机器学习理论简明手册

Theoretical Machine Learning:
A Handbook for Everyone

滕佳烨、王伯元、张臻

www.tengjiaye.com/mlbook
2025 年 4 月 15 日　　第一版

目录

序言

机器学习是一门极具实用价值的学科。尽管如此，许多人对机器学习的理论基础理解尚浅。事实上，许多本科生甚至没有机会接触到诸如凸优化、VC 维度等基础概念。由此，我写下这本书，旨在普及机器学习理论中的基本概念。

与常规的机器学习教材不同，本书并不以机器学习模型构建为核心，而是着重探讨这些模型背后的理论框架。因此，本书特别适合那些已经对机器学习模型有所了解，并对理论探索充满热情的读者。我们有意省略了模型构建的许多细节，转而专注于模型中与理论的关联部分。

为了让读者能更快地理解这些理论，我们尽量简化了底层逻辑的推导过程，例如一些复杂的数学推导，而采用更朴素的语言来厘清我们为什么要这么做。我们期望书中的每一句话都能让读者在阅读时产生“我们就该这么思考问题”的认同感。

此外，本书还试图回答一个核心问题：我们为何要学习这些理论。每个机器学习理论框架的提出都是与模型的特性紧密相关的。我们将从模型出发，探讨各种理论的实际应用和重要性。

在本书的撰写过程中，滕佳烨主要负责内容的编排与撰写，王伯元主要负责插图等的制作，张臻主要负责定理等的细节证明与检查核对。本书的电子版可见 `www.tengjiaye.com/mlbook`。

鉴于作者的知识和经验有限，书中内容可能存在不足之处。若有任何疏漏或错误，恳请读者不吝赐教，提出宝贵意见。

第一章　机器学习与机器学习理论

“ All learning is a kind of recollection of what we already know”

–Plato

1.1　学习与机器学习

1.1.1　学习

什么是学习?

当我们还是婴儿时，我们对世界一无所知，也无法判断猫和狗。为了识别这些动物，我们的家人反复向我们展示猫和狗的图片，帮助我们逐渐建立起对这些动物的认知。这种不断积累的过程，是学习。

当我们是学生时，我们要面临考试的压力。在考试前我们往往需要通过大量的练习题来巩固知识，总结自己曾经犯过的错误。这种通过实践和反思来提升自己的过程，也是学习。

通过这些例子可以看出，学习不单单是获取信息，更是一种将已有的经验（猫狗的图片、练习题）转化为知识（识别猫狗的能力、考试的能力）的过程。粗浅的来看，

学习是将过去的经验提炼为知识的过程。

1.1.2　机器学习

当学习的主体转变为机器时，我们称之为机器学习。在机器学习领域，“过去的经验” 指的是训练集 (Training Set)，而 “获得的知识” 则体现为学习到的模型 (Model)。

什么是模型？在机器学习中，我们经常面临的问题是如何找到一个函数 h，使得对于给定的输入 $\boldsymbol{x}$ （例如猫狗的图片），模型能够预测出相应的类别 y （例如猫狗的标签），即：

$$h(\boldsymbol{x}) \approx y.$$

这样的架构几乎能够涵盖世界上的大部分任务，而这里的函数 h 即是我们所说的模型。

如何找到这样的模型？我们首先需要拥有一组训练集，然后通过算法 (Algorithm) 来寻找合适的模型 h。算法的目标往往是：找到一个模型 h，使得对于训练集中的每一个样本 $(\boldsymbol{x}_i, y_i)$，都可以让 $h(\boldsymbol{x}_i)$ 足够接近 y_i，即

$$h(\boldsymbol{x}_i) \approx y_i.$$

可以看出，算法是一种将信息从训练集（过往的经验）提炼为函数（知识）的方法。

如何知道我们找到了好模型？由于训练集是有限的，即使我们满足了 $h(\boldsymbol{x}_i) \approx y_i$，在新的数据上模型有可能仍然无法满足 $h(\boldsymbol{x}) \approx y$。这类似于一个学生可能做了很多练习题，在所有考原题的测试中均表现优异，但在真正的高考中却无法发挥出应有的水平。因此，除了训练集之外，我们还需要使用一组独立的测试集 (Test Set) 来验证模型是否真的学到了知识。模型在训练集上表现良好，但在测试集上表现很差的情况，被称为过拟合 (Overfitting)。

总结来说，机器学习的目标是通过算法从过去的经验（即训练集）中提炼出知识（即模型），并且希望这个知识能够在新的数据集（测试集）上同样有效。

1.1.3 机器学习理论

然而，学到好的模型是十分困难的。因此机器学习理论的一个非常核心的问题是：在什么样的数据分布下（例如高斯分布），使用什么样的模型架构（例如神经网络），通过什么样的算法（例如随机梯度下降），能够让最终训练得到的模型满足

$$h(\boldsymbol{x}) \approx y.$$

此外，随着机器学习领域的不断发展，研究者们也开始在实践中关注模型的其他性质（例如鲁棒性 Robustness、公平性 Fairness），以及训练过程中可能出现的现象（例如 Edge of Stability, Scaling Law）。

1.2 常用术语与符号

数学符号。我们使用 $[n] = \{1, 2, \cdots, n\}$ 代表一个集合。我们使用 $\|\cdot\|$ 代表范数。一般来说，小写字母（如 a）代表一个标量，小写加粗字母（如 $\boldsymbol{a}$）代表一个向量，大写字母（如 A）代表一个矩阵，花体字母（如 $\mathcal{A}$）代表一个集合或分布。一般来说，带 $\hat{}$ 符号（如 $\hat{a}$）的参数均与训练后有关。

数据。在机器学习理论中，我们常常使用 $\boldsymbol{z} \triangleq (\boldsymbol{x}, y) \sim \mathcal{P} \subset \mathbb{R}^d \times \mathbb{R}$ 代表样本的特征 (feature) 与标签 (response) 对，其中 $\mathcal{P}$ 为数据服从的分布，d 为特征的维度。记训练集为 $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i\in[n]}$ 或 $\mathcal{D} = \{\boldsymbol{z}_i\}_{i\in[n]}$，这里 $\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)$ 代表第 i 个样本，n 代表数据集中的样本数量。记测试样本为 $\boldsymbol{z}' = (\boldsymbol{x}', y')$。上述的所有样本均为从分布 $\mathcal{P}$ 中独立同分布抽取。

为了表达方便，有时会将样本罗列为设计矩阵 $X \triangleq [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]^\top \in \mathbb{R}^{n\times d}$ 与响应向量 $Y \triangleq [y_1, y_2, \cdots, y_n] \in \mathbb{R}^n$。

模型。我们常常考虑由参数 $\boldsymbol{w} \in \mathbb{R}^p$ 对应的模型函数 $h_{\boldsymbol{w}}$，其中 p 代表模型中的参数个数。我们可以使用 $\hat{y} = h_{\boldsymbol{w}}(\boldsymbol{x})$ 代表模型的输出。注意到 $h_{\boldsymbol{w}}$ 有可能无法表达所有的函数，例如 $h_{\boldsymbol{w}}(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x}$ 无法表达非线性函数。这将会与我们后面的讨论有关。

损失函数。我们定义损失函数 $\ell(h_{\boldsymbol{w}}; \boldsymbol{z})$，其代表了模型预测 $h_{\boldsymbol{w}}(\boldsymbol{x})$ 和真实值 y 之间的距离，例如 $\ell(h_{\boldsymbol{w}}; \boldsymbol{z}) = (y - h_{\boldsymbol{w}}(\boldsymbol{x}))^2$。当上下文清晰时，我们也会将其表示为 $\ell(\boldsymbol{w}; \boldsymbol{z})$。据此，我们可以定义训练损失 $L_n(\boldsymbol{w}) \triangleq \frac{1}{n}\sum_{i\in[n]} \ell(\boldsymbol{w}; \boldsymbol{z}_i)$ 和测试损失 $L(\boldsymbol{w}) \triangleq \mathbb{E}_{\boldsymbol{z}'\sim\mathcal{P}} \ell(\boldsymbol{w}; \boldsymbol{z}')$。

机器学习理论的核心问题，就是希望证明在数据分布 $\mathcal{P}$ 下，对于模型 $h_{\boldsymbol{w}}$，利用算法找到的参数 $\hat{\boldsymbol{w}}$ 能够让测试误差近似达到最低

$$L(h_{\hat{\boldsymbol{w}}}) \approx \min_h L(h).$$

这里的最低误差 $\min_h L(h)$ 被称为贝叶斯最优误差 (Bayesian Optimal Error)。

1.3 No-Free-Lunch

在学习各种机器学习模型前，我们需要先回答这样一个问题：我们为什么要学习不同的模型架构？既然我们的目标就是找到一个合适的函数 h，我们为什么不直接学习这个函数 h，而是需要借助各式各样的模型架构（比如线性模型、神经网络）来学习这样的 h？

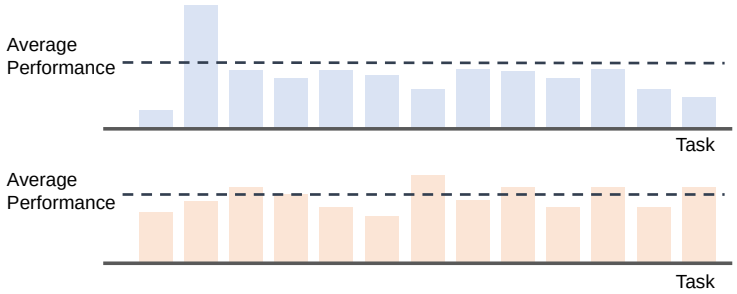


图 1.1: 不同模型架构在不同数据分布上的表现各异。蓝色模型在任务 2 表现突出，橙色模型在所有任务中接近平均性能。但我们无法声称蓝色与橙色模型谁是更好的模型。

实际上，这个问题有一个理论的回答。我们可以从理论上证明：不存在一个模型，使得它在任何任务上都是效果最好的模型。这个定理被称为 No-Free-Lunch 定理。

定理 1.1 (No-Free-Lunch Theorem)**.** *考虑分类任务 $y \in \{-1, 1\}$ 以及 0-1 损失函数 $\ell(h_{\boldsymbol{w}}; \boldsymbol{z}) = \mathbb{I}(y = h_{\boldsymbol{w}})$。假设 $\boldsymbol{x}$ 的定义域为 $\mathcal{X}$。则对于任意一个算法，若训练集样本数量 $m < \frac{1}{2}|\mathcal{X}|$，则一定存在一个数据分布，使得*

1. *存在一个函数 h，使得其在该数据分布上的测试误差达到零 $L(h) = 0$。*
2. *在该数据分布下，由算法得到的模型 $\hat{h}$ 的测试误差满足 $\mathbb{P}(L(\hat{h}) \geq 1/8) \geq 1/7$，该随机性来源于训练集的选取。*

根据 No-Free-Lunch 定理，算法失败的原因是我们没有对数据分布进行合理的假设。在现实中，人们往往使用模型结构隐式地对数据分布进行假设。例如，使用线性模型往往代表我们预先认为数据分布拥有类似的线性结构。总之，我们常常需要在一个假设类 (Hypothesis Class) 里寻找相应的模型，这里的假设类是一个函数族，例如线性函数族。

定理 1.1 的证明. 该证明的核心是：由于没有对数据分布的假设，任何算法均无法从训练集中获得未见过的测试集的信息。因此当训练集较小时，未见过的数据较多，算法只能随机猜测，从而一定有一个分布使得测试误差较大。

接下来我们从理论的角度严格刻画这一现象。简单起见，我们首先考虑数据集中只有两个样本 $|\mathcal{X}| = 2m = 2$，且训练集中只有一个样本 $|\mathcal{D}| = m = 1,$，并不妨设 $\mathcal{X} = \{a, b\}$。此时，我们考虑四个对称的两点分布[1]

$$\begin{aligned}
\mathcal{P}_1 &= \begin{cases} 1/2 & \text{if } x = a, y = -1 \\ 1/2 & \text{if } x = b, y = -1 \end{cases}, \qquad \mathcal{P}_2 = \begin{cases} 1/2 & \text{if } x = a, y = -1 \\ 1/2 & \text{if } x = b, y = +1 \end{cases}, \\
\mathcal{P}_3 &= \begin{cases} 1/2 & \text{if } x = a, y = +1 \\ 1/2 & \text{if } x = b, y = -1 \end{cases}, \qquad \mathcal{P}_4 = \begin{cases} 1/2 & \text{if } x = a, y = +1 \\ 1/2 & \text{if } x = b, y = +1 \end{cases},
\end{aligned} \tag{1.1}$$

由于训练集中只有一个数据点的信息，则算法对另一个点的判断完全是随机的。例如，若训练集是 $x = a, y = +1$, 则若算法判定 $x = b$ 时 $y = -1$，则会在 $\mathcal{P}_1, \mathcal{P}_3$ 上测试误差超过 $1/2$；反之，则会在 $\mathcal{P}_2, \mathcal{P}_4$ 上测试误差超过 $1/2$。综合来看，无论多么强大的算法，一定会在至少一半的分布上测试误差超过 $1/2$，因此，若记在 $\mathcal{D}$ 上训练的参数为 $\hat{\boldsymbol{w}}(\mathcal{D})$，在 $\mathcal{P}$ 上计算的测试误差记为 $L(\cdot; \mathcal{P})$。此时可得：

$$\mathbb{E}_{\mathcal{D}} \frac{1}{4} \sum_{i \in [4]} L(\hat{\boldsymbol{w}}(\mathcal{D}); \mathcal{P}_i) \geq \frac{1}{2} * \frac{1}{2} = \frac{1}{4}.$$

我们可以将其扩展到一般的结果中（情况类似，请读者自行证明!），注意到此时每个 $\boldsymbol{x}$ 均有 ± 1 两种取值，共 2^{2m} 种情况。由此当训练集内参数数量少于 $m/2$ 时：

$$\mathbb{E}_{\mathcal{D}} \frac{1}{2^{2m}} \sum_{i \in [2^{2m}]} L(\hat{\boldsymbol{w}}(\mathcal{D}); \mathcal{P}_i) \geq \frac{1}{4}.$$

由于期望（均值）一定小于最大值，因此至少存在一个分布 $\mathcal{P}_0$，使得

$$\mathbb{E}_{\mathcal{D}} L(\hat{\boldsymbol{w}}(\mathcal{D}); \mathcal{P}_0) \geq \frac{1}{4}.$$

利用一些代数的基本技巧[2]，可得以概率至少 $1/7$，有

$$\mathbb{P}(\mathbb{E}_{\mathcal{D}} L(\hat{\boldsymbol{w}}(\mathcal{D}); \mathcal{P}_0) \geq \frac{1}{8}) \geq 1/7.$$

□

[1] 这里有一个常见的直觉：由于我们对数据集没有任何假设，那么定义一些对称的分布往往能够帮助我们获得下界。

[2] 对任意随机变量 X，若 $0 < X < 1$，则有 $\mathbb{E}X \leq \mathbb{P}(X \geq 1/8) \times 1 + (1 - \mathbb{P}(X \geq 1/8)) \times 1/8$。证明该不等式的技巧是将期望写为积分形式。这里的常数 $1/8$ 可以自行选择。

1.4 机器学习理论摘要

我们的核心目标是证明训练出的模型 $h_{\hat{\boldsymbol{w}}}$ 满足

$$L(h_{\hat{\boldsymbol{w}}}) - \min_{h} L(h) \approx 0.$$

根据上一节的讨论，我们经常在一个函数族里训练机器学习模型，因此我们考虑这样的拆分[3]

$$\underbrace{[L(h_{\hat{\boldsymbol{w}}}) - \min_{h\in\mathcal{H}} L(h)]}_{\text{Excess Risk}} + \underbrace{[\min_{h\in\mathcal{H}} L(h) - \min_{h} L(h)]}_{\text{Approximation Error}} \approx 0.$$

在这里，第一项是我们在机器学习理论中经常需要处理的项：超额风险 (Excess Risk)，它衡量了一个模型和最好的模型相差多少。第二项是表达误差，代表了函数族 $\mathcal{H}$ 在多大程度上涵盖了最优的函数 $h_{\boldsymbol{w}^*}$。这里我们将 $\mathcal{H}$ 中达到最低测试误差的函数记为 $h_{\boldsymbol{w}^*}$，这常常是不唯一的。

接下来，我们考虑超额风险的分解：

$$L(\hat{\boldsymbol{w}}) - L(\boldsymbol{w}^*) = \underbrace{[L(\hat{\boldsymbol{w}}) - L_n(\hat{\boldsymbol{w}})]}_{\text{Generalization Gap}} + \underbrace{[L_n(\hat{\boldsymbol{w}}) - L_n(\boldsymbol{w}^*)]}_{\text{Training Error}} + \underbrace{L_n(\boldsymbol{w}^*) - L(\boldsymbol{w}^*)}_{\text{Small Term}}. \tag{1.2}$$

第一项 $L(\hat{\boldsymbol{w}}) - L_n(\hat{\boldsymbol{w}})$ 代表了模型的泛化误差，是一个我们经常处理的项。第二项 $L_n(\hat{\boldsymbol{w}}) - L_n(\boldsymbol{w}^*) \leq L_n(\hat{\boldsymbol{w}}) - \min_{\boldsymbol{w}} L_n(\boldsymbol{w})$ 和模型的训练误差有关。当 $\hat{\boldsymbol{w}}$ 能够最小化训练误差时（人们称这种情况为 ERM, Empirical Risk Minimization），这一项一定小于 0。第三项 $L_n(\boldsymbol{w}^*) - L(\boldsymbol{w}^*)$ 一般是一个很小的项，我们通常可以利用集中不等式（如 hoffding 不等式）证明。

根据上述表述，我们可以拆分机器学习理论为以下三个方面

1. 表达能力：$\min_{h\in\mathcal{H}} L(h) - \min_{h} L(h)$;
2. 优化能力：$L_n(\hat{\boldsymbol{w}}) - \min_{\boldsymbol{w}} L_n(\boldsymbol{w})$;
3. 泛化能力：$L(\hat{\boldsymbol{w}}) - L_n(\hat{\boldsymbol{w}})$.

机器学习理论的终极目标，就是证明上述三个界均约等于 0。

[3] 当定义超额风险时，我们往往需要定义是超过什么的风险。在本书中，我们默认超额风险定义在一个函数族的最优误差上，即；$L(h_{\hat{\boldsymbol{w}}}) - \min_{h\in\mathcal{H}} L(h)$；也有其他教材将超额风险定义在贝叶斯最优误差上，即 $L(h_{\hat{\boldsymbol{w}}}) - \min_{h} L(h)$。

第二章　k 近邻算法与维度灾难

“ Tell me with whom you associate, and I will tell you who you are ”
–Goethe

2.1　k 近邻算法

背景知识：
A : Das ist eine deutsche Redewendung.　B : 日本語のフレーズだ。
C : Это русская фраза.　D : C'est une expression française.
问题：请问E属于哪种语言？
E : これは何語ですか?

请观察上述四种语言。请问语句 E 属于哪种语言?

