▶ 模型扩展与改进——软间隔SVM

➤ 怎么从硬间隔SVM改进为软间隔SVM?



- 所有的样本点均到决策超平面的距离均不小于1，硬间隔SVM的下限为：

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

- 当数据中存在噪声点或误分类点时，软间隔SVM为每个点引入了一个变量 ξ_i ，这个变量称为松弛变量，这个变量将所有点分为三类（针对右图右下侧支持向量构成的平面）：

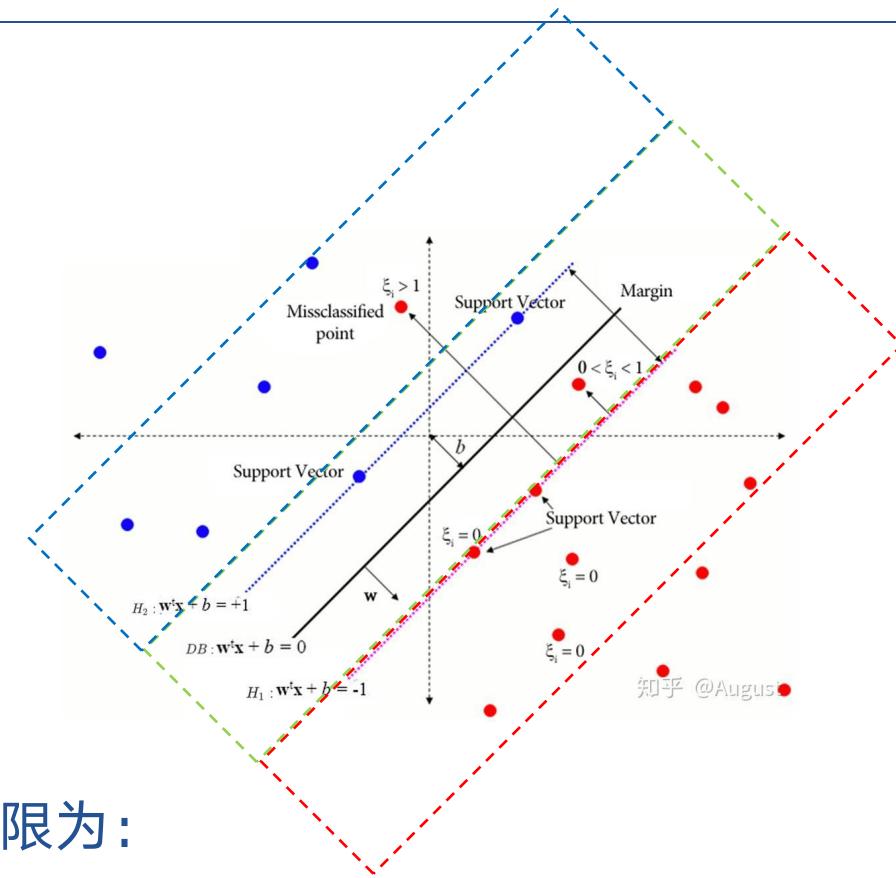
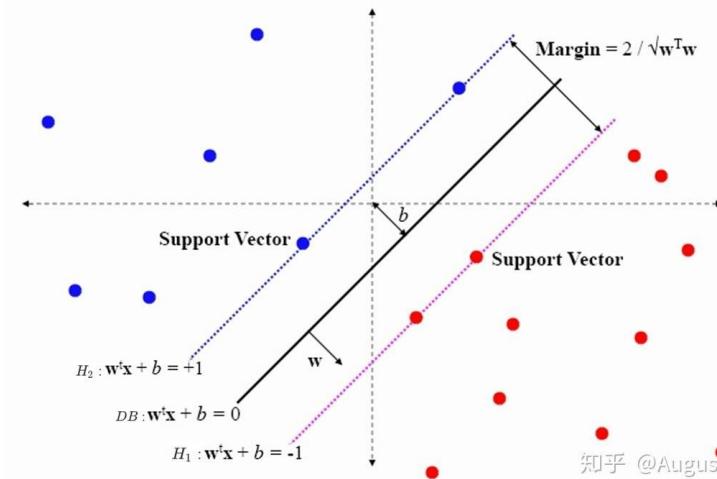
红色方框 $\xi=0$: 正确分类点, 该点在间隔之外 (分类正确且满足边界要求)。

绿色方框 $0 < \xi < 1$: 噪声点, 该点落在间隔内, 但分类仍正确。

蓝色方框 $\xi > 1$: 误分类点, 从标签来看本来应该属于另一类的

▶ 模型扩展与改进——软间隔SVM

➤ 怎么从硬间隔SVM改进为软间隔SVM?



- 加入松弛变量之后即称为软间隔SVM，该下限为：

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

- $1 - \zeta_i$ 的几何意义：代表样本点到决策超平面之间的有向距离。

红色方框 $1 - \zeta = 0$: 正确分类点到超平面的距离

绿色方框 $0 < 1 - \zeta < 1$: 噪声点到超平面的距离

蓝色方框 $1 - \zeta < 0$: 误分类点到超平面的距离

► 模型扩展与改进——软间隔SVM

➤ 软间隔SVM的原始问题推导（从硬间隔SVM开始）

■ 在没有噪声、数据线性可分的情况下，SVM 的目标是：

$$\min_{w,b} \frac{1}{2} \|w\|^2 \quad s.t. \quad y_i(w^T x_i + b) \geq 1, \quad i = 1, \dots, m$$

- 目标函数 $1/2\|w\|^2$: 最大化间隔。
- 约束 $y_i(w^T x_i + b) \geq 1$: 所有点都必须在边界外侧。

■ 为了解决不可分问题，引入松弛变量 ξ_i ，允许部分样本“违反间隔约束”：

$$y_i(w^T x_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0$$

■ 仅仅引入 ξ_i 还不够，还要惩罚这些违反约束的点。常见的做法是在目标函数里加入惩罚：

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

- 这第一项 $1/2\|w\|^2$ 保持间隔最大化，第二项 $C \sum \xi_i$ 惩罚违反约束的样本。其中超参数 $C > 0$ ，代表控制“间隔大小”与训练误差之间的权衡， C 大：更严格地惩罚错误 \rightarrow 间隔小但训练误差低； C 小：允许更多的误差 \rightarrow 间隔大但可能欠拟合。

● 综上，得到软间隔SVM的原始问题为：

$$\begin{aligned} & \min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\ & s.t. \quad y_i(w^T x_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, m, \\ & \quad \xi_i \geq 0, \quad i = 1, \dots, m, \\ & \quad C > 0 \end{aligned}$$

► 模型扩展与改进——软间隔SVM

➤ 首先就是需要利用拉格朗日乘子法将其转换为对偶问题

■ 由于软间隔SVM的原始问题为：

$$\begin{aligned} & \min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i \\ \text{s.t. } & y_i(w^\top x_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, m, \\ & \xi_i \geq 0, \quad i = 1, \dots, m, \\ & C > 0 \end{aligned}$$

● 根据拉格朗日乘子法：

$$\begin{aligned} \mathcal{L}(w, b, \xi; \alpha, \mu) = & \frac{1}{2} \|w\|^2 + C \sum_{i=1}^m \xi_i - \sum_{i=1}^m \alpha_i [y_i(w^\top x_i + b) - 1 + \xi_i] - \sum_{i=1}^m \mu_i \xi_i, \\ & \alpha_i \geq 0, \quad \mu_i \geq 0 \end{aligned}$$

● 对原始变量求驻点条件：

$$\frac{\partial \mathcal{L}}{\partial w} = 0 \quad \rightarrow \quad w = \sum_{i=1}^m \alpha_i y_i x_i \quad \frac{\partial \mathcal{L}}{\partial b} = 0 \quad \rightarrow \quad \sum_{i=1}^m \alpha_i y_i = 0 \quad \frac{\partial \mathcal{L}}{\partial \xi_i} = 0 \quad \rightarrow \quad C - \alpha_i - \mu_i = 0 \implies 0 \leq \alpha_i \leq C$$

