

# Probabilistic Models Project Proposal

Final Project

Computer Science  
IDC  
March 1, 2017

## Intro

Probabilistic Graphical Models (PGM) use a graph representation to describe complex distribution over high dimensional space. Every node in the graph correspond a random variable. Edges in the graph correspond to direct probabilistic interactions between variables. The representation a set of in-dependencies holds in the distribution. In high dimensional distribution the graph representation encode the probability in small factors rather than over every possible assignment of the variables, the joined distribution defined as the a product of all these factors. In this kind of representation we can define Bayesian network and Markov network distributions. The Bayesian network is represented as a directed graph and Markov as undirected graph.

This representation allow humans to better evaluate properties and semantics of a variables distribution, it can help understand unexplained or undesirable answers. In addition using the graph it is possible to run efficient algorithm to posterior probability of variables given the evidence of others. Another characteristic is learning from data model provides a good approximation of past experience.

The main objective of this project is to develop a user-friendly software tool for constructing PGMs and experimenting with different aspects and inference algorithms. This tool will be designed both for educational use in academic courses and for self exploration by scientists and developers in the industry. This tool illustrates different aspects of probabilistic models in 6 different units:

1. Unit 1 - conditional independence in Bayesian networks
2. Unit 2 - undirected representations of Bayesian networks
3. Unit 3 - elimination orders
4. Unit 4 - inference algorithms
5. Unit 5 - parameter inference
6. Unit 6 - sampling algorithms on general Probabilistic Graphical Models

A more comprehensive explanation on every unit can be found in proposed design section. The tool designed to experiment basic understanding from dependencies of random variables, until running different algorithms on Bayesian networks.

In our research we came across a couple of libraries implements PGMs. Although those libraries have some aspects in common with our tool, we believe our tool helps to experiment in PGMs more gradually. Here is the list of tools:

1. Kevin Murphy - Matlab implementation <http://www.cs.ubc.ca/~murphyk/Software/>
2. Kevin Murphy - Hidden Markov Model Matlab implementation <http://www.cs.ubc.ca/~murphyk/Software/HMM/hmm/>
3. OpenGM - A C++ Implementation <http://hciweb2.iwr.uni-heidelberg.de/opengm/>
4. Probabilistic Graphical Models hosted on Github <https://github.com/anhncs/Probabilistic-Graphical-Models>

## Overview

Create a tool for students to both learn and experiment in probabilistic models. The idea is to illustrate different aspects of probabilistic models, all is defined by the units presented in the scope section. The tool is design to help from the basic understanding of dependencies for random variables, to running different algorithms on Bayesian networks. This file is going to explain the scope and way to create a tool for students to run various probabilistic models described in this document. The tools is going to be assembled from the following components:

1. Visual representation of model
2. JSON representation for models
3. CLI command tool to run operation against a model and to see reports

### Network Definition

In the visualization part the user will have the ability to:

1. Define Bayesian network in a couple of steps:
  - (a) Define all the different nodes. Where every node have a name and domain.
  - (b) Define the dependencies between the nodes.

At the end of this process the user is able to see a graph representing the network.

2. Run different algorithms on a Bayesian network such as elimination, message passing. And in addition run transformations such as to moralized graph

**Define Conditional Table** The user is able to add conditional table for the nodes in an easy way. The user will be able to choose a node in the network defined in network definition table, and the node will be highlighted and an empty conditional distribution table will be presented, where the user is able to set the probability for every value of the node given his parents  $P(X_v|X_{\pi_v})$

## Proposed Design

As stated in the introduction the tool is build from 6 units, in this section we are going to elaborate on each one. Every unit extends the capabilities of the former unit.

### 1. Unit 1

This unit is the designed to learn the basics of probability modeling of data. After understanding the basics in probability, visualize dependency in discrete random variables using a Bayesian network.

- (a) Use predefined Bayesian networks to illustrate dependence vs independence in random variables
- (b) Allow the user to define a Bayesian network, and define dependency between variables. The graph representation is giving the user the ability to better understand the network.
- (c) show conditional independence using D separation rules. The rules are:
  - i. Path contains a node  $w \in A$  and not both adages that touch w are incoming:
    - A.  $X_u \rightarrow x \rightarrow X_v$
    - B.  $X_u \leftarrow x \leftarrow X_v$
    - C.  $X_u \leftarrow x \rightarrow X_v$
  - ii. path contains a node  $w \notin A$  with two incoming edges  $X_u \rightarrow w \leftarrow X_v$

The D separation is a general criterion for  $X_v \perp\!\!\!\perp X_u | X_A$ . Where  $X_A$  is a set of RVs need to condition on in order that v will be independent from u. The user is going to be able to choose two nodes, the tool is going to show id those nodes are depended or not using D separation. If they are conditional independent show the user all the paths that does not block dependency, otherwise show the path that block dependency.

