DS4001-25SP-HW3：Bayes Net

TA：孙婧雯

2025 年 5 月 7 日

作业说明

本次作业包括书面部分（问题回答）和编程部分（代码填空），其中：

- **本次作业书面部分较多，书面部分可以使用手写拍照的方式或者 Markdown 完成，但必须以 PDF 形式提交。如果想使用 LaTeX，你可以将公式或手写照片插入到模板 `report_template.tex` 对应的位置。**
- 编程部分的完整代码位于 `Project` 文件夹中。你仅需在 `submission.py` 中指定位置完成代码，具体如下：

```
    # BEGIN_YOUR_CODE

    # END_YOUR_CODE
```

  除 `submission.py` 之外，**请勿修改其他任何文件**。每一道代码题的得分将有 $\frac{1}{3}$ 来自你的实验报告的描述，其余来自助教打分。**本次作业没有（隐藏）样例**。

- 完成所有内容后，请将 `submission.py` 与 `report.pdf` 打包成名为“学号 _ 姓名 _HW3.zip”的压缩文件，并上传至 Bb 系统。助教将根据你提交的文件进行评分。

注意事项:

- 本次作业允许使用大模型、生成式 AI 进行辅助，允许使用搜索引擎，但不允许直接用 AI 生成所有答案后复制粘贴，也不允许相互抄袭。如发现有互相抄袭的情况，抄袭者将平分获得的分数。
- 作业的截止时间为 2024 年 5 月 30 日 23:59:59。迟交 1/2/4/7 天分别会扣除 5/15/40/100 分。
- 如果在做作业过程中遇到问题，可以在 Github Issue 中提问或者线下在答疑课提问，也欢迎同学们相互解答问题。

作业正文

1 马尔可夫链 [24%]

马尔可夫链（Markov Chain, MC）是表示动态过程的一类概率图模型（PGM），适用于表示流程状态如何随时间变化。例如在一个非常简单的天气模型中，我们假设全天天气恒定不变，并且有三种可能的状态：晴天、阴天、下雨。此外，我们假设某一天的天气只取决于前一天的天气如何。这样，这个模型就可以被视为一个马尔科夫链：第二天的天气状态 S_{t+1} 只与前一天的天气状态 S_t 有关，即 $\mathrm{P}(S_{t+1} \mid S_t, S_{t-1}, ...) = \mathrm{P}(S_{t+1} \mid S_t)$。

我们可以用如下的非常简单的模型图来表示这个天气模型，其中每个节点表示特定天数的状态随机变量，随机变量之间用有向弧连接，表示具有概率相关性。

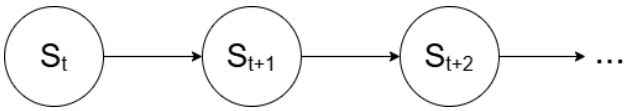


图 1: 简单天气模型的马尔可夫链模型图

随机变量 S_t 的取值有晴天、阴天、下雨三种，可记为 q_1、q_2、q_3，据此我们可以定义这个马尔可夫链：

- 状态集：$Q = \{q_1, q_2, ...q_n\}$
- 先验概率向量：$\mathbf{\Pi} = \{\pi_1, \pi_2, ...\pi_n\}$，其中 $\pi_i = \mathrm{P}(S_0 = q_i)$
- 转移概率矩阵：$\boldsymbol{A} = \{a_{ij}\}$, $i = [1..n]$, $j = [1..n]$，其中 $a_{ij} = \mathrm{P}(S_{t+1} = q_j \mid S_t = q_i)$

其中 n 是状态数，S_0 是初始状态，这个 MC 可简单表示为 $\theta = \{\boldsymbol{A}, \mathbf{\Pi}\}$，其先验概率和转移概率如下面两张表所示。

晴天	阴天	下雨
0.2	0.5	0.3

表 1: 简单天气模型的先验概率

	q_1	q_2	q_3
q_1	0.8	0.1	0.1
q_2	0.2	0.6	0.2
q_3	0.3	0.3	0.4

表 2: 简单天气模型的转移概率，每一行表示从当前状态到未来状态的概率，每行和为 1

在上两张表所有参数已知的情况下，我们可以回答很多关于天气模型的问题。例如，我们想知道以下状态序列的概率：$Q =$ 晴天、晴天、下雨、下雨、晴天、阴天、晴天，那么

$$\mathrm{P}(Q) = \pi_1 a_{11} a_{a13} a_{33} a_{31} a_{12} a_{21} = 3.84 \times 10^{-5}$$

