

## Key Point: 马尔可夫过程

- model\_name: gpt4
- model\_version: gpt4o
- prompt:

Markov chain Monte Carlo的思想和原理是什么，它是如何工作的

- response:

好的，以下是使用Markdown语法和LaTeX公式包裹的详细回答。

## Markov Chain Monte Carlo (MCMC) 的思想和原理

MCMC (Markov Chain Monte Carlo) 方法结合了马尔可夫链和蒙特卡洛方法，主要用于从复杂的概率分布中抽样，特别是在贝叶斯推断和高维统计问题中。MCMC的核心思想是通过构建一个马尔可夫链，使其平稳分布为目标分布，从而通过马尔可夫链的随机游走来采样。

### 核心原理

#### 1. 马尔可夫链 (Markov Chain) :

- 马尔可夫链是一种随机过程，其状态转移只依赖于当前状态，而与过去的状态无关，具有“无记忆”性质。
- 马尔可夫链的目标是设计一个转移过程，使其最终的平稳分布 (stationary distribution) 就是我们所关注的目标分布。

#### 2. 目标分布 (Target Distribution) :

- MCMC的目的是从一个难以直接采样的目标分布 (例如后验分布) 中生成样本。
- 通过构造一个马尔可夫链，使得其平稳分布就是目标分布，经过足够的时间，链的状态将趋近于目标分布。

#### 3. 蒙特卡洛方法 (Monte Carlo) :

- MCMC的目标是利用蒙特卡洛方法，通过随机抽样来估计目标分布的统计特性。通过反复采样，可以估算期望值、方差等。

### MCMC的主要步骤

#### 1. 初始化:

- 选择一个初始状态  $x_0$ 。

#### 2. 转移过程:

- 根据某种准则，生成一个候选状态  $x'$ ，通常通过一个提议分布  $q(x'|x)$  从当前状态  $x$  生成。

#### 3. 接受或拒绝:

- 使用接受-拒绝准则 (例如Metropolis-Hastings算法) 决定是否接受候选样本。接受的概率由目标分布和提议分布的比值来决定:

$$A(x \rightarrow x') = \min \left( 1, \frac{p(x')q(x' \rightarrow x)}{p(x)q(x \rightarrow x')} \right)$$

其中,  $p(x)$  是目标分布,  $q(x' \rightarrow x)$  是提议分布。

#### 4. 迭代:

- 重复步骤2和3，直到获得足够多的样本。

## MCMC算法的关键组成部分

### 1. Metropolis-Hastings算法:

- 是最基本的MCMC算法之一。它通过提议分布生成候选样本，并根据接受概率决定是否接受该候选样本。

### 2. Gibbs采样:

- 是另一种特殊的MCMC方法，特别适用于条件分布已知的情况下。每次迭代时，更新每个变量，给定其它变量的当前值。

## 收敛性问题

为了确保通过MCMC采样得到的样本能够很好地近似目标分布，需要确保马尔可夫链收敛到目标分布。这通常需要：

- **不可约性 (Irreducibility)**：从任何状态都能到达其他状态。
- **遍历性 (Ergodicity)**：长期来看，链将遍历整个状态空间。
- **烧入期 (Burn-in Period)**：初始样本通常不代表目标分布，通常会丢弃前几个样本。

## 应用领域

- **贝叶斯推断**：从后验分布中采样，尤其在目标分布难以解析计算时，MCMC提供了一种可行的数值解法。
- **物理模拟**：用于计算物理系统的状态分布，常见于统计物理。
- **计算生物学**：例如基因组学中的模型推断。
- **图像处理**：图像去噪、图像分割等问题中使用MCMC进行采样和模型估计。

## 优点与挑战

- **优点：**
  - MCMC适用于高维复杂的概率分布，且只需能够计算目标分布的比例（无需完全知道分布的规范化常数）。
- **挑战：**
  - 选择合适的提议分布、收敛性验证、烧入期的处理、计算效率等问题是MCMC算法常见的挑战。

## 总结

MCMC通过构建马尔可夫链，使其能够有效地从复杂的分布中采样。通过马尔可夫链的状态转移过程，结合蒙特卡洛方法进行采样，使得MCMC成为解决许多高维和复杂统计问题的重要工具。

- prompt:

