机器学习实验报告

支持向量机

朱天泽

（日期：2022 年 4 月 17 日）

摘要 在《机器学习》第 6 章中，我学习了支持向量机。此次实验，我实现了使用核函数和软间隔的支持向量机，在西瓜数据集 3.0α 上用多个核函数进行了训练，并对不同核函数的分类效果进行了比较。

关键词 SVM；核函数；分类

1 题目理解

题目要求：在西瓜数据集 3.0α 上分别用线性核和高斯核训练一个 SVM。

针对不同的核函数 κ，其底层的数学原理都类似，给一个 SVM 替换核函数也相对方便。因此本次实验计划不止用线性核和高斯核训练 SVM，而是使用多个核函数分别训练一个 SVM，并对不同核函数得到的 SVM 的分类效果做对比。

2 支持向量机原理阐述

2.1 间隔与支持向量

给定训练样本集 $D=\{(\boldsymbol{x}_1,y_1),(\boldsymbol{x}_2,y_2),\cdots,(\boldsymbol{x}_m,y_m)\}$，$y_i\in\{-1,1\}$，分类学习最基本的想法是基于训练集 D 在样本空间中找到一个划分超平面，将不同类别的样本分开。

在样本空间中，划分超平面可通过如下线性方程来描述：

$$\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0 \tag{1}$$

其中 $\boldsymbol{w}=(w_1,w_2,\cdots,w_d)^{\mathrm{T}}$ 为超平面的法向量，b 为位移项，决定了超平面与原点之间的距离。显然，划分超平面可被法向量 $\boldsymbol{w}$ 和位移 b 确定，下面我们将超平面记为 $(\boldsymbol{w},b)$。

对于任意一点 $\boldsymbol{x}_0=(x_1^0,x_2^0,\cdots,x_n^0)^{\mathrm{T}}$, 设其在超平面 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b=0$ 上的投影点为 $\boldsymbol{x}_1=(x_1^1,x_2^1,\cdots,x_n^1)^{\mathrm{T}}$，则 $\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_1+b=0$, 设其到超平面的距离为 r；由于向量 $\overrightarrow{\boldsymbol{x}_1\boldsymbol{x}_0}$ 与法向量 $\boldsymbol{w}$ 平行，因此

$$\left|\boldsymbol{w}\cdot\overrightarrow{\boldsymbol{x}_1\boldsymbol{x}_0}\right|=\left|\|\boldsymbol{w}\|\cdot\cos\pi\cdot\left\|\overrightarrow{\boldsymbol{x}_1\boldsymbol{x}_0}\right\|\right|=\|\boldsymbol{w}\|\cdot\left\|\overrightarrow{\boldsymbol{x}_1\boldsymbol{x}_0}\right\|=\|\boldsymbol{w}\|\cdot r \tag{2}$$

又有

$$\begin{aligned}\boldsymbol{w}\cdot\overrightarrow{\boldsymbol{x}_1\boldsymbol{x}_0}&=w_1\left(x_1^0-x_1^1\right)+w_2\left(x_2^0-x_2^1\right)+\cdots+w_n\left(x_n^0-x_n^1\right)\\&=w_1x_1^0+w_2x_2^0+\cdots+w_nx_n^0-\left(w_1x_1^1+w_2x_2^1+\cdots+w_nx_n^1\right)\\&=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_0-\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_1\\&=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_0+b\end{aligned} \tag{3}$$

故 $\left|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_0+b\right|=\|\boldsymbol{w}\|\cdot r$，因此对于任意点 $\boldsymbol{x}$，其到超平面 $(\boldsymbol{w},b)$ 的距离为

$$r=\frac{\left|\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b\right|}{\|\boldsymbol{w}\|} \tag{4}$$

将训练样本分开的划分超平面有很多，如图1。直观上看应该去找位于两类训练样本“正中间”的划分超平面，即距离超平面最近的正反例到超平面的距离“相当”，该划分超平面对训练样本局部扰动的容忍性最好。例如，由于训练集的局限性或噪声的因素，训练集外的样本可能比图1中的训练样本更接近两个类的分隔界，这将使许多划分超平面出现错误，而“正中间”的超平面受影响最小。换言之，这个划分超平面所产生的分类结果是最鲁棒的，对未见示例的泛化能力最强。

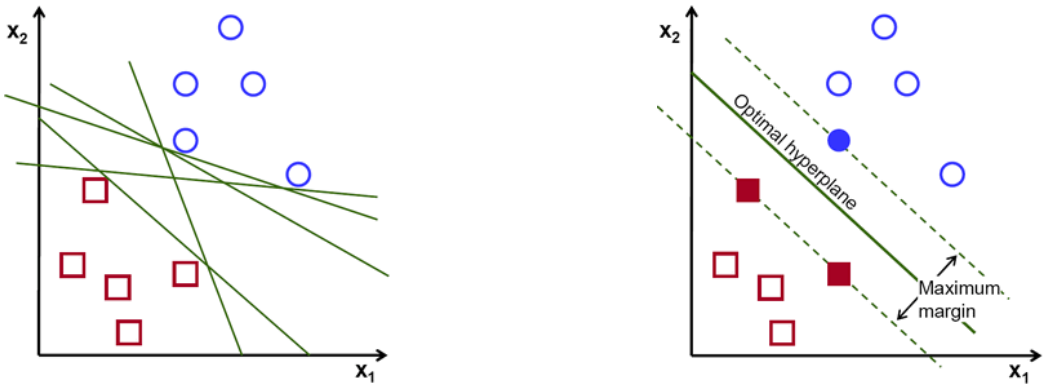

图 1: 有多个划分超平面将两类训练样本分开

对于给定的数据集 D 和超平面 $(\boldsymbol{w},b)$，定义任意一个样本点 $(\boldsymbol{x}_i,y_i)$，$y_i\in\{-1,1\}$，$i=1,2,\cdots,m$ 关于超平面的几何间隔为

$$\gamma_i=\frac{y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\right)}{\|\boldsymbol{w}\|} \tag{5}$$

当对样本 $(\boldsymbol{x}_i,y_i)$ 正确分类时，$\gamma_i>0$，几何间隔此时也等价于点到超平面的距离；而没有正确分类时，$\gamma_i<0$。

对于给定的数据集 D 和超平面 $(\boldsymbol{w},b)$，定义数据集 D 关于超平面的几何间隔为数据集 D 中所有样本点的几何间隔最小值

$$\gamma=\min_{i=1,2,\cdots,m}\gamma_i \tag{6}$$

给定线性可分数据集 D，支持向量机模型希望求得使得数据集 D 的几何间隔 γ 达到最大的超平面，然后利用 sign 函数实现分类功能

$$y=\operatorname{sign}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b\right)=\begin{cases}1, & \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b>0\\ -1, & \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}+b<0\end{cases} \tag{7}$$

