

第六章 随机逼近

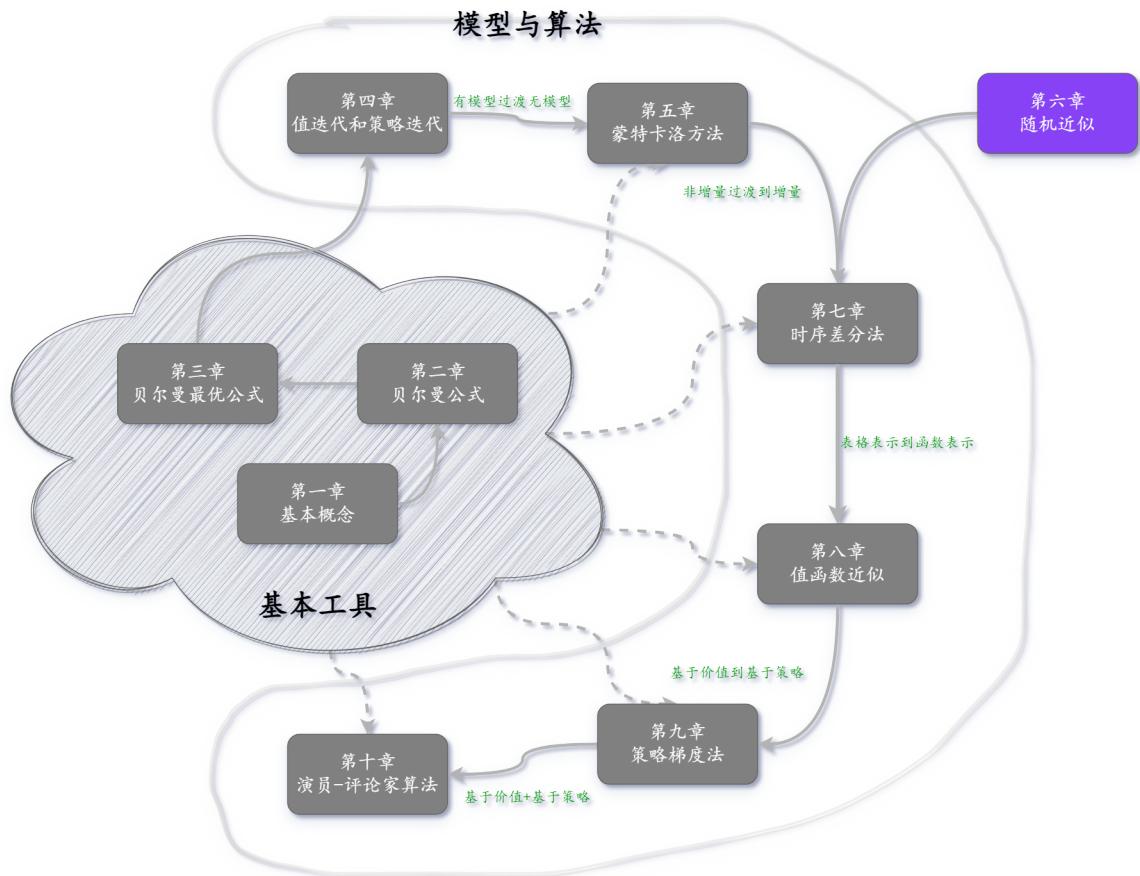


图 5.1

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，简称 TD-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研究的算法非常不同。许多第一次看到时序差分算法的读者常常想知道这些算法最初是如何设计的以及为什么它们能够有效地工作。事实上，前后章节之间存在知识差距：到目前为止我们学习的算法是非增量的，但后续章节我们将学习的算法是**增量的（Incremental）**。

我们在本章中通过介绍**随机逼近（Stochastic Approximation）**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，但为后续章节的学习奠定了必要的基础。我们将在第 7 章中看到，时序差分算法可以被视为特殊的随机逼近算法。本章还介绍了机器学习中广泛使用的著名**随机梯度下降算法（Stochastic Gradient Descent Algorithms）**。

6.1 示例：均值估计

接下来，我们通过均值估计问题来演示如何将非增量算法转换为增量算法。

考虑一个随机变量 X , 它从有限集 \mathcal{X} 中获取值。我们的目标是估计 $\mathbb{E}[X]$ 。假设我们有一个独立同分布的样本 $\{x_i\}_{i=1}^n$ 。 X 的期望值可以近似为

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

(6.1) 中的近似是蒙特卡罗估计的基本思想, 如第 5 章所介绍的。根据大数定律, 我们知道当 $n \rightarrow \infty$ 时, $\bar{x} \rightarrow \mathbb{E}[X]$ 。

接下来我们展示可以使用两种方法来计算 (6.1) 中的 \bar{x} 。第一种非增量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, 如果样本数量较多, 我们可能需要等待很长时间才能收集完所有样本。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

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

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, 我们得到如下增量算法:

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

该算法可用于以增量方式计算平均值 \bar{x} 。可以验证的是

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

式 (6.2) 的优点是每次收到样本时都可以立即计算平均值。该平均值可用于近似 \bar{x} , 从而近似 $\mathbb{E}[X]$ 。值得注意的是, 由于样本不足, 一开始近似值可能不准确。不过, 有总比没有好。随着样本的增多, 根据大数定律, 估计精度可以逐渐提高。另外, 还可以定义 $w_{k+1} = \frac{1}{k+1} \sum_{i=1}^{k+1} x_i$ 和 $w_k = \frac{1}{k} \sum_{i=1}^k x_i$ 。这样做不会产生任何重大差异。在这种情况下, 相应的迭代算法为

$$w_{k+1} = w_k - \frac{1}{k+1}(w_k - x_{k+1}).$$

此外，考虑一个具有更通用表达式的算法：

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

该算法很重要并且在本章中经常使用。与 (6.2) 相同，只是将系数 $1/k$ 替换为 $\alpha_k > 0$ 。由于未给出 α_k 的表达式，因此我们无法像(6.3) 那样获得 w_k 的显式表达式。然而，我们将在下一节中证明，如果 $\{\alpha_k\}$ 满足一些温和的条件，则当 $k \rightarrow \infty$ 时， $w_k \rightarrow \mathbb{E}[X]$ 。在第 7 章中，我们将看到时序差分法具有类似（但更复杂）的表达式。

