Crafting a Lightweight Bayesian Inference Engine

Feng-Jen Yang, *Member, IAENG*

***Abstract*—Many expert systems are counting on Bayesian inference to perform probabilistic inferences and provide quantitative supports for decisions, hypotheses, forecasts and diagnoses. With this concern in mind, I implemented this probability-based inference engine that is domain independent and can be plugged into future systems to speed up the development life cycle of machine learning and/or inference systems.**

***Index Terms*—Bayesian inference, Bayesian theory, Inference engine.**

I. Introduction

BAYESIAN inference is rooted from the well-known posthumous theory of Thomas Bayes that was formulated in the 18th century and soon adopted as the mathematical rationale for the processing of uncertain information and the drawing of probabilistic conclusions based on the evidences that have been observed [2]. A counter intuitive calculation that makes Bayesian inference hard to comprehend is that it turns around the evidence and the hypothesis within a conditional probability in a way that most of the beginners are not feeling quite comfortable with. Nonetheless, this unnatural formula derivation is the essential idea that transforms the mathematical notations to meet the probabilistic inference purposes in most of the stochastic-based expert systems. Mathematically this concept can be demonstrated as computing $P(H|E)$ in terms of $P(E|H)$, where H is denoting a hypothesis and E is denoting an evidence to support this hypothesis.

As stated in most of the probability and statistics books, the algebra involved in this calculation of a conditional probability is illustrated as:

$$P(H|E) = \frac{P(H \cap E)}{P(E)}$$

From the standing point of inference, the real semantic behind this calculation is representing the inference of:

If the evidence E is observed, how much likely is the hypothesis H true?

While designing a system, if a domain expert can provide the values of $P(H \cap E)$ and $P(E)$ then the inference of $P(H|E)$ is just a simple division. However, $P(H \cap E)$ and $P(E)$ are not friendly cognitive values that human experts can easily construct mentally. As a result, most of the knowledge engineers are trying an indirect knowledge extraction to compute $P(H|E)$ as follows:

$$P(H|E) = \frac{P(H \cap E)}{P(E)}$$
$$= \frac{P(E|H) \times P(H)}{P(E|H) \times P(H) + P(E|\bar{H}) \times P(\bar{H})}$$

Where $P(H)$ is the prior probability of H being true, $P(\bar{H})$ is the prior probability of H being false, $P(E|H)$ is the conditional probability of see E when H is true, $P(E|\bar{H})$ is the conditional probability of see E when H is false, and $P(H|E)$ is the posterior probability of H being true by seeing E.

Manuscript received March 2, 2016; revised March 16, 2016. This research is funded by the Seed Grant of Florida Polytechnic University to Dr. Feng-Jen Yang. The implementation detail of this project does not reflect the policy of Florida Polytechnic University and no official endorsement should be inferred.

F. Yang is with the Department of Computer Science and Information Technology, Florida Polytechnic University, FL 33805, USA, e-mail: fyang@flpoly.org.

II. The Probabilistic Reasoning

A more generalized Bayesian inference can be represented as a series of mathematical derivations. Considering a given problem domain with n evidences and m hypotheses, the inference of:

If evidences E_1, E_2, ... and E_n are observed, how much likely is the hypothesis Hi true, where $1 \leq i \leq m$?

The detail inference steps can be performed by the following computations [1]:

$$P(H_i|(E_1 \cap E_2 \cap ... \cap E_n))$$
$$= \frac{P(H_i \cap (E_1 \cap E_2 \cap ... \cap E_n))}{P(E_1 \cap E_2 \cap ... \cap E_n)}$$
$$= \frac{P((E_1 \cap E_2 \cap ... \cap E_n) \cap H_i)}{\sum_{j=1}^{m} P((E_1 \cap E_2 \cap ... \cap E_n) \cap H_j)}$$
$$= \frac{P((E_1 \cap E_2 \cap ... \cap E_n)|H_i) \times P(H_i)}{\sum_{j=1}^{m} P((E_1 \cap E_2 \cap ... \cap E_n)|H_j) \times P(H_j)}$$

Where $1 \leq i \leq m$.