● 带回拉格朗日化简：

$$\begin{aligned} \mathcal{L}(w, b, \xi; \alpha, \mu) = & \frac{1}{2} \|w\|^2 + C \sum_i \xi_i - \sum_i \alpha_i [y_i(w^\top x_i + b) - 1 + \xi_i] - \sum_i \mu_i \xi_i \\ \xrightarrow{\text{substitute } w, \text{ and eliminate } \xi, \mu} & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j K(x_i, x_j), \end{aligned}$$

● 整理得到最终的对偶问题为：

$$\boxed{\begin{aligned} \max_{\alpha} W(\alpha) = & \sum_{i=1}^m \alpha_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j K(x_i, x_j), \\ \text{s.t. } & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, m, \\ & \sum_{i=1}^m \alpha_i y_i = 0. \end{aligned}}$$

► 模型扩展与改进——软间隔SVM

➤ 对于一个凸优化+不等式约束的问题，最优解必须满足KKT条件，即：

- 1) 原始可行性 (Primal feasibility) :

$$y_i(w^T x_i + b) \geq \xi - 1, \quad \xi_i \geq 0, \quad i = 1, \dots, m$$

- 2) 对偶可行性 (Dual feasibility) :

$$\alpha_i \geq 0, \quad \mu_i \geq 0, \quad 0 \leq \alpha_i \leq C \quad i = 1, \dots, m$$

- 3) 驻点条件 (Stationarity) :

$$\frac{\partial L}{\partial w} \rightarrow w = \sum_{i=1}^m \alpha_i y_i x_i, \quad \frac{\partial L}{\partial b} \rightarrow \sum_{i=1}^m \alpha_i y_i = 0$$

- 4) 互补松弛条件 (Complementary slackness) :

$$\alpha_i(y_i(w^T x_i + b) - 1 + \xi_i) = 0, \quad \mu_i \xi_i = 0, \quad i = 1, \dots, m$$

- 根据互补松弛条件：可以推断两种情况（在软间隔条件下可以分为三种）：

$$\begin{cases} \alpha_i = 0 & \Rightarrow y_i(w^T x_i + b) \geq 1, \\ 0 < \alpha_i < C & \Rightarrow y_i(w^T x_i + b) = 1 - \xi_i, \\ \alpha_i = C & \Rightarrow y_i(w^T x_i + b) \leq 1 - \xi_i. \end{cases}$$

- $\alpha_i = 0$: 点在间隔外且分类正确； $0 < \alpha_i < C$: 点在间隔边界上，是支持向量； $\alpha_i = C$: 点在间隔内或被误分类。

► 模型扩展与改进——软间隔SVM

- 根据对偶问题推导+KKT条件能得到什么结论呢？
- 1) 根据互补松弛条件，可以得到不同 α 有不同的取值范围：

$$\begin{cases} \alpha_i = 0 & \Rightarrow y_i(w^\top x_i + b) \geq 1, \\ 0 < \alpha_i < C & \Rightarrow y_i(w^\top x_i + b) = 1 - \xi_i, \\ \alpha_i = C & \Rightarrow y_i(w^\top x_i + b) \leq 1 - \xi_i. \end{cases}$$

- 2) 根据站立性条件：可以推断出决策函数的形式
- 分类的超平面为：

$$f(x) = w^T x + b$$

- 代入驻点条件：

$$w = \sum_{i=1}^m \alpha_i y_i x_i$$

- 得到：

$$f(x) = (\sum_{i=1}^m \alpha_i y_i x_i)^T x + b = \sum_{i=1}^m \alpha_i y_i (x_i^T x) + b$$

- 如果使用核函数 $K(x_i, x) = x_i^T x$ ，推广成：

$$f(x) = \sum_{i=1}^m \alpha_i y_i K(x_i, x) + b$$

► 模型扩展与改进——软间隔SVM

- 软间隔SVM的求解类似与硬间隔SVM，但是某些步骤上有细微差异（红色标注）
- 第一步：初始化 α_i 和 b
 - 将所有 $\alpha_i=0$ ，初始化偏置 $b=0$ 。
 - 还可以设置容差 ϵ 判断KKT条件是否满足。
- 第二步：选择两个优化变量 (α_i, α_j)
 - 首先选择第一个变量 α_i ，也称为外循环，这个变量需要选择对目标函数影响最大且距离真实值最离谱的变量，也就是违反 KKT 约束条件最严重的样本点，即违反
$$\begin{cases} \alpha_i = 0 & \Rightarrow y_i(w^\top x_i + b) \geq 1, \\ 0 < \alpha_i < C & \Rightarrow y_i(w^\top x_i + b) = 1 - \xi_i, \\ \alpha_i = C & \Rightarrow y_i(w^\top x_i + b) \leq 1 - \xi_i. \end{cases}$$
 - 优先选择违反哪个条件的点？由于支持向量可以决定超平面，所以我们应该**优先选择 $0 < \alpha_i < C$ 的点**，即第二个，如果这些支持向量都满足KKT条件，再选择违反其他条件的点。
 - 然后选择第二个变量 α_j ，也称为内循环。 α_i 的选择在于满足 α_j 的变化最大，即 $|E_i - E_j|$ 最大值对应的 α_j 。其中 E_i 表示 α_i 的误差， E_j 表示 α_j 的误差。

► 模型扩展与改进——软间隔SVM

➤ 第三步：我们已经选择了两个拉格朗日乘子 α_i, α_j ，根据线性约束条件求得 α_i, α_j 的取值范围：

- 这是每次只优化两个拉格朗日乘子 α_i, α_j 【因为两个变量的子问题是二次函数，带边界约束，可以直接用解析公式求解】，这样可以保持线性约束不变。然后迭代所有样本，直到所有 α 满足 KKT 条件 \rightarrow SVM 收敛。
- 那么在更新这两个时，唯一需要保持的约束就是：

$$y_i\alpha_i + y_j\alpha_j + \sum_{k \neq i,j} y_k\alpha_k = 0$$

- 由于其余项是常数，等式约束可以简化为：

$$y_i\alpha_i^{\text{new}} + y_j\alpha_j^{\text{new}} = \text{const} = \zeta \rightarrow \alpha_i^{\text{new}} = \frac{\zeta - y_j\alpha_j^{\text{new}}}{y_i}$$

- 因为 $y_i \in \{-1, +1\}$ ，所以这就是一条直线；可行域还必须满足 $\alpha_i \geq 0, \alpha_j \geq 0$ 。
- a. 当 $y_i \neq y_j$ ，即 $y_i = -y_j$ （注意 $|y|=1$ ），由于

$$\zeta = y_i\alpha_i^{\text{old}} + y_j\alpha_j^{\text{old}} = y_i(\alpha_i^{\text{old}} - \alpha_j^{\text{old}})$$

- 带回可得：

$$\alpha_i^{\text{new}} = \frac{y_i(\alpha_i^{\text{old}} - \alpha_j^{\text{old}}) - (-y_i)\alpha_j^{\text{new}}}{y_i} = \alpha_i^{\text{old}} - \alpha_j^{\text{old}} + \alpha_j^{\text{new}} = \alpha_i^{\text{old}} + (\alpha_j^{\text{new}} - \alpha_j^{\text{old}})$$

► 模型扩展与改进——软间隔SVM

- 根据 $0 \leq \alpha_i, \alpha_j \leq C$:

$$\begin{aligned}\alpha_i^{\text{new}} &\geq 0 \rightarrow \alpha_j^{\text{new}} \geq \alpha_j^{\text{old}} - \alpha_i^{\text{old}} \\ \alpha_j^{\text{new}} &\geq 0\end{aligned}$$

- 合并得到下界:

$$L = \max(0, \alpha_j^{\text{old}} - \alpha_i^{\text{old}})$$

- 根据 $\alpha_i, \alpha_j \leq C$:

$$\begin{aligned}\alpha_j^{\text{new}} &\leq C \\ \alpha_i^{\text{new}} \leq C &\rightarrow \alpha_j^{\text{new}} \leq C + \alpha_j^{\text{old}} - \alpha_i^{\text{old}}\end{aligned}$$

