支持向量机（SVM）算法详解

Lanrudan

2025 年 7 月 20 日

什么是支持向量机（SVM）？

支持向量机 (Support Vector Machine, SVM) 是一种强大的**监督学习**算法，主要用于**分类**和**回归**任务，但它在分类问题上表现尤为出色。

SVM 的核心思想是找到一个**最优的超平面（hyperplane）**，能够将不同类别的数据点最大限度地分开。这个超平面不仅要能分开数据，还要使不同类别中离它最近的数据点（称为**支持向量**）到超平面的距离最大化。这个距离被称为**间隔（margin）**。最大化间隔是 SVM 的独特之处和强大之处。

为什么最大化间隔很重要？

- **泛化能力强：** 间隔越大，模型对新数据的**泛化能力**越强。**泛化能力**是指一个机器学习模型在**没有见过的新数据**上的表现能力。间隔大意味着超平面离两类数据都尽可能远，这种" 安全距离" 使得模型在处理新的、略有偏差的数据时，也能保持分类的正确性，从而降低过拟合的风险，提高了对未知数据的预测准确性。

- **鲁棒性好：** 即使数据中存在一些异常值，只要它们不影响支持向量的位置，模型的分类结果也不会受到太大影响。

SVM 的基本原理：线性可分情况

为了更好地理解 SVM，我们首先从最简单的情况入手：**线性可分数据**。

假设我们有两类数据点，它们可以在一个二维平面上用一条直线（这就是超平面）完全分开。

1. 超平面

在二维空间中，超平面是一条直线。在三维空间中，超平面是一个平面。在更高维空间中，超平面是一个 $(n-1)$ 维的子空间，其中 n 是特征的数量。

超平面的方程可以表示为：

$$w^T x + b = 0$$

其中：

- x 是空间中的任意一点的坐标向量，比如在二维空间中 $x = [x_1, x_2]^T$。

- w 是一个与超平面**垂直的向量**（法向量）。它的方向指明了超平面的" 朝向"。在二维空间中 $w = [w_1, w_2]^T$。
- b 是一个偏置项，它决定了超平面距离原点的远近。你可以把它理解为直线的截距，或者平面的偏移量。

这个方程是线性代数中表示平面（或直线，或更高维子空间）的标准方式。所有满足 $w^T x + b = 0$ 的点 x 构成了一个超平面。

2. 支持向量

支持向量是离超平面最近的那些数据点。它们是决定超平面位置和方向的关键点。如果移除或改变一个支持向量，超平面可能会发生变化。

3. 间隔（Margin）

间隔是超平面与最近的数据点之间的距离。在 SVM 中，我们目标是找到一个超平面，使得这个间隔最大化。

具体来说，对于一个二分类问题，我们希望将一类数据点标记为 $+1$，另一类标记为 -1。支持向量上的点满足：

- $w^T x_+ + b = +1$ （对于正类别支持向量，$y_i = +1$）
- $w^T x_- + b = -1$ （对于负类别支持向量，$y_i = -1$）

这两个方程定义的平面 $w^T x + b = +1$ 和 $w^T x + b = -1$ 称为**间隔边界（margin boundaries）**。

间隔的宽度可以计算为 $2/\|w\|$。因此，最大化间隔等价于**最小化** $\|w\|^2$。

为什么间隔宽度是 $2/\|w\|$ **?** 这两个间隔边界之间的距离就是沿着法向量 w 方向上的最短距离。通过数学推导，我们可以得出：两个平行超平面 $w^T x + b_1 = 0$ 和 $w^T x + b_2 = 0$ 之间的距离是 $|b_1 - b_2|/\|w\|$。对于间隔边界，b_1 对应于 1 (或 -1 调整后)，b_2 对应于 -1 (或 1 调整后)，所以 $(w^T x_+ + b) = 1$ 和 $(w^T x_- + b) = -1$ 这两个平面间的距离就是 $|1 - (-1)|/\|w\| = 2/\|w\|$。

4. 优化问题

所以，对于线性可分的情况，SVM 的目标是解决以下**凸优化问题**：

目标函数： 最小化 $\frac{1}{2}\|w\|^2$

约束条件： $y_i(w^T x_i + b) \geq 1$ 对于所有数据点 (x_i, y_i)，其中 y_i 是类别标签（+1 或 -1）。

为什么约束条件是大于等于 1 ? 这个约束条件统一表达了所有数据点必须被正确分类，并且位于各自类别间隔边界之外或其上。

- 对于正类别数据点 $(y_i = +1)$，我们希望 $w^T x_i + b \geq +1$。
- 对于负类别数据点 $(y_i = -1)$，我们希望 $w^T x_i + b \leq -1$。将其两边乘以 -1 并结合 $y_i = -1$，得到 $y_i(w^T x_i + b) = (-1)(w^T x_i + b) \geq 1$。