不同的采样方式（如 Metropolis-Hastings, Gibbs, Slice和 Multiple-try Metropolis）对MCMC有什么影响

- response:

在MCMC方法中，不同的采样策略（如 Metropolis-Hastings、Gibbs、Slice 和 Multiple-try Metropolis）对采样的效率、收敛性和适用性有着重要影响。每种方法的核心思想和优势不同，因此适用于不同类型的目标分布。

## 1. Metropolis-Hastings (MH) 算法

Metropolis-Hastings 是最常用的 MCMC 算法，广泛应用于复杂的概率分布采样。其核心思想是通过一个提议分布生成候选样本，并根据某些接受准则决定是否接受候选样本。

### 工作原理：

- 提议分布：**从当前状态 $x$  根据提议分布  $q(x'|x)$  生成一个候选样本  $x'$ 。
- 接受准则：**接受候选样本  $x'$  的概率为：

$$A(x \rightarrow x') = \min \left( 1, \frac{p(x')q(x|x')}{p(x)q(x'|x)} \right)$$

其中， $p(x)$  是目标分布， $q(x'|x)$ 是提议分布。

### 优势：

- 广泛适用：**Metropolis-Hastings 可以与任何提议分布配合使用，灵活性高。
- 收敛性：**通过适当设计，能够保证收敛到目标分布。

### 挑战：

- 接受率低：**如果提议分布与目标分布相差较大，接受率会很低，导致采样效率低下。
- 难以选择合适的提议分布：**提议分布不合适时，可能导致链的收敛速度慢。

## 2. Gibbs 采样

Gibbs采样是一个特殊的 MCMC 算法，适用于条件分布已知的情况。它通过逐一更新每个变量，给定其它变量的当前值来进行采样。每次迭代都会更新某个变量的状态，条件在其它变量已知的情况下。

### 工作原理：

- 假设目标分布是联合分布  $p(x_1, x_2, \dots, x_d)$ ，且每个变量的条件分布  $p(x_i|x_{-i})$  已知，其中  $x_{-i}$  是除了  $x_i$  以外的所有变量。
- 在每次迭代中，按顺序更新每个变量 $x_i$ ：

$$x_i^{(t+1)} \sim p(x_i|x_{-i}^{(t)})$$

### 优势：

- 高效：**对于条件分布容易计算的模型，Gibbs采样非常高效。
- 适用于高维问题：**尤其当变量之间有条件独立性时，Gibbs采样特别有用。

### 挑战：

- 条件分布的要求：**需要知道每个变量的条件分布，这在某些模型中可能不容易获得。
- 收敛性问题：**如果变量之间高度相关，Gibbs采样可能收敛缓慢。

### 3. Slice 采样

Slice 采样是另一种基于马尔可夫链的采样方法，通常适用于无法直接抽样的目标分布。它通过引入一个辅助变量来切割目标分布的一个“切片”，然后在这个切片内采样。

#### 工作原理：

- 对目标分布  $p(x)$ ，首先引入一个辅助变量  $u \sim U(0, p(x))$ ，然后通过以下过程生成新的候选样本：
  - 找到所有  $x'$  使得  $p(x') \geq u$ 。
  - 从这个区域内均匀地采样新的状态  $x'$ 。

#### 优势：

- 不需要计算归一化常数：** Slice采样是一种自适应的采样方法，不需要知道目标分布的归一化常数（常常是难以计算的）。
- 自适应性强：** 在高维问题中，能够适应目标分布的形状。

#### 挑战：

- 效率较低：** 相比于 Metropolis-Hastings，Slice采样在某些情况下效率较低，特别是对于高维数据或复杂分布。
- 依赖于切片区域的构建：** 切片区域的构建和采样步骤可能导致效率较差，特别是在目标分布非常复杂的情况下。

### 4. Multiple-try Metropolis (MTM) 算法

Multiple-try Metropolis 是 Metropolis-Hastings 方法的一个扩展，旨在提高采样效率。与常规的 Metropolis-Hastings 算法不同，Multiple-try Metropolis 通过同时生成多个候选样本，然后选择其中一个作为新状态。

#### 工作原理：

- 在每次迭代中，生成  $m$  个候选样本  $x'_1, x'_2, \dots, x'_m$ 。
- 计算所有候选样本的接受概率，并选择一个具有最高接受概率的候选样本作为新的状态。

