Bayesianism as a Whole

reasoning, inference and learning
using Bayesian techniques

Abstract

The current paper provides an overview of Bayesian techniques applied to frequently appearing issues in the field of Artificial Intelligence. It covers design of belief systems, reasoning under uncertainty, use in advanced learning systems such as Bayesian Problem Learning and application to various fields. Lastly it touches limitation and disadvantages of such approaches to discussed problems.

I. Introduction

Early models of intelligent systems followed the model of reasoning used in first order predicate calculus. They incorporated assumed *true* premises, correct *sound* inference rules and conclusions that are guaranteed to be *true* and correct. However, such systems have little or no use in practical problem applications.

If description of domain is presented as *true* or *false* clauses of predicate calculus, to sufficiently describe necessary states of given domain one needs maintaining a huge table of clauses. Quickly, such table grows to the extent that it can no longer be efficiently traced. Nevertheless, given such a table with efficient ways to query its entries, the system cannot be considered intelligent because it operates on given set of axioms that has been predefined by a designer and never modified during running time.

Ideally, intelligent systems should respond to changing environment conditions and infer new information about domain. This new information often contradicts with pre-coded *true* assumptions, and thus assumptions must be modified. Early examples have seen use of *nonmonotonic* logic, that is reasoning using non-traditional mathematical logic. In traditional, *monotonic* reasoning, once knowledge is added to the system, it will never make the set of true statements decrease. This is different from how human draw conclusions based on their current system of beliefs, we often reconsider the values of our beliefs based on gained experience.

One of the first approaches to *nonmonotonicity* was incorporation of various techniques based on *abduction*, which is an unsound inference rule, meaning that when P implies Q conclusion is not necessarily true for every interpretation in which premises are true. *Nonmonotonic reasoning* required advanced table management systems to maintain appropriate number of *true* assumptions and modify existing *true* axioms during running time, hence Truth Maintenance System (TMS) has been introduced. Unfortunately, such approach was not practical also: once new knowledge was added to the system, not only directly affected *true* statements were needed to be modified, but also all dependable statements should have been changed accordingly. In practical

applications, there is exponential number of dependencies in a domain and thus modification becomes intractable task.

Current and most successful approaches that handle intractability in reasoning under uncertainty employ wide use of statistics and decomposition of problems using various graph models to reduce search spaces. *Probability* and *possibility* factors have been assigned to decisions where choice is necessary. One of widely used model is Belief Networks, which is a graphical representation that captures relationships between variables in domain. Examples: and/or graphs, Markov models, Dynamic Belief Networks, Naïve Bayes, Bayesian Belief networks.

II. Inference under Uncertainty

Bayesian Statistics allows joint probability be decomposed into product of conditional probabilities and to compute *posterior* probabilities, thus modifying systems degree of belief after new knowledge has been discovered. However, computation of all joint probabilities using Bayesian Statistics alone is intractable.

Consider a toy example with just five variables and given prior probability as *true* or *false* to each variable. Let's assume the set of variables is {a,b,e,j,m}. Joint probability for entire system is

$$\boldsymbol{P(a,b,e,j,m) = P(j|a,b,e,m)P(m|a,b,e)P(e|a,b)P(b|a)p(a)}.$$

It is easy to notice that the cost of producing joint probability table is exponential, in our toy example it would have 2^5 possible entries. Therefore, computing joint probabilities directly cannot be done for practical application involving decent number of variables.

Belief Networks

Belief network is a probabilistic graphical model that is a representation of probability conditions and dependencies. Belief Network dramatically reduces dependability while maintaining necessary four primitive relationships of likelihood, conditioning, relevance and causation. Pearl (1988). Bayesian Belief Network (BBN) is a network that uses Bayesian techniques to compute *posteriori* probabilities, that is to modify *priori* probabilities once new knowledge has been discovered.

BBN is a DAG (Directed Acyclic Graph), where random variables are represented as nodes of a graph and conditional probabilities are edges between nodes representing dependent variables. Nodes that do not have ancestors are independent variables. In BBN, the full joint distribution is expressed as a product of conditionals of smaller complexities:

$$\boldsymbol{P(X_1, X_2, \dots, X_n) = \prod_{i=1}^{n} P(X_i|parent(X_i))}$$

That is, computation of joint probability is reduced to computation of conditional probability of direct parent variables only.

Consider again our toy example with five random variables (Fig 1). Using *brute force* approach to compute joint probability of *P(a,b,e,j,m)*, the complexity is exponential in terms of input parameters, that is 2^5.However, when we reduce problem to BBN, complexity decreases to exponential in terms of number of parents of the node. For example, the same probability becomes: $\boldsymbol{P(a,b,e,j,m) = P(b)P(e)P(a|b,e)P(j|a)P(m|a).}$

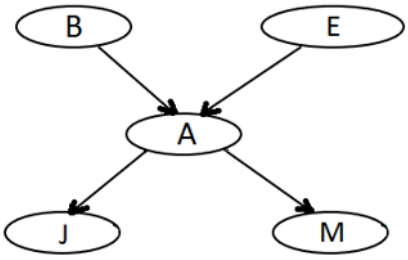

Fig. 1

Structure of BBN

BBN