第6章 随机近似 (Stochastic Approximation)

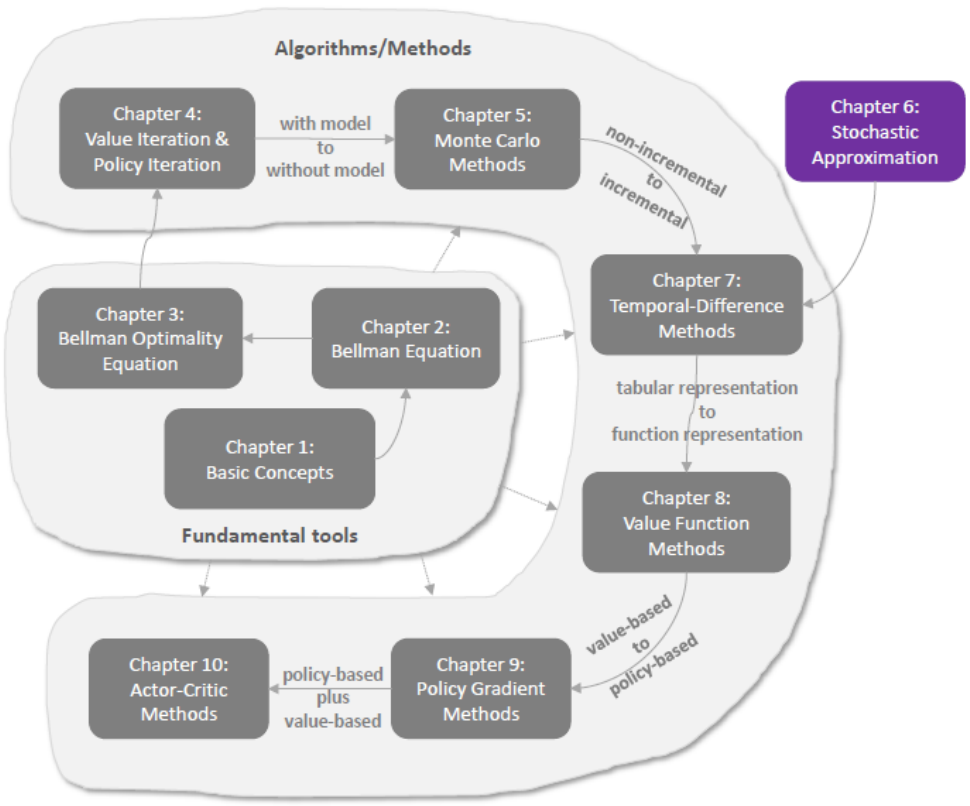

Figure 6.1: Where we are in this book.

图 6.1：本书的结构脉络。

第 5 章介绍了第一类基于蒙特卡洛估计的无模型强化学习算法。在下一章（第 7 章）中，我们将介绍另一类无模型强化学习算法：*时序差分学习 (temporal-difference learning)*。

然而，在进入下一章之前，我们需要按下暂停键，以便更好地做好准备。这是因为时序差分算法与我们迄今为止学习的算法截然不同。许多初次接触时序差分算法的读者常常会好奇，这些算法最初是如何设计出来的，以及它们为何能有效地工作。事实上，在前几章和后续章节之间存在一个*知识空白 (knowledge gap)*：我们迄今为止学习的算法是***非增量式的 (non-incremental)***，但我们将在后续章节中学习的算法是***增量式的 (incremental)***。

我们利用本章通过介绍随机逼近的基础知识来填补这一知识空白。虽然本章没有介绍任何具体的强化学习算法，但它为学习后续章节奠定了必要的基础。我们将在第 7 章看到，时序差分算法可以被视为特殊的随机逼近算法。机器学习中广泛使用的著名的随机梯度下降算法也在本章中进行了介绍。

6.1 激励性示例：均值估计 (Motivating example: Mean estimation)

接下来，我们通过考察均值估计问题来演示如何将非增量算法转换为增量算法。

考虑一个从有限集合 $\mathcal{X}$ 中取值的随机变量 X。我们的目标是估计 $\mathbb{E}[X]$。假设我们有一系列独立同分布 (i.i.d.) 的样本 $\{x_i\}_{i=1}^n$。X 的期望值可以近似为

$$\mathbb{E}[X] \approx \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \quad (6.1)$$

式 (6.1) 中的近似是第 5 章介绍的蒙特卡洛估计的基本思想。根据大数定律，我们知道当 $n \to \infty$ 时，$\bar{x} \to \mathbb{E}[X]$。

接下来我们展示可以用两种方法来计算 (6.1) 中的 $\bar{x}$。第一种*非增量 (non-incremental)* 方法首先收集所有样本，然后计算平均值。这种方法的缺点是，如果样本数量很大，我们可能必须等待很长时间直到收集完所有样本。第二种方法可以避免这个缺点，因为它以*增量 (incremental)* 方式计算平均值。具体来说，假设

$$w_{k+1} \doteq \frac{1}{k}\sum_{i=1}^{k} x_i, \quad k = 1, 2, \ldots$$

因此

$$w_k = \frac{1}{k-1}\sum_{i=1}^{k-1} x_i, \quad k = 2, 3, \dots$$

那么，w_{k+1} 可以用 w_k 表示为

$$w_{k+1} = \frac{1}{k}\sum_{i=1}^{k} x_i = \frac{1}{k}\left(\sum_{i=1}^{k-1} x_i + x_k\right) = \frac{1}{k}((k-1)w_k + x_k) = w_k - \frac{1}{k}(w_k - x_k).$$

因此，我们得到以下增量算法：

$$w_{k+1} = w_k - \frac{1}{k}(w_k - x_k). \quad (6.2)$$

该算法可用于以增量方式计算均值 $\bar{x}$ 。可以验证

$$\begin{aligned}
w_1 &= x_1, \\
w_2 &= w_1 - \frac{1}{1}(w_1 - x_1) = x_1, \\
w_3 &= w_2 - \frac{1}{2}(w_2 - x_2) = x_1 - \frac{1}{2}(x_1 - x_2) = \frac{1}{2}(x_1 + x_2), \\
w_4 &= w_3 - \frac{1}{3}(w_3 - x_3) = \frac{1}{3}(x_1 + x_2 + x_3), \\
&\vdots \\
w_{k+1} &= \frac{1}{k}\sum_{i=1}^{k} x_i. \quad (6.3)
\end{aligned}$$

式 (6.2) 的优势在于，**每当我们收到一个样本时，就可以立即计算平均值**。这个平均值可以用来近似 $\bar{x}$ ，进而近似 $\mathbb{E}[X]$ 。值得注意的是，由于样本不足，这种近似在开始时可能并不准确。然而，这总比没有好。随着获得更多样本，根据大数定律，估计精度可以逐渐提高。此外，人们也可以定义 $w_{k+1} = \frac{1}{1+k}\sum_{i=1}^{k+1} x_i$ 和 $w_k = \frac{1}{k}\sum_{i=1}^{k} x_i$ 。这样做不会产生任何显著差异。在这种情况下，相应的迭代算法为 $w_{k+1} = w_k - \frac{1}{1+k}(w_k - x_{k+1})$ 。

此外，考虑一个具有更一般表达式的算法：

$$w_{k+1} = w_k - \alpha_k(w_k - x_k). \quad (6.4)$$

该算法很重要，并且在本章中经常使用。除了系数 $1/k$ 被替换为 $\alpha_k > 0$ 之外，它与 (6.2) 相同。由于没有给出 α_k 的表达式，我们无法像 (6.3) 那样获得 w_k 的显式表达式。然而，我们将在下一节中展示，如果 $\{\alpha_k\}$ 满足一些温和的条件，当 $k \to \infty$ 时，$w_k \to \mathbb{E}[X]$ 。在第 7 章中，我们将看到时序差分算法具有相似（但更复杂）的表达式。

6.2 Robbins-Monro 算法 (Robbins-Monro algorithm)

随机逼近 (Stochastic approximation) 指的是用于**求解求根或优化问题的一大类随机迭代算法** [24]。与许多其他求根算法（如基于梯度的算法）相比，随机逼近之所以强大，是因为它不需要目标函数的表达式或其导数。

Robbins-Monro (RM) 算法是随机逼近领域的开创性工作 [24–27]。著名的随机梯度下降算法是 RM 算法的一种特殊形式，如 6.4 节所示。接下来我们介绍 RM 算法的细节。

假设我们要寻找方程

$$g(w) = 0,$$

的根，**其中 $w \in \mathbb{R}$ 是未知变量**，$g : \mathbb{R} \to \mathbb{R}$ 是一个函数。

许多问题都可以表述为求根问题。例如，如果 $J(w)$ 是要优化的目标函数，那么这个优化问题可以转换为求解

$$g(w) \doteq \nabla_w J(w) = 0 \text{。}$$

此外，像 $g(w) = c$ （其中 c 是常数）这样的方程也可以通过将 $g(w) - c$ 重写为一个新函数来转换为上述方程。

如果已知 g 的表达式或其导数，可以使用许多数值算法。**然而，我们面临的问题是函数 g 的表达式是未知的。**例如，该函数可能由一个结构和参数未知的人工神经网络表示。此外，**我们只能获得 $g(w)$ 的含噪观测值**：

$$\tilde{g}(w, \eta) = g(w) + \eta,$$

其中 $\eta \in \mathbb{R}$ 是观测误差，它可能不是高斯的。总之，**这是一个黑盒系统，只有输入 w 和含噪输出 $\tilde{g}(w,\eta)$ 是已知的（见图 6.2）**。我们的目标是使用 w 和 $\tilde{g}$ 求解 $g(w)=0$。

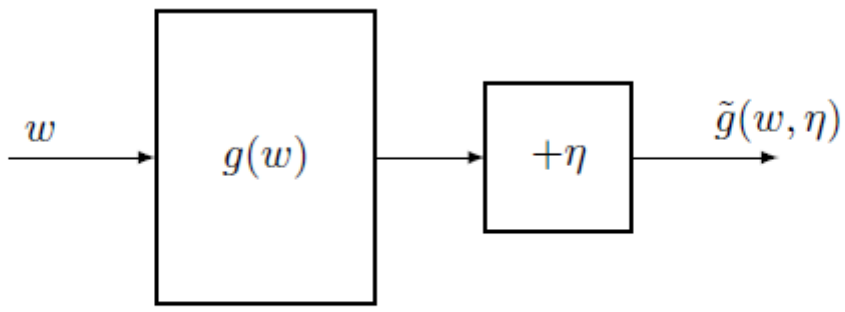

Figure 6.2: An illustration of the problem of solving $g(w) = 0$ from w and $\tilde{g}$.

