# 三峡大学

## 计 算 机 与 信 息 学 院

## 《人工智能》课 程 作 业

## 2024 年 秋 季 学 期

课程类型： 专业核心课

学　　号： 202210120510

姓　　名： 向申赤

专　　业： 计算机科学与技术

授课教师： 徐义春

完成日期：2024 年 9 月 20 日

# 足球比赛结果预测

## 一、 案例内容与要求

### 1. 1. 实验内容与任务

三支足球队 A,B,C 两两之间各赛一场，总共需要赛三场，分别是 A 对 B, A 对 C, B 对 C。对一支球队来说，一场比赛的结果可能是胜、平、负之一。假设每场比赛的结果以某种概率取决于两队的实力，而球队实力为一个 0-3 之间的整数。现已知前两场比赛结果是 A 战胜了 B，A 和 C 战平，请预测最后一场比赛 B 对 C 的结果。

### 1. 2. 实验过程及要求

1. 实验环境要求：Windows/Linux 操作系统，Python 编译环境，numpy, random 等程序库。
2. 建立足球比赛的贝叶斯网络，设置贝叶斯网络的条件概率表。
3. 分别实现精确求解方法、拒绝采样方法、似然加权采样方法、Gibbs 采 样方法，获得 BC 比赛结果的后验分布；
4. 调整采样次数，观测几个近似方法相对于精确解的差距。
5. 撰写实验报告。

## 二、 原理论述及解决方法

### 2. 1. 原理概述

不确定性推理利用概率论知识来处理状态和采取的行为。通过完全联合概率分布可以计算多个变量的任何分布问题，但是当变量过多时，计算量是巨大的，最后可能多到不可操作。实际问题中，如果变量之间的存在独立关系或者条件独立关系，则计算概率分布时计算量要小很多。应用贝叶斯网络模型来表示变量之间的依赖关系，是进行不确定性推理的重要工具。

贝叶斯网络是一种用于建模不确定性问题的概率图模型，图中的每个节点表示一个随机变量，节点之间的有向边表示随机变量之间的条件依赖关系。通过条件概率表（CPT），可以对每个节点的取值进行建模，条件概率表给出了某节点在其父节点取不同值的情况下的概率分布。

应用贝叶斯网络模型，进行概率分布的精确计算依然可能有较大的计算量，此时可以用采样的方法完成计算。当然采样计算是一种近似计算，但当采样规模增大时，计算结果逼近精确结果。

在本实验中，球队的实力用随机变量 XA、XB、XC 表示，分别对应 A、B、C 队伍的实力。每场比赛的结果取决于比赛双方的实力，使用条件概率表 PS 表示比赛结果。比如：

- P(sAB|XA, XB) 表示 A 队与 B 队比赛结果的条件概率
- P(sAC|XA, XC) 表示 A 队与 C 队比赛结果的条件概率
- P(sBC|XB, XC) 表示 B 队与 C 队比赛结果的条件概率

