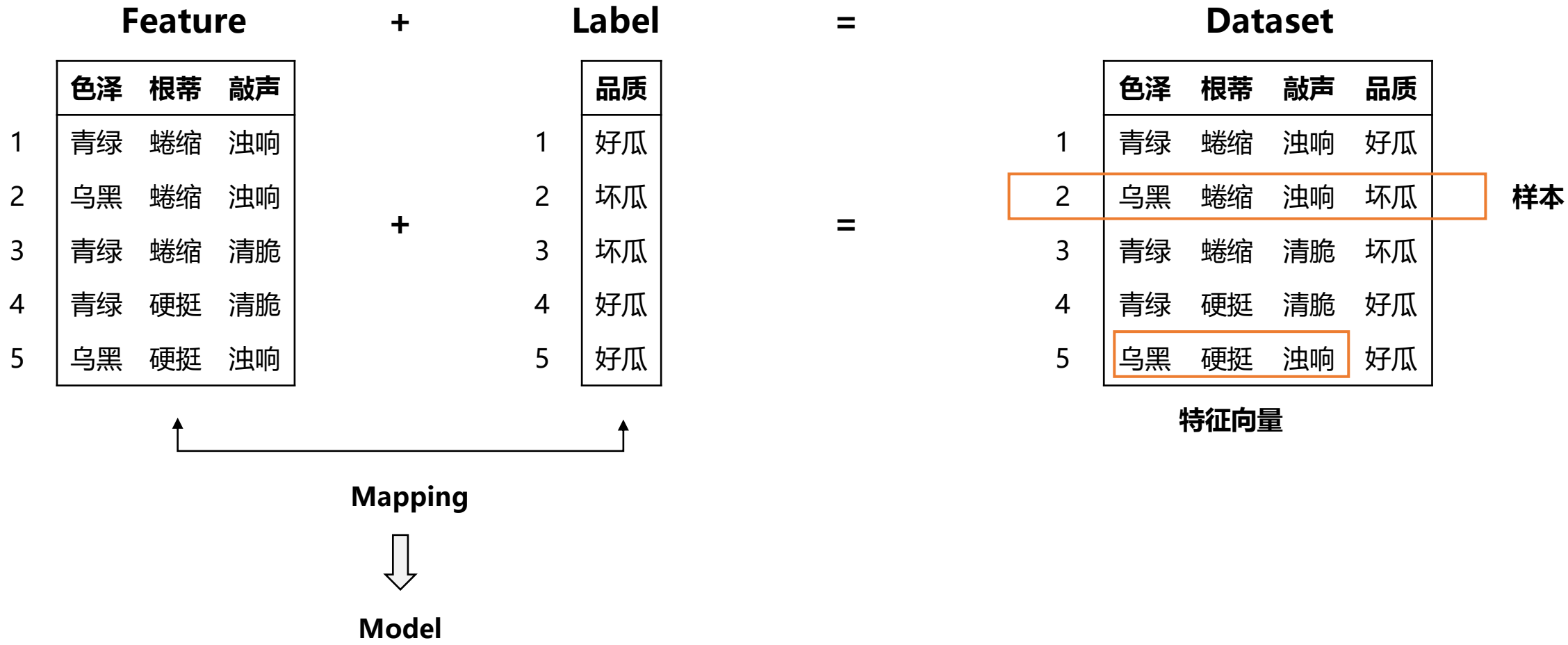
第一部分

基于概率图模型的贝叶斯机器学习理论

1. 机器学习简单理解


Feature
+
Label
=
Dataset
色泽 根蒂 敲声
1 青绿 蜷缩 浊响
2 乌黑 蜷缩 浊响
3 青绿 蜷缩 清脆
4 青绿 硬挺 清脆
5 乌黑 硬挺 浊响
+
品质
1 好瓜
2 坏瓜
3 坏瓜
4 好瓜
5 好瓜
=
色泽 根蒂 敲声 品质
1 青绿 蜷缩 浊响 好瓜
2 乌黑 蜷缩 浊响 坏瓜
3 青绿 蜷缩 清脆 坏瓜
4 青绿 硬挺 清脆 好瓜
5 乌黑 硬挺 浊响 好瓜
样本
特征向量
Mapping
Model

2. 贝叶斯决策论

- 贝叶斯决策论(Bayesian decision theory):

在所有相关概率都已知的理想情形下，如何根据已知概率和误判风险来进行最合理的分类。

- 贝叶斯决策论的目标——贝叶斯判定准则(Bayes decision rule):

选择能够最小化分类错误率的贝叶斯分类器

$$h^*(\vec{x}) = \underset{c \in y}{\arg\max}\, P(c|\vec{x})$$

即对每个样本 $\vec{x}$，选择能使后验概率$P(c|\vec{x})$最大的类别标记c。

- 判别式模型(discriminative models)

  给定$\vec{x}$，通过直接建模$P(c|\vec{x})$, 来预测c。(决策树、BP神经网络、支持向量机都属于这一类)

- 生成式模型(generative models)

  先对联合概率分布$P(\vec{x}, c)$ 建模，再由此获得$P(c|\vec{x})$。

先验　C类的条件概率，又称“似然”

后验概率　$P(c|\vec{x}) = \dfrac{P(\vec{x}, c)}{P(\vec{x})} = \dfrac{P(c)P(\vec{x}|c)}{P(\vec{x})}$

证据因子

3. 朴素贝叶斯

假设所有样本独立同分布　　全概率公式

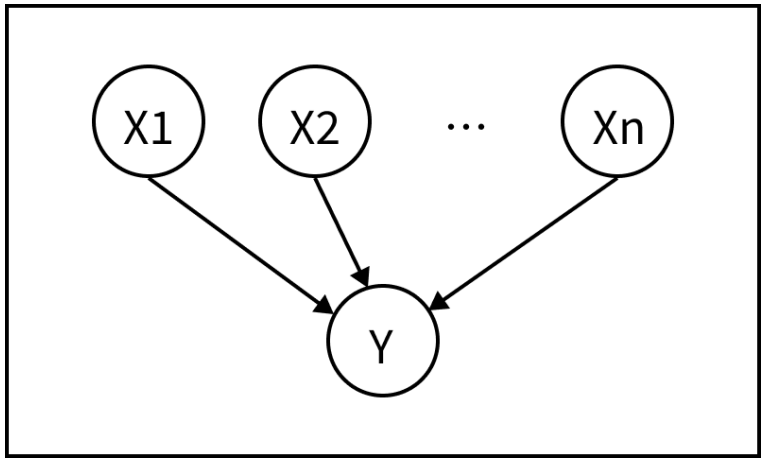
$$P(c|\vec{x}) = \frac{P(\vec{x},c)}{P(\vec{x})} = \frac{P(c)\,P(\vec{x}|c)}{P(\vec{x})} = \frac{P(c)}{P(\vec{x})}\prod_{i=1}^{d} P(x_i|c)$$

此时的贝叶斯判定准则为：

$$h_{nb}(\vec{x}) = \underset{c \in y}{\arg\max} P(c)\prod_{i=1}^{d} P(x_i|c)$$

其中：

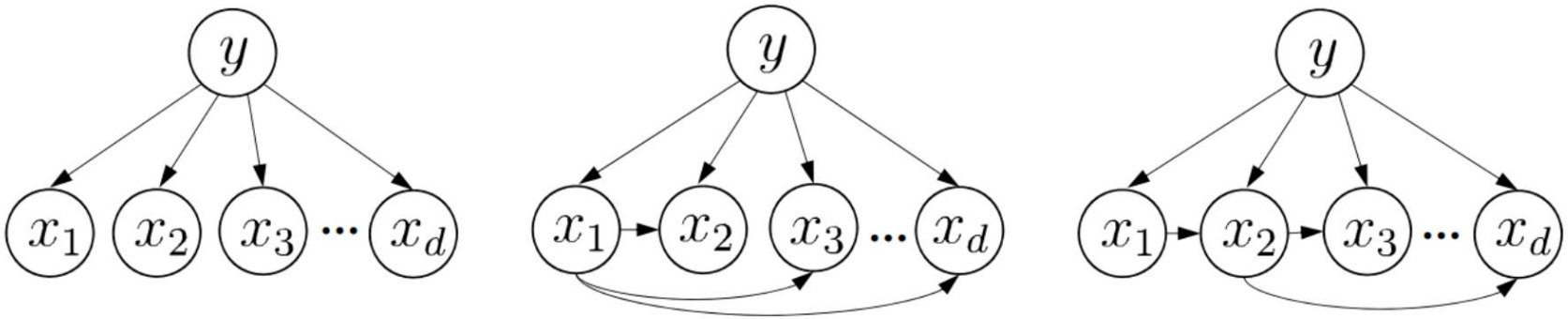
- $P(c)$很容易估计： $P(c) = \frac{|D_C|}{|D|}$
- $P(x_i|c)$表示在c类别的前提下，第i个属性取值为x_i的概率：
  - 对于离散属性： $P(x_i|c) = \frac{|D_{C,x_i}|}{|D_C|}$
  - 对于连续属性：假设 $P(x_i|c) \sim N(\mu_{c,i}, \sigma_{c,i}^2)$，则$P(x_i|c) = \frac{1}{\sqrt{2\pi}\sigma_{c,i}}\exp(-\frac{(x_i-\mu_{c,i})^2}{2\sigma_{c,i}^2})$