- 合并得到上界:

$$H = \min(C, C + \alpha_j^{\text{old}} - \alpha_i^{\text{old}})$$

- b. 当 $y_i=y_j$ (注意 $|y|=1$) , 这时:

$$y_i(\alpha_i^{\text{new}} + \alpha_j^{\text{new}}) = \zeta \rightarrow \alpha_i^{\text{new}} + \alpha_j^{\text{new}} = \zeta' = \alpha_i^{\text{old}} + \alpha_j^{\text{old}}$$

- 即:

$$\alpha_i^{\text{new}} = \alpha_i^{\text{old}} + \alpha_j^{\text{old}} - \alpha_j^{\text{new}}$$

► 模型扩展与改进——软间隔SVM

- 同理，根据 $0 \leq \alpha_i, \alpha_j \leq C$

$$\begin{aligned}\alpha_i^{\text{new}} &\geq 0 \rightarrow \alpha_j^{\text{new}} \leq \alpha_i^{\text{old}} + \alpha_j^{\text{old}} \\ \alpha_i^{\text{new}} &\leq C \rightarrow \alpha_j^{\text{new}} \geq \alpha_i^{\text{old}} + \alpha_j^{\text{old}} - C \\ 0 \leq \alpha_j^{\text{new}} &\leq C\end{aligned}$$

- 因此上下界为

$$L = \max(0, \alpha_i^{\text{old}} + \alpha_j^{\text{old}} - C)$$

$$H = \min(C, \alpha_i^{\text{old}} + \alpha_j^{\text{old}})$$

- 总结：

$$\begin{cases} y_i \neq y_j : & L = \max(0, \alpha_j^{\text{old}} - \alpha_i^{\text{old}}), \quad H = \min(C, C + \alpha_j^{\text{old}} - \alpha_i^{\text{old}}), \\ y_i = y_j : & L = \max(0, \alpha_i^{\text{old}} + \alpha_j^{\text{old}} - C), \quad H = \min(C, \alpha_i^{\text{old}} + \alpha_j^{\text{old}}). \end{cases}$$

- 与硬间隔不同之处在于上下界被限制到盒约束[0,C]，因此H不再为 $+\infty$ 而是取与C的最小值、同时下界也由0与相关组合取最大值。

► 模型扩展与改进——软间隔SVM

- 第四步：根据 α 的上下界限，然后根据迭代法快速求得 α （与硬间隔SVM完全一致）
 - 之前根据目前关于 α 的函数（对偶问题的最终形式，将 i 用 p, q 代替便于后续的推导）：

$$W(\alpha) = \sum_{k=1}^m \alpha_k - \frac{1}{2} \sum_{p=1}^m \sum_{q=1}^m \alpha_p \alpha_q y_p y_q K_{pq}.$$

◆ SMO算法的想法是根据牛顿迭代法进行快速迭代更新（二次型），即

$$\alpha_j^{\text{new}} = \alpha_j^{\text{old}} - \frac{\frac{dW}{d\alpha_j}}{\frac{d^2W}{d\alpha_j^2}}$$

- 结合约束条件+边界条件（之前推导的 α 的上下界限），这就意味着这意味着 α_i 可以用 α_j 表示，因此 W 最终只依赖于 α_j 。所以目前的任务是求得 $W(\alpha)$ 的一阶导数和二阶导数：
- ◆ 先求二阶导数：我们首先把 W 中与 α_i, α_j 有关的二次项挑出来：

$$W(\alpha) = \sum_{k=1}^m \alpha_k + \boxed{\frac{1}{2} \sum_{p=1}^m \sum_{q=1}^m \alpha_p \alpha_q y_p y_q K_{pq}}.$$

- 再次观察对偶函数，这里重点在第二项，它是一个双重求和的二次项，包含所有 α 两两之间的乘积。
- 所以我们需要把所有和 α_i, α_j 表示有关的项挑出来，其他项作为常数放到一边。

► 模型扩展与改进——软间隔SVM

■ 展开求和的时候，涉及 α_i, α_j 的二次项有：

◆ 1) α_i^2 的项目，当p=i, q=i的时候：

$$-\frac{1}{2}\alpha_i^2 y_i^2 K(x_i, x_i) = -\frac{1}{2}K_{ii}\alpha_i^2$$

◆ 2) α_j^2 的项目，当p=j, q=j的时候：

$$-\frac{1}{2}\alpha_j^2 y_j^2 K(x_j, x_j) = -\frac{1}{2}K_{jj}\alpha_j^2$$

◆ 3) $\alpha_i\alpha_j$ 的交叉项目，这里分为两种情况：

● a.p=i, q=j的时候：

$$-\frac{1}{2}\alpha_i\alpha_j y_i y_j K(x_i, x_j)$$

● a.p=j, q=i的时候：

$$-\frac{1}{2}\alpha_j\alpha_i y_j y_i K(x_j, x_i)$$

● 由于 $K(x_i, x_j) = K(x_j, x_i)$ ，所以两个是一样的，加起来得到：

$$-\alpha_i\alpha_j y_i y_j K(x_i, x_j)$$

■ 最后合并起来，所以 α_i, α_j 相关的二次项就是：

$$Q(\alpha_i, \alpha_j) = -\frac{1}{2}[K_{ii}\alpha_i^2 + K_{jj}\alpha_j^2 + 2y_i y_j K_{ij}\alpha_i\alpha_j], \quad K_{ii} = K(x_i, x_i), K_{jj} = K(x_j, x_j), K_{ij} = K(x_i, x_j)$$

► 模型扩展与改进——软间隔SVM

- 下一步就是用约束关系 $\alpha_i^{\text{new}} = \frac{\zeta - y_j \alpha_j^{\text{new}}}{y_i} = \zeta' - y_i y_j \alpha_j^{\text{new}}$ 替换其中的一个变量，将其变为

$$Q(\alpha_i, \alpha_j) = -\frac{1}{2}[K_{ii}\alpha_i^2 + K_{jj}\alpha_j^2 + 2y_i y_j K_{ij}\alpha_i\alpha_j] \rightarrow Q(\alpha_j) = -\frac{1}{2}[K_{ii}(\zeta' - y_i y_j \alpha_j)^2 + K_{jj}(\alpha_j)^2 + 2y_i y_j K_{ij}\alpha_j(\zeta' - y_i y_j \alpha_j)]$$

- 展开二次项，关注 α_j^2 的系数

$$Q(\alpha_j) = \dots - \frac{1}{2}[(K_{ii} + K_{jj} - 2K_{ij})\alpha_j^2 + (\text{first-order term}) + \text{const}]$$

- 最终，二阶导数为：

$$\boxed{\frac{d^2W}{d\alpha_j^2} = -(K_{ii} + K_{jj} - 2K_{ij})}$$

- ◆ 为什么这么写？

- 因为核函数是个内积形式，代表两个变量在特征空间的相似程度：

$$K(x_p, x_q) = \langle \phi(x_p), \phi(x_q) \rangle$$

- 于是：

$$K_{ii} + K_{jj} - 2K_{ij} = \|\phi(x_i) - \phi(x_j)\|^2 \geq 0$$

- 这就是几何上的来源：实际上就是两个样本在特征空间里的“平方距离”
【核空间中两个点的距离平方】。

► 模型扩展与改进——软间隔SVM

- ◆ 二阶导数目前已经求出，还差一阶导数：

$$W(\alpha) = \sum_{k=1}^m \alpha_k - \boxed{\frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j K(x_i, x_j)}$$

- 直接进行求一阶导数，第一项直接：

$$\frac{\partial}{\partial \alpha_k} \left(\sum_{t=1}^m \alpha_t \right) = 1$$

- 计求和项目为S，乘以系数1/2为第二项T：

$$S := \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j K_{ij} \rightarrow T := \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i \alpha_j y_i y_j K_{ij}$$

- 首先对乘积进行求导，使用克罗内克：

$$\frac{\partial(\alpha_i \alpha_j)}{\partial \alpha_k} = \frac{\partial \alpha_i}{\partial \alpha_k} \alpha_j + \alpha_i \frac{\partial \alpha_j}{\partial \alpha_k} = \delta_{ik} \alpha_j + \alpha_i \delta_{jk}$$

$$\delta_{ik} = \begin{cases} 1 & \text{if } i = k, \\ 0 & \text{if } i \neq k. \end{cases}$$

