贝叶斯网络

贝叶斯概率的一些基础思想

- **概率：**某人对一个命题信任的程度，这个概率不像频率概率范畴是描述某个随机事件发生的可能性的一个未知的定值，而是指一个未知的可以变化的值。
- **先验概率：**不知道其他信息情况下，根据以往经验和分析得到的概率
- **后验概率：**在得到随机事件的结果之后对先验概率进行修正之后的概率
- **全概率公式**：将复杂事件的概率求解转换为简单概率求解的方法
- **贝叶斯公式/定理：**贝叶斯概率与贝叶斯统计的基础，基本上是用来计算后验概率的公式

相关公式

- 变量相互独立：

$$\mathbf{P}(X|Y) = \mathbf{P}(X) \quad 或 \quad \mathbf{P}(Y|X) = \mathbf{P}(Y) \quad 或 \quad \mathbf{P}(X, Y) = \mathbf{P}(X)\mathbf{P}(Y)$$

- 条件概率：

$$\mathbf{P}(\mathrm{Y}|\mathrm{X}) = \frac{\mathbf{P}(\mathrm{X}, \mathrm{Y})}{\boldsymbol{P}(X)}$$

- 贝叶斯规则：

$$\mathbf{P}(Y \mid X) = \frac{\mathbf{P}(X \mid Y)\mathbf{P}(Y)}{\mathbf{P}(X)}$$

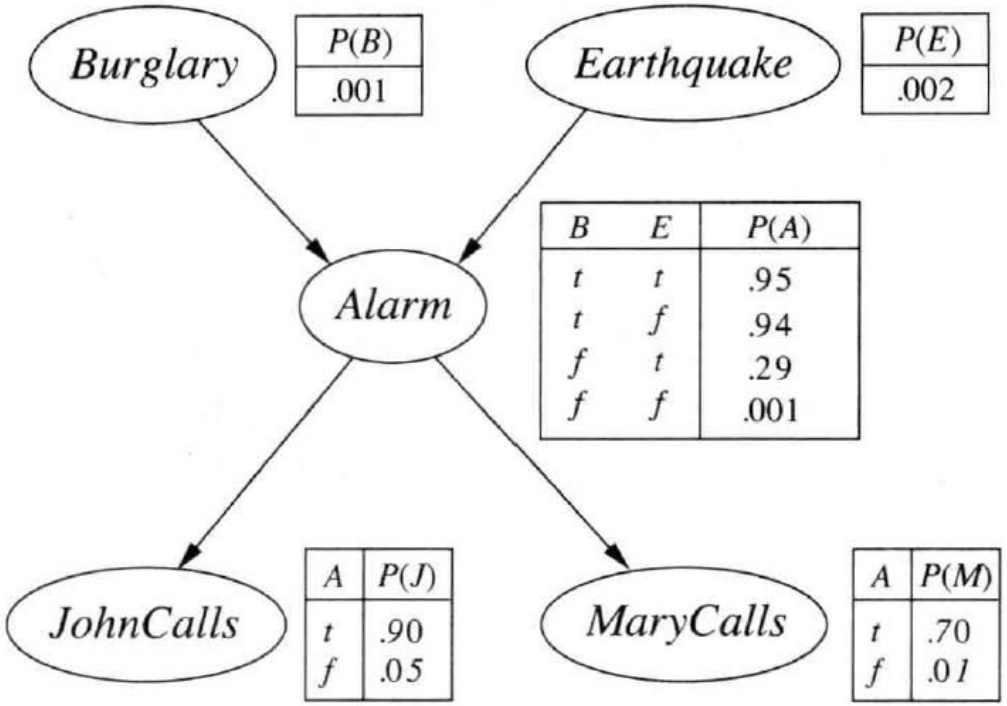
$$\mathbf{P}(\mathrm{X}, \mathrm{Y}) = \mathbf{P}(\mathrm{X}|\mathrm{Y})\mathbf{P}(\mathrm{Y}) = \mathbf{P}(\mathrm{Y}|\mathrm{X})\mathbf{P}(\mathrm{X})$$

- 链式法则：

$$\mathbf{P}(X_1, X_2, \dots, X_N) = \boldsymbol{P}(X_1|X_2, \dots, X_N) * \boldsymbol{P}(X_3|X_4, \dots, X_N) * \dots * \boldsymbol{P}(X_{N-1}|X_N) * \boldsymbol{P}(X_N)$$