但是如果我们对上述参数无法直接观察怎么办呢？这便属于隐马尔可夫模型（Hidden Markov Model, HMM）所研究的问题。HMM 是一个不可直接观察其状态的马尔可夫链，下图是一个表示 HMM 的图模型，上面的变量代表隐藏状态（不可直接观察），下面的变量代表观察结果：

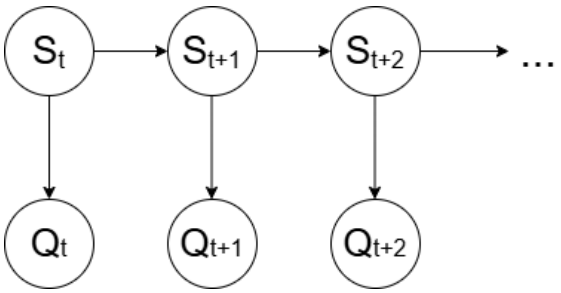


图 2: 表示隐马尔可夫模型的图模型

形式上，一个 HMM 可以定义如下：

- 状态集：$Q = \{q_1, q_2, ...q_n\}$
- 观察集：$O = \{o_1, o_2, ...o_m\}$
- 先验概率向量：$\mathbf{\Pi} = \{\pi_1, \pi_2, ...\pi_n\}$，其中 $\pi_i = \mathrm{P}(S_0 = q_i)$
- 转移概率矩阵：$\boldsymbol{A} = \{a_{ij}\}$, $i = [1..n]$, $j = [1..n]$，其中 $a_{ij} = \mathrm{P}(S_{t+1} = q_j \mid S_t = q_i)$
- 观察概率矩阵：$\boldsymbol{B} = \{b_{ij}\}$, $i = [1..n]$, $j = [1..m]$，其中 $b_{ij} = \mathrm{P}(O_t = o_j \mid S_t = q_i)$

其中 n 是状态数，m 是观察数，S_0 是初始状态，这个 HMM 可简单表示为 $\theta = \{\boldsymbol{A}, \boldsymbol{B}, \mathbf{\Pi}\}$。与 MC 的情况一样，我们所讨论的 HMM 也满足：下一个状态的概率只取决于当前状态、转移概率和观察概率不随时间变化，而且观察值仅依赖于当前状态，与更久之前的状态无关。高阶 HMM 如 $\mathrm{P}(S_{t+1} \mid S_t, S_{t-1}, ...) = \mathrm{P}(S_{t+1} \mid S_t, S_{t-1})$ 不在本次作业的讨论范围内。

接下来我们学习一种可以通过给定的观察序列来学习 HMM 的参数的参数估计算法——Baum-Welch 算法。Baum-Welch 算法和 EM（Expectation-Maximization）算法的关系可以总结为一句话：Baum-Welch 是 EM 算法在 HMM 上的特例实现。

EM 算法是一种迭代优化算法，用于在含有隐变量的概率模型中求最大似然估计。它包含两个步骤：

- E 步（Expectation）：在当前参数下，计算隐变量的后验期望；
- M 步（Maximization）：在固定隐变量期望下，最大化完全数据对数似然，更新参数。

在隐马尔可夫模型中，我们的目标是给定观测序列 $O = (o_1, \ldots, o_m)$，最大化模型参数 $\theta = \{\boldsymbol{A}, \boldsymbol{B}, \mathbf{\Pi}\}$ 的观察概率 $\mathrm{P}(O \mid \theta)$。但是，隐藏状态序列 $Q = (q_1, \ldots, q_n)$ 是不可见的，因此，HMM 的最大似然学习问题正好符合 EM 的使用场景。Baum-Welch 的算法结构如下：

- E 步
  - 计算每个时刻隐藏状态的边缘概率 γ
  - 计算每对相邻隐藏状态转移的联合概率 ξ

- M 步

  使用上述期望来更新初始状态分布 $\boldsymbol{\Pi}$、状态转移矩阵 $\boldsymbol{A}$、观测发射概率矩阵 $\boldsymbol{B}$。

M 步具体到公式大致如下：

$$\pi_i = \gamma_1(i) \tag{1}$$

$$a_{ij} = \sum_{t=1}^{T-1} \xi_t(i,j) \;/\; \sum_{t=1}^{T-1} \gamma_t(i) \tag{2}$$

$$b_{ij} = \sum_{t=1}^{T} \gamma_t(j) \text{ and } O_t = k \;/\; \sum_{t=1}^{T} \gamma_t(j) \tag{3}$$

1.1 回答问题 [24%]

1.1.1 手动计算 [4%]