通过这些条件概率表，构建一个贝叶斯网络模型。已知前两场比赛的结果是 A 战胜 B，A 与 C 打平，实验目标是预测 B 与 C 的比赛结果。

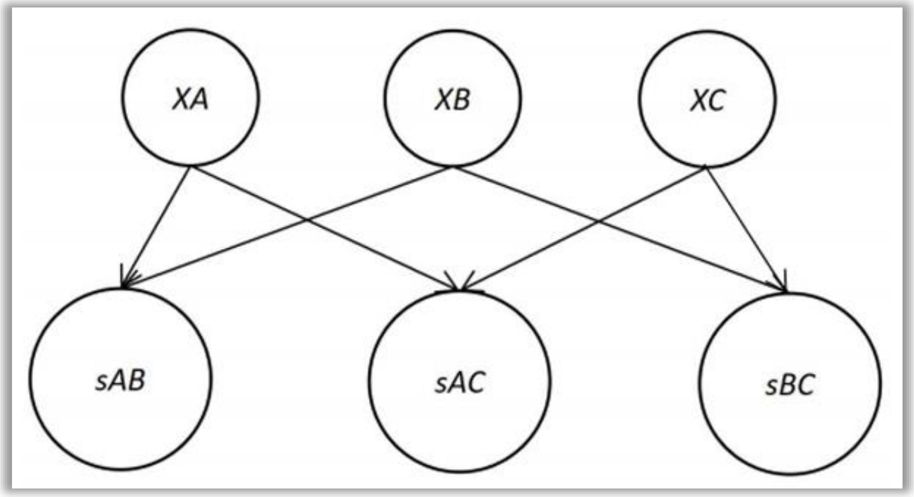



图 1 比赛问题的贝叶斯网络

记三队的实力为 XA, XB, XC，其先验分别满足分布 PA(X), PB(X), PC(X), 其中 X 取值{0，1，2，3}。一场比赛结果与队伍实力的关系的表现为条件分布，如 P(sAB|XA, XB)，其中 sAB 是 A 队对战 B 队时 A 队的结果，假设胜、平、负分别用 0，1，2 表示。根据实力和比赛结果的关系，构建贝叶斯网络图 1。实验任务为求 P(sBC|sAB=0, sAC=1)。

假设在足球比赛问题中，XA, XB, XC 结点的条件概率表 PA, PB, PC 分别是

```
PA=[0.3, 0.3, 0.2, 0.2]
PB=[0.4, 0.4, 0.1, 0.1]
PC=[0.2, 0.2, 0.3, 0.3]
```

因为实验中比赛结果取决于实力，因此 sAB, sAC, sBC 共享一个条件概率表 PS

```
PS=[
[[0.2,0.6,0.2], [0.1,0.3,0.6], [0.05,0.2,0.75], [0.01,0.1,0.89]],
[[0.6,0.3,0.1], [0.2,0.6,0.2], [0.1,0.3,0.6], [0.05,0.2,0.75]],
[[0.75,0.2,0.05], [0.6,0.3,0.1], [0.2,0.6,0.2], [0.1,0.3,0.6]],
[[0.89,0.1,0.01], [0.75,0.2,0.05], [0.6,0.3,0.1], [0.2,0.6,0.2]]]
```

PS 是一个 4×4×3 的表，$PS[i][j]$ 是一个比赛结果的分布，表示参赛两队的实力为 i, j 时比赛结果的分布。

## 2. 2. 解决方法

分别使用：①精确算法、②拒绝采样方法、③似然加权采样方法、④Gibbs 采样方法对问题进行求解。

### ① 精确算法

精确算法通过遍历所有可能的实力组合（XA, XB, XC），并使用条件概率表（CPT）对各个比赛结果的概率进行计算。计算公式为联合概率的条件概率形式：

P(sBC|sAB=0, sAC=1)=
α * ∑XA∑XB∑XC P(sBC|XB, XC) *
P(sAB=0|XA, XB) *
P(sAC=1|XA, XC) *
PA(XA) * PB(XB) * PC(XC)

代码实现如下：

```
# 直接计算（精确算法）方法
def direct_cal():
  res = [0,0,0]
  for XA in range(4):
    for XB in range(4):
      for XC in range(4):
        for sBC in range(3):
          res[sBC] += (
              PA[XA] * PB[XB] * PC[XC] *
              PS[XA][XB][0] * PS[XA][XC][1] * PS[XB][XC][sBC]
          )
  return normal(res) # normal(X) = X/sum(X)将计数变成概率
```