Naive Bayes

4. 半朴素贝叶斯

- 朴素贝叶斯终究还是太理想化了，现实任务的各项属性往往不是独立同分布的。
- 独依赖估计(ODE)——假设每个属性在类别之外最多仅依赖一个其他属性
- 超父(SPODE)——假设所有属性都依赖于同一个属性称为“超父”(super-parenet)，然后通过交叉验证等模型选择方法来确定超父属性。
- TAN——基于“最大带权生成树算法”的基础上，将属性间依赖关系简化。



(a) NB　　(b) SPODE　　(c) TAN

5. 贝叶斯网络

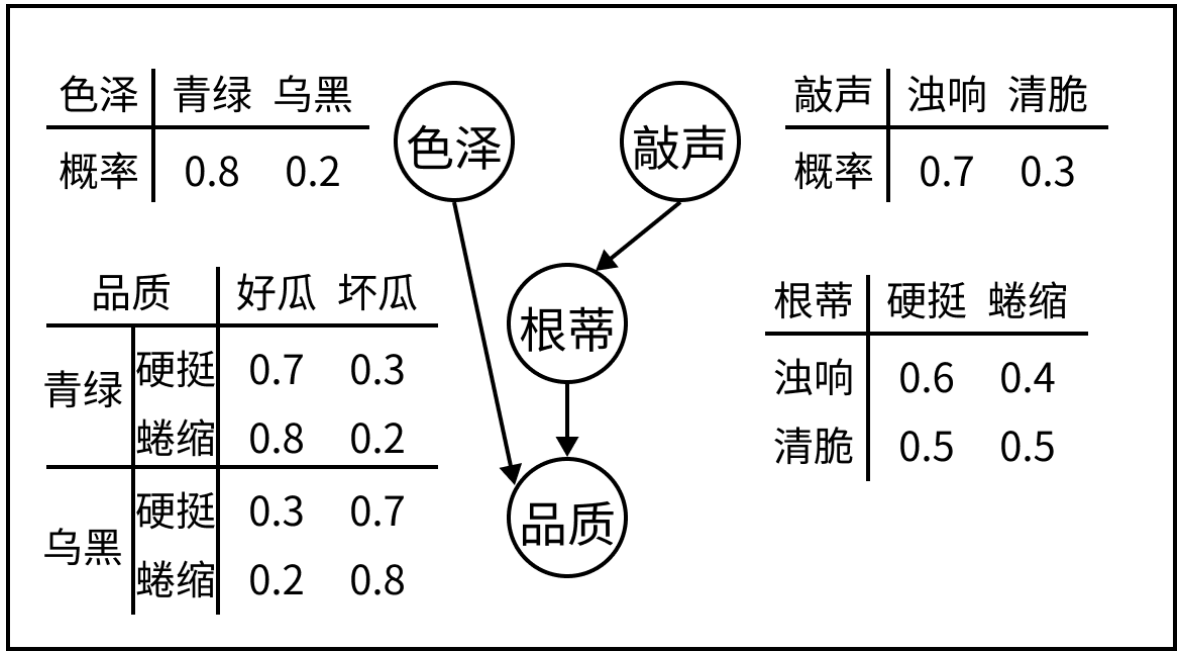
5.1 基本概念

贝叶斯网络(Bayesian Network)亦称为“信念网”(Belief Network)，它借助有向无环图(DAG)来刻画属性间的依赖关系，并使用条件概率表(CPT)或条件概率分布(CPD)来描述属性的联合概率分布。

- 组成部分
- ① 结构G——有向无环图
- ② 参数Θ——定量描述属性间的依赖关系

	色泽	根蒂	敲声	品质
1	青绿	蜷缩	浊响	好瓜
2	乌黑	蜷缩	浊响	坏瓜
3	青绿	蜷缩	清脆	坏瓜
…	…	…	…	…
n	乌黑	硬挺	浊响	好瓜

贝叶斯建模



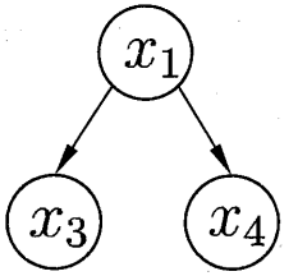
5. 贝叶斯网络

5.2 基本结构

贝叶斯网络$B =< G, \Theta >$将属性$(x_1, x_2, \dots, x_d)$的联合概率分布定义为：

$$P_B(x_1, x_2, \dots, x_d) = \prod_{i=1}^{d} P_B(x_i|\pi_i) = \prod_{i=1}^{d} \theta_{x_i|\pi_i}$$

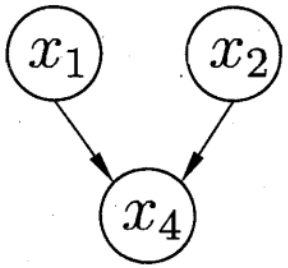
贝叶斯网络一共有三种基本结构：



同父结构

$P(x_1, x_3, x_4) = P(x_1)\ P(x_3|x_1)P(x_4|x_1)$

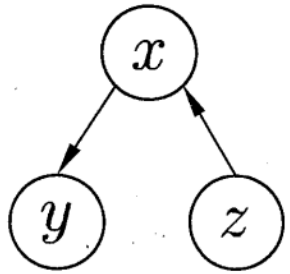
给定x_1，则x_3, x_4相互独立



V型结构

$P(x_1, x_2, x_4) = P(x_1)\ P(x_2)P(x_4|x_1, x_2)$

给定x_4，则x_1, x_2一定不独立



顺序结构

$P(x, y, z) = P(z)\ P(x|z)P(y|x)$

给定x，则y, z 相互独立

5. 贝叶斯网络

5.3 参数学习与结构学习

- 参数学习：若贝叶斯网络结构已知，即属性间的依赖关系已知，则只需根据训练样本计算每个节点的条件概率分布(CPD)即可。

- 结构学习：若贝叶斯网络结构未知，则需要在训练集中先找出结构最“恰当”的贝叶斯网络：
  - 评分函数(Score Function)：评估贝叶斯网络与训练数据的契合程度
  - 最小描述长度准则(Minimal Description Length, MDL)——选择综合编码长度最短的贝叶斯网络

- 然而，在网络结构中搜索最优贝叶斯网络是一个NP难问题，有两种方法：
  - ① 贪心法：从某个网络结构出发，每次调整一条边，直至评分函数不再增加
  - ② 约束法：给网络结构施加约束

- Pgmpy库给出的结构学习方法：
  - ① PC (Constraint-Based Estimator)算法——基于约束的算法
  - ② 爬山算法——用于启发式爬山搜索 DAG 的类，以从数据中学习网络结构。
  - ③ MMHC算法——实现 MMHC 混合结构估计过程，用于从离散数据中学习贝叶斯网络。
  - ④ 结构分数——评分的方法

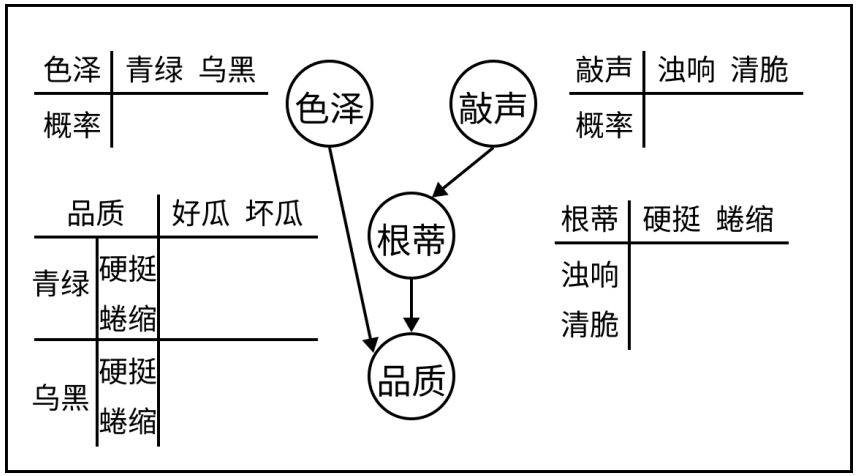
如果训练样本的属性不完整，存在“未观测”变量下，使用EM算法对模型进行参数估计。
- 未观测变量被称为“隐变量”(latent variable)。
- EM算法分为E步和M步：
- E步：根据训练数据推断出最优的隐变量
- M步：若隐变量已知，则直接对参数做最大似然估计