对于简单天气模型，用表1和表2的参数计算：

1. 序列 { 阴天、下雨、晴天、晴天、晴天、阴天、下雨、下雨 } 的概率；
2. 连续四天下雨的概率；
3. 预计持续下雨可以下多少天？（提示：不需要随机过程相关知识，只要理解转移矩阵的含义就可以计算）

1.1.2 代码填空与参数估计 [16%]

你需要做的：回顾课程第 9 章 PPT 关于前向-后向算法（Forward-Backward Algorithm）的介绍，在 HW3 文件夹给出的 `baum-welch.py` 中进行代码填空，**将需要填写的 3 处内容复制在本题的回答区**；运行你填好的程序，将截图插入在本题的回答区。

提示：`alpha` 的含义：在时间步 t，系统处于隐状态 i，并且观测到了从时间 0 到 t 的观测序列 $o_0, o_1, ..., o_t$ 的联合概率；`beta` 的含义：在时间步 t，已知系统当前处于隐状态 i，后续从 t+1 到序列末尾的观测值 $o_{t+1}, ..., o_T$ 的条件概率。如果你依然发现难以入手，可以参考这篇文章。

1.1.3 运行结果对比 [4%]

吉布斯采样（Gibbs Sampling）也是一种重要的参数估计方法。请你回顾课程第 8 章 PPT 的相关介绍，并运行 HW3 文件夹给出的 `gibbs.py`，观察其结果和 Baum-Welch 算法的结果的区别，回答：两种方法的结果如何，为什么会有较大区别？（当然你需要除去随机初始化的因素，你可以多次运行并观察，从稳定性、精度、平滑性、算法机制甚至形象理解来回答）

2 贝叶斯网络 [64%]

贝叶斯网络（Bayesian Network）也是一种 PGM，用于表示变量之间的条件依赖关系。它由一个有向无环图（DAG）和条件概率分布表（CPT）组成，DAG 中每个节点表示一个随机变量，边表示因果或依赖关系。贝叶斯网络结合了图结构的可视化表达能力和贝叶斯推理的数学基础，能有效建模不确定性和因果推理，在农业规划、自然语言处理、医学诊断等领域广泛应用。

每一类图模型都可以从表征、推理、学习三个方面来了解，在上一部分，我们其实已经做到了 HMM 的表征和学习（参数估计），这一部分我们将关注贝叶斯网络的推理。

2.1 贝叶斯网络：推理 [54%]

在贝叶斯网络中，根据是否可以获得精确解，推理方法大致可分为精确推理（Exact Inference）与模糊推理（Approximate Inference）两类。

精确推理方法在理论上能够输出真实的后验概率分布，其代表性算法包括变量消除（Variable Elimination）、信念传播（Belief Propagation）以及前向-后向算法。尽管精确推理一般都能够给出理论最优的解，但其计算复杂度通常随状态空间呈指数增长，不适用于大规模模型或长序列，而且如果算法设计不佳，可能陷入局部最优解或者极端收敛。为克服精确推理的计算瓶颈，研究者们提出了多种模糊推理算法。这类方法通过随机采样、粒子模拟或近似因子图分解来获得对后验分布的近似估计，典型方法代表包括粒子滤波（Particle Filtering）、马尔可夫链蒙特卡洛（MCMC，例如 Gibbs 采样）和变分推理（Variational Inference）。模糊推理虽然牺牲了一定精度，但在高维问题中具有更强的扩展性，尤其适用于在线处理和部分可观测环境（如部分可观测马尔可夫决策过程，POMDP）。

似然权重采样（Likelihood Weighting Sampling）是一种经典的模糊推理方法，其基本思想是：每次采样完整变量赋值 $x_{1:n}$，其中观测变量保持固定，而对每个样本分配一个权重：

$$w^{(i)} = \prod_{j \in \text{Obs}} P(o_j \mid \text{Pa}(o_j)^{(i)}) \tag{4}$$

若父节点的采样值为 $\text{Pa}(o_j)^{(i)}$，那么 $P(o_j \mid \text{Pa}(o_j)^{(i)})$ 就是根据 CPT 查到的似然值，所有观测节点的似然值相乘，形成样本的权重。至于这里为什么会出现“父节点”这个称呼，就要靠之前提到的模型图来理解了，“节点”指的是某个时间点的状态或观测变量，是我们用来构建推理模型的抽象变量。o_j 的父节点表示影响 o_j 的变量，也就是它的直接上游因子。

由于样本之间权重差异可能很大，LW 方法容易在深层图结构中出现“权重塌缩”问题，使得大部分样本的有效性极低。因此，LW 更适合在网络深度较浅的离线场景的情境下使用。不过我们本次的实验是一个很小的项目，所以可以通过 LW 来了解模糊推理、与粒子滤波进行对比。