6.2 Robbins - Monro 算法

随机逼近是指用于解决求根或优化问题的一类广泛的随机迭代算法¹。与许多其他求根算法（例如基于梯度的算法）相比，随机逼近的强大之处在于它不需要目标函数或其导数的表达式。

Robbins-Monro (RM) 算法是随机逼近领域的一项开创性工作^{1 2 3 4}。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。接下来我们详细介绍 RM 算法。

假设我们想要找到方程的根

$$g(w) = 0,$$

其中 $w \in \mathbb{R}$ 是未知变量， $g : \mathbb{R} \rightarrow \mathbb{R}$ 是函数。许多问题都可以表述为求根问题。例如，如果 $J(w)$ 是待优化的目标函数，则该优化问题可以转化为求解 $g(w) \doteq \nabla_w J(w) = 0$ 。另外，诸如 $g(w) = c$ 这样的方程，其中 c 是常数，也可以通过将 $g(w) - c$ 重写为新函数来转换为上述问题。

如果 g 或其导数的表达式已知，则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，该函数可以由结构和参数未知的人工神经网络来表示。此外，我们只能获得 $g(w)$ 的噪声观测值：

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

其中 $\eta \in \mathbb{R}$ 是观测误差，可能是也可能不是高斯分布。总之，它是一个黑盒系统，其中仅输入 w 和输出噪声 $\tilde{g}(w, \eta)$ 已知（见图 6.2）。我们的目标是使用 w 和 \tilde{g} 求解 $g(w) = 0$ 。

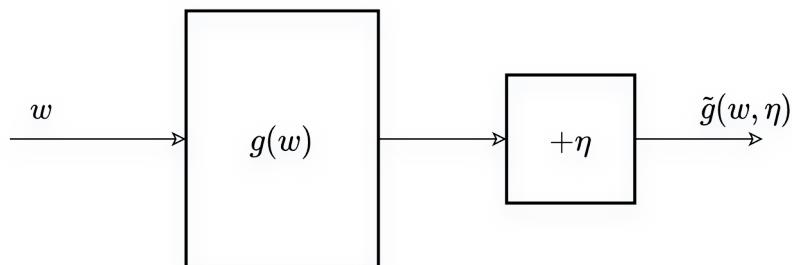


图 6.2

可以求解 $g(w) = 0$ 的 RM 算法是

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

其中 w_k 是根的第 k 个估计, $\tilde{g}(w_k, \eta_k)$ 是第 k 个噪声观测值, α_k 是正系数。可以看出, RM 算法不需要任何有关函数的信息。它只需要输入和输出。

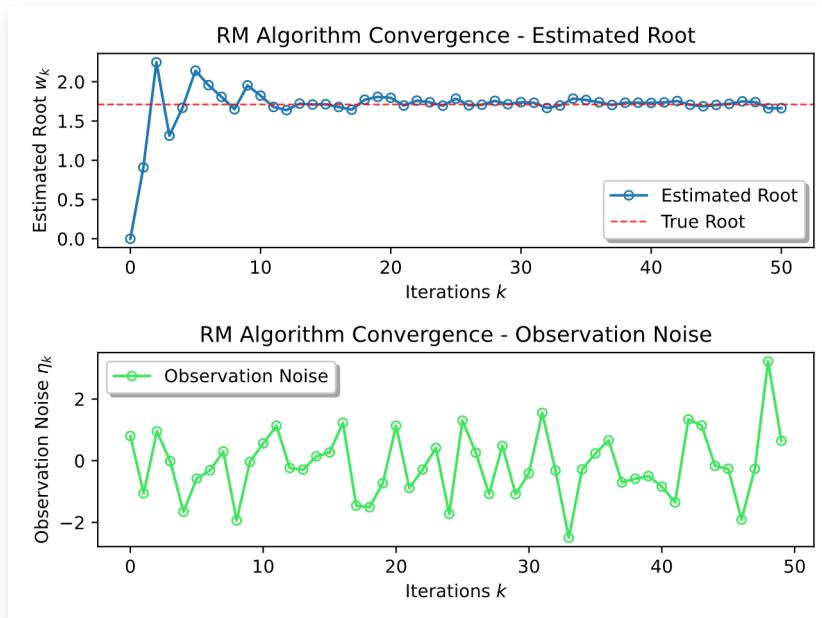


图 6.3

图 6.3 的 Python 代码如下:

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3  from matplotlib_inline import backend_inline
4
5  backend_inline.set_matplotlib_formats("svg")
6
7  # ===== #
8  def g(w):
9      return np.cbrt(w ** 3 - 5)
10
11
12  def noisy_observation(w, eta):
13      return g(w) + eta
14
15
16  def rm_algorithm(w0, alpha_func, num_iterations):
17      w_values = [w0]
18      eta_values = []
19      for k in range(1, num_iterations + 1):
20          alpha_k = alpha_func(k)
21          eta_k = np.random.normal(0, 1)
22          w_k = w_values[-1] - alpha_k * noisy_observation(w_values[-1], eta_k)
23          w_values.append(w_k)
24          eta_values.append(eta_k)

```