易知：

- 当超平面没有正确划分样本时，必存在 $\gamma_i<0$，故 $\gamma<0$；
- 当超平面正确划分样本时，有 $\gamma\geqslant 0$，且越靠近样本间隔中央 γ 越大

因此，使得几何间隔 γ 最大的超平面就是位于两类训练样本“正中间”的划分超平面。

给定线性可分数据集 D，设 D 中几何间隔最小的样本为 $(\boldsymbol{x}_{\min},y_{\min})$，称其为“支持向量”，如图2所示。

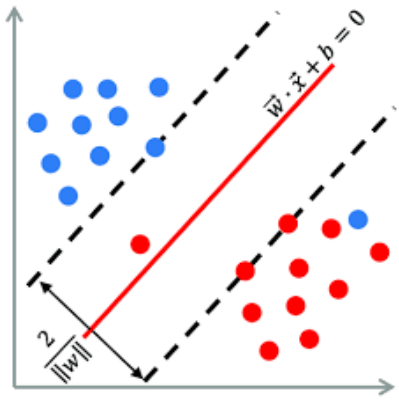

图 2: 支持向量与几何间隔

SVM 寻找最优超平面的过程可以转化为如下带约束条件的优化问题

$$\begin{aligned} &\max \quad \gamma \\ &\text{s.t.} \quad \gamma_i \geqslant \gamma, \quad i=1,2,\cdots,m \end{aligned} \tag{8}$$

即

$$\begin{aligned} &\max_{\boldsymbol{w},b} \frac{y_{\min}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_{\min}+b\right)}{\|\boldsymbol{w}\|} \\ &\text{s.t. } y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\right) \geqslant y_{\min}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_{\min}+b\right), \quad i=1,2,\cdots,m \end{aligned} \tag{9}$$

假设该问题的最优解为 $(\boldsymbol{w}^*,b^*)$，那么 $(k\boldsymbol{w}^*,kb^*), k\in\mathbb{R}^+$ 也是最优解，且超平面不变，因此需要对 $(\boldsymbol{w},b)$ 做一定限制才能使得上述优化问题有唯一解。那么不妨令 $y_{\min}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_{\min}+b\right)=1$，对于特定的 $(\boldsymbol{x}_{\min},y_{\min})$ 来说，能使得 $y_{\min}\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_{\min}+b\right)=1$ 的 k 有且仅有一个。因此，上述优化问题进一步转化为

$$\begin{aligned} &\max_{\boldsymbol{w},b} \quad \frac{1}{\|\boldsymbol{w}\|} \\ &\text{s.t.} \quad y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\right) \geqslant 1, \quad i=1,2,\cdots,m \end{aligned} \tag{10}$$

上述问题等价于

$$\begin{aligned} &\min_{w,b} \quad \frac{1}{2}\|\boldsymbol{w}\|^2 \\ &\text{s.t.} \quad 1-y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\right) \leqslant 0, \quad i=1,2,\cdots,m \end{aligned} \tag{11}$$

这就是 SVM 的基本型。

2.2 对偶问题

我们称式(11)为主问题，对式(11)使用拉格朗日乘子法，其拉格朗日函数为

$$L(\boldsymbol{w},b,\boldsymbol{\alpha})=\frac{1}{2}\|\boldsymbol{w}\|^2+\sum_{i=1}^{m}\alpha_i\left(1-y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\right)\right) \tag{12}$$

其中 $\boldsymbol{\alpha}=(\alpha_1,\alpha_2,\cdots,\alpha_m)^{\mathrm{T}}$。令 $L(\boldsymbol{w},b,\boldsymbol{\alpha})$ 对 $\boldsymbol{w}$ 和 b 的偏导数都为零，可得

$$\boldsymbol{w}=\sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i \tag{13}$$

$$\sum_{i=1}^{m}\alpha_i y_i = 0 \tag{14}$$

将式(13)代入(12)

$$\begin{aligned}\inf_{\boldsymbol{w},b} L(\boldsymbol{w},b,\boldsymbol{\alpha}) &= \frac{1}{2}\boldsymbol{w}^T\boldsymbol{w} + \sum_{i=1}^{m}\alpha_i - \sum_{i=1}^{m}\alpha_i y_i \boldsymbol{w}^T\boldsymbol{x}_i - \sum_{i=1}^{m}\alpha_i y_i b \\ &= \frac{1}{2}\boldsymbol{w}^T\sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i - \boldsymbol{w}^T\sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i + \sum_{i=1}^{m}\alpha_i - b\sum_{i=1}^{m}\alpha_i y_i \\ &= -\frac{1}{2}\boldsymbol{w}^T\sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i + \sum_{i=1}^{m}\alpha_i - b\sum_{i=1}^{m}\alpha_i y_i\end{aligned} \tag{15}$$

再将(14)代入(15)，于是有

$$\begin{aligned}\inf_{\boldsymbol{w},b} L(\boldsymbol{w},b,\boldsymbol{\alpha}) &= -\frac{1}{2}\boldsymbol{w}^T\sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i + \sum_{i=1}^{m}\alpha_i \\ &= -\frac{1}{2}\left(\sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i\right)^T\left(\sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i\right) + \sum_{i=1}^{m}\alpha_i \\ &= -\frac{1}{2}\sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i^T\sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i + \sum_{i=1}^{m}\alpha_i \\ &= \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^T\boldsymbol{x}_j\end{aligned} \tag{16}$$

最终得到式(11)的对偶问题

$$\begin{aligned}&\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m}\alpha_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j \\ &\text{s.t.} \quad \sum_{i=1}^{m}\alpha_i y_i = 0 \\ &\qquad \alpha_i \geqslant 0, \quad i = 1,2,\cdots,m\end{aligned} \tag{17}$$

注意到主问题，即式(11)，是凸优化问题，且除支持向量外，存在样本点满足 $1 - y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i + b) < 0$；因此强对偶性成立，对偶问题的解就是原问题的解，且必须满足 KKT 条件

$$\begin{cases}\alpha_i \geqslant 0 \\ y_i f(\boldsymbol{x}_i) - 1 \geqslant 0 \\ \alpha_i\left(y_i f(\boldsymbol{x}_i) - 1\right) = 0\end{cases} \tag{18}$$

解得 $\boldsymbol{\alpha}$ 后，求出 $\boldsymbol{w}$ 和 b 即可得到模型

$$\begin{aligned}f(\boldsymbol{x}) &= \boldsymbol{w}^{\mathrm{T}}\boldsymbol{x} + b \\ &= \sum_{i=1}^{m}\alpha_i y_i \boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x} + b\end{aligned} \tag{19}$$