5. 贝叶斯网络

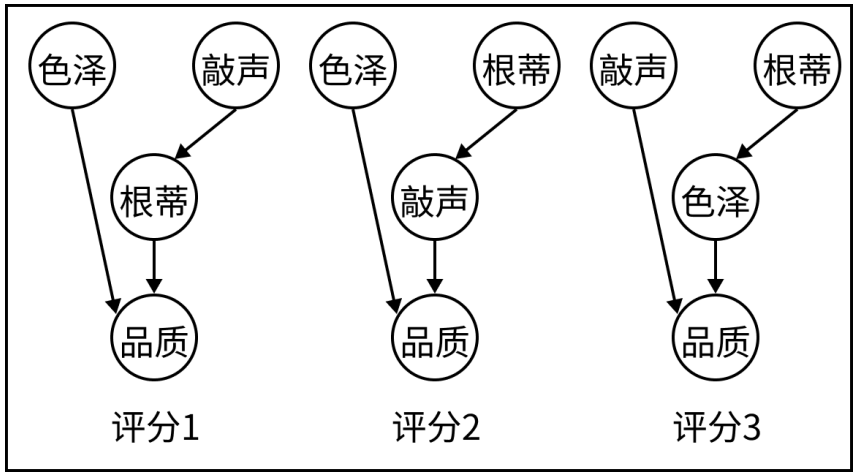
5.3 参数学习与结构学习

	色泽	根蒂	敲声	品质
1	青绿	蜷缩	浊响	好瓜
2	乌黑	蜷缩	浊响	坏瓜
3	青绿	蜷缩	清脆	坏瓜
…	…	…	…	…
n	乌黑	硬挺	浊响	好瓜

参数学习



结构学习

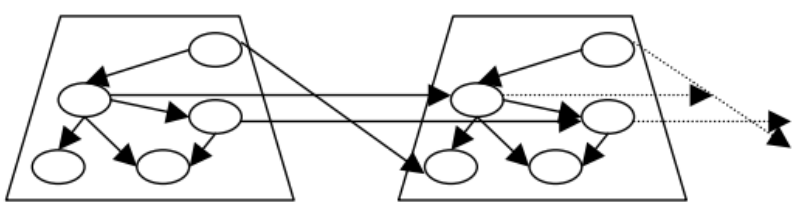


6. 动态贝叶斯网络

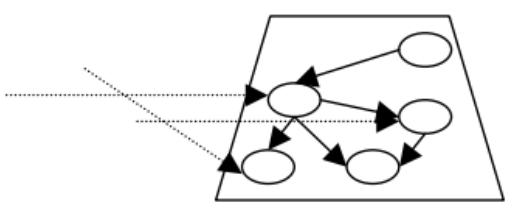
动态贝叶斯网络在贝叶斯网络的基础上加入时序状态，我们通常假设动态贝叶斯网络满足两个条件：

①网络拓扑结构不随时间发生改变，即除去初始时刻，其余时刻的变量及其概率依存关系相同。

②满足一阶隐马尔可夫条件：随机过程中某事件的发生只取决于它的上一事件，是「无记忆」过程，即t时刻的所有属性至于t-1时刻相关。

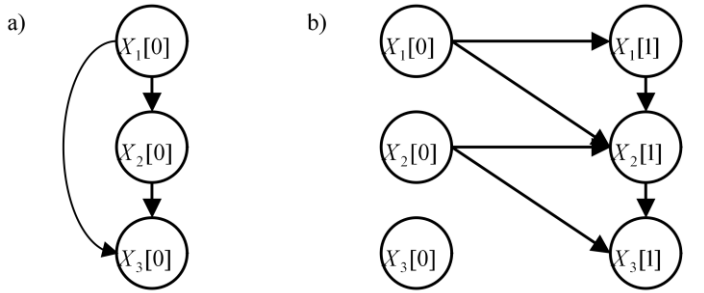


T=1　　T=2



T=t-1　　T=t

网络展开



动态贝叶斯网络图

先验网络
(初始时刻的
网络模型)

+

转移网络
(连续两个时刻间
结点的依赖关系)

动态贝叶斯网络的结构可以理解成：

①每个时间片的结构是统一且一定的，每个时间片就是个普通的贝叶斯网络。

②时间片和时间片之间的某一个状态可能依赖于上一个时刻的某几个状态和/或当前时刻的某几个状态

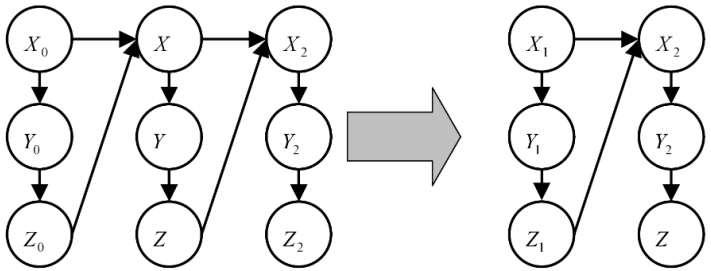
6. 动态贝叶斯网络

更深入下去，动态贝叶斯网络存在两项重要挑战：

① 在概率推断中，我们必须要对网络模型进行展开（unrolling or rolling up），同时要保证各结点的依赖关系，以及隐结点对观测结点的影响。

② 在参数学习中，主要难点在于——部分隐结点的概率分布情况的正确程度（correctness）是很难获知的。

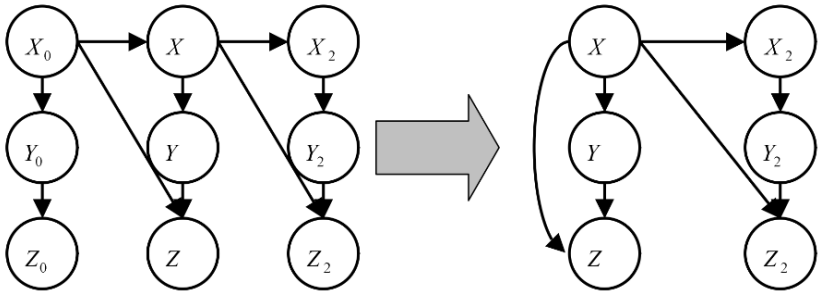
对于网络展开问题，动态贝叶斯网络分为时变网络和时不变网络。



时态不变网络，只有结点X受到上个时刻结点的影响

时不变网络

初始时刻的先验网络和单个时间片的转移网络结构相同



时态变化网络

时态变化网络

结点X、Z都受上一时刻影响，因此二者之间是不独立的，转移网络相较于先验网络增加了X指向Z的CPD。

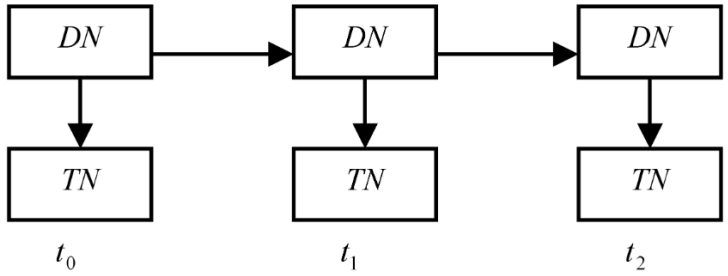
6. 动态贝叶斯网络

DBN中的结点分为三个类别：

- dynamic nodes（DN）：随时间进行演变的对象
- static nodes（SN）：不随时间进行演变的对象
- temporary nodes（TN）：不同时刻接收不同的取值的对象，也称为evidence nodes

2.4.1 DN和TN间依赖

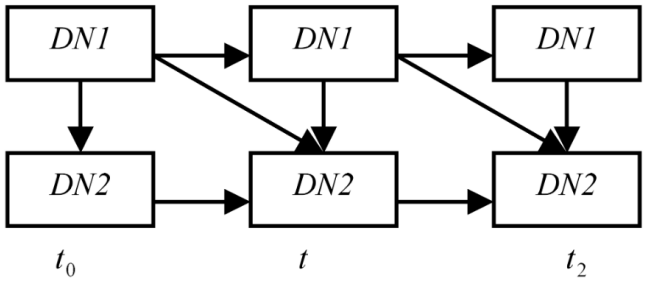
TN也称为evidence nodes。



DN和TN间依赖，TN只受到DN的独立影响

2.4.2 DN间依赖

假如上图中的TN也会随着时间进行动态变化，则可以表达为下图的形式。



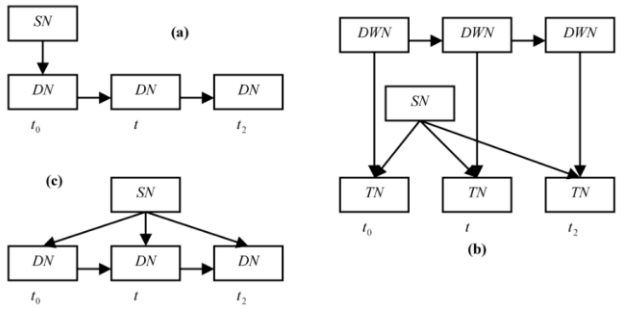
DN间依赖，DN2受到DN1在不同时刻的影响

2.4.3 SN和DN间依赖

一个DN可以有父SN，如下图(a)所示。

在图(b)中，DN被认为是在每个时刻创建的TN。同时，DWN被认为是不受SN影响的DN。底下的TN受到SN和DWN的影响。

图(c)中，SN也可以被看做DN，以此来简化该模型，但是可能会导致更为不准确的推断结果。



SN和TN间依赖

6. 动态贝叶斯网络

补充：隐马尔可夫模型（最特殊最精简的动态贝叶斯网络）

隐马尔可夫模型实际上可以看作结构最简单的动态贝叶斯网络。隐马尔可夫模型同样满足一阶马尔可夫条件。

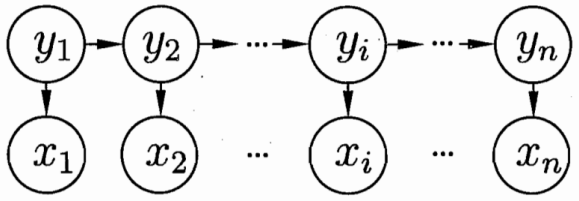
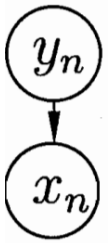