```

25     return w_values, eta_values
26
27 # ===== #
28 w0 = 0
29 alpha_func = lambda k: 1/k
30 num_iterations = 50
31
32 w_values, eta_values = rm_algorithm(w0, alpha_func, num_iterations)
33
34 plt.subplot(2, 1, 1)
35 plt.plot(w_values, label='Estimated Root', marker='o', markersize=5,
36           markerfacecolor='none')
36 plt.axhline(y=5***(1/3), color='r', linestyle='--', label='True Root', alpha=0.8,
37           lw=1)
37 plt.xlabel('Iterations $k$')
38 plt.ylabel('Estimated Root $w_k$')
39 plt.legend(shadow=True, fancybox=True)
40 plt.title('RM Algorithm Convergence - Estimated Root')
41
42 plt.subplot(2, 1, 2)
43 plt.plot(eta_values, label='Observation Noise', marker='o', markersize=5,
44           markerfacecolor='none', color="#3EE65E")
44 plt.xlabel('Iterations $k$')
45 plt.ylabel('Observation Noise $\eta_k$')
46 plt.legend(shadow=True, fancybox=True)
47 plt.title('RM Algorithm Convergence - Observation Noise')
48
49 plt.tight_layout()
50 plt.show()

```

为了说明 RM 算法, 请考虑一个 $g(w) = w^3 - 5$ 的示例。真根为 $5^{1/3} \approx 1.71$ 。现在, 假设我们只能观察输入 w 和输出 $\tilde{g}(w) = g(w) + \eta$, 其中 η 是独立同分布的。并服从均值为 0、标准差为 1 的标准正态分布。初始猜测为 $w_1 = 0$, 系数为 $\alpha_k = 1/k$ 。 w_k 的演化过程如图 6.3 所示。即使观测值被噪声 η_k 破坏, 估计 w_k 仍然可以收敛到真根。请注意, 必须正确选择初始猜测 w_1 , 以确保特定函数 $g(w) = w^3 - 5$ 的收敛。在下面的小节中, 我们将介绍 RM 算法对于任何初始猜测收敛的条件。

6.2.1 收敛性

为什么 (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想法, 然后给出严格的收敛分析。

考虑图 6.4 中所示的示例。在此示例中, $g(w) = \tanh(w - 1)$ 。 $g(w) = 0$ 的真根是 $w^* = 1$ 。我们应用 RM 算法, 其中 $w_1 = 3$ 且 $\alpha_k = 1/k$ 。为了更好地说明收敛的原因, 我们简单地设置 $\eta_k \equiv 0$, 因此 $\tilde{g}(w_k, \eta_k) = g(w_k)$ 。本例中的 RM 算法为

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

RM 算法生成的结果 $\{w_k\}$ 如图 6.4 所示。可以看出 w_k 收敛到 $w^* = 1$ 。

这个简单的例子可以说明 RM 算法收敛的原因：

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

无论哪种情况, w_{k+1} 都比 w_k 更接近 w^* 。因此, w_k 收敛于 w^* 是直观的。

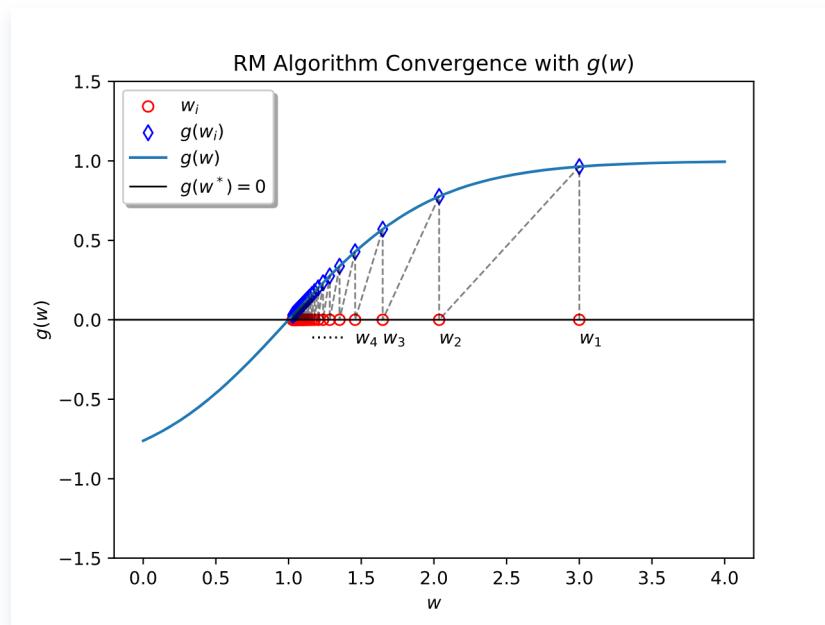


图 6.4

图 6.4 的 Python 代码如下：

```

1  import numpy as np
2  import matplotlib.pyplot as plt
3  from matplotlib_inline import backend_inline
4
5  backend_inline.set_matplotlib_formats("svg")
6
7  # ===== #
8  def g(w):
9      return np.tanh(w - 1)
10
11 # 定义 RM 算法
12 def rm_algorithm(w0, alpha_func, iterations):
13     w_values = [w0]
14     for k in range(1, iterations + 1):
15         w_k = w_values[-1] - alpha_func(k) * g(w_values[-1])

```

```

16         w_values.append(w_k)
17     return np.array(w_values)
18
19 # ===== #
20 w0 = 3
21 iterations = 50
22 alpha_func = lambda k: 1 / k # Alpha function (1/k)
23
24 # 应用 RM 算法
25 w_values = rm_algorithm(w0, alpha_func, iterations)
26
27 # 生成绘图用的 w 值
28 w_values_plot = np.linspace(0, 4, 1000)
29 g_values_plot = g(w_values_plot)
30 key = np.repeat(w_values, 2).reshape(-1).tolist()
31 value = np.vstack((np.zeros(iterations + 1), g(w_values))).reshape((-1,), order='F').tolist()
32
33 # 绘制图形
34 plt.scatter(
35     w_values, [0 for w in w_values], c='none', marker='o', edgecolors='r',
36     label="$w_i$"
37 )
38 plt.scatter(
39     w_values, [g(w) for w in w_values], c='none', marker='d', edgecolors='b',
40     label="$g(w_i)$"
41 )
42 plt.plot(
43     w_values_plot, g_values_plot, label="$g(w)$"
44 )
45
46 plt.axhline(
47     y=0, color='k', ls='-', label="$g(w^*) = 0$", lw=1
48 )
49
50 plt.plot(
51     key, value, ls='--', color='black', lw=1, alpha=0.5
52 )
53
54 for i in range(1, 6):
55     if i == 5:
56         plt.annotate(".....", xy=(w_values[i-1]-0.2, 0), xytext=(w_values[i-1]-0.2, -0.15))
57     else:

```