图 6.2：利用 w 和 $\tilde{g}$ 求解 $g(w)=0$ 问题的图示。

可以求解 $g(w)=0$ 的 RM 算法为

$$w_{k+1} = w_k - a_k \tilde{g}(w_k, \eta_k), \quad k = 1, 2, 3, \dots \tag{6.5}$$

其中 w_k 是根的第 k 次估计值，$\tilde{g}(w_k,\eta_k)$ 是第 k 次含噪观测值，a_k 是一个正系数。可以看出，RM 算法不需要关于函数的任何信息。它只需要输入和输出。

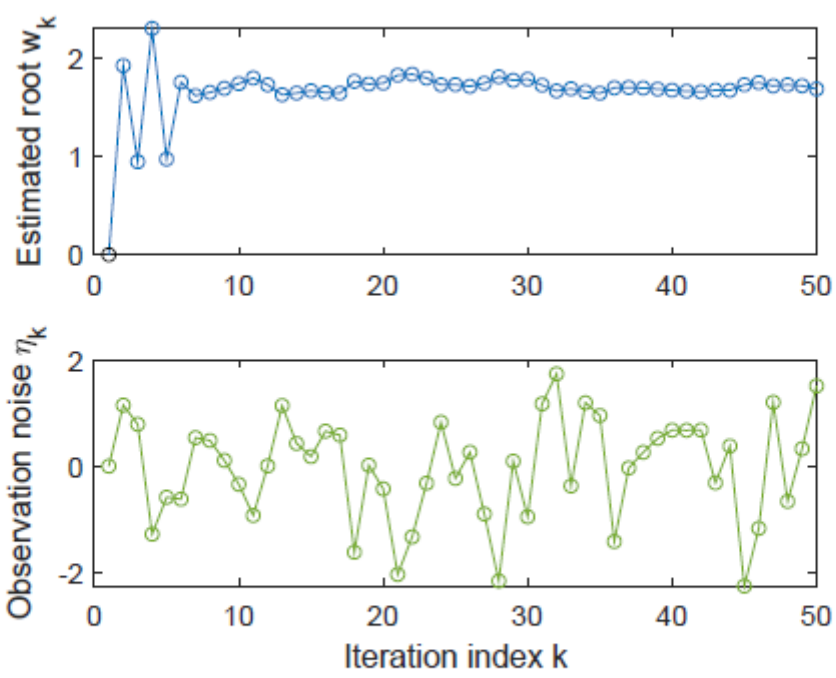

Figure 6.3: An illustrative example of the RM algorithm.

图 6.3：RM 算法的一个说明性示例。

为了说明 RM 算法，考虑一个例子，其中 $g(w) = w^3 - 5$。真实的根为 $5^{1/3} \approx 1.71$。现在，假设我们只能观测到输入 w 和输出 $\tilde{g}(w) = g(w) + \eta$，其中 η 是独立同分布的 (i.i.d.) 且服从均值为零、标准差为 1 的标准正态分布。初始猜测为 $w_1 = 0$，系数为 $a_k = 1/k$。w_k 的演变过程如图 6.3 所示。尽管观测值受到噪声 η_k 的干扰，估计值 w_k 仍然可以收敛到真实的根。**需要注意的是，对于特定的函数 $g(w) = w^3 - 5$，必须恰当地选择初始猜测 w_1 以确保收敛。**在下一小节中，我们将介绍 RM 算法在任意初始猜测下都能收敛的条件。

6.2.1 收敛性质 (Convergence properties)

为什么式 (6.5) 中的 RM 算法能找到 $g(w)=0$ 的根？ 接下来我们用一个例子来说明这个思想，然后提供严格的收敛性分析。

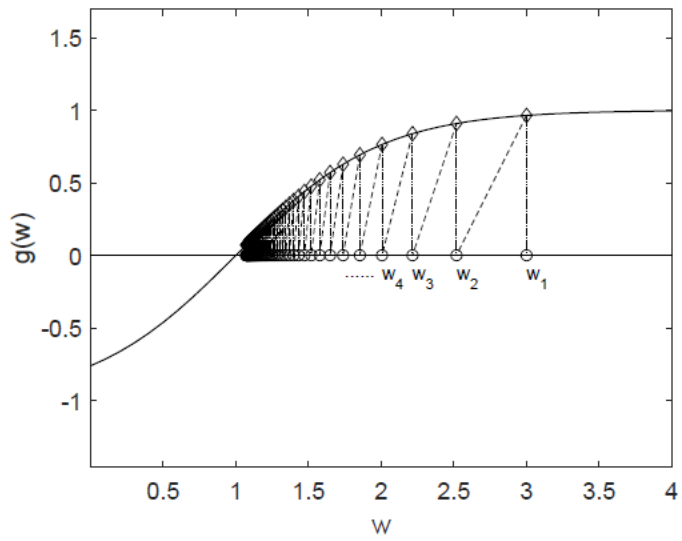

Figure 6.4: An example for illustrating the convergence of the RM algorithm.

图 6.4：说明 RM 算法收敛性的一个例子。

考虑图 6.4 所示的例子。在这个例子中，$g(w) = \tanh(w-1)$。$g(w)=0$ 的真实根是 $w^*=1$。我们应用 RM 算法，设定 $w_1=3$ 和 $a_k=1/k$。为了更好地说明收敛的原因，我们简单地设定 $\eta_k \equiv 0$，因此 $\tilde{g}(w_k,\eta_k)=g(w_k)$。这种情况下的 RM 算法为 $w_{k+1}=w_k-a_k g(w_k)$。RM 算法生成的 $\{w_k\}$ 如图 6.4 所示。可以看出 w_k 收敛到真实的根 $w^*=1$。

这个简单的例子可以说明为什么 RM 算法会收敛。

- 当 $w_k > w^*$ 时，我们有 $g(w_k)>0$。于是，$w_{k+1}=w_k-a_k g(w_k)<w_k$。如果 $a_k g(w_k)$ 足够小，我们有 $w^*<w_{k+1}<w_k$。结果是，w_{k+1} 比 w_k 更接近 w^*。
- 当 $w_k < w^*$ 时，我们有 $g(w_k)<0$。于是，$w_{k+1}=w_k-a_k g(w_k)>w_k$。如果 $|a_k g(w_k)|$ 足够小，我们有 $w^*>w_{k+1}>w_k$。结果是，w_{k+1} 比 w_k 更接近 w^*。

在这两种情况下，w_{k+1} 都比 w_k 更接近 w^*。因此，直观上 w_k 会收敛到 w^*。

上述例子很简单，因为假设观测误差为零。在存在随机观测误差的情况下分析收敛性并非易事。下面给出了严格的收敛性结果。

定理 6.1 (Robbins-Monro 定理)。*在 (6.5) 中的 Robbins-Monro 算法中，如果*

(a) 对于所有 w，有 $0<c_1 \le \nabla_w g(w) \le c_2$；

(b) $\sum_{k=1}^{\infty} a_k = \infty$ 且 $\sum_{k=1}^{\infty} a_k^2 < \infty$；

(c) $\mathbb{E}[\eta_k|\mathcal{H}_k]=0$ 且 $\mathbb{E}[\eta_k^2|\mathcal{H}_k]<\infty$；

其中 $\mathcal{H}_k=\{w_k, w_{k-1},\dots\}$，那么 w_k 几乎处处 (almost surely) 收敛到满足 $g(w^*)=0$ 的根 w^*。