图 14.1 隐马尔可夫模型的图结构

状态变量(隐变量)

观测变量



每个时刻的贝叶斯网络涉及两个节点

不过HMM解决的问题是从观测序列$\vec{x} = (x_1, x_2, \dots, x_t)$预测状态序列$\vec{y} = (y_1, y_2, \dots, y_t)$，而不是仅仅求解当前时刻节点的状态变量$y_n$的概率，这和贝叶斯网络有本质区别。

隐马尔可夫模型涉及三种参数：

① 状态转移概率：状态变量由当前时刻y_t转到下一时刻y_{t+1}下状态变化的概率矩阵，称为矩阵A。

② 输出观测概率：当前时刻下根据当前状态y_t获得各观测值x_t的概率矩阵，称为矩阵B。

③ 初始状态概率：模型在初始时刻各状态y_1出现的概率，称为矩阵π。

隐马尔可夫模型$\boldsymbol{\lambda} = [\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}]$

隐马尔可夫模型解决三个基本问题：

① 给定模型$\boldsymbol{\lambda} = [\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}]$，计算观测序列$\vec{x} = (x_1, x_2, \dots, x_t)$产生概率。

②给定模型$\boldsymbol{\lambda} = [\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}]$和观测序列$\vec{x} = (x_1, x_2, \dots, x_t)$，找到与之匹配的状态序列$\vec{y} = (y_1, y_2, \dots, y_t)$。

③ 给定$\vec{x} = (x_1, x_2, \dots, x_t)$，如何调整模型$\boldsymbol{\lambda} = [\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{\pi}]$使其更好的描述观测数据。

第二部分

基于Pgmpy库和GeNIe软件搭建动态贝叶斯网络

1. 机器学习相关工具

- Python——编程语言，借助Python.exe实现编程语言的解释
- Anaconda——Python的正式发行版
- Pip/Conda——包管理器，以命令行形式安装、卸载和管理各类第三方开发包
- PyCharm/Jupyter Notebook——集成开发环境/编译器
- 虚拟环境的概念——建议使用Python开发熟练使用虚拟环境
- 机器学习常用框架：
  - PyTorch/TenslowFlow——基于深度学习神经网络的大模型框架(CNN、LSTM、RNN、GAN等等以神经网络为结构，依靠张量Tensor计算的模型)
  - Scikit-learn库——精小的机器学习框架，适合典型机器学习（决策树/随机森林/SVM/k-means/聚类/朴素贝叶斯/集成学习等等)
- 不过上述这些框架并不适合以概率图模型为结构的贝叶斯网络。目前开发贝叶斯网络的工具有：
  - Pgmpy库 for Python：测试成功贝叶斯网络的建模和学习，但仅支持离散数据输入，且动态贝叶斯网络不支持参数学习和结构学习
  - BNT工具包 for MATLAB：功能最全、最稳定的贝叶斯网络工具了，很多文献都用它来搭建和训练，不过上次更新已经是2014年。
  - bnlearn for R语言：因为不懂R语言，所以还没尝试
  - GeNIe软件：美国匹兹堡大学的贝叶斯网络可视化软件，可以可视化实现动静态贝叶斯网络的搭建。
  - 还有一些其他的python第三方包，不过功能都不全，官方文档和教程都不全。

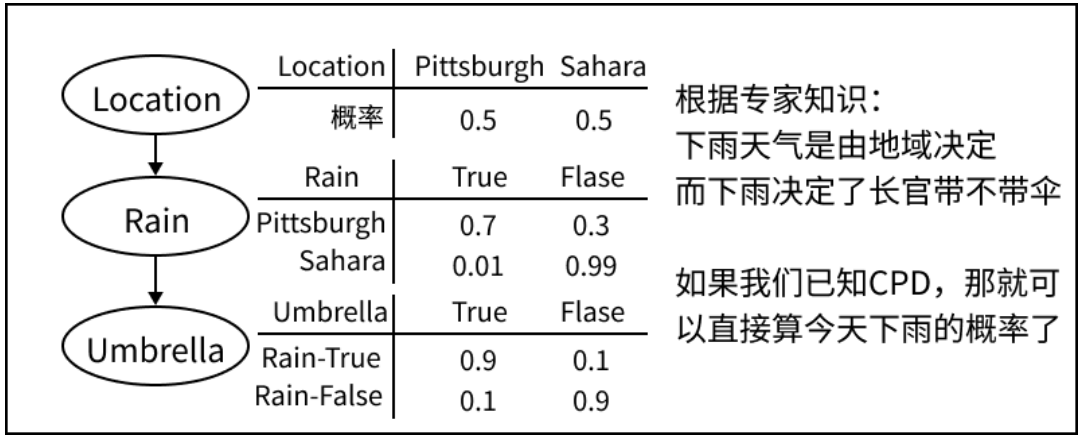
2. 贝叶斯网络代码实践

假设情境：

- 有一个人被关在某个地方的监狱，这个地方可能是匹兹堡或者撒哈拉——【Location】 Pittsburgh/Sahara
- 每天长官会过来巡查，长官来的时候有时会带着伞——【Umbrella】 Ture/False
- 问这个人能否通过长官带伞和他所处的位置推测今天有没有下雨——【Rain】 True/False

① 贝叶斯网络+赋值CPD方法

```python
# step1 定义贝叶斯网络模型
from pgmpy.models import BayesianNetwork
model = BayesianNetwork([('Location', 'Rain'),
                         ('Rain', 'Umbrella')])

# step2 添加概率
from pgmpy.factors.discrete import TabularCPD
cpd_Location = TabularCPD(variable='Location', variable_card=2,
                          values=[[0.5], [0.5]])
cpd_Rain = TabularCPD(variable='Rain', variable_card=2,
                      values=[[0.7, 0.01], [0.3, 0.99]],
                      evidence=['Location'],
                      evidence_card=[2])
cpd_Umbrella = TabularCPD(variable='Umbrella', variable_card=2,
                          values=[[0.9, 0.1], [0.1, 0.9]],
                          evidence=['Rain'],
                          evidence_card=[2])

# step3 添加概率分布到贝叶斯网络模型
model.add_cpds( *cpds: cpd_Location, cpd_Rain, cpd_Umbrella)

# step4 进行推理,输入特征，输出概率结果
from pgmpy.inference import VariableElimination
cancer_infer = VariableElimination(model)
q = cancer_infer.query(variables=['Rain'], evidence={'Location': 1, 'Umbrella': 1})
print(q)
```



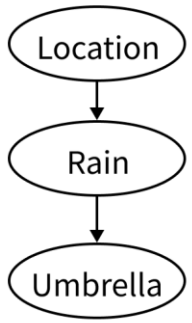

```
D:\Software\Anaconda3\envs\pgmpy\python.exe "D:\WorkSpace\PyCharm Project\231027 BayesPresentation\1 BN.py"
+---------+-------------+
| Rain    |   phi(Rain) |
+=========+=============+
| Rain(0) |      0.0011 |
+---------+-------------+
| Rain(1) |      0.9989 |
+---------+-------------+

Process finished with exit code 0
```

2. 贝叶斯网络代码实践

② 贝叶斯网络的参数学习

准备数据集:

	A	B	C
1	Location	Rain	Umbrella
2	Pittsburgh	TRUE	FALSE
3	Pittsburgh	FALSE	TRUE
4	Sahara	FALSE	TRUE
5	Sahara	FALSE	FALSE
6	Pittsburgh	FALSE	FALSE
7	Pittsburgh	TRUE	TRUE
8	Sahara	TRUE	TRUE
9	Sahara	FALSE	FALSE
10	Sahara	FALSE	TRUE
11	Pittsburgh	FALSE	FALSE
12	Sahara	FALSE	FALSE
13	Pittsburgh	FALSE	FALSE
14	Pittsburgh	TRUE	TRUE



根据专家知识:
下雨天气是由地域决定
而下雨决定了长官带不带伞

我们通过参数学习获取CPD

源码:

```python
import pandas as pd
from pgmpy.models import BayesianNetwork

# step1 读取数据集
data = pd.read_csv("./DataSet/BNData.csv")

# step2 搭建贝叶斯网络
model = BayesianNetwork([('Location', 'Rain'),
                         ('Rain', 'Umbrella')])

# step3 训练模型
model.fit(data)

# step4 进行推理,输入特征，输出概率结果
from pgmpy.inference import VariableElimination
cancer_infer = VariableElimination(model)
q = cancer_infer.query(variables=['Rain'], evidence={'Location': 'Sahara', 'Umbrella': 1})
print(q)