相信大部分读者并没有系统学习过这四种语言。然而，绝大多数人都能正确选择 B 选项。其原因是 E 中的文字与 B 更为接近。在现实中，人们常常这样思考。当遇到一个新问题时，我们可以将其与曾经见过的问题进行比对并找出相似案例，从而得出新问题的判断。这就是 K 近邻算法（K-NN, K-nearest neighbor）的核心思想。

我们将上面的想法正式地表述出来。在 K-NN 中，我们首先从训练集中找到 K 个与测试点最近的样本，这些样本的标签的众数即为我们的预测结果。其算法可以概括为算法1。

确定 K 值，即选择的最近邻居的数量。计算测试点与训练集中每个点之间的距离。找出距离测试点最近的 K 个训练样本。根据这 K 个样本的标签，确定测试点的预测标签。

在 K-NN 中，还有两个关键问题需要考虑:

Algorithm 1 K 近邻算法

输入： 训练数据集 $\{\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)\}_{i\in[n]}$，测试点 $\boldsymbol{x}'$，近邻数 K，数据间的度量函数 $D(\cdot,\cdot) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$;

1: 对每一个数据点 $\boldsymbol{z}_i$，计算 $D_i = D(\boldsymbol{x}', \boldsymbol{x}_i)$;

2: 找到距离 $\boldsymbol{z}'$ 最近的 K 个样本点 $\mathcal{I} = \{i : \sum_{j\in[n]} \mathbb{I}(D_i < D_j) \le K\}$;[1]

输出： $y_i, i \in \mathcal{I}$ 的众数，即 $\hat{y} = \arg\max_{c_i} \sum_{i\in\mathcal{I}} \mathbb{I}(y_i = c_i)$。

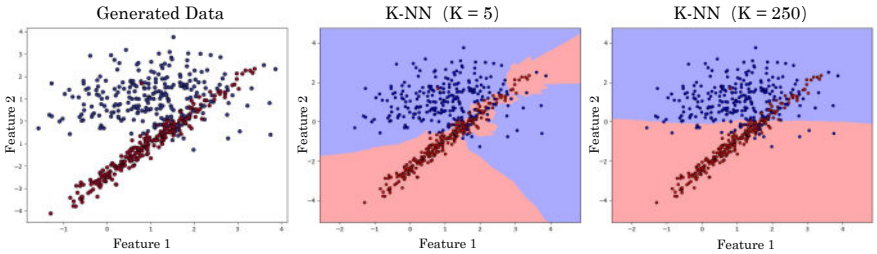


图 2.1: 在 K-NN 算法中，随着超参数 K 的增加，决策边界趋向平滑与稳定

Q1. 如何选择最优的超参数 K：一个最简单直接的办法是使用交叉验证 (cross-validation)。一般来说，较小的 K 会使得决策边界更加陡峭，而较大的 K 会使得决策边界更加平滑。

Q2. 如何判断“最近”：对于不同的数据分布，我们有时会选用不同的度量函数来判断“最近”。最常用的度量函数即为欧氏距离，也有马氏距离、曼哈顿距离、切比雪夫距离等其他距离度量方法。我们将在拓展内容中介绍不同距离的计算方法。

2.2 维度灾难

接下来，我们考虑 K-NN 的算法性质。为了简单起见，我们接下来只考虑 $K = 1$ 的情况。当 $K = 1$ 时，对于任意的测试样本，我们只需要考虑训练集中与其最近的样本。在训练样本足够大时，测试样本与训练集中最近的样本可以足够接近。因此，只要测试样本不在决策边界且样本无噪声，K-NN 就是一个完美分类器。基于这样的直觉，我们可以得到以下定理。

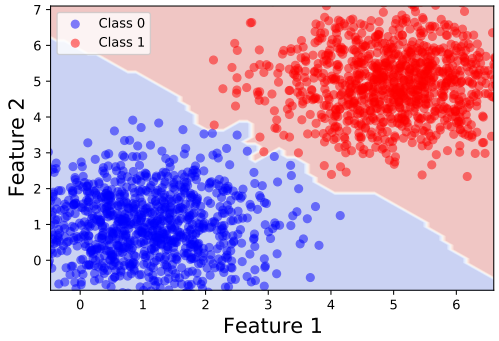


图 2.2: 样本不在决策边界且无噪的情况下，当训练样本充足时，KNN 可以达到完美分类器

定理 2.1 (K-NN 是完美分类器). *假设数据无噪，即 $y = f(\boldsymbol{x})$ 是一个确定性函数。此时，对于任意不在决策 δ-边界上[a]的样本 $(\boldsymbol{x}', y')$，其对应 1-NN 的输出 $\hat{y}$ 一定满足*

$$\lim_{n\to\infty} \mathbb{P}(\hat{y} = y') = 1.$$

[a]点 $(\boldsymbol{x}', y')$ 不在决策 δ-边界上，即满足以下两个条件的点：(a) 对于任意 $(\boldsymbol{x}_0, y_0)$，如果 $D(\boldsymbol{x}_0, \boldsymbol{x}') \leq \delta$，一定满足 $y_0 = y'$；(b) 存在 $\epsilon > 0$ 使得 $\mathbb{P}(\boldsymbol{x}_0 : D(\boldsymbol{x}_0, \boldsymbol{x}') \leq \delta) > \epsilon$。

定理 2.1的证明. 根据上述不在 δ-决策边界的定义，对于测试点 $(\boldsymbol{x}', y')$，一定存在一个集合 $S = \{\boldsymbol{x} : \|\boldsymbol{x} - \boldsymbol{x}'\| \leq \delta\}$ 使得：
(a) 任取 $\boldsymbol{x}_0 \in S$, 若其在样本的可行域内，其对应的标签 $y_0 = y'$；
(b) 该集合在特征空间中非空，即存在 $\epsilon > 0$ 使得 $\mathbb{P}(S) > \epsilon$。
此时，根据 1-NN 的算法，若在训练集中存在 S 中的点 $\boldsymbol{x}_0$，则其输出 $\hat{y}$ 一定为 $y_0 = y'$。因此，模型预测正确的概率大于在训练集中取得这样的 $\boldsymbol{x}_0$ 的概率

$$\mathbb{P}(\hat{y} = y') \geq 1 - (1 - \mathbb{P}(\boldsymbol{x} : \|\boldsymbol{x}_0 - \boldsymbol{x}'\| \leq \delta))^n = 1 - (1 - \epsilon)^n.$$

当 $n \to \infty$ 时，此概率趋向于 1。 □

从上面的定理，我们可能会认为 K-NN 已经是一个完美分类器了。然而，事实并非如此。从上述推导可以看出，完美分类的前提是训练集足够大，以至于能够覆盖整个特征空间，这在高维空间中是非常困难的，因为随

着维度的增加，所需的样本数量呈指数级增长。这被称为维度灾难 (curse of dimension)。

定理 2.2 (K-NN 与维度灾难)**.** *存在至少一个分布，使得对任意 K，K-NN 达到 ϵ 误差时所需的样本至少为*

$$\Omega(\epsilon^{-d}).$$

这里，d 表示特征空间的维度。这个公式说明了维度对于所需样本数量的影响。随着维度的增加，需要大量的样本来保证分类的准确性。这就是所谓的维度灾难，即在高维空间中，分类器的性能会因为样本数量的不足而急剧下降。维度灾难时机器学习中常见的问题，我们可以从两个角度理解 K-NN 的维度灾难。

*1. 高维空间距离函数的维度灾难。*在 d 维空间（$d \to \infty$）中，定义标准正方体为边长是 1 的正方体，其体积为 1。与之对比，边长为 0.9 的正方体体积为 $0.9^d \to 0$，而边长为 1.1 的正方体体积为 $1.1^d \to \infty$。如果我们希望用边长为 ϵ 的小正方体[2]充满这个正方体，我们需要的小正方体数量至少是 $(1/\epsilon^d)$ 个。换个角度来说，如果我们在高维空间中随机取点，其分布常常极其稀疏。为了尽量填满整个空间，我们需要的样本点数至少是指数多个 $\Omega(1/\epsilon^d)$。

*2. K-NN 未对分布产生任何限制。*注意到 K-NN 实际上可以拟合任意数据分布，其符合 No-Free-Lunch 定理的适用条件。实际上，在没有限制数据分布的情况下，任何分类器均会导致维度灾难。因此，我们需要通过限制模型复杂度（例如在一个函数族内考虑模型）等方式隐式地限制数据分布，详细内容可以参考上一章的 PAC-Learning 框架。更多的理论分析可以参考 Shalev-Shwartz[1](263 页)。

[2]这能保证在 L_∞ 度量下每个正方体内的点都是距离较近的点。

第三章　决策树与模型表达能力

“ Decision is a collection of rules ”

3.1　决策树算法

我们再来看一下第二章的引入例子。

背景知识：

A : Das ist eine deutsche Redewendung.　B : 日本語のフレーズだ。

C : Это русская фраза.　D : C'est une expression française.

问题：请问E属于哪种语言？

E : これは何語ですか?

请观察上述四种语言。请问语句 E 属于哪种语言?

在第二章中，我们使用了 K-NN 算法成功找到了接近的语言。一些读者可能会采用不同的策略。首先，我们可以归纳四个选项的语言特点。例如，A 选项全部使用了英文字母，B 选项的文字有很多横平竖直的特点，D 选项在文字内有一个’ 符号，如果没有上述特征，则只能为 C 选项。此时，我们再看 E 选项的文字，就能够看出其拥有和 B 选项类似的特点。因此我们选择 B 选项。

在上述思考过程中，我们在推断过程并没有重新参考训练集，通过在训练集中归纳出的特征，使用特征进行 if-then 判断。这种模型被称为基于规则的模型 (rule-based model)。然而，单纯的一条规则在现实生活中往往是不够的。因此，我们可以使用一棵树的结构来整理这些规则。这就是决策树 (Decision Tree)。

决策树的主要构成包括结点和有向边。每一个有向边代表一个决策规则，最终样本从最上方的根结点依照规则被分类到最终的叶结点上。对于每

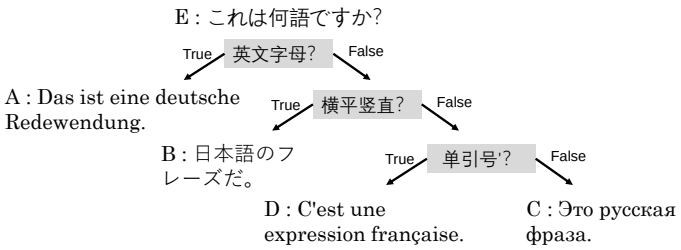


图 3.1: 利用决策树判断语句 E 属于哪种语言

一个叶结点而言，其均代表了一个分类的 if-then 判断。因此决策树可以被视为是 rule-based model 的集合。

用决策树做推断是相对容易的，我们只需要根据有向边的规则对每一个测试样本进行判断，并最终输送到叶结点上。考虑这个例子（图3.2）：假设我们是一位骨科医生，拥有一万个历史患者的病历信息。通过决策树，我们可以快速通过已有规则判断新患者是否存在脊柱后凸的风险。与之相比，如果使用 K-NN 算法，每当有新患者来访，我们都需要遍历所有一万名患者，计算并存储距离。因此决策树往往比 K-NN 具有更短的推断时间。

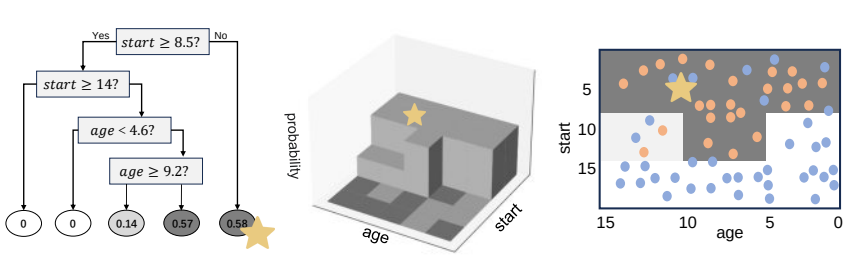


图 3.2: 利用决策树快速推断患者是否具有脊椎后凸风险

然而，如何根据训练集生成一棵决策树？一般来说，人们通常使用贪婪算法 (greedy)。在贪婪算法中，我们从只有一个根结点的决策树出发。每一步我们只生成一个叶子结点。最终获得一棵完整的树。在这个流程中，我们应该考虑以下问题

Q1: 选择什么样的特征生成叶子结点：生成叶子结点的目的是为了在推断时对所给出的标签拥有更高的确定性，也就是选取不确定性更小（增益更大）的特征划分叶子结点。熵、错误分类率、基尼系数是常用的

Algorithm 2 决策树算法 DT($\mathcal{Z}$)

输入： 训练数据集 $\mathcal{Z} = \{(x_i, y_i)\}_{i\in[n]}$，增益度量函数 $\mathcal{G}(\cdot)$;

if 当前节点所有训练集样本属于同一类别 y_0 **then**
  return 叶子结点类别 y_0;
else
  通过增益度量函数 $\mathcal{G}(\cdot)$ 选取增益最大的特征 F;
  并将数据集按特征 F 的不同取值分为 $\mathcal{Z}_1$ 与 $\mathcal{Z}_2$;
  构建子树 $T_1 = \mathrm{DT}(\mathcal{Z}_1)$ 与子树 $T_2 = \mathrm{DT}(\mathcal{Z}_2)$;

输出： 一棵完整的分类决策树，包含每个决策节点以及最终叶子结点类别 y_0。

度量指标。

Q2: 何时终止生成叶子结点：我们期望在叶子结点中消除不确定性，因此叶子节点中相同标签的样本越多越好。叶子结点中所有样本具有相同标签时，这便是最简单的终止条件。

Q3: 应用何种剪枝策略：剪枝的根本目的是为了减少模型的复杂性，减少过拟合。预剪枝和后剪枝是两种常见的剪枝策略。

决策树的具体流程可被总结在算法2中。

3.2 模型表达能力

我们永远无法让一个只会中文的人独自写出阿拉伯语。正如我们很难让一个模型拟合世界上的所有函数。那么，对于决策树而言，它究竟能表达什么样的函数呢?

我们接下来介绍一个定理，说明当我们不限制决策树的大小时，决策树能够近似表达任意 Lipschitz 函数。

定理 3.1 (决策树的表达能力)**.** *对任意的 L-Lipshitz 且定义域有界的函数 $f(\cdot)$ 和给定的 ϵ，均存在一个决策树 $Tree(\cdot)$，使得对任意给定 $\boldsymbol{x}$，满足*

$$|f(\boldsymbol{x}) - Tree(\boldsymbol{x})| \leq \epsilon.$$

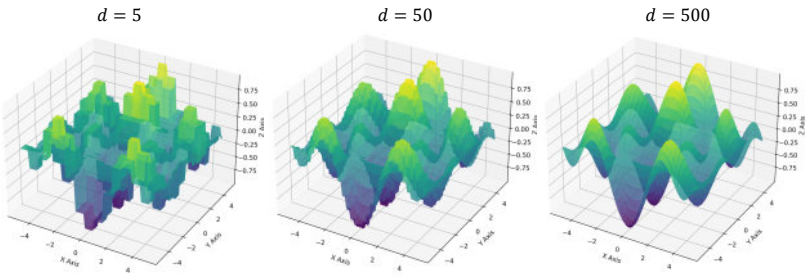


图 3.3: 决策树能够轻松分割任意随机数据

这种研究模型自身能够近似拟合何种函数族的性质，被称作模型的表达能力（expressivity/representation）。研究表达能力是非常重要的，因为表达能力欠缺的模型常常会出现欠拟合的问题。例如，线性模型在表达能力方面往往是不足的，而神经网络则具备相对较好的表达能力。一个著名的结论（一致逼近定理）是：在一定条件下，无限宽的两层神经网络可以逼近任意光滑函数。在这类研究工作当中，我们通常会关注模型的规模大小（比如决策树里结点的数量）与表达误差（如定理中所涉及的 ϵ）之间的关系。

定理*3.1*的证明. 我们首先注意到：根据决策树的决策形式，函数 $\mathrm{Tree}(\cdot)$ 是一个分段常数函数，即满足形式

$$\mathrm{Tree}(\boldsymbol{x}) = y_i, \text{ if } \boldsymbol{x} \in [a_1^l, a_1^u] \times [a_2^l, a_2^u] \times \cdots \times [a_d^l, a_d^u],$$

这里 $\boldsymbol{x} \in \mathbb{R}^d$. 接下来，我们证明任意这样分段常数函数均可由决策树进行表达。如图3.3所示，对于任意分段常数函数，我们均可以构造一棵决策树，其中每一层为一个特征，每层中的每个结点均代表了该特征的一个分割点。

我们可以证明任意 Lipschitz 函数可以由这样分段常数函数进行近似。对任意 L-Lipschitz 函数，设其每个维度上的定义域界为 $[-M, M]$。我们将其定义域每隔 ϵ/L 进行分离，则共有 $(2ML/\epsilon)^d$ 个分割点，且每个分割点内任意两点 $\boldsymbol{x}, \boldsymbol{x}'$ 的距离[1]满足

$$\|\boldsymbol{x}' - \boldsymbol{x}\|_\infty \leq \epsilon/L.$$

[1]注意到这里我们使用了 ℓ_∞ 范数。不同范数间存在着一个转化关系，因此我们后文不会特别突出是几范数。

接下来我们考虑任给的一个分割点，若设该分段常数函数在分割点间的值为 $\text{Tree} = f(\boldsymbol{x}')$，其中 $\boldsymbol{x}'$ 为分割点间给定的任意一点，则对任意的分割点间的点 $\boldsymbol{x}$，均有

$$|\text{Tree} - f(\boldsymbol{x})| = |f(\boldsymbol{x}') - f(\boldsymbol{x})| \leq L\|\boldsymbol{x}' - \boldsymbol{x}\| \leq \epsilon.$$

因此，对任意给定的 $\boldsymbol{x}$，均有

$$|\text{Tree}(\boldsymbol{x}) - f(\boldsymbol{x})| \leq \epsilon.$$

□

第四章　线性函数与凸优化

“ The non-linear world is reluctantly understood through linear models ”

4.1　线性回归

正如我们在第一节介绍的那样，机器学习的一个重要目标是寻找一个函数 f，使得 $f(\boldsymbol{x}) \approx y$。然而，直接描述任意一个函数 f 是十分困难的，这通常需要无穷个样本点才能实现。因此，在机器学习中，人们通常只考虑一组函数的集合，即函数族 (function class)，并使用参数 (parameter) 来表达函数族中的每一个函数。

线性函数族是机器学习中最基本的函数族之一，可以表示为：

$$\mathcal{F} = \{f(\boldsymbol{x}) = \boldsymbol{w}^\top \boldsymbol{x} : \boldsymbol{w} \in \mathbb{R}^d\},$$

其中 $\boldsymbol{w}$ 是一个 d 维的参数向量，它能够代表函数族中的线性函数 $\boldsymbol{w}^\top \boldsymbol{x}$。由此，原有的寻找映射 f 的任务转化为寻找最优的参数 $\boldsymbol{w}$，使得 $\boldsymbol{w}^\top \boldsymbol{x} \approx y$。

在线性函数族中寻找最优参数，最常见的模型即为线性回归，这时我们使用均方误差（Mean Squared Error, MSE）作为损失函数，其定义为：

$$\mathrm{MSE}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \boldsymbol{w}^\top \boldsymbol{x}_i)^2 = \frac{1}{n}\|Y - X\boldsymbol{w}\|^2,$$