### ② 拒绝采样方法

拒绝采样通过先随机生成一系列样本，然后筛选符合已知条件（sAB=0, sAC=1）的样本，最后统计这些样本中 sBC 的分布情况。这个方法的核心步骤如下：

1. 随机采样每个队伍的实力（XA, XB, XC）
2. 根据条件概率表生成比赛结果
3. 筛选出符合前两场比赛结果的样本
4. 统计第三场比赛的结果

代码实现如下：

```
# 拒绝采样方法
def reject_sampling():
  n = 5000
  res = [0,0,0]
  for i in range(n):
    XA = np.random.choice(4,p = PA)
    XB = np.random.choice(4,p = PB)
    XC = np.random.choice(4,p = PC)
    sAB = np.random.choice(3,p = PS[XA][XB])
    sAC = np.random.choice(3,p = PS[XA][XC])
```

```
    sBC = np.random.choice(3,p = PS[XB][XC])
    if sAB == 0 and sAC == 1:
      res[sBC] += 1
  return normal(res)
```

### ③ 似然加权采样方法

似然加权采样通过对证据变量（sAB, sAC）加权，避免拒绝采样中因证据不匹配导致的样本浪费。对于每个样本，计算其与证据的匹配权重，最终将加权后的样本用于估计比赛结果。

代码实现如下：

```
# 似然加权采样方法
def likehood_weighting():
  n = 5000
  res = [0,10,0]
  for i in range(n):
    w = 1
    XA = np.random.choice(4,p = PA)
    XB = np.random.choice(4,p = PB)
    XC = np.random.choice(4,p = PC)

    w = w * PS[XA][XB][0] # sAB加权
    w = w * PS[XA][XC][1] # sAC加权

    sBC = np.random.choice(3,p = PS[XB][XC])
    res[sBC] += w
  return normal(res)
```

### ④ Gibbs 采样方法

Gibbs 采样从一个初始样本开始，依次更新每个非证据变量的取值，形成一系列相关的样本点。然后通过这些样本点对查询变量进行统计。

代码实现如下：

```
# Gibbs采样方法
def Gibs():
  n = 4999
  res = [0,0,0]
  XA,XB,XC,sAB,sAC,sBC = 0,0,1,0,1,1
  for k in range(n):
    _PA = normal([PA[i] * PS[i][XB][sAB] * PS[i][XC][sAC] \
         for i in range(4)])
    XA = np.random.choice(4,p = PA)
```

```
        _PB = normal([PB[i] *
                  PS[XA][i][sAB] *
                  PS[i][XC][sBC] \
                  for i in range(4)])
        XB = np.random.choice(4,p = PB)

        _PC = normal([PC[i] *
                  PS[XA][i][sAC] *
                  PS[XB][i][sBC] \
                  for i in range(4)])
        XC = np.random.choice(4,p = PC)

        sBC = np.random.choice(3,p = PS[XB][XC])
        res[sBC] += 1
    return normal(res)
```

# 三、 计算结果与讨论

## 3.1. 计算结果

实验数据 1：


精确算法计算结果：
[0.12358212738673048, 0.2842082503895563, 0.5922096222237132]
------------------------------------------------------------------------
拒绝采样结果：
[0.14923076923076922, 0.2723076923076923, 0.5784615384615385]
------------------------------------------------------------------------

似然加权采样结果：
[0.11952448692816127, 0.2975845578243105, 0.5828909552475283]

---

Gibbs 采样结果：
[0.21464292858571715, 0.3170634126825365, 0.46829365873174633]

实验数据 2：


精确算法计算结果：
[0.12358212738673048, 0.2842082503895563, 0.5922096222237132]

---

拒绝采样结果：
[0.10307692307692308, 0.2876923076923077, 0.6092307692307692]

---

似然加权采样结果：
[0.11629835111484853, 0.2842461046436885, 0.599455544241463]

---

Gibbs 采样结果：
[0.20144028805761152, 0.3120624124824965, 0.48649729945989195]