```

58     plt.annotate(f"${w}_{\{i\}}$",
59                   xy=(w_values[i-1], 0), xytext=(w_values[i-1],
60                         -0.15))
61     plt.xlabel('$w$')
62     plt.ylabel('$g(w)$')
63     plt.ylim(-1.5, 1.5)
64     plt.legend(shadow=True, fancybox=True)
65     plt.title('RM Algorithm Convergence with $g(w)$')
66     plt.show()

```

上面的例子很简单，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。下面给出严格的收敛结果。

定理 6.1 (RM 定理)

在 (6.5) 的 Robbins-Monro 算法中，如果

1. 对任意的 w , $0 < c_1 \leq \nabla_w g(w) \leq c_2$;
2. $\sum_{k=1}^{\infty} a_k = \infty$ 且 $\sum_{k=1}^{\infty} a_k^2 < \infty$;
3. $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$ 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 几乎必然会收敛到满足 $g(w^*) = 0$ 的根 w^* 。

我们把这个定理的证明推迟到 6.3.3 节。该定理依赖于附录 B 中介绍的几乎必然收敛的概念。

定理 6.1 中的三个条件解释如下：

- 在第一个条件中, $0 < c_1 \leq \nabla_w g(w)$ 表示 $g(w)$ 是单调递增函数。此条件确保 $g(w) = 0$ 的根存在且唯一。如果 $g(w)$ 单调递减, 我们可以简单地将 $-g(w)$ 视为单调递增的新函数。

作为一个应用, 我们可以将目标函数为 $J(w)$ 的优化问题表述为求根问题: $g(w) \doteq \nabla_w J(w) = 0$ 。此时, $g(w)$ 单调递增的条件表明 $J(w)$ 是凸的, 这是优化问题中常用的假设。

$\nabla_w g(w) \leq c_2$ 表示 $g(w)$ 的梯度从上方有界。例如, $g(w) = \tanh(w - 1)$ 满足此条件, 但 $g(w) = w^3 - 5$ 则不满足。

- 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情况。特别地, 条件 $\sum_{k=1}^{\infty} a_k^2 < \infty$ 意味着 $\lim_{n \rightarrow \infty} \sum_{k=1}^n a_k^2$ 上有界。它要求当 $k \rightarrow \infty$ 时, a_k 收敛到零。条件 $\sum_{k=1}^{\infty} a_k = \infty$ 意味着 $\lim_{n \rightarrow \infty} \sum_{k=1}^n a_k$ 无穷大。它要求 a_k 不能太快地收敛到零。这些条件具有有趣的特性, 稍后将对其进行详细分析。
- 第三种情况是轻微的。它不要求观测误差 η_k 是高斯分布。一个重要的特殊情况是 $\{\eta_k\}$ 是独立同分布。满足 $\mathbb{E}[\eta_k] = 0$ 且 $\mathbb{E}[\eta_k^2] < \infty$ 的随机序列。在这种情况下, 第三个条件有效, 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研究关于系数 $\{a_k\}$ 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？

当我们稍后对上述定理进行严格证明时，这个问题自然可以得到解答。在这里，我们想提供一些富有洞察力的直觉。

首先， $\sum_{k=1}^{\infty} a_k^2 < \infty$ 表示当 $k \rightarrow \infty$ 时， $a_k \rightarrow 0$ 。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 $a_k \rightarrow 0$ ，则 $a_k \tilde{g}(w_k, \eta_k) \rightarrow 0$ ，因此 $w_{k+1} - w_k \rightarrow 0$ ，表明当 $k \rightarrow \infty$ 时， w_{k+1} 和 w_k 彼此接近。否则，如果 a_k 不收敛，那么当 $k \rightarrow \infty$ 时， w_k 仍然可能波动。

其次， $\sum_{k=1}^{\infty} a_k = \infty$ 表明 a_k 不应过快收敛到零。为什么这个条件很重要？总结方程两边 $w_2 - w_1 = -a_1 \tilde{g}(w_1, \eta_1)$, $w_3 - w_2 = -a_2 \tilde{g}(w_2, \eta_2)$, $w_4 - w_3 = -a_3 \tilde{g}(w_3, \eta_3)$, ……，得到

$$w_1 - w_{\infty} = \sum_{i=1}^{\infty} a_i \tilde{g}(w_i, \eta_i).$$

如果 $\sum_{k=1}^{\infty} a_k < \infty$ ，则 $|\sum_{i=1}^{\infty} a_i \tilde{g}(w_i, \eta_i)|$ 也是有界的。令 b 表示有限上界，使得

$$|w_1 - w_{\infty}| = \left| \sum_{i=1}^{\infty} a_i \tilde{g}(w_i, \eta_i) \right| \leq b. \quad (6.6)$$

如果初始猜测 w_1 选择远离 w^* 使得 $|w_1 - w^*| > b$ ，则根据 (6.6) 不可能有 $w_{\infty} = w^*$ 。这表明 RM 算法在这种情况下无法找到真解 w^* 。因此，在给定任意初始猜测的情况下，条件 $\sum_{k=1}^{\infty} a_k = \infty$ 是确保收敛性所必需的。

- 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：

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

一方面，有

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

其中 $k \approx 0.577$ 称为 Euler - Mascheroni 常数⁵（或欧拉常数）。由于当 $n \rightarrow \infty$ 时， $\ln n \rightarrow \infty$ ，我们有

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

事实上， $H_n = \sum_{i=1}^{\infty} \frac{1}{k}$ 在数论中被称为调和数⁶。另一方面，有

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

寻找 $\sum_{i=1}^{\infty} \frac{1}{k^2}$ 的值被称为巴塞尔问题⁷。

综上所述，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，稍微修改一下，例如 $a_k = 1/(k+1)$ 或 $a_k = c_k/k$ （其中 c_k 有界），也可以保留此条件。

在 RM 算法中，在许多应用中， a_k 通常被选为足够小的常数。虽然在这种情况下不再满足第二个条件，因为是 $\sum_{k=1}^{\infty} a_k = \infty$ 而不是 $\sum_{k=1}^{\infty} a_k < \infty$ ，但该算法在某种意义上仍然可以收敛。另外，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充分（不是任意），RM 算法仍然可以找到根。

6.2.2 应用：均值估计

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

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