贝叶斯网络

- 通过利用变量间的独立性和条件独立关系，把复杂的联合概率分布分解成一系列相对简单的模块，减少为定义完全联合概率分布所指定的概率数目，从而可以降低知识获取和概率推理的复杂度的一种模型。

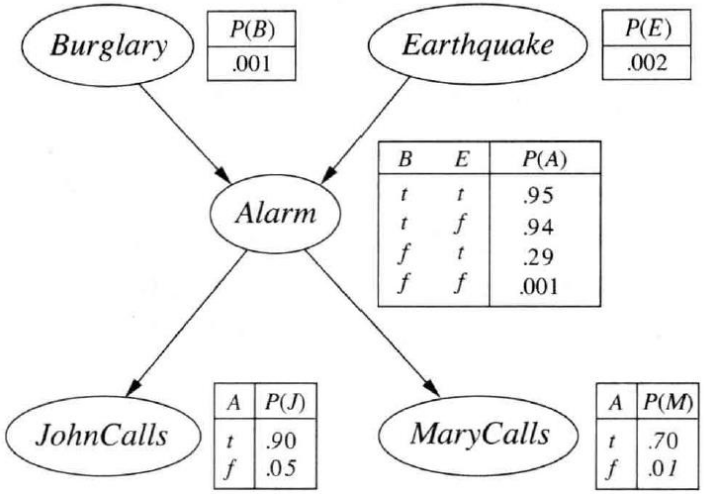


贝叶斯网络学习

- 变量定义：确定为了对问题域建模所需要的变量集合
- 结构学习：发现变量之间的图关系
- 参数学习：发现变量之间相互关联的量化关系

贝叶斯网络学习—变量定义

- 结点：对问题域建模所需要的变量集合
- 变量排序



{Burglary, Earthquake, Alarm, MarryCalls, JohnCalls}

贝叶斯网络学习—结构学习

- 选择具有最大后验概率的结构
- 基于评分搜索的方法
  - 评分函数： 主要是基于贝叶斯统计的评分函数和基于信息理论的评分函数，如K2评分，BD评分，MDL评分，BIC评分，AIC评分等等
  - 搜索策略： NP-Hard问题，通常采用启发式搜索算法
- 基于约束（依赖分析或条件独立性测试）的方法
  - 核心思想：首先对训练数据进行统计性测试，尤其是条件独立性测试，确定出变量之间的条件独立性；然后利用变量之间的条件独立性，构造一个有向无环图，以尽可能多地涵盖这些条件独立性。典型算法：SGS算法，PC算法，TPDA算法等。
- 基于评分搜索和基于约束相结合的方法
- 随机抽样方法

详细介绍可参阅博客： https://blog.csdn.net/jbb0523/article/details/78804386

举例 贝叶斯网络结构学习之K2算法

- 评分函数:

$$g(i, \pi_i) = \prod_{j=1}^{q_i} \frac{(r_i - 1)!}{(N_{ij} + r_i - 1)!} \prod_{k=1}^{r_i} N_{ijk}!$$

- 搜索策略:

```
1. procedure K2;
2. {Input: A set of n nodes, an ordering on the nodes, an upper bound u on the
3.         number of parents a node may have, and a database D containing m cases.}
4. {Output: For each node, a printout of the parents of the node.}
5. for i := 1 to n do
6.     π_i := ∅;
7.     P_old := g(i, π_i); {This function is computed using equation (12).}
8.     OKToProceed := true
9.     while OKToProceed and |π_i| < u do
10.        let z be the node in Pred(x_i) − π_i that maximizes g(i, π_i ∪ {z});
11.        P_new := g(i, π_i ∪ {z});
12.        if P_new > P_old then
13.            P_old := P_new;
14.            π_i := π_i ∪ {z}
15.        else OKToProceed := false;
16.    end {while};
17.    write('Node:', x_i, 'Parents of this node:', π_i)
18. end {for};
19. end {K2};
```

[Cooper and Herskovits, 1991] Cooper, G. and Herskovits, E. (January, 1991). A Bayesian method for the induction of probabilistic networks from data.Technical Report SMI-91-1, Section on Medical Informatics, Stanford University.