2.3 核函数

在前面的讨论中，我们假设训练样本是线性可分的，即存在一个划分超平面能将训练样本正确分类。然而在现实任务中，原始样本空间内也许并不存在一个能正确划分两类样本的超平面。对这样的问题，可将样本从原始空间映射到-个更高维的特征空间，使得样本在这个特征空间内线性可分。

令 $\phi(\boldsymbol{x})$ 表示将 $\boldsymbol{x}$ 映射后的特征向量，于是，在特征空间中划分超平面所对应的模型可表示为

$$f(x)=w^{\mathrm{T}}\phi(x)+b \tag{20}$$

其中 $\boldsymbol{w}$ 和 b 是模型参数。类似式(11)，有

$$\begin{aligned}\min_{\boldsymbol{w},b}\quad & \tfrac{1}{2}\|\boldsymbol{w}\|^2\\ \text{s.t.}\quad & 1-y_i\left(\boldsymbol{w}^{\mathrm{T}}\phi\left(\boldsymbol{x}_i\right)+b\right)\leqslant 0,\quad i=1,2,\cdots,m\end{aligned} \tag{21}$$

其对偶问题是

$$\begin{aligned}\max_{\boldsymbol{\alpha}}\ & \sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j\phi(\boldsymbol{x}_i)^{\mathrm{T}}\phi(\boldsymbol{x}_j)\\ \text{s.t.}\ & \sum_{i=1}^{m}\alpha_i y_i=0\\ & \alpha_i\geqslant 0,\quad i=1,2,\cdots,m\end{aligned} \tag{22}$$

求解式(22)涉及到计算 $\phi\left(\boldsymbol{x}_i\right)^{\mathrm{T}}\phi\left(\boldsymbol{x}_j\right)$，这是样本 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$ 映射到特征空间之后的内积。由于特征空间维数可能很高，甚至可能是无穷维，因此直接计算 $\phi\left(\boldsymbol{x}_i\right)^{\mathrm{T}}\phi\left(\boldsymbol{x}_j\right)$ 通常是困难的。为了避免这样的障碍，可以使用核技巧，定义核函数

$$\kappa\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)=\left\langle\phi\left(\boldsymbol{x}_i\right),\phi\left(\boldsymbol{x}_j\right)\right\rangle=\phi\left(\boldsymbol{x}_i\right)^{\mathrm{T}}\phi\left(\boldsymbol{x}_j\right) \tag{23}$$

即 $\boldsymbol{x}_i$ 与 $\boldsymbol{x}_j$ 在特征空间中的内积等于它们在原始样本空间中通过函数 $\kappa(\cdot,\cdot)$ 计算的结果。有了核函数，我们就不必直接去计算高维甚至无穷维特征空间中的内积，于是式(22)可重写为

$$\begin{aligned}\max_{\boldsymbol{\alpha}}\ & \sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j\kappa\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right)\\ \text{s.t.}\ & \sum_{i=1}^{m}\alpha_i y_i=0,\\ & \alpha_i\geqslant 0,\quad i=1,2,\ldots,m\end{aligned} \tag{24}$$

求解后即可得到

$$\begin{aligned}f(\boldsymbol{x})&=\boldsymbol{w}^{\mathrm{T}}\phi(\boldsymbol{x})+b\\ &=\sum_{i=1}^{m}\alpha_i y_i\phi\left(\boldsymbol{x}_i\right)^{\mathrm{T}}\phi(\boldsymbol{x})+b\\ &=\sum_{i=1}^{m}\alpha_i y_i\kappa\left(\boldsymbol{x},\boldsymbol{x}_i\right)+b\end{aligned} \tag{25}$$

定理 令 χ 为输入空间，$\kappa(\cdot,\cdot)$ 是定义在 $\chi\times\chi$ 上的对称核函数，则 κ 是核函数当且仅当对于任意数据 $D=\left\{\boldsymbol{x}_i\right\}_{i=1}^{m}$，“核矩阵” $\mathbf{K}$ 总是半正定的：

$$\mathbf{K}=\begin{bmatrix} \kappa(\boldsymbol{x}_1,\boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_1,\boldsymbol{x}_j) & \cdots & \kappa(\boldsymbol{x}_1,\boldsymbol{x}_m) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \kappa(\boldsymbol{x}_i,\boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_i,\boldsymbol{x}_j) & \cdots & \kappa(\boldsymbol{x}_i,\boldsymbol{x}_m) \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ \kappa(\boldsymbol{x}_m,\boldsymbol{x}_1) & \cdots & \kappa(\boldsymbol{x}_m,\boldsymbol{x}_j) & \cdots & \kappa(\boldsymbol{x}_m,\boldsymbol{x}_m) \end{bmatrix}$$

在现实中，我们通常不知道 $\phi(\cdot)$ 的具体形式，但是根据上述定理，我们不必知道 $\phi(\cdot)$ 的具体形式，就能得到核函数；事实上，对于一个半正定核矩阵，总能找到一个与之对应的映射 ϕ。我们希望样本在特征空间内线性可分，因此特征空间的好坏对支持向量机的性能至关重要。需注意的是，在不知道特征映射的形式时，我们并不知道什么样的核函数是合适的，而核函数也仅是隐式地定义了这个特征空间。于是，“核函数选择”成为 SVM 的最大变数。若核函数选择不合适，则意味着将样本映射到了一个不合适的特征空间，很可能导致性能不佳。

常见的核函数如表1所示。

名称	表达式	参数说明
线性核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j$	
多项式核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\left(\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j\right)^d$	$d\geqslant 1$ 为多项式的次数
高斯核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=e^{-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}{2\sigma^2}}$	$\sigma>0$ 为高斯核的带宽
RBF 核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=e^{-\gamma\|\boldsymbol{x}_i-\boldsymbol{x}_j\|^2}$	$0<\gamma<1$
拉普拉斯核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=e^{-\frac{\|\boldsymbol{x}_i-\boldsymbol{x}_j\|}{\sigma}}$	$\sigma>0$
Sigmoid 核	$\kappa(\boldsymbol{x}_i,\boldsymbol{x}_j)=\tanh\left(\beta\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j+\theta\right)$	tanh 为双曲正切函数，$\beta>0$，$\theta<0$

表 1: 常用核函数

2.4 软间隔

在前面的讨论中，我们一直假定训练样本在样本空间或特征空间中是线性可分的，即存在一个超平面能将不同类的样本完全划分开。然而，在现实任务中往往很难确定合适的核函数使得训练样本在特征空间中线性可分。

