目录

第六章　支持向量机

0.1　间隔与支持向量

0.1.1　点到超平面距离

样本空间中任一点 $\mathbf{x}$ 到超平面 $(\mathbf{w}, b)$ 的距离为：

$$r = \frac{|\mathbf{w}^T\mathbf{x} + b|}{\|\mathbf{w}\|}$$

证明方法：
1）点到直线距离公式。
2）向量方法。投影向量 $\mathbf{w}$ 垂直于超平面，点 x 对应向量 $\mathbf{x}$，过点 x 作超平面的垂线，交点 x_0 对应向量 $\mathbf{x_0}$。假设由点 x_0 指向点 x 的向量为 $\mathbf{r}$，长度（也即点 x 与超平面的距离）为 r。由向量加法定义可得 $\mathbf{x} = \mathbf{x_0} + \mathbf{r}$。其中向量 $\mathbf{r}$ 等于这个方向的单位向量乘上 r，也即有 $\mathbf{r} = \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot r$。因此又有 $\mathbf{x} = \mathbf{x_0} + \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot r$。由于点 x_0 在超平面上，所以有 $\mathbf{w}^T\mathbf{x_0} + b = 0$。由 $\mathbf{x} = \mathbf{x_0} + \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot r$ 可得 $\mathbf{x_0} = \mathbf{x} - \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot r$，代入直线方程消去 $\mathbf{x_0}$：

$$\mathbf{w}^T\mathbf{x_0} + b = \mathbf{w}^T(\mathbf{x} - \frac{\mathbf{w}}{\|\mathbf{w}\|} \cdot r) + b = 0$$

简单变换即可得到:

$$r = \frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|}$$

又因为取距离为正值，所以要加上绝对值符号：

$$r = \frac{|\mathbf{w}^T\mathbf{x} + b|}{\|\mathbf{w}\|}$$

0.1.2　间隔

根据定义，所有支持向量都满足：

$$\mathbf{w}^T\mathbf{x} + b = +1, \quad y_i = +1$$

$$\mathbf{w}^T\mathbf{x} + b = -1, \quad y_i = -1$$

代入前面的距离公式可以得到支持向量到超平面的距离为 $\frac{1}{\|\mathbf{w}\|}$。

定义间隔（margin）为两个异类支持向量到超平面的距离之和：

$$\gamma = 2 \cdot \frac{1}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}$$

SVM 的目标便是找到**具有最大间隔**（maximum margin）的划分超平面，也即找到使 γ 最大的参数 $\mathbf{w}$ 和 b：

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

看上去间隔只与 $\mathbf{w}$ 有关，但实际上位移项 b 也通过约束影响着 $\mathbf{w}$ 的取值，进而对间隔产生影响。由于最大化 $\|\mathbf{w}\|^{-1}$ 等价于最小化 $\|\mathbf{w}\|^2$，所以可以重写目标函数为：

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

这便是支持向量机的基本型。

0.2 对偶问题

式（1）是一个**带约束的凸二次规划**（convex quadratic programming）问题（凸问题就意味着必定能求到全局最优解，而不会陷入局部最优）。下面介绍一种求解方法。

首先为式（1）的每条约束添加拉格朗日乘子 $a_i \geq 0$（对应 m 个样本的 m 条约束），得到该问题的拉格朗日函数：

$$L(\mathbf{w},b,\mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + \sum_{i=1}^{m} a_i(1-y_i(\mathbf{w}^T\mathbf{x}+b)) \tag{2}$$

其中 $\mathbf{a}=(a_1;a_2;...;a_m)$，对拉格朗日函数求 $\mathbf{w}$ 和 b 的偏导，并令偏导为 0 可以得到：

$$\mathbf{w} = \sum_{i=1}^{m} a_i y_i \mathbf{x}i \tag{3}$$

$$0 = \sum_{i=1}^{m} a_i y_i \tag{4}$$

将式（3）代入式（2）可以消去 $\mathbf{w}$ 和 b，然后再考虑式（4）的约束就得到了式（1）的对偶问题（dual problem）：

$$\max_{\mathbf{a}} \sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j y_i y_j \mathbf{x}_i^T \mathbf{x}j \quad s.t. \quad \sum_{i=1}^{m} a_i y_i = 0, \quad a_i \geq 0, \quad i=1,2,...,m \tag{5}$$

只要求出对偶问题的解 $\mathbf{a}$，就可以推出 $\mathbf{w}$ 和 b，从而得到模型（不过实际计算时一般不这样做，特别是需要用核函数映射到高唯空间时，因为映射后做内积很困难，而用少量支持向量进行表示，在原始空间进行计算显然更优）：

$$\begin{aligned} f(\mathbf{x}) &= \mathbf{w}^T\mathbf{x}+b \\ &= \sum_{i=1}^{m} a_i y_i \mathbf{x}_i^T\mathbf{x}+b \end{aligned} \tag{6}$$