我们将该定理的证明推迟到 6.3.3 节。该定理依赖于*几乎处处收敛 (almost sure convergence)* 的概念，该概念在附录 B 中介绍。

定理 6.1 中的三个条件解释如下。

- 在第一个条件中，$0<c_1 \le \nabla_w g(w)$ 表明 $g(w)$ 是一个单调递增函数。
  - 这个条件确保了 $g(w)=0$ 的根存在且唯一。如果 $g(w)$ 是单调递减的，我们可以简单地将 $-g(w)$ 视为一个新的单调递增函数。
  - 作为一个应用，我们可以将一个目标函数为 $J(w)$ 的优化问题表述为一个求根问题：$g(w) \doteq \nabla_w J(w)=0$。在这种情况下，$g(w)$ 单调递增的条件表明 $J(w)$ 是*凸 (convex)* 的，这是优化问题中普遍采用的一个假设。
  - 不等式 $\nabla_w g(w) \le c_2$ 表明 $g(w)$ 的梯度是有上界的。例如，$g(w)=\tanh(w-1)$ 满足这个条件，但 $g(w)=w^3-5$ 不满足。
- 关于 $\{a_k\}$ 的第二个条件很有趣。我们经常在强化学习算法中看到这样的条件。
  - 具体来说，条件 $\sum_{k=1}^{\infty} a_k^2 < \infty$ 意味着 $\lim_{n\to\infty}\sum_{k=1}^{n} a_k^2$ 是有上界的。它要求当 $k\to\infty$ 时 a_k 收敛于零。条件 $\sum_{k=1}^{\infty} a_k = \infty$ 意味着 $\lim_{n\to\infty}\sum_{k=1}^{n} a_k$ 是无限大的。它要求 a_k 收敛于零的速度不能太快。这些条件具有有趣的性质，稍后将对此进行详细分析。

- 第三个条件很温和。它并不要求观测误差 η_k 是高斯的。
  - 一个重要的特例是 $\{\eta_k\}$ 是一个满足 $\mathbb{E}[\eta_k]=0$ 和 $\mathbb{E}[\eta_k^2]<\infty$ 的独立同分布 (i.i.d.) 随机序列。
  - 在这种情况下，第三个条件成立，因为 η_k 独立于 $\mathcal{H}_k$ ，因此我们有 $\mathbb{E}[\eta_k|\mathcal{H}_k]=\mathbb{E}[\eta_k]=0$ 且 $\mathbb{E}[\eta_k^2|\mathcal{H}_k]=\mathbb{E}[\eta_k^2]$。

接下来我们更仔细地考察关于系数 $\{a_k\}$ 的第二个条件。

- 为什么第二个条件对于 RM 算法的收敛很重要?
  - 当我们稍后给出上述定理的严格证明时，这个问题自然会得到回答。在这里，我们想提供一些有见地的直观解释。
  - 首先，$\sum_{k=1}^{\infty}a_k^2<\infty$ 表明当 $k\to\infty$ 时 $a_k\to 0$。为什么这个条件很重要？假设观测值 $\tilde{g}(w_k,\eta_k)$ 总是有界的。由于

$$w_{k+1}-w_k=-a_k\tilde{g}(w_k,\eta_k),$$

    如果 $a_k\to 0$，那么 $a_k\tilde{g}(w_k,\eta_k)\to 0$，因此 $w_{k+1}-w_k\to 0$，这表明当 $k\to\infty$ 时 w_{k+1} 和 w_k 彼此接近。**否则，如果 a_k 不收敛，那么当 $k\to\infty$ 时 w_k 可能仍然会波动**。

  - 其次，$\sum_{k=1}^{\infty}a_k=\infty$ 表明 a_k 收敛到零的速度不应太快。为什么这个条件很重要？对 $w_2-w_1=-a_1\tilde{g}(w_1,\eta_1), w_3-w_2=-a_2\tilde{g}(w_2,\eta_2), w_4-w_3=-a_3\tilde{g}(w_3,\eta_3),\dots$ 等方程两边求和可得

$$w_1-w_\infty=\sum_{k=1}^{\infty}a_k\tilde{g}(w_k,\eta_k).$$

    如果 $\sum_{k=1}^{\infty}a_k<\infty$，那么 $|\sum_{k=1}^{\infty}a_k\tilde{g}(w_k,\eta_k)|$ 也是有界的。设 b 为有限上界，使得

$$|w_1-w_\infty|=\left|\sum_{k=1}^{\infty}a_k\tilde{g}(w_k,\eta_k)\right|\le b. \quad (6.6)$$

    如果初始猜测 w_1 选择得离 w^* 很远，使得 $|w_1-w^*|>b$，那么根据 (6.6) 式，不可能有 $w_\infty=w^*$。这表明在这种情况下 RM 算法无法找到真实解 w^*。因此，条件 $\sum_{k=1}^{\infty}a_k=\infty$ 对于确保在给定*任意*初始猜测下的收敛性是必要的。

- 什么样的序列满足 $\sum_{k=1}^{\infty}a_k=\infty$ 和 $\sum_{k=1}^{\infty}a_k^2<\infty$？

  一个典型的序列是

$$a_k=\frac{1}{k}.$$

  一方面，成立的是

$$\lim_{n\to\infty}\left(\sum_{k=1}^{n}\frac{1}{k}-\ln n\right)=\kappa,$$

  其中 $\kappa\approx 0.577$ 被称为欧拉-马歇罗尼常数（或欧拉常数）[28]。由于当 $n\to\infty$ 时 $\ln n\to\infty$，我们有

$$\sum_{k=1}^{\infty}\frac{1}{k}=\infty.$$

  事实上，$H_n=\sum_{k=1}^{n}\frac{1}{k}$ 在数论中被称为调和数 [29]。另一方面，成立的是

$$\sum_{k=1}^{\infty}\frac{1}{k^2}=\frac{\pi^2}{6}<\infty.$$

  求解 $\sum_{k=1}^{\infty}\frac{1}{k^2}$ 的值被称为巴塞尔问题 [30]。

总之，序列 $\{a_k=1/k\}$ 满足定理 6.1 中的第二个条件。值得注意的是，稍作修改，例如 $a_k=1/(k+1)$ 或 $a_k=c_k/k$（其中 c_k 是有界的），也保持了这一条件。

在 RM 算法中，**在许多应用中** a_k **通常被选为一个足够小的** ***常数 (constant)***。尽管在这种情况下不再满足第二个条件，因为 $\sum_{k=1}^{\infty} a_k^2 = \infty$ 而不是 $\sum_{k=1}^{\infty} a_k^2 < \infty$，但算法仍然可以在某种意义上收敛 [24, Section 1.5]。此外，图 6.3 所示示例中的 $g(x) = x^3 - 5$ 不满足第一个条件，但如果初始猜测选择得当（而非任意选择），RM 算法仍然可以找到根。

6.2.2 应用于均值估计 (Application to mean estimation)

接下来，我们要应用 Robbins-Monro 定理来分析均值估计问题，该问题已在 6.1 节中讨论过。回想一下

$$w_{k+1} = w_k + \alpha_k (x_k - w_k)$$

是式 (6.4) 中的均值估计算法。当 $\alpha_k = 1/k$ 时，我们可以获得 w_{k+1} 的解析表达式为 $w_{k+1} = 1/k \sum_{i=1}^{k} x_i$。然而，当给定 α_k 的一般值时，我们将无法获得解析表达式。在这种情况下，收敛性分析是不平凡的。我们可以证明，这种情况下的算法是一种特殊的 RM算法，因此其收敛性自然成立。

> 💡 **注RM算法的一般表达式：** $w_{k+1} = w_k - a_k \tilde{g}(w_k, \eta_k), \quad k = 1, 2, 3, \ldots \quad (6.5)$

特别地，定义一个函数为

$$g(w) \doteq w - \mathbb{E}[X].$$

原始问题是获取 $\mathbb{E}[X]$ 的值。这个问题被表述为一个求解 $g(w) = 0$ 的求根问题。给定一个 w 值，我们可以获得的含噪观测是 $\tilde{g} \doteq w - x$，其中 x 是 X 的一个样本。注意 $\tilde{g}$ 可以写成

$$\begin{aligned} \tilde{g}(w, \eta) &= w - x \\ &= w - x + \mathbb{E}[X] - \mathbb{E}[X] \\ &= (w - \mathbb{E}[X]) + (\mathbb{E}[X] - x) \doteq g(w) + \eta, \end{aligned}$$

其中 $\eta \doteq \mathbb{E}[X] - x$。

用于求解该问题的 RM 算法是

$$w_{k+1} = w_k - \alpha_k \tilde{g}(w_k, \eta_k) = w_k - \alpha_k (w_k - x_k),$$

这正是式 (6.4) 中的算法。结果是，定理 6.1 保证了如果 $\sum_{k=1}^{\infty} \alpha_k = \infty$，$\sum_{k=1}^{\infty} \alpha_k^2 < \infty$，且 $\{x_k\}$ 是独立同分布 (i.i.d.) 的，则 w_k 几乎处处 (almost surely) 收敛到 $\mathbb{E}[X]$。值得一提的是，该收敛性质不依赖于关于 X 分布的任何假设。

6.3 Dvoretzky 收敛定理 (Dvoretzky’ s convergence theorem)

到目前为止，RM 算法的收敛性尚未得到证明。为了做到这一点，我们接下来介绍 Dvoretzky 定理 [31, 32]，这是随机逼近领域的一个经典结果。该定理可用于分析 RM 算法和许多强化学习算法的收敛性。

本节的数学强度略大。推荐对随机算法收敛性分析感兴趣的读者学习本节。否则，可以跳过本节。

定理 6.2 (Dvoretzky 定理)。考虑一个随机过程

$$\Delta_{k+1} = (1 - \alpha_k)\Delta_k + \beta_k \eta_k,$$

其中 $\{\alpha_k\}_{k=1}^{\infty}, \{\beta_k\}_{k=1}^{\infty}, \{\eta_k\}_{k=1}^{\infty}$ **是随机序列。这里对所有** k **都有** $\alpha_k \geq 0, \beta_k \geq 0$**。那么，如果满足以下条件，** Δ_k **几乎处处收敛于零：**