The above series of computations are mathematically sound, but the involvement of joint conditional probabilities that have to deal with all possible combinations of evidences and hypotheses is way too complicate for any human experts to conclude from their real life experiences. To make this mathematical rationale more practical, the conditional independence among these evidences is usually assumed so that:

$$P((E_1 \cap E_2 \cap ... \cap E_n)|H_i)$$
$$= P(E_1|H_i) \times P(E_2|H_i) \times ... \times P(E_n|H_i) \times P(H_i)$$

So the mathematical result can be further simplified into:

$$P(H_i|(E_1 \cap E_2 \cap ... \cap E_n))$$
$$= \frac{P(E_1|H_i) \times P(E_2|H_i) \times ... \times P(E_n|H_i) \times P(H_i)}{\sum_{j=1}^{m} P(E_1|H_j) \times P(E_2|H_j) \times ... \times P(E_n|H_j) \times P(H_j)}$$

Where $1 \leq i \leq m$. This assumption significantly simplifies the complexity of the domain experts mental burden and makes the inference processing cognitively workable. Instead of considering all evidences at once, the experts are now considering an evidence and a hypothesis at a time [3].

ISBN: 978-988-14048-0-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I
THE DESIGN OF THE PRIOR PROBABILITY CLASS TYPE

Class Name: PrioProb	
Purpose: Representing a prior probability	
Method	**What It Does**
Constructor	Initializing the prior probability
ToString	Converting the prior probability into a string representation
Attribure	**What It Means**
hypo	The hypothesis in the prior probability
prob	The likelihood value of the prior probability

TABLE II
THE DESIGN OF THE CONDITIONAL PROBABILITY CLASS TYPE

Class Name: CondProb	
Purpose: Representing a conditional probability	
Method	**What It Does**
Constructor	Initializing the conditional probability
ToString	Converting the conditional probability into a string representation
Attribure	**What It Means**
hypo	The hypothesis in the conditional probability
evid	The evidence in the conditional probability
prob	The likelihood value of the conditional probability

III. THE DESIGN OF CLASS TYPES

Although this probabilistic inference process looks specific, it involves intensive calculations among prior probabilities, conditional probabilities and posterior probabilities. To design this inference engine as an open source module, four class types at the top level are presented in the following subsections.

A. The Class Type Representing a Prior Probability

A prior probability within a Bayesian inference domain is an unconditional likelihood that supports a hypothesis without taking any evidence into consideration, such as $P(H_1)$, $P(H_2)$, ,or $P(H_n)$. The design of methods and attributes within the prior probability class type are described in Table 1.

B. The Class Type Representing a Conditional Probability

A conditional probabilities within a Bayesian inference domain is a conditional likelihood that an evidence will be revealed if a hypothesis is true, such as $P(E_i|H_1)$, $P(E_i|H_2)$, ,or $P(E_i|H_n)$. The design of methods and attributes within the conditional probability class type are described in Table 2.

C. The Class Type Representing a Knowledge Base

The knowledge base within a Bayesian inference domain consists of a list of prior probabilities and a list of conditional probabilities. The design of methods and attributes within the domain knowledge class type are described in Table 3.

TABLE III
THE DESIGN OF THE KNOWLEDGE BASE CLASS TYPE

Class Name: KnowledgeBase	
Purpose: Representing a knowledge domain	
Method	**What It Does**
Constructor	Initializing the knowledge base
ToString	Converting the knowledge base into a string representation
Attribure	**What It Means**
listOfPriorProbs	The list of prior probabilities in the knowledge domain
listOfCondProbs	The list of conditional probabilities in the knowledge domain

TABLE IV
THE DESIGN OF THE POSTERIOR PROBABILITY CLASS TYPE

Class Name: PostProb	
Purpose: Representing a resultant posterior probability	
Method	**What It Does**
Constructor	Initializing the domain knowledge
Inference	Applying Bayesian inference to compute the resultant posterior probability
ToString	Converting the resultant posterior probability into a string representation
Attribure	**What It Means**
kb	The current knowledge base
hypo	The hypothesis of the posterior probability
listOfEvids	The list of evidences observed

D. The Class Type Representing a Posterior Probability

A posterior probability within a Bayesian inference domain is an inference result that represents the likelihood of a hypothesis by seeing some evidences, such as $P(H_i|(E_1 \cap E_2 \cap ... \cap E_n))$.The design of methods and attributes within the posterior probability class type are described in Table 4.

IV. THE IMPLEMENTATION OF THIS INFERENCE ENGINE

This inference engine consists of the aforementioned four class types is implemented in Python programming language to be a reusable module as shown in Appendix A. This module is lightweight, domain independent and can be used in future probabilistic inference applications to speed up the development life cycle.