#### 优势：

- 提高采样效率：** 通过生成多个候选样本并选择最优的样本，可以显著提高采样效率。
- 减少随机性：** 相比传统的 Metropolis-Hastings，多个尝试降低了受单一候选样本接受率低的影响。

#### 挑战：

- 计算复杂性：** 需要生成多个候选样本并计算其接受概率，可能增加计算开销，尤其在高维问题中。
  - 参数选择：** 需要选择合适的候选样本数量 ( $m$ )，如果选择不当，可能影响算法的效果。
-

# 不同采样方式对MCMC的影响

采样方式	优势	挑战	适用场景
Metropolis-Hastings	灵活，广泛适用	接受率低时效率低，提议分布难以选择	当目标分布复杂且提议分布可得时
Gibbs采样	高效，适用于条件分布已知的情形	需要知道条件分布，变量之间相关性强时可能收敛慢	条件分布已知或近似已知的高维模型
Slice采样	无需计算归一化常数，适应目标分布形状	计算开销大，效率较低，尤其在高维空间	当目标分布难以直接抽样时
Multiple-try Metropolis	提高采样效率，减少受接受率低的影响	计算复杂性增加，尤其在高维情况下	高维或复杂分布的情况下，提高采样效率

## 总结

不同的采样方式通过各自独特的方式优化了MCMC算法的效率与适用性。**Metropolis-Hastings** 适用范围广，但可能存在较低的接受率；**Gibbs采样** 高效，尤其适用于条件分布已知的情形；**Slice采样** 对归一化常数不敏感，但效率可能较低；**Multiple-try Metropolis** 则通过同时生成多个候选样本提高效率。选择哪种方法取决于具体问题的结构和目标分布的特点。

- prompt:

相比显式的马尔可夫模型，隐马尔可夫模型的优点是什么

- response:

隐马尔可夫模型（Hidden Markov Model，HMM）相比于显式的马尔可夫模型（Markov Model）的主要优点在于它能够处理那些不可观察的、隐藏的状态变量。隐马尔可夫模型通过引入隐状态来捕捉更加复杂的动态系统，而显式的马尔可夫模型则假设系统的每个状态都是可观测的。

下面是隐马尔可夫模型相对于显式马尔可夫模型的几个主要优点：

### 1. 能处理隐状态

隐马尔可夫模型的核心特点是**隐藏的状态**。在显式的马尔可夫模型中，系统的所有状态都是可观察的，而在隐马尔可夫模型中，系统的状态不可直接观察，而是通过观察到的**输出（观测值）**来间接推测状态。这使得隐马尔可夫模型能够建模那些难以直接观察的系统，广泛应用于语音识别、基因组学、金融市场建模等领域。

例如：

- 在语音识别中，实际的“语音状态”是无法直接观察的，我们只能通过声波信号来推断说话者的语言或发音状态。隐马尔可夫模型能够通过观测到的声波信号来推测说话人的隐含发音状态。
- 在生物信息学中，HMM可以用来建模DNA序列中的隐性状态，如基因组中的转录因子结合或基因表达状态。

## 2. 灵活性和适应性

隐马尔可夫模型比显式马尔可夫模型更加灵活和适应性强，因为它能根据隐状态之间的转移概率和观测模型来建模现实世界中的复杂过程。通过引入隐状态，HMM可以模拟出具有内在动态和长期依赖性的数据序列。

显式的马尔可夫模型只有直接观察到的状态转换过程，因此它的描述能力较为有限，而隐马尔可夫模型能够通过隐含的状态表示更复杂的现实系统。

## 3. 建模复杂的序列数据

隐马尔可夫模型特别擅长建模序列数据中的时间依赖关系。显式马尔可夫模型假设系统的每个状态是可观察的并且可以直接转移，但对于序列数据中的长期依赖关系，它不能很好地处理。而隐马尔可夫模型通过引入隐状态，使得它能够描述一个序列中的长期依赖关系和更复杂的动态系统。

例如，在天气预测中，显式马尔可夫模型只能根据当前天气状态预测下一步的天气，隐马尔可夫模型可以在当前观测数据的基础上推测隐藏的天气状态，并根据这些状态对未来天气进行预测。