(a) $\sum_{k=1}^{\infty} \alpha_k = \infty, \sum_{k=1}^{\infty} \alpha_k^2 < \infty$, 且 $\sum_{k=1}^{\infty} \beta_k^2 < \infty$ 一致几乎处处 (uniformly almost surely) 成立；

(b) $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$ 且 $\mathbb{E}[\eta_k^2 | \mathcal{H}_k] \leq C$ 几乎处处 (almost surely) 成立；

其中 $\mathcal{H}_k = \{\Delta_k, \Delta_{k-1}, \ldots, \eta_{k-1}, \ldots, \alpha_{k-1}, \ldots, \beta_{k-1}, \ldots\}$。

在给出该定理的证明之前，我们先澄清一些问题。

- 在 RM 算法中，系数序列 $\{\alpha_k\}$ 是确定性的。然而，Dvoretzky 定理允许 $\{\alpha_k\}, \{\beta_k\}$ 为依赖于 $\mathcal{H}_k$ 的随机变量。因此，在 α_k 或 β_k 是 Δ_k 的函数的情况下，这更为有用。

- 在第一个条件中，陈述为“一致几乎处处”。这是因为 α_k 和 β_k 可能是随机变量，因此它们极限的定义必须是在随机意义下的。在第二个条件中，也陈述为“几乎处处”。这是因为 $\mathcal{H}_k$ 是一个随机变量序列，而不是特定的数值。因此，$\mathbb{E}[\eta_k|\mathcal{H}_k]$ 和 $\mathbb{E}[\eta_k^2|\mathcal{H}_k]$ 是随机变量。这种情况下的条件期望的定义是在“几乎处处”的意义下的（附录 B）。
- 定理 6.2 的陈述与 [32] 略有不同，因为定理 6.2 在第一个条件中不需要 $\sum_{k=1}^{\infty}\beta_k = \infty$。当 $\sum_{k=1}^{\infty}\beta_k < \infty$ 时，特别是在所有 k 都有 $\beta_k = 0$ 的极端情况下，序列仍然可以收敛。

🎉 它研究一个**带噪声的递推过程**：

$$\Delta_{k+1} = (1-\alpha_k)\Delta_k + \beta_k\eta_k$$

你可以把它想成：

- (Δ_k)：第 (k) 步的**误差**（比如“离目标还差多少”）
- ($(1-\alpha_k)\Delta_k$)：每一步把误差**往 0 拉回去一点**（收缩/纠正）
- ($\beta_k\eta_k$)：但每一步又会被**随机噪声**推一下（扰动）

定理结论：只要满足一些“拉回去的力量足够、噪声别太大”的条件，那么误差 (\Delta_k) 会 **几乎必然（几乎处处）收敛到 0**。

“几乎处处/几乎必然（almost surely）”的意思是：
在随机性导致的所有可能结果里，**除了概率为 0 的极少数倒霉情况**，其他情况下都收敛到 0。

条件 (a) 和 (b) 分别在控制什么？

(a) 关于 (α_k, β_k) 的“大小规则”

图里写的是：

- $\sum_{k=1}^{\infty}\alpha_k = \infty$
  纠错的总力度要够：虽然每一步可能越来越小，但“拉回 0”的累计效果不能太弱。
  （直觉：如果你纠错太快衰减，后面就拉不动了。）
- $\sum_{k=1}^{\infty}\alpha_k^2 < \infty$
  每一步不要太猛，而且要衰减得够快（技术上用来保证一些随机项可控）。
  常见学习率 $\alpha_k = 1/k$ 就满足：$\sum 1/k = \infty$，$\sum 1/k^2 < \infty$。
- $\sum_{k=1}^{\infty}\beta_k^2 < \infty$
  噪声的“能量”总和要有限：噪声虽然一直有，但它被 (\beta_k) 缩小得足够快，累计起来不会把系统一直炸飞。

> 图中提到“一致几乎处处（uniformly a.s.）”，主要是因为 (α_k, β_k) 可能本身也是随机的（甚至依赖过去历史），所以这些级数收敛/发散的结论要在“几乎每一条随机样本路径上”都成立。

(b) 关于噪声 (η_k) 的“性质”

- $\mathbb{E}[\eta_k \mid \mathcal{H}_k] = 0$
  意思是：在已知过去所有信息 $\mathcal{H}_k$ 的情况下，噪声**平均不偏**（不系统性地一直往某个方向推）。
  这就是“**鞅差（martingale difference）**”的典型条件。
- $\mathbb{E}[\eta_k^2 \mid \mathcal{H}_k] \le C$
  意思是：噪声的方差（强度）**不至于无限大**，有个上界。

$\mathcal{H}_k$ 就是“到第 (k) 步为止你知道的一切”（过去的 Δ、过去的噪声、过去的步长等等）。

直觉总结成一句话

这个定理就是在说：

> **只要你一直在认真纠错（$\sum \alpha_k = \infty$），同时纠错步子在变小且噪声也被压小（$\sum \alpha_k^2 < \infty, \sum \beta_k^2 < \infty$），并且噪声不带偏、强度有限（条件 b），那么误差最终会被“磨”到 0。**

这个定理有什么作用?

它是随机算法里非常常用的“收敛工具”，典型用途：

1. **证明随机逼近/迭代算法收敛**
   比如 Robbins–Monro 随机逼近、很多在线估计方法。
2. **理解/证明 SGD（随机梯度下降）为什么能收敛（在合适条件下）**
   SGD 的误差在很多情况下可以近似写成类似
   $\Delta_{k+1} \approx (1 - \alpha_k \cdot 某个正数)\Delta_k + \alpha_k \cdot (随机噪声)$，
   于是用这类定理就能推出 ($\Delta_k \to 0$)（或收敛到最优点附近）。
3. **允许步长是“随机的、依赖历史的”**
   图里也强调了： α_k, β_k 不一定是固定公式，也可以依赖 $\mathcal{H}_k$ （比如根据当前误差自适应调整），这让它在分析更复杂算法时很有用。

6.3.1 Dvoretzky 定理的证明 (Proof of Dvoretzky’ s theorem)

Dvoretzky 定理的原始证明是在 1956 年给出的 [31]。也有其他的证明。接下来我们给出一个基于*拟鞅 (quasimartingales)* 的证明。有了拟鞅的收敛结果，Dvoretzky 定理的证明就变得直截了当了。关于拟鞅的更多信息可以在附录 C 中找到。

Dvoretzky 定理的证明。 令 $h_k \doteq \Delta_k^2$ 。那么，

$$\begin{aligned} h_{k+1} - h_k &= \Delta_{k+1}^2 - \Delta_k^2 \\ &= (\Delta_{k+1} - \Delta_k)(\Delta_{k+1} + \Delta_k) \\ &= (-\alpha_k \Delta_k + \beta_k \eta_k)[(2 - \alpha_k)\Delta_k + \beta_k \eta_k] \\ &= -\alpha_k(2 - \alpha_k)\Delta_k^2 + \beta_k^2 \eta_k^2 + 2(1 - \alpha_k)\beta_k \eta_k \Delta_k . \end{aligned}$$

对上述方程两边取期望可得

$$\mathbb{E}[h_{k+1} - h_k | \mathcal{H}_k] = \mathbb{E}[-\alpha_k(2 - \alpha_k)\Delta_k^2 | \mathcal{H}_k] + \mathbb{E}[\beta_k^2 \eta_k^2 | \mathcal{H}_k] + \mathbb{E}[2(1 - \alpha_k)\beta_k \eta_k \Delta_k | \mathcal{H}_k]. \quad (6.7)$$

首先，由于 Δ_k 包含在 $\mathcal{H}_k$ 中并由其确定，**因此可以将其从期望中提取出来（见引理 B.1 中的性质 (e)）**。

其次，考虑简单的情况, 其中 α_k, β_k 由 $\mathcal{H}_k$ 决定。这种情况在（例如）$\{\alpha_k\}$ 和 $\{\beta_k\}$ 是 Δ_k 的函数或确定性序列时是有效的。那么，它们也可以从期望中提取出来。因此，(6.7) 变为

$$\mathbb{E}[h_{k+1} - h_k | \mathcal{H}_k] = -\alpha_k(2 - \alpha_k)\Delta_k^2 + \beta_k^2 \mathbb{E}[\eta_k^2 | \mathcal{H}_k] + 2(1 - \alpha_k)\beta_k \Delta_k \mathbb{E}[\eta_k | \mathcal{H}_k]. \quad (6.8)$$

对于第一项，由于 $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$ 意味着 $\alpha_k \to 0$ 几乎处处成立，因此存在一个有限的 n ，使得对于所有 $k \geq n$ ， $\alpha_k \leq 1$ 几乎处处成立。不失一般性，我们接下来仅考虑 $\alpha_k \leq 1$ 的情况。那么， $-\alpha_k(2 - \alpha_k)\Delta_k^2 \leq 0$ 。

对于第二项，根据假设我们有 $\beta_k^2 \mathbb{E}[\eta_k^2 | \mathcal{H}_k] \leq \beta_k^2 C$ 。

第三项等于零，因为根据假设 $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$ 。因此，(6.8) 变为

$$\mathbb{E}[h_{k+1} - h_k | \mathcal{H}_k] = -\alpha_k(2 - \alpha_k)\Delta_k^2 + \beta_k^2 \mathbb{E}[\eta_k^2 | \mathcal{H}_k] \leq \beta_k^2 C, \quad (6.9)$$

因此

$$\sum_{k=1}^{\infty} \mathbb{E}[h_{k+1} - h_k | \mathcal{H}_k] \leq \sum_{k=1}^{\infty} \beta_k^2 C < \infty.$$