- 因此：

$$\frac{\partial S}{\partial \alpha_k} = \sum_{i=1}^m \sum_{j=1}^m (\delta_{ik} \alpha_j + \alpha_i \delta_{jk}) y_i y_j K_{ij}$$

- 把两项拆开：

$$\frac{\partial S}{\partial \alpha_k} = \underbrace{\sum_{j=1}^m \alpha_j y_k y_j K_{kj}}_{(A)} + \underbrace{\sum_{i=1}^m \alpha_i y_i y_k K_{ik}}_{(B)}$$

► 模型扩展与改进——软间隔SVM

- 利用核函数的对称性合并两项，把A和B的求和索引重命名为l，可以看到两个是相同的

$$(A) = \sum_{l=1}^m \alpha_l y_k y_l K_{kl}, \quad (B) = \sum_{l=1}^m \alpha_l y_k y_l K_{lk} = \sum_{l=1}^m \alpha_l y_k y_l K_{kl}$$

- 因此

$$\frac{\partial S}{\partial \alpha_k} = 2 \sum_{l=1}^m \alpha_l y_k y_l K_{kl}$$

- 于是

$$\frac{\partial T}{\partial \alpha_k} = \frac{1}{2} \frac{\partial S}{\partial \alpha_k} = \sum_{l=1}^m \alpha_l y_k y_l K_{kl}$$

- 合并第一项和第二项的导数，最终得到：

$$\frac{\partial W}{\partial \alpha_k} = 1 - \sum_{l=1}^m \alpha_l y_k y_l K_{kl}$$

- 通常把 y_k 提出来写成更醒目的形式：

$$\boxed{\frac{\partial W}{\partial \alpha_k} = 1 - y_k \sum_{l=1}^m \alpha_l y_l K_{kl}}$$

► 模型扩展与改进——软间隔SVM

- 用j替换k后注意到后面求和的部分和决策函数的求和部分一致，即：

$$\frac{\partial W}{\partial \alpha_j} = 1 - y_k \sum_{l=1}^m \alpha_l y_l K_{jl}$$
$$f(x_j) = \sum_{l=1}^m \alpha_l y_l K_{lj} + b \rightarrow \sum_{l=1}^m \alpha_l y_l K_{lj} = f(x_j) - b$$

- 将上述公式合并后，可得：

$$\frac{\partial W}{\partial \alpha_j} = 1 - y_j(f(x_j) - b)$$

- 将误差定义为预测值-真实值：

$$E_j = f(x_j) - y_j$$

- 代入偏导中，得到：

$$\frac{\partial W}{\partial \alpha_j} = 1 - y_j((E_j + y_j) - b) = 1 - y_j E_j - y_j^2 + y_j b = -y_j E_j + y_j b$$

- 同理，对于ak有：

$$\frac{\partial W}{\partial \alpha_i} = -y_i E_i + y_i b$$

- 由于约束条件：

$$y_i \alpha_i + y_j \alpha_j = \text{const} \rightarrow \Delta \alpha_i = -\frac{y_j}{y_i} \Delta \alpha_j$$

► 模型扩展与改进——软间隔SVM

- 当 α_j 变化时， α_i 必须跟着联动，所以沿着约束方向的总导数：

$$\frac{dW}{d\alpha_j} = \frac{\partial W}{\partial \alpha_j} \frac{d\alpha_j}{d\alpha_j} + \frac{\partial W}{\partial \alpha_i} \frac{d\alpha_i}{d\alpha_j} = \frac{\partial W}{\partial \alpha_j} + \frac{\partial W}{\partial \alpha_i} \frac{d\alpha_i}{d\alpha_j} = \frac{\partial W}{\partial \alpha_j} + \frac{\partial W}{\partial \alpha_i} \left(-\frac{y_j}{y_i}\right)$$

- 根据之前推导的：

$$\frac{\partial W}{\partial \alpha_i} = -y_i E_i + y_i b$$

$$\frac{\partial W}{\partial \alpha_j} = -y_j E_j + y_j b$$

- 代入上式，得到：

$$\begin{aligned}\frac{dW}{d\alpha_j} &= (-y_j E_j + y_j b) + (-y_i E_i + y_i b) \left(-\frac{y_j}{y_i}\right) \\ &= y_j (E_i - E_j)\end{aligned}$$

- 将之前求得的二阶导数和上述的一阶导数代入之前牛顿迭代公式中，得到：

$$\alpha_j^{\text{new}} = \alpha_j^{\text{old}} - \frac{\frac{dW}{d\alpha_j}}{\frac{d^2W}{d\alpha_j^2}} = \alpha_j^{\text{old}} + \frac{y_j (E_i - E_j)}{K_{ii} + K_{jj} - 2K_{ij}}$$

- 需要注意的是，更新后的 α_j 必须落在之前我们推导的合法区间 $[L, H]$ 以内。

这条公式就是 SMO 的核心更新公式，把牛顿迭代和 SVM 的约束、误差信息完美结合在一起。

► 模型扩展与改进——软间隔SVM

➤ 第五步：首先将更新的 α_j 投影到区间 $[L, H]$ ，然后再利用线性约束关系进行求解 α_i ：

- 之前我们已经求得区间两端的值 L, H 是：

$$\begin{cases} y_i \neq y_j : & L = \max(0, \alpha_j^{\text{old}} - \alpha_i^{\text{old}}), \quad H = \min(C, C + \alpha_j^{\text{old}} - \alpha_i^{\text{old}}), \\ y_i = y_j : & L = \max(0, \alpha_i^{\text{old}} + \alpha_j^{\text{old}} - C), \quad H = \min(C, \alpha_i^{\text{old}} + \alpha_j^{\text{old}}). \end{cases}$$

- 如果 α_j 落在区间内，则不变，反之则将其变为区间的两端的值

$$\alpha_j^{\text{new,clipped}} = \begin{cases} H, & \alpha_j^{\text{new}} > H, \\ \alpha_j^{\text{new}}, & L \leq \alpha_j^{\text{new}} \leq H, \\ L, & \alpha_j^{\text{new}} < L. \end{cases}$$

- 然后根据线性约束关系：

$$y_i \alpha_i + y_j \alpha_j = \text{const} \rightarrow \Delta \alpha_i = -\frac{y_j}{y_i} \Delta \alpha_j \rightarrow \Delta \alpha_i = -\frac{y_i y_j}{y_i^2} \Delta \alpha_j \rightarrow \Delta \alpha_i = -y_i y_j \Delta \alpha_j$$

- 可以推断：

$$\alpha_i^{\text{new}} = \alpha_i^{\text{old}} + y_i y_j (\alpha_j^{\text{old}} - \alpha_j^{\text{new}})$$

► 模型扩展与改进——软间隔SVM

- 第六步：更新偏置b。
 - 为什么更新偏置b？更新了 α_i, α_j （其余 α 不变）后，要求新的偏置 b_{new} 使得支持向量仍满足KKT条件（落在间隔边界上）常用的两个候选来自对 x_i 与 x_j 分别强制 $y_k f_{\text{new}}(x_k) = 1$ 求得。
 - 由于无论硬间隔还是软间隔，对偶问题的决策函数形式都是（偏置b更新公式一致）

$$f_{\text{new}} = \sum_k \alpha_k^{\text{new}} y_k K(x_k, x) + b_{\text{new}} = f_{\text{old}} + \Delta \alpha_i y_i K(x_i, x) + \Delta \alpha_j y_j K(x_j, x) + (b_{\text{new}} - b_{\text{old}})$$

- 在 $x=x_i$ 处代入并用KKT条件（若把 x_i 视为支持向量，要求落在边界）

$$y_i f_{\text{new}}(x_i) = 1$$

- 代入上式：

$$y_i [f_{\text{old}}(x) + \Delta \alpha_i y_i K(x_i, x) + \Delta \alpha_j y_j K(x_j, x) + (b_{\text{new}} - b_{\text{old}})] = 1$$

