# 第 11 章 线性模型用于分类

## 11.1 二分类的线性模型

**定义 11.1.1 (误差函数回顾)**

给定线性评分函数 $s = w^\top x$，二分类标签 $y \in \{-1, +1\}$，三种线性模型的假设与误差函数如下：

| 模型 | 假设 $h(x)$ | 误差函数 $\mathrm{err}(h, x, y)$ | 备注 |
|---|---|---|---|
| 线性分类 | $\mathrm{sign}(s)$ | $\mathbb{I}[\mathrm{sign}(s) \neq y]$ | 0/1 误差 |
| 线性回归 | $s$ | $(s-y)^2$ | 平方误差 |
| 逻辑回归 | $\sigma(s) = \dfrac{1}{1+e^{-s}}$ | $-\ln \sigma(ys) = \ln\left(1 + e^{-ys}\right)$ | 交叉熵误差 |

统一视角：所有误差均可写成关于"分类正确性得分" $z = ys$ 的函数

$$
\begin{aligned}
\mathrm{err}_{0/1}(s, y) &= \mathbb{I}[sign(s) \neq y] = \mathbb{I}[sign(ys) \neq 1] = \mathbb{I}[sign(z) \neq 1], \\
\mathrm{err}_{\mathrm{SQR}}(s, y) &= (s-y)^2 = (ys-1)^2 = (z-1)^2, \\
\mathrm{err}_{\mathrm{CE}}(z) &= \ln(1 + e^{-ys}) = \ln(1 + e^{-z}).
\end{aligned}
$$

♣

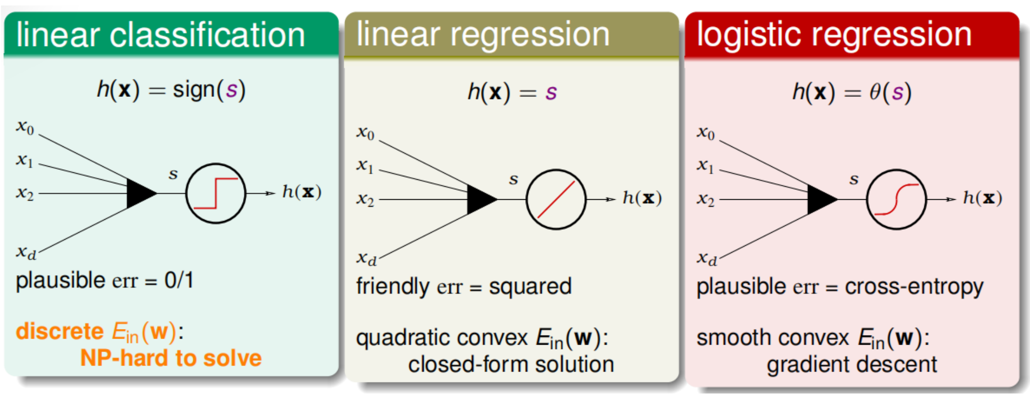


图 11.1.1: 线性分类、线性回归与逻辑回归对比示意图

**定义 11.1.2 (误差函数可视化与上界性质)**

令 $z = ys$ 为"分类正确性得分"，三种误差函数如下：

- **0/1**：$\mathrm{err}_{0/1}(z) = \mathbb{I}[z \leq 0]$，仅在 $z \leq 0$ 时取 1。
- **SQR**：$\mathrm{err}_{\mathrm{SQR}}(z) = (z-1)^2$，在 $z < 1$ 时较大；当 $z > 1$ 时过度惩罚。
- **CE**：$\mathrm{err}_{\mathrm{ce}}(z) = \ln(1 + e^{-z})$，较小的 $\mathrm{err}_{\mathrm{CE}}$ 等价于较小的 $\mathrm{err}_{0/1}$；
- **Scaled CE**：$\mathrm{err}_{\mathrm{SCE}}(z) = \log_2(1 + e^{-z})$，单调递减且为 0/1 误差的紧致上界：

$$
\mathrm{err}_{0/1}(z) \leq \mathrm{err}_{\mathrm{SCE}}(z), \quad \forall z \in \mathbb{R}.
$$