其中 n 代表样本数量，$(\boldsymbol{x}_i, y_i)$ 代表第 i 个样本。为简化符号，我们重新记 $X = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_n]^\top \in \mathbb{R}^{n\times d}$ 与 $Y = [y_1, y_2, \cdots, y_n] \in \mathbb{R}^n$。通过最小化这个损失函数，我们能够得到训练集上的最优参数 $\hat{\boldsymbol{w}}$。

$$\hat{\boldsymbol{w}} = \left(XX^\top\right)^{-1} X^\top Y = \left(\sum_{i\in[n]} \boldsymbol{x}_i \boldsymbol{x}_i^\top\right)^{-1} \sum_{i\in[n]} \boldsymbol{x}_i y_i.$$

证明. 考虑 MSE 损失的导数可以被写为

$$\nabla_{\boldsymbol{w}}\mathrm{MSE}(\boldsymbol{w}) = -\frac{1}{n}X^\top(Y - X\boldsymbol{w}) = -\frac{1}{n}(XY - X^\top X\boldsymbol{w}).$$

令导数为 0，可解得

$$\hat{\boldsymbol{w}} = (XX^\top)^{-1}X^\top Y.$$

□

4.2 逻辑斯蒂回归

在线性回归里，我们重点关注的是回归任务。而在分类任务中的线性模型即为逻辑斯蒂回归 (logistic regression)。

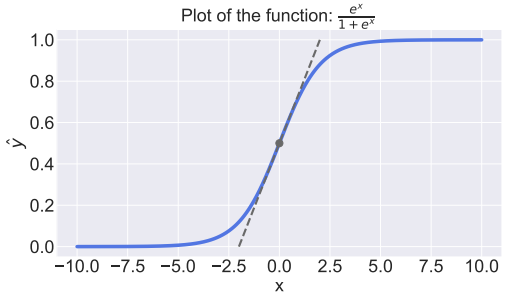


图 4.1: 当 $\hat{y} \approx 0.5$ 时，损失的变化相对更加剧烈

在逻辑斯蒂回归中，考虑分类数据 $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \{0, 1\}$。此时，由于标签只有 0 1 两个取值，MSE 损失函数不再合理，举例来说，$\hat{y} = 0.6$ 与 $\hat{y} = 0.8$ 差别很小，而与 $\hat{y} = 0.4$ 差别很大。因此，我们考虑损失函数 cross-entropy

$$\mathrm{CE}(\boldsymbol{w}) = y \log \hat{y} + (1 - y) \log(1 - \hat{y}),$$

这里 $\hat{y} = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x})}{1+\exp(\boldsymbol{w}^\top \boldsymbol{x})} \in (0, 1)$。如图4.1所示，在 $\boldsymbol{w}^\top \boldsymbol{x} \approx 0$，即 $\hat{y} \approx 0.5$ 时，损失的变化相对更加剧烈。

然而，和线性回归不同的是，逻辑斯蒂回归无法获得关于 $\hat{\boldsymbol{w}}$ 的显式解。因此，我们需要使用梯度下降的方法寻找 $\hat{\boldsymbol{w}}$。

4.3 梯度下降

在逻辑斯蒂回归中，我们希望能够找到一个合适的 $\hat{\boldsymbol{w}}$ 最小化 CE($\boldsymbol{w}$)。我们将这个问题抽象成下列优化问题

$$\min_{\boldsymbol{w}\in\mathbb{R}^d} g(\boldsymbol{w}).$$

我们想象 $\boldsymbol{w}$ 是一个地图上的二维经纬度，而 $g(\boldsymbol{w})$ 是对应的三维高度。因此，可以近似想想成一个山峰，而我们的目标则是找到高度最低的地方。就像图4.2展示的那样。然而，我们如何能够找到下山的途径?

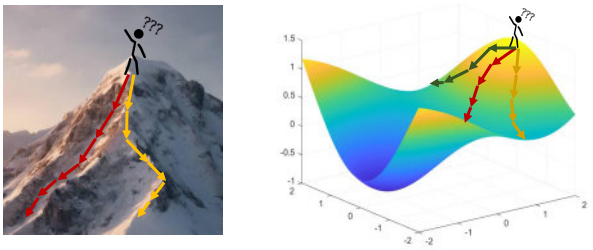


图 4.2: 当身处局部一点，我们无法直接看到全局最优路径

首先，我们很难获得整个山峰的情况（这可能需要指数数量级的 $\boldsymbol{w}$ 和对应的 $g(\boldsymbol{w})$ 取值），因此我们选择使用函数的局部信息。同时，一下子找到最优解是十分困难的，因此我们选择使用迭代算法。此时，一个常见的策略是贪婪算法。既然我们无法保证整个路径最优，我们便退而求其次，希望至少在每一步都是较优的，也就是我们希望每一步都是在下山。如何做到这件事呢? 我们可以用函数的梯度（导数）信息来实现。正如图4.3展示的那样，在梯度为正的位置，我们希望能向左走，在梯度为负的位置，我们希望能向右走。

因此，梯度下降算法（Gradient Descent, GD）应运而生。在 GD 里，我们首先随机初始化一个参数 $\boldsymbol{w}[0]$，而后使用迭代

$$\boldsymbol{w}[t+1] = \boldsymbol{w}[t] - \eta_t \nabla g(\boldsymbol{w}[t]), \tag{4.1}$$

这里 η_t 代表学习率，代表每一次迭代的强度。

接下来我们证明 GD 的有效性。我们首先关注下面的图4.4，函数二收敛到最低点仅需 32 步，函数一收敛到最低点需要 315 步。相比之下，函数

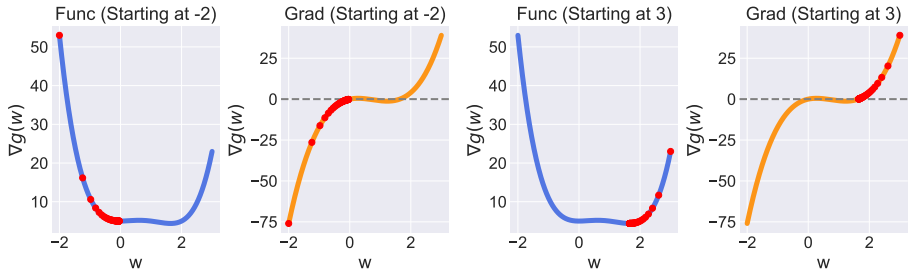


图 4.3: 梯度的正负影响了 GD 算法叠代的方向

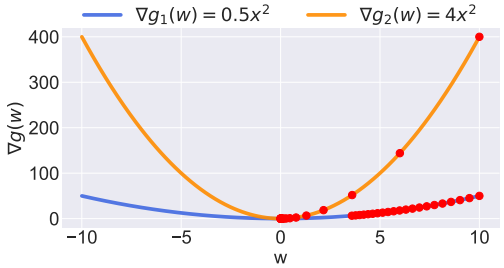


图 4.4: 在相同初始点和学习率下，更平滑的函数通常能更快达到最低点

二 GD 可以很好的优化，而函数一就比较难。这其中的区别在于函数的光滑性 (smoothness)。

定义 4.1 (光滑性, Smooth)**.** *函数 g 被称为 M-smooth 或 M-光滑，如果对任意 $\boldsymbol{w} \in \mathbb{R}^d$, 其满足*

$$\nabla^2 g(w) \preceq MI.$$

紧接着，定理4.1将会证明：在原函数 g 是光滑且学习率不高的情况下，只要梯度不为 0 ，则 GD 的每一步都能够降低 $g(\boldsymbol{w})$。

定理 4.1 (GD 与光滑性)**.** *当 $g(\boldsymbol{w})$ 是 M-光滑时，如果学习率满足 $\eta_t \leq 1/M$，则 GD 满足*

$$g(\boldsymbol{w}[t+1]) \leq g(\boldsymbol{w}[t]) - \frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2.$$

定理4.1的证明. 对 $g(\cdot)$ 在 $\boldsymbol{w}[t]$ 处做泰勒展开，可得

$$g(\boldsymbol{w}[t+1]) \leq g(\boldsymbol{w}[t]) + \nabla^{\top} g(\boldsymbol{w}[t])(\boldsymbol{w}[t+1] - \boldsymbol{w}[t]) + \frac{M}{2}\|\boldsymbol{w}[t+1] - \boldsymbol{w}[t]\|^2.$$

代入迭代式(4.1), 可得

$$\begin{aligned} g(\boldsymbol{w}[t+1]) &\leq g(\boldsymbol{w}[t]) - \eta_t \|\nabla g(\boldsymbol{w}[t])\|^2 + \frac{M\eta_t^2}{2}\|\nabla g(\boldsymbol{w}[t])\|^2 \\ &= g(\boldsymbol{w}[t]) - \eta_t \left(1 - \frac{M\eta_t}{2}\right) \|\nabla g(\boldsymbol{w}[t])\|^2. \end{aligned}$$

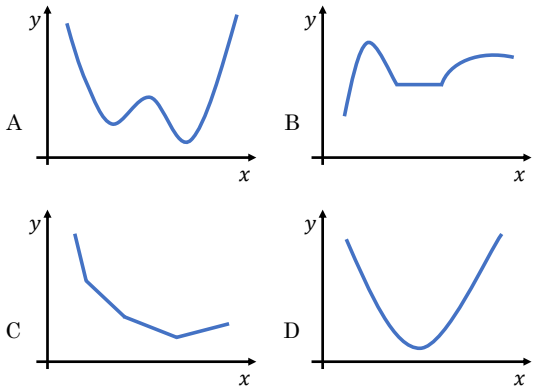
因此，当 $\eta_t \leq 1/M$，可得

$$g(\boldsymbol{w}[t+1]) \leq g(\boldsymbol{w}[t]) - \frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2.$$

事实上，当 $\eta_t < 2/M$ 时，GD 的每一步对函数均有提升。但为了简便我们只考虑了 $\eta_t \leq 1/M$ 的情况。 □

从定理4.1可知,GD 的每一步都会让原函数 $g(\boldsymbol{w})$ 至少降低 $\frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2$。上述定理保证了在导数不为 0 的时候，通过选取合适的学习率，GD 在每一步都能降低函数值。然而，这最多能够说明在收敛情况下 GD 会收敛到梯度为 0 的地方。如何保证 GD 能够收敛到训练误差最小的解？我们接下来会利用凸优化的相关知识回答这个问题。

4.4 凸训练与梯度下降



观察下列图像，哪个能被 GD 优化到全局最优点?

显然，如果使用 GD，A,B 曲线都有可能卡在中间。而曲线 C 当学习率选取合适时一定可以收敛到全局最优，就像我们随机放一个玻璃球它一定可以滚到最低点。这种函数即为凸函数 (convex)。我们给出严格的凸函数定义。

定义 4.2 (凸函数与强凸函数)**.** *函数 g 被称为凸函数，如果对任意 $\boldsymbol{w} \in \mathbb{R}^d$, 其满足*

$$\nabla^2 g(\boldsymbol{w}) \succeq 0$$

这里我们用 $A \succeq B$ 指代 $A - B$ 矩阵半负定。此外，函数 g 被称为 μ-强凸函数，如果对任意 $\boldsymbol{w} \in \mathbb{R}^d$, 其满足

$$\nabla^2 g(\boldsymbol{w}) \succeq \mu I.$$

在凸函数里，梯度为 0 的解一定是全局最优解。因此我们可以证明，在凸函数（定理4.2）或强凸函数（定理4.3）条件下，GD 能够到达函数最小值。

定理 4.2 (GD 与凸函数)**.** *当 $g(\boldsymbol{w})$ 是 M-光滑且凸时，如果学习率满足 $\eta_t < 1/M$，则*

$$g(\boldsymbol{w}[T]) - g(\hat{\boldsymbol{w}}^*) \leq \frac{\|\boldsymbol{w}[0] - \boldsymbol{w}^*\|^2}{\sum_{t\in[T]} 2\eta_{t-1}}.$$

因此，当 $\eta_t = \eta$ 为常数时，GD 的收敛速率[a]为 $\frac{1}{t}$。

[a]一般情况下我们考虑 $\|\boldsymbol{w}[0] - \boldsymbol{w}^*\|^2$ 为一个常数。

定理4.2的证明. 记函数的全局最小值点为 $\hat{\boldsymbol{w}}^*$。根据凸函数的定义和泰勒展开，我们可知：

$$g(\hat{\boldsymbol{w}}^*) \geq g(\boldsymbol{w}[t]) + \nabla^\top g(\boldsymbol{w}[t])(\hat{\boldsymbol{w}}^* - \boldsymbol{w}[t]).$$

带入迭代式4.1并消掉梯度项，可得

$$g(\hat{\boldsymbol{w}}^*) \geq g(\boldsymbol{w}[t]) - \frac{1}{\eta_t}(\boldsymbol{w}[t+1] - \boldsymbol{w}[t])^\top(\hat{\boldsymbol{w}}^* - \boldsymbol{w}[t]). \tag{4.2}$$

注意到这里我们将导数代为参数。同样的操作我们可以处理定理4.1的结论，

可得

$$g(\boldsymbol{w}[t+1]) \leq g(\boldsymbol{w}[t]) - \frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2 = g(\boldsymbol{w}[t]) - \frac{1}{2\eta_t}\|\boldsymbol{w}[t+1] - \boldsymbol{w}[t]\|^2. \tag{4.3}$$

将式(4.2)中的 $g(\boldsymbol{w}[t])$ 带入式(4.3)的结果，可得

$$\begin{aligned}
g(\boldsymbol{w}[t+1]) \leq & g(\hat{\boldsymbol{w}}^*) + \frac{1}{\eta_t}(\boldsymbol{w}[t+1] - \boldsymbol{w}[t])^\top(\hat{\boldsymbol{w}}^* - \boldsymbol{w}[t]) - \frac{1}{2\eta_t}\|\boldsymbol{w}[t+1] - \boldsymbol{w}[t]\|^2 \\
= & g(\hat{\boldsymbol{w}}^*) + \frac{1}{2\eta_t}(\boldsymbol{w}[t] - \boldsymbol{w}[t+1])^\top(\boldsymbol{w}[t] + \boldsymbol{w}[t+1] - 2\hat{\boldsymbol{w}}^*) \\
= & g(\hat{\boldsymbol{w}}^*) + \frac{1}{2\eta_t}((\boldsymbol{w}[t] - \boldsymbol{w}^*) - (\boldsymbol{w}[t+1] - \boldsymbol{w}^*))^\top((\boldsymbol{w}[t] - \hat{\boldsymbol{w}}^*) + (\boldsymbol{w}[t+1] - \hat{\boldsymbol{w}}^*)) \\
= & g(\hat{\boldsymbol{w}}^*) + \frac{1}{2\eta_t}(\|\boldsymbol{w}[t] - \boldsymbol{w}^*\|^2 - \|\boldsymbol{w}[t+1] - \boldsymbol{w}^*\|^2),
\end{aligned}$$

将上式变形后按 t 累加，可得（这一步一般被称为 telescoping）

$$\sum_{t\in[T]} 2\eta_{t-1}(g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*)) \leq (\|\boldsymbol{w}[0] - \boldsymbol{w}^*\|^2 - \|\boldsymbol{w}[T+1] - \boldsymbol{w}^*\|^2) \leq \|\boldsymbol{w}[0] - \boldsymbol{w}^*\|^2.$$

同时，根据光滑性性质，$g(\boldsymbol{w}[t])$ 的值单调递减，即 $g(\boldsymbol{w}[T]) \leq g(\boldsymbol{w}[T-1]) \leq \cdots \leq g(\boldsymbol{w}[0])$，因此

$$g(\boldsymbol{w}[T]) - g(\hat{\boldsymbol{w}}^*) \leq \frac{\sum_{t\in[T]} 2\eta_{t-1}(g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*))}{\sum_{t\in[T]} 2\eta_{t-1}} \leq \frac{\|\boldsymbol{w}[0] - \boldsymbol{w}^*\|^2}{\sum_{t\in[T]} 2\eta_{t-1}}.$$

□

定理 4.3 (GD 与强凸函数)**.** 当 $g(\boldsymbol{w})$ 是 M-光滑且 μ-强凸时，如果学习率满足 $\eta_t < 1/M$，则

$$g(\boldsymbol{w}[T]) - g(\hat{\boldsymbol{w}}^*) \leq \exp\left(-\mu \sum_{t\in[T]} \eta_{t-1}\right)(g(\boldsymbol{w}[0]) - g(\hat{\boldsymbol{w}}^*)).$$

定理4.3的证明. 为了得到和最优值的距离，我们需要获得 $g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*)$ 的迭代式。首先，回忆光滑性的结论

$$\begin{aligned}
& g(\boldsymbol{w}[t+1]) \leq g(\boldsymbol{w}[t]) - \frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2. \\
\Leftrightarrow & g(\boldsymbol{w}[t+1]) - g(\hat{\boldsymbol{w}}^*) \leq g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*) - \frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2.
\end{aligned}$$

为了得到 $g(\boldsymbol{w}[t])-g(\hat{\boldsymbol{w}}^*)$ 的迭代式，我们需要用强凸性质处理 $\|\nabla g(\boldsymbol{w}[t])\|^2$。利用强凸性，将函数 $g(\hat{\boldsymbol{w}}^*)$ 在 $\boldsymbol{w}[t]$ 处展开

$$g(\hat{\boldsymbol{w}}^*) \geq g(\boldsymbol{w}[t]) + \nabla^\top g(\boldsymbol{w}[t])(\hat{\boldsymbol{w}}^* - \boldsymbol{w}[t]) + \frac{\mu}{2}\|\hat{\boldsymbol{w}}^* - \boldsymbol{w}[t]\|^2.$$

注意在需要处理的 $\|\nabla g(\boldsymbol{w}[t])\|^2$ 中并没有出现 $\hat{\boldsymbol{w}}^*$ 项。因此我们考虑将右侧的 $\hat{\boldsymbol{w}}^*$ 项进行放缩。注意到右侧对 $\hat{\boldsymbol{w}}^*$ 是一个二次函数，且二次函数的最小值取在 $\hat{\boldsymbol{w}}_0^* = \boldsymbol{w}[t] - \frac{1}{\mu}\nabla g(\boldsymbol{w}[t])$ 处，因此

$$\begin{aligned} g(\hat{\boldsymbol{w}}^*) &\geq g(\boldsymbol{w}[t]) + \nabla^\top g(\boldsymbol{w}[t])(\hat{\boldsymbol{w}}_0^* - \boldsymbol{w}[t]) + \frac{\mu}{2}\|\hat{\boldsymbol{w}}_0^* - \boldsymbol{w}[t]\|^2 \\ &= g(\boldsymbol{w}[t]) - \frac{1}{2\mu}\|\nabla g(\boldsymbol{w}[t])\|^2. \end{aligned} \tag{4.4}$$

将这里得到的 $\|\nabla g(\boldsymbol{w}[t])\|^2$ 项带入上式，可得

$$\begin{aligned} g(\boldsymbol{w}[t+1]) - g(\hat{\boldsymbol{w}}^*) \leq& g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*) - \frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2 \\ \leq& g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*) - \frac{\eta_t}{\mu}(g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*)) \\ =& (1-\eta_t\mu)(g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*)). \end{aligned}$$

进一步的，将上式进行迭代，并由 $\exp(-x) \geq 1 - x$ 可得

$$g(\boldsymbol{w}[T]) - g(\hat{\boldsymbol{w}}^*) \leq \exp\left(-\mu \sum_{t\in[T]} \eta_{t-1}\right)(g(\boldsymbol{w}[0]) - g(\hat{\boldsymbol{w}}^*)).$$