## 4. 能够处理观测数据与状态之间的关系

隐马尔可夫模型通过**观测模型**（通常是一个概率分布）来描述隐状态与观测数据之间的关系。这意味着，HMM可以在状态空间较大、并且观测数据难以直接与状态对应的情境下，依然有效地进行建模和推断。

在显式马尔可夫模型中，观测和状态是直接相关的，无法有效处理隐含的、不直接可观测的关联。而隐马尔可夫模型则能够通过隐状态将这些不直接可见的关联提取出来，并有效建模。

## 5. 高效的推断和学习算法

隐马尔可夫模型具有**高效的推断和学习算法**，比如：

- 前向-后向算法**（Forward-Backward Algorithm）：用于计算给定观测序列的隐状态的后验概率。
- 维特比算法**（Viterbi Algorithm）：用于给定观测序列时，找到最可能的隐状态序列。
- Baum-Welch算法**：用于HMM的参数估计，可以通过EM算法（期望最大化）来学习模型的参数。

这些算法使得隐马尔可夫模型能够在实际应用中高效地进行推断和学习，即使在数据量庞大的情况下，也能快速估计隐状态并进行参数更新。

## 总结

优点	解释
能够处理隐状态	隐马尔可夫模型能够建模不可观察的状态，适用于许多需要通过观测值推测隐藏信息的场景，如语音识别、基因组学等。
灵活性和适应性	通过引入隐状态，HMM能够灵活地建模更加复杂的动态系统，适应更复杂的序列数据。
建模复杂的序列数据	隐马尔可夫模型擅长处理序列数据中的时间依赖性，能够捕捉数据中的长期依赖和复杂的时间动态。
处理隐含的观测-状态关系	隐马尔可夫模型通过观测模型，能够处理隐状态与观测数据之间的复杂关系，适用于那些观测数据与状态之间不容易直接关联的情况。

优点	解释
高效的推断和学习算法	隐马尔可夫模型具有高效的推断和学习算法，如前向-后向算法、维特比算法等，能够在数据量较大的情况下高效地进行推断和参数估计。

## 结论

相比显式的马尔可夫模型，隐马尔可夫模型能够通过引入隐状态来有效地处理那些不可直接观察的、复杂的系统动态，使其能够更好地适应实际问题中的隐含结构。它的灵活性和高效的学习与推断算法，使得它在语音识别、自然语言处理、基因组学、金融建模等多个领域得到了广泛应用。

- prompt:

请详细说明Viterbi 算法是如何求解隐马尔可夫模型的

- response:

### Viterbi算法：求解隐马尔可夫模型的最优隐状态序列

Viterbi算法是一种动态规划算法，用于在隐马尔可夫模型（HMM）中给定观测序列的情况下，找到最可能的隐状态序列（即最优路径）。它广泛应用于语音识别、自然语言处理、生物信息学等领域。

## 问题背景

- 输入：**隐马尔可夫模型  $(\mathcal{S}, \mathcal{O}, A, B, \pi)$  和观测序列  $O = (o_1, o_2, \dots, o_T)$ 。
  - $\mathcal{S} = \{s_1, s_2, \dots, s_N\}$ ：隐状态集合。
  - $\mathcal{O} = \{o_1, o_2, \dots\}$ ：观测符号集合。
  - $A = [a_{ij}]$ ：状态转移概率矩阵， $a_{ij} = P(s_j | s_i)$ 。
  - $B = [b_j(o)]$ ：观测概率矩阵， $b_j(o_t) = P(o_t | s_j)$ 。
  - $\pi = [\pi_i]$ ：初始状态分布， $\pi_i = P(s_i)$ 。
- 目标：**找到最可能的隐状态序列  $S^* = (s_1^*, s_2^*, \dots, s_T^*)$ ，使得：

$$S^* = \arg \max_S P(S | O, \lambda)$$

其中  $\lambda = (\mathcal{S}, \mathcal{O}, A, B, \pi)$  是模型参数。

## 算法核心思想

Viterbi算法通过动态规划避免了暴力枚举所有可能的隐状态序列（复杂度为  $N^T$ ）。它的核心是递归地计算以下两个值：

- 最优路径概率：**到时间  $t$  为止，以隐状态  $s_i$  结尾的最优路径概率。

$$\delta_t(i) = \max_{s_1, s_2, \dots, s_{t-1}} P(s_1, s_2, \dots, s_{t-1}, s_t = s_i, O | \lambda)$$