因此，Scaled CE 可用于设计算法误差函数，保证“小的 Scaled CE ⇒ 小的 0/1 误差”。♣

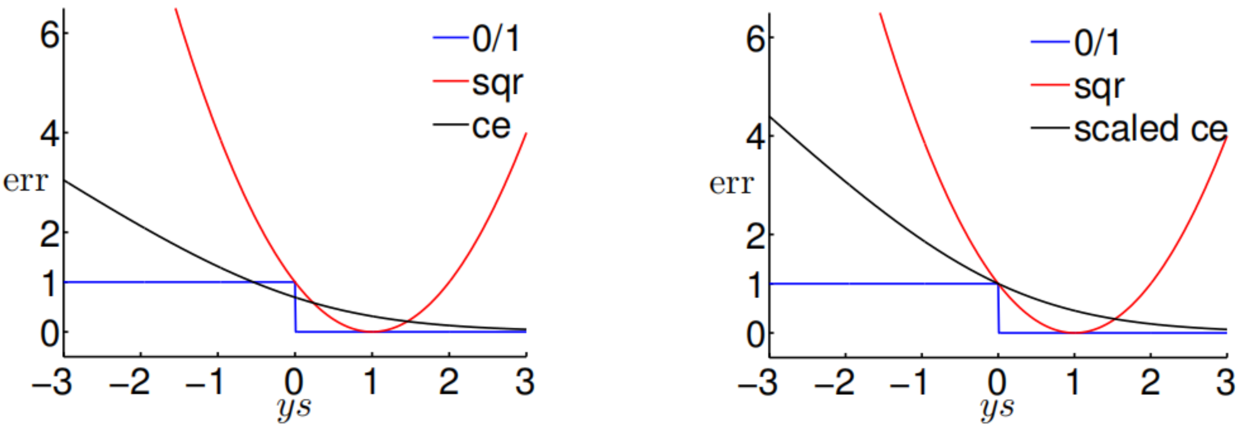


图 11.1.2: 二分类问题中不同误差度量对比图

**命题 11.1.1 (上界的理论含义)**

设 $s = w^\top x$。对任意 $y \in \{-1,+1\}$ 有逐点误差关系

$$\mathrm{err}_{0/1}(s,y) \le \mathrm{err}_{\mathrm{SCE}}(s,y) = \frac{1}{\ln 2}\cdot \mathrm{err}_{\mathrm{CE}}(s,y), \qquad \mathrm{err}_{\mathrm{CE}}(s,y) = \ln\big(1+e^{-ys}\big).$$

从而整体误差满足

$$E_{\mathrm{in}}^{0/1}(w) \le E_{\mathrm{in}}^{\mathrm{SCE}}(w) = \frac{1}{\ln 2}\cdot E_{\mathrm{in}}^{\mathrm{CE}}(w).$$

$$E_{\mathrm{out}}^{0/1}(w) \le E_{\mathrm{out}}^{\mathrm{SCE}}(w) = \frac{1}{\ln 2}\cdot E_{\mathrm{out}}^{\mathrm{CE}}(w),$$

泛化界结合

- 对 0/1 误差用 VC 界：$E_{\mathrm{out}}^{0/1}(w) \le E_{\mathrm{in}}^{0/1}(w) + \Omega^{0/1} \le \frac{1}{\ln 2}E_{\mathrm{in}}^{CE}(w) + \Omega^{0/1}$；
- 对交叉熵误差用 VC-Reg 界：$E_{\mathrm{out}}^{0/1}(w) \le \frac{1}{\ln 2}E_{\mathrm{out}}^{\mathrm{CE}}(w) \le \frac{1}{\ln 2}E_{\mathrm{in}}^{CE}(w) + \frac{1}{\ln 2}\Omega^{0/1}$。