缓解该问题的一个办法是允许支持向量机在一些样本上出错。为此，要引入“软间隔”的概念，如图3所示。

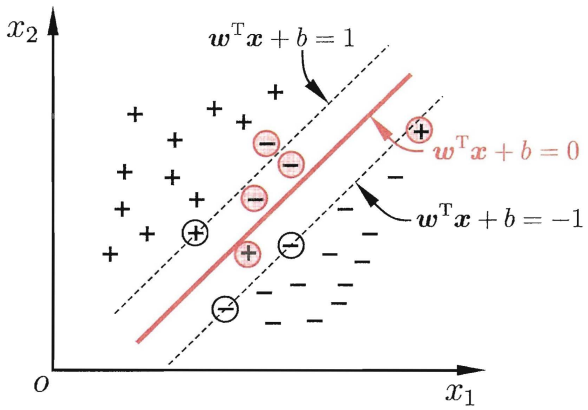

图 3: 软间隔示意图，红色圈出了一些不满足约束的样本。

以线性 SVM 为例，其形式要求所有样本都划分正确，即 $y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\right)\geqslant 1$，软间隔允许某些样本不满足这一约束。

当然，在最大化间隔的同时，不满足约束的样本应尽可能少。于是，优化目标可写为

$$\min_{\boldsymbol{w},b}\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{m}\ell_{0/1}\left(y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\right)-1\right) \tag{26}$$

其中 $C>0$ 是一个常数，$\ell_{0/1}$ 是 “0/1 损失函数”。$\ell_{0/1}$ 非凸、非连续，我们不妨使用 hinge 损失代替：

$$\ell_{\text{hinge}}(z)=\max(0,1-z) \tag{27}$$

则式(26)可写成

$$\min_{\boldsymbol{w},b}\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{m}\max\left(0,1-y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\right)\right) \tag{28}$$

引入 “松弛变量” $\xi_i\geqslant 0$，可将式(28)重写为

$$\begin{aligned}\min_{w,b,\xi_i}\quad & \frac{1}{2}\|w\|^2+C\sum_{i=1}^{m}\xi_i\\ \text{s.t.}\quad & y_i\left(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b\right)\geqslant 1-\xi_i\\ & \xi_i\geqslant 0,\quad i=1,2,\cdots,m\end{aligned} \tag{29}$$

这就是常用的软间隔 SVM。

式(29)与式(11)类似，仍是一个二次规划问题；于是，类似式(12)，通过拉格朗日乘子法可得到式(29)的拉格朗日函数

$$L(\boldsymbol{w},b,\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\mu})=\frac{1}{2}\|\boldsymbol{w}\|^2+C\sum_{i=1}^{m}\xi_i+\sum_{i=1}^{m}\alpha_i[1-\xi_i-y_i(\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}_i+b)]-\sum_{i=1}^{m}\mu_i\xi_i \tag{30}$$

其中 $\alpha_i\geqslant 0$，$\mu_i\geqslant 0$ 是拉格朗日因子。

令 $L(\boldsymbol{w},b,\boldsymbol{\alpha},\boldsymbol{\xi},\boldsymbol{\mu})$ 对 $\boldsymbol{w},b,\xi_i$ 的偏导为零可得

$$\boldsymbol{w}=\sum_{i=1}^{m}\alpha_i y_i\boldsymbol{x}_i \tag{31}$$

$$\sum_{i=1}^{m}\alpha_i y_i=0 \tag{32}$$

$$C=\alpha_i+\mu_i \tag{33}$$

将式(31)-(33)代入式(30)即可得到式(29)的对偶问题

$$\begin{aligned}\max_{\boldsymbol{\alpha}}\ & \sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j\boldsymbol{x}_i^{\mathrm{T}}\boldsymbol{x}_j\\ \text{s.t.}\ & \sum_{i=1}^{m}\alpha_i y_i=0,\\ & 0\leqslant\alpha_i\leqslant C,\quad i=1,2,\ldots,m\end{aligned} \tag{34}$$

在线性的软间隔 SVM 的基础上，我们引入核函数 κ，对偶问题为

$$\begin{aligned}
&\max_{\boldsymbol{\alpha}} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) \\
&\text{s.t.} \sum_{i=1}^{m} \alpha_i y_i = 0, \\
&\qquad 0 \leqslant \alpha_i \leqslant C, \quad i = 1, 2, \ldots, m
\end{aligned} \tag{35}$$

类似式(18)，软间隔 SVM 的 KKT 条件为

$$\begin{cases}
\alpha_i \geqslant 0, \quad \mu_i \geqslant 0 \\
y_i f\left(\boldsymbol{x}_i\right) - 1 + \xi_i \geqslant 0 \\
\alpha_i \left(y_i f\left(\boldsymbol{x}_i\right) - 1 + \xi_i\right) = 0 \\
\xi_i \geqslant 0, \quad \mu_i \xi_i = 0
\end{cases} \tag{36}$$

3 算法设计思路

3.1 对偶问题转化

将式(35)表示的对偶问题转化为凸优化问题

$$\begin{aligned}
&\min_{\boldsymbol{\alpha}} \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \kappa\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \sum_{i=1}^{m} \alpha_i \\
&\text{s.t.} \sum_{i=1}^{m} \alpha_i y_i = 0, \\
&\qquad 0 \leqslant \alpha_i \leqslant C, \quad i = 1, 2, \ldots, m
\end{aligned} \tag{37}$$

根据式(36)表示的 KKT 条件和式(33)表示的约束条件

$$\begin{aligned}
\alpha_i = 0 &\Rightarrow y_i f(\boldsymbol{x}_i) - 1 \geqslant 0 \\
0 < \alpha_i < C &\Rightarrow y_i f(\boldsymbol{x}_i) - 1 = 0 \\
\alpha_i = C &\Rightarrow y_i f(\boldsymbol{x}_i) - 1 \leqslant 0
\end{aligned} \tag{38}$$

考虑到计算机运算时的误差，我们引入计算误差项 $\varepsilon \geqslant 0$，并记 $E_i = f(\boldsymbol{x}_i) - y_i$，得到

$$\begin{aligned}
\alpha_i = 0 &\Rightarrow y_i E_i \geqslant -\varepsilon \\
0 < \alpha_i < C &\Rightarrow |y_i E_i| < \varepsilon \\
\alpha_i = C &\Rightarrow y_i E_i \leqslant \varepsilon
\end{aligned} \tag{39}$$

进一步归纳，得到 α_i 的约束条件

$$\begin{aligned}
\alpha_i > 0 &\Rightarrow y_i E_i \leqslant \varepsilon \\
\alpha_i < C &\Rightarrow y_i E_i \geqslant -\varepsilon
\end{aligned} \tag{40}$$

3.2 SMO 算法

3.2.1 基本思想

SMO 的基本思路是先固定 α_i 之外的所有参数，然后求 α_i 上的极值。由于存在约束 $\sum_{i=1}^{m} \alpha_i y_i = 0$，若固定 α_i 之外的其它变量，则 α_i 可由其它变量导出。于是，SMO 算法每次选择两个变量 α_i 和 α_j，并固定其