是 (6.4) 中的均值估计算法。当 $\alpha_k = 1/k$ 时，可得 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，收敛分析就很重要了。我们可以证明，这种情况下的算法是一种特殊的 RM 算法，因此它的收敛自然遵循。

特别的，将函数定义为

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

原问题是得到 $\mathbb{E}[X]$ 的值。这个问题被表述为求解 $g(w) = 0$ 的求根问题。给定 w 值，我们可以获得的噪声观测值是 $\tilde{g} \doteq w - x$ ，其中 x 是 X 的样本。请注意， \tilde{g} 可以写为

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

其中， $\eta \doteq \mathbb{E}[X] - x$ 。

解决这个问题的 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 (6.4) 中的算法。因此，定理 6.1 保证，如果 $\sum_{k=1}^{\infty} a_k = \infty$ ， $\sum_{k=1}^{\infty} a_k^2 < \infty$ ，并且 $\{x_k\}$ 是独立同分布的，则 w_k 几乎肯定收敛于 $\mathbb{E}[X]$ 。值得一提的是，收敛性不依赖于任何有关 X 分布的假设。

6.3 德沃列茨基收敛定理

到目前为止，RM 算法的收敛性尚未得到证明。为此，我们接下来介绍德沃列茨基定理^{8 9}，这是随机逼近领域的经典结果。这个定理可以用来分析 RM 算法和很多强化学习算法的收敛性。本节的数学内容稍微密集。建议对随机算法的收敛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。这里对于所有 k , $\alpha_k \geq 0$, $\beta_k \geq 0$ 。那么, 如果满足以下条件, 则 Δ_k 几乎肯定收敛于 0:

1. $\sum_{k=1}^{\infty} \alpha_k = \infty$, $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$ 且 $\sum_{k=1}^{\infty} \beta_k^2 < \infty$ 几乎肯定一致;
2. $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$ 且 $\mathbb{E}[\eta_k^2 | \mathcal{H}_k] \leq C$ 几乎肯定;

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

在证明这个定理之前, 我们首先说明一些问题。

- 在 RM 算法中, 系数序列 $\{\alpha_k\}$ 是确定性的。然而, 德沃列茨基定理允许 $\{\alpha_k\}$ 、 $\{\beta_k\}$ 为依赖于 \mathcal{H}_k 的随机变量。因此, 在 α_k 或 β_k 是 Δ_k 的函数的情况下更有用。
- 第一个条件被表述为“一致几乎肯定”。这是因为 α_k 和 β_k 可能是随机变量, 因此它们的极限的定义必须是随机意义上的。在第二个条件中, 也表述为“几乎肯定”。这是因为 \mathcal{H}_k 是随机变量序列而不是特定值。因此, $\mathbb{E}[\eta_k | \mathcal{H}_k]$ 和 $\mathbb{E}[\eta_k^2 | \mathcal{H}_k]$ 是随机变量。在这种情况下, 条件期望的定义是“几乎确定”的意义上的 (附录 B)。
- 定理 6.2 的表述与略有不同, 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 , $\beta_k = 0$ 的极端情况下, 序列仍然可以收敛。

6.3.1 德沃列茨基定理的证明

德沃列茨基定理的最初证明是在 1956 年给出的。还有其他证据。接下来我们提出一个基于拟鞅 (Quasimartingales) 的证明。有了拟鞅的收敛结果, 德沃列茨基定理的证明就很简单了。关于拟鞅的更多信息可以在附录 C 中找到。

□ 德沃列茨基定理的证明

设 $h_k \doteq \Delta_k^2$ 。则

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

对上式两边取期望, 得到

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

首先, 由于 Δ_k 被包括在内, 因此由 \mathcal{H}_k 决定, 因此可以从期望中取出它 (参见引理 B.1 中的性质 (e))。其次, 考虑 α_k 、 β_k 由 \mathcal{H}_k 确定的简单情况。例如, 当 $\{\alpha_k\}$ 和 $\{\beta_k\}$ 是 Δ_k 或确定性序列的函数时, 这种情况是有效的。然后, 它们也可以被取出来。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 $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$ 几乎可以肯定意味着 $\alpha_k \rightarrow 0$ 。结果, 存在一个有限的 n , 使得对于所有 $k \geq n$ 几乎肯定 $\alpha_k \leq 1$ 。不失一般性, 我们可以简单地考虑 $k \geq n$ 的情况, 因此几乎可以肯定 $\alpha_k \leq 1$ 。那么, $-\alpha_k(2 - \alpha_k)\Delta_k^2 \leq 0$ 。对于第二项, 我们假设 $\beta_k^2\mathbb{E}[\eta_k^2 | \mathcal{H}_k] \leq \beta_k^2 C$ 。第三项等于 0, 因为假设 $\mathbb{E}[\eta_k | \mathcal{H}_k] = 0$ 。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

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

最后一个不等式是因为条件 $\sum_{k=1}^{\infty} \beta_k^2 < \infty$ 。然后, 根据附录 C 中的拟鞅收敛定理, 我们得出 h_k 几乎肯定收敛的结论。 \square

虽然我们现在知道 h_k 是收敛的, Δ_k 也是收敛的, 但接下来我们要确定 Δ_k 收敛到什么值。由 (6.9) 可知

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

右侧第一项按假设是有界的。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, 左侧的 $\sum_{k=1}^{\infty} \alpha_k(2 - \alpha_k)\Delta_k^2$ 也是有界的。由于我们考虑 $\alpha_k \leq 1$ 的情况, 我们有

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

因此, $\sum_{k=1}^{\infty} \alpha_k\Delta_k^2$ 有界。由于 $\sum_{k=1}^{\infty} \alpha_k = \infty$, 我们几乎肯定有 $\Delta_k \rightarrow 0$ 。

6.3.2 均值估计的应用

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, 但我们接下来表明它的收敛性也可以通过 Dvoretzky 定理直接证明。

\square 证明:

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

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

设 $\Delta \doteq w - w^*$ 。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，因此我们有 $\mathbb{E}[x_k | \mathcal{H}_k] = \mathbb{E}[x_k] = w^*$ 。因此，如果 x_k 的方差是有限的，则 $\mathbb{E}[\eta_k | \mathcal{H}_k] = \mathbb{E}[x_k - w^* | \mathcal{H}_k] = 0$ 且 $\mathbb{E}[\eta_k^2 | \mathcal{H}_k] = \mathbb{E}[x_k^2 | \mathcal{H}_k] - (w^*)^2 = \mathbb{E}[x_k^2] - (w^*)^2$ 是有界的。根据 Dvoretzky 定理，我们得出结论 Δ_k 收敛到零，因此 w_k 几乎肯定收敛到 $w^* = \mathbb{E}[X]$ 。