最后一个不等式是由于条件 $\sum_{k=1}^{\infty} \beta_k^2 < \infty$ 。然后，**基于附录 C 中的拟鞅收敛定理 (quasimartingale convergence theorem)**，我们得出结论： h_k 几乎处处收敛。

接下来我们确定 Δ_k 收敛到什么值。由 (6.9) 可知

$$\sum_{k=1}^{\infty}\alpha_k(2-\alpha_k)\Delta_k^2 = \sum_{k=1}^{\infty}\beta_k^2\mathbb{E}[\eta_k^2|\mathcal{H}_k] - \sum_{k=1}^{\infty}\mathbb{E}[h_{k+1}-h_k|\mathcal{H}_k].$$

右边的第一项根据假设是有界的。第二项也是有界的，因为 h_k 收敛，因此 $h_{k+1}-h_k$ 是可求和的。因此，左边的 $\sum_{k=1}^{\infty}\alpha_k(2-\alpha_k)\Delta_k^2$ 也是有界的。由于我们考虑 $\alpha_k \le 1$ 的情况，我们有

$$\infty > \sum_{k=1}^{\infty}\alpha_k(2-\alpha_k)\Delta_k^2 \ge \sum_{k=1}^{\infty}\alpha_k\Delta_k^2 \ge 0.$$

因此，$\sum_{k=1}^{\infty}\alpha_k\Delta_k^2$ 是有界的。由于 $\sum_{k=1}^{\infty}\alpha_k = \infty$，我们必然有 $\Delta_k \to 0$ 几乎处处成立。

6.3.2 应用于均值估计 (Application to mean estimation)

虽然均值估计算法 $w_{k+1} = w_k + \alpha_k(x_k - w_k)$ 已经使用 RM 定理进行了分析，但我们接下来展示其收敛性也可以直接通过 Dvoretzky 定理来证明。

证明。令 $w^* = \mathbb{E}[X]$。均值估计算法 $w_{k+1} = w_k + \alpha_k(x_k - w_k)$ 可以重写为

$$w_{k+1} - w^* = w_k - w^* + \alpha_k(x_k - w^* + w^* - w_k).$$

令 $\Delta \doteq w - w^*$。那么，我们有

$$\begin{aligned}\Delta_{k+1} &= \Delta_k + \alpha_k(x_k - w^* - \Delta_k)\\ &= (1-\alpha_k)\Delta_k + \alpha_k\underbrace{(x_k - w^*)}_{\eta_k}.\end{aligned}$$

由于 $\{x_k\}$ 是独立同分布的 (i.i.d.)，我们有 $\mathbb{E}[x_k|\mathcal{H}_k] = \mathbb{E}[x_k] = w^*$。结果是，$\mathbb{E}[\eta_k|\mathcal{H}_k] = \mathbb{E}[x_k - w^*|\mathcal{H}_k] = 0$ 且 $\mathbb{E}[\eta_k^2|\mathcal{H}_k] = \mathbb{E}[x_k^2|\mathcal{H}_k] - (w^*)^2 = \mathbb{E}[x_k^2] - (w^*)^2$，**如果 x_k 的方差是有限的，则这些项是有界的。遵循 Dvoretzky 定理，我们得出结论：Δ_k 收敛于零，因此 w_k 几乎处处收敛于 $w^* = \mathbb{E}[X]$。**

6.3.3 应用于 Robbins-Monro 定理 (Application to the Robbins-Monro theorem)

我们现在准备使用 Dvoretzky 定理来证明 Robbins-Monro 定理。

Robbins-Monro 定理的证明。RM 算法旨在寻找 $g(w) = 0$ 的根。假设根为 w^*，使得 $g(w^*) = 0$。RM 算法为

$$\begin{aligned}w_{k+1} &= w_k - a_k\tilde{g}(w_k, \eta_k)\\ &= w_k - a_k[g(w_k) + \eta_k].\end{aligned}$$

那么，我们有

$$w_{k+1} - w^* = w_k - w^* - a_k[g(w_k) - g(w^*) + \eta_k].$$

根据中值定理 (mean value theorem) [7,8]，我们有 $g(w_k) - g(w^*) = \nabla_w g(w'_k)(w_k - w^*)$，

其中 $w'_k \in [w_k, w^*]$。令 $\Delta_k \doteq w_k - w^*$。上述方程变为

$$\begin{aligned}\Delta_{k+1} &= \Delta_k - a_k[\nabla_w g(w'_k)(w_k - w^*) + \eta_k]\\ &= \Delta_k - a_k\nabla_w g(w'_k)\Delta_k + a_k(-\eta_k)\\ &= \underbrace{[1 - a_k\nabla_w g(w'_k)]}_{\alpha_k}\Delta_k + a_k(-\eta_k).\end{aligned}$$

注意 $\nabla_w g(w)$ 是有界的，因为假设了 $0 < c_1 \le \nabla_w g(w) \le c_2$。由于假设了 $\sum_{k=1}^{\infty}a_k = \infty$ 和 $\sum_{k=1}^{\infty}a_k^2 < \infty$，我们知道 $\sum_{k=1}^{\infty}\alpha_k = \infty$ 和 $\sum_{k=1}^{\infty}\alpha_k^2 < \infty$。因此，Dvoretzky 定理的所有条件都满足，故 Δ_k 几乎处处收敛于零。

RM 定理的证明展示了 Dvoretzky 定理的威力。特别是，证明中的 α_k 是一个依赖于 w_k 的随机序列，而不是一个确定性序列。在这种情况下，Dvoretzky 定理仍然适用。

6.3.4 Dvoretzky 定理的一个扩展 (An extension of Dvoretzky’s theorem)

接下来，我们将 Dvoretzky 定理扩展为一个更通用的定理，它可以处理多个变量。这个由 [32] 提出的通用定理可以用来分析随机迭代算法（如 Q-learning）的收敛性。

定理 6.3。*考虑一个实数的有限集合 $\mathcal{S}$。对于随机过程*

$$\Delta_{k+1}(s) = (1 - \alpha_k(s))\Delta_k(s) + \beta_k(s)\eta_k(s),$$

> 🏖️ 注： $s \in \mathcal{S}$ **可以视为有限索引集合**（比如 $1, 2, \ldots, n\{1, 2, \ldots, n\}1, 2, \ldots, n$ 这种）
>
> $\Delta_k(s)$ 表示第 k步时，对索引 s 这个分量的误差/量
>
> $\alpha_k(s)$、$\beta_k(s)$、$\eta_k(s)$ 都是**带索引** s 的随机序列/随机变量。

如果对于 $s \in \mathcal{S}$ 满足以下条件，则对于每一个 $s \in \mathcal{S}$，$\Delta_k(s)$ 几乎处处收敛于零：

(a) $\sum_k \alpha_k(s) = \infty, \sum_k \alpha_k^2(s) < \infty, \sum_k \beta_k^2(s) < \infty$，且 $\mathbb{E}[\beta_k(s)|\mathcal{H}_k] \leq \mathbb{E}[\alpha_k(s)|\mathcal{H}_k]$ 一致几乎处处成立；

(b) $\|\mathbb{E}[\eta_k(s)|\mathcal{H}_k]\|_\infty \leq \gamma\|\Delta_k\|_\infty$，其中 $\gamma \in (0, 1)$；

(c) $\text{var}[\eta_k(s)|\mathcal{H}_k] \leq C(1 + \|\Delta_k(s)\|_\infty)^2$，其中 C 是一个常数。

这里，$\mathcal{H}_k = \{\Delta_k, \Delta_{k-1}, \ldots, \eta_{k-1}, \ldots, \alpha_{k-1}, \ldots, \beta_{k-1}, \ldots\}$ 表示历史信息。项 $\|\cdot\|_\infty$ 指的是最大范数。

证明。作为一个扩展，该定理可以基于 Dvoretzky 定理来证明。详细信息可以在 [32] 中找到，此处省略。 □

下面给出了关于该定理的一些注记。

- 我们首先澄清定理中的一些符号。**变量 s 可以被看作是一个索引。在强化学习的背景下，它表示一个状态或状态-动作对。**最大范数 $\|\cdot\|_\infty$ 是在一个集合上定义的。它类似于但不同于向量的 L^∞ 范数。特别地，$\|\mathbb{E}[\eta_k(s)|\mathcal{H}_k]\|_\infty \doteq \max_{s\in\mathcal{S}} |\mathbb{E}[\eta_k(s)|\mathcal{H}_k]|$ 且 $\|\Delta_k(s)\|_\infty \doteq \max_{s\in\mathcal{S}} |\Delta_k(s)|$。

> 🏖️ $\|\Delta_k(s)\|_\infty \doteq \max_{s\in\mathcal{S}} |\Delta_k(s)|$ 所有分量里最大的误差

- 这个定理比 Dvoretzky 定理更通用。首先，由于最大范数操作，它可以处理多变量的情况。这对于存在多个状态的强化学习问题很重要。其次，虽然 Dvoretzky 定理要求 $\mathbb{E}[\eta_k(s)|\mathcal{H}_k] = 0$ 且 $\text{var}[\eta_k(s)|\mathcal{H}_k] \leq C$，但该定理仅要求期望和方差被误差 Δ_k 有界。
- 需要注意的是，对于所有 $s \in \mathcal{S}$，$\Delta(s)$ 的收敛要求条件对每一个 $s \in \mathcal{S}$ 都有效。因此，在应用此定理证明强化学习算法的收敛性时，我们需要证明条件对每一个状态（或状态-动作对）都有效。