□

4.5 随机梯度下降

根据 GD 的迭代 $\boldsymbol{w}[t+1] = \boldsymbol{w}[t] - \eta_t \nabla g(\boldsymbol{w}[t])$，带入机器学习的损失函数 $g(\boldsymbol{w}[t]) = \frac{1}{n}\sum_{i=1}^n \ell(y_i, f_{\boldsymbol{w}}(\boldsymbol{x}_i))$ 可得

$$\boldsymbol{w}[t+1] = \boldsymbol{w}[t] - \eta_t \frac{1}{n}\sum_{i=1}^n \nabla \ell(y_i, f_{\boldsymbol{w}}(\boldsymbol{x}_i)).$$

注意到在 GD 里，我们需要对每一个样本的损失函数进行求导。然而，真实的训练环境中，我们经常面临着计算资源有限的问题。而计算导数往往又是开销很大的步骤之一。因此，我们接下来要介绍随机梯度下降（Stochastic Gradient Descent, SGD），它能够极大降低 GD 中的计算难度。

大数定律告诉我们，通过样本均值可以估计总体均值。如果我们把 $\sum_{i=1}^{n}\nabla\ell(y,f_{\boldsymbol{w}}(\boldsymbol{x}))$ 用其中的一项去替代，其仍然是无偏的。从这个直觉出发，在 SGD 中，我们不再在每一步中计算每一个样本的导数，而是在每一步中只计算一个或多个样本的导数，这被总结为如下迭代

$$\boldsymbol{w}[t+1]=\boldsymbol{w}[t]-\eta_t\frac{1}{n}\sum_{i\in\mathcal{I}_t}\nabla\ell(y_i,f_{\boldsymbol{w}}(\boldsymbol{x}_i)),$$

其中，$\mathcal{I}_t$ 为每一次 SGD 抽取的样本标号，其可能是一个样本，也可能是一组样本。当 $\mathcal{I}_t=[n]$ 时，SGD 等价于 GD。

在优化的理论分析中，我们常常将这个随机梯度建模成真实梯度加上一个随机噪声，即

$$\boldsymbol{w}[t+1]=\boldsymbol{w}[t]-\eta_t(\nabla g(\boldsymbol{w}[t])+\xi_t),$$

其中 ξ_t 是满足期望 $\mathbb{E}\xi_t=0$，方差 $\mathbb{V}(\xi_t)\le\sigma^2$ 的独立随机噪声。在 SGD 中，我们面临着这样一个 tradeoff：当 $|\mathcal{I}|$ 较大，则占用资源变多，$\mathcal{I}$ 较小，则梯度噪声过大，训练不够稳定。在实践中可以通过交叉验证确定 $|\mathcal{I}|$ 的大小。类似于 GD，我们仍然可以在定理4.4中得到在光滑条件下每一步的提升

定理 4.4 (SGD 与光滑性)**.** *当 $g(\boldsymbol{w})$ 是 M-光滑时，如果学习率满足 $\eta_t\le 1/M$，则 SGD 满足*

$$\mathbb{E}g(\boldsymbol{w}[t+1])\le g(\boldsymbol{w}[t])-\frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2+\frac{\eta_t^2M}{2}\sigma^2,$$

这里的期望加在梯度噪声 ξ_t 上。

和 GD 比较，SGD 多了一项由噪声带来的损失 $\frac{\eta_t}{2}\sigma^2$。

定理4.4的证明. 整体证明流程非常类似于 GD 的部分。首先，对 $g(\cdot)$ 在 $\boldsymbol{w}[t]$ 处做泰勒展开，可得

$$g(\boldsymbol{w}[t+1])\le g(\boldsymbol{w}[t])+\nabla^\top g(\boldsymbol{w}[t])(\boldsymbol{w}[t+1]-\boldsymbol{w}[t])+\frac{M}{2}\|\boldsymbol{w}[t+1]-\boldsymbol{w}[t]\|^2.$$

代入迭代式并取期望 $\mathbb{E}\xi_i=0,\mathbb{V}\xi_i=\sigma^2$，可得

$$\mathbb{E}g(\boldsymbol{w}[t+1])\le g(\boldsymbol{w}[t])-\eta_t(1-\frac{M\eta_t}{2})\|\nabla g(\boldsymbol{w}[t])\|^2+\frac{\eta_t^2M}{2}\sigma^2.$$

当 $\eta_t \leq 1/M$，可得

$$\mathbb{E}g(\boldsymbol{w}[t+1]) \leq g(\boldsymbol{w}[t]) - \frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2 + \frac{\eta_t^2 M}{2}\sigma^2.$$

□

4.6 凸训练与 SGD

类似于 GD，我们可以得到如下定理4.5来关注凸训练与 SGD 的结果。

定理 4.5 (SGD 与凸函数)**.** 当 $g(\boldsymbol{w})$ 是 M-光滑且凸时，如果学习率满足 $\eta_t < 1/M$，则

$$g(\bar{\boldsymbol{w}}[T]) - g(\hat{\boldsymbol{w}}^*) \leq \frac{\|\boldsymbol{w}[0] - \boldsymbol{w}^*\|^2}{\sum_{t\in[T]} 2\eta_t} + \frac{\sum_{t\in[T]} \eta_t^2\sigma^2}{\sum_{t\in[T]} \eta_t}.$$

因此，当 $\eta_t = \eta$ 为常数时，

$$g(\bar{\boldsymbol{w}}[T]) - g(\hat{\boldsymbol{w}}^*) \leq \frac{\|\boldsymbol{w}[0] - \boldsymbol{w}^*\|^2}{2\eta T} + \eta\sigma^2.$$

通过令 $\eta = \frac{1}{\sqrt{T}\sigma}$，其收敛率约为 $\mathcal{O}(1/\sqrt{T})$.

定理4.5的证明. 与定理4.2的证明类似。首先我们处理凸性质部分，注意到（类似于式(4.2)）

$$\begin{aligned} g(\hat{\boldsymbol{w}}^*) &\geq g(\boldsymbol{w}[t]) + \nabla^\top g(\boldsymbol{w}[t])(\hat{\boldsymbol{w}}^* - \boldsymbol{w}[t]) \\ &= g(\boldsymbol{w}[t]) - \frac{1}{\eta_t}\mathbb{E}(\boldsymbol{w}[t+1] - \boldsymbol{w}[t])^\top(\hat{\boldsymbol{w}}^* - \boldsymbol{w}[t]), \end{aligned} \tag{4.5}$$

此时的期望取在梯度噪声 ξ_t 上。我们其次关注光滑性质部分。首先注意到，根据梯度噪声的定义，可得 $\mathbb{E}\|\boldsymbol{w}[t+1] - \boldsymbol{w}[t]\|^2 \leq \eta_t^2\|\nabla g(\boldsymbol{w}[t])\|^2 + \eta_t^2\sigma^2$。而后，与式(4.3)类似，通过上式在定理4.4中消去梯度可得（注意 $\eta_t \leq 1/M$）

$$\begin{aligned} \mathbb{E}g(\boldsymbol{w}[t+1]) &\leq g(\boldsymbol{w}[t]) - \frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2 + \frac{\eta_t^2 M}{2}\sigma^2 \\ &\leq g(\boldsymbol{w}[t]) - \frac{\eta_t}{2}\frac{\mathbb{E}\|\boldsymbol{w}[t+1] - \boldsymbol{w}[t]\|^2 - \eta_t^2\sigma^2}{\eta_t^2} + \frac{\eta_t}{2}\sigma^2 \\ &= g(\boldsymbol{w}[t]) - \frac{1}{2\eta_t}\mathbb{E}\|\boldsymbol{w}[t+1] - \boldsymbol{w}[t]\|^2 + \eta_t\sigma^2. \end{aligned} \tag{4.6}$$

因此，类似于凸函数的证明部分，式(4.5)和式(4.3)合并并消去 $g(\boldsymbol{w}[t])$ 可得

$$\begin{aligned}
&\mathbb{E}g(\boldsymbol{w}[t+1]) - g(\hat{\boldsymbol{w}}^*) \\
\leq& g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*) - \frac{1}{2\eta_t}\mathbb{E}\|\boldsymbol{w}[t+1] - \boldsymbol{w}[t]\|^2 + \eta_t\sigma^2 \\
\leq& \frac{1}{\eta_t}\mathbb{E}(\boldsymbol{w}[t+1] - \boldsymbol{w}[t])^\top(\hat{\boldsymbol{w}}^* - \boldsymbol{w}[t]) - \frac{1}{2\eta_t}\mathbb{E}\|\boldsymbol{w}[t+1] - \boldsymbol{w}[t]\|^2 + \eta_t\sigma^2 \\
\leq& \frac{1}{2\eta_t}\mathbb{E}(\|\boldsymbol{w}[t] - \boldsymbol{w}^*\|^2 - \|\boldsymbol{w}[t+1] - \boldsymbol{w}^*\|^2) + \eta_t\sigma^2.
\end{aligned}$$

通过 telescoping 累加，可得

$$\begin{aligned}
\sum_{t\in[T]} 2\eta_{t-1}\mathbb{E}(g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*)) \leq& \mathbb{E}(\|\boldsymbol{w}[0] - \boldsymbol{w}^*\|^2 - \|\boldsymbol{w}[T+1] - \boldsymbol{w}^*\|^2) + \sum_{t\in[T]} 2\eta_t^2\sigma^2 \\
\leq& \mathbb{E}\|\boldsymbol{w}[0] - \boldsymbol{w}^*\|^2 + \sum_{t\in[T]} 2\eta_t^2\sigma^2,
\end{aligned}$$

此时的期望取在整个算法上。令 $\bar{\boldsymbol{w}}[T] = \frac{\sum_{t\in[T]}\eta_{t-1}\boldsymbol{w}[t]}{\sum_{t\in[T]}\eta_{t-1}}$ 为优化轨迹的加权平均，根据 Jensen 不等式，

$$\begin{aligned}
\mathbb{E}g(\bar{\boldsymbol{w}}[T]) - g(\hat{\boldsymbol{w}}^*) =& \mathbb{E}g(\frac{\sum_{t\in[T]}\eta_{t-1}\boldsymbol{w}[t]}{\sum_{t\in[T]}\eta_{t-1}}) - g(\hat{\boldsymbol{w}}^*) \\
\leq& \frac{1}{\sum_{t\in[T]}\eta_{t-1}}\sum_{t\in[T]}\eta_t\mathbb{E}(g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*)) \\
\leq& \frac{1}{2\sum_{t\in[T]}\eta_{t-1}}\mathbb{E}(\|\boldsymbol{w}[0] - \boldsymbol{w}^*\|^2 + \sum_{t\in[T]} 2\eta_t^2\sigma^2) \\
=& \frac{1}{2\sum_{t\in[T]}\eta_{t-1}}\mathbb{E}\|\boldsymbol{w}[0] - \boldsymbol{w}^*\|^2 + \frac{\sum_{t\in[T]}\eta_t^2\sigma^2}{\sum_{t\in[T]}\eta_t}.
\end{aligned}$$

□

注意到对于 SGD 而言，由于每一步的更新含有噪声，最后一个迭代 $\boldsymbol{w}[t]$ 往往是不稳定的。因此我们在 SGD 中往往更加关注某个加权平均如 $\bar{\boldsymbol{w}}[t]$ 的收敛。类似于之前的技巧，我们也可以获得强凸背景下的 SGD 收敛结果（定理4.6）。

定理 4.6 (SGD 与强凸函数). 当 $g(\boldsymbol{w})$ 是 M-光滑且 μ-强凸时，如果常数学习率满足 $\eta_t = \eta < 1/M$，则

$$\mathbb{E}g(\boldsymbol{w}[T]) - g(\hat{\boldsymbol{w}}^*) \leq \exp\left(-\eta T/\mu\right) \mathbb{E}(g(\boldsymbol{w}[0]) - g(\hat{\boldsymbol{w}}^*)) + \frac{\eta}{2}\frac{M}{\mu}\sigma^2.$$

此时，若选取 $\eta = \frac{\log T}{T}$，则收敛率为 $\mathcal{O}(\log T/T)$。

定理4.6的证明. 证明流程类似于定理4.3。首先，回忆 SGD 的光滑性得到的结论（定理4.4）

$$\begin{aligned}
&\mathbb{E}g(\boldsymbol{w}[t+1]) \leq g(\boldsymbol{w}[t]) - \frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2 + \frac{\eta_t^2 M}{2}\sigma^2. \\
\Leftrightarrow &\mathbb{E}g(\boldsymbol{w}[t+1]) - g(\hat{\boldsymbol{w}}^*) \leq g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*) - \frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2 + \frac{\eta_t^2 M}{2}\sigma^2.
\end{aligned} \tag{4.7}$$

类似的，我们可以使用强凸函数的性质（式(4.4)），即

$$g(\hat{\boldsymbol{w}}^*) \geq g(\boldsymbol{w}[t]) - \frac{1}{2\mu}\|\nabla g(\boldsymbol{w}[t])\|^2. \tag{4.8}$$

将式(4.7)与式(4.8) 带入，并消去梯度项可得：

$$\begin{aligned}
\mathbb{E}g(\boldsymbol{w}[t+1]) - g(\hat{\boldsymbol{w}}^*) \leq & g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*) - \frac{\eta_t}{2}\|\nabla g(\boldsymbol{w}[t])\|^2 + \frac{\eta_t^2 M}{2}\sigma^2 \\
\leq & g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*) - \frac{\eta_t}{\mu}(g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*)) + \frac{\eta_t^2 M}{2}\sigma^2 \\
= & (1 - \eta_t\mu)(g(\boldsymbol{w}[t]) - g(\hat{\boldsymbol{w}}^*)) + \frac{\eta_t^2 M}{2}\sigma^2.
\end{aligned}$$

我们进一步将上式进行迭代。为了简单起见，使用常数步长 $\eta_t = \eta$ 可得

$$\mathbb{E}g(\boldsymbol{w}[T]) - g(\hat{\boldsymbol{w}}^*) \leq \exp\left(-\eta T/\mu\right) (g(\boldsymbol{w}[0]) - g(\hat{\boldsymbol{w}}^*)) + \frac{\eta}{2}\frac{M}{\mu}\sigma^2.$$

此时，选取 $\eta = o_T(1)$ 即可得到收敛速率。

此外，若选取 $\eta_t = \Theta(1/t)$ 还能进一步得到更优的 $\mathcal{O}(1/T)$ 的收敛结果。 □

第五章　感知机模型与对偶

“万物皆有两端，独中又自有对”

–朱熹

5.1　感知机模型

在上一章中，我们介绍了逻辑斯蒂回归（Logistic Regression）来处理分类任务。逻辑斯蒂回归的输出格式为

$$\hat{y} = \frac{\exp(\boldsymbol{w}^\top \boldsymbol{x})}{1 + \exp(\boldsymbol{w}^\top \boldsymbol{x})} \in (0, 1),$$

并且我们需要使用 cross-entropy loss。在本章中，我们仍然考虑分类任务，但将从另一个视角来提出不一样的输出格式与损失函数。

首先注意到，在分类任务中，线性函数的分类依据一般等价于

$$\hat{y} = \begin{cases} +1, & \boldsymbol{w}^\top \boldsymbol{x} \geq 0 \\ -1, & \boldsymbol{w}^\top \boldsymbol{x} < 0 \end{cases} \tag{5.1}$$

其中，模型的标签为 $y \in \{-1, +1\}$，对应预测为 $\hat{y} \in \{-1, +1\}$（注意到这里等价于 $y \in \{0, 1\}$）。上述的逻辑斯蒂回归中，以 0.5 为阈值进行二值化后等价于上述分类依据。作为分类依据的核心，我们希望能够直接围绕着 $\boldsymbol{w}^\top \boldsymbol{x}$ 构建一个损失函数。一个自然的想法是：当 $y = +1$，我们希望 $\boldsymbol{w}^\top \boldsymbol{x}$ 越大越好，反之亦然。因此，我们可以构建如下的损失函数：

$$\mathrm{Lin}(\boldsymbol{w}) = \max\{-y\boldsymbol{w}^\top \boldsymbol{x}, 0\}.$$

我们可以使用 SGD 对该损失函数进行训练，即获得了感知机模型的优

化算法。注意到上述损失函数的导数为

$$\nabla \mathrm{Lin}(\boldsymbol{w}) = \begin{cases} 0, & y\boldsymbol{w}^\top \boldsymbol{x} \geq 0 \\ -y\boldsymbol{x}, & y\boldsymbol{w}^\top \boldsymbol{x} < 0 \end{cases} \tag{5.2}$$

因此我们可以得到其 SGD 训练

Algorithm 3 感知机模型的训练

输入： 训练数据集 $\{\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)\}_{i\in[n]}$，学习率 η，训练步数 T;

1: 随机选取初值 $\boldsymbol{w}_0$;
2: **for** $t \in [T]$ **do**
3: 在数据集中任选数据点 $(\boldsymbol{x}_i, y_i)$;
4: **if** $y_i \boldsymbol{w}[t]^\top \boldsymbol{x}_i < 0$ **then**
5: $\boldsymbol{w}[t+1] = \boldsymbol{w}[t] + \eta y_i \boldsymbol{x}_i$;

输出： $\boldsymbol{w}[T]$。

遗憾的是，感知机的训练并不总是收敛的。也就是说，即使训练了很多步，$\boldsymbol{w}[T-1]$ 与 $\boldsymbol{w}[T]$ 的值可能仍然相差很大。接下来我们考虑这样一个例子：

Example 5.1.1 (感知机训练不收敛)**.** *考虑一个二维平面上的数据集 $\mathcal{D}$，其中正样本包括 $(1,1),(-1,-1)$，而负样本包括 $(1,-1),(-1,1)$。此时，对任意 T，均有*

$$\mathbb{E}\|\boldsymbol{w}[T] - \boldsymbol{w}[T-1]\| \geq \frac{\sqrt{2}}{4}\eta, \tag{5.3}$$

其中 $\mathbb{E}$ 来自 SGD 算法的随机性。此外，可以进一步证明 $\lim_{T\to\infty} \boldsymbol{w}[T]$ 不存在。因此，感知机训练无法收敛。

例5.1.1的证明十分简单。由于原始样本是非线性可分的，因此在每一步里都有至少 1/4 的概率存在错误样本，且更新强度为 $\|\eta y_i \boldsymbol{x}_i\| = \sqrt{2}\eta$。此外，假设当前有三个点分类正确，则算法3会不断在分类错误的点上进行迭代，并最终使得该点分类正确。而由于数据集非线性可分，这又会使得另一个点被分类错误，从而可知 $\lim_{T\to\infty} \boldsymbol{w}[T]$ 不存在。我们在图5.1可视化了这个例子。基于此，什么时候感知机的训练可以收敛呢？一个自然的想法是，由于感知机是个线性模型，如果数据集线性可分，也许其能够收敛。现实情况也确实是这样的。定理5.1证明了当数据集线性可分时，感知机能够确保收敛。