6.3.3 RM 定理的应用

我们现在准备使用德沃列茨基定理证明 RM 定理。

□ 证明

RM 算法的目标是找到 $g(w) = 0$ 的根。假设根是 w^* ，使得 $g(w^*) = 0$ 。RM 算法是

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

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，我们有 $g(w_k) - g(w^*) = \nabla_w g(w'_k)(w_k - w^*)$ ，其中 $w'_k \in [w_k, w^*]$ 。设 $\Delta_k \doteq w_k - w^*$ 。上式变为

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

请注意，如假设所示， $\nabla_w g(w)$ 的边界为 $0 < c_1 \leq \nabla_w g(w) \leq c_2$ 。由于假设 $\sum_{k=1}^{\infty} a_k = \infty$ 且 $\sum_{k=1}^{\infty} a_k^2 < \infty$ ，因此我们知道 $\sum_{k=1}^{\infty} \alpha_k = \infty$ 且 $\sum_{k=1}^{\infty} \alpha_k^2 < \infty$ 。因此，德沃列茨基定理中的所有条件都得到满足，因此 Δ_k 几乎肯定收敛于零。□

RM 定理的证明了德沃列茨基定理的威力。特别是，证明中的 α_k 是取决于 w_k 的随机序列，而不是确定性序列。在这种情况下，德沃列茨基定理仍然适用。

6.3.4 德沃列茨基定理的推广

接下来，我们将德沃列茨基定理扩展到可以处理多个变量的更一般的定理。其可以用来分析随机迭代算法（例如 Q - Learning）的收敛性。

定理 6.3

考虑实数的有限集 \mathcal{S} 。对于随机过程

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

对于每个 $s \in \mathcal{S}$, $\Delta_k(s)$ 几乎肯定收敛于 0, 如果 $s \in \mathcal{S}$ 满足以下条件：

1. $\sum_k \alpha_k(s) = \infty$, $\sum_k \alpha_k^2(s) < \infty$, $\sum_k \beta_k^2(s) < \infty$ 且 $\mathbb{E}[\beta_k(s) | \mathcal{H}_k] \leq \mathbb{E}[\alpha_k(s) | \mathcal{H}_k]$ 几乎肯定一致;
2. $\|\mathbb{E}[\eta_k(s) | \mathcal{H}_k]\|_\infty \leq \gamma \|\Delta_k\|_\infty$, 其中 $\gamma \in (0, 1)$;
3. $\text{Var}[\eta_k(s) | \mathcal{H}_k] \leq C(1 + \|\Delta_k(s)\|_\infty)^2$, 其中 C 是常数;

其中 $\mathcal{H}_k = \{\Delta_1, \Delta_2, \dots, \eta_{k-1}, \dots, \alpha_{k-1}, \dots, \beta_{k-1}, \dots\}$ 代表历史信息, $\|\cdot\|_\infty$ 指的是最大范数。

下面给出关于该定理的一些解读：

- 我们首先澄清定理中的一些符号。变量 x 可以被视为一个索引。在强化学习的背景下，它表示一种状态或状态 - 动作对。最大范数 $\|\cdot\|_\infty$ 是在集合上定义的。它与向量的 L^∞ 范数相似但不同。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

$$\|\Delta_k(s)\|_\infty \doteq \max_{s \in \mathcal{S}} |\Delta_k(s)|.$$

- 该定理比德沃列茨基定理更普遍。首先，由于最大范数运算，它可以处理多个变量的情况。这对于存在多个状态的强化学习问题非常重要。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受误差 Δ_k 限制。
- 需要注意的是，对于所有 $s \in \mathcal{S}$, Δ_k 的收敛性要求条件对于每一个 $s \in \mathcal{S}$ 都有效。因此，当应用该定理证明强化学习算法的收敛性时，我们需要证明条件对于每个状态（或状态 - 动作对）都有效。

6.4 随机梯度下降

本节介绍随机梯度下降（SGD）算法，该算法广泛应用于机器学习领域。我们会看到 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。

考虑一下优化问题：

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

其中 w 是要优化的参数， X 是随机变量。期望是相对于 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) 的一个简单方法是**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

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

这种梯度下降算法可以在一些温和的条件下（例如 f 的凸性）找到最优解 w^* 。有关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，该分布在实践中通常是未知的。另一种方法是收集大量独立同分布对 X 的进行采样 $\{x_i\}_{i=1}^n$ ，以便期望值可以近似为

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

那么，(6.11) 变为

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

(6.12) 中算法的一个问题是它需要每次迭代中的所有样本。在实践中，如果样本是一一收集的，那么每次收集样本时更新 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 收集的样本。这就是著名的随机梯度下降算法。该算法称为“随机”，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 $\nabla_w f(w_k, x_k)$ 代替了真实梯度 $\mathbb{E}[\nabla_w f(w, X)]$ 。由于 $\nabla_w f(w_k, x_k) \neq \mathbb{E}[\nabla_w f(w, X)]$ ，这样的替换仍然可以确保当 $k \rightarrow \infty$ 时， $w_k \rightarrow w^*$ 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第 6.4.5 节。

特别是因为

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

(6.13) 中的 SGD 算法可以重写为

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

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，因此我们有 $\mathbb{E}_{x_k}[\nabla_w f(w_k, x_k)] = \mathbb{E}_X[\nabla_w f(w, X)]$ 。因此，

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

因此，扰动项 η_k 的均值为零，这直观地表明它可能不会危害收敛性。SGD 收敛性的严格证明在第 6.4.5 节中给出。

6.4.1 均值估计的应用

接下来我们应用 SGD 来分析均值估计问题，并表明 (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 $f(w, X) = \|w - X\|^2/2$ 且梯度为 $\Delta_w f(w, X) = w - X$ 。可以验证最优解通过求解 $\nabla_w J(w) = 0$ 得到 $w^* = \mathbb{E}[X]$ 。因此，这个优化问题等价于均值估计问题。