- 把 $f_{\text{old}}(x_i) = y_i + E_i$ 代入（因为 $E_i = f_{\text{old}}(x_i) - y_i$ ）并注意到 $y_i^2 = 1$ ，整理得到

$$\begin{aligned} b_{\text{new}} &= b_{\text{old}} - E_i - y_i \Delta \alpha_i K_{ii} - y_j \Delta \alpha_j K_{ij} \rightarrow \\ &= b_{\text{old}} - E_i - y_i (\alpha_i^{\text{new}} - \alpha_i^{\text{old}}) K_{ii} - y_j (\alpha_j^{\text{new}} - \alpha_j^{\text{old}}) K_{ij} \end{aligned}$$

- 同理，在 $x=x_j$ 处代入KKT条件并进行计算，最后整理得到：

$$b'_{\text{new}} = b_{\text{old}} - E_j - y_i (\alpha_i^{\text{new}} - \alpha_i^{\text{old}}) K_{ij} - y_j (\alpha_j^{\text{new}} - \alpha_j^{\text{old}}) K_{jj}$$

► 模型扩展与改进——软间隔SVM

■ 问题来了，根据 $x=x_i$ 以及 $x=x_j$ 得到两个候选的更新偏置 b ，到底选择哪一个呢？

- 回顾硬间隔SVM（无松弛、无上界【后面扩展可知】）下，KKT条件为：

$$\begin{cases} \alpha_i = 0 & \Rightarrow y_i(w^\top x_i + b) \geq 1, \\ 0 < \alpha_i < C & \Rightarrow y_i(w^\top x_i + b) = 1 - \xi_i, \\ \alpha_i = C & \Rightarrow y_i(w^\top x_i + b) \leq 1 - \xi_i. \end{cases}$$

- 因此如果 $0 < \alpha_i < C$ ($x=x_i$)，取：

$$b_1 = b_{\text{old}} - E_i - y_i(\alpha_i^{\text{new}} - \alpha_i^{\text{old}})K_{ii} - y_j(\alpha_j^{\text{new}} - \alpha_j^{\text{old}})K_{ij}$$

- 因此如果 $0 < \alpha_j < C$ ($x=x_j$)，取：

$$b_2 = b_{\text{old}} - E_j - y_i(\alpha_i^{\text{new}} - \alpha_i^{\text{old}})K_{ij} - y_j(\alpha_j^{\text{new}} - \alpha_j^{\text{old}})K_{jj}$$

- 若二者都不大于 0（少见的退化情形）或都大于0，取

$$b = \frac{b_1 + b_2}{2}$$

- 总结：

$$b_{\text{new}} = \begin{cases} b_1, & 0 < \alpha_i^{\text{new}} < C, \\ b_2, & 0 < \alpha_j^{\text{new}} < C, \\ \frac{b_1 + b_2}{2}, & \text{otherwise.} \end{cases}$$

注：当 $\alpha_k=0$ 时样本非支持向量（对决策函数无贡献）；当 $\alpha_k=C$ 时该样本可能违反间隔或被惩罚（即 $\xi_k>0$ ），此时不保证 $y_k f(x_k)=1$ 。因此在更新偏置 b 时应按软间隔规则选择：若某个更新后 α_i 满足 $0 < \alpha_i < C$ 则用对应的 b_1 ；否则若 α_j 满足 $0 < \alpha_j < C$ 则用 b_2 ；若两者都在边界（等于0或等于C）或两者都落在(0,C)（数值上会非常接近），常以 $(b_1+b_2)/2$ 作为稳健选择。实际实现中还应加入数值容差（例如把“>0”改为 $>\varepsilon$ ，把“ $<C$ ”改为 $<C-\varepsilon$ ）以避免浮点误差导致的不稳定。

► 模型扩展与改进——软间隔SVM

➤ 第七步：重复之前的步骤一直迭代直到收敛：

- 1) 检查所有 α 是否满足 KKT 条件，如果满足则算法停止，否则将继续选择违反KKT条件的样本进行更新。
- 2) 或者迭代次数达到上限。

➤ 总结：SMO算法步骤（硬间隔，下一章将扩展为软间隔）：

- 1. 初始化 $\alpha=0, b=0$,以及设置容差 ϵ 。

- 2.选择两个优化变量 (α_i, α_j) 。

- 3.根据线性约束条件求得 α_i, α_j 的取值范围：

$$\begin{cases} \alpha_i = 0 & \Rightarrow y_i(w^\top x_i + b) \geq 1, \\ 0 < \alpha_i < C & \Rightarrow y_i(w^\top x_i + b) = 1 - \xi_i, \\ \alpha_i = C & \Rightarrow y_i(w^\top x_i + b) \leq 1 - \xi_i. \end{cases}$$

- 4.根据牛顿迭代法快速求得 α_j

$$\alpha_j^{\text{new}} = \alpha_j^{\text{old}} - \frac{\frac{dW}{d\alpha_j}}{\frac{d^2W}{d\alpha_j^2}} = \alpha_j^{\text{old}} + \frac{y_j(E_i - E_j)}{K_{ii} + K_{jj} - 2K_{ij}}$$

- 5.然后将更新的 α_j 投影到区间 $[L, H]$ ，再利用线性约束关系进行求解 α_i :

$$\alpha_i^{\text{new}} = \alpha_i^{\text{old}} + y_i y_j (\alpha_j^{\text{old}} - \alpha_j^{\text{new}})$$

- 6.更新偏置 b :

$$b_{\text{new}} = \begin{cases} b_1, & 0 < \alpha_i^{\text{new}} < C, \\ b_2, & 0 < \alpha_j^{\text{new}} < C, \\ \frac{b_1 + b_2}{2}, & \text{otherwise.} \end{cases}$$

- 7.重复步骤2~6，直到收敛。

► 模型扩展与改进——软间隔SVM（梯度下降法）

- 上一节提到硬间隔SVM不适用于梯度下降法，但是软间隔SVM可以使用梯度下降法求解
 - 软间隔SVM的二次规划形式为：

$$\min_{w,b,\xi} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \xi \quad s.t. \quad y_i(w^T x_i + b) \geq 1 - \xi, \quad \xi \geq 0$$

- 第一项表示仍在最大间隔/控制复杂度；第二项表示度量落入间隔或分类错误的总量。
- 消去松弛变量 \Rightarrow 铰链损失出现

- 对固定的(w, b)，第*i*个样本的 ξ_i 只出现在

$$\min_{\xi_i} C \xi_i \quad s.t. \quad \xi_i \geq 0, \quad \xi_i \geq 1 - y_i(w^T x_i + b)$$

- 该线性规划的最优是取最小可行值：

$$\xi_i^* = \max(0, 1 - y_i(w^T x_i + b))$$

- 把 ξ^* 带回目标函数，得到无约束问题：

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \max(0, 1 - y_i(w^T x_i + b))$$

- 记

$$\ell_{\text{hinge}}(y, z) = \max(0, 1 - yz)$$

- 即得经典的“正则化 + 铰链损失”形式：

- 等价性要点：对每个样本， ξ^* 的可行域是 $[\max(0, 1-y_i f_i), \infty]$ 。目标中它的系数为 $C > 0$ ，因此最优必取下界，恰好产生铰链形式。

$$\min_{w,b} \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n \ell_{\text{hinge}}(y_i, w^T x_i + b)$$

► 模型扩展与改进——软间隔SVM（梯度下降法）

➤ 下一步进行梯度计算

- 目标函数：

$$J(w, b) = \frac{1}{2} \|w\|^2 + C \sum_{i=1}^n L_i(w, b), \quad L_i = \max(0, 1 - y_i(w^T x_i + b))$$

- 对每个样本梯度：

$$\nabla_w L_i = \begin{cases} 0, & \text{if } y_i(w^T x_i + b) \geq 1 \\ -y_i x_i, & \text{if } y_i(w^T x_i + b) < 1 \end{cases} \quad \nabla_b L_i = \begin{cases} 0, & \text{if } y_i(w^T x_i + b) \geq 1 \\ -y_i, & \text{if } y_i(w^T x_i + b) < 1 \end{cases}$$