## 2. Unit 2

- (a) Covert graph representation of a model to moralized graph. Create an undirected version of the Bayesian network, where two RVs have an edge if  $u \in n(X_v)$ . Emphasize that nodes associated with a potential function form a clique in the graph.
- (b) Validate if graph is chordal or not
- (c) Convert moralized graph to factor graph

## 3. Unit 3

- (a) Show elimination algorithm implementation on graphs.
  - i. Show elimination in moralized graph. The user can choose an elimination order, the moralized graph will reflect how the elimination of a node creates new edges for the node neighbors (if needed).
- (b) Find perfect elimination for tree. The perfect elimination can be found using the moralized graph from unit 2, validate if the graph is chordal or not.
- (c) Show message passing algorithm in graph representation. Display the marginal distribution tables of vertexes along the algorithm.

## 4. Unit 4

- (a) Parameter inference through data in tree models

## 5. Unit 5

- (a) Markov chain Monte Carlo algorithm

### Model Representation

A model is represented using:

- 1. Random variables. Where every random variable can have:
  - (a) Name (X)
  - (b) Domain  $\Omega$  where every  $x$  in  $\Omega$  have the probability  $P(X = x)$

Example for JSON file, representation of random variable variable:

- name: x1
  - domain:
    - 0
    - 1
- name: x2
  - domain:
    - 0
    - 1
    - 2

### Visual Representation

- 1. Represent Bayesian network using DAG. Represent all the nodes, with the direct edges from parents to node  $P(X_v|X_{\pi_v})$
- 2. Represent the elimination algorithm for a Bayesian network.  $f(x_{hidden}) = P(x_{hidden}, X_{observed}) = \sum_{\forall x \in V} P(X_v|X_{\pi_v})$ . This is designed to show the elimination algorithm step by step.  
Setup:

- (a) Set all the nodes data and with the observed nodes value.
- (b) set the elimination order

Initialization:

- (a) Set potential function for all the nodes  $\Psi_{v \in V}$ , where  $\psi_v = P(X_v|X_{\pi_v})$
- (b) create a matrix  $|V \setminus observed|X|\Psi|$  of all RVs associate with every potential function.

- (c) Set the list of elimination order of RVs in  $X_{V \setminus (observed \cup inferred)}$

Elimination Loop:

- (a)  $X_v$  is the next RV in the elimination order so  $\Psi_v$  is the potential functions have RV in the matrix
- (b)  $n(X_v)$  are the RVs that involved in one of the potential function in  $\Psi_v$
- (c) set  $m_v$  of all RVs in  $n(X_v)$ . So  $m_v(n(X_v)) = \sum_{X_v} \Psi_v$
- (d) replace  $\Psi_v$  with  $m_v$

By setting up elimination order the user is able see the impact on the complexity.

Output of the run - the elimination step by step with potential matrix.

3. Convert Bayesian network to moralized graph. In this scenario the objective is to show how the operation is done step by step.
4. Message passing algorithm. Visual representation of the message passing. Step by step running of eliminate procedure
5. Factor Tree and message passing algorithm. Represent the conversion between a direct graph to the moralized version and then to factor graph

## CLI tool

1. Running elimination algorithm for given node setup with elimination order.

Command: eliminate "path to file" "elimination order for all the nodes"

Input:

- (a) JSON file path, representing the RVs, nodes and conditional tables for all nodes  $P(X_v = x_v | X_{\pi_v} = x_{\pi_v})$ .
- (b) elimination order of the nodes

Output:

Joint probability  $f(\overline{x_{inferred}}) = P(X_{inferred}, X_{observed})$

2. Message passing algorithm in trees. Run only eliminate procedure. Command: message passing "sync/async" "path to file" "elimination order for all the nodes"

Give the ability to run in two modes:

- (a) synchronous - eliminate one child of a node at a time
- (b) asynchronous - eliminate child nodes in parallel

Message passing algorithm with compute marginal for all nodes. Run the procedures collect and distribute

3. Message passing with most probable joint assignment. The input is the same as the other message passing CLI arguments. The output in this case is maximum probability and the assignment.
4. Run message algorithm on poly tree. Given as input:

Command: message passing "sync/async" "path to file" "elimination order for all the nodes"

- (a) poly tree with nodes V and edges E
- (b) conditional tables for all nodes  $P(X_v = x_v | X_{\pi_v} = x_{\pi_v})$

Output:

Marginal distribution for all the nodes

## Design