6.4 随机梯度下降 (Stochastic gradient descent)

本节介绍随机梯度下降 (SGD) 算法，该算法在机器学习领域被广泛使用。我们将看到 SGD 是一种特殊的 RM 算法，而均值估计算法是一种特殊的 SGD 算法。

考虑以下优化问题：

$$\min_w J(w) = \mathbb{E}[f(w, X)], \quad (6.10)$$

其中 w 是要优化的参数，X 是一个随机变量。期望是关于 X 计算的。这里，w 和 X 可以是标量或向量。函数 $f(\cdot)$ 是一个标量。

求解 (6.10) 的一种直接方法是*梯度下降 (gradient descent)*。特别地，$\mathbb{E}[f(w, X)]$ 的梯度为 $\nabla_w \mathbb{E}[f(w, X)] = \mathbb{E}[\nabla_w f(w, X)]$。那么，梯度下降算法为

$$w_{k+1} = w_k - \alpha_k \nabla_w J(w_k) = w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)]. \quad (6.11)$$

该梯度下降算法可以在一些温和的条件下（如 f 的凸性）找到最优解 w^*。关于梯度下降算法的预备知识可以在附录 D 中找到。

梯度下降算法需要期望值 $\mathbb{E}[\nabla_w f(w_k, X)]$。获得期望值的一种方法是基于 X 的概率分布。然而，在实践中分布通常是未知的。另一种方法是收集 X 的大量独立同分布 (i.i.d.) 样本 $\{x_i\}_{i=1}^n$，这样期望值可以近似为

$$\mathbb{E}[\nabla_w f(w_k, X)] \approx \frac{1}{n}\sum_{i=1}^{n} \nabla_w f(w_k, x_i).$$

那么，(6.11) 变为

$$w_{k+1} = w_k - \frac{\alpha_k}{n}\sum_{i=1}^{n} \nabla_w f(w_k, x_i). \quad (6.12)$$

算法 (6.12) 的一个问题是它在每次迭代中都需要所有样本。在实践中，如果样本是逐个收集的，那么每收集到一个样本就更新 w 是有利的。为此，我们可以使用以下算法：

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k), \quad (6.13)$$

其中 x_k 是在时间步 k 收集到的样本。这就是著名的 ***随机梯度下降 (stochastic gradient descent) 算法***。该算法被称为“随机的”，**因为它依赖于随机样本** $\{x_k\}$。

与 (6.11) 中的梯度下降算法相比，SGD 用 *随机梯度 (stochastic gradient)* $\nabla_w f(w_k, x_k)$ 替换了真实梯度 $\mathbb{E}[\nabla_w f(w, X)]$。由于 $\nabla_w f(w_k, x_k) \neq \mathbb{E}[\nabla_w f(w, X)]$，这种替换还能保证当 $k \to \infty$ 时 $w_k \to w^*$ 吗？答案是肯定的。接下来我们给出一个直观的解释，并将收敛性的严格证明推迟到 6.4.5 节。

具体来说，由于(**通过噪声对随机梯度下降进行重写，加个噪声扰动**)

$$\begin{aligned}\nabla_w f(w_k, x_k) &= \mathbb{E}[\nabla_w f(w_k, X)] + (\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]) \\ &\doteq \mathbb{E}[\nabla_w f(w_k, X)] + \eta_k,\end{aligned}$$

SGD 算法 (6.13) 可以重写为

$$w_{k+1} = w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)] - \alpha_k \eta_k.$$

因此，SGD 算法与常规梯度下降算法相同，只是它有一个扰动项 $\alpha_k \eta_k$。由于 $\{x_k\}$ 是独立同分布的，我们有 $\mathbb{E}_{x_k}[\nabla_w f(w_k, x_k)] = \mathbb{E}_X[\nabla_w f(w_k, X)]$。结果是，

$$\mathbb{E}[\eta_k] = \mathbb{E}\left[\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]\right] = \mathbb{E}_{x_k}[\nabla_w f(w_k, x_k)] - \mathbb{E}_X[\nabla_w f(w_k, X)] = 0.$$

因此，扰动项 η_k 的均值为零，这直观地表明它可能不会破坏收敛性质。关于SGD的严格收敛性证明在 6.4.5 节中给出。

6.4.1 应用于均值估计 (Application to mean estimation)

接下来，我们应用 SGD 来分析均值估计问题，并展示式 (6.4) 中的均值估计算法是一种特殊的 SGD 算法。为此，我们将均值估计问题表述为一个优化问题：

$$\min_w J(w) = \mathbb{E}\left[\frac{1}{2}\|w - X\|^2\right] \doteq \mathbb{E}[f(w, X)], \quad (6.14)$$

其中 $f(w, X) = \|w - X\|^2/2$，其梯度为 $\nabla_w f(w, X) = w - X$。可以通过求解 $\nabla_w J(w) = 0$ 验证最优解为 $w^* = \mathbb{E}[X]$。因此，这个优化问题等价于均值估计问题。

> 🍰 注（可以和这个对比）：$g(w) \doteq w - \mathbb{E}[X]$.
>
> 原始问题是获取 $\mathbb{E}[X]$ 的值。这个问题被表述为一个求解 $g(w) = 0$ 的求根问题

- 用于求解 (6.14) 的梯度下降算法为

$$\begin{aligned}w_{k+1} &= w_k - \alpha_k \nabla_w J(w_k) \\ &= w_k - \alpha_k \mathbb{E}[\nabla_w f(w_k, X)] \\ &= w_k - \alpha_k \mathbb{E}[w_k - X].\end{aligned}$$

该梯度下降算法不适用，因为右侧的 $\mathbb{E}[w_k - X]$ 或 $\mathbb{E}[X]$ 是未知的（事实上，这正是我们需要求解的）。

- **用于求解 (6.14) 的 SGD 算法为**

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k) = w_k - \alpha_k (w_k - x_k),$$

- **其中 x_k 是在时间步 k 获得的样本。**值得注意的是，这个 SGD 算法与式 (6.4) 中的迭代均值估计算法相同。因此，(6.4) 是专门为解决均值估计问题而设计的 SGD 算法。

6.4.2 SGD 的收敛模式 (Convergence pattern of SGD)

SGD 算法的思想是用随机梯度代替真实梯度。然而，由于随机梯度是随机的，人们可能会问 SGD 的收敛速度是慢还是随机的。幸运的是，SGD 通常可以高效收敛。一个有趣的**收敛模式 (convergence pattern)** 是：

- 当估计值 w_k 远离最优解 w^* 时，它的表现类似于常规的梯度下降算法。
- 只有当 w_k 接近 w^* 时，**SGD 的收敛才会表现出更多的随机性**。

下面给出了对这种模式的分析和一个说明性示例。

- **分析 (Analysis)**：随机梯度和真实梯度之间的**相对误差 (relative error)** 为

$$\delta_k \doteq \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]|}{|\mathbb{E}[\nabla_w f(w_k, X)]|}.$$

为了简单起见，我们考虑 w 和 $\nabla_w f(w, x)$ 都是*标量 (scalars)* 的情况。由于 w^* 是最优解，因此成立 $\mathbb{E}[\nabla_w f(w^*, X)] = 0$。那么，相对误差可以重写为

$$\delta_k = \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]|}{|\mathbb{E}[\nabla_w f(w_k, X)] - \mathbb{E}[\nabla_w f(w^*, X)]|} = \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]|}{|\mathbb{E}[\nabla_w^2 f(\tilde{w}_k, X)](w_k - w^*)|}, \quad (6.15)$$

其中最后一个等式是由于中值定理 (mean value theorem) [7, 8] 以及 $\tilde{w}_k \in [w_k, w^*]$。假设 f 是严格凸的，使得对于所有 w, X，都有 $\nabla_w^2 f \geq c > 0$。那么，(6.15) 中的分母变为

$$|\mathbb{E}[\nabla_w^2 f(\tilde{w}_k, X)](w_k - w^*)| = |\mathbb{E}[\nabla_w^2 f(\tilde{w}_k, X)]||(w_k - w^*)| \geq c|w_k - w^*|.$$

将上述不等式代入 (6.15) 可得

$$\delta_k \leq \frac{\overbrace{|\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]|}^{\text{随机梯度} \quad - \quad \text{真实梯度}}}{\underbrace{c|w_k - w^*|}_{\text{到最优解的距离}}}.$$

上述不等式表明了 SGD 的一种有趣的收敛模式：**相对误差 δ_k 与 $|w_k - w^*|$ 成反比**。结果是，当 $|w_k - w^*|$ 很大时，δ_k 很小。**在这种情况下，SGD 算法的表现类似于梯度下降算法，因此 w_k 快速收敛到 w^*。当 w_k 接近 w^* 时，相对误差 δ_k 可能会很大，收敛表现出更多的随机性。**

- **示例 (Example)**：演示上述分析的一个好例子是均值估计问题。考虑 (6.14) 中的均值估计问题。当 w 和 X 都是标量时，我们有 $f(w, X) = |w - X|^2/2$，因此

$$\nabla_w f(w, x_k) = w - x_k,$$
$$\mathbb{E}[\nabla_w f(w, x_k)] = w - \mathbb{E}[X] = w - w^*.$$