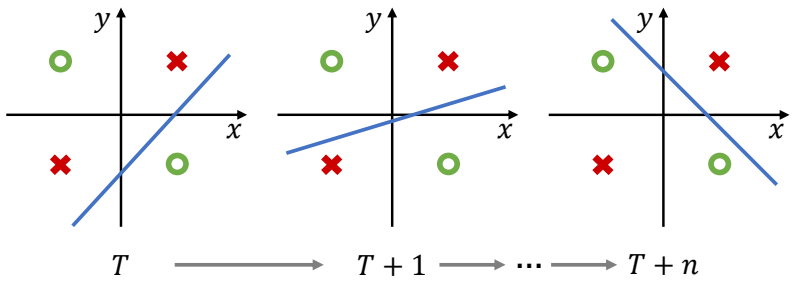


图 5.1: 在线性不可分的数据集上，感知机模型难以收敛

定理 5.1 (感知机收敛性). 设训练数据集 $\mathcal{D} = \{(\boldsymbol{x}_i, y_i)\}_{i\in[n]}$ γ-线性可分的，即存在归一化的 $\boldsymbol{w}^*$ 使得（此时 $\|\boldsymbol{w}^*\| = 1$）

$$y_i \boldsymbol{x}_i^\top \boldsymbol{w}^* \geq \gamma. \tag{5.4}$$

此时，若 $\|\boldsymbol{x}_i\| \leq R$ 且初始点 $\boldsymbol{w}_0 = \boldsymbol{0}$，则算法3的误分类次数 K 一定满足（这里的误分类次数指算法3中第 4-5 步的发生次数）

$$K \leq (R/\gamma)^2.$$

定理5.1的证明. 我们首先关注 $\boldsymbol{w}[T(K)]$ 与 $\boldsymbol{w}^*$ 的接近程度，这里我们用 $T(K)$ 代表遇到 K 个误分样本时的迭代次数。这一般可以用其内积进行表达。根据算法3的迭代式，有

$$\begin{aligned} &\boldsymbol{w}^\top[T(K)]\boldsymbol{w}^* \\ =&[\boldsymbol{w}^\top[T(K-1)] + \eta y_i \boldsymbol{x}_i^\top]\boldsymbol{w}^* \\ \geq&\boldsymbol{w}^\top[T(K-1)]\boldsymbol{w}^* + \eta\gamma \\ \geq&K\eta\gamma. \end{aligned} \tag{5.5}$$

一般来说我们更加关注归一化后的内积，因此需要继续关注 $\|\boldsymbol{w}[T(K)]\|$。注

意到

$$\begin{aligned}
&\|\boldsymbol{w}[T(K)]\|^2 \\
=&[\boldsymbol{w}[T(K-1)]+\eta y_i \boldsymbol{x}_i]^2 \\
=&\|\boldsymbol{w}[T(K-1)]\|^2+\eta^2\|\boldsymbol{x}_i\|^2+\eta y_i \boldsymbol{x}_i^\top \boldsymbol{w}[T(K-1)] \\
\leq&\|\boldsymbol{w}[T(K-1)]\|^2+\eta^2 R^2 \\
\leq&K\eta^2 R^2,
\end{aligned} \tag{5.6}$$

这里我们用到了样本被误分的信息，即 $y_i \boldsymbol{x}_i^\top \boldsymbol{w}[T(K-1)] \leq 0$。因此，我们有

$$K\eta\gamma \leq \boldsymbol{w}^\top[T(K)]\boldsymbol{w}^* \leq \|\boldsymbol{w}^\top[T(K)]\|\|\boldsymbol{w}^*\| \leq \sqrt{K\eta^2 R^2}. \tag{5.7}$$

因此误分次数存在上界

$$K \leq (R/\gamma)^2. \tag{5.8}$$

□

5.2 感知机模型的对偶

通过观察算法3的内容可知，如果算法从 $\boldsymbol{w}[0]=\boldsymbol{0}$ 开始训练，由于每一步的迭代都是在参数上增加 $y_i \boldsymbol{x}_i$，则最终得到的参数一定满足其线性组合

$$\boldsymbol{w}[T] = \sum_{i\in[n]} \alpha_i y_i \boldsymbol{x}_i. \tag{5.9}$$

其中 α_i 为线性组合的系数。因此，在训练时训练 $\boldsymbol{\alpha} \in \mathbb{R}^n$ 等同于训练 $\boldsymbol{w} \in \mathbb{R}^d$。如何训练 $\boldsymbol{\alpha}$？我们再返回原有的算法3中。可以发现，只有当 $y_i \boldsymbol{w}^\top \boldsymbol{x}_i < 0$ 时，$\boldsymbol{w}$ 进行一次更新，且更新幅度为 $\eta y_i \boldsymbol{x}_i$。在训练 $\boldsymbol{\alpha}$ 时，这代表了对 α_i 进行一次强度为 η 的更新。基于此，我们可以将原始的感知机算法3等价表达为算法4。事实上，这就是感知机模型的对偶训练。

算法4与算法3的唯一区别即是迭代，原始的算法3是在 $\boldsymbol{w}$ 上进行迭代，而算法4则是在 $\boldsymbol{\alpha}$ 上进行迭代。根据上面的讨论，这二者是等价的。当然，我们也可以利用式(5.9)修改损失函数搭配 SGD 进行推导。其对应的损失函数可表达为

$$\mathrm{Lin}(\boldsymbol{\alpha}) = \max\{-y(\sum_{i\in[n]} \alpha_i y_i \boldsymbol{x}_i)^\top \boldsymbol{x}, 0\}.$$

Algorithm 4 感知机模型训练的对偶表达

输入： 训练数据集 $\{\boldsymbol{z}_i = (\boldsymbol{x}_i, y_i)\}_{i\in[n]}$，学习率 η，训练步数 T;

选取初值 $\boldsymbol{\alpha}_0 = 0$;

for $t \in [T]$ **do**

  在数据集中任选数据点 $(\boldsymbol{x}_i, y_i)$;

  if $y_i \boldsymbol{w}^\top \boldsymbol{x}_i < 0$ **then**

    $\alpha_i[i+1] = \alpha_i[i] + \eta$;

输出： $\boldsymbol{w}[T] = \sum_{i\in[n]} \alpha_i[T] y_i \boldsymbol{x}_i$.

直接利用上式进行 SGD，这和算法4有所不同。然而，由于形式上的便利性，人们一般倾向于选择 $\alpha_i[i+1] = \alpha_i[i] + \eta$ 这样的迭代。

实际上，我们还可以从另一个角度查看 $\boldsymbol{w}[T] = \sum_{i\in[n]} \alpha_i y_i \boldsymbol{x}_i$ 这个形式。我们可以将原问题重新写为

$$\min_{\boldsymbol{w}} \max_{\lambda\geq 0} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i\in[n]} \lambda_i (y_i \boldsymbol{w}^\top \boldsymbol{x}_i).$$

注意到，由于 $\lambda \geq 0$，解内方程一定需要 $y_i \boldsymbol{w}^\top \boldsymbol{x}_i \geq 0$。然而，由于 $\boldsymbol{w}$ 的大小在分类任务中并不重要（方向才是关键），因此此时的 $\boldsymbol{w}$ 会有无数个解。为了最终能够产生唯一解，我们选择最小化 $\|\boldsymbol{w}\|$，即得到了上面的式子。交换 min 和 max 的顺序，可得

$$\max_{\lambda\geq 0} \min_{\boldsymbol{w}} \frac{1}{2}\|\boldsymbol{w}\|^2 + \frac{1}{n}\sum_{i\in[n]} \lambda_i (y_i \boldsymbol{w}^\top \boldsymbol{x}_i).$$

解内循环，可得最优 $\boldsymbol{w} = \sum_{i\in[n]} -\frac{\lambda_i}{n} y_i \boldsymbol{x}_i$，即为式(5.9). 这一切都发生的刚刚好！然而，为什么我们可以交换顺序？为什么获得的 $\boldsymbol{w}$ 新形式刚好和我们的观察一致？这些并不是巧合，我们接下来会介绍对偶相关的知识。

5.3 对偶

在优化问题中，我们常常需要考虑满足特定约束的优化：

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & f(\boldsymbol{x}) \\ \text{Subject to} \quad & g_i(\boldsymbol{x}) \leq 0, i \in [I] \\ & h_j(\boldsymbol{x}) = 0, j \in [J] \end{aligned} \tag{5.10}$$

对于这样的问题，我们通常使用拉格朗日函数来近似求解。

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = f(\boldsymbol{x}) + \sum_{i\in[I]} \lambda_i g_i(\boldsymbol{x}) + \sum_{j\in[J]} \nu_i h_j(\boldsymbol{x}). \tag{5.11}$$

这里，我们将 λ_i, ν_i 称为拉格朗日乘子。此时，原问题可被转为：

$$\min_{\boldsymbol{x}} \max_{\lambda\geq 0,\nu} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \min_{\boldsymbol{x}} \max_{\lambda\geq 0,\nu} \{f(\boldsymbol{x}) + \sum_{i\in[I]} \lambda_i g_i(\boldsymbol{x}) + \sum_{j\in[J]} \nu_i h_j(\boldsymbol{x})\}. \tag{5.12}$$

其最优值记为 M_p^*. 注意到，这两个问题是等价的。当存在某个 $g_i(\boldsymbol{x}) > 0$ 或 $h_j(\boldsymbol{x}) \neq 0$，内问题的最优解一定是取 $\lambda \to \infty$ 或 $\nu \to \infty$，从而内函数趋于无穷。而满足约束 $g_i(\boldsymbol{x}) \leq 0$ 及 $h_j(\boldsymbol{x}) = 0$，则应取 $\lambda = 0$。从而问题等价于 $\min_{\boldsymbol{x}} f(\boldsymbol{x})$。

其对偶问题形式为

$$\max_{\lambda\geq 0,\nu} \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \max_{\lambda\geq 0,\nu} \min_{\boldsymbol{x}} \{f(\boldsymbol{x}) + \sum_{i\in[I]} \lambda_i g_i(\boldsymbol{x}) + \sum_{j\in J} \nu_i h_j(\boldsymbol{x})\}. \tag{5.13}$$

其最优值记为 M_d^*.

一个自然的关系是：

定理 5.2 (弱对偶定理)**.** 对于原问题的最优值 M_p^* 与对偶问题的最优值 M_d^*，一定成立

$$M_d^* \leq M_p^*. \tag{5.14}$$

定理5.2的证明. 首先，我们一定有对任意 $\boldsymbol{x}, \lambda, \nu$，

$$L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq \max_{\lambda\geq 0,\nu} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}).$$

对等式两侧取 $\min_{\boldsymbol{x}}$，可得对任意 λ, ν，

$$\min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq \min_{\boldsymbol{x}} \max_{\lambda\geq 0,\nu} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = M_p^*.$$

再对等式两侧取 $\max_{\lambda\geq 0,\nu}$，可得

$$M_d^* = \max_{\lambda\geq 0,\nu} \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) \leq \min_{\boldsymbol{x}} \max_{\lambda\geq 0,\nu} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = M_p^*.$$

□

注意到，两个式子实际上只交换了 max 和 min 的顺序，那么是否有可能 $M_d^* = M_p^*$ 成立呢？这就是强对偶定理。事实上，强对偶定理有诸多版本，例如使用 Slater 条件。我们接下来证明一个稍微简单的线性版本的强对偶定理。

定理 5.3 (强对偶定理)**.** *考虑线性规划问题*

$$\begin{aligned} \min_{\boldsymbol{x}} \quad & \boldsymbol{c}^\top \boldsymbol{x} \\ \textit{Subject to} \quad & A\boldsymbol{x} = \boldsymbol{b}. \\ & \boldsymbol{x} \succeq 0. \end{aligned} \tag{5.15}$$

其中 $\boldsymbol{c}, \boldsymbol{b} \in \mathbb{R}^d, A \in \mathbb{R}^{J \times d}$ 为问题常数。其对偶问题可表达为

$$\begin{aligned} \min_{\boldsymbol{\nu}} \quad & \boldsymbol{b}^\top \boldsymbol{\nu} \\ \textit{Subject to} \quad & \boldsymbol{c} + A^\top \boldsymbol{\nu} \succeq 0. \end{aligned} \tag{5.16}$$

记原问题的最优值为 M_p^*，对偶问题的最优值为 M_d^*，则一定成立

$$M_d^* = M_p^*.$$

定理5.3证明了在线性规划问题下强对偶定理的成立。事实上，当 f, g_i 均为凸函数，且 h_j 为线性函数时，强对偶定理往往成立（但并不总是成立）。具体的条件，我们需要使用 Slater 条件（一种强对偶定理的充分条件）。

定理5.3的证明. 考虑原函数的 Lagrange 方程[1]

$$\min_{\boldsymbol{x}} \max_{\lambda \preceq \mathbf{0}, \boldsymbol{\nu}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{\lambda}^\top \boldsymbol{x} + \boldsymbol{\nu}^\top (A\boldsymbol{x} - \boldsymbol{b}).$$

其对应的对偶问题为

$$\max_{\lambda \preceq \mathbf{0}, \boldsymbol{\nu}} \min_{\boldsymbol{x}} L(\boldsymbol{x}, \boldsymbol{\lambda}, \boldsymbol{\nu}) = \boldsymbol{c}^\top \boldsymbol{x} + \boldsymbol{\lambda}^\top \boldsymbol{x} + \boldsymbol{\nu}^\top (A\boldsymbol{x} - \boldsymbol{b}).$$

注意到该问题对 $\boldsymbol{x}$ 而言是一个线性函数，当系数不为 0 时，其最小值一定

[1] 注意到这里使用的是 $\lambda \preceq \mathbf{0}$，这是由于原函数中是 $\boldsymbol{x} \succeq 0$ 而非 $\boldsymbol{x} \preceq 0$。读者可以验证此时取 max 仍然符合原方程。

为负无穷。因此，解内层函数可得对偶问题等价于

$$\begin{aligned} \max_{\boldsymbol{\lambda}\preceq\mathbf{0},\boldsymbol{\nu}} \quad & -\boldsymbol{b}^\top\boldsymbol{\nu} \\ \text{Subject to} \quad & \boldsymbol{c}+\boldsymbol{\lambda}+A^\top\boldsymbol{\nu}=0. \end{aligned} \tag{5.17}$$

注意到 $\boldsymbol{\lambda}$ 并没有参与到目标函数内，而只在约束函数中出现。因此我们可以直接将其化简。从而我们整理可得其对偶形式

$$\begin{aligned} \min_{\boldsymbol{\nu}} \quad & \boldsymbol{b}^\top\boldsymbol{\nu} \\ \text{Subject to} \quad & \boldsymbol{c}+A^\top\boldsymbol{\nu}\succeq 0. \end{aligned} \tag{5.18}$$

注意到这个对偶形式仍然是一个线性规划问题！因此我们可以对这个对偶形式再次取对偶，可得对偶问题的对偶形式。注意到对偶问题的对偶形式刚好就是原问题（这件事并不总是成立）！应用两次弱对偶定理，我们即可得到

$$M_d^* = M_p^*.$$

□

根据强弱对偶定理，我们可知原问题与对偶问题天然有很多联系。因此当原问题求解受阻时，我们经常可以求助其对偶问题，从而获得原问题的诸多性质。例如，我们可以利用对偶性质推导 KKT 条件，这是一个带约束优化问题（在一定正则条件下）有最优解的必要条件。对约束优化问题(5.10)，其对应的 KKT 条件为

$$\begin{aligned} \text{平稳性} \quad & \nabla_{\boldsymbol{x}} L(\boldsymbol{x},\boldsymbol{\lambda},\boldsymbol{\nu})=\mathbf{0}; \\ \text{原问题可行性} \quad & g_i(\boldsymbol{x})\le 0, h_j(\boldsymbol{x})=0, \forall i,j; \\ \text{对偶问题可行性} \quad & \lambda_i\ge 0, \forall i; \\ \text{互补松弛定理} \quad & \lambda_i g_i(\boldsymbol{x})=0, \forall i. \end{aligned} \tag{5.19}$$

前三个性质可以从原问题与对偶问题的拉格朗日形式直接得到。而互补松弛定理可以这样非正式地理解：由于原问题和对偶问题可以视作从左右两个角度求解目标函数最优解。而约束的放松一定会使得目标函数值变差。因此，最终原问题和对偶问题的平衡点（最优解）一定出现在两个问题的交界部分，即对应的约束在平衡的 $\lambda_i g_i(\boldsymbol{x})=0$ 位置。更多关于对偶、KKT 的细节理解请参考书籍[2]。

第六章　支持向量机与一致收敛

“鱼，我所欲也，熊掌亦我所欲也；二者不可得兼，舍鱼而取熊掌者也”
–孟子

6.1　支持向量机 SVM

在上一章里，我们介绍了感知机算法。然而，感知机算法的收敛性依赖于数据的线性可分。接下来我们将会介绍支持向量机 (Support Vector Machine, SVM)，其核心内容与感知机非常像。

正如感知机算法一样，我们仍然考虑一个线性分类器。因此我们希望的目标仍然是

$$\boldsymbol{w}^\top \boldsymbol{x} \geq 0 \text{ if } y = +1,$$
$$\boldsymbol{w}^\top \boldsymbol{x} < 0 \text{ if } y = -1.$$

因此，我们希望考虑的仍然是 $y\boldsymbol{w}^\top \boldsymbol{x} \geq 0$。考虑这一项 $y\boldsymbol{w}^\top \boldsymbol{x}$，在实践中我们当然希望他越大越好。然而，对于同一方向但不同大小的参数，其可能有不同的 $y\boldsymbol{w}^\top \boldsymbol{x}$，但从分类的角度来看，这两个参数实际上是等价的。因此，一个很重要的项是

$$\gamma = \frac{y\boldsymbol{w}^\top \boldsymbol{x}}{\|\boldsymbol{w}\|}.$$

这一项被我们称为几何间隔。

我们希望能够最大化每个样本的几何间隔，即

$$\max_{\boldsymbol{w}} \min_{i \in [n]} \frac{y_i \boldsymbol{w}^\top \boldsymbol{x}_i}{\|\boldsymbol{w}\|}. \tag{6.1}$$

这等价于

$$\begin{aligned} \max_{\boldsymbol{w}} \quad & \gamma \\ \text{Subject to} \quad & \frac{y_i \boldsymbol{w}^\top \boldsymbol{x}_i}{\|\boldsymbol{w}\|} \geq \gamma, i \in [n] \end{aligned} \tag{6.2}$$

注意到在这个优化问题中，调整 $\|\boldsymbol{w}\|$ 并不会改变优化问题。因此，固定 $\|\boldsymbol{w}\|$ 并不会改变整个优化问题。我们可以直接选取 $\|\boldsymbol{w}\| = 1/\gamma$，从而得到优化问题

$$\begin{aligned} \max_{\boldsymbol{w}} \quad & 1/\|\boldsymbol{w}\| \\ \text{Subject to} \quad & y_i \boldsymbol{w}^\top \boldsymbol{x}_i \geq 1, i \in [n] \end{aligned} \tag{6.3}$$

由于 $1/\|\boldsymbol{w}\|$ 不凸，正如之前讨论过的，我们希望其是凸优化。幸运的是，以上问题等价于

$$\begin{aligned} \min_{\boldsymbol{w}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 \\ \text{Subject to} \quad & y_i \boldsymbol{w}^\top \boldsymbol{x}_i \geq 1, i \in [n] \end{aligned} \tag{6.4}$$