粒子滤波是一种适用于动态系统的时序采样方法，它用一组带权重的粒子 $\{x_t^{(i)}, w_t^{(i)}\}_{i=1}^N$ 来近似后验分布：

$$P(X_t \mid o_{1:t}) \approx \sum_{i=1}^{N} w_t^{(i)} \cdot \delta(x_t - x_t^{(i)}) \tag{5}$$

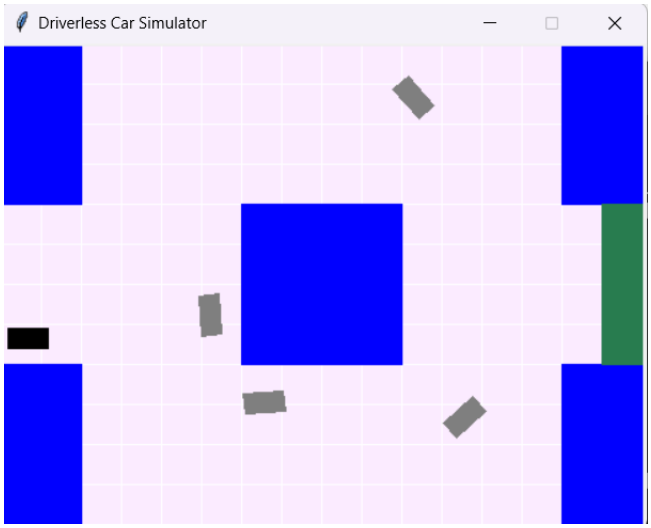
粒子的权重根据观测值进行更新，通常也使用似然加权：

$$w_t^{(i)} \propto w_{t-1}^{(i)} \cdot P(o_t \mid x_t^{(i)}) \tag{6}$$

其中，$x_t^{(i)}$：第 i 个粒子的状态，$w_t^{(i)}$：第 i 个粒子的权重，δ：Dirac delta 函数（取 1 代表此粒子确定地处于 $x_t^{(i)}$）。

我们将使用贝叶斯网络推理来完成一个非常简单的自动驾驶小车。你的任务是实现一个汽车追踪器，它通过贝叶斯推理手段计算你对其他车辆位置的信念并进行更新，从而免于发生碰撞。

为了简化问题，我们将世界离散化为由 (row,col) 对表示的二维网格中，其中 $0 \leq row < numRows$ 且 $0 \leq col < numCols$ 。对于每个网格我们都有一个对应的信念，你可以通过 `self.belief.getProb(row, col)` 访问这些值。要从一个网格转换为一个位置，使用 `util.rowToY(row)` 和 `util.colToX(col)`。代码位于 HW3 文件夹下的 code.zip。建议使用 Python=3.10 及以上版本。你需要完成以下内容：（**记得看代码注释！**）



2.1.1 精确推理 [16%]

完成后使用 `python drive.py -a -d -i ExactInference` 来享受你的智驾小车。你不必过于担心车辆的确切路径。相反，应专注于你的小车是否正确地推断出其他车辆的位置并进行规避。

(a) 补全 `submission.py` 中 `ExactInference` 类的 `observe` 函数。对于小车世界的每个网格，实现计算该位置和 agent 实际位置之间的欧式距离，并计算观测概率（发射概率）。`util.py` 中的一些函数（在注释里提到了）会很有助于你完成代码。

(b) 补全 `submission.py` 中 `ExactInference` 类的 `elapseTime` 函数，实现根据已经学习到的转移概率来更新信念，`util.py` 中的一些函数（在注释里提到了）会很有助于你完成代码。

2.1.2 模糊推理 [16%]

完成后使用 `python drive.py -a -d -i LikelihoodWeighting` 来享受你的智驾小车。(这一部分你可以参照粒子滤波的函数来写，不过不可能直接 `Ctrl+C+V`)

(c) 补全 `submission.py` 中 `LikelihoodWeighting` 类的 `updateBelief` 函数，实现根据当前样本和权重来更新信念，

(d) 补全 `submission.py` 中 `LikelihoodWeighting` 类的 `elapseTime` 函数，实现根据已经学习到的转移概率来更新信念，并根据转移概率采样下一个位置。

2.1.3 粒子滤波 [16%]

完成后使用 `python drive.py -a -d -i ParticleFilter` 来享受你的智驾小车。

(e) 补全 `submission.py` 中 `ParticleFilter` 类的 `init` 操作，根据已建立的转移概率字典来从所有可转移的位置均匀采样。

(f) 补全 `submission.py` 中 `ParticleFilter` 类的 `reweighting` 操作，实现粒子滤波算法的重加权，即公式6。

2.1.4 思考题 [6%]