- 求解 (6.14) 的梯度下降算法为

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

这种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，这是我们需要解决的）。

- 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，该 SGD 算法与 (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

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？幸运的是，SGD 一般情况下可以高效收敛。一个有趣的收敛模式是，当估计值 w_k 远离最优解 w^* 时，它的行为与常规梯度下降算法类似。只有当 w_k 接近 w^* 时，SGD 的收敛才表现出更多的随机性。

下面给出了对该模式的分析和说明性示例：

- 分析：随机梯度与真实梯度之间的相对误差为

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

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

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

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

$$\begin{aligned} \left| \mathbb{E}[\nabla_w^2 f(\tilde{w}_k, X)(w_k - w^*)] \right| &= \left| \mathbb{E}[\nabla_w f(\tilde{w}_k, X)] \right| \cdot \left| (w_k - w^*) \right| \\ &\geq c |w_k - w^*|. \end{aligned}$$

将上述不等式代入 (6.15) 得

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

上述不等式表明了 SGD 的一个有趣的收敛模式：相对误差 δ_k 与 $|w_k - w^*|$ 成反比。结果，当 $|w_k - w^*|$ 较大， δ_k 较小。在这种情况下，SGD 算法的行为类似于梯度下降算法，因此 w_k 很快收敛到 w^* 。当 w_k 接近 w^* 时，相对误差 δ_k 可能较大，收敛表现出更多的随机性。

- 示例：演示上述分析的一个很好的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

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

因此，相对误差为

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

相对误差的表达式清楚地表明 δ_k 与 $|w_k - w^*|$ 成反比。因此，当 w_k 远离 w^* 时，相对误差很小，SGD 的行为类似于梯度下降。此外，由于 δ_k 与 $|\mathbb{E}[X] - x_k|$ 成正比，因此 δ_k 的均值与 X 的方差成正比。

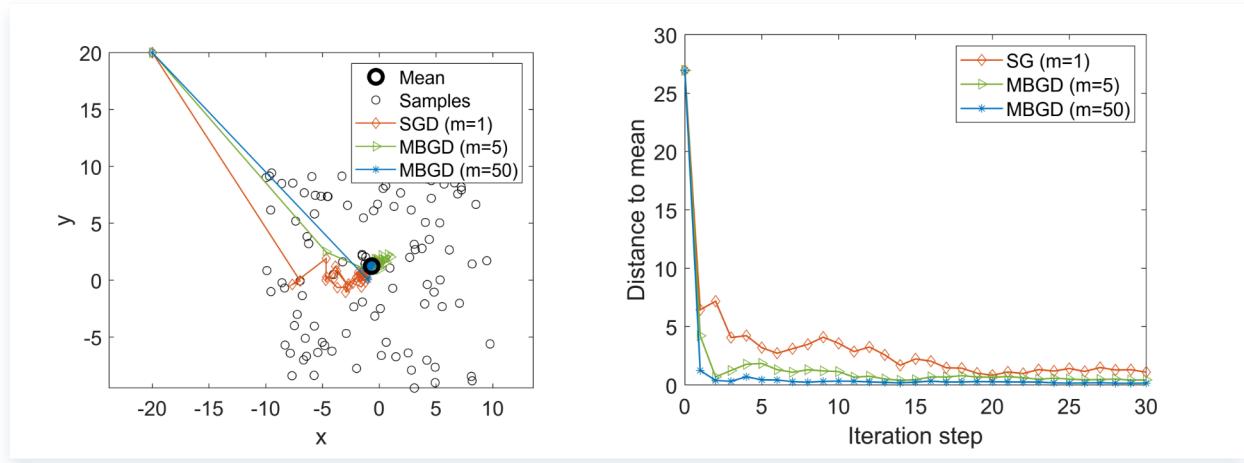


图 6.5

仿真结果如图 6.5 所示。这里， $X \in \mathbb{R}^2$ 表示平面中的随机位置。它的分布在以原点为中心且 $\mathbb{E}[X] = 0$ 的正方形区域中是均匀的。平均值估计基于 100 个独立同分布的样本。虽然最初的均值猜测与真实值相差甚远，但可以看出 SGD 估计很快就逼近了原点的邻域。当估计值接近原点时，收敛过程表现出一定的随机性。

6.4.3 随机梯度下降的确定性公式

式 (6.13) 中的 SGD 公式涉及随机变量。人们可能经常会遇到不涉及任何随机变量的 SGD 确定性公式。

特别是，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。要解决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 $f(w, x_i)$ 是参数化函数， w 是要求的参数优化。解决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，我们每次只能获取一个数字。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 取到的数字，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。那么，很多问题就出现了。比如这个算法是 SGD 吗？我们应该如何使用有限数集 $\{x_i\}_{i=1}^n$ ？我们应该按照一定的顺序对这些数字进行排序，然后逐一使用它们，还是应该从集合中随机抽取一个数字？

对上述问题的快速回答是，虽然上述公式中不涉及随机变量，但我们可以引入随机变量将确定性公式转换为随机公式。特别地，令 X 为在集合 $\{x_i\}_{i=1}^n$ 上定义的随机变量。假设其概率分布是均匀的，使得 $p(X = x_i) = 1/n$ 。然后，确定性优化问题就变成了随机问题：

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

上式中的最后一个等式是严格的而不是近似的。因此，(6.16) 中的算法是 SGD，并且如果 x_k 是从 $\{x_i\}_{i=1}^n$ 均匀且独立地采样的，则估计收敛。请注意， x_k 可能会在 $\{x_i\}_{i=1}^n$ 中重复取相同的数字，因为它是随机采样的。

6.4.4 批量梯度下降、随机梯度下降和小批量梯度下降

虽然 SGD 在每次迭代中使用单个样本，但我们接下来引入**小批量梯度下降（MBGD）**，它在每次迭代中使用更多样本。当每次迭代都使用所有样本时，该算法称为**批量梯度下降（BGD）**。

特别是，假设我们希望找到能够最小化 $J(w) = \mathbb{E}[f(w, X)]$ 的最优解，给定 X 的一组随机样本 $\{x_i\}_{i=1}^n$ 。BGD、SGD、MBGD 算法分别是

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

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

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

在 BGD 算法中，每次迭代都会使用所有样本。当 n 较大时， $(1/n) \sum_{i=1}^n \nabla_w f(w_k, x_i)$ 接近真实梯度 $\mathbb{E}[\nabla_w f(w_k, X)]$ 。在 MBGD 算法中， \mathcal{I}_k 是 $1, \dots, n$ 在时间 k 获得。集合的大小为 $|\mathcal{I}_k| = m$ 。 \mathcal{I}_k 中的样本也被假定为独立同分布。在 SGD 算法中， x_k 是在 k 时刻从 $\{x_i\}_{i=1}^n$ 中随机采样的。

MBGD 可以看作是 SGD 和 BGD 之间的中间版本。与 SGD 相比，MBGD 的随机性较小，因为它使用更多样本，而不是像 SGD 那样仅使用一个样本。与 BGD 相比，MBGD 不需要在每次迭代中使用所有样本，使其更加灵活。如果 $m = 1$ ，则 MBGD 变为 SGD。然而，如果 $m = n$ ，MBGD 可能不会变成 BGD。这是因为 MBGD 使用 n 个随机获取的样本，而 BGD 使用所有 n 个样本。这 n 个随机获取的样本可能多次包含相同的样本，因此可能无法覆盖 $\{x_i\}_{i=1}^n$ 中的所有 n 个样本。

MBGD 的收敛速度总体上比 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 $(1/m) \sum_{j \in \mathcal{I}_k} \nabla_w f(w_k, x_j)$ ，由于随机性被平均，因此更接近真实梯度。MBGD 算法的收敛性可以与 SGD 情况类似地证明。

演示上述分析的一个很好的例子是均值估计问题。特别是，给定一些数字 $\{x_i\}_{i=1}^n$ ，我们的目标是计算平均值 $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ 。该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 $w^* = \bar{x}$ 。解决这个问题的三种算法分别是：