结论　当交叉熵误差 $E_{\mathrm{in}}^{\mathrm{CE}}(w)$ 足够小时，0/1 误差 $E_{\mathrm{out}}^{0/1}(w)$ 也必然小，因此逻辑回归 / 线性回归可作为线性分类的高效近似方法。♠

**命题 11.1.2 (回归用于分类：比较与权衡)**

给定二分类数据 $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$，$y_n \in \{-1,+1\}$。

- 线性回归（**Linear Regression**）

  优点　解析解，计算最快；

  缺点　仅在数据线性可分时表现良好，否则需借助 Pocket 等启发式；

  用途　常作为 PLA / Pocket / Logistic 回归的初始权重 $w_0$。

- 逻辑回归（**Logistic Regression**）

  优点　凸优化，有强理论保证（交叉熵误差 ⇒ 0/1 误差上界）；

  缺点　相比线性回归，优化更耗时；

  结论　实践中通常优于 Pocket，是首选的线性分类器。♠

## 11.2 随机梯度下降

**算法 11.2.1:** 两种迭代优化方案

**输入:** 训练集 $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$，初值 $w_0$

**输出:** 权重向量 $w$

**for** $t = 0, 1, 2, \ldots$ **do**

- **if** *PLA* **then**
  - // 每次只看一个样本
  - 随机选取误分类样本 $(x_n, y_n)$;
  - 更新 $w_{t+1} \leftarrow w_t + y_n x_n$;
  - 单次迭代复杂度 $O(1)$;
- **else if** *Logistic Regression*（批量）**then**
  - // 每次遍历全部样本
  - 计算梯度 $\nabla E(w_t) = \frac{1}{N}\sum_{n=1}^N \sigma(-y_n w_t^\top x_n)(-y_n x_n)$;
  - 更新 $w_{t+1} \leftarrow w_t - \eta \nabla E(w_t)$;
  - 单次迭代复杂度 $O(N)$;
- **else if** *Pocket* **then**
  - 与 PLA 类似，但保留历史最佳权重，复杂度 $O(N)$;

当满足停止准则时，返回最终 $w$ 作为 $g$;

**算法 11.2.2:** 逻辑回归的随机梯度下降（SGD）

**输入:** 数据集 $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$，学习率 $\eta > 0$

**输出:** 权重向量 $w$

初始化 $w_0$;

**for** $t = 0, 1, 2, \ldots$ **do**

- 从 $\{1, \ldots, N\}$ 中均匀随机抽取一个下标 $n$;
- 计算单个样本的梯度

$$\nabla_w \mathrm{err}(w_t, x_n, y_n) = \sigma\big(-y_n w_t^\top x_n\big)(-y_n x_n);$$

- 更新权重

$$w_{t+1} \leftarrow w_t + \eta \underbrace{\sigma\big(-y_n w_t^\top x_n\big)\, y_n x_n}_{\text{随机梯度}};$$

- **if** 满足停止准则 **then break;**

**return** 最终 $w$;

注 随机梯度 $\nabla_w \mathrm{err}(w, x_n, y_n)$ 是对真实梯度

$$\nabla_w E_{\mathrm{in}}(w) = \frac{1}{N}\sum_{n=1}^N \nabla_w \mathrm{err}(w, x_n, y_n)$$

的无偏估计且每迭代仅需 $O(1)$ 时间，体现出了随机梯度的两个关键性质：**无偏性**和**高效性**。

**命题 11.2.1 (随机梯度下降 SGD)**

设真实梯度

$$\nabla E_{\text{in}}(w) = \frac{1}{N}\sum_{n=1}^{N} \nabla \text{err}(w, x_n, y_n),$$

随机梯度

$$\nabla_{\text{SGD}}(w) = \nabla \text{err}(w, x_n, y_n) \quad (n\ \text{均匀随机}),$$