```

结果:

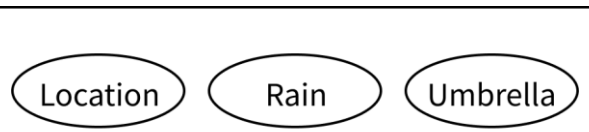
```
D:\Software\Anaconda3\envs\pgmpy\python.exe "D:\WorkSpace\PyCharm Project\231027 BayesPresentation\2 BN Parameter Learning.py"
+------------+------------+
| Rain       |   phi(Rain) |
+============+============+
| Rain(False) |     0.9294 |
+------------+------------+
| Rain(True)  |     0.0706 |
+------------+------------+

Process finished with exit code 0
```

2. 贝叶斯网络代码实践

③ 贝叶斯网络的结构学习

准备数据集:

	A	B	C
1	Location	Rain	Umbrella
2	Pittsburgh	TRUE	FALSE
3	Pittsburgh	FALSE	TRUE
4	Sahara	FALSE	TRUE
5	Sahara	FALSE	FALSE
6	Pittsburgh	FALSE	FALSE
7	Pittsburgh	TRUE	TRUE
8	Sahara	TRUE	TRUE
9	Sahara	FALSE	FALSE
10	Sahara	FALSE	TRUE
11	Pittsburgh	FALSE	FALSE
12	Sahara	FALSE	FALSE
13	Pittsburgh	FALSE	FALSE
14	Pittsburgh	TRUE	TRUE



这下连专家知识都没了
需要从各类属性中选择最优的
贝叶斯网络
再进行训练

源码:

```python
import pandas as pd
from pgmpy.models import BayesianNetwork


# step1 读取数据集
data = pd.read_csv("./DataSet/BNData.csv")

# step2 使用爬山算法选择合适Bayes模型
from pgmpy.estimators import HillClimbSearch
hc = HillClimbSearch(data)
best_model = hc.estimate(scoring_method='bicscore')
print(best_model.edges())

# step3 建立Bayes模型
model = BayesianNetwork(best_model)

# step4 训练模型
model.fit(data)

# step5 进行推理,输入特征，输出概率结果
from pgmpy.inference import VariableElimination
cancer_infer = VariableElimination(model)
q = cancer_infer.query(variables=['Rain'], evidence={'Location': 'Sahara', 'Umbrella': 1})
print(q)
```

结果:

```
D:\Software\Anaconda3\envs\pgmpy\python.exe "D:\WorkSpace\PyCharm Project\231027 BayesPresentation\3 BN Structure Learning.py"
  0%|          | 2/1000000 [00:00<1:56:40, 142.86it/s]
[('Rain', 'Location'), ('Rain', 'Umbrella')]
+-------------+-------------+
| Rain        |   phi(Rain) |
+=============+=============+
| Rain(False) |      0.9294 |
+-------------+-------------+
| Rain(True)  |      0.0706 |
+-------------+-------------+

Process finished with exit code 0
```

2. 贝叶斯网络代码实践

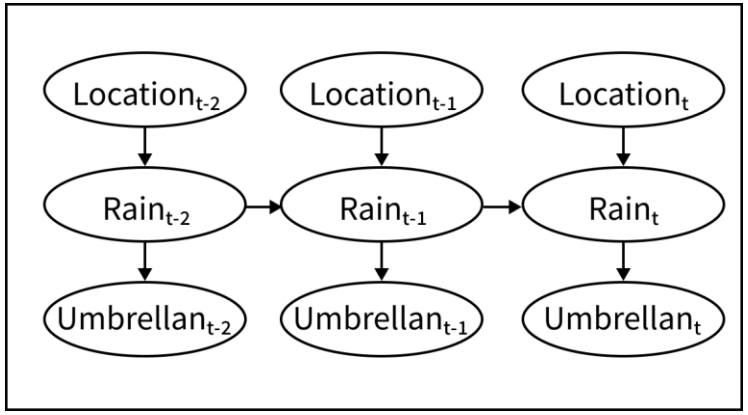
④ 动态贝叶斯网络

准备数据集:

	A	B	C	D	E	F	G	H	I
1	Location	Rain	Umbrella	Location	Rain_1	Umbrella_1	Location	Rain_2	Umbrella_2
2	Pittsburgh	TRUE	FALSE	Pittsburgh	TRUE	TRUE	Pittsburgh	TRUE	TRUE
3	Pittsburgh	FALSE	TRUE	Pittsburgh	FALSE	TRUE	Pittsburgh	FALSE	TRUE
4	Sahara	FALSE	TRUE	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE
5	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE	Sahara	FALSE	TRUE
6	Pittsburgh	FALSE	FALSE	Pittsburgh	TRUE	TRUE	Pittsburgh	TRUE	TRUE
7	Pittsburgh	TRUE	TRUE	Pittsburgh	FALSE	FALSE	Pittsburgh	TRUE	TRUE
8	Sahara	TRUE	TRUE	Sahara	FALSE	FALSE	Sahara	FALSE	TRUE
9	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE
10	Sahara	FALSE	TRUE	Sahara	FALSE	TRUE	Sahara	FALSE	FALSE
11	Pittsburgh	FALSE	FALSE	Pittsburgh	TRUE	FALSE	Pittsburgh	TRUE	TRUE
12	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE
13	Pittsburgh	FALSE	FALSE	Pittsburgh	TRUE	TRUE	Pittsburgh	FALSE	FALSE
14	Pittsburgh	TRUE	TRUE	Pittsburgh	FALSE	FALSE	Pittsburgh	TRUE	TRUE
15	Sahara	FALSE	TRUE	Sahara	FALSE	TRUE	Sahara	FALSE	TRUE
16	Pittsburgh	TRUE	TRUE	Pittsburgh	FALSE	FALSE	Pittsburgh	FALSE	FALSE
17	Sahara	FALSE	FALSE	Sahara	FALSE	TRUE	Sahara	FALSE	FALSE
18	Pittsburgh	TRUE	TRUE	Pittsburgh	FALSE	FALSE	Pittsburgh	FALSE	FALSE
19	Pittsburgh	TRUE	TRUE	Pittsburgh	TRUE	TRUE	Pittsburgh	TRUE	TRUE
20	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE
21	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE
22	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE
23	Pittsburgh	TRUE	TRUE	Pittsburgh	TRUE	TRUE	Pittsburgh	FALSE	FALSE
24	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE
25	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE	Sahara	FALSE	FALSE
26	Pittsburgh	TRUE	TRUE	Pittsburgh	TRUE	TRUE	Pittsburgh	TRUE	TRUE
27	Pittsburgh	TRUE	TRUE	Pittsburgh	TRUE	TRUE	Pittsburgh	TRUE	TRUE

输入连续三天的数据

动态贝叶斯网络:

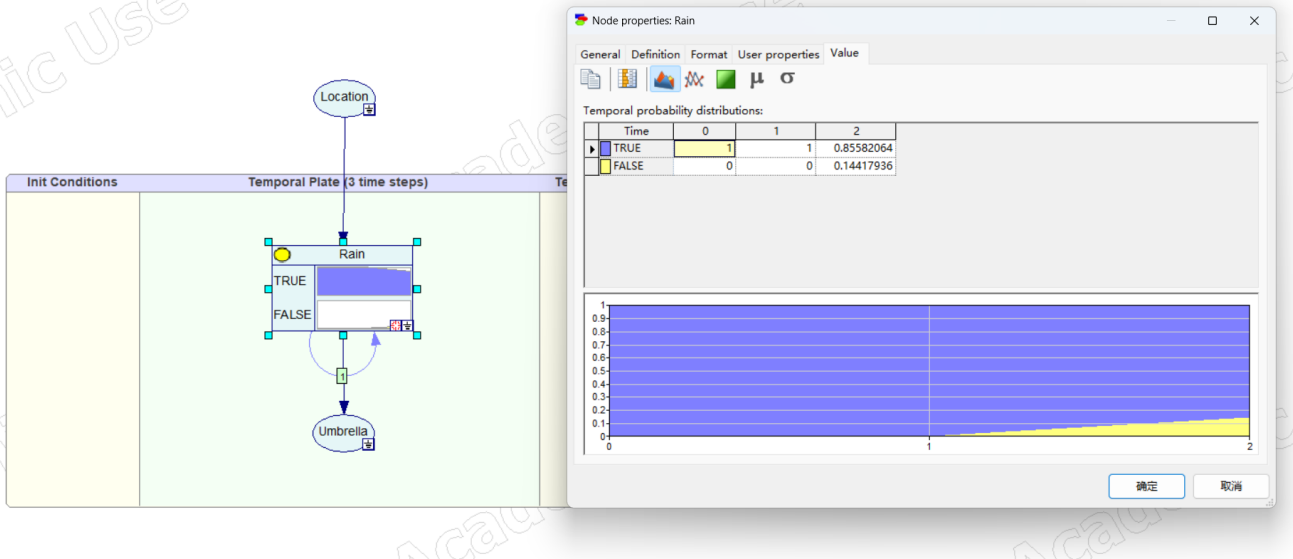



```python
import pandas as pd
import pgmpy.inference
from pgmpy.models import DynamicBayesianNetwork as DBN

# step1 读取数据集
colnames = []
for t in range(3):
    colnames.extend([("Location", t), ("Rain", t), ("Umbrella", t)])
data = pd.read_csv("./DataSet/DBNData.csv")
df = pd.DataFrame(data.values, columns=colnames)
df.replace( to_replace: False,  value: 0, inplace=True)
df.replace( to_replace: True,  value: 1, inplace=True)
df.replace( to_replace: 'Pittsburgh',  value: 1, inplace=True)
df.replace( to_replace: 'Sahara',  value: 0, inplace=True)
print(df)

# step2 搭建动态贝叶斯网络
model = DBN()
model.add_nodes_from(["Location", "Rain", "Umbrella"])
model.add_edges_from(
    [
        (("Location", 0), ("Rain", 0)),
        (("Rain", 0), ("Umbrella", 0)),
        (("Rain", 0), ("Rain", 1))
    ]
)
# step3 训练模型
model.fit(df)

# step4 进行推理,输入特征，输出概率结果
infer = pgmpy.inference.DBNInference(model)
q = infer.forward_inference(variables=[("Rain", 2)],
                            evidence={
                                ('Location', 0): 1,
                                ('Location', 1): 1,
                                ('Location', 2): 1,
                                ('Rain', 0): 1,
                                ('Rain', 1): 1,
                                ('Umbrella', 0): 1,
                                ('Umbrella', 1): 1,
                                ('Umbrella', 2): 1}
                            )
print(q[("Rain", 2)].values)