- 整体梯度：

$$\nabla_w J = w + C \sum_{i=1}^n \nabla_w L_i, \quad \nabla_b J = C \sum_{i=1}^n \nabla_b L_i$$

➤ 梯度下降更新公式

$$\begin{aligned} w^{(t+1)} &= w^{(t)} - \eta \nabla_w J(w^{(t)}, b^{(t)}) \\ b^{(t+1)} &= b^{(t)} - \eta \nabla_b J(w^{(t)}, b^{(t)}) \end{aligned}$$

- 可以改为随机梯度下降（SGD），每次只用一个或几个样本计算梯度，加速训练。

➤ 收敛条件

- 迭代直到 $\|w^{(t+1)} - w^{(t)}\| < \epsilon$ 或最大迭代次数，最终得到 w^* , b^* ，支持向量对应的样本是满足 $y_i(w^T x_i + b) \leq 1$ 的样本。

► 模型扩展与改进——其他

- 1) 核 SVM (Kernel SVM)
 - 利用核函数 $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$ 将数据映射到高维特征空间，使得在原空间线性不可分的问题在高维空间变得线性可分。
 - 常见核函数：多项式核、径向基函数核（RBF）、sigmoid 核。
- 2) ν-SVM (Nu-SVM)
 - 用参数 $\nu \in (0,1]$ 替代 C ，它能同时控制支持向量的比例与允许的训练误差比例。
 - 在某些情况下比 C-SVM 更直观、易于调节。
- 3) 支持向量回归 (SVR)
 - 将 SVM 思想推广到回归问题，引入 ε -不敏感损失函数，使得预测值只要落在真实值 $\pm \varepsilon$ 的区间内就不算误差。
- 4) 结构化 SVM (Structured SVM)
 - 用于预测复杂输出（如序列、树、图结构），广泛应用于自然语言处理（NLP）、计算机视觉等领域。
- 5) 一类 SVM (One-Class SVM)
 - 用于异常检测 / 新颖性检测，仅用正类样本训练，学习一个边界将大部分训练样本包含在内，检测偏离边界的点作为异常。

SVM 的扩展包括 软间隔 SVM（解决噪声问题）、核 SVM（处理非线性）、ν-SVM（参数化更直观）、SVR（回归问题）、结构化 SVM（复杂输出）、一类 SVM（异常检测），大大拓展了 SVM 的应用范围。

► 小结

- SVM 的扩展从“软间隔”解决噪声，到“核方法”解决非线性，再到 ν -SVM、SVR、结构化 SVM、一类 SVM，逐步拓展到分类、回归、复杂预测和异常检测等多领域。

| 扩展形式 | 核心思想 | 主要作用/场景 |
|---------------------------|---------------------------|-----------------|
| 软间隔 SVM (Soft Margin SVM) | 引入松弛变量 ξ_i ，用C平衡间隔与误差 | 解决噪声、非完全线性可分问题 |
| 核 SVM (Kernel SVM) | 利用核函数将数据映射到高维特征空间 | 处理非线性可分问题 |
| ν -SVM (Nu-SVM) | 参数 ν 控制支持向量比例与误差比例 | 更直观的参数调节 |
| 支持向量回归 (SVR) | ε -不敏感损失 | 用于回归问题，预测连续值 |
| 结构化SVM (Structured SVM) | 输出复杂结构（序列、树、图） | NLP、图像识别等复杂预测任务 |
| 一类SVM (One-Class SVM) | 仅用正类数据学习边界 | 异常检测、新颖性检测 |

SVM 的核心是“最大间隔 + 核方法”，其扩展通过“软间隔、多分类、稀疏化与核技巧”不断增强适应性与泛化能力。

目录章节

CONTENTS

01

KNN的概念和原理

02

模型求解与评估

03

模型扩展与改进

04

模型实现与实战

05

总结

▶ 算法实现：numpy实现

➤ 实现：MySVM类，第一步：先定义类及初始化方法及内部（私有_）方法。

```
import numpy as np

class MySVM:
    def __init__(self, C=1.0, tol=1e-3, max_iter=1000, kernel='linear',
                 method='SMO', sigma=1.0, use_sgd=False, eta=0.001):
        """
        C: 惩罚系数
        tol: KKT容差
        max_iter: 最大迭代次数
        kernel: 'linear' 或 'rbf'
        method: 'SMO' 或 'GD'
        sigma: RBF核带宽
        use_sgd: 仅在GD方法下有效，是否使用随机梯度下降
        eta: 学习率(GD方法)
        """
        self.C = C
        self.tol = tol
        self.max_iter = max_iter
        self.kernel_type = kernel
        self.method = method
        self.sigma = sigma
        self.use_sgd = use_sgd
        self.eta = eta

    def _kernel(self, x1, x2):
        if self.kernel_type == 'linear':
            return np.dot(x1, x2)
        elif self.kernel_type == 'rbf':
            return np.exp(-np.linalg.norm(x1 - x2) ** 2 / (2 * self.sigma ** 2))
        else:
            raise ValueError("Unknown kernel type")

    def _compute_kernel_matrix(self, X):
        m = X.shape[0]
        K = np.zeros((m, m))
        for i in range(m):
            for j in range(m):
                K[i, j] = self._kernel(X[i], X[j])
        return K

    def _E(self, i):
        return self.predict_one(self.X[i]) - self.y[i]

    def _select_j(self, i, m, Ei):
        max_diff = -1
        j = -1
        Ej = 0
        for k in range(m):
            if k == i:
                continue
            Ek = self._E(k)
            diff = abs(Ei - Ek)
            if diff > max_diff:
                max_diff = diff
                j = k
                Ej = Ek
        if j == -1:
            j = np.random.randint(0, m)
            while j == i:
                j = np.random.randint(0, m)
            Ej = self._E(j)
        return j, Ej
```

- **类定义及初始化：**可选参数包括C值、容差值、最大迭代次数、核函数种类、高斯核带宽。
- **_kernel方法（_表示私有）：**根据核函数种类定义核函数实现，包括'linear'（线性核）、'rbf'（RBF核）。
- **_compute_kernel_matrix方法（_表示私有）：**计算核矩阵。
- **_E方法（_表示私有）：**计算误差（预测值-真实值）
- **_select_j方法（_表示私有）：**选择第二个变量j的实现。