- 回溯指针：**记录从时间  $t - 1$  的哪个隐状态转移到当前时间  $t$  的隐状态  $s_i$  使得路径概率最大。

$$\psi_t(i) = \arg \max_j [\delta_{t-1}(j) a_{ji}]$$

## 算法步骤

### 1. 初始化 (时间 $t = 1$ ) :

对于每个隐状态  $s_i$ :

$$\delta_1(i) = \pi_i b_i(o_1)$$

$$\psi_1(i) = 0$$

- $\delta_1(i)$  表示初始时刻从  $s_i$  出发并生成观测  $o_1$  的概率。
  - $\psi_1(i)$  没有实际意义, 此时默认为0。
- 

### 2. 递归计算 (时间 $t = 2, 3, \dots, T$ ) :

对于每个时间  $t$  和每个隐状态  $s_i$ , 计算:

$$\delta_t(i) = \max_j [\delta_{t-1}(j) a_{ji}] b_i(o_t)$$

$$\psi_t(i) = \arg \max_j [\delta_{t-1}(j) a_{ji}]$$

- $\delta_t(i)$  是到时间  $t$  为止, 以状态  $s_i$  结尾的最优路径概率。
  - $\psi_t(i)$  记录了在时间  $t - 1$  的哪个隐状态转移到当前隐状态  $s_i$  使路径概率最大。
- 

### 3. 终止 (时间 $t = T$ ) :

找到最终最优路径的终止状态  $s_T^*$ :

$$P^* = \max_i \delta_T(i)$$

$$s_T^* = \arg \max_i \delta_T(i)$$

- $P^*$  是最优路径的概率。
  - $s_T^*$  是最优路径在时间  $T$  的隐状态。
- 

### 4. 回溯路径:

从时间  $t = T - 1, T - 2, \dots, 1$  逐步回溯, 得到完整的最优隐状态序列:

$$s_t^* = \psi_{t+1}(s_{t+1}^*)$$

## 关键思想与动态规划关系

- **子问题重叠**: 通过存储  $\delta_t(i)$  和  $\psi_t(i)$ , 避免重复计算。
  - **最优子结构**: 最优路径到达某个隐状态的子路径也一定是最优的。
- 

## 时间复杂度

- Viterbi算法的复杂度为  $O(N^2T)$ , 其中  $N$  是隐状态数量,  $T$  是观测序列长度。
  - 通过合理优化  $A$  和  $B$  的表示, 可以进一步降低实现中的常数因子。
-



## 总结

Viterbi算法是一种高效的动态规划算法，通过递归地计算最优路径概率和回溯指针，最终求解出隐马尔可夫模型中最优的隐状态序列。它利用了马尔可夫假设和动态规划思想，能够在有限的计算时间内找到最优路径，是处理序列数据的重要工具。

- prompt:

马尔可夫模型的推理速度受其链式结构制约，是否可以对其进行加速

- response:

是的，马尔可夫模型的推理速度确实受到其链式结构的制约，特别是在处理高维状态空间或长时间序列时，计算和推理的复杂度可能非常高。然而，有多种方法可以对其进行加速，提升效率。这些方法通常通过利用马尔可夫模型的结构特性、并行计算或近似推理技术来降低计算成本。以下是几种常见的加速技术：

### 1. 利用稀疏性 (Sparse Representations)

在许多实际应用中，马尔可夫模型的转移矩阵 (A) 和观测矩阵 (B) 可能是**稀疏**的，即只有少数转移或观测概率是非零的。通过利用这些稀疏性，可以减少计算量，从而加速推理过程。

- 稀疏矩阵乘法**：如果转移矩阵和观测矩阵是稀疏的，可以使用高效的稀疏矩阵乘法算法（如 CSR 格式）来加速矩阵运算。
- 稀疏性优化**：当使用动态规划算法（如 Viterbi 或前向后向算法）时，可以避免计算那些对应零概率的转移和观测，从而减少不必要的计算。

### 2. 并行化计算 (Parallelization)

在许多马尔可夫模型中，尤其是隐马尔可夫模型 (HMM)，不同的时间步骤或状态之间是独立的，可以通过并行化来加速推理过程。可以在多个核心或节点上并行计算不同时间步骤的结果。