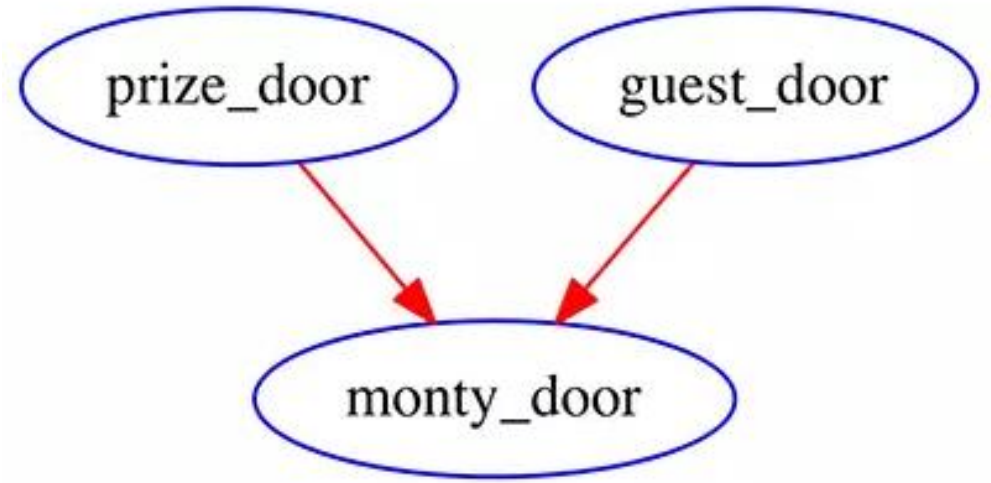
贝叶斯网络学习—参数学习

- 最大似然估计
  - 完全基于数据，不需要先验概率
- 贝叶斯估计
  - 有了最大似然估计的函数，加上先验概率，然后最大化后验概率
- 缺值数据最大似然估计(EM算法)

  (1)基于对数据进行修补，使之完整 (E-step)

  (2)基于修补后的完整数据计算的最大似然估计(M-Step)
- 隐结构模型学习

用贝叶斯网络解决“三门问题”



用贝叶斯网络解决“三门问题”

```
from pomegranate import *

# The guests initial door selection is completely random
guest = DiscreteDistribution( { 'A': 1./3, 'B': 1./3, 'C': 1./3 } )

# The door the prize is behind is also completely random
prize = DiscreteDistribution( { 'A': 1./3, 'B': 1./3, 'C': 1./3 } )

    # Monty is dependent on both the guest and the prize.
monty = ConditionalProbabilityTable(
        [[ 'A', 'A', 'A', 0.0 ],
         [ 'A', 'A', 'B', 0.5 ],
         [ 'A', 'A', 'C', 0.5 ],
         [ 'A', 'B', 'A', 0.0 ],
         [ 'A', 'B', 'B', 0.0 ],
                  ...
         [ 'C', 'C', 'A', 0.5 ],
         [ 'C', 'C', 'B', 0.5 ],
         [ 'C', 'C', 'C', 0.0 ]], [guest, prize] )
```

P(monty='B' | guest='A' && prize='A')

用贝叶斯网络解决“三门问题”

```
# State objects hold both the distribution, and a high level name.
s1 = State( guest, name="guest" )
s2 = State( prize, name="prize" )
s3 = State( monty, name="monty" )


# Create the Bayesian network object with a useful name
model = BayesianNetwork( "Monty Hall Problem" )

# Add the three states to the network
model.add_states(s1, s2, s3)


# Add transitions which represent conditional dependencies
model.add_transition(s1, s3)
model.add_transition(s2, s3)
model.bake()
```

用贝叶斯网络解决“三门问题”

```
print model.probability(['A', 'B', 'C'])
```

Out: 0.11111111111111109

```
print model.probability(['A', 'B', 'B'])
```

Out: 0.0

用贝叶斯网络解决“三门问题”

- 使用predict_proba函数运行推理

```
model.predict_proba({})

model.predict_proba({'guest': 'A'})

model.predict_proba({'guest': 'A', 'monty': 'C'})
```

思考题

- 对于我刚刚讲到的K2算法，有什么可以改进的思路？

实验要求

- 构建并使用贝叶斯网络计算Task中的概率
  - 可以使用python中的包以及matlab中的FullBNT
  - 手动实现贝叶斯网络可以加分
- 提交文件
  - 概率计算结果：16******_wangxiaoming.txt。每一行对应的是Task样例的计算结果。
  - 实验报告：16******_wangxiaoming.pdf。
  - 代码：16******_wangxiaoming.zip。如果代码分成多个文件，最好写份readme。
- DDL: 2018-12-11 23:00:00

附录

- **贝叶斯网络结构学习若干问题解释**
  https://blog.csdn.net/jbb0523/article/details/78804753