▶ 算法实现：numpy实现

➤ 实现：MySVM类，第二步：实现核心的模型训练和预测方法：fit(X, y), predict(X)。

```
def fit(self, X, y):
    m = X.shape[0]
    y = y.astype(float)
    self.X, self.y = X, y

    if self.method == 'SMO':
        self.alpha = np.zeros(m)
        self.b = 0
        self.K = self._compute_kernel_matrix(X)
        iters = 0
        while iters < self.max_iter:
            alpha_changed = 0
            for i in range(m):
                Ei = self._E(i)
                if (y[i]*Ei < -self.tol and self.alpha[i] < self.C) or \
                    (y[i]*Ei > self.tol and self.alpha[i] > 0):
                    j, Ej = self._select_j(i, m, Ei)
                    alpha_i_old, alpha_j_old = self.alpha[i], self.alpha[j]

                    # 计算边界
                    if y[i] != y[j]:
                        L = max(0, self.alpha[j] - self.alpha[i])
                        H = min(self.C, self.C + self.alpha[j] - self.alpha[i])
                    else:
                        L = max(0, self.alpha[i] + self.alpha[j] - self.C)
                        H = min(self.C, self.alpha[i] + self.alpha[j])
                    if L == H:
                        continue

                    # eta
                    eta = self.K[i,i] + self.K[j,j] - 2*self.K[i,j]
                    if eta <= 0:
                        continue

                    # 更新 alpha_j
                    self.alpha[j] += y[j]*(Ei - Ej)/eta
                    self.alpha[j] = np.clip(self.alpha[j], L, H)
                    if abs(self.alpha[j] - alpha_j_old) < 1e-5:
                        continue
                    # 更新 alpha_i
                    self.alpha[i] += y[i]*y[j]*(alpha_j_old - self.alpha[j])

                    # 更新偏置 b
                    b1 = self.b - Ei - y[i]*(self.alpha[i]-alpha_i_old)*self.K[i,i] - \
                        y[j]*(self.alpha[j]-alpha_j_old)*self.K[i,j]
                    b2 = self.b - Ej - y[i]*(self.alpha[i]-alpha_i_old)*self.K[j,j] - \
                        y[j]*(self.alpha[j]-alpha_j_old)*self.K[i,j]
                    if 0 < self.alpha[i] < self.C:
                        self.b = b1
                    elif 0 < self.alpha[j] < self.C:
                        self.b = b2
                    else:
                        self.b = (b1 + b2)/2

                    alpha_changed += 1
                    if alpha_changed == 0:
                        iters += 1
                    else:
                        iters = 0

    elif self.method == 'GD':
        self.w = np.zeros(X.shape[1])
        self.b = 0
        for iteration in range(self.max_iter):
            if self.use_sgd:
                indices = np.random.permutation(m)
            else:
                indices = np.arange(m)

            dw = np.zeros_like(self.w)
            db = 0
            for i in indices:
                condition = y[i]*(np.dot(self.w, X[i]) + self.b)
                if condition < 1:
                    dw -= self.C * y[i] * X[i]
                    db -= self.C * y[i]
                dw += self.w
                self.w -= self.eta * dw
                self.b -= self.eta * db
            else:
                raise ValueError("Unsupported method")
        return self
```

```
def predict_one(self, x):
    if self.method == 'GD':
        return np.dot(self.w, x) + self.b
    else:
        result = 0
        for i in range(len(self.alpha)):
            if self.alpha[i] > 1e-6:
                result += self.alpha[i]*self.y[i]*self._kernel(self.X[i], x)
        return result + self.b

def predict(self, X):
    return np.sign(np.array([self.predict_one(x) for x in X]))
```

- SVM训练流程：1) SMO算法：遍历训练样本，检查是否违反 KKT 条件->若违反条件，则选择第二个样本点（根据最大 $|E_i - E_j|$ 原则）->计算一对拉格朗日乘子 α_i, α_j 的可行范围L、H->更新 α_j ，并相应修正 α_i ->更新偏置项b，保证间隔条件尽量满足。->重复迭代，直到所有样本基本满足 KKT 条件或达到最大迭代次数；2) 梯度下降算法：构建目标函数 → 计算梯度 → 参数更新 → 迭代收敛。

- SVM预测流程：遍历测试样本->计算其与所有支持向量的核函数结果，并加权求和->根据 $f(x)$ 的符号决定类别： $y_{pred} = \text{sign}(f(x))$ ->输出所有预测结果

► 算法实现：numpy实现

- 实现：MySVM类，第三步：实现评估方法。

```
def score(self, X, y):  
    """ 计算准确率 """  
    y_pred = self.predict(X)  
    return np.mean(y_pred == y)
```

- 这里只实现了分类的作用，所以只实现分类的准确率指标。

▶ 算法实现：pytorch实现（练习）

➤ 利用Pytorch实现关键步骤：

```
def fit(self, X, y):
    X = torch.tensor(X, dtype=torch.float32)
    y = torch.tensor(y, dtype=torch.float32)
    self.X = X
    self.y = y
    n_samples, n_features = X.shape

    if self.method == 'GD':
        # 线性SVM权重和偏置
        self.w = nn.Parameter(torch.zeros(n_features))
        self.b = nn.Parameter(torch.zeros(1))
        optimizer = optim.SGD([self.w, self.b], lr=self.eta)

    for it in range(self.max_iter):
        if self.use_sgd:
            indices = torch.randperm(n_samples)
        else:
            indices = torch.arange(n_samples)

        for i in indices:
            optimizer.zero_grad()
            xi, yi = X[i], y[i]
            # 计算整体损失函数（正则化项+铰链损失项）
            margin = None # TODO, 计算margin, 提示: yi * (xi @ w + b)
            loss = None # TODO: 计算损失(正则化项目), 提示: 1/2 * ||w||^2
            if margin < 1:
                loss += None # 计算损失(铰链损失项), 提示: C * max(0, 1 - margin)
            loss.backward()
            optimizer.step()
```

```
elif self.method == 'SMO':
    # alpha参数初始化
    self.alpha = torch.zeros(n_samples, dtype=torch.float32)
    self.b = torch.tensor(0.0, dtype=torch.float32)
    K = self._kernel(X, X)

    iters = 0
    while iters < self.max_iter:
        alpha_changed = 0
        for i in range(n_samples):
            Ei = torch.sum(self.alpha * y * K[:, i]) + self.b - y[i]
            if (y[i]*Ei < -self.tol and self.alpha[i] < self.C) or \
                (y[i]*Ei > self.tol and self.alpha[i] > 0):
                # 选择第二个变量j
                j, Ej = self._select_j(i, n_samples, Ei)

                alpha_i_old = self.alpha[i].clone()
                alpha_j_old = self.alpha[j].clone()

                # 计算边界L, H (分类两种情况)
                if y[i] != y[j]:
                    L = None # TODO: 计算L
                    H = None # TODO: 计算H
                else:
                    L = None # TODO: 计算L
                    H = None # TODO: 计算H
                if L == H:
                    continue

                eta = None # TODO: 计算eta, 提示: Kii + Kjj - 2 * Kij
                if eta <= 0:
                    continue

                # 更新alpha[j]并限制在[L, H]之间
                self.alpha[j] += None # TODO: 更新alpha[j], 提示: yj * (Ei - Ej) / KiiKjj - 2*Kij(之前
                self.alpha[j] = None # TODO: 限制alpha[j]在[L, H]之间, 提示: torch.clamp()方法
                if torch.abs(self.alpha[j] - alpha_j_old) < 1e-5:
                    continue
                # 更新alpha[i]
                self.alpha[i] += None # TODO: 更新alpha[i], 提示: yi * yj * (alpha_i_old - alpha_j_new)

                # 更新偏置b
                b1 = None # TODO: 计算b1, 提示: self.b - Ei - y[i]*(self.alpha[i]-alpha_i_old)*K[i,i] -
                b2 = None # TODO: 计算b2, 提示: self.b - Ej - y[i]*(self.alpha[i]-alpha_i_old)*K[i,j] -
                if 0 < self.alpha[j] < self.C:
                    pass # TODO: 更新偏置b, 提示: b1赋给对象的b属性
                elif 0 < self.alpha[j] < self.C:
                    pass # TODO: 更新偏置b, 提示: b2赋给对象的b属性
                else:
                    pass # TODO: 更新偏置b, 提示: 平均(b1, b2)赋给对象的b属性

                alpha_changed += 1
                if alpha_changed == 0:
                    iters += 1
                else:
                    iters = 0
            else:
                raise ValueError("Unsupported method")
    return self
```

● 任务1:计算梯度下降法中的整体损失函数。

● 任务2：计算边界 L, H 。

● 任务3：更新 α_i, α_j 。

● 任务3：更新偏置b。

▶ 算法实战：手写数字识别

- 如之前一样，首先第一步就是导入数据，并查看数据的情况，决定下一步该干什么。

```
import pandas as pd

# 读取数据
data = pd.read_csv('train.csv')
texts = data['text'].values
labels = data['label'].values

# 查看数据
print("数据样本: ", texts[:5])
print("标签样本: ", labels[:5])
print("数据集大小: ", len(texts))
```

数据样本: ['I rented I AM CURIOUS-YELLOW from my video store because of all the controversy that surrounded "I Am Curious: Yellow" is a risible and pretentious steaming pile. It doesn\'t matter what one\'s politics are, if you can\'t stand it, just turn it off. If only to avoid making this type of film in the future. This film is interesting as an experiment but that\'s about it.', 'If only to avoid making this type of film in the future. This film is interesting as an experiment but that\'s about it.', 'This film was probably inspired by Godard\'s *Masculin, féminin* and I urge you to see that film instead. ', 'Oh, brother...after hearing about this ridiculous film for umpteen years all I can think of is that old', '标签样本: [0 0 0 0 0]