完成所有代码后，请你观察 `LikelihoodWeighting` 和 `ParticleFilter` 类，你会发现两种算法调用 `self.updateBelief()` 的位置和次数存在差异。请你从算法思想和过程的角度，解释为什么会出现这些差异。

`code.zip` 中的各文件（夹）作用如下：

`/engine`	小车 agent 的各种接口和物理模拟	`/layouts`	地图
`/learned`	学到的转移概率表	`autoDriver.py`	实现自动驾驶
`drive.py`	设置驾驶参数	`learn.py`	保存转移概率
`learner.py`	计算概率分布	`none.py`	无推理情况下的驾驶
`submission.py`	你最喜欢的文件	`util.py`	也是你最喜欢的文件

2.2 贝叶斯网络：学习 [10%]

贝叶斯网络的学习通常包括两个方面：学习结构和学习参数，本次作业我们仅涉及后者。

无法观测的变量称为“隐变量”(latentvariable)，我们将考虑如何从观测到的数据中学习隐变量的分布，并且学习 CPT 的参数。我们将使用 EM 算法来学习这些参数，回顾课程第 9 章 PPT 和在介绍 Baum-Welch 算法时的知识，你应该知道：EM 算法的基本思想是首先初始化参数，然后交替地进行两个步骤：E 步骤和 M 步骤，直到参数收敛。在 E 步骤中，我们计算隐变量的后验分布，即给定观测数据和当前参数时隐变量的分布。在 M 步骤中，我们更新参数，以最大化对数似然。

2.2.1 回答问题 [10%]

有两枚硬币 A、B，其正面朝上的概率分别为 θ_A 和 θ_B，均未知。你随机地以概率 π 选中硬币 A，以概率 $1-\pi$ 选中硬币 B，并抛掷该硬币 5 次，观测到一串正反面结果。你进行了 4 轮这样的试验，只知道所有的抛掷结果，但不知道每一轮到底选用了哪枚硬币。请用 EM 算法估计参数集合 $\Theta = (\pi, \theta_A, \theta_B)$。

你需要做的：设初始化参数为 $\pi^{\text{old}} = 0.5$, $\theta_A^{\text{old}} = 0.6$, $\theta_B^{\text{old}} = 0.4$，计算第 1 轮观测后来自硬币 A 和硬币 B 的后验概率 $\gamma_{1,A}, \gamma_{1,B}$，也就是做一次 E 步。然后利用上一步算出的 $\gamma_{1,A}, \gamma_{1,B}$ 以及对其它三轮的初值近似（可假设它们的后验与第 1 轮相同）计算新的混合系数 π^{new} 和两枚硬币的偏置 $\theta_A^{\text{new}}, \theta_B^{\text{new}}$。

抛硬币的结果如下：第一轮：正、正、反、正、正；第二轮：反、反、正、反、反；第三轮：正、反、正、反、正；第四轮：反、正、反、反、反；

3 贝叶斯深度学习 [6%]

传统深度学习虽然取得了很多瞩目的成果，但是随着深度学习应用领域的扩展，人们发现典型的深度学习网络存在诸如过高置信预测导致危险、对分布外数据预测很差、易受对抗性操纵的影响等表现。贝叶斯网络虽然无法彻底消除这些问题，但是可以帮我们构建更鲁棒的机器学习系统。因此贝叶斯网络结合深度学习，就诞生了贝叶斯深度学习（Bayes Deep Learning, BDL），这一领域包含了一系列利用深度网络进行近似贝叶斯推理的方法，增强了深度学习系统的鲁棒性。

请你查阅相关资料，或者询问你最喜欢的大模型，回答：贝叶斯网络是如何帮助解决这些问题的？你的回答中应该至少包含一个现实的应用例子，例如 Charles Blundell 等人在 2015 年发表的论文 *Weight Uncertainty in Neural Networks*，介绍了贝叶斯反向传播（BBP）。

推荐阅读 [0%]

- Christopher Bishop 的 Pattern Recognition and Machine Learning 有关 PGM 的篇章
- Kevin Murphy 的 Machine Learning: A Probablistic Perspective，从概率论的角度详细论述了机器学习

作业反馈

体验反馈 [6%]

你可以写下任何反馈，包括但不限于以下几个方面：课堂、作业、助教工作等等。

(a) [**必做**] 你在本次作业花费的时间大概是?

(b) [**选做**] 你可以随心吐槽课程不足的方面，或者给出合理的建议。若没有想法，直接忽略本小题即可。你的回答不会影响给分。`o(> w < )o`