因此，无论 y_i 是 $+1$ 还是 -1，条件 $y_i(w^T x_i + b) \geq 1$ 都保证了数据点被正确分类且位于间隔边界之外或其上。

解决线性不可分问题：软间隔 SVM (Soft Margin SVM)

在现实世界中，数据往往不是完全线性可分的。可能会有一些噪音或者离群点，使得无法找到一个完美的直线或超平面将两类数据完全分开。这时，我们就需要引入**软间隔（Soft Margin）**的概念。

1. 松弛变量 (Slack Variables)

为了允许一些数据点可以被错误分类，或者位于间隔之内，我们引入了**松弛变量（ξ_i，读作”克西”）**。

- 当 $\xi_i = 0$ 时，表示数据点被正确分类且在间隔之外。
- 当 $0 < \xi_i < 1$ 时，表示数据点在间隔之内，但仍然被正确分类。
- 当 $\xi_i \geq 1$ 时，表示数据点被错误分类。

为什么会出现数据点在间隔之内的情况？在硬间隔 SVM 中，我们确实强制所有点都在间隔之外。但在**现实的非线性可分数据**中，这样做可能导致无解，因为无法找到一个能完美分隔且满足间隔约束的超平面。软间隔 SVM 的出现就是为了解决这个问题。它通过引入松弛变量，允许一些数据点”违规”（即位于间隔之内或被错误分类），从而使模型在面对复杂数据时仍然有解。同时，模型会通过优化尽量减少这些”违规”的情况。

2. 惩罚参数 C

为了控制对错误分类的容忍度，我们引入了一个**惩罚参数 C**。
目标函数： 最小化 $\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m}\xi_i$
约束条件： $y_i(w^T x_i + b) \geq 1 - \xi_i$ $\xi_i \geq 0$ 对于所有数据点 (x_i, y_i)。
参数 C 的作用：

- **C 越大：** 对误分类的惩罚越大，模型会更努力地将所有点正确分类，即使这意味着减小间隔。这可能导致**过拟合**。
- **C 越小：** 对误分类的惩罚越小，模型更倾向于选择一个更大的间隔，允许更多的误分类。这可能导致**欠拟合**。

选择合适的 C 值通常需要通过交叉验证来确定。

拉格朗日函数与对偶问题

在解决带有约束条件的优化问题时，**拉格朗日函数**提供了一种强大的数学工具，可以将这些约束”融入”到目标函数中，从而将一个**带约束优化问题**转化为一个**无约束优化问题**。这在 SVM 的求解中至关重要。

拉格朗日函数

对于一个一般的带约束优化问题：

- 最小化 $f(x)$
- 约束条件 $g_i(x) \leq 0$ （不等式约束）
- $h_j(x) = 0$ （等式约束）

它的**拉格朗日函数** $L(x, \alpha, \beta)$ 构造如下：

$$L(x, \alpha, \beta) = f(x) + \sum_i \alpha_i g_i(x) + \sum_j \beta_j h_j(x)$$

其中：

- $\alpha_i \geq 0$ 和 β_j 是**拉格朗日乘子**

KKT 条件

在求解带不等式约束的优化问题时，Karush-Kuhn-Tucker (KKT) 条件是最优解的必要条件：

1. 原始约束：$g_i(x) \leq 0$
2. 对偶约束：$\alpha_i \geq 0$
3. 互补松弛：$\alpha_i g_i(x) = 0$
4. 梯度条件：$\nabla_x L = 0$

这些条件对于理解 SVM 的解的结构至关重要。

SVM 的对偶问题

在实际求解 SVM 时，我们通常会将其转化为**对偶问题**来解决。这样做有几个重要的原因：

1. **便于引入核函数**
2. **更容易求解**
3. **支持向量的自然体现**

SVM 的拉格朗日函数为：

$$L(w, b, \xi, \alpha, \mu) = \frac{1}{2}\|w\|^2 + C\sum_{i=1}^{m} \xi_i + \sum_{i=1}^{m} \alpha_i(1 - y_i(w^T x_i + b) - \xi_i) + \sum_{i=1}^{m} \mu_i(-\xi_i)$$

通过求导并令导数为零，我们得到：

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

代入后得到对偶问题：

$$\begin{aligned} \text{最大化} \quad & \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \\ \text{约束} \quad & 0 \leq \alpha_i \leq C \\ & \sum_{i=1}^{m} \alpha_i y_i = 0 \end{aligned}$$