V. A DEMONSTRATION OF HOW TO APPLY THIS INFERENCE ENGINE

A demonstration of applying this inference engine is shown in Appendix B in which the following domain knowl-

ISBN: 978-988-14048-0-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

edge is instantiated:

$$P(H_1|E_1) = 0.2$$
$$P(H_2) = 0.3$$
$$P(H_3) = 0.6$$
$$P(E_1|H_1) = 0.3$$
$$P(E_2|H_1) = 0.9$$
$$P(E_3|H_1) = 0.6$$
$$P(E_1|H_2) = 0.8$$
$$P(E_2|H_2) = 0.1$$
$$P(E_3|H_2) = 0.7$$
$$P(E_1|H_3) = 0.5$$
$$P(E_2|H_3) = 0.7$$
$$P(E_3|H_3) = 0.9$$

and the following three posterior probabilities are inferred:

$$P(H_1|(E_1 \cap E_2 \cap E_3) = 0.136$$
$$P(H_2|(E_1 \cap E_2 \cap E_3) = 0.071$$
$$P(H_3|(E_1 \cap E_2 \cap E_3) = 0.793$$

The program output from this demonstration are shown in Appendix C.

VI. CONCLUSION

Nowadays, Bayesian theorem has offered a stochastic foundation for expert systems to deal with forecast and classification problems, ranging from pattern recognition, medical diagnostic, weather forecast, to natural language processing [4].

This inference engine is based on the theory of Nave Bayesian Network and implemented in Python programming language. In light of its ubiquity, this inference is designed to be domain-independent. As a performance-centered design this inference engine is functioning comprehensively without consuming excessive computation resources. I am aiming at publishing this inference engine as an open source to further benefit both industrial application developments and academic researches.

REFERENCES

[1] T. Bayes, "An Essay Towards Solving a Problem in the Doctrine of Chances," *Philosophical Transactions of the Royal Society*, Volume 53, Issue 1, pp. 370-418, 1763.
[2] J. Tabak, *Probability and Statistics: The Science of Uncertainty*, NY: Facts On File, Inc., USA, pp. 46-50, 2004.
[3] F. Yang, "Eliciting an Overlooked Aspect of Bayesian Reasoning," *ACM SIGCSE Bulletin*, Volume 39, Issue 4, pp. 45-48, 2007.
[4] G. DAgnostini, *Bayesian Reasoning in Data Analysis: A Critical Introduction*, NJ: World Scientific Publishing Co., Pte. Ltd., USA, 2003

Dr. Feng-Jen Yang became a Member (M) of IAENG in 20012. He received the B.E. degree in Information Engineering from Feng Chia University, Taichung, Taiwan, in 1989, the M.S. degree in Computer Science from California State University, Chico, California, in 1995, and the Ph.D. degree in Computer Science from Illinois Institute of Technology, Chicago, Illinois, in 2001, respectively. Currently, he is an associate professor of Computer Science and Information Technology at Florida Polytechnic University. Besides the currently academic career, he also has some prior research experiences. He once was a research assistant at the Chung Shan Institute of Science and Technology (CSIST), Taoyuan, Taiwan, from 1989 to 1993, as well as an engineer at the Industrial Technology Research Institute (ITRI), Hsinchu, Taiwan, from 1995 to 1996. His research areas include Artificial Intelligence, Expert Systems, and Software Engineering.

ISBN: 978-988-14048-0-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

APPENDIX A
THE IMPLEMENTATION OF THE BAYESIAN INFERENCE ENGINE

```
"""
File Name: bayesian.py
Creator: Feng-Jen Yang
Purpose: Implementing a Bayesian inference engine
"""

class PrioProb(object):
    """Representing a prior probability"""

    def __init__(self, initHypo, initProb):
        """Initializing the prior probability"""
        self.hypo = initHypo
        self.prob = initProb

    def __str__(self):
        """Returning a string representation of the prior probability"""
        return "P(" + self.hypo + ") = " + str(self.prob)

class CondProb(object):
    """Representing a conditional probability"""

    def __init__(self, initEvid, initHypo, initProb):
        """Initializing the conditional probability"""
        self.evid = initEvid
        self.hypo = initHypo
        self.prob = initProb

    def __str__(self):
        """Returning a string representation of the conditional probability"""
        return "P(" + self.evid + "|" + self.hypo + ") = " + str(self.prob)

class KnowledgeBase(object):
    """Representing a knowledge domain"""

    def __init__(self, initPrioProbs, initCondProbs):
        """Initializing the prior probabilities and conditional probabilities"""
        self.listOfPrioProbs = initPrioProbs
        self.listOfCondProbs = initCondProbs

    def __str__(self):
        """Returning a string representation of the knowledge base"""
        result = "The Knowledge Base:"
        for p in self.listOfPrioProbs:
            result += "\n" + str(p)
        for p in self.listOfCondProbs:
            result += "\n" + str(p)
        return result

class PostProb(object):
    """Representing Bayesian inference"""
    def __init__(self, initKB, initHypo, initEvids):
        """Initializing the knowledge base"""
        self.kb = initKB
        self.hypo = initHypo
        self.listOfEvids = initEvids

    def inference(self):
        """Computing the posterior probability"""
```