它参数。这样，在参数初始化后，SMO 不断执行如下步骤直至收敛：

- 选取一对需要更新的 α_i 和 α_j；
- 固定 α_i 和 α_j 以外的参数，求解式(37)获得更新后的 α_i 和 α_j。

3.2.2 单变量无约束迭代

下面来推导更新 α_i 和 α_j 的迭代式。由于 $\boldsymbol{\alpha}$ 中所有维度都是等价的，我们不妨简化问题，选取 α_1 和 α_2，推导更新 α_i 和 α_j 的迭代式。

令

$$\Psi(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m}\alpha_i\alpha_j y_i y_j \kappa\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) - \sum_{i=1}^{m}\alpha_i \tag{41}$$

我们选取 α_1 和 α_2，固定 $\boldsymbol{\alpha}$ 中其它维度，省略 $\Psi(\boldsymbol{\alpha})$ 中的常数项，得到

$$\begin{aligned}\Psi(\alpha_1, \alpha_2) =& \frac{1}{2}\alpha_1^2\kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) + \frac{1}{2}\alpha_2^2\kappa(\boldsymbol{x}_2, \boldsymbol{x}_2) \\ &+ y_1y_2\alpha_1\alpha_2\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) + \nu_1 y_1\alpha_1 + \nu_2 y_2\alpha_2 - \alpha_1 - \alpha_2\end{aligned} \tag{42}$$

其中

$$\begin{aligned}\nu_i &= \sum_{j=3}^{m}\alpha_j y_j \kappa(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ &= f(\boldsymbol{x}_i) - y_1\alpha_1\kappa(\boldsymbol{x}_i, \boldsymbol{x}_1) - y_2\alpha_2\kappa(\boldsymbol{x}_i, \boldsymbol{x}_2) - b, \quad i = 1, 2\end{aligned} \tag{43}$$

根据约束条件式(32)，有

$$\alpha_1 y_1 + \alpha_2 y_2 = -\sum_{i=3}^{m}\alpha_i y_i = \zeta \tag{44}$$

其中 ζ 为常数。针对 α_1 和 α_2，假设更新之前为 α_1^{old} 和 α_2^{old}，更新之后为 α_1^{new} 和 α_2^{new}；由于 $\boldsymbol{\alpha}$ 中其余维度固定，因此必须满足

$$\alpha_1^{\text{old}} y_i + \alpha_2^{\text{old}} = \alpha_2^{\text{new}} y_1 + \alpha_2^{\text{new}} y_2 = \zeta \tag{45}$$

两个变量不好同时求解，可以先求 α_2^{new}，然后根据式(45)表示 α_1^{new}。

由式(44)得

$$\alpha_1 = (\zeta - \alpha_2 y_2) y_1 \tag{46}$$

将式(46)代入式(41)，得

$$\begin{aligned}\Psi(\alpha_2) =& \frac{1}{2}(\zeta - \alpha_2 y_2)^2\kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) + \frac{1}{2}\alpha_2^2\kappa(\boldsymbol{x}_2, \boldsymbol{x}_2) + \alpha_2 y_2(\zeta - \alpha_2 y_2)\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) \\ &+ \nu_1(\zeta - \alpha_2 y_2) + \nu_2 y_2\alpha_2 - (\zeta - \alpha_2 y_2) y_1 - \alpha_2\end{aligned} \tag{47}$$

令 $\frac{\mathrm{d}\Psi(\alpha_2)}{\mathrm{d}\alpha_2} = 0$，得

$$\alpha_2 = \frac{y_2[y_2 - y_1 + v_1 - v_2 + \zeta(\kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) - \kappa(\boldsymbol{x}_1, \boldsymbol{x}_2))]}{\kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) + \kappa(\boldsymbol{x}_2, \boldsymbol{x}_2) - 2\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2)} \tag{48}$$

将式(43)代入式(48)，再结合式(44)，得未修剪的 α_2^{new} 为

$$\alpha_2^{\text{unclipped}} = \alpha_2^{\text{old}} + \frac{y_2(E_1 - E_2)}{\kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) + \kappa(\boldsymbol{x}_2, \boldsymbol{x}_2) - 2\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2)} \tag{49}$$

3.2.3 修剪 $\alpha_2^{\text{unclipped}}$

α_i 必须满足不等式约束

$$0 \leqslant \alpha_i \leqslant C \tag{50}$$

式(49)未考虑到不等式约束，是未修剪的值。下面考虑不等式约束，由式(45)，有

$$\alpha_1^{\text{new}} = \alpha_1^{\text{old}} + y_1 y_2 (\alpha_2^{\text{old}} - \alpha_2^{\text{new}}) \tag{51}$$

当 $y_1 = y_2$，由(51)有

$$\alpha_1^{\text{new}} = \alpha_1^{\text{old}} + \alpha_2^{\text{old}} - \alpha_2^{\text{new}} \tag{52}$$

故

$$0 \leqslant \alpha_1^{\text{new}} \leqslant C \Leftrightarrow \alpha_1^{\text{old}} + \alpha_2^{\text{old}} - C \leqslant \alpha_2^{\text{new}} \leqslant \alpha_1^{\text{old}} + \alpha_2^{\text{old}} \tag{53}$$

又 $0 \leqslant \alpha_i \leqslant C$，所以

$$\max\left(0, \alpha_1^{\text{old}} + \alpha_2^{\text{old}} - C\right) \leqslant \alpha_2^{\text{new}} \leqslant \min(C, \alpha_1^{\text{old}} + \alpha_2^{\text{old}}) \tag{54}$$

同理，当 $y_1 \neq y_2$ 时

$$\max\left(0, \alpha_2^{\text{old}} - \alpha_1^{\text{old}}\right) \leqslant \alpha_2^{\text{new}} \leqslant \min\left(C, \alpha_2^{\text{old}} - \alpha_1^{\text{old}} + C\right) \tag{55}$$

设 $[L, H]$ 为考虑不等式约束后 α_2 的可行域，整理得到

$$[L, H] = \begin{cases} \left[\max\left(0, \alpha_1^{\text{old}} + \alpha_2^{\text{old}} - C\right), \min(C, \alpha_1^{\text{old}} + \alpha_2^{\text{old}})\right], & y_1 = y_2 \\ \left[\max\left(0, \alpha_2^{\text{old}} - \alpha_1^{\text{old}}\right), \min\left(C, \alpha_2^{\text{old}} - \alpha_1^{\text{old}} + C\right)\right], & y_1 \neq y_2 \end{cases} \tag{56}$$

对 $\alpha_2^{\text{unclipped}}$ 的修剪方法 clip 为

$$\alpha_2^{\text{new}} = \text{clip}\left(\alpha_2^{\text{unclipped}}\right) = \begin{cases} L, & \alpha_2^{\text{unclipped}} < L \\ \alpha_2^{\text{unclipped}}, & L \leqslant \alpha_2^{\text{unclipped}} \leqslant H \\ H, & \alpha_2^{\text{unclipped}} > H \end{cases} \tag{57}$$