注意，由于式（1）的约束条件是不等式约束，所以求解过程要求满足 KKT（Karush-Kuhn-Tucker）条件：

$$\begin{cases} a_i \geq 0; \\ y_i f(\mathbf{x}_i) - 1 \geq 0; \\ a_i(y_i f(\mathbf{x}_i) - 1) = 0. \end{cases} \tag{7}$$

KKT 条件说明了，对任何一个样本来说，要么对应的拉格朗日乘子为 0，要么函数间隔等于 1（即式（1）的约束条件取等号）。如果拉格朗日乘子为 0，则这个样本对式（6）毫无贡献，不会影响到模型；如果函数间隔为 1，则表明这个样本位于最大间隔边界上，是一个支持向量。它揭示了 SVM 的一个重要性质：**最终模型只与支持向量有关，因此训练完成后，大部分的训练样本都不需保留**。

通常使用 **SMO 算法**求解式（5）。

0.3 核函数

0.3.1 如何处理非线性划分

在现实任务中，我们更常遇到的是在原始样本空间中非线性可分的问题。对这样的问题，一种常用的思路是将样本从原始空间映射到一个更高维的特征空间，使得样本在该特征空间中线性可分。幸运的是，**只要原始空间是有限维的（也即属性数目有限），那就必然存在一个高维特征空间使样本线性可分**。

0.3.2 核函数

假设用 $\phi(\mathbf{x})$ 来表示映射所得的特征向量。则在映射的高维特征空间中，用于划分的线性超平面可以表示为：

$$f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b$$

类似式（1），可以得到此时的目标函数为：

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 \quad s.t. \quad y_i(\mathbf{w}^T\phi(\mathbf{x}) + b) \geq 1, \quad i = 1, 2, ..., m \tag{8}$$

对应的对偶问题为：

$$\max_{\mathbf{a}} \sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j y_i y_j \phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j) \quad s.t. \ \sum_{i=1}^{m} a_i y_i = 0, \ a_i \geq 0, \ i = 1, 2, ..., m \tag{9}$$

注意到对偶问题中，涉及到 $\phi(\mathbf{x}_i)^T\phi(\mathbf{x}_j)$ 的计算，也即 x_i 和 x_j 映射到高维特征空间后的内积（比如 $x_i = (1, 2, 3)$，$x_j = (4, 5, 6)$，那么内积 $x_i^T x_j$ 就等于 $1 * 4 + 2 * 5 + 3 * 6 = 32$），由于特征空间维数可能很高，所以直接计算映射后特征向量的内积是很困难的，如果映射后的特征空间是无限维，根本无法进行计算。为了解决这样的问题，就引入了核函数（kernel function）。

假设输入空间是二维的，每个样本点有两个属性 x 和 y，存在映射将每个样本点映射到三维空间：

$$\phi(\mathbf{x}) = \phi(x, y) = (x^2, \sqrt{2}xy, y^2)$$

给定原始空间中的两个样本点 $\mathbf{v}_1 = (x_1, y_1)$ 和 $\mathbf{v}_2 = (x_2, y_2)$，则它们映射到高维特征空间后的内积可以写作：

$$\begin{aligned}
&\phi(\mathbf{v}_1)^T\phi(\mathbf{v}_2) =< \phi(\mathbf{v}_1), \phi(\mathbf{v}_2) > \\
=&< (x_1^2, \sqrt{2}x_1y_1, y_1^2), (x_2^2, \sqrt{2}x_2y_2, y_2^2) > \\
=& x_1^2x_2^2 + 2x_1x_2y_1y_2 + y_1^2y_2^2 = (x_1x_2 + y_1y_2)^2 \\
=&< \mathbf{v}_1, \mathbf{v}_2 >^2 \\
=& \kappa(\mathbf{v}_1, \mathbf{v}_2)
\end{aligned}$$

高维特征空间中两个点的内积，可以写成一个关于原始空间中两个点的函数 $\kappa(\cdot, \cdot)$，这就是核函数。

为什么需要核函数?