这就是 SVM 算法的核心优化目标。通过求解该问题即可得到最优的 $\boldsymbol{w}^*$。

然而，上述算法没有考虑到数据线性不可分的情况。当模型不能线性可分时，上述问题无解。这时，需要我们对原有的问题进行一个软间隔近似，可以写为：

$$\begin{aligned} \max_{\boldsymbol{w}} \ & \frac{1}{2}\|\boldsymbol{w}\|^2 + C \sum_{i \in [n]} \xi_i \\ \text{Subject to } & y_i \boldsymbol{w}^\top \boldsymbol{x}_i \geq 1 - \xi_i, i \in [n] \\ & \xi_i \geq 0 \end{aligned} \tag{6.5}$$

可见，我们允许部分样本的间隔有一个 ξ_i 的放松。这样，SVM 就突破了只能处理线性可分数据的限制。

6.2 一致收敛与 VC 维

在 SVM 中，我们可以优化到一个解。然而，这个解是否是我们想要的解，其对应的泛化性质是否是好的？这需要我们进一步用泛化理论的知识进行解答。接下来的讨论中，我们主要考虑硬间隔的 SVM 算法，即式6.4。

回顾一下，我们考虑泛化差距

$$L(h_{\hat{\boldsymbol{w}}}) - L_n(h_{\hat{\boldsymbol{w}}}) = \mathbb{E}_{\boldsymbol{z}} \ell(h_{\hat{\boldsymbol{w}}}; \boldsymbol{z}) - \frac{1}{n} \sum_{i \in [n]} \ell(h_{\hat{\boldsymbol{w}}}; \boldsymbol{z}_i). \tag{6.6}$$

上式很像是统计中的大数定律/中心极限定理，但有一些区别。在中心极限定理中，我们要求每一项是独立同分布的，而 $\ell(h_{\hat{\boldsymbol{w}}}; \boldsymbol{z}_i)$ 中 $h_{\hat{\boldsymbol{w}}}$ 即是由训练集 $\{\boldsymbol{z}_i\}_{i=1}^n$ 训练得到的，因此我们无法直接用中心极限定理的结果。这事实上是泛化问题最核心、最本质的矛盾，即训练样本和模型之间的依赖性 (dependency)。为了处理这个问题，一个常用的做法是使用一致收敛。

定义 6.1 (一致收敛). 在一致收敛中，我们将泛化差距放松为

$$L(h_{\hat{\boldsymbol{w}}}) - L_n(h_{\hat{\boldsymbol{w}}}) \leq \sup_{h \in \mathcal{H}} |L(h) - L_n(h)|, \tag{6.7}$$

其中 $\mathcal{H}$ 是一个函数族。

在一致收敛的定义中，我们通过在函数族上取 sup，使得原始的依赖性被破坏掉。注意到 $\mathcal{H}$ 往往和训练无关，因此右侧又像中心极限定理的形式了。我们可以基于此使用一些 concentration 的相关技巧。

例如，当 $\mathcal{H}$ 中只有有限个函数时，我们可以获得以下结果

Proposition 6.2.1. 假设 $\ell(\cdot) \leq B$，则下式以至少 $1-\delta$ 概率成立：

$$\sup_{h \in \mathcal{H}} |L(h) - L_n(h)| \leq \sqrt{\frac{B^2}{2n} \log(\frac{|\mathcal{H}|}{\delta})} \tag{6.8}$$

证明. 对于每个 h，根据 hoffding 不等式，

$$\mathbb{P}(|L(h) - L_n(h)| \geq t) \leq \exp(-\frac{2nt^2}{B^2}). \tag{6.9}$$

因此，通过 union bound，有

$$\mathbb{P}(\max_{h \in \mathcal{H}} |L(h) - L_n(h)| \geq t) \leq \sum_{h \in \mathcal{H}} \mathbb{P}(|L(h) - L_n(h)| \geq t) \leq |\mathcal{H}| \exp(-\frac{2nt^2}{B^2}). \tag{6.10}$$

取 $t = \sqrt{\frac{B^2}{2n} \log(\frac{|\mathcal{H}|}{\delta})}$ 即完成证明。 □

然而，对于 SVM 而言，显然 $\mathcal{H} = \{\text{sign}(\boldsymbol{w}^\top \boldsymbol{x})\}$ 中含有无穷多个函数。因此，我们需要用进一步的工具获得一致收敛的界。一个常见的思路是：将函数族 $\mathcal{H}$ 切割成有穷个部分，其中每个部分的表现很一致，而后在这有穷个部分上使用 union bound。例如，对于一个边长为 1 的 d 维正方体而言，可以用 $\mathcal{N} = (1/\epsilon)^d$ 个边长为 ϵ 的小正方体包裹。这里的 $\mathcal{N}$ 代表了原始空

间被切割的程度，与函数族 $\mathcal{H}$、细分程度 ϵ，以及度量空间有关，其被叫做覆盖数 (covering number)。在分类问题中，覆盖数的上界可以通过计算 VC 维获得。得到覆盖数之后，我们可以通过 Dudley 不等式获得相应的泛化界。

Lemma 6.2.1 (基于覆盖数的泛化界)**.** *对于一个布尔函数族 $\mathcal{H} : \mathcal{X} \to \{-1, 1\}$，如果 (1) 存在 d_{vc} 个样本点能够被假设空间 $\mathcal{H}$ 中的函数以所有的 $2^{d_{vc}}$ 种方式进行划分；(2) 对任意大小为 $d_{vc}+1$ 的样本集，$\mathcal{H}$ 中的函数无法满足所有的 $2^{d_{vc}+1}$ 种标签分配方式，则函数族 $\mathcal{H}$ 的 VC 维为 d_{vc}。此时，函数族 $\mathcal{H}$ 的覆盖数为*

$$\mathcal{N}(\mathcal{H}, L^2(\mu), \epsilon) \leq \left(\frac{2}{\epsilon}\right)^{C_1 d_{vc}}, \tag{6.11}$$

其中 C_1 是一个常数。进一步的，当满足通过 Dudley 不等式可以直接获得其泛化界

$$\mathbb{E} \sup_{h \in \mathcal{H}} |L(h) - L_n(h)| \leq C_2 \sqrt{\frac{d_{vc}}{n}}, \tag{6.12}$$

其中 C_2 是一个常数。

证明. 由上文的例子中我们可以直观的感受出，覆盖数 $\approx (\frac{1}{\text{细分程度}})^{\text{维数}}$，其中“维数”的定义与原始空间和度量空间的选取有关。在这里，VC 维衡量了布尔函数族的分类能力，VC 维越高意味着函数族能生成更加复杂的分类边界，因此其可以作为对布尔函数族“维数”的刻画，式6.11的详细证明见 Vershynin[3]定理 8.3.18。

基于覆盖数，我们可以将泛化误差分解为在覆盖数集合上的误差以及逼近误差，注意到这样的分解是和覆盖数的细分程度 ϵ 相关的，不同的 ϵ 可以得到不同的误差上界，因此我们可以利用 Dudley 不等式在这些不同的 ϵ 尺度上做平均来得到一个多尺度的误差上界从而得出式6.12的结论，详细证明见 Vershynin[3]定理 8.3.23。 □

对于 SVM 这一类的线性函数族，我们可以获得如下结果：

定理 6.1 (SVM 的泛化界). 对于线性函数族 $\mathcal{H} = \{sign(\boldsymbol{w}^\top \boldsymbol{x}) : \boldsymbol{w} \in \mathbb{R}^d\}$，其 *VC* 维为 d, 因此基于引理6.2.1，*SVM* 的泛化界为

$$\mathbb{E} \sup_{h \in \mathcal{H}} |L(h) - L_n(h)| \leq C\sqrt{\frac{d}{n}}. \tag{6.13}$$

其中 C 是一个常数。

定理6.1的证明. 基于引理6.2.1的结论，我们只需证明 d 维齐次的线性函数族 $\mathcal{H} = \{\text{sign}(\boldsymbol{w}^\top \boldsymbol{x}) : \boldsymbol{w} \in \mathbb{R}^d\}$ 的 VC 维为 d。

首先我们证明 $\mathcal{H}$ 的 VC 维至少为 d，选取 $\mathbb{R}^d$ 空间中的 d 个标准基向量 $\boldsymbol{e}_i$（第 i 个分量为 1，其余为 0）。对于任意标签组合 $[y_1, y_2, \ldots, y_d]$, $y_i \in \{\pm 1\}$, 选取对应的权重 $\boldsymbol{w} = [y_1, y_2, \ldots, y_d]^\top$，则有 $\boldsymbol{w}^\top \boldsymbol{e}_i = y_i$。因此 $\mathcal{H}$ 可以正确分类这 d 个标准基向量的任意标签组合，故 $\mathcal{H}$ 的 VC 维至少为 d。

其次我们证明 $\mathcal{H}$ 的 VC 维不超过 $d+1$，在 $\mathbb{R}^d$ 空间中，任意 $d+1$ 个向量 $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{d+1}$ 必定线性相关，因此存在不全为 0 的标量 $\alpha_1, \ldots, \alpha_{d+1} \in \mathbb{R}$, 使得：$\sum_{i=1}^{d+1} \alpha_i \boldsymbol{x}_i = \boldsymbol{0}$。不失一般性，假设 α_i 中至少有一个是正数，定义标签 $y_i = \text{sign}(\alpha_i)$，若 $\alpha_i \neq 0$；$y_i =$ 任意 ± 1，若 $\alpha_i = 0$。假设存在 $\boldsymbol{w}$ 使得 $\text{sign}(\boldsymbol{w}^\top \boldsymbol{x}_i) = y_i$ 对所有 i 成立，此时 α_i 与 $\boldsymbol{w}^\top \boldsymbol{x}_i$ 同符号，从而 $\sum_{i=1}^{d+1} \alpha_i \boldsymbol{w}^\top \boldsymbol{x}_i = \sum_{i=1}^{d+1} |\alpha_i| \cdot |\boldsymbol{w}^\top \boldsymbol{x}_i| > 0$。这与 $\sum_{i=1}^{d+1} \alpha_i \boldsymbol{x}_i = \boldsymbol{0}$ 矛盾，因此 $\mathcal{H}$ 无法正确分类 $d+1$ 个点的任意标签组合，因此 $\mathcal{H}$ 的 VC 维小于 $d+1$。 □

6.3 kernel SVM

与感知机算法类似，我们可以获得式6.4的对偶形式如下

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \frac{1}{2}\|\lambda_i y_i \boldsymbol{x}_i\|^2 - \sum_{i \in [n]} \lambda_i \\ \text{Subject to} \quad & \sum_{i \in [n]} \alpha_i y_i = 0 \\ & \boldsymbol{\alpha} \succeq 0 \end{aligned} \tag{6.14}$$

此时参数可表达为 $\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i$，对应的预测为 $\hat{y} = \sum_i \alpha_i y_i \boldsymbol{x}_i^\top \boldsymbol{x}$。能够证明，SVM 的原问题和对偶问题满足强对偶定理。且注意到，对偶优化问题和最终的预测函数均与 $\boldsymbol{x}_i^\top \boldsymbol{x}_j$ 有关。

事实上，现实中的数据往往都不是线性可分的，因此我们有时需要手动构造一些特征 $\phi(\boldsymbol{x}) \in \mathbb{R}^{d_\phi}$ 并假设数据在构造的新特征下线性可分。那么，如果我们将原始的线性预测函数 $\text{sign}(\boldsymbol{w}^\top \boldsymbol{x})$ 改为 $\text{sign}(\boldsymbol{w}^\top \phi(\boldsymbol{x}))$，其对偶形式很自然的会变为如下形式：

$$\begin{aligned} \max_{\boldsymbol{\lambda}} \quad & \frac{1}{2}\|\lambda_i y_i \phi(\boldsymbol{x}_i)\|^2 - \sum_{i\in[n]} \lambda_i \\ \text{Subject to} \quad & \sum_{i\in[n]} \alpha_i y_i = 0 \\ & \boldsymbol{\alpha} \succeq 0 \end{aligned} \tag{6.15}$$

其中有关 $\boldsymbol{x}$ 的部分只有内积 $\phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_i)$。然而，一般来说构造的特征维度远远大于原始的数据维度，即 $d_\phi \gg d$。因此，直接计算 $\phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_i)$ 的开销很大。幸运的是，我们可以通过 Mercer 定理降低上述内积的计算开销。

定理 6.2 (Mercer 定理)**.** *对于函数 $K(\cdot,\cdot)$，如果矩阵 $K \in \mathbb{R}^{n\times n}$ 中的每一个元素为 $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$，且 $K \succeq 0$，则可以用 $K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ 来对应代替 $\phi(\boldsymbol{x}_i)^\top \phi(\boldsymbol{x}_i)$。*

一个常见的核函数是高斯核 $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp(-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|/2\sigma^2)$，其对应的 ϕ 是一个无穷维上的函数。如果我们直接计算 $\phi(\boldsymbol{x}) \in \mathbb{R}^\infty$ 对应的参数 $\boldsymbol{w} \in \mathbb{R}^\infty$，其计算量是完全无法接受的。然而，当我们采用了上述的核技巧 (kernel trick) 后，所有的计算便完全可以接受。

6.4 拉德马赫复杂度

当使用了核技巧后，我们就无法使用 VC 维来计算泛化界了。这是因为对应的泛化界将会是 $\sqrt{d_\phi/n}$，而往往 $\phi \gg n$ （如高斯核）。因此，我们要进一步探索函数族的拉德马赫复杂度（Rademacher Complexity）。

由于误差函数 L 是损失函数 ℓ 和分类器 h 的复合，因此接下来我们考虑函数族 $\mathcal{F} = \{f(\cdot) = \ell(h;\cdot) : h \in \mathcal{H}\}$ 的拉德马赫复杂度，其定义如下：

定理 6.3 (Rademacher Complexity)**.** 函数族 $\mathcal{F}$ 在样本集 $\mathcal{D} = \{\boldsymbol{z}_i\}_{i=1}^n$ 下的拉德马赫复杂度被定义为

$$\mathcal{R}(\mathcal{F} \circ \mathcal{D}) = \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}} \sup_{f \in \mathcal{F}} \sum_{i=1}^{n} \sigma_i f(\boldsymbol{z}_i) \tag{6.16}$$

其中 σ_i 是 *i.i.d.* 的拉德马赫随机变量满足 $\mathbb{P}(\sigma_i = 1) = \mathbb{P}(\sigma_i = -1) = \frac{1}{2}$，则通过拉德马赫复杂度可以获得相应的泛化界

$$\mathbb{E} \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{\boldsymbol{Z}}[f(\boldsymbol{Z})] - \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{z}_i) \right) \leq 2\mathcal{R}(\mathcal{F} \circ \mathcal{D}) \tag{6.17}$$

定理*6.3*的证明. 我们首先引入 “幽灵样本” $\mathcal{D}' = \{\boldsymbol{z}_i'\}_{i=1}^n$ 是与 $\mathcal{D} = \{\boldsymbol{z}_i\}_{i=1}^n$ 独立同分布的另一组样本，则有

$$\mathbb{E}_{\mathcal{D}} \sup_{f \in \mathcal{F}} \left(\mathbb{E}_{\boldsymbol{z} \sim P}[f(\boldsymbol{z})] - \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{z}_i) \right) = \mathbb{E}_{\mathcal{D}, \mathcal{D}'} \sup_{f \in \mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{z}_i') - \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{z}_i) \right).$$

利用独立样本 $\mathcal{D}$ 与 $\mathcal{D}'$ 的对称性，我们有：

$$\mathbb{E}_{\mathcal{D}, \mathcal{D}'} \sup_{f \in \mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} \Big(f(\boldsymbol{z}_i') - f(\boldsymbol{z}_i) \Big) \right) \leq \mathbb{E}_{\mathcal{D}, \mathcal{D}', \boldsymbol{\sigma}} \sup_{f \in \mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} \sigma_i \Big(f(\boldsymbol{z}_i') - f(\boldsymbol{z}_i) \Big) \right),$$

由于 $\mathcal{D}$ 和 $\mathcal{D}'$ 独立且分布相同，且 σ_i 对称，所以

$$\mathbb{E}_{\mathcal{D}, \mathcal{D}', \boldsymbol{\sigma}} \sup_{f \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \sigma_i \Big(f(\boldsymbol{z}_i') - f(\boldsymbol{z}_i) \Big) \leq \mathbb{E}_{\mathcal{D}, \boldsymbol{\sigma}} \sup_{f \in \mathcal{F}} \frac{2}{n} \sum_{i=1}^{n} \sigma_i f(\boldsymbol{z}_i) = 2\,\mathcal{R}(\mathcal{F} \circ \mathcal{D}).$$

这里用到了对称性，即 $\mathbb{E}_{\mathcal{D}, \mathcal{D}', \boldsymbol{\sigma}}$ 中对 $\boldsymbol{z}_i$ 与 $\boldsymbol{z}_i'$ 的贡献是相同的。
这种证明方法在统计学习理论中非常常见，利用了对称化技术从而将泛化误差与拉德马赫复杂度联系起来。 □

利用 McDiarmid 不等式，我们可以得到式6.17的高概率表达式。

定理 6.4 (泛化界高概率表达式)**.** 若对任意 $f \in \mathcal{F}$，有 $|f(\cdot)| \leq c$ 成立，则对任意 $\delta > 0$，以下的式子以至少 $1 - \delta$ 的概率成立：对任意 $f \in \mathcal{F}$

$$\mathbb{E}_{\boldsymbol{Z}}[f(\boldsymbol{Z})] - \frac{1}{n} \sum_{i=1}^{n} f(\boldsymbol{z}_i) \leq 2\mathbb{E}\left[\mathcal{R}(\mathcal{F} \circ \mathcal{D})\right] + c\sqrt{\frac{2\log(1/\delta)}{n}} \tag{6.18}$$

证明. 记 $\Phi(\mathcal{D}) = \sup_{f\in\mathcal{F}}\left(\mathbb{E}_{\boldsymbol{Z}}[f(\boldsymbol{Z})] - \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{z}_i)\right)$，可以注意到对任意 $f \in \mathcal{F}$ 均有 $|f(\cdot)| \le c$，因此，对于函数 $\Phi(\mathcal{D})$ 满足以下有界变差条件：

$$|\Phi(\boldsymbol{z}_1,\ldots,\boldsymbol{z}_i,\ldots,\boldsymbol{z}_n) - \Phi(\boldsymbol{z}_1,\ldots,\boldsymbol{z}_i',\ldots,\boldsymbol{z}_n)| \le \frac{c}{n}.$$

根据 McDiarmid 不等式，对于满足有界差条件的函数 $\Phi(\mathcal{D})$，有

$$\mathbb{P}\left\{\Phi(\mathcal{D}) - \mathbb{E}[\Phi(\mathcal{D})] \ge t\right\} \le \exp\left(-\frac{2t^2}{\sum_{i=1}^{n}(c/n)^2}\right) = \exp\left(-\frac{2nt^2}{c^2}\right).$$

令 $\exp\left(-\frac{2nt^2}{c^2}\right) = \delta$ 可得 $t = c\sqrt{\frac{\log(1/\delta)}{2n}}$，因此，我们得到：

$$\mathbb{P}\left\{\Phi(\mathcal{D}) \ge \mathbb{E}[\Phi(\mathcal{D})] + c\sqrt{\frac{\log(1/\delta)}{2n}}\right\} \le \delta.$$

利用式6.17将其代上述不等式中，可以得到最后的结论：

$$\mathbb{P}\left\{\Phi(\mathcal{D}) \le \mathbb{E}[\Phi(\mathcal{D})] + c\sqrt{\frac{2\log(1/\delta)}{n}}\right\} \ge 1-\delta.$$

□

接下来我们考虑在以下假设下，SVM 算法的泛化界：

1. 输入空间约束：所有样本满足 $\|\boldsymbol{x}_i\| \le R$。
2. 线性可分性：存在权重向量 $\boldsymbol{w}^* \in \mathbb{R}^d$，使得对训练集 $\mathcal{D}$，有 $y_i(\boldsymbol{x}_i^\top \boldsymbol{w}^*) \ge 1$，$\forall i = 1,\ldots,n$，对应的几何间隔定义为 $\gamma = 1/\|\boldsymbol{w}^*\|$。
3. 假设空间：考虑系数有界的假设空间 $\mathcal{H} = \{\text{sign}(\boldsymbol{w}^\top\boldsymbol{x}) : \boldsymbol{w} \in \mathbb{R}^d, \|w\| \le W\}$，其中 $W = 1/\gamma$。

定理 6.5 (SVM 的 Margin-based 泛化界)**.** *基于以上三个假设，以至少 $1-\delta$ 的概率，SVM 算法得到的分类器 $\hat{h}$ 满足：*

$$\mathbb{P}(Y \ne \hat{h}(\boldsymbol{X})) \le \frac{2R}{\gamma\sqrt{n}} + \sqrt{\frac{\log(1/\delta)}{2n}} \tag{6.19}$$

证明. 首先我们给出假设空间 $\mathcal{H} = \{\text{sign}(\boldsymbol{w}^\top\boldsymbol{x}) : \boldsymbol{w} \in \mathbb{R}^d, \|\boldsymbol{w}\| \le W\}$ 的拉德马赫复杂度上界，当输入样本满足 $\|\boldsymbol{x}\| \le R$ 时，$\mathcal{R}(\mathcal{H}\circ\mathcal{D}) \le \frac{RW}{\sqrt{n}}$.,

详细证明见 Shalev-Shwartz[1], 引理 26.10。在本定理中取 $W = \frac{1}{\gamma}$，故有 $\mathcal{R}(\mathcal{H} \circ \mathcal{D}) \leq \frac{R}{\gamma\sqrt{n}}$.