3.2.4 计算 α_1^{new}

由于式(56)是从式(51) 开始，根据不等式约束 $0 \leqslant \alpha_i \leqslant C$ 得到，故由式(51)得到的 α_1^{new} 必然满足不等式约束，α_1 的迭代式就为式(51)。

3.2.5 计算 b^{new}

若 $0 < \alpha_1^{\text{new}} < C$，由式(25)，有

$$\begin{aligned} b^{\text{new}} = b_1^{\text{new}} &= y_1 - \sum_{i=3}^{m} \alpha_i y_i \kappa(\boldsymbol{x}_1, \boldsymbol{x}_i) - \alpha_1^{\text{new}} y_1 \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) - \alpha_2^{\text{new}} y_2 \\ &= (\alpha_1^{\text{old}} - \alpha_1^{\text{new}}) y_1 \kappa(\boldsymbol{x}_1, \boldsymbol{x}_1) + (\alpha_2^{\text{old}} - \alpha_2^{\text{new}}) y_1 \kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) - E_1^{\text{old}} + b^{\text{old}} \end{aligned} \tag{58}$$

同理，若 $0 < \alpha_2^{\text{new}} < C$

$$b^{\text{new}} = b_2^{\text{new}} = (\alpha_1^{\text{old}} - \alpha_1^{\text{new}}) y_1 \kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) + (\alpha_2^{\text{old}} - \alpha_2^{\text{new}}) y_1 \kappa(\boldsymbol{x}_2, \boldsymbol{x}_2) - E_2^{\text{old}} + b^{\text{old}} \tag{59}$$

若同时满足 $0 < \alpha_1^{\text{new}} < C$ 和 $0 < \alpha_2^{\text{new}} < C$，则显然有 $b^{\text{new}} = b_1^{\text{new}} = b_2^{\text{new}}$；若 $\alpha_1^{\text{new}}, \alpha_2^{\text{new}} \in \{0, C\}$ 且 $L \neq H$，则 $b_1^{\text{new}}, b_2^{\text{new}}$ 之间的值都能使两个样本满足 KKT 条件，取 $b^{\text{new}} = \frac{b_1^{\text{new}} + b_2^{\text{new}}}{2}$；若 $\alpha_1^{\text{new}}, \alpha_2^{\text{new}} \in \{0, C\}$ 且 $L = H$，则不更新 b。

3.2.6 启发式选择

根据前面的推导，每一步选择两个维度 α_i 和 α_j 进行优化

$$\begin{aligned} \alpha_i^{\text{new}} &= \alpha_i^{\text{old}} + y_1 y_2 (\alpha_j^{\text{old}} - \alpha_j^{\text{new}}) \\ \alpha_j^{\text{new}} &= \text{clip}\left(\alpha_j^{\text{old}} + \frac{y_j (E_i - E_j)}{\kappa(\boldsymbol{x}_i, \boldsymbol{x}_i) + \kappa(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)} \right) \end{aligned} \tag{60}$$

注意到只需选取的 α_i 和 α_j 中有一个不满足式(40)的约束，目标函数就会在迭代后减小。所以 SMO 算法先选取违背式(40)的维度，得到 α_i。选择 α_j 的启发规则是选择样本 $\kappa(\boldsymbol{x}_j, y_j)$ 来使得优化的步长最大化，也就是使式(57)中 $\frac{y_j(E_j - E_j)}{\kappa(\boldsymbol{x}_i, \boldsymbol{x}_i) + \kappa(\boldsymbol{x}_j, \boldsymbol{x}_j) - 2\kappa(\boldsymbol{x}_i, \boldsymbol{x}_j)}$ 最大化。由于计算核函数比较耗时间，所以 SMO 使用简单的近似，用 $|E_i - E_j|$ 来近似步长；也就是说，选择样本 $(\boldsymbol{x}_j, y_j)$ 使 $|E_i - E_j|$ 最大，α_j 的更新步长就能尽可能大。

3.3 算法流程总结

经过上面的讨论，算法流程如图4所示。

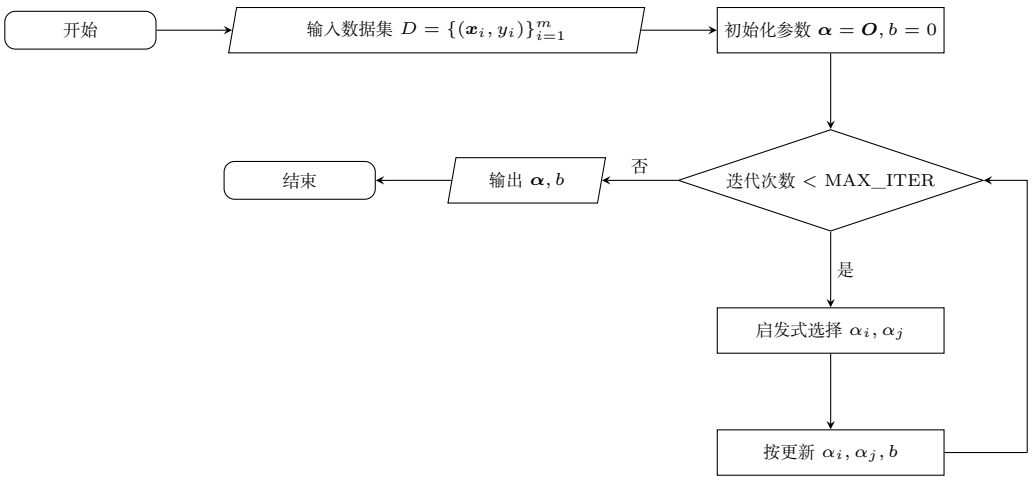

图 4: 算法流程

4 核心代码分析

4.1 数据定义

代码中定义最大迭代次数，用矩阵 $\mathbf{X}$ 表示西瓜数据集 3.0α 的数据部分，每个样本有密度、含糖率两个维度，用向量 y 表示每个西瓜的真实标记。初始化 $\boldsymbol{\alpha} = \boldsymbol{O}$，$b = 0$。定义常数 $C = 1$，$\varepsilon = 0.00001$。

```
max_iter = 50
m = 17
X = np.array([[0.697, 0.460], [0.774, 0.376], [0.634, 0.264], [0.608, 0.318], [0.556, 0.215], [0.403, 0.237],
     [0.481, 0.149], [0.437, 0.211], [0.666, 0.091], [0.243, 0.267], [0.245, 0.057], [0.343, 0.099], [0.639,
    0.161], [0.657, 0.198], [0.360, 0.370], [0.593, 0.042], [0.719, 0.103]])
# 真实标记
y = np.array([1, 1, 1, 1, 1, 1, 1, 1, -1, -1, -1, -1, -1, -1, -1, -1, -1])
# 初始化
alpha = np.zeros(m)
b = 0
# 常数取值
epsilon = 0.00001
C = 1
```