假设原始空间是二维的，那么对于两个属性 x 和 y，取一阶二阶的组合只有 5 个（也即 x^2，y^2，x，y，xy）。但当原始空间是三维的时候，仍然取一阶二阶，组合就多达 19 个了（也即 x，y，z，xy，xz，yz，x^2y，x^2z，y^2x，y^2z，z^2x，z^2y，x^2yz，xy^2z，xyz^2，x^2y^2z，x^2yz^2，xy^2z^2，xyz）。随着原始空间维数增长，新空间的维数是呈爆炸性上升的。然而有了核函数，我们就可以在原始空间中通过函数 $\kappa(\cdot;\cdot)$ 计算（这称为**核技巧**（kernel trick）），而不必直接计算高维甚至无穷维特征空间中的内积。

使用核函数后，对偶问题式（9）可以重写为：

$$\max_{\mathbf{a}} \sum_{i=1}^{m} a_i - \frac{1}{2}\sum_{i=1}^{m}\sum_{j=1}^{m} a_i a_j y_i y_j \kappa(\mathbf{x}_i; \mathbf{x}j) \quad s.t. \ \sum_{i=1}^{m} a_i y_i = 0, \ a_i \geq 0, \quad i = 1, 2, ..., m \tag{10}$$

求解后得到的模型可以表示为：

$$f(\mathbf{x}) = \mathbf{w}^T\phi(\mathbf{x}) + b = \sum_{i=1}^{m} a_i y_i \phi(\mathbf{x}_i)^T \phi(\mathbf{x}) + b = \sum_{i=1}^{m} a_i y_i \kappa(\mathbf{x}_i; \mathbf{x}) + b$$

这表明了模型最优解可通过训练样本的核函数展开，称为支持向量展式（support vector expansion）。

注意，**核函数本身不等于映射。它只是用来计算两个数据点映射到高维空间之后的内积的一种简便方法。** 当发现数据在原始空间线性不可分时，会有把数据映射到高维空间来实现线性可分的想法，比方说引入原有属性的幂或者原有属性之间的乘积作为新的维度。假设把数据点都映射到了一个维数很高甚至无穷维的特征空间，而模型求解和预测的过程需要用到映射后两个数据点的内积，这时直接计算就没辙了。但又幸运地发现，**原来高维空间中两点的内积在数值上等于原始空间通过某个核函数算出的函数值，无需先映射再求值**，就很好地解决了计算的问题了。

0.3.3 核函数定理

给定一个输入空间 $\mathcal{X}$，函数 $\kappa(\cdot;\cdot)$ 是定义在 $\mathcal{X} \times \mathcal{X}$ 上的对称函数。当且仅当对于任意数据集 $D = \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_m$, 对应的**核矩阵（kernel matrix）都是半正定**的时候，κ 是核函数。

任何一个核函数都隐式地定义了一个称为“再生核希尔伯特空间(Reproducing Kernel Hilbert Space，简称 RKHS)”的特征空间。

0.3.4 常用核函数

- 线性核：$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^T \mathbf{x}_j$
- 多项式核：$\kappa(\mathbf{x}_i, \mathbf{x}_j) = (\mathbf{x}_i^T \mathbf{x}_j)^d$，其中 $d \geq 1$ 为多项式的次数，d=1 时退化为线性核
- 高斯核（亦称 RBF 核）：$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2})$，$\sigma > 0$ 为高斯核的带宽（width）
- 拉普拉斯核：$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|}{\sigma})$，$\sigma > 0$
- Sigmoid 核：$\kappa(\mathbf{x}_i, \mathbf{x}_j) = \tanh(\beta \mathbf{x}_i^T \mathbf{x}_j + \theta)$，tanh 为双曲正切函数，$\beta > 0, \theta < 0$

核函数组合：

- 若 κ_1 和 κ_2 都是核函数，则 $a\kappa_1 + b\kappa_2$ 也是核函数，其中 $a > 0, b > 0$。
- 若 κ_1 和 κ_2 都是核函数，则其直积 $\kappa_1 \otimes \kappa_2(\mathbf{x}, \mathbf{z}) = \kappa_1(\mathbf{x}, \mathbf{z})\kappa_2(\mathbf{x}, \mathbf{z})$ 也是核函数。
- 若 κ_1 是核函数，则对于任意函数 $g(\mathbf{x})$，$\kappa(\mathbf{x}, \mathbf{z}) = g(\mathbf{x})\kappa_1(\mathbf{x}, \mathbf{z})g(\mathbf{z})$ 也是核函数。

0.4 软间隔与正则化

0.4.1 软间隔

软间隔是相对于硬间隔（hard margin）的一个概念，硬间隔要求所有样本都必须划分正确，也即约束：

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

软间隔则允许某些样本不满足约束（根据约束条件的不同，有可能某些样本出现在间隔内，甚至被误分类)。此时目标函数可以重写为：

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \ell_{0/1}(y_i(\mathbf{w}^T\mathbf{x} + b) - 1) \tag{11}$$

其中 $\ell_{0/1}$ 是 0/1 损失函数：

$$\ell_{0/1}(z) = \begin{cases} 1, & if\ z < 0; \\ 0, & otherwise. \end{cases} \tag{12}$$

它的含义很简单：如果分类正确，那么函数间隔必定大于等于 1，此时损失为 0；如果分类错误，那么函数间隔必定小于等于-1，此时损失为 1。

而 C 则是一个大于 0 的常数，当 C 趋于无穷大时，式（11）等效于带约束的式（1），因为此时对误分类的惩罚无限大，也即要求全部样本分类正确。当 C 取有限值时，允许某些样本分类错误。

