ReadMe

This is the document about the implemented algorithm Belief Propagation, including introduction, structure design and future works. In future works, there is a potential application idea, which might be helpful for those who want to continue the work.

Introduction

The codes implemented the Belief Propagation algorithm, based on the Bayesian Network Model.

In the test codes (main.swift), there are four nodes (shown in the **Fig1** below). The inputs can be found in the test codes: the prior probabilities of R and S, as well the boundary conditions P(W|R) and P(H|R,S), the output will be the Beliefs of W and/or H (shown in the **Fig2** below). Each node has two states, true or false.

To run the code, just run from **main.swift**, the input can be changed with different probabilities. The output (you might want to see other nodes' belief) can also be changed, please find the comments in the codes.

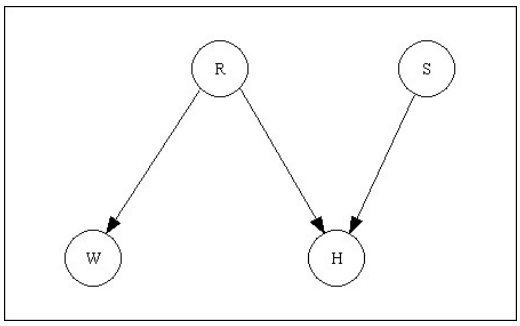

Fig 1. The testing graph

```
Result for node S: (0.338235294117647, 0.661764705882353)FALSE
Result for node R: (0.735294117647059, 0.264705882352941)TRUE
```

Fig 2. Output

Structure

There are mainly 7 swift files in the project, among which 2 of them were written for future extension.

- **main** : the main function, contains inputs. **Please run this directly**.

- BayesianNetwork: the data structure of the Bayesian network, including three classes: *Edge, Vertex and SwiftGraph*.
    - In *SwiftGraph* class, the part of undirected graph is to be extended, which are commented.
- BeliefTable: The essential part of BP algorithm, including *BeliefTable* class and *BeliefRow* class. In *BeliefTable*, some equators were implemented. For some functions inside, the comments gave an introduction of the rules, along with an example.
- ConditionalPropability: The data structure for prior probability table. Examples also can be found in the comments.
- StatTools: Including math functions, like dot product.
- Node and StateKeys: To be extended: define multiple states.

Future Works

1. **Multiple States** for each node: depends on the application of future use. The implemented algorithm only supports a binary condition. For example, if it will be applied on the weather forecast, a node will have more than two states: cloudy, windy and rainy. The future work will be on the node state.
2. **Terminate Conditions of Loops**. The core part of the algorithm is finished; further works will be on top of the implemented functions --- make it be able to deal with loops, with carefully designed terminate conditions (until there is no or slight changes on all belief values of each node).
3. **Other Graphical Models.** According to the paper written by *Jonathan S.Yedidia*[1], there are three basic graphical models: factor graph, MRF (Pairwise Markov Random Fields) and Bayesian Network. Currently the algorithm is on top of the Bayesian one. But the paper mentioned that the three models could be transferred to each other, which will be another direction of further research.
4. **Accuracy Estimation.** Inference or prediction algorithms are designed to estimate something that might happen in the future, which makes the accuracy as an important criterion. One way is that for different scenarios, L-BP can be compared to different algorithms on machine learning or data mining, for example: when making predictions on a time series data, it can be compared with *Linear Regression* or *HMM (Hidden Markov Model).*

5. **A potential Application on iWatch.**
   Human interaction is becoming more and more important, and as for sensors on the iWatch, it is perfect if we combine the algorithm with the iWatch.
   This idea is focusing on time series data. The inference model will be used to recognize patterns for some certain movements. An application is that a supervising app used on the gym (to see weather the movements are standard). Another application can be an educational app: if someone wants to learn Yoga or dance, when they are focusing on the arm, some

pre-defined patterns can be stored in advance, then the app can supervise if the movement is correct.
A product called *Valedo* [2] is an example, although can not find the technology on programming level, it is still an interesting idea.

Reference

[1] *Understanding Belief Propagation and its Generalizations,* Jonathan S.Yedidia, William T. Freeman, and Yair Weiss, TR2001-22 November 2001
[2] https://www.youtube.com/watch?v=hEztotzKaWw.

Readings

1. *Loopy Belief Propagation for Approximate Inference*: *An Empirical Study*, Kevin P. Murphy and Yair Weiss and Michael I. Jordan
2. *Variational Inference: Loopy Belief Propagation*, Lecturer: Eric P. Xing Scribes: Rajarshi Das, Zhengzhong Liu, Dishan Gupta, 10-708: Probabilistic Graphical Models 10-708, Spring 2014
3. Introduction to Loopy Belief Propagation, http://cseweb.ucsd.edu/classes/sp06/cse151/lectures/belief-propagation.pdf

Other Information

Written by: Irene Li
E-mail : irenelizihui@gmail.com

Detailed files that related to the algorithm or application, please find past ppts on the dropbox.
For those who are willing to working on this part, if have any advice or questions, please feel free to contact me!