```

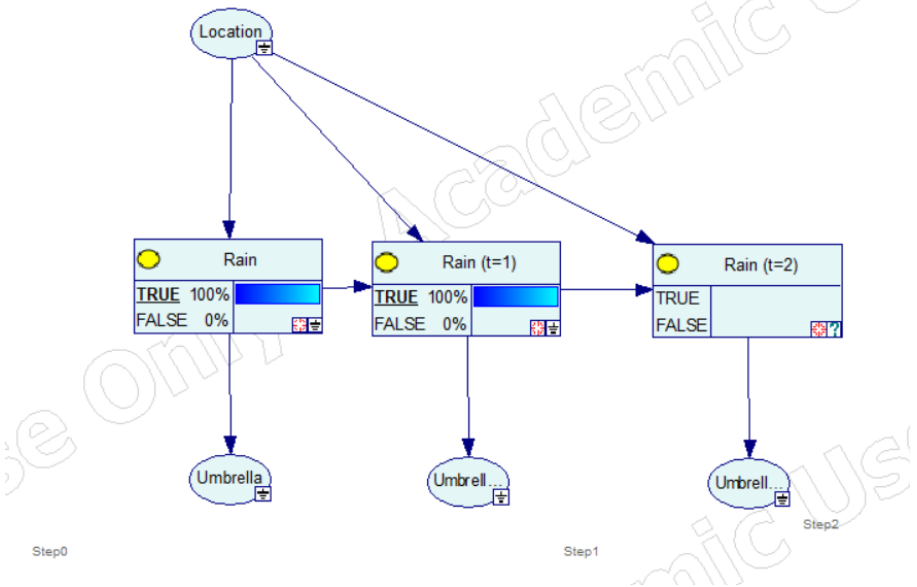
```
D:\Software\Anaconda3\envs\pgmpy\python.exe "D:\WorkSpace\PyCharm Project\231027 BayesPresentation\4 DBN Parameter Learnin
+---------------+--------------------+
| ('Rain', 2)   |   phi(('Rain', 2)) |
+===============+====================+
| ('Rain', 2)(0) |             0.1483 |
+---------------+--------------------+
| ('Rain', 2)(1) |             0.8517 |
+---------------+--------------------+

Process finished with exit code 0
```

2. 贝叶斯网络代码实践

④ 动态贝叶斯网络(GeNIe)

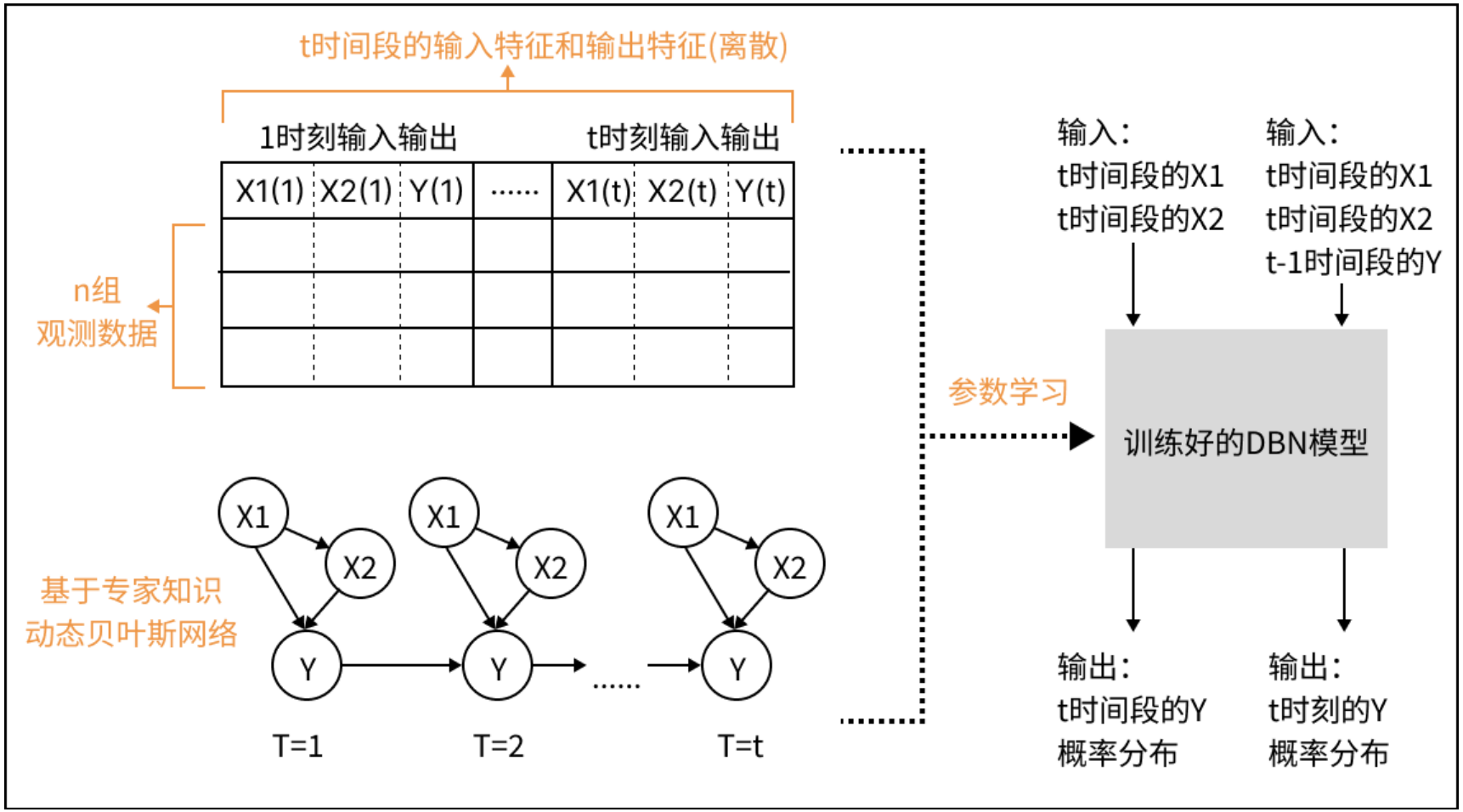


⇨ 网络展开



2. 贝叶斯网络代码实践

⑤ 总结



第三部分

文献分享

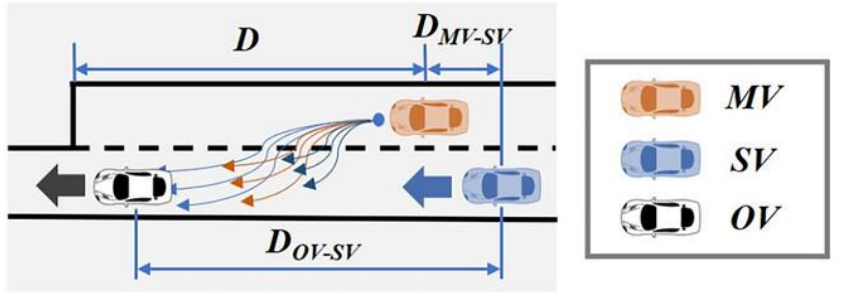
变道冲突时超车让路意图估计：一种基于语义的贝叶斯推理方法

1. 场景描述

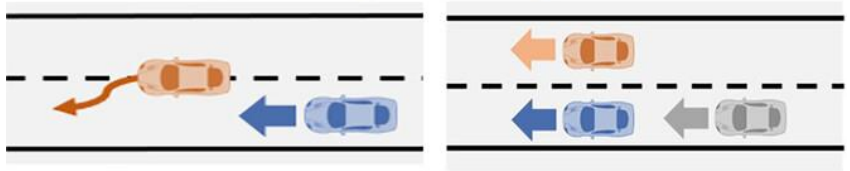
变道冲突发生在由3辆车组成的单元内：并道车辆(MV)、直行车辆(SV)和前方障碍车辆(OV)。