数据集大小: 25000

- 数据由两部分包含而成，一部分是完整的文本，一部分是情感标签（0/1代表消极/积极）。
- 当然不能直接把文本送入机器学习模型进行学习，需要将文本拆分成之前我们熟悉的每一列特征。

▶ 算法实战：房价预测

- 根据上面的分析，进行数据预处理，并进行数据集划分（训练集、测试集）。

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
# 文本向量化
# SVM 不能直接处理文本，所以需要把文本转成数值向量，这里用 TF-IDF:
# max_features=5000: 保留最重要的 5000 个词; stop_words='english': 去掉英文停用词
vectorizer = TfidfVectorizer(max_features=5000, stop_words='english')
X = vectorizer.fit_transform(texts)
y = labels

print("文本向量化: ", X[:5, :5].toarray())
print("文本向量化后形状: ", X.shape)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

文本向量化: [[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0.09559651 0. 0.]
 [0. 0. 0. 0. 0.]]
文本向量化后形状: (25000, 5000)

- 预处理之后，特征变为5000【保留了5000个最重要的词】

▶ 算法实战：房价预测

- 定义模型进行训练、预测，最后通过准确率、召回率、F1分数等指标评估效果。

```
from sklearn.metrics import accuracy_score, classification_report
from sklearn.svm import SVC
svm_model = SVC(kernel='linear', C=1.0) # 线性核效果通常很好
svm_model.fit(X_train, y_train)

# 预测
y_pred = svm_model.predict(X_test)

print("SVM预测准确率:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

SVM预测准确率: 0.881

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.89 | 0.87 | 0.88 | 2515 |
| 1 | 0.87 | 0.89 | 0.88 | 2485 |
| accuracy | | | 0.88 | 5000 |
| macro avg | 0.88 | 0.88 | 0.88 | 5000 |
| weighted avg | 0.88 | 0.88 | 0.88 | 5000 |

- 各项指标均在0.85以上，说明在训练集拟合效果较好。

▶ 算法实战：房价预测

- 为了测试模型的鲁棒性以及泛化性，可以进一步在测试集对训练好的模型进行预测评估：

```
# 读取外部测试集并进行预测评估
test_data = pd.read_csv("test.csv")
X_test_texts = test_data['text'].values
X_test = vectorizer.transform(X_test_texts)
y_test_true = test_data['label'].values

y_test_pred = svm_model.predict(X_test)

print("外部测试集合的SVM预测准确率:", accuracy_score(y_test_true, y_test_pred))
print(classification_report(y_test_true, y_test_pred))
```

外部测试集合的SVM预测准确率: 0.86852

| | precision | recall | f1-score | support |
|--------------|-----------|--------|----------|---------|
| 0 | 0.87 | 0.87 | 0.87 | 12500 |
| 1 | 0.87 | 0.87 | 0.87 | 12500 |
| accuracy | | | 0.87 | 25000 |
| macro avg | 0.87 | 0.87 | 0.87 | 25000 |
| weighted avg | 0.87 | 0.87 | 0.87 | 25000 |

- 各项指标均在0.85以上，说明在外部测试集拟合效果较好，模型泛化效果较好。

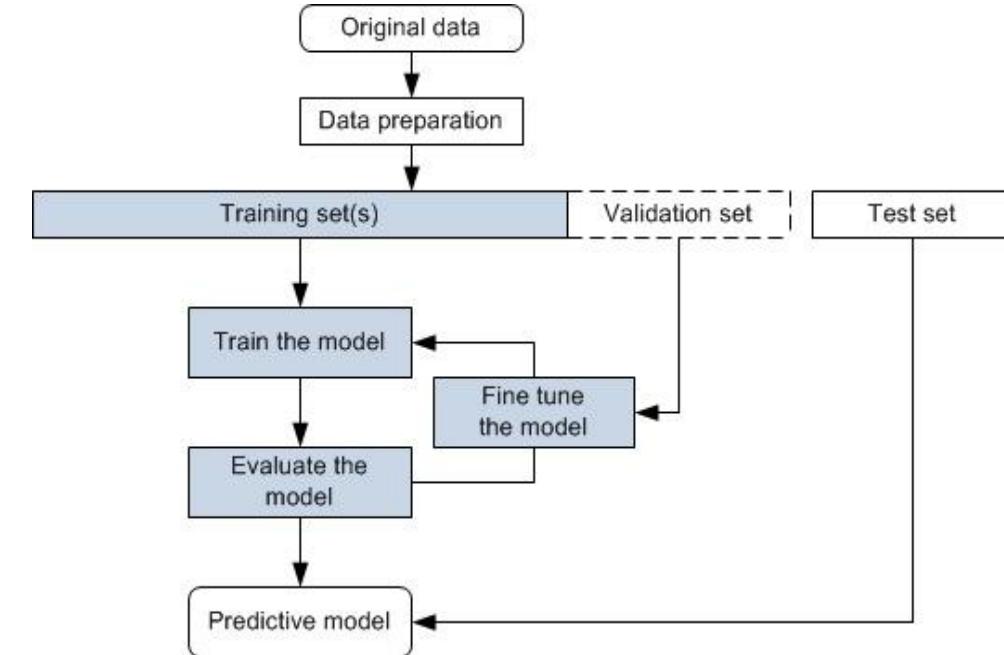
▶ 小结

➤ 模型实现：

- 主要的方法：初始化参数（`__init__`），拟合方法（`fit`），预测方法（`predict`），评估指标（`score`），实现这些方法的私有方法（`_xxx`）。

➤ 模型实战：

- 数据准备：导入数据、**将文本转换为特征**。
- 数据划分：训练集 / 测试集（常用70%/30%或80%/20%）。
- 特征预处理：标准化/归一化/去NULL值/独热编码等。
- 模型训练：SVM训练找最优分割超平面，预测靠超平面分类，支持核函数处理非线性。
- 模型评估：通过准确率，精准率，召回率，F1分数等进行评估。



KNN 的实现核心是保存训练样本和标签，在预测时通过计算距离找到最近邻，并根据权重策略投票或平均输出预测结果，同时通过评价指标评估模型性能。

目录章节

CONTENTS

01

KNN的概念和原理

02

模型求解与评估

03

模型扩展与改进

04

模型实现与实战

05

总结

▶ 总结

➤ SVM（支持向量机）的概念与原理

- ✓ SVM 是一种基于间隔最大化的监督学习算法，通过寻找最优分割超平面来实现分类或回归。SVM 在小样本、高维度场景下表现优秀，但对参数和核函数选择敏感，计算开销较大。
- ✓ 它利用支持向量确定决策边界，并可通过核函数将数据映射到高维空间处理非线性问题。
- ✓ 模型假设：1) 数据近似线性可分（或通过核函数在高维空间可分）；2) 分类的鲁棒性来自最大间隔假设；3) 样本之间的关系可用内积/核函数度量。

➤ SVM（支持向量机）的求解与评估

- ✓ SVM 通过寻找最优分割超平面，将样本分到不同类别；对于非线性问题，可以借助核函数将数据映射到高维空间，再在高维空间中构建超平面；同时，SVM 也可扩展用于回归任务（SVR），通过设置间隔带来拟合连续输出。在分类任务中，常用的评估指标包括准确率、精确率、召回率和 F1 值等。

➤ SVM（支持向量机）的扩展与改进

- ✓ SVM 的扩展与改进主要有：1) 软间隔 SVM；2) 核 SVM；3) ν -SVM；4) SVR；5) 结构化 SVM；6) 一类 SVM；涉及分类、回归、复杂预测和异常检测等多领域。

SVM 通过寻找最大间隔的分割超平面实现分类或回归预测，可在分类与回归任务中分别用对应指标评估性能，同时可通过选择核函数、正则化参数和间隔松弛变量扩展以增强适用性。

感谢聆听



Personal Website: <https://www.miaopeng.info/>



Email: miaopeng@stu.scu.edu.cn



Github: <https://github.com/MMeowhite>



Youtube: <https://www.youtube.com/@pengmiao-bmm>