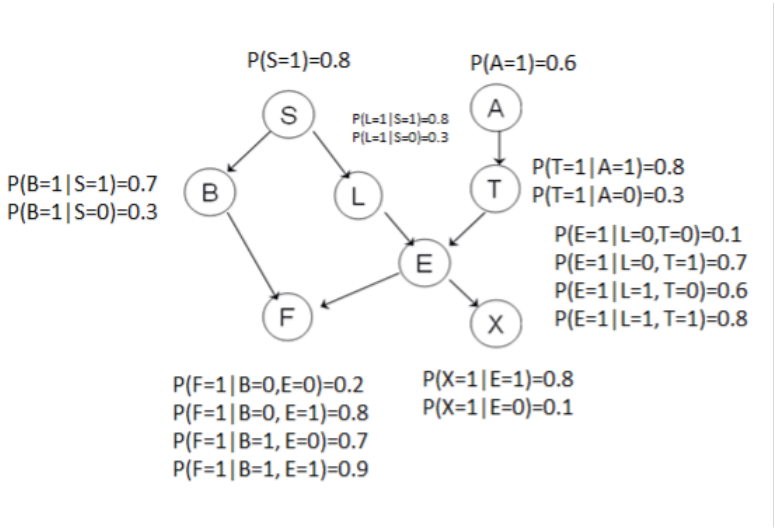
ECSE 6810 Project-1 Report

Shiuli Subhra Ghosh

Given The BN below,



In this project I have implemented sum-product inference algorithm using Python and computed $P(X_q|L=1, X=0)$ where $X_q \in \{S, A, B, T, E, F\}$.

1 Bayesian Network Inference

Bayesian networks are a type of probabilistic graphical model comprised of nodes and directed edges. Bayesian network models are able to capture conditional independencies among the random variables. Using Bayesian network we can estimate the probability of a random variable which can occur while other dependent or independent variables are acting as an evidence.

1.1 Types of Bayesian Inference

1. Posterior probability inference (Sum Product)
2. Maximum a Posteriori (MAP) inference
3. Most probable Explanation (MPE) inference
4. Model Likelihood Inference.

1.2 Mathematical Modelling

Posterior Probability Inference Let a Bayesian Network is G over some random variables $X = \{X_U, X_E\}$, where X_U is the set of non-evidence nodes and X_E are the set of evidence nodes. We want to find $P(X_Q|X_E)$ using Sum product inference where, $X_Q \subseteq X_U$.

$$P(X_Q|X_E) = \sum_{X_U \setminus X_Q} P(X_U|X_E) \propto \sum_{X_U \setminus X_Q} P(X_U, X_E)$$

Maximum Probable Explanation Inference (MPE) et a Bayesian Network is G over some random variables $X = \{X_U, X_E\}$, where X_U is the set of non-evidence nodes and X_E are the set of evidence nodes.

$$X_U^* = \underset{X_U}{\operatorname{argmin}}\, P(X_U|X_E)$$

2 Belief Propagation Algorithm

In Bayesian Inference, there are a few exact and non-exact methods.

2.1 Exact Methods

1. Variable Elimination
2. Belief Propagation
3. Clustering and Junction Tree

2.2 Approximate Methods

1. Loopy Belief Propagation
2. Sampling Methods
3. Variational Method

In this Assignment we will focus on Belief Propagation which is one of the most important Exact Inference Algorithms.

Variable Elimination is a very basic exact inference method but it is computationally expensive. Though choosing a right elimination order can reduce inference complexity, yet finding out optimal elimination order is NP-hard. Hence Belief Propagation algorithm can be implemented with exact inference capability and reduced complexity. According to Pearl the main purpose of Belief Propagation is "fusing and propagating the impact of new evidence and beliefs through Bayesian networks so that each proposition eventually will be assigned a certainty measure consistent with the axioms of probability theory." (Pearl, 1988, p 143)

Algorithm 1 Pearl's Belief Propagation Algorithm (Sum-Product)

Require: Evidence Nodes X_E , Non-Evidence Nodes X_Q
  Initialize the messages for all nodes
  while Until Convergence **do**
    for $X_i \in X_E$ **do**
      V_i = parents of X_i
      C_i = children of X_i
      for $p \in V_i$ **do**
        $\pi_p(X_i) = \pi(p) \prod_{c \in ch(p) \setminus X_i} \lambda_c(X_i)$ ▷ S-1: Computing messages from parents
      end for
      $\pi(X_i) = \sum_{V_i} P(X_i|V_i) \prod_{v_i \in V_i} \pi_{v_i}(X_i)$ ▷ S-2: Computing total messages from parents
      for $ch \in C_i$ **do**
        $\lambda_{ch}(X_i) = \sum_{ch} \lambda_{ch} \sum_{u_k \in \pi(ch) \setminus X_i} P(ch|X_i, \{u_k\}) \prod_{k=1}^{q} \pi_{u_k}(ch)$ ▷ S-3: messages received from children
      end for
      $\lambda(X_i) = \prod_{c_i \in C_i} \lambda_{c_i}(X_i)$ ▷ S-4: Computing total messages from all children
      $P(X_i|X_E) = \alpha \pi(X_i) \lambda(X_i)$ ▷ S-5: Compute Belief and normalize
    end for
    Check for convergence
    if the Belief doesn't change from the previous iteration **then**
      Break
    else
      Continue with the next iteration
    end if
  end while