4.2 定义核函数 κ

kappa 函数提供六种核函数，包括线性核、高斯核、多项式核、拉普拉斯核、Sigmoid 核、RBF 核，函数默认使用高斯核。

```
def kappa(x1, x2, func='gauss'):
    if func == 'gauss':
        sigma = 0.11855
        return np.exp(-(np.linalg.norm(x1 - x2) ** 2) / (2 * sigma ** 2))
    elif func == 'linear':
        return np.dot(x1, x2)
    elif func == 'polynomial':
        d = 5
        return np.dot(x1, x2) ** d
    elif func == 'laplace':
        sigma = 0.11855
        return np.exp(-np.linalg.norm(x1 - x2) / sigma)
    elif func == 'sigmoid':
        beta = 0.8
        theta = -0.2
        return np.tanh(beta * np.dot(x1, x2) + theta)
    elif func == 'RBF':
        gamma = 0.2
        return np.exp(-gamma * np.linalg.norm(x1 - x2) ** 2)
    else:
        raise Exception('核函数不存在')
```

4.3 修剪操作

修剪操作，实际上就是让超过可行域的取值 a 落到可行域的边界上。

```
def clip(a, _L, _H):
    if a < _L:
        return _L
    elif a > _H:
        return _H
    else:
        return a
```

4.4 SMO 过程

按照图4的流程，每一次迭代都按照启发式规则选择 α_i 和 α_j：首先选择不满足式(40)的 α_i，若都满足约束则随机选择；接着再选择 $\alpha_{\substack{\arg\max \\ 1\leqslant j\leqslant m, j\neq i}} |E_i - E_j|$，更新 α_i、α_j、b。

```
for _ in  range(max_iter):
    index1 = index2 = -1
    # 寻找不满足约束的alpha_i
    for i in  range(m):
        Ei = f(X[i]) - y[i]
        if alpha[i] < C and y[i] * Ei < -epsilon:
            index1 = i
            break
        if alpha[i] > 0 and y[i] * Ei > epsilon:
            index1 = i
            break

    # 都满足约束则随机选择
    if index1 == -1:
        index1 = np.random.randint(0, m)

    E1 = f(X[index1]) - y[index1]

    # 寻找使更新步长最大的alpha_j
    E2 = E1
    for i in  range(m):
        if i == index1:
            continue
        Ei = f(X[i]) - y[i]
        if np. abs(Ei - E1) > np. abs(E2 - E1):
            E2 = Ei
            index2 = i

    alpha1_old = alpha[index1]
    alpha2_old = alpha[index2]
    # 更新
    alpha2_new = alpha2_old + y[index2] * (E1 - E2) / (kappa(X[index1], X[index1]) + kappa(X[index2], X[
        index2]) - 2 * kappa(X[index1], X[index2]))
```

```
    # 确定可行域
    if y[index1] == y[index2]:
        L =  max(0, alpha1_old + alpha2_old - C)
        H =  min(C, alpha2_old + alpha1_old)
    else:
        L =  max(0, alpha2_old - alpha1_old)
        H =  min(C, alpha2_old - alpha1_old + C)

    # 修剪
    alpha2_new = clip(alpha2_new, L, H)

    alpha1_new = alpha1_old + y[index1] * y[index2] * (alpha2_old - alpha2_new)
    alpha[index1] = alpha1_new
    alpha[index2] = alpha2_new

    # 更新b
    b1 = -E1 - y[index1] * kappa(X[index1], X[index1]) * (alpha1_new - alpha1_old) - y[index2] * kappa(X[
        index1], X[index2]) * (alpha2_new - alpha2_old) + b
    b2 = -E2 - y[index1] * kappa(X[index1], X[index2]) * (alpha1_new - alpha1_old) - y[index2] * kappa(X[
        index2], X[index2]) * (alpha2_new - alpha2_old) + b
    if 0 < alpha1_new < C:
        b = b1
    elif 0 < alpha2_new < C:
        b = b2
    elif L != H:
        b = (b1 + b2) / 2
```

4.5 结果预测

利用式(25)得到 $f(\boldsymbol{x}_i)$。

```
def f(_x):
    ans = b
    for i in  range(m):
        ans += alpha[i] * y[i] * kappa(_x, X[i])
    return ans
```

5 实验结果与分析

此次实验中用到了表1中所列举的核函数，使用不同核函数，对西瓜数据集 3.0α 的分类效果如图5-10所示。

基于在西瓜数据集 3.0α 上的分类效果，我们可以初步得到如下结论：

- 线性核、多项式核、RBF 核、Sigmoid 核的分类效果类似，都没有将好瓜与坏瓜完全区分开来；
- 高斯核、拉普拉斯核的分类效果相似，将好瓜与坏瓜完全区分开来。

如表1所示，高斯核、拉普拉斯核、RBF 核的数学形式相似，因此高斯核、拉普拉斯核的分类效果相似，RBF 核分类效果差的原因在于 γ 取值的限制。在 scikit-learn 提供的 SVM 包中，取消了 $0 < \gamma < 1$ 这一限制，仅提供 RBF 核和拉普拉斯核核，并默认 $\gamma = \frac{1}{\text{n\_features} \cdot \text{X.var()}}$，也能取得较好的分类效果。