实验数据 3：


精确算法计算结果:
[0.12358212738673048, 0.2842082503895563, 0.5922096222237132]
---------------------------------------------------------------------------
拒绝采样结果:
[0.13864306784660768, 0.3112094395280236, 0.5501474926253688]
---------------------------------------------------------------------------
似然加权采样结果:
[0.12222743085539454, 0.29485201595873695,0.5829205531858684]
---------------------------------------------------------------------------
Gibbs 采样结果:
[0.20864172834566913, 0.3150630126025205, 0.47629525905181036]

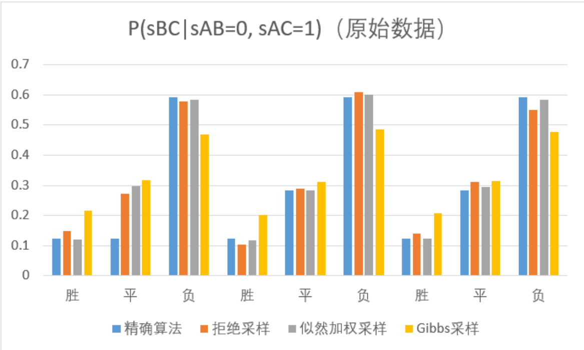
在本次实验中，通过不同的方法（精确算法、拒绝采样、似然加权采样、Gibbs 采样）对足球比赛结果进行了推断。实验结果表明：

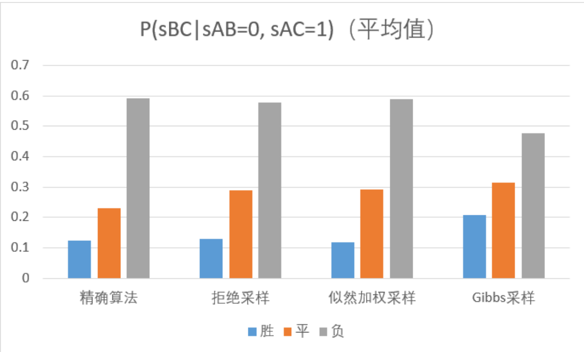
1. **精确算法**作为基准，给出的结果最为稳定。通过对所有可能的实力组合进行遍历，精确计算得出 BC 比赛结果的后验概率分布，实验三次的结果为：[0.1236, 0.2842, 0.5922]，表示 B 队胜、平、负的概率分布。
2. **拒绝采样**方法通过随机采样筛选出符合条件的样本，虽然该方法相对精确算法较为简便，但会有一定的样本浪费。实验结果与精确算法相差不大，但略有波动，表现为 B 队胜平负的概率略微不同，三次实验结果分别为：[0.1492, 0.2723, 0.5785]、[0.1031, 0.2877, 0.6092]、[0.1386, 0.3112, 0.5501]。
3. **似然加权采样**通过对符合条件的样本进行加权，可以减少样本浪费，使得结果更为接近精确解。三次实验结果表明加权采样在逼近精确解方面表现优异，概率分布分别为：[0.1195, 0.2976, 0.5829]、[0.1163, 0.2842, 0.5995]、[0.1222, 0.2949, 0.5829]。
4. **Gibbs 采样**在进行采样时，不断更新每个非证据变量，通过生成相关样本得出后验分布。尽管其结果较为接近，但由于其样本之间存在相关性，波动性稍大。三次实验结果为：[0.2146, 0.3171, 0.4683]、[0.2014, 0.3121, 0.4865]、[0.2086, 0.3151, 0.4763]。

## 3. 2. 实验讨论

| 原始数据 | 精确算法 | 拒绝采样 | 似然加权采样 | Gibbs采样 |
|---|---|---|---|---|
| 胜 | 0.123582127 | 0.149230769 | 0.119524487 | 0.214642929 |
| 平 | 0.123582127 | 0.272307692 | 0.297584558 | 0.317063413 |
| 负 | 0.592209622 | 0.578461538 | 0.582890955 | 0.468293659 |
| 胜 | 0.123582127 | 0.103076923 | 0.116298351 | 0.201440288 |
| 平 | 0.28420825 | 0.287692308 | 0.284246105 | 0.312062412 |
| 负 | 0.592209622 | 0.609230769 | 0.599455544 | 0.486497299 |
| 胜 | 0.123582127 | 0.138643068 | 0.122227431 | 0.208641728 |
| 平 | 0.28420825 | 0.31120944 | 0.294852016 | 0.315063013 |
| 负 | 0.592209622 | 0.550147493 | 0.582920553 | 0.476295259 |