- 多线程/多核处理**：对于前向-后向算法、Viterbi算法等，可以将每个时间步骤的计算独立处理，并行执行。
- GPU加速**：在需要大量矩阵运算的情况下，使用GPU（图形处理单元）来加速矩阵乘法和向量操作，能够显著提高推理速度，特别是在处理大规模数据时。

### 3. 近似推理 (Approximate Inference)

对于状态空间非常大的马尔可夫模型（例如，高维隐马尔可夫模型），直接计算可能会非常慢。通过采用一些近似推理方法，可以减少计算的复杂度并加速推理过程。

- 粒子滤波 (Particle Filtering)**：粒子滤波是一种基于蒙特卡洛采样的近似推理方法。通过在状态空间中采样多个粒子，更新这些粒子的权重，并通过加权平均来估计状态的分布，粒子滤波可以高效地处理高维和复杂的马尔可夫模型。
- 变分推理 (Variational Inference)**：变分推理通过将推理问题转化为优化问题，使用近似分布来估计真实分布。变分推理特别适用于大规模隐马尔可夫模型 (HMM) 和其他图模型，能够在保持合理精度的前提下加速计算。
- 蒙特卡洛方法 (Monte Carlo Methods)**：在一些情况下，通过使用MCMC（马尔可夫链蒙特卡洛）或其他采样方法，可以近似计算马尔可夫链的后验分布，避免精确计算，从而加速推理过程。

## 4. 剪枝技术 (Pruning)

在马尔可夫模型中，尤其是在序列很长的情况下，很多状态和路径的概率会非常小，几乎不会被选中。剪枝技术可以用来在推理过程中删除不太可能的状态或路径，从而加速计算。

- 低概率剪枝**：在Viterbi算法或前向算法中，当某些状态的概率低于某个阈值时，可以直接跳过这些状态或路径的计算，减少不必要的计算量。
- 合并不相关的状态**：如果某些状态之间的差异非常小或不重要，可以将它们合并，从而减少需要处理的状态数量。

## 5. 分层模型 (Hierarchical Models)

通过将马尔可夫模型分解为多个层次，尤其是对于复杂的马尔可夫链，可以通过**分层建模**来加速推理过程。每个层次的状态只依赖于上一层的状态，从而简化推理过程。

- 分层隐马尔可夫模型 (Hierarchical HMMs)**：将复杂的状态空间分解为多个较小的子空间，可以通过逐层计算来加速推理。
- 分块计算**：如果马尔可夫模型的状态空间可以分解成几个相对独立的部分，则可以分别在这些部分上并行计算，并在最后合并结果。

## 6. 利用低秩近似 (Low-Rank Approximation)

在一些马尔可夫模型中，状态转移矩阵和观测矩阵可能具有低秩结构，或者可以通过低秩近似来减少计算量。例如，通过奇异值分解 (SVD) 将转移矩阵进行近似，可以在降低计算复杂度的同时保持较好的近似精度。

- 矩阵近似**：通过低秩近似来简化转移矩阵和观测矩阵，能够减少计算和存储成本，加速推理过程。

## 7. 贝叶斯网络结构 (Bayesian Networks)

如果马尔可夫模型可以被表示为一个贝叶斯网络，可以通过贝叶斯网络的推理算法来加速推理。例如，使用**因子化模型** (factorized models) 将复杂的联合概率分布分解为更简单的部分，从而减少计算复杂度。

- 因子化推理**：通过对联合概率进行因子化，减少状态空间的维度和计算量，可以更有效地进行推理。

## 总结

通过上述方法，可以在不同场景下加速马尔可夫模型的推理过程。以下是主要的加速策略：

- 稀疏矩阵运算**：利用转移矩阵和观测矩阵的稀疏性来减少计算量。
- 并行化计算**：利用多线程或GPU加速并行计算。
- 近似推理**：通过粒子滤波、变分推理或蒙特卡洛方法进行近似计算。
- 剪枝技术**：去除不太可能的路径，减少计算量。
- 分层建模**：通过分解复杂的模型结构来简化计算。
- 低秩近似**：使用低秩近似减少矩阵运算的复杂度。
- 贝叶斯网络推理**：将马尔可夫模型表示为贝叶斯网络，通过因子化推理减少计算复杂度。

这些技术可以根据实际问题的特点和模型的规模灵活选择，从而显著提高推理速度。