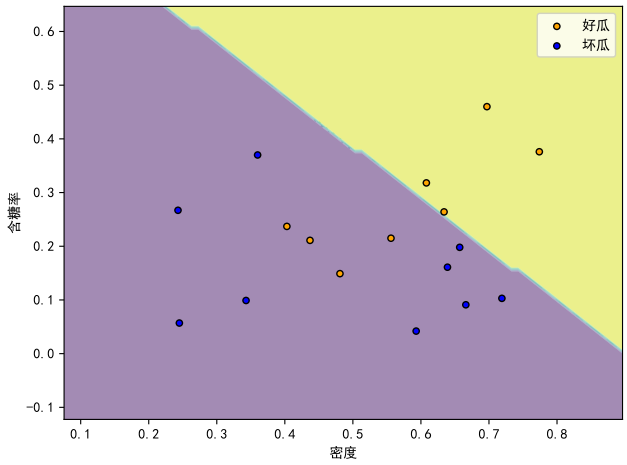

图 5: 线性核

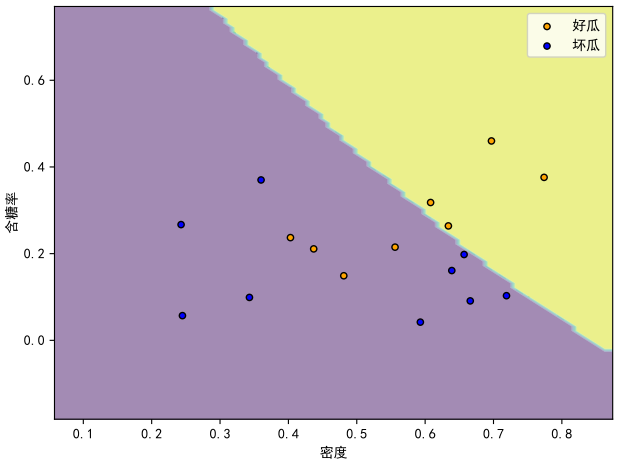

图 6: 多项式核

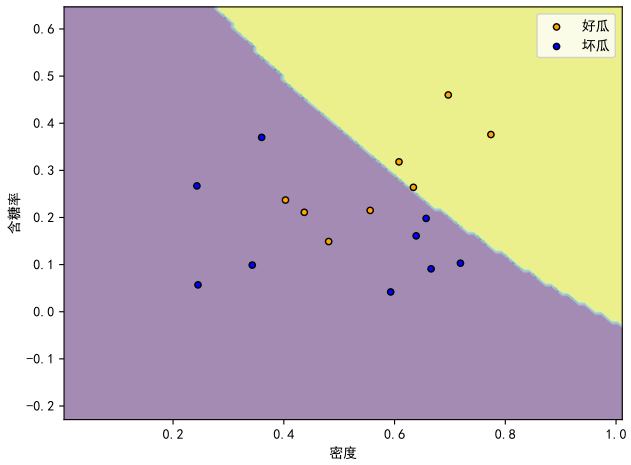

图 7: RBF 核

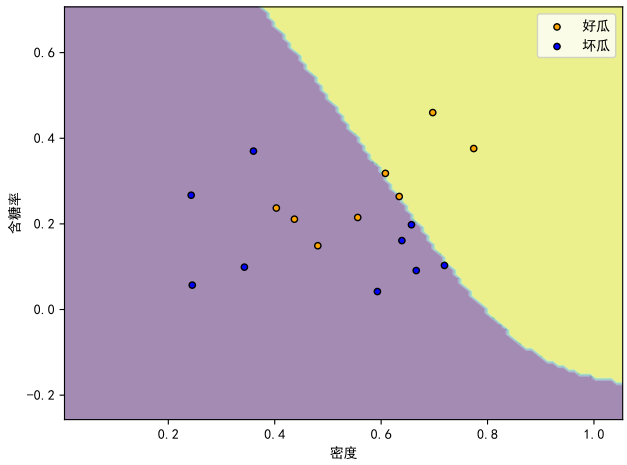

图 8: Sigmoid 核

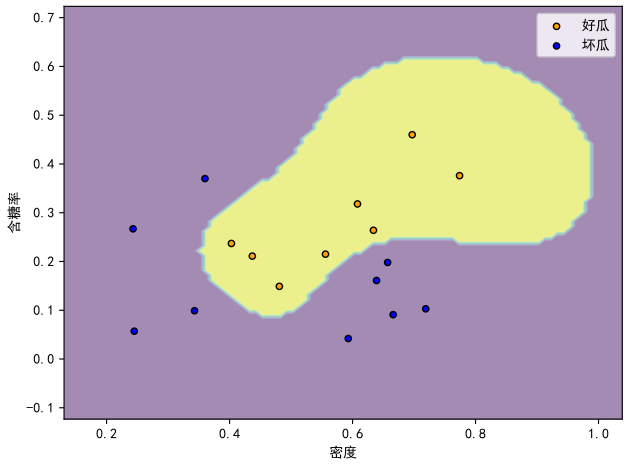

图 9: 高斯核

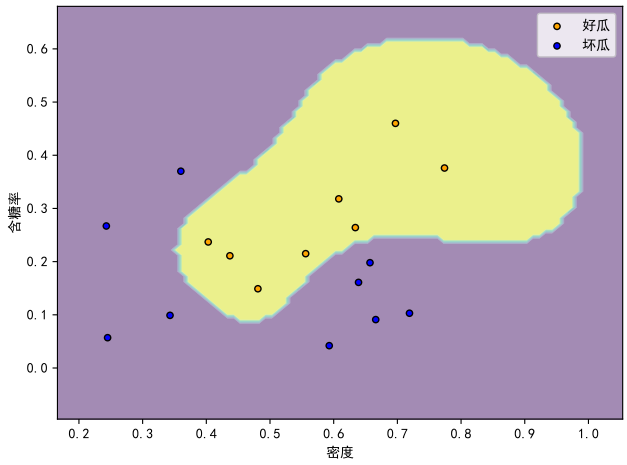

图 10: 拉普拉斯核

线性核、多项式核、Sigmoid 核表示的函数在二维空间中都是不封闭的曲线，可以看到二维空间中的分类边界就类似这些函数的曲线，仅从西瓜数据集 3.0α 来看，并不适合作为 SVM 的核函数。

线性核计算快、可计算得到明确的边界，但是只能解决线性可分问题。多项式核、Sigmoid 核可解决非线性问题，多项式核对于大数量级的幂数计算较慢，而 Sigmoid 核更常用于训练多层感知机神经网络。高斯

核、RBF 核、拉普拉斯核能将样本映射到无穷维空间，决策边界更为多样，分类效果好，但是难以计算清晰的分类边界。

6 学习收获

在这次实验过程中，我从头实现了 SVM，并对多个不同核函数的分类效果进行了比较。基于西瓜数据集 3.0α 上的分类效果，得出了不同核函数的优缺点和适用场景。

关于 SMO 方法、KKT 条件等知识，书上提及甚少，我因此查阅了大量资料，自己推导了 SMO 算法的诸多迭代式。实际上，在 SMO 过程中，我们还可以动态更新 E_i，而不必每一步迭代都重新计算 E_i，将来需要在代码中完善这一过程，进而提高训练 SVM 的速度。

此次实验还留下疑问，即 Sigmoid 核与多层感知机神经网络之间的联系。

7 参考资料

- 《机器学习》6.1,6.2,6.3,6.4；周志华；清华大学出版社
- 支持向量机（scikit-learn 中文社区）
- sklearn.svm.SVC（scikit-learn 中文社区）
- Why does the RBF (radial basis function) kernel map into infinite dimensional space, mentioned many times in machine learning lectures?
- 【吃瓜教程】《机器学习公式详解》（南瓜书）与西瓜书公式推导直播合集 | 第六章-支持向量机
- 【机器学习详解】SMO 算法剖析
- SEVEN MOST POPULAR SVM KERNELS
- 支持向量机（SVM）| SMO 方法篇
- 机器学习算法实践-SVM 中的 SMO 算法
- 支持向量机原理详解 (五): KKT 条件 (Part II)
- 支持向量机原理详解 (六): 序列最小最优化 (SMO) 算法 (Part I)