Support Vector Machines and k-Nearest Neighbors (Lecture Notes)

根据
纽约大学柯朗数学研究所 Mehryar Mohri 的 *Foundations of Machine Learning*
利物浦大学Dom Richards的*COMP336/COMP529 Big Data Analytics Lecture Note*
编写
ChatGPT-4o 翻译

1 支持向量机（SVM）

1.1 线性可分的严格定义

给定一个训练数据集

$$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^n, \quad \mathbf{x}_i \in \mathbb{R}^d, \quad y_i \in \{-1, +1\},$$

我们称该数据集在输入空间 $\mathbb{R}^d$ 中是线性可分的，若存在一个超平面由 $\mathbf{w} \in \mathbb{R}^d$ 与标量 $b \in \mathbb{R}$ 确定，使得对所有 $i \in \{1, \ldots, n\}$ 均有

$$y_i\big(\mathbf{w}^\top \mathbf{x}_i + b\big) > 0.$$

更严格地，还可以要求存在某个正数 $\gamma > 0$，对所有 i 都有

$$y_i\big(\mathbf{w}^\top \mathbf{x}_i + b\big) \geq \gamma.$$

若满足此条件，则称 γ 为几何间隔（geometric margin）。在线性可分情况下，我们期望在所有能将数据正确分类的超平面中，找到几何间隔最大的那一个。

1.2 硬间隔线性可分情形下的优化问题

在线性可分的理想情形下，SVM 试图求解以下**硬间隔最大化**问题:

$$\min_{\mathbf{w}, b} \quad \frac{1}{2}\|\mathbf{w}\|^2$$

$$\text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1, \quad i = 1, \ldots, n.$$

将几何间隔定义为 $\frac{2}{\|\mathbf{w}\|}$，最小化 $\frac{1}{2}\|\mathbf{w}\|^2$ 等价于最大化几何间隔（倒数）。

1.2.1 拉格朗日乘子法与对偶问题

为便于求解，我们将上述带约束的原问题转化为对偶问题。构造拉格朗日函数:

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^n \alpha_i \Big[y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \Big],$$

其中 $\alpha_i \geq 0$ 为拉格朗日乘子（Lagrange multipliers）。接下来，对 $\mathbf{w}$ 与 b 分别求偏导并令其为 0，可得

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0 \quad \Longrightarrow \quad \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i,$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0 \quad \Longrightarrow \quad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

将上述结果代入拉格朗日函数，可得到**对偶问题**：

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x}_i^\top \mathbf{x}_j$$

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

这是一个**二次规划**（Quadratic Programming, QP）问题，采用成熟的优化方法可高效求解。

1.2.2 KKT 条件

KKT 条件（Karush-Kuhn-Tucker conditions）是最优解 $(\mathbf{w}^*, b^*, \boldsymbol{\alpha}^*)$ 必须满足的一组必要条件，包括：

1. **原始可行性（Primal feasibility）**：

$$y_i\big(\mathbf{w}^\top \mathbf{x}_i + b\big) - 1 \;\; \geq \;\; 0, \quad \forall i;$$

2. **对偶可行性（Dual feasibility）**：

$$\alpha_i \;\; \geq \;\; 0, \quad \forall i;$$

3. **互补松弛（Complementary slackness）**：

$$\alpha_i \; \Big[\, y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 \Big] \;\; = \;\; 0, \quad \forall i;$$

4. **站立性（Stationarity）**：

$$\mathbf{w} - \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i = 0, \quad \sum_{i=1}^{n} \alpha_i y_i = 0.$$

根据互补松弛条件可知，若 $\alpha_i > 0$，则 $y_i(\mathbf{w}^\top \mathbf{x}_i + b) - 1 = 0$，这些对应样本称为支持向量；若 $\alpha_i = 0$，则不一定严格落在边界上。

1.3 软间隔与正则化

在现实任务中，线性可分往往不成立，因此需允许少量分类错误。通过引入松弛变量 $\xi_i \geq 0$ 并加入正则化参数 $C > 0$，得到**软间隔**优化问题：

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

$$\text{subject to} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i, \quad \xi_i \geq 0.$$