对于 SVM 常用的 hinge loss（或其它 margin loss），该函数通常是 1-Lipschitz 的，因此通过对 $\mathcal{F} = \{\ell(h(\boldsymbol{x}), y) : h \in \mathcal{H}\}$ 应用收缩引理（contraction lemma，详见 Shalev-Shwartz[1]，引理 26.9），可以得到：

$$\mathcal{R}(\mathcal{F} \circ \mathcal{D}) \leq \mathcal{R}(\mathcal{H} \circ \mathcal{D}) \leq \frac{R}{\gamma\sqrt{n}}.$$

从而根据定理6.4，我们可以得到

$$\mathbb{P}\left\{\Phi(\mathcal{D}) \leq \mathbb{E}[\Phi(\mathcal{D})] + \sqrt{\frac{\log(1/\delta)}{2n}}\right\} \geq 1 - \delta.$$

其中 $\Phi(\mathcal{D}) = \sup_{f \in \mathcal{F}}\left(\mathbb{E}_{\boldsymbol{Z}}[f(\boldsymbol{Z})] - \frac{1}{n}\sum_{i=1}^{n} f(\boldsymbol{z}_i)\right)$.

最后，由于线性可分性假设保证存在理想分离超平面，因此 SVM 在训练集上能实现零经验 hinge loss，即 $\frac{1}{n}\sum_{i=1}^{n} \ell\big(\hat{h}(\boldsymbol{x}_i), y_i\big) = 0$. 同时，由于 hinge losss 是 0-1 损失的上界，因此有

$$\mathbb{P}\big(Y \neq \hat{h}(\boldsymbol{X})\big) \leq \mathbb{E}\big[\ell(\hat{h}(\boldsymbol{X}), Y)\big].$$

故以至少 $1 - \delta$ 的概率，有

$$\mathbb{P}\big(Y \neq \hat{h}(\boldsymbol{X})\big) \leq \mathbb{E}\big[\ell(\hat{h}(\boldsymbol{X}), Y)\big] \leq \frac{2R}{\gamma\sqrt{n}} + \sqrt{\frac{\log(1/\delta)}{2n}}.$$

□

进一步对于 kernel SVM，以上的三个假设条件相应的需要变为：

1. 核空间的约束：存在映射 $\phi : \mathcal{X} \to \mathcal{X}'$，将输入映射到再生核希尔伯特空间 $\mathcal{X}'$，且所有的样本满足 $\|\phi(\boldsymbol{x}_i)\|_{\mathcal{X}'} \leq R_\phi$。

2. 特征空间上的线性可分性：存在权重向量 $\boldsymbol{w}^* \in \mathcal{X}'$ 使得对训练集 $\mathcal{D}$，有 $y_i\langle \boldsymbol{w}^*, \phi(\boldsymbol{x}_i)\rangle_{\mathcal{X}'} \geq 1$，$\forall i = 1, \ldots, n$，对应的几何间隔定义为 $\gamma_\phi = 1/\|\boldsymbol{w}^*\|_{\mathcal{X}'}$。

3. 假设空间：考虑系数有界的假设空间 $\mathcal{H}_\phi = \{\text{sign}(\langle \boldsymbol{w}, \phi(\cdot)\rangle)_{\mathcal{X}'} : \boldsymbol{w} \in \mathcal{X}', \|w\|_{\mathcal{X}'} \leq W_\phi\}$，其中 $W_\phi = 1/\gamma_\phi$。

利用相同的证明技术我们可以得到 kernel SVM 的 Margin-based 泛化界。

定理 6.6 (kernel SVM 的 Margin-based 泛化界)**.** 基于以上三个假设，以至少 $1-\delta$ 的概率，*SVM* 算法得到的分类器 $\hat{h}$ 满足：

$$\mathbb{P}(Y \neq \hat{h}(\boldsymbol{X})) \leq \frac{2R_\phi}{\gamma_\phi \sqrt{n}} + \sqrt{\frac{\log(1/\delta)}{2n}} \tag{6.20}$$

第七章　正则与算法稳定性

“ Numquam ponenda est pluralitas sine necessitate ”

–Ockham

7.1　正则化

在训练模型时，我们常常希望训出的模型能够满足一定性质。这通常可以通过正则化 (regularization) 实现。在正则中，我们考虑损失函数

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{i \in [n]} \ell(y_i, f_{\boldsymbol{w}}(\boldsymbol{x}_i)) + \lambda R(\boldsymbol{w}), \tag{7.1}$$

其中，$R(\boldsymbol{w})$ 为正则项 (regularizer)，λ 为正则强度。

为了更好说明正则的意义，我们考虑这样一个例子。在机器学习中，我们经常面临模型过大以至于难以部署的问题。为了解决这一问题，我们希望将大模型转化为更小的模型，同时尽量保持其性能。这个过程通常被称为模型剪枝（pruning）。一种直接的方法是首先训练一个大模型，然后删除那些较小的参数。这种方法虽然直观且易于实现，但它依赖于一个假设：原始训练出的模型中存在大量可以被删除的小参数。然而，这并不总是成立的。为了促进大模型参数的稀疏性，我们可以在训练过程中引入正则化。一个有效的策略是在损失函数中加入 ℓ_1 正则项：

$$\min_{\boldsymbol{w}} \frac{1}{N} \sum_{i \in [n]} \ell(y_i, f_{\boldsymbol{w}}(\boldsymbol{x}_i)) + \lambda \|\boldsymbol{w}\|_1. \tag{7.2}$$

其中，λ 是正则化系数，$\|\boldsymbol{w}\|_1$ 是参数的 ℓ_1 范数。ℓ_1 正则化项可以看作是 ℓ_0 正则化的替代，而 ℓ_0 范数通过惩罚非零参数的个数来鼓励模型参数的稀疏性。关于 ℓ_1 正则项对模型稀疏性的提升，可以参考压缩感知或 LASSO。

在实践中，我们常常需要根据不同的问题结构设计不同的正则项，包括 ℓ_1 正则项，ℓ_2 正则项等等。

7.2 岭回归

根据上一章的泛化结果，模型参数的 ℓ_2 范数与泛化密切相关。因而加入 ℓ_2 正则项可能会提升模型的泛化效果。在这一节中，我们介绍线性回归中加入 ℓ_2 正则项，这又被成为岭回归 (ridge regression)。在岭回归中，其损失函数被定义为

$$\mathrm{MSE}_r(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^\top \boldsymbol{x}_i - y_i)^2 + \lambda\|\boldsymbol{w}\|_2^2 = \|Y - X\boldsymbol{w}\|^2 + \lambda\|\boldsymbol{w}\|^2,$$

其中，$(\boldsymbol{x}_i, y_i)$ 为第 i 个样本，λ 为正则强度。与线性回归类似，当 $d < n$ 时我们可以得到其显式解

$$\hat{w}_r = (XX^\top + \lambda I)^{-1}X^\top Y.$$

可见，与线性回归相比，求逆部分多了一项 λI。这在 $XX^\top$ 特征值较小时非常关键，能够有效提升模型效果与计算稳定性。

7.3 算法稳定性

本节旨在从算法稳定性的角度探讨正则项对模型泛化能力的影响。在讨论算法稳定性之前，我们首先介绍一些必要的符号定义。我们定义算法 $\mathcal{A}$ 为一个映射函数，它将数据集映射到参数空间。具体的，我们可以用 $\boldsymbol{w} = \mathcal{A}(\mathcal{D})$ 表示这一过程。基于此，算法稳定性定义为：

定义 7.1 (算法稳定性)**.** 我们称算法 $\mathcal{A}$ 是 γ-稳定的，如果对于任意两个数据集 $\mathcal{D}$ 和 $\mathcal{D}'$ 且两个数据集只有一个数据点不同时：

$$\sup_{\tilde{\boldsymbol{z}}} \mathbb{E}_{\mathcal{A}}[\ell(\mathcal{A}(\mathcal{D}); \tilde{\boldsymbol{z}}) - \ell(\mathcal{A}(\mathcal{D}'); \tilde{\boldsymbol{z}})] \leq \gamma,$$

这里 $\boldsymbol{z} = (\boldsymbol{x}, y)$ 代表样本点，$\ell(\cdot; \boldsymbol{z})$ 代表损失函数，期望为对算法和训练集的期望。

算法稳定性被认为与泛化密切相关。对于 γ-稳定的算法，我们可以证明如下结论

定理 7.1 (稳定性与泛化)**.** 对于 γ-稳定的算法 $\mathcal{A}$，其对应的泛化差满足

$$\mathbb{E}_{\mathcal{A},\mathcal{D}}[\mathbb{E}_{\boldsymbol{z}\sim\mathcal{P}}\ell(\mathcal{A}(\mathcal{D});\boldsymbol{z}) - \mathcal{L}_n(\mathcal{A}(\mathcal{D});\mathcal{D})] \leq \gamma, \tag{7.3}$$

这里 $\mathbb{E}_{\boldsymbol{z}\sim\mathcal{P}}\ell(\mathcal{A}(\mathcal{D});\boldsymbol{z})$ 代表测试误差，$\mathcal{L}_n(\mathcal{A}(\mathcal{D});\mathcal{D})$ 代表训练误差。

定理*7.1*的证明. 我们直接考虑如下等式：

$$\begin{aligned}
&\mathbb{E}_{\mathcal{A},\mathcal{D}}\mathbb{E}_{\boldsymbol{z}\sim\mathcal{P}}\ell(\mathcal{A}(\mathcal{D});\boldsymbol{z}) \\
\overset{(i)}{=}&\mathbb{E}\ell(\mathcal{A}(\{\boldsymbol{z}_1,\boldsymbol{z}_2,\cdots,\boldsymbol{z}_n\});\boldsymbol{z}') \\
\overset{(ii)}{\leq}&\mathbb{E}\ell(\mathcal{A}(\{\boldsymbol{z}',\boldsymbol{z}_2,\cdots,\boldsymbol{z}_n\});\boldsymbol{z}') + \gamma \\
\overset{(iii)}{=}&\mathbb{E}\mathcal{L}_n(\mathcal{A}(\mathcal{D});\mathcal{D}) + \gamma.
\end{aligned} \tag{7.4}$$

其中，(i) 可由测试误差的定义得到，(ii) 可由算法稳定性的定义得到，(iii) 是因为 $\boldsymbol{z}'$ 出现在了训练集中。 □

接下来，我们考虑凸训练中随机梯度下降是稳定的算法。

定理 7.2 (稳定性与凸训练)**.** 假设 $\ell(\cdot;\boldsymbol{z})$ 对任意 $\boldsymbol{z}$ 均是 *L-Lipshitz*，*M*-光滑函数。

若 $\ell(\cdot;\boldsymbol{z})$ 对任意 $\boldsymbol{z}$ 均是 μ-强凸，且 $\eta_t \leq \mu/M^2$，则其稳定性满足

$$\sup_{\tilde{\boldsymbol{z}}} \mathbb{E}_{\mathcal{A}}[\ell(\boldsymbol{w}[T];\tilde{\boldsymbol{z}}) - \ell(\boldsymbol{w}'[T];\tilde{\boldsymbol{z}})] \leq \frac{4L^2}{n\mu}.$$

若 $\ell(\cdot;\boldsymbol{z})$ 对任意 $\boldsymbol{z}$ 均是凸的，且 $\eta_t \leq 2/M$，则其稳定性满足

$$\sup_{\tilde{\boldsymbol{z}}} \mathbb{E}_{\mathcal{A}}[\ell(\boldsymbol{w}[T];\tilde{\boldsymbol{z}}) - \ell(\boldsymbol{w}'[T];\tilde{\boldsymbol{z}})] \leq \frac{2L^2}{n}\sum_{i\in[T]}\eta_t.$$

在图7.1中可以发现，加入 ℓ_2 正则项可以让凸函数变为强凸函数，也可以增强原有强凸函数的凸性。因此，这可以被视为是使用 ℓ_2 正则项能够提升泛化性的一项证据。

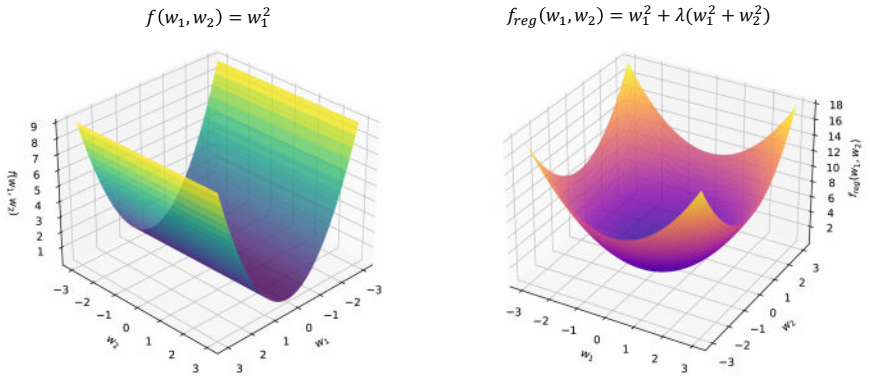


图 7.1: ℓ_2 正则项可以使凸函数（左图）变为强凸函数（右图）

定理7.2的证明. 在证明中，为了符号的简洁，我们记下列符号：

$$\boldsymbol{w}[t] = \mathcal{A}_t(\mathcal{D}),\ \boldsymbol{w}'[t] = \mathcal{A}_t(\mathcal{D}').$$

根据稳定性的定义，我们使用相同的算法，因此其初始化应是相同的，即 $\boldsymbol{w}[0] = \boldsymbol{w}'[0]$. 记两个数据集分别为 $\mathcal{D} = \{\boldsymbol{z}_i\}_{i\in[n]}$, $\mathcal{D}' = \{\boldsymbol{z}'_i\}_{i\in[n]}$。不妨记不同的数据点为第 n 个，即 $\boldsymbol{z}_i = \boldsymbol{z}'_i, i \in [n-1]$, $\boldsymbol{z}_n \neq \boldsymbol{z}'_n$. 对于 SGD 算法，其迭代更新满足

$$\begin{aligned}\boldsymbol{w}[t+1] &= \boldsymbol{w}[t] - \eta\nabla\ell(\boldsymbol{w}[t]; \boldsymbol{z}_{(t)}),\\ \boldsymbol{w}'[t+1] &= \boldsymbol{w}'[t] - \eta\nabla\ell(\boldsymbol{w}'[t]; \boldsymbol{z}'_{(t)}),\end{aligned}$$

其中，$\boldsymbol{z}_{(t)}$ 为第 t 步抽取的样本，(t) 为其样本标号。由于两个迭代使用了完全相同的算法，其抽取的样本标号相同。其次，注意到当损失函数 $\ell(\cdot; \boldsymbol{z})$ 满足对任意的 $\boldsymbol{z}$ 都是 L-Lipschitz 时，算法稳定性满足

$$\sup_{\tilde{\boldsymbol{z}}} \mathbb{E}_{\mathcal{A}}[\ell(\boldsymbol{w}[t]; \tilde{\boldsymbol{z}}) - \ell(\boldsymbol{w}'[t]; \tilde{\boldsymbol{z}})] \leq L\mathbb{E}_{\mathcal{A}}\|\boldsymbol{w}[t] - \boldsymbol{w}'[t]\|. \tag{7.5}$$

因此，我们接下来考虑参数的迭代对 $\|\boldsymbol{w}[t] - \boldsymbol{w}'[t]\|$ 的影响。

Part I: 强凸。在每一步的迭代过程中均可能会发生两种情况：两个迭代抽到了相同的样本 ($(t) \in [n-1]$, 概率为 $1-1/n$) 或不同的样本 ($(t) = n$, 概率为 $1/n$)。这两种情况的表现非常不同，我们分别来看。

Case 1: 抽取不同样本 $(t)=n$，则迭代满足

$$\begin{aligned}\|\boldsymbol{w}[t+1]-\boldsymbol{w}'[t+1]\| =&\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]+\eta_t(\nabla\ell(\boldsymbol{w}'[t];\boldsymbol{z}'_{(t)})-\nabla\ell(\boldsymbol{w}[t];\boldsymbol{z}_{(t)})\|\\ \leq&\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|+2\eta_t L,\end{aligned} \tag{7.6}$$

这里我们用到了性质：$\ell(\cdot;\boldsymbol{z})$ 对任意 $\boldsymbol{z}$ 均是 L-Lipschitz。

Case 2: 抽取相同样本 $(t)\in[n-1]$，则迭代满足

$$\|\boldsymbol{w}[t+1]-\boldsymbol{w}'[t+1]\| \leq(1-\frac{1}{2}\eta_t\mu)\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|, \tag{7.7}$$

这里我们使用了引理7.3.1。综合式(7.6)和(7.7)，可得

$$\begin{aligned}&\mathbb{E}\|\boldsymbol{w}[t+1]-\boldsymbol{w}'[t+1]\|\\ \leq&\mathbb{E}\frac{1}{n}(\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|+2\eta_t L)+(1-\frac{1}{n})(1-\frac{1}{2}\eta_t\mu)\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|\\ =&(1-\frac{1}{2}\eta_t\mu(1-1/n))\mathbb{E}\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|+\frac{2\eta_t L}{n}\\ \leq&(1-\frac{1}{2}\eta_t\mu)\mathbb{E}\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|+\frac{2\eta_t L}{n}.\end{aligned} \tag{7.8}$$