则

$$\nabla_{\text{SGD}}(w) = \nabla E_{\text{in}}(w) + \underbrace{\text{零均值噪声}}_{\text{零期望}}.$$

算法

$$w_{t+1} \leftarrow w_t - \eta_t \nabla_{\text{SGD}}(w_t).$$

优缺点

- 优点：计算简单、成本低廉，适合大数据或在线学习；
- 缺点：方向噪声大，收敛路径不稳定。

逻辑回归 **SGD** 更新

$$w_{t+1} \leftarrow w_t + \eta\, \underbrace{\sigma(-y_n w_t^\top x_n)\, y_n x_n}_{-\nabla \text{err}(w, x_n, y_n)}.$$

注 SGD 的更新规则是沿着负随机梯度方向更新参数 $w$。

**命题 11.2.2 (PLA 再探：与 SGD 逻辑回归的对应)**

令 $s_n = w_t^\top x_n$，则两种算法的更新可写成统一形式：

- **SGD** 逻辑回归

$$w_{t+1} = w_t + \eta\, \sigma(-y_n s_n)\,(y_n x_n).$$

- **PLA**

$$w_{t+1} = w_t + \mathbb{I}[y_n \neq \text{sign}(s_n)]\,(y_n x_n).$$

关系

- SGD 逻辑回归可视为“软”PLA：权重 $\sigma(-y_n s_n) \in (0,1)$ 随置信度连续变化；
- PLA 相当于 SGD 逻辑回归在 $|s_n|$ 很大时的极限特例，取 $\eta = 1$ 且仅当犯错才更新。

两条实用经验

- 停止条件：迭代次数 $t$ 足够大即可；
- 学习率：若输入 $x$ 已归一化到合理范围，取 $\eta \approx 0.1$。

### 例题 11.1 选择题：大数据线性回归的 SGD 更新方向

考虑在大数据场景下对线性回归应用随机梯度下降（SGD）。当使用负随机梯度时，其更新方向是什么？

1) $\boldsymbol{x}_n$
2) $y_n \boldsymbol{x}_n$

3) $2(\boldsymbol{w}_t^T\boldsymbol{x}_n - y_n)\boldsymbol{x}_n$

4) $2(y_n - \boldsymbol{w}_t^T\boldsymbol{x}_n)\boldsymbol{x}_n$

**解答** 正确选项为 $\boxed{4}$。在线性回归中，平方误差函数为 $\text{err}(w, x_n, y_n) = \left(w^\top x_n - y_n\right)^2$。对其关于 $w$ 求梯度可得

$$\nabla_w \text{err}(w, x_n, y_n) = 2\left(w^\top x_n - y_n\right)\boldsymbol{x}_n.$$

SGD 的更新规则是沿着负随机梯度方向更新参数 $w$，即 $w_{t+1} = w_t - \eta \cdot \nabla_w \text{err}(w_t, x_n, y_n)$，所以更新方向为负梯度：$-\nabla_w \text{err}(w_t, x_n, y_n) = 2\left(y_n - \boldsymbol{w}_t^T\boldsymbol{x}_n\right)\boldsymbol{x}_n$。

该更新规则具有直观的物理解释：通过与残差 $(y_n - \boldsymbol{w}_t^T\boldsymbol{x}_n)$ 成比例地“修正”来改进 $\boldsymbol{w}_t$。 ■

## 11.3 逻辑回归的多分类

**定义 11.3.1 (硬分类（Hard Classification）)**

在多类或多标签预测任务中，若将每个样本唯一地（多类）或唯一地集合（多标签）分配到一个或多个确定的类别，而不输出任何概率或置信度，则称该过程为硬分类。在组合二元分类器（如一对多或一对一策略）的框架下，硬分类通过以下方式实现：

- 多类：各二元分类器输出 $\{-1, +1\}$，最终经投票或最大响应得到唯一类别标签。
- 多标签：各标签由独立的二元分类器给出 $\{0, 1\}$ 硬决策，组合后形成样本的标签集合。