其中 C 用于平衡间隔的最大化和误分类的惩罚，实际应用中通常通过交叉验证来选择合适的 C。

1.4 核函数与核技巧（Kernel Trick）

1.4.1 高维映射的动机

对于线性不可分的数据，可以构造一个映射

$$\phi : \mathbb{R}^d \ \to \ \mathcal{H},$$

将输入空间 $\mathbb{R}^d$ 中的每个向量 $\mathbf{x}$ 映射到某个（甚至无限维的）希尔伯特空间 $\mathcal{H}$。在 $\mathcal{H}$ 中，我们可能获得线性可分的优势。SVM 在该空间中寻找超平面

$$\mathbf{w}^\top \phi(\mathbf{x}) + b = 0.$$

然而，显式地计算 $\phi(\mathbf{x})$ 可能会非常昂贵或不可行（尤其在高维或无限维情形）。

1.4.2 核函数（Kernel Function）

若存在核函数

$$k(\mathbf{x}_i, \mathbf{x}_j) \ = \ \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j),$$

则 SVM 的**对偶目标函数**只需要在训练过程中计算形如 $k(\mathbf{x}_i, \mathbf{x}_j)$ 的内积，而不需要显式构造 $\phi(\cdot)$。这个过程被称为**核技巧**（kernel trick）。

1.4.3 常见核函数

线性核（Linear Kernel）

$$k(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^\top \mathbf{x}_j.$$

这是最简单的核函数，对应的映射 $\phi(\mathbf{x})$ 就是 $\mathbf{x}$ 本身。在线性数据、或高维稀疏数据场景中常被使用。

多项式核（Polynomial Kernel）

$$k(\mathbf{x}_i, \mathbf{x}_j) = \big(\mathbf{x}_i^\top \mathbf{x}_j + c\big)^p,$$

其中 $c \geq 0$, p 为多项式次数。高阶多项式核可以捕捉更复杂的决策边界，维数易爆炸，核技巧大幅降低了这种复杂度。

高斯核（RBF Kernel）

$$k(\mathbf{x}_i, \mathbf{x}_j) = \exp\Big(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\Big).$$

对应了将数据映射到无穷维空间，RBF 核具有平滑和局部特性，是最常用的核之一。

Sigmoid 核（神经网络核）

$$k(\mathbf{x}_i, \mathbf{x}_j) = \tanh\big(\alpha\, \mathbf{x}_i^\top \mathbf{x}_j + c\big).$$

与两层神经网络的激活函数类似，但并非总是满足 Mercer 条件，需要根据具体超参数判断是否正定。

一旦选择合适的核函数，SVM 的对偶问题变为

$$\max_{\boldsymbol{\alpha}} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j \, y_i y_j \, k(\mathbf{x}_i, \mathbf{x}_j),$$

并得到决策函数

$$f(\mathbf{x}) = \operatorname{sgn}\Big(\sum_{i=1}^{n} \alpha_i y_i \, k(\mathbf{x}, \mathbf{x}_i) + b \Big).$$

1.5 再生核希尔伯特空间（RKHS）

1.5.1 RKHS 的定义与再生性质

考虑一个希尔伯特空间 $\mathcal{H}$ 上定义的内积 $\langle \cdot, \cdot \rangle_{\mathcal{H}}$。若存在一个核函数 $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ 满足：

$$k(\mathbf{x}, \mathbf{y}) = \big\langle k(\mathbf{x}, \cdot),\ k(\mathbf{y}, \cdot) \big\rangle_{\mathcal{H}},$$

且对任何 $f \in \mathcal{H}$ 与 $\mathbf{x} \in \mathcal{X}$ 都满足

$$f(\mathbf{x}) = \big\langle f,\ k(\mathbf{x}, \cdot) \big\rangle_{\mathcal{H}},$$

则称 $\mathcal{H}$ 为**再生核希尔伯特空间**（Reproducing Kernel Hilbert Space, RKHS），而 k 称为**再生核**（reproducing kernel）。

这里：

$$k(\mathbf{x}, \cdot)$$

是一个从 $\mathcal{X}$ 到 $\mathbb{R}$ 的函数，位于 $\mathcal{H}$ 当中，使得核函数正好是它们在 $\mathcal{H}$ 中的内积。此性质称为**再生性质（reproducing property）**。