支持向量的重要性

支持向量是 SVM 的核心概念，它们决定了最终模型：

- **模型稀疏性：** 只有支持向量对模型有影响
- **高效预测：** 预测时只需考虑支持向量
- **鲁棒性：** 模型对非支持向量的变化不敏感

表 1: 支持向量类型及其特性

α_i 值	位置	作用
0	间隔外	不影响模型
$(0, C)$	间隔边界上	决定超平面位置
C	间隔内或误分类	代表异常点

核方法与非线性 SVM

当数据线性不可分时，核方法允许 SVM 在更高维空间中寻找线性超平面。

核技巧的数学基础

根据 Mercer 定理，任何半正定函数都可以作为核函数。常见的核函数包括：

- **线性核：** $K(x_i, x_j) = x_i^T x_j$
- **多项式核：** $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$

- **RBF 核：** $K(x_i, x_j) = \exp(-\gamma\|x_i - x_j\|^2)$
- **Sigmoid 核：** $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$

RBF 核深入解析

径向基函数（RBF）核是最常用的核函数：

$$K(x_i, x_j) = \exp\left(-\gamma\|x_i - x_j\|^2\right)$$

- γ 控制模型的复杂度：γ 越大，决策边界越复杂
- 隐式映射到无限维空间
- 适用于各种复杂的数据分布

SVM 实践指南

参数选择与调优

SVM 的性能很大程度上取决于参数选择：

- **C（惩罚参数）**：控制间隔大小与分类错误的权衡
  - 值范围：10^{-3} 到 10^3
  - 小 C：大间隔，高偏差
  - 大 C：小间隔，高方差
- γ **（RBF 核参数）**：控制单个样本的影响范围
  - 值范围：10^{-3} 到 10^3
  - 小 γ：决策边界平滑
  - 大 γ：决策边界复杂

推荐使用网格搜索（Grid Search）或随机搜索（Random Search）结合交叉验证进行参数优化。

数据预处理

- **标准化：** SVM 对特征尺度敏感，建议标准化特征（均值为 0，方差为 1）
- **处理不平衡数据：** 使用类别权重参数 `class_weight`
- **特征选择：** 移除不相关特征可提高性能

SVM 的扩展与变体

支持向量回归（SVR）

SVM 也可用于回归问题，核心思想是：

- 定义一个 ϵ-不敏感带
- 最小化带松弛变量的目标函数
- 使用核技巧处理非线性关系

多类 SVM

SVM 本质上是二分类器，但可通过以下策略处理多类问题：

- **一对多（One-vs-Rest）**：为每个类别训练一个二分类器
- **一对一（One-vs-One）**：为每对类别训练一个分类器
- **有向无环图（DAG）**：层次化决策结构

结构化 SVM

用于结构化输出预测，如：

- 序列标注
- 自然语言解析
- 图像分割

SVM 的优缺点

优点：

- **高维有效：** 特征维度 > 样本数时仍有效
- **内存高效：** 仅依赖支持向量
- **灵活性强：** 通过核函数适应各种数据
- **理论基础强：** 基于统计学习理论
- **全局最优解：** 凸优化保证找到全局最优

缺点：

- **参数敏感：**需要仔细调参
- **大规模训练慢：**时间复杂度 $O(n^2)$ 到 $O(n^3)$
- **概率估计难：**需额外处理（如 Platt scaling）
- **核选择困难：**无明确准则选择最佳核函数
- **可解释性差：**黑盒模型，尤其使用复杂核时

SVM 的应用场景

- **文本分类：**垃圾邮件检测、情感分析、主题分类
- **图像识别：**手写数字识别、人脸检测、目标识别
- **生物信息学：**基因表达分析、蛋白质结构预测
- **金融：**信用评分、欺诈检测
- **医学：**疾病诊断、医学图像分析

SVM 与其他算法比较

表 2: SVM 与其他分类算法比较

特性	SVM	逻辑回归	决策树
处理高维数据	优	良	差
处理非线性	优（核）	差	优
可解释性	中	良	优
训练速度	慢	快	快
抗噪声	优	中	差

实现建议与资源

实践建议

- 从小数据集开始，理解算法行为
- 使用 `scikit-learn` 库的 SVM 实现
- 可视化决策边界以理解模型行为
- 对重要参数（C, γ）进行系统调优

学习资源

- **经典教材：** Vapnik V. *The Nature of Statistical Learning Theory*
- **在线课程：** Andrew Ng 的机器学习课程（Coursera）
- **实用指南：** Scikit-learn 文档（`https://scikit-learn.org/stable/modules/svm.html`）
- **开源实现：** LIBSVM、LIBLINEAR

参考文献

[1] Vapnik, V. (1995). *The Nature of Statistical Learning Theory*. Springer.

[2] Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine Learning*.

[3] Pedregosa et al. (2011). Scikit-learn: Machine Learning in Python. *JMLR*.