ISBN: 978-988-14048-0-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

```
        numerator = 1
        for cp in self.kb.listOfCondProbs:
            if (cp.hypo == self.hypo) and (cp.evid in self.listOfEvids):
                numerator *=cp.prob
        for pp in self.kb.listOfPrioProbs:
            if pp.hypo == self.hypo:
                numerator *= pp.prob
                break

        denominator = 0
        for pp in self.kb.listOfPrioProbs:
            term = 1
            for cp in self.kb.listOfCondProbs:
                if (cp.hypo == pp.hypo) and (cp.evid in self.listOfEvids):
                    term *=cp.prob
            term *= pp.prob
            denominator += term
        return round(numerator/denominator, 3)

    def __str__(self):
        """Returning a string representation of the inference reault"""
        result = "The Posterior Probability:\nP(" + self.hypo + "|("
        for e in self.listOfEvids:
            result += e

            if e != self.listOfEvids[-1]:
                result += "^"
        result +=  ")) = " + str(self.inference())
        return result
```

APPENDIX B
A DEMONSTRATION OF APPLYING THIS INFERENCE ENGINE

```
"""
File Name: demo.py
Creator: Feng-Jen Yang
Puopose: A demonstration of using this Bayesian inference engine
"""

from bayesian import *

def main():
    """The main function that coordinates the program execution"""

    #Instantiate the prior probabilities
    listOfPrioProbs = [PrioProb("h1", 0.2), PrioProb("h2", 0.3), PrioProb("h3", 0.6)]

    #Instantiate conditional probabilities
    listOfCondProbs = [CondProb("e1", "h1", 0.3), CondProb("e2", "h1", 0.9),\
                       CondProb("e3", "h1", 0.6), CondProb("e1", "h2", 0.8),\
                       CondProb("e2", "h2", 0.1), CondProb("e3", "h2", 0.7),\
                       CondProb("e1", "h3", 0.5), CondProb("e2", "h3", 0.7),\
                       CondProb("e3", "h3", 0.9)]

    #Instantiate the knowledge base
    kb = KnowledgeBase(listOfPrioProbs, listOfCondProbs)

    #Display the knowledge base
    print(kb, "\n")

    #Perform the Bayesian inference to compute the posterior probability
```

ISBN: 978-988-14048-0-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

```
    pp = PostProb(kb, "h1", ["e1", "e2", "e3"])

    #display the inference result
    print(pp, "\n")

    #Perform the Bayesian inference to compute the posterior probability
    pp = PostProb(kb, "h2", ["e1", "e2", "e3"])

    #display the inference result
    print(pp, "\n")

    #Perform the Bayesian inference to compute the posterior probability
    pp = PostProb(kb, "h3", ["e1", "e2", "e3"])

    #display the inference result
    print(pp, "\n")

#The entry point of program execution
main()
```

APPENDIX C
THE RESULTS OF THE DEMONSTRATION

```
The Knowledge Base:
P(h1) = 0.2
P(h2) = 0.3
P(h3) = 0.6
P(e1|h1) = 0.3
P(e2|h1) = 0.9
P(e3|h1) = 0.6
P(e1|h2) = 0.8
P(e2|h2) = 0.1
P(e3|h2) = 0.7
P(e1|h3) = 0.5
P(e2|h3) = 0.7
P(e3|h3) = 0.9

The Posterior Probability:
P(h1|(e1^e2^e3)) = 0.136

The Posterior Probability:
P(h2|(e1^e2^e3)) = 0.071

The Posterior Probability:
P(h3|(e1^e2^e3)) = 0.793
```

ISBN: 978-988-14048-0-0
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)