因此，相对误差为

$$\delta_k = \frac{|\nabla_w f(w_k, x_k) - \mathbb{E}[\nabla_w f(w_k, X)]|}{|\mathbb{E}[\nabla_w f(w_k, X)]|} = \frac{|(w_k - x_k) - (w_k - \mathbb{E}[X])|}{|w_k - w^*|} = \frac{|\mathbb{E}[X] - x_k|}{|w_k - w^*|}.$$

相对误差的表达式清楚地表明 δ_k 与 $|w_k - w^*|$ 成*反比 (inversely proportional)*。结果是，当 w_k 远离 w^* 时，相对误差很小，SGD 的表现类似于梯度下降。此外，由于 δ_k 正比于 $|\mathbb{E}[X] - x_k|$，δ_k 的均值正比于 X 的方差。

仿真结果如图 6.5 所示。这里，$X \in \mathbb{R}^2$ 表示平面上的一个随机位置。它的分布是在以原点为中心的方形区域内的均匀分布，且 $\mathbb{E}[X] = 0$。均值估计基于 100 个独立同分布 (i.i.d.) 的样本。虽然均值的初始猜测值远离真实值，但可以看出 SGD 估计值迅速接近原点附近。当估计值接近原点时，收敛过程表现出一定的随机性。

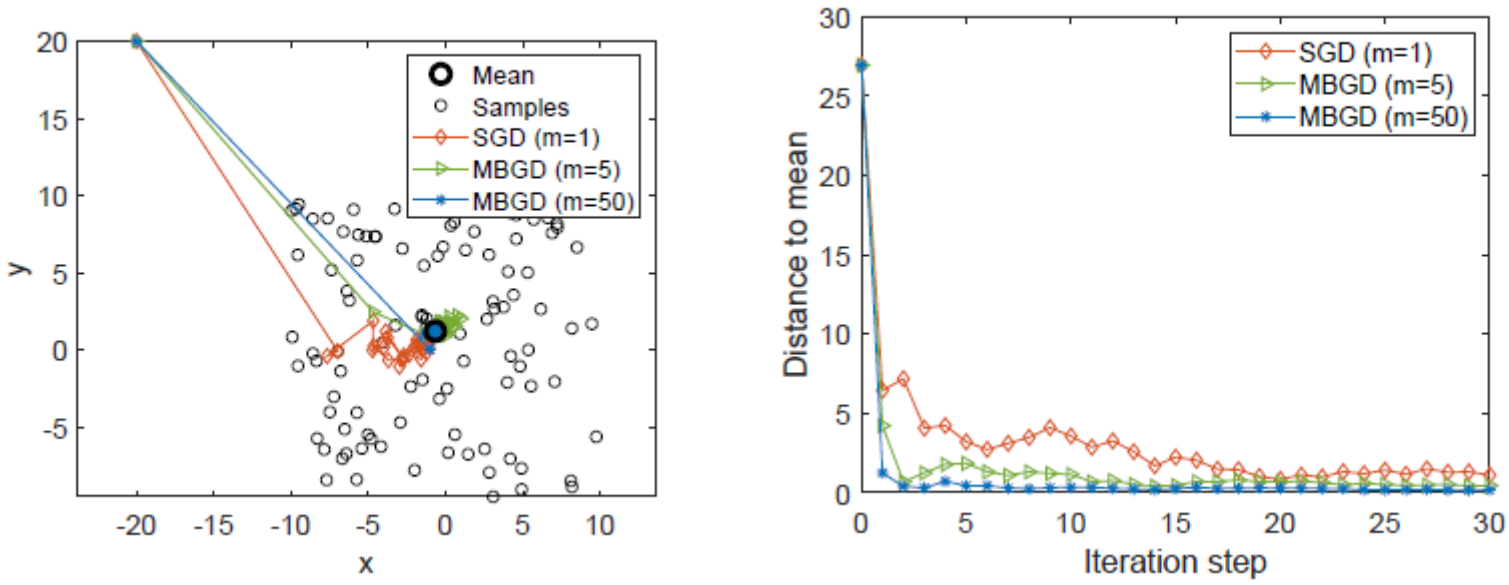

图 6.5：演示随机梯度下降和小批量 (mini-batch) 梯度下降算法的一个例子。$X \in \mathbb{R}^2$ 的分布是在以原点为中心、边长为 20 的正方形区域内的均匀分布。均值为 $\mathbb{E}[X] = 0$。均值估计基于 100 个独立同分布 (i.i.d.) 样本。

6.4.3 SGD 的一种确定性表述 (A deterministic formulation of SGD)

式 (6.13) 中的 SGD 表述涉及随机变量。人们经常会遇到不涉及任何随机变量的 SGD 的确定性表述。

特别地，考虑一组实数 $\{x_i\}_{i=1}^n$，其中 x_i 不必是任何随机变量的样本。待求解的优化问题是最小化平均值：

$$\min_w J(w) = \frac{1}{n} \sum_{i=1}^n f(w, x_i),$$

其中 $f(w, x_i)$ 是一个参数化函数，**w 是待优化的参数**。用于求解该问题的梯度下降算法为

$$w_{k+1} = w_k - \alpha_k \nabla_w J(w_k) = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^n \nabla_w f(w_k, x_i).$$

假设集合 $\{x_i\}_{i=1}^n$ 很大，并且我们每次只能取到一个数。

在这种情况下，以增量方式更新 w_k 是有利的：

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k). \quad (6.16)$$

必须注意的是，**这里的 x_k 是在时间步 k 获取的数**，而不是集合 $\{x_i\}_{i=1}^n$ 中的第 k 个元素。

(6.16) 中的算法与 SGD 非常相似，但其问题表述略有不同，因为它不涉及任何随机变量或期望值。

因此，许多问题随之而来。例如，这个算法是 SGD 吗？我们应该如何使用有限的数字集合 $\{x_i\}_{i=1}^n$？我们应该按特定顺序对这些数字进行排序然后逐个使用它们，还是应该从集合中随机采样一个数字？

对上述问题的一个快速回答是，尽管上述表述中不涉及随机变量，但我们可以通过引入随机变量将*确定性表述 (deterministic formulation)* 转换为*随机表述 (stochastic formulation)*。具体来说，令 X 为定义在集合 $\{x_i\}_{i=1}^n$ 上的随机变量。假设其概率分布是均匀的，使得 $p(X = x_i) = 1/n$。那么，确定性优化问题就变成了一个随机优化问题：

$$\min_w J(w) = \frac{1}{n} \sum_{i=1}^n f(w, x_i) = \mathbb{E}[f(w, X)].$$

上述方程中的最后一个等号是严格成立的，而不是近似的。因此，(6.16) 中的算法就是 SGD，并且如果 x_k 是从 $\{x_i\}_{i=1}^n$ 中*均匀且独立*采样的，则估计值会收敛。需要注意的是，由于是随机采样的，x_k 可能会重复取到 $\{x_i\}_{i=1}^n$ 中的同一个数。

6.4.4 BGD, SGD 和 mini-batch GD (BGD, SGD, and mini-batch GD)

虽然 SGD 在每次迭代中使用单个样本，但我们接下来介绍*小批量梯度下降 (mini-batch gradient descent, MBGD)*，它在每次迭代中使用稍微多一些的样本。当每次迭代使用所有样本时，该算法称为*批量梯度下降 (batch gradient descent, BGD)*。

具体而言，假设给定一组 X 的随机样本 $\{x_i\}_{i=1}^n$，我们要找到能最小化 $J(w) = \mathbb{E}[f(w, X)]$ 的最优解。用于解决此问题的 BGD、SGD 和 MBGD 算法分别为：

$$w_{k+1} = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} \nabla_w f(w_k, x_i), \qquad \text{(BGD)}$$

$$w_{k+1} = w_k - \alpha_k \frac{1}{m} \sum_{j \in \mathcal{I}_k} \nabla_w f(w_k, x_j), \qquad \text{(MBGD)}$$

$$w_{k+1} = w_k - \alpha_k \nabla_w f(w_k, x_k). \qquad \text{(SGD)}$$

- 在 BGD 算法中，每次迭代都使用所有样本。当 n 很大时，$(1/n)\sum_{i=1}^{n} \nabla_w f(w_k, x_i)$ 接近真实梯度 $\mathbb{E}[\nabla_w f(w_k, X)]$。
- 在 MBGD 算法中，$\mathcal{I}_k$ 是在时刻 k 获得的 $\{1, \ldots, n\}$ 的一个子集。该集合的大小为 $|\mathcal{I}_k| = m$。假设 $\mathcal{I}_k$ 中的样本也是独立同分布 (i.i.d.) 的。
- 在 SGD 算法中，x_k 是在时刻 k 从 $\{x_i\}_{i=1}^{n}$ 中随机采样的。

MBGD 可以被视为介于 SGD 和 BGD 之间的一个中间版本。与 SGD 相比，MBGD 具有较小的随机性，因为它使用了更多的样本，而不是像 SGD 那样只使用一个样本。与 BGD 相比，MBGD 不需要每次迭代都使用所有样本，这使得它更加灵活。如果 $m = 1$，那么 MBGD 就变成了 SGD。然而，如果 $m = n$，MBGD 可能不会变成 BGD。这是因为 MBGD 使用了 n 个随机获取的样本，而 BGD 使用的是所有的 n 个数。这 n 个随机获取的样本可能包含多次相同的数字，因此可能无法覆盖 $\{x_i\}_{i=1}^{n}$ 中的所有 n 个数。