注意到 $\|\boldsymbol{w}[0]-\boldsymbol{w}'[0]\|=0$，将其不断根据上式展开并利用等比数列前 n 项和公式可得：

$$\mathbb{E}\|\boldsymbol{w}[T]-\boldsymbol{w}'[T]\| \leq \frac{4L}{n\mu}. \tag{7.9}$$

综合式(7.5)，即可得

$$\sup_{\tilde{\boldsymbol{z}}}\mathbb{E}_{\mathcal{A}}[\ell(\boldsymbol{w}[T];\tilde{\boldsymbol{z}})-\ell(\boldsymbol{w}'[T];\tilde{\boldsymbol{z}})] \leq \frac{4L^2}{n\mu}. \tag{7.10}$$

Part II: 一般凸函数。类似于强凸部分的讨论，我们也需要考虑两个情况。*Case 1:* 抽取不同样本 $(t)=n$，则迭代满足（与强凸相同）

$$\|\boldsymbol{w}[t+1]-\boldsymbol{w}'[t+1]\| \leq \|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|+2\eta_t L, \tag{7.11}$$

Case 2: 抽取相同样本 $(t)\in[n-1]$，则迭代满足（根据引理7.3.1）

$$\|\boldsymbol{w}[t+1]-\boldsymbol{w}'[t+1]\| \leq \|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|, \tag{7.12}$$

综合式(7.11)和(7.12)，可得

$$\begin{aligned}&\mathbb{E}\|\boldsymbol{w}[t+1]-\boldsymbol{w}'[t+1]\|\\ \leq&\mathbb{E}\frac{1}{n}(\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|+2\eta_t L)+(1-\frac{1}{n})\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|\\ =&\mathbb{E}\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|+\frac{2\eta_t L}{n}.\end{aligned} \tag{7.13}$$

因此，根据条件 $\|\boldsymbol{w}[0]-\boldsymbol{w}'[0]\|=0$，通过累加并综合式(7.5)可得

$$\|\boldsymbol{w}[T]-\boldsymbol{w}'[T]\| \leq \frac{2L}{n}\sum_{i\in[T]}\eta_{t-1}.$$

$$\sup_{\tilde{\boldsymbol{z}}}\mathbb{E}_{\mathcal{A}}[\ell(\boldsymbol{w}[T];\tilde{\boldsymbol{z}})-\ell(\boldsymbol{w}'[T];\tilde{\boldsymbol{z}})] \leq \frac{2L^2}{n}\sum_{i\in[T]}\eta_{t-1}.$$

□

Lemma 7.3.1 (扩张性质)**.** 假设 $\ell(\cdot;\boldsymbol{z})$ 对任意 $\boldsymbol{z}$ 均是 L-*Lipshitz*，M-光滑函数。若在第 t 步时算法选择了相同的样本（即 $(t)\in[n-1]$）
若 $\ell(\cdot;\boldsymbol{z})$ 对任意 $\boldsymbol{z}$ 均是 μ-强凸，且 $\eta_t\leq\mu/M^2$，则迭代满足

$$\|\boldsymbol{w}[t+1]-\boldsymbol{w}'[t+1]\| \leq (1-\frac{1}{2}\eta_t\mu)\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|.$$

若 $\ell(\cdot;\boldsymbol{z})$ 对任意 $\boldsymbol{z}$ 均是凸的，且 $\eta_t\leq 2/M$，则迭代满足

$$\|\boldsymbol{w}[t+1]-\boldsymbol{w}'[t+1]\| \leq \|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|.$$

引理*7.3.1*的证明. **Part I: 强凸**. 考虑到，当使用同样的样本 $\boldsymbol{z}_{(t)}=\boldsymbol{z}'_{(t)}$ 时：

$$\begin{aligned}
&\|\boldsymbol{w}[t+1]-\boldsymbol{w}'[t+1]\|^2\\
=&\|\boldsymbol{w}[t]-\eta_t\nabla\ell(\boldsymbol{w}[t];\boldsymbol{z}_{(t)})-\boldsymbol{w}'[t]+\eta_t\nabla\ell(\boldsymbol{w}'[t];\boldsymbol{z}_{(t)})\|^2\\
=&\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|^2+\eta_t^2\|\nabla\ell(\boldsymbol{w}[t];\boldsymbol{z}_{(t)})-\nabla\ell(\boldsymbol{w}'[t];\boldsymbol{z}_{(t)})\|^2\\
&-2\eta_t(\boldsymbol{w}[t]-\boldsymbol{w}'[t])^\top(\nabla\ell(\boldsymbol{w}[t];\boldsymbol{z}_{(t)})-\nabla\ell(\boldsymbol{w}'[t];\boldsymbol{z}_{(t)})).
\end{aligned} \tag{7.14}$$

我们希望将式(7.18)右边的三项均化为 $\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|^2$ 的相关形式。对于第二项，由于损失函数是 M-光滑的，可知

$$\eta_t^2\|\nabla\ell(\boldsymbol{w}[t];\boldsymbol{z}_{(t)})-\nabla\ell(\boldsymbol{w}'[t];\boldsymbol{z}_{(t)})\|^2 \leq \eta_t^2M^2\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|^2. \tag{7.15}$$

对于第三项，由于损失函数是 μ-强凸的，根据事实7.4可知

$$-2\eta_t(\boldsymbol{w}[t]-\boldsymbol{w}'[t])^\top(\nabla\ell(\boldsymbol{w}[t];\boldsymbol{z}_{(t)})-\nabla\ell(\boldsymbol{w}'[t];\boldsymbol{z}_{(t)})) \leq -2\eta_t\mu\|\boldsymbol{w}[t+1]-\boldsymbol{w}'[t+1]\|^2. \tag{7.16}$$

将式(7.15)和式(7.16)带入式(7.18)，可得

$$\|\boldsymbol{w}[t+1]-\boldsymbol{w}'[t+1]\|^2 \leq (1+\eta_t^2M^2-2\eta_t\mu)\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|^2 \leq (1-\eta_t\mu)\|\boldsymbol{w}[t]-\boldsymbol{w}'[t]\|^2.$$

最后一个等式是由于假设 $\eta_t \leq \mu/M^2$. 因此，可得

$$\|\boldsymbol{w}[t+1] - \boldsymbol{w}'[t+1]\| \leq \sqrt{(1-\eta_t\mu)}\|\boldsymbol{w}[t] - \boldsymbol{w}'[t]\| \leq (1 - \frac{1}{2}\eta_t\mu)\|\boldsymbol{w}[t] - \boldsymbol{w}'[t]\|. \tag{7.17}$$

这里我们使用了不等式 $\sqrt{1-x} \leq 1 - x/2, x \in (0,1)$.

Part II: 凸。对于一般凸的样本，可得

$$\begin{aligned}
&\|\boldsymbol{w}[t+1] - \boldsymbol{w}'[t+1]\|^2 \\
=&\|\boldsymbol{w}[t] - \eta_t\nabla\ell(\boldsymbol{w}[t];\boldsymbol{z}_{(t)}) - \boldsymbol{w}'[t] + \eta_t\nabla\ell(\boldsymbol{w}'[t];\boldsymbol{z}_{(t)})\|^2 \\
=&\|\boldsymbol{w}[t] - \boldsymbol{w}'[t]\|^2 + \eta_t^2\|\nabla\ell(\boldsymbol{w}[t];\boldsymbol{z}_{(t)}) - \nabla\ell(\boldsymbol{w}'[t];\boldsymbol{z}_{(t)})\|^2 \\
&- 2\eta_t(\boldsymbol{w}[t] - \boldsymbol{w}'[t])^\top(\nabla\ell(\boldsymbol{w}[t];\boldsymbol{z}_{(t)}) - \nabla\ell(\boldsymbol{w}'[t];\boldsymbol{z}_{(t)})) \\
\overset{(i)}{\leq}&\|\boldsymbol{w}[t] - \boldsymbol{w}'[t]\|^2 - (\frac{2\eta_t}{M} - \eta_t^2)\|\nabla\ell(\boldsymbol{w}[t];\boldsymbol{z}_{(t)}) - \nabla\ell(\boldsymbol{w}'[t];\boldsymbol{z}_{(t)})\|^2 \\
\overset{(ii)}{\leq}&\|\boldsymbol{w}[t] - \boldsymbol{w}'[t]\|^2.
\end{aligned} \tag{7.18}$$

这里我们在式 (i) 中使用了引理7.3.1，在式 (ii) 中使用了条件 $\eta_t \leq 2/M$。

□

Claim 7.3. 对于 μ-强凸函数 $f(\boldsymbol{w})$，其满足

$$(\nabla f(\boldsymbol{w}') - \nabla f(\boldsymbol{w}))^\top(\boldsymbol{w}' - \boldsymbol{w}) \geq \mu\|\boldsymbol{w} - \boldsymbol{w}'\|^2.$$

证明. 通过泰勒展开可知：

$$\begin{aligned}
f(\boldsymbol{w}) \geq& f(\boldsymbol{w}') + \nabla f(\boldsymbol{w}')(\boldsymbol{w} - \boldsymbol{w}') + \frac{\mu}{2}\|\boldsymbol{w} - \boldsymbol{w}'\|^2. \\
f(\boldsymbol{w}') \geq& f(\boldsymbol{w}) + \nabla f(\boldsymbol{w})(\boldsymbol{w}' - \boldsymbol{w}) + \frac{\mu}{2}\|\boldsymbol{w} - \boldsymbol{w}'\|^2.
\end{aligned}$$

相加可得

$$(\nabla f(\boldsymbol{w}') - \nabla f(\boldsymbol{w}))^\top(\boldsymbol{w}' - \boldsymbol{w}) \geq \mu\|\boldsymbol{w} - \boldsymbol{w}'\|^2.$$

□

Claim 7.4. 对于 M-光滑函数 $f(\boldsymbol{w})$，其满足

$$(\nabla f(\boldsymbol{w}') - \nabla f(\boldsymbol{w}))^\top(\boldsymbol{w}' - \boldsymbol{w}) \geq \frac{1}{M}\|\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{w}')\|^2.$$

证明. 构造函数 $g(\boldsymbol{w}) = f(\boldsymbol{w}) - \nabla^\top f(\boldsymbol{w}')(\boldsymbol{w} - \boldsymbol{w}')$。注意到 $g(\boldsymbol{w})$ 仍是一个凸光滑函数，考虑其泰勒展开

$$g(\boldsymbol{w}') \leq g(\boldsymbol{w}) + \nabla^\top g(\boldsymbol{w})(\boldsymbol{w}' - \boldsymbol{w}) + \frac{M}{2}\|\boldsymbol{w}' - \boldsymbol{w}\|^2. \tag{7.19}$$

通过对两侧 $\boldsymbol{w}'$ 取最小值（右侧为一个二次函数），可得

$$\min_{\boldsymbol{w}'} g(\boldsymbol{w}') \leq \min_{\boldsymbol{w}'} g(\boldsymbol{w}) + \nabla^\top g(\boldsymbol{w})(\boldsymbol{w}' - \boldsymbol{w}) + \frac{M}{2}\|\boldsymbol{w}' - \boldsymbol{w}\|^2 = g(\boldsymbol{w}) - \frac{1}{2M}\|\nabla^\top g(\boldsymbol{w})\|^2. \tag{7.20}$$

因此，

$$\begin{aligned} f(\boldsymbol{w}) - f(\boldsymbol{w}') - \nabla^\top f(\boldsymbol{w}')(\boldsymbol{w} - \boldsymbol{w}') =& g(\boldsymbol{w}) - g(\boldsymbol{w}') \\ \geq& g(\boldsymbol{w}) - \min_{\boldsymbol{w}'} g(\boldsymbol{w}') \\ \geq& \frac{1}{2M}\|\nabla^\top g(\boldsymbol{w})\|^2 = \frac{1}{2M}\|\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{w}')\|^2. \end{aligned} \tag{7.21}$$

根据 $\boldsymbol{w}$ 与 $\boldsymbol{w}'$ 的对称性，同时可知

$$\begin{aligned} f(\boldsymbol{w}) - f(\boldsymbol{w}') - \nabla^\top f(\boldsymbol{w}')(\boldsymbol{w} - \boldsymbol{w}') \geq \frac{1}{2M}\|\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{w}')\|^2, \\ f(\boldsymbol{w}') - f(\boldsymbol{w}) - \nabla^\top f(\boldsymbol{w})(\boldsymbol{w}' - \boldsymbol{w}) \geq \frac{1}{2M}\|\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{w}')\|^2. \end{aligned}$$

相加可得

$$(\nabla f(\boldsymbol{w}') - \nabla f(\boldsymbol{w}))^\top (\boldsymbol{w}' - \boldsymbol{w}) \geq \frac{1}{M}\|\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{w}')\|^2.$$

□

第八章　过参数模型与隐式正则化

“道常无为，而无不为”

–老子

8.1　过参数线性回归

与传统统计分析不同，在机器学习中我们常常需要考虑过参数 (overparameterization) 背景，即模型参数量远大于样本量 $p \gg n$。例如，在神经网络中，ResNet18 （一种常见的神经网络结构）的参数量大约是 $p \approx 10^6$，而样本量往往没有这么多（例如 CIFAR-10 的样本量约为 $n \approx 10^5$）。因此，在理论分析中考虑过参数化背景十分重要。

在这里，我们考虑一个简单的过参数模型：过参数线性回归。回顾一下，在线性回归中，我们考虑 MSE 损失

$$\mathrm{MSE}(\boldsymbol{w}) = \frac{1}{n}\sum_{i=1}^{n}(\boldsymbol{w}^\top \boldsymbol{x}_i - y_i)^2$$

注意到方程组 $\boldsymbol{w}^\top \boldsymbol{x}_i - y_i = 0, i \in [n]$ 中有 p 个变量和 n 个约束方程。当 $p \gg n$ 时，该方程有无数个解能够使得 MSE 损失最小，即 $\mathrm{MSE}(\boldsymbol{w}) = 0$。因此，不同的算法会得到不同的解。在这一章中，我们将应用梯度下降算法获得一个解，并分析最终得到的解的性质。

8.2　隐式正则化

正如上文所述，过参数化背景下一般有无穷个解能够使得训练误差达到 0。然而，算法最终一般只会收敛到其中一个解，且不同的算法会获得不同的解。给定的算法会收敛到什么样的解？这个解有什么性质？这便是隐式正则化 (implicit bias) 关心的问题。

我们首先观察过参数线性回归中梯度下降的迭代

$$\boldsymbol{w}[t+1] = \boldsymbol{w}[t] - \frac{\eta}{n} X^\top (Y - X\boldsymbol{w}[t])$$

注意到：每一步的迭代均是 $X^\top v_t$ 的形式，其中 $v_t = \frac{\eta}{n}(Y - X\boldsymbol{w}[t])$ 是一个向量。由于每一步的迭代是累加的，当初始化也满足 $\boldsymbol{w}[0] = X^\top v[0]$ 形式时（例如 0 初始化），最终的训练参数 $\hat{\boldsymbol{w}}$ 一定满足形式

$$\hat{\boldsymbol{w}} = X^\top v, \tag{8.1}$$

其中 v 是一个向量。我们接下来计算 v 的形式。由于线性回归的 MSE 损失是凸函数，当梯度下降参数设置得当时，最终训练一定收敛到训练误差最小的点，即 $Y = X\hat{\boldsymbol{w}}$。将式(8.1)带入，可得

$$Y = XX^\top v.$$

注意到 $XX^\top \in \mathbb{R}^{n\times n}$ 可逆，因此 $v = (XX^\top)^{-1}Y$。重新带入式(8.1)，可得

$$\hat{\boldsymbol{w}} = X^\top (XX^\top)^{-1} Y.$$

由此，我们总结为以下结论

Claim 8.1. *在过参数线性回归 (MSE 损失) 中，如果使用 0 初始化的梯度下降法且训练误差降为 0，则最终优化到的解 $\hat{\boldsymbol{w}}$ 的形式为 $\hat{\boldsymbol{w}} = X^\top (XX^\top)^{-1} Y$。*

接下来，我们希望证明这个解是所有满足 $X\boldsymbol{w} = Y$ 的参数中的最小范数解 (min-norm solution)[1]。首先，对任意 $\boldsymbol{w}'$ 满足 $X\boldsymbol{w}' = Y$，由于已知 $X\hat{\boldsymbol{w}} = Y$，可知

$$X(\boldsymbol{w}' - \hat{\boldsymbol{w}}) = 0.$$

注意到 $\hat{\boldsymbol{w}}^\top$ 中刚好有 X 项，因此

$$\hat{\boldsymbol{w}}^\top (\boldsymbol{w}' - \hat{\boldsymbol{w}}) = 0.$$

从而，我们可知

$$\|\boldsymbol{w}'\|^2 - \|\hat{\boldsymbol{w}}\|^2 = (\boldsymbol{w}' + \hat{\boldsymbol{w}})^\top (\boldsymbol{w}' - \hat{\boldsymbol{w}}) = (\boldsymbol{w}' - \hat{\boldsymbol{w}})^\top (\boldsymbol{w}' - \hat{\boldsymbol{w}}) = \|\boldsymbol{w}' - \hat{\boldsymbol{w}}\|^2 \geq 0$$

故而 $\hat{\boldsymbol{w}}$ 为最小范数解。

[1] 在不加以强调时，这里所提到的范数均是 2 范数。

Claim 8.2. $\hat{\boldsymbol{w}} = X^\top (XX^\top)^{-1} Y$ 为集合 $\{\boldsymbol{w} : X\boldsymbol{w} = Y\}$ 中的最小范数解。

从之前分析可知，虽然我们没有显式指定任何正则条件（例如优化时加入正则项 $\|\boldsymbol{w}\|^2$），模型依然找到了一个具有特殊条件（如最小范数）的解。这样的现象，我们称之为隐式正则化 (Implicit Bias)。综上，我们得到了梯度下降在过参数线性回归中的隐式正则化，总结在定理8.3中。

定理 8.3 (过参数线性回归的隐式正则化)**.** *在过参数线性回归 (MSE 损失) 中，如果使用 0 初始化的梯度下降法且训练误差降为 0，则最终优化到的解 $\hat{\boldsymbol{w}}$ 是最小范数解，即在集合 $\{\boldsymbol{w} : X\boldsymbol{w} = Y\}$ 中，$\hat{\boldsymbol{w}}$ 具有最小的二范数。*

致谢

在本书编撰过程中，我们希望构建一个清晰的机器学习理论框架，揭示理论背后的直觉。因此，我们力求对每一个概念进行详细解释，消除读者与本书的隔阂。然而，我们也意识到，繁琐的文字和复杂的证明过程会掩盖一些显而易见的直觉。特此感谢试读会的读者们给本书提出了宝贵的意见，帮助本书在详尽与简洁之间保持平衡。

（排名不分先后）柏子杰、龚子瑄、李澜琪、梁钰姗、卢羿舟、牟思宜、闵逸洲、孙逸凡、校一皓。

参考文献

[1] SHALEV-SHWARTZ S BEN-DAVID SUnderstanding Machine Learning: From Theory to Algorithms. Cambridge University Press, 2014.

[2] BOYD S. Convex optimization. Cambridge UP, 2004.

[3] VERSHYNIN RHigh-dimensional probability: An introduction with applications in data science: vol. 47. Cambridge university press, 2018.

（微信）打赏我们