Algorithm 2 Belief Propagation Algorithm (Max-Product)

Require: Evidence Nodes X_E , Non-Evidence Nodes X_Q
  Initialize the messages for all nodes
  while Until Convergence **do**
    for $X_i \in X_E$ **do**
      V_i = parents of X_i
      C_i = children of X_i
      for $p \in V_i$ **do**
        $\pi_p(X_i) = \pi(p) \prod_{c \in ch(p) \setminus X_i} \lambda_c(X_i)$ ▷ Computing messages from parents
      end for
      $\pi(X_i) = \max_{V_i} P(X_i|V_i) \prod_{v_i \in V_i} \pi_{v_i}(X_i)$ ▷ Computing total messages from parents
      for $ch \in C_i$ **do**
        $\lambda_{ch}(X_i) = \max_{ch} \lambda_{ch} \max_{u_k \in \pi(ch) \setminus X_i} P(ch|X_i, \{u_k\}) \prod_{k=1}^{q} \pi_{u_k}(ch)$ ▷ messages received from children
      end for
      $\lambda(X_i) = \prod_{c_i \in C_i} \lambda_{c_i}(X_i)$ ▷ Computing total messages from all children
      $P(X_i|X_E) = \alpha \pi(X_i) \lambda(X_i)$ ▷ Compute Belief and normalize
    end for
    Check for convergence
    if the Belief doesn't change from the previous iteration **then**
      Break
    else
      Continue with the next iteration
    end if
  end while

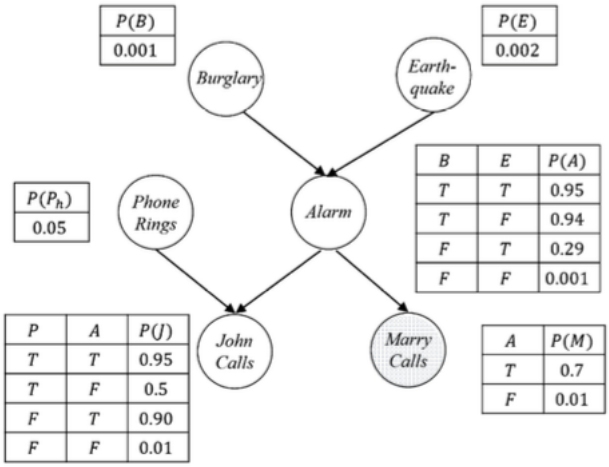
2.3 Initialization and boundary conditions

- Initialization Step
  - For all $X_i \in X_E$: $\lambda(X_i) = 1$ for $X_i = e$, 0 otherwise
  - For all $X_i \in X_E$: $\pi(X_i) = 1$ for $X_i = e$, 0 otherwise
- Boundary Conditions
  - For nodes without parents $\pi_{X_i} = P(X_i)$ (Prior probabilities)
  - For nodes without parents $\lambda_{X_i} = 1$
  - For nodes without children $\lambda(X_i) = 1$
- Intermediate Nodes
  - $\pi(X_i) = 1$ and $\lambda(X_i) = 1$

Each node can update it's total π and λ messages independently, based on the current λ or π messages of its neighbours. This means belief propagation can be implemented in parallel and asynchronously. The order of the nodes doesn't matter here like Variable Elimination.

3 Burglary Example

Using this Algorithm, I first implemented the example from class.



Evidence: Marry call ($M = 1$)

This example converges after 2 iteration and the results are as follows,

```
Iterations terminated in 2th step
################################################
P(A = 1) | L = 1, X = 0 = 0.8499098822502404
P(B = 1) | L = 1, X = 0 = 0.943882545961085
P(E = 1) | L = 1, X = 0 = 0.9641190847135442
P(J = 1) | L = 1, X = 0 = 0.8352217777932087
P(P = 1) | L = 1, X = 0 = 0.9500000000000001
################################################
```

4 Results

The assignment problem also converged in the step 2 and the results are as follows,

```
Iterations terminated in 2th step
##########################################
P(B = 1) | L = 1, X = 0 = 0.6657142857142858
P(E = 1) | L = 1, X = 0 = 0.3636363636363636
P(A = 1) | L = 1, X = 0 = 0.5575757575757576
P(F = 1) | L = 1, X = 0 = 0.6542077922077921
P(S = 1) | L = 1, X = 0 = 0.9142857142857143
P(T = 1) | L = 1, X = 0 = 0.515151515151515
##########################################
```

The code in the zip file has 2 scripts. cal.py computes the messages and update the beliefs in the back end. main.py script is the driver script, where the graph is created by calling the Node class from cal.py file. Then for multiple iterations and for each non-evidence nodes the belief is calculated and updated by passing messages from parents and children. The main.py script prints the inference result for B,E,A,F,S,T nodes given L=1 and T=0.