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

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

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

其中 $\bar{x}_k^{(m)} = \sum_{j \in \mathcal{I}_k} x_j / m$ 。此外，若 $\alpha_k = 1/k$ ，则上式的解如下：

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

$$w_{k+1} = \frac{1}{k} \sum_{i=1}^k \bar{x}_i^{(m)}, \quad (\text{MBGD})$$

$$w_{k+1} = \frac{1}{k} \sum_{i=1}^k x_i, \quad (\text{SGD})$$

上式的推导与 (6.3) 类似，此处不再赘述。可以看出，BGD 在每一步给出的估计正是最优解 $w^* = \bar{x}$ 。MBGD 比 SGD 更快地收敛到平均值，因为 $\bar{x}_k^{(m)}$ 已经是平均值。

图 6.5 给出了一个仿真例子来演示 MBGD 的收敛性。设 $\alpha_k = 1/k$ 。结果表明，所有具有不同小批量大小的 MBGD 算法都可以收敛到平均值。 $m = 50$ 的情况收敛最快，而 $m = 1$ 的 SGD 最慢。这与上面的分析是一致的。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

下面给出 SGD 收敛性的严格证明。

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 几乎肯定收敛到 $\nabla_w \mathbb{E}[f(w, X)] = 0$ 的根：

1. $0 < c_1 \leq \nabla_w^2 f(w, X) \leq c_2$ ；
2. $\sum_{k=1}^{\infty} a_k = \infty$ 和 $\sum_{k=1}^{\infty} a_k^2 < \infty$ ；
3. $\{x_k\}_{k=1}^{\infty}$ 是独立同分布的。

下面讨论定理 6.4 中的三个条件：

- 条件 (1) 是关于 f 的凸性的。它要求 f 的曲率从上方和下方受到限制。这里， w 是标量， $\nabla_w^2 f(w, X)$ 也是标量。这个条件可以推广到向量情况。当 w 是向量时， $\nabla_w^2 f(w, X)$ 就是众所周知的 Hessian 矩阵。
- 条件 (2) 与 RM 算法的条件类似。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。尽管在这种情况下不满足条件 (2)，但该算法在某种意义上仍然可以收敛。

- 条件 (3) 是一项常见要求。

□ 证明定理 6.4

接下来我们证明 SGD 算法是一种特殊的 RM 算法。那么，SGD 的收敛自然就遵循 RM 定理。

SGD 要解决的问题是最小化 $J(w) = \mathbb{E}[f(w, X)]$ 。这个问题可以转化为求根问题。也就是说，求 $\nabla_w J(w) = \mathbb{E}[\nabla_w f(w, X)] = 0$ 的根。让

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

然后，SGD 的目标是找到 $g(w) = 0$ 的根。这正是 RM 算法解决的问题。我们可以测量的量是 $\tilde{g} = \nabla_w f(w, x)$ ，其中 x 是 X 的样本。请注意， \tilde{g} 可以重写为

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

那么，求解 $g(w) = 0$ 的 RM 算法为

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

与 6.13 中的 SGD 算法相同。因此，SGD 算法是一种特殊的 RM 算法。接下来我们证明定理 6.1 中的三个条件都满足。那么，SGD 的收敛性自然由定理 6.1 得出。

- 由于 $\nabla_w g(w) = \nabla_w \mathbb{E}[\nabla_w f(w, X)] = \mathbb{E}[\nabla_w^2 f(w, X)]$ ，因此从 $c_1 \leq \nabla_w^2 f(w, X) \leq c_2$ 得出 $c_1 \leq \nabla_w g(w) \leq c_2$ 。因此，满足定理 6.1 中的第一个条件。
- 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 $\{x_k\}$ 是独立同分布的，因此对于所有 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 $\mathcal{H}_k = \{w_k, w_{k-1}, \dots\}$ 并且 x_k 与 \mathcal{H}_k 无关，右侧第一项变为 $\mathbb{E}[\nabla_w f(w_k, x_k) | \mathcal{H}_k] = \mathbb{E}_{x_k}[\nabla_w f(w_k, x_k)]$ 。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，可以证明对于给定任何 x 的所有 w ，如果 $|\nabla_w f(w, x)| < \infty$ ，则 $\mathbb{E}[\eta_k^2 | \mathcal{H}_k] < \infty$ 。

由于满足定理 6.1 中的三个条件，因此 SGD 算法是收敛的。

1. H.-F. Chen, Stochastic approximation and its applications, vol. 64. Springer Science & Business Media, 2006. [↩](#) [↩](#)
2. H. Robbins and S. Monro, “A stochastic approximation method,” *The Annals of Mathematical Statistics*, pp. 400–407, 1951. [↩](#)
3. J. Venter, “An extension of the Robbins-Monro procedure,” *The Annals of Mathematical Statistics*, vol. 38, no. 1, pp. 181–190, 1967. [↩](#)
4. D. Ruppert, “Efficient estimations from a slowly convergent Robbins-Monro process,” tech. rep., Cornell University Operations Research and Industrial Engineering, 1988. [↩](#)
5. J. Lagarias, “Euler’s constant: Euler’s work and modern developments,” *Bulletin J. Lagarias*, “Euler’s constant: Euler’s work and modern developments,” *Bulletin* [↩](#)
6. J. H. Conway and R. Guy, *The book of numbers*. Springer Science & Business Media, 1998. [↩](#)
7. S. Ghosh, “The Basel problem,” arXiv:2010.03953, 2020. [↩](#)
8. A. Dvoretzky, “On stochastic approximation,” in *The Third Berkeley Symposium on Mathematical Statistics and Probability*, 1956. [↩](#)
9. T. Jaakkola, M. I. Jordan, and S. P. Singh, “On the convergence of stochastic iterative dynamic programming algorithms,” *Neural Computation*, vol. 6, no. 6, pp. 1185–1201, 1994. [↩](#)