两类意图：

- 意图1：MV passes SV，即MV通过SV并并入SV和OV之间的间隙(图b)
- 意图2：MV yields to SV，即MV让SV先通过(图c)



(a) Modeling of lane-change conflict



(b) MV passes SV

(c) MV yields to SV

2. 场景相关特征

可观察状态：

- 三辆车的运动信息(位移、速度、加速度)，属于连续数据

潜在语义状态：考虑了驾驶意图的影响因素，模仿了驾驶员的决策过程，包含三个关键语义：：

- 条件S：MV是否在安全条件下并道，属于离散数据(True或False)
- 行为B：单帧车辆运动信息的语义解释，属于离散数据(行为状态1、2、3……)
- 意图I：上述描述的两种意图

变道冲突时超车让路意图估计：一种基于语义的贝叶斯推理方法

3. 定义贝叶斯网络节点

① 条件S： MV是否在安全条件下并道

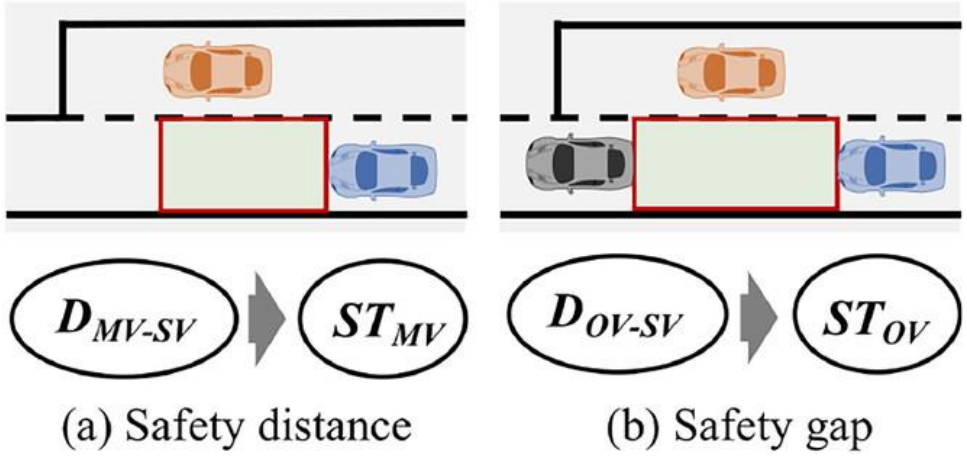
属性值为True：同时满足

安全距离(SMV)——MV和SV之间有足够的空间；

安全间隙(SOV)——SV和OV之间有足够的空间。

属性值为False：

上述有一条件不满足，则为False



(a) Safety distance (b) Safety gap

② 行为B：单帧车辆运动状态的语义。根据以下两个指标分成七种行为。这些行为通过可观测状态可以算出。

Longitudinal/Lateral 纵向运动/横向运动

Request/Achieve 请求他人先移动/自己要先移动

Vehicle	Direction	Request-Achieve	Pass-Yield
MV	Lateral (BY_{MV})	Request	Pass Yield
		Achieve	Pass
	Longitudinal (BX_{MV})	Request	Pass Yield
SV	longitudinal (BX_{SV})	Request	Pass Yield
		Achieve	Yield