| 平均值 | 精确算法 | 拒绝采样 | 似然加权采样 | Gibbs采样 |
|---|---|---|---|---|
| 胜 | 0.123582127 | 0.13031692 | 0.11935009 | 0.208241648 |
| 平 | 0.230666209 | 0.290403147 | 0.292227559 | 0.314729613 |
| 负 | 0.592209622 | 0.579279933 | 0.588422351 | 0.477028739 |




实验结果显示，精确算法在所有方法中结果最为稳定且准确，但其计算复杂度较高。拒绝采样由于样本浪费，导致结果的波动较大，尽管能大致反映出趋势，但在精度上略逊一筹。似然加权采样方法由于对样本进行加权，使得其结果与精确算法更为接近，且样本浪费较少。Gibbs 采样通过更新非证据变量来生成样本，结果也较为接近精确解，但由于样本相关性，波动较大。具体分析如下：

1. **精确求解方法**：能够直接计算出后验概率分布，但计算复杂度随着变量数量增加而迅速增长，导致计算时间过长。

2. **拒绝采样方法**：虽然简单易实现，但由于采样过程中会拒绝大量不符合证据的样本，导致效率低下，且收敛速度较慢。

3. **似然加权采样方法**：通过对证据变量加权，减少了样本浪费。相比拒绝采样，效率更高，且能够在较少的采样次数下得到较接近精确解的结果。

4. **Gibbs 采样方法**：通过依次更新非证据变量，能够快速收敛到稳定的后验分布。与其他采样方法相比，Gibbs 采样在计算复杂度和收敛速度方面表现最佳。

综上所述，**似然加权采样**在综合精度和计算成本方面表现最优，而 **Gibbs 采样**则更适用于大规模采样问题。

# 四、 作业总结

本次实验探讨了贝叶斯网络在不确定性推理中的应用，并以足球比赛为例，展示了如何利用贝叶斯网络预测比赛结果。通过本次实验，我深入学习并掌握了贝叶斯网络及其在不确定性推理中的应用。贝叶斯网络作为概率图模型，可以有效表示变量之间的依赖关系，并通过条件概率表进行推理。实验过程中，我使用了多种方法，包括精确算法和三种采样方法（拒绝采样、似然加权采样、Gibbs 采样），并使用这些方法对问题进行求解，经过实验数据统计分析后，这些方法在计算复杂度和精度上各有优劣。

在实验中，我不仅加深了对贝叶斯网络的理解，还学会了如何根据问题特点选择合适的推理算法。通过实验数据的分析，我发现似然加权采样在保持较高精度的同时能有效减少计算成本，是一种非常实用的近似算法。

此外，实验中通过调整采样次数观察了各近似方法与精确解的差异，进一步提升了我对不同采样方法的认识。这为我在处理大规模不确定性推理问题时提供了宝贵的经验和启发：

- 贝叶斯网络建模：需要明确变量之间的关系，并构建合理的条件概率表。
- 采样方法选择：需要根据问题的特点选择合适的采样方法，并考虑采样次数对结果的影响。
- 近似解的评估：需要评估近似解的精度和可靠性，并结合实际应用场景进行判断。

总体来说，本次实验让我系统地了解了贝叶斯网络在推理中的强大作用，并通过编程实现了各种推理算法，培养了我的动手能力和分析能力，对今后的学习和研究具有重要的帮助。

# 五、 参考文献

[1] 徐义春. 人工智能案例与实验[M]. 2024 年 5 月第 1 版. 清华大学出版社, 2024.