通常，MBGD 的收敛速度比 SGD 快。这是因为 SGD 使用 $\nabla_w f(w_k, x_k)$ 来近似真实梯度，而 MBGD 使用 $(1/m)\sum_{j \in \mathcal{I}_k} \nabla_w f(w_k, x_j)$，后者更接近真实梯度，因为随机性被平均掉了。MBGD 算法的收敛性可以类似于 SGD 的情况进行证明。

演示上述分析的一个很好的例子是均值估计问题。特别地，给定一些数字 $\{x_i\}_{i=1}^{n}$，我们的目标是计算均值 $\bar{x} = \sum_{i=1}^{n} x_i/n$。这个问题可以等价地表述为以下优化问题：

$$\min_w J(w) = \frac{1}{2n} \sum_{i=1}^{n} \|w - x_i\|^2,$$

其最优解为 $w^* = \bar{x}$。用于解决该问题的三种算法分别是：

$$w_{k+1} = w_k - \alpha_k \frac{1}{n} \sum_{i=1}^{n} (w_k - x_i) = w_k - \alpha_k (w_k - \bar{x}), \qquad \text{(BGD)}$$

$$w_{k+1} = w_k - \alpha_k \frac{1}{m} \sum_{j \in \mathcal{I}_k} (w_k - x_j) = w_k - \alpha_k \left(w_k - \bar{x}_k^{(m)} \right), \qquad \text{(MBGD)}$$

$$w_{k+1} = w_k - \alpha_k (w_k - x_k), \qquad \text{(SGD)}$$

其中 $\bar{x}_k^{(m)} = \sum_{j \in \mathcal{I}_k} x_j/m$。此外，如果 $\alpha_k = 1/k$，上述方程可以求解如下所示：

$$w_{k+1} = \frac{1}{k} \sum_{j=1}^{k} \bar{x} = \bar{x}, \qquad \text{(BGD)}$$

$$w_{k+1} = \frac{1}{k} \sum_{j=1}^{k} \bar{x}_j^{(m)}, \qquad \text{(MBGD)}$$

$$w_{k+1} = \frac{1}{k} \sum_{j=1}^{k} x_j. \qquad \text{(SGD)}$$

上述方程的推导与 (6.3) 类似，在此省略。可以看出，BGD 在每一步给出的估计值正是最优解 $w^* = \bar{x}$。MBGD 收敛到均值的速度比 SGD 快，因为 $\bar{x}_k^{(m)}$ 已经是一个平均值。

图 6.5 给出了一个仿真示例，用于演示 MBGD 的收敛性。设 $\alpha_k = 1/k$。结果显示，具有不同小批量大小的所有 MBGD 算法都能收敛到均值 $m = 50$ 的情况收敛最快，而 $m = 1$ 的 SGD 最慢。这与上述分析一致。尽管如此，SGD 的收敛速度仍然很快，特别是当 w_k 远离 w^* 时。

6.4.5 SGD 的收敛性 (Convergence of SGD)

SGD 收敛性的严格证明如下。

定理 6.4 (SGD 的收敛性)。*对于 (6.13) 中的 SGD 算法，如果满足以下条件，则 w_k 几乎处处收敛于 $\nabla_w \mathbb{E}[f(w,X)]=0$ 的根。*

(a) $0<c_1\le \nabla_w^2 f(w,X)\le c_2$；

(b) $\sum_{k=1}^{\infty} a_k=\infty$ 且 $\sum_{k=1}^{\infty} a_k^2<\infty$；

(c) $\{x_k\}_{k=1}^{\infty}$ 是独立同分布 (i.i.d.) 的。

定理 6.4 中的三个条件讨论如下。

- 条件 (a) 是关于 f 的凸性。它要求 f 的曲率有上界和下界。这里，w 是一个标量，因此 $\nabla_w^2 f(w,X)$ 是二阶导数。这个条件可以推广到向量的情况。当 w 是向量时，$\nabla_w^2 f(w,X)$ 是著名的 Hessian 矩阵。
- 条件 (b) 类似于 RM 算法的条件。事实上，SGD 算法是一种特殊的 RM 算法（如方框 6.1 中的证明所示）。在实践中，a_k 通常被选为一个足够小的*常数*。尽管这种情况下不满足条件 (b)，但算法仍然可以在某种意义上收敛 [24, Section 1.5]。
- 条件 (c) 是一个常见的要求。

✏️ **方框 6.1：定理 6.4 的证明**

接下来我们展示 SGD 算法是一种特殊的 RM 算法。然后，SGD 的收敛性自然由 RM 定理得出。

SGD 要解决的问题是最小化 $J(w)=\mathbb{E}[f(w,X)]$。这个问题可以转换为求根问题。即，寻找 $\nabla_w J(w)=\mathbb{E}[\nabla_w f(w,X)]=0$ 的根。令

$$g(w)=\nabla_w J(w)=\mathbb{E}[\nabla_w f(w,X)].$$

那么，SGD 旨在找到 $g(w)=0$ 的根。这正是 RM 算法所解决的问题。我们可以测量的量是 $\tilde{g}=\nabla_w f(w,x)$，其中 x 是 X 的样本。注意 $\tilde{g}$ 可以重写为

$$\begin{aligned}\tilde{g}(w,\eta) &= \nabla_w f(w,x)\\ &= \mathbb{E}[\nabla_w f(w,X)]+\underbrace{\nabla_w f(w,x)-\mathbb{E}[\nabla_w f(w,X)]}_{\eta}.\end{aligned}$$

那么，用于求解 $g(w)=0$ 的 RM 算法是

$$w_{k+1}=w_k-a_k\tilde{g}(w_k,\eta_k)=w_k-a_k\nabla_w f(w_k,x_k),$$

这与 (6.13) 中的 SGD 算法相同。结果是，SGD 算法是一种特殊的 RM 算法。接下来我们展示定理 6.1 中的三个条件都得到了满足。那么，SGD 的收敛性自然由定理 6.1 得出。

- 由于 $\nabla_w g(w)=\nabla_w \mathbb{E}[\nabla_w f(w,X)]=\mathbb{E}[\nabla_w^2 f(w,X)]$，根据 $c_1\le\nabla_w^2 f(w,X)\le c_2$ 可知 $c_1\le\nabla_w g(w)\le c_2$。因此，定理 6.1 中的第一个条件得到满足。
- 定理 6.1 中的第二个条件与本定理中的第二个条件相同。
- 定理 6.1 中的第三个条件要求 $\mathbb{E}[\eta_k|\mathcal{H}_k]=0$ 且 $\mathbb{E}[\eta_k^2|\mathcal{H}_k]<\infty$。由于 $\{x_k\}$ 是独立同分布 (i.i.d.) 的，对于所有 k 我们有 $\mathbb{E}_{x_k}[\nabla_w f(w,x_k)]=\mathbb{E}[\nabla_w f(w,X)]$。因此，

$$\mathbb{E}[\eta_k|\mathcal{H}_k]=\mathbb{E}[\nabla_w f(w_k,x_k)-\mathbb{E}[\nabla_w f(w_k,X)]|\mathcal{H}_k].$$

由于 $\mathcal{H}_k=\{w_k,w_{k-1},\dots\}$ 且 x_k 独立于 $\mathcal{H}_k$，右边的第一项变为 $\mathbb{E}[\nabla_w f(w_k,x_k)|\mathcal{H}_k]=\mathbb{E}_{x_k}[\nabla_w f(w_k,x_k)]$。第二项变为 $\mathbb{E}[\mathbb{E}[\nabla_w f(w_k,X)]|\mathcal{H}_k]=\mathbb{E}[\nabla_w f(w_k,X)]$，因为 $\mathbb{E}[\nabla_w f(w_k,X)]$ 是 w_k 的函数。因此，

$$\mathbb{E}[\eta_k|\mathcal{H}_k]=\mathbb{E}_{x_k}[\nabla_w f(w_k,x_k)]-\mathbb{E}[\nabla_w f(w_k,X)]=0.$$

类似地，可以证明如果对于任意给定的 x 和所有的 w 都有 $|\nabla_w f(w,x)|<\infty$，则 $\mathbb{E}[\eta_k^2|\mathcal{H}_k]<\infty$。

由于满足定理 6.1 中的三个条件，SGD 算法的收敛性由此得出。

6.5 总结 (Summary)

本章没有介绍新的强化学习算法，而是介绍了随机逼近的预备知识，如 RM 和 SGD 算法。与许多其他求根算法相比，RM 算法不需要目标函数的表达式或其导数。我们已经展示了 SGD 算法是一种特殊的 RM 算法。此外，本章经常讨论的一个重要问题是均值估计。均值估计算法 (6.4) 是我们在本书中介绍的第一个随机迭代算法。我们展示了它是一种特殊的 SGD 算法。我们将在第 7 章看

到，时序差分学习算法具有相似的表达式。最后，“随机逼近”这一名称最早由 Robbins 和 Monro 在 1951 年使用 [25]。关于随机逼近的更多信息可以在 [24] 中找到。

6.6 问与答 (Q&A)

- 问：什么是随机逼近 (stochastic approximation)?

答： 随机逼近指的是一类广泛的用于求解求根或优化问题的随机迭代算法。

- 问：我们为什么要学习随机逼近?

答： 这是因为将在第 7 章介绍的时序差分强化学习算法可以被视为随机逼近算法。掌握了本章介绍的知识，我们可以更好地做好准备，这样在第一次看到这些算法时就不会感到突兀。

- 问：为什么我们在本章经常讨论均值估计问题?

答： 这是因为状态和动作价值被定义为随机变量的均值。第 7 章介绍的时序差分学习算法类似于用于均值估计的随机逼近算法。