1.5.2 将 RKHS 引入 SVM 的动机

- **统一的理论框架：** RKHS 提供了核方法的严谨数学基础，使得“用核函数代替内积”的做法有了系统化解释。
- **代表定理（Representer Theorem）：** 在许多核学习方法（包括带有平方范数正则化的 SVM）中，最优解 $f^* \in \mathcal{H}$ 可以写成

  $$f^*(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i \, k(\mathbf{x}_i, \mathbf{x}).$$

  该定理说明，虽然 $\mathcal{H}$ 可能是无限维，但最优解实际仅在训练点对应的核函数上取线性组合。
- **可解释的几何：** 在 RKHS 中，$\|f\|_{\mathcal{H}}$ 可以解释为函数的复杂度度量。最小化 $\|f\|_{\mathcal{H}}^2$ + 损失函数相当于在函数复杂度与经验误差之间求折中，这与 SVM 最大间隔或软间隔的思想相吻合。

1.5.3 符号与公式含义

在以上描述中：

- $\mathcal{X}$ 是输入空间，如 $\mathbb{R}^d$。
- $\mathcal{H}$ 是一个希尔伯特空间（可能是无限维），其中的向量可以视为函数。
- $k(\cdot,\cdot)$ 是**再生核**，满足对任意 $\mathbf{x},\mathbf{y}\in\mathcal{X}$，

$$k(\mathbf{x},\mathbf{y}) = \langle\, k(\mathbf{x},\cdot),\ k(\mathbf{y},\cdot)\rangle_{\mathcal{H}}.$$

- $\langle\cdot,\cdot\rangle_{\mathcal{H}}$ 表示 $\mathcal{H}$ 上的内积。
- $f(\mathbf{x})$ 对应在 $\mathcal{H}$ 中的某个向量（函数）与 $k(\mathbf{x},\cdot)$ 的内积：

$$f(\mathbf{x}) = \langle f,\ k(\mathbf{x},\cdot)\rangle_{\mathcal{H}}.$$

在核方法中，这一框架解释了为什么我们可以用 $\mathbf{x}_i$ 与 $\mathbf{x}_j$ 的核函数值（而非显式映射后的内积）来进行学习和推断，同时也说明了为什么存在如 SVM 对偶解这样的“稀疏”形式。

2 k-近邻（kNN）

2.1 算法流程

对于给定的数据集

$$\{(\mathbf{x}_i, y_i)\}_{i=1}^{n},\quad y_i\in\{-1,+1\},$$

k-近邻算法的**基本步骤**可总结为：

1. **确定距离度量：** 常见如欧式距离 $d(\mathbf{x}_i,\mathbf{x}_j)=\|\mathbf{x}_i-\mathbf{x}_j\|_2$。具体任务下也可选择更合适的度量。
2. **寻找 k 个邻居：** 对目标点 $\mathbf{x}$，计算它与训练集所有点的距离，选出与其距离最小的 k 个样本。
3. **投票或加权投票：** 在分类任务中，根据这 k 个邻居的标签 $\{y_i\}$ 进行多数表决或加权表决，得到最终预测类别。

2.2 数学视角下的近邻

kNN 是一种**非参数方法**：它并不对训练数据进行显式的参数化拟合，而是在做预测时依赖数据“局部”分布。可将 kNN 看作对后验概率 $P(Y|\mathbf{x})$ 的一个邻域估计：

$$\hat{P}(y\mid\mathbf{x}) \approx \frac{1}{k}\sum_{\mathbf{x}_i\in\mathcal{N}_k(\mathbf{x})}\mathbf{1}_{\{y_i=y\}},$$

其中 $\mathcal{N}_k(\mathbf{x})$ 表示 $\mathbf{x}$ 的 k 个近邻，$\mathbf{1}_{\{\cdot\}}$ 为指示函数。

2.3 距离度量及加权策略

在实践中，距离度量对 kNN 影响较大。除了欧式距离，还可采用曼哈顿距离、切比雪夫距离等。对于特征维度不同或量纲差异大的数据，可做归一化或选用更专业的距离。投票时常使用加权形式，例如距离的倒数：

$$w_i = \frac{1}{d(\mathbf{x}, \mathbf{x}_i) + \varepsilon},$$

使离得更近的邻居权重更大。

2.4 k 的选择

通过**交叉验证**等方式可选定合适的 k。当 k 太小，容易过拟合；当 k 太大，则欠拟合并丧失局部决策优势。