# 六、 实验代码

```
import numpy as np
import random

PA = [0.3, 0.3, 0.2, 0.2]
PB = [0.4, 0.4, 0.1, 0.1]
PC = [0.2, 0.2, 0.3, 0.3]

PS = [
    [[0.2, 0.6, 0.2], [0.1, 0.3, 0.6], [0.05, 0.2, 0.75], [0.01, 0.1, 0.89]],
    [[0.6, 0.3, 0.1], [0.2, 0.6, 0.2], [0.1, 0.3, 0.6], [0.05, 0.2, 0.75]],
    [[0.75, 0.2, 0.05], [0.6, 0.3, 0.1], [0.2, 0.6, 0.2], [0.1, 0.3, 0.6]],
    [[0.89, 0.1, 0.01], [0.75, 0.2, 0.05], [0.6, 0.3, 0.1], [0.2, 0.6, 0.2]]
```

```python
]

# 归一化处理
def normal(x):
    return [i / sum(x) for i in x]


# 直接计算（精确算法）方法
def direct_cal():
    res = [0, 0, 0]
    for XA in range(4):
        for XB in range(4):
            for XC in range(4):
                for sBC in range(3):
                    res[sBC] += (
                        PA[XA] * PB[XB] * PC[XC] *
                        PS[XA][XB][0] * PS[XA][XC][1] * PS[XB][XC][sBC]
                    )
    return normal(res)  # normal(X) = X/sum(X)将计数变成概率


# 拒绝采样方法
def reject_sampling():
    n = 5000
    res = [0, 0, 0]
    for i in range(n):
        XA = np.random.choice(4, p = PA)
        XB = np.random.choice(4, p = PB)
        XC = np.random.choice(4, p = PC)
        sAB = np.random.choice(3, p = PS[XA][XB])
        sAC = np.random.choice(3, p = PS[XA][XC])
        sBC = np.random.choice(3, p = PS[XB][XC])
        if sAB == 0 and sAC == 1:
            res[sBC] += 1
    return normal(res)


# 似然加权采样方法
def likehood_weighting():
    n = 5000
    res = [0, 10, 0]
    for i in range(n):
        w = 1
        XA = np.random.choice(4, p = PA)
```

```
        XB = np.random.choice(4,p = PB)
        XC = np.random.choice(4,p = PC)

        w = w * PS[XA][XB][0] # sAB加权
        w = w * PS[XA][XC][1] # sAC加权

        sBC = np.random.choice(3,p = PS[XB][XC])
        res[sBC] += w
    return normal(res)


# Gibbs采样方法
def Gibs():
    n = 4999
    res = [0,0,0]
    XA,XB,XC,sAB,sAC,sBC = 0,0,1,0,1,1
    for k in range(n):
        _PA = normal([PA[i] * PS[i][XB][sAB] * PS[i][XC][sAC] \
                for i in range(4)])
        XA = np.random.choice(4,p = PA)

        _PB = normal([PB[i] *
                PS[XA][i][sAB] *
                PS[i][XC][sBC] \
                for i in range(4)])
        XB = np.random.choice(4,p = PB)

        _PC = normal([PC[i] *
                PS[XA][i][sAC] *
                PS[XB][i][sBC] \
                for i in range(4)])
        XC = np.random.choice(4,p = PC)

        sBC = np.random.choice(3,p = PS[XB][XC])
        res[sBC] += 1
    return normal(res)


# 主函数
if __name__ == "__main__":
    print("精确算法计算结果：\n",direct_cal())
    print('-------------------------------------------------------------------------------------------------')
    print("拒绝采样结果：\n",reject_sampling())
```

```python
print('------------------------------------------------------------------------------')
print("似然加权采样结果：\n",likehood_weighting())
print('------------------------------------------------------------------------------')
print("Gibbs采样结果：\n",Gibs())
```