③ 意图I：两种意图

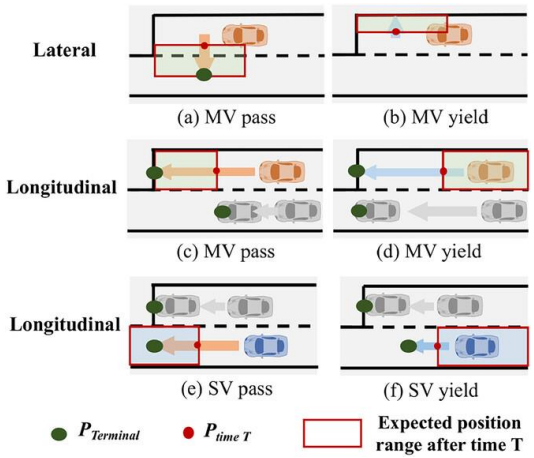
- 超车意图
- 让道意图

变道冲突时超车让路意图估计：一种基于语义的贝叶斯推理方法

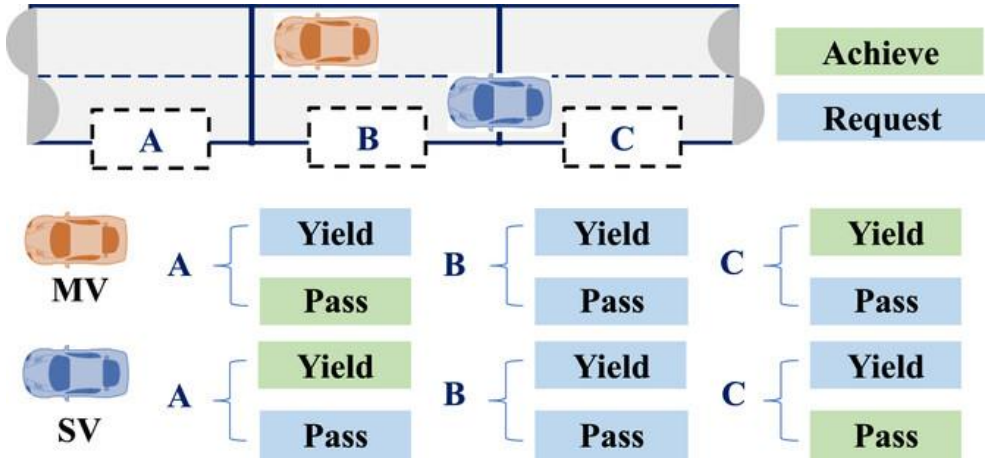
3. 定义贝叶斯网络节点

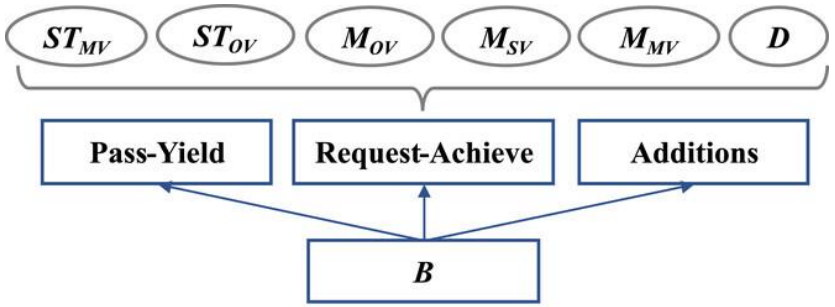
行为B两个指标的具体含义：

① Lateral/Longitudinal 横向运动/纵向运动



② Request/Achieve 请求他人先移动/自己要先移动



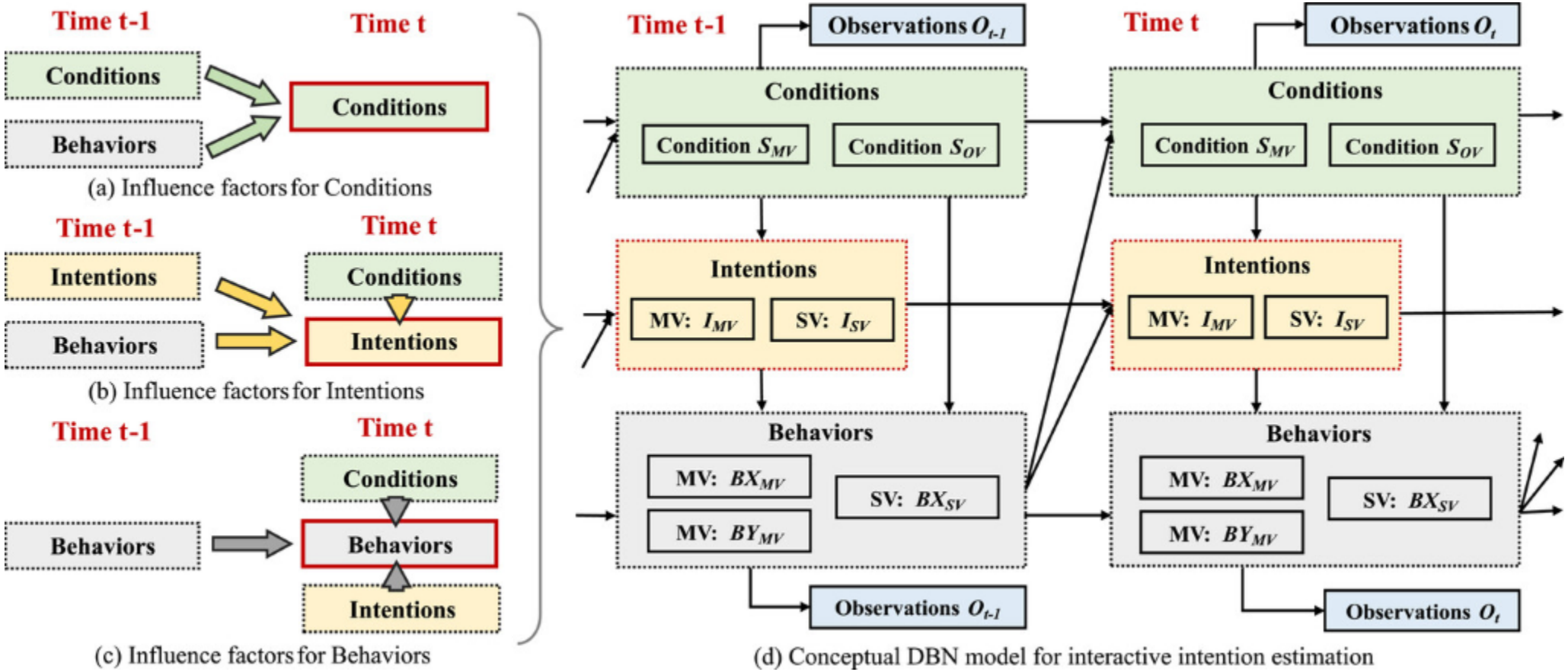


$$B \in \{ BM_1, BM_2, BM_3, BM_4, BM_5, BS_1, BS_2, BS_3 \}$$

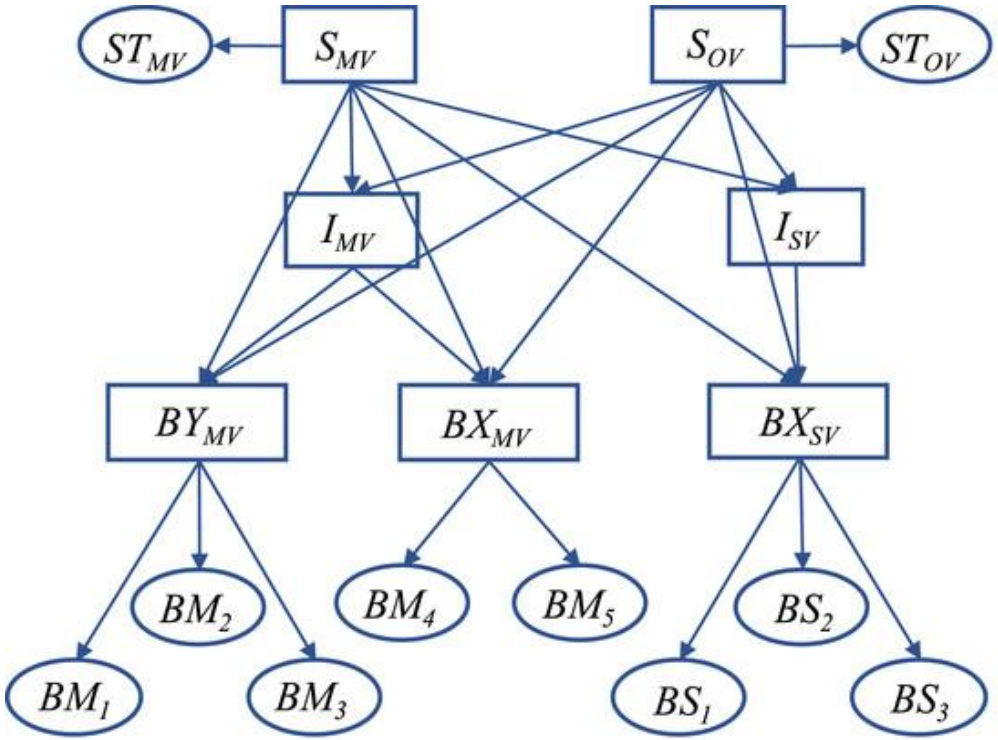
变道冲突时超车让路意图估计：一种基于语义的贝叶斯推理方法

4. 搭建动态贝叶斯网络

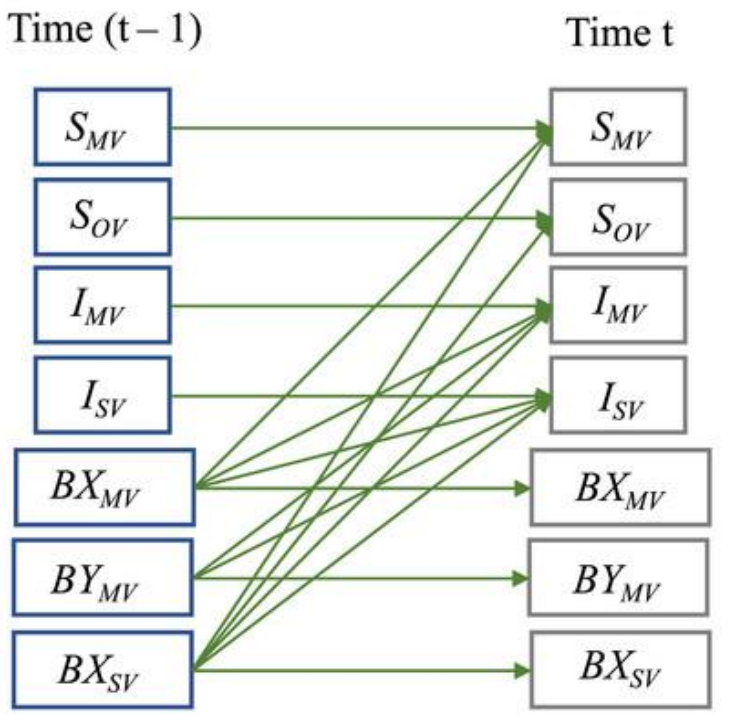
条件S、行为B、意图I三者在马尔可夫条件下的影响因素：



4. 搭建动态贝叶斯网络



(a) Probability relationships within a timeframe



(b) Transition model between two time frames

先验模型prior network

(单时间片的贝叶斯网络)

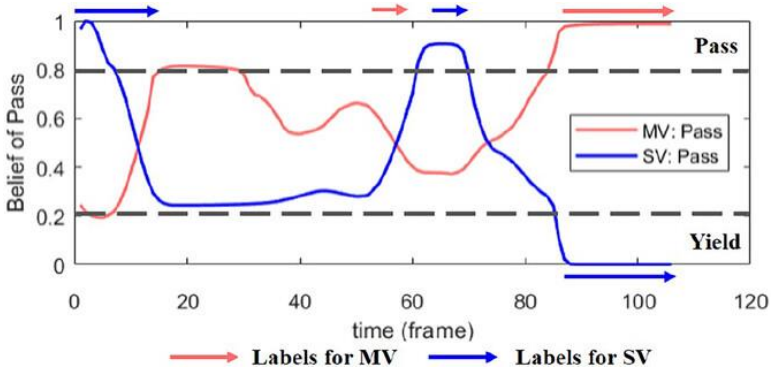
转移网络transition network

(连续两个时刻间结点的依赖关系)

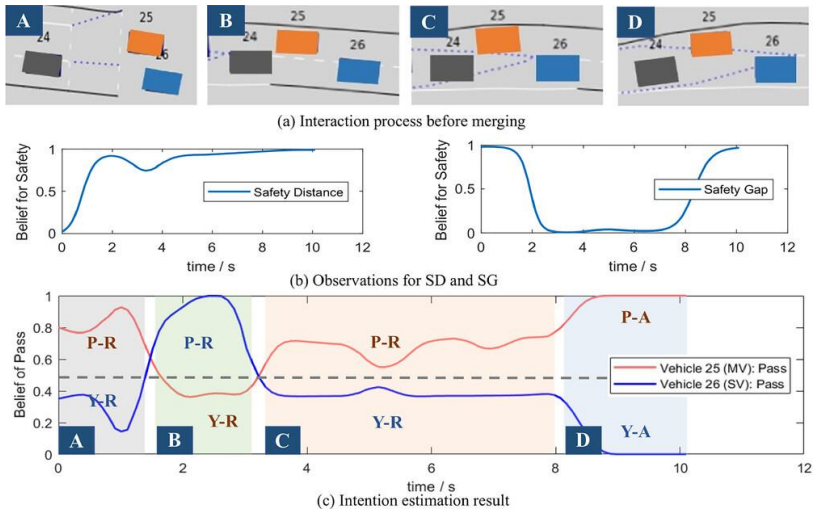
变道冲突时超车让路意图估计：一种基于语义的贝叶斯推理方法

5. 意图估计结果

① 意图二分结果



② 意图连续预测



③ 轨迹预测