由于 0/1 损失函数是一个非凸不连续函数，所以式（11）难以求解，于是在实际任务中，采用一些凸的连续函数来取替它，这样的函数就称为**替代损失（surrogate loss）函数**。

最常用的有以下三种：

- hinge 损失：$\ell_{hinge}(z) = \max(0, 1 - z)$
- 指数损失（exponential loss）：$\ell_{\exp}(z) = \exp(-z)$
- 对率损失（logistic loss）：$\ell_{\log}(z) = \log(1 + \exp(-z))$

实际任务中最常用的是 **hinge 损失**，这里就以 hinge 损失为例，替代 0/1 损失函数，此时目标函数式（11）可以重写为：

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x} + b)) \tag{13}$$

引入松弛变量（slack variables）$\xi_i \geq 0$，可以把式（13）重写为：

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{m} \xi_i \quad s.t. \quad y_i(\mathbf{w}^T\mathbf{x} + b) \geq 1 - \xi_i, \quad \xi_i \geq 0, \quad i = 1, 2, ..., m \tag{14}$$

该式描述的就是软间隔支持向量机，其中**每个样本都对应着一个松弛变量，用以表征该样本误分类的程度，值越大，程度越高。**

0.4.2 支持向量机和逻辑回归的联系与区别

上面用的是 hinge 损失，不过也提到了还有其他一些替代损失函数，事实上，使用对率损失时，SVM 得到的模型和 LR 是非常类似的。

支持向量机和逻辑回归的相同点：

- 都是线性分类器，模型求解出一个划分超平面
- 两种方法都可以增加不同的正则化项
- 通常来说性能相当

支持向量机和逻辑回归的不同点：

- LR 使用对率损失，SVM 一般用 hinge 损失
- 在 LR 的模型求解过程中，每个训练样本都对划分超平面有影响，影响力随着与超平面的距离增大而减小，所以说 **LR 的解受训练数据本身的分布影响**；SVM 的模型只与占训练数据少部分的支持向量有关，所以说，**SVM 不直接依赖数据分布**，所得的划分超平面不受某一类点的影响
- 如果数据类别不平衡比较严重，LR 需要先做相应处理再训练，SVM 则不用
- **SVM 依赖于数据表达的距离测度**，需要先把数据**标准化**，LR 则不用（但实际任务中可能会为了方便选择优化过程的初始值而进行标准化）。如果数据的距离测度不明确（特别是高维数据），那么最大间隔可能就变得没有意义

- LR 的**输出有概率意义**，SVM 的输出则没有
- LR 可以**直接用于多分类任务**，SVM 则需要进行扩展（但更常用 one-vs-rest）
- LR 使用的对率损失是光滑的单调递减函数，无法导出支持向量，解依赖于所有样本，因此预测开销较大；SVM 使用的 hinge 损失有"零区域"，因此解具有稀疏性（书中没有具体说明这句话的意思，但按我的理解是解出的拉格朗日乘子 a 具有稀疏性，而不是权重向量 w），从而不需用到所有训练样本。
- **在实际运用中，LR 更常用于大规模数据集，速度较快；SVM 适用于规模小，维度高的数据集。**

如果 Feature 的数量很大，跟样本数量差不多，这时候选用 LR 或者是 Linear Kernel 的 SVM。
如果 Feature 的数量比较小，样本数量一般，不算大也不算小，选用 SVM+Gaussian Kernel。
如果 Feature 的数量比较小，而样本数量很多，需要手工添加一些 feature 变成第一种情况。

0.5 核方法

0.5.1 表示定理（representer theorem）

令 $\mathbb{H}$ 为核函数 κ 对应的再生希尔伯特空间，$\|h\|_{\mathbb{H}}$ 表示 $\mathbb{H}$ 空间中关于 h 的范数，对于任意单调递增函数 $\Omega : [0, \infty] \longmapsto \mathbb{R}$ 和任意非负损失函数 $\ell : \mathbb{R}^m \longmapsto [0, \infty]$，优化问题

$$\min_{h \in \mathbb{H}} F(h) = \Omega(\|h\|_{\mathbb{H}}) + \ell(h(\mathbf{x}_1), h(\mathbf{x}_2), ..., h(\mathbf{x}_m)) \tag{14}$$

的解总可写为：

$$h^*(\mathbf{x}) = \sum_{i=1}^{m} a_i \kappa(\mathbf{x}, \mathbf{x}_i)$$

这个定理表明，**旨在最小化损失和正则化项之和的优化问题，解都可以表示为核函数的线性组合。**
基于核函数的学习方法，统称为核方法（kernal methods）。**最常见的就是通过核化（引入核函数），将线性学习器扩展为非线性学习器。**