硬分类的核心特征为：仅输出离散类别标签，不涉及概率估计或软决策。 ♣

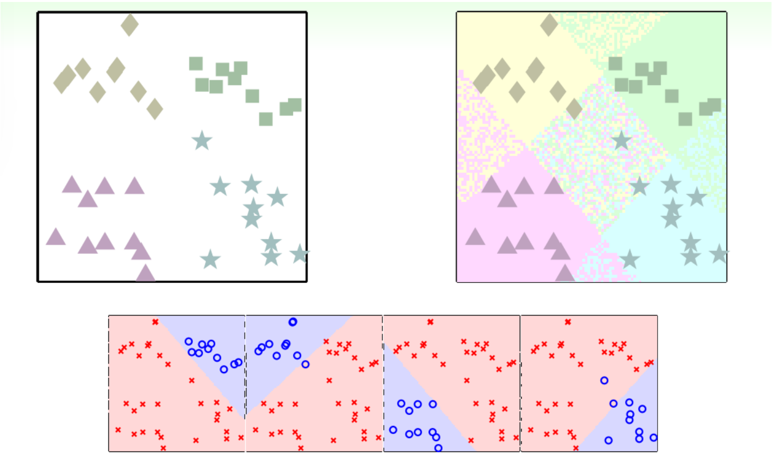

图 11.3.1: 硬分类模型分类效果示例图

**定义 11.3.2 (多类预测：组合软分类器（Soft Classification）)**

在多类或多标签任务中，若将若干个二元概率分类器（如逻辑回归、软输出 SVM）的输出概率或置信度进行组合，以产生对每一类别（或标签）的后验概率估计，并据此做出最终预测，则称该过程为组合软分类器。

形式化地，给定 $K$ 个类别，通过策略（一对多、一对一、纠错输出码等）训练 $M$ 个二元分类器，每个分类器输出

$$h_m(x) \in [0, 1], \quad m = 1, \ldots, M,$$

表示样本 $x$ 属于某一子集的概率。组合函数

$$\hat{P}(y=k \mid x)=g\big(h_1(x),\dots,h_M(x)\big), \quad k=1,\dots,K$$

将各分类器软输出映射为类别后验概率，最终预测取

$$\hat{y}=\arg\max_k \hat{P}(y=k \mid x).$$

由于整个过程保留并利用了概率输出，而非直接取硬标签，故称“软分类器组合”。♣

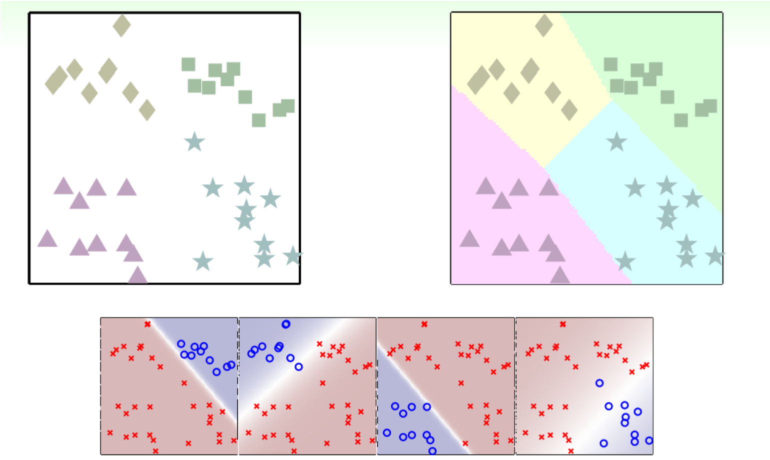

图 11.3.2: 软分类模型分类效果示例图

**算法 11.3.3:** One-Versus-All (OVA) 多类逻辑回归

**输入:** 训练集 $\mathcal{D}=\{(x_n,y_n)\}_{n=1}^N$，类别数 $K$

**输出:** 预测函数 $g:\mathbb{R}^d\to\{1,\dots,K\}$

**for** $k=1$ **to** $K$ **do**

构造二分类标签

$$y_n^{(k)} \leftarrow \begin{cases} +1, & y_n=k, \\ -1, & \text{otherwise}; \end{cases}$$

在 $\{(x_n,y_n^{(k)})\}$ 上运行逻辑回归，得到权重 $w_{[k]}$;

定义最终预测

$$g(x)=\arg\max_{k\in\{1,\dots,K\}} w_{[k]}^\top x;$$

**return** $g$

**优点**：高效、可复用任意逻辑回归类算法

**缺点**：$K$ 大时子集极不平衡（损失函数的权重倾斜：负样本的损失贡献占比更高）

**扩展**：多项逻辑回归可缓解不平衡

**例题 11.2 选择题：OVA 分解的训练代价**

考虑对大小为 $N$ 的 $K$ 类分类数据，基于逻辑回归进行 OVA 分解。下列哪一项最能描述其训练代价？

1) 学习 $K$ 个逻辑回归假设，每个使用大小为 $N/K$ 的数据
2) 学习 $K$ 个逻辑回归假设，每个使用大小为 $N\ln K$ 的数据
3) 学习 $K$ 个逻辑回归假设，每个使用大小为 $N$ 的数据

4) 学习 $K$ 个逻辑回归假设，每个使用大小为 $NK$ 的数据

**解答** 正确选项为 $\boxed{3}$。OVA 分解将 $K$ 类分类问题拆解为 $K$ 个二分类子问题。对于第 $k$ 个子问题，构造的数据集包含原始数据中所有属于类别 $k$ 的样本（正样本）和所有不属于类别 $k$ 的样本（负样本），即每个子问题的数据集大小为 $N$。

因此，每个二分类子问题都需在完整的 $N$ 个样本上训练逻辑回归模型，总需学习 $K$ 个逻辑回归假设，每个对应的数据量为 $N$。 ■

## 11.4 二元分解多分类

**定义 11.4.1 (多类预测：组合成对分类器（One-vs-One, OvO）)**

设类别集合 $\mathcal{Y} = \{1, 2, \ldots, K\}$。成对分解（One-vs-One）通过以下两步将多类问题转化为多个二元问题：

1. 训练阶段对每一对类别 $(i, j)$，$1 \le i < j \le K$，仅保留类别 $i$ 与 $j$ 的样本，构造二元数据集

$$\mathcal{D}_{ij} = \{(x_n, y_n) \mid y_n \in \{i, j\}\}, \quad y_n^{(ij)} = \begin{cases} +1, & y_n = i, \\ -1, & y_n = j. \end{cases}$$

   在此数据集上训练 $K(K-1)/2$ 个二元分类器

$$h_{ij} : \mathbb{R}^d \to \{-1, +1\}.$$

2. 预测阶段对测试样本 $x$，将所有 $h_{ij}(x)$ 的预测结果进行投票：

$$g(x) = \arg\max_{k \in \mathcal{Y}} \sum_{i<j} \mathbb{I}[h_{ij}(x) = \mathrm{sign}(k-i) \cdot \mathrm{sign}(k-j)].$$

   得票最多的类别即为最终输出。

性质

- 每个二元子集仅含两类别，通常样本更均衡；
- 需训练 $\binom{K}{2}$ 个分类器，预测时需 $\binom{K}{2}$ 次调用；
- 投票平局可用置信度或距离度量解决。

♣

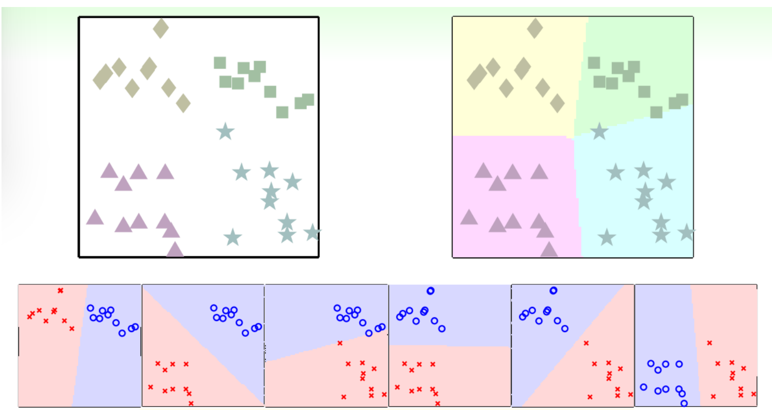

图 11.4.1: 组合成对分类器分类效果示例图

**算法 11.4.4:** 一对一成对分解（OVO）多类算法

**输入:** 训练集 $\mathcal{D} = \{(x_n, y_n)\}_{n=1}^N$，类别数 $K$

**输出:** 预测函数 $g : \mathbb{R}^d \to \{1, \ldots, K\}$

**for** 每对类别 $(k, \ell) \in \{1, \ldots, K\} \times \{1, \ldots, K\}$, $k < \ell$ **do**

构造二分类子集

$$\mathcal{D}_{k,\ell} = \{(x_n, \tilde{y}_n) \mid y_n = k \text{ 或} y_n = \ell\},\ \tilde{y}_n = +1 \text{ if } y_n = k,\ -1 \text{ if } y_n = \ell.$$

在 $\mathcal{D}_{k,\ell}$ 上训练二元分类器 $w_{[k,\ell]}$;

**预测阶段**

对输入 $x$ 执行所有 $w_{[k,\ell]}$ 并投票：

$$g(x) = \arg\max_k \sum_{i<j} \mathbb{I}[\text{sign}(w_{[i,j]}^\top x) = \text{vote for } k].$$

**优点**

- 每个子问题仅含两类，通常更小、更均衡、更稳定；
- 可复用任意二元分类算法。

**缺点**

- 需存储及训练 $O(K^2)$ 个分类器；
- 预测时需 $O(K^2)$ 次调用，空间与时间开销大。

OVO 是另一种简单实用的多类元算法，值得放入工具箱。

**例题 11.3 选择题：OVO 分解的总计算代价**

考虑对包含 10 个类别的分类问题采用 OVO（One-Versus-One）分解，总样本数为 $N$ 且各类别样本数均衡。若单个二分类器在大小为 $M$ 的数据集上的 CPU 时间为 $M^3$，则 OVO 分解的总 CPU 时间为：

1) $\frac{9}{200}N^3$
2) $\frac{9}{25}N^3$
3) $\frac{4}{5}N^3$
4) $N^3$

**解答** 正确选项为 $\boxed{2}$。OVO 分解将 $K$ 类问题拆解为 $\binom{K}{2} = \frac{K(K-1)}{2}$ 个二分类子问题。对于 10 类问题，需训练 $\binom{10}{2} = 45$ 个分类器。

由于样本均衡，每个类别含 $\frac{N}{10}$ 个样本，每对子问题仅使用两个类别的样本，故单个子问题的数据集大小为 $\frac{N}{10} + \frac{N}{10} = \frac{N}{5}$。

单个子问题的 CPU 时间为 $\left(\frac{N}{5}\right)^3 = \frac{N^3}{125}$，总 CPU 时间为 $45 \times \frac{N^3}{125} = \frac{9}{25}N^3$。 ■

注 使用相同算法的 OVA 分解将花费 $10N^3$ 的时间，这比 OVO 差得多。

## 11.5 总结

笔记 [线性模型用于分类]

- 二分类的线性模型：三种模型各有适用场景。
- 随机梯度下降：沿负的随机梯度方向更新参数。
- 逻辑回归的多分类：选取估计概率最大的类别 $\arg\max_k \hat{P}(k \mid x)$。
- 二元分解多分类：通过锦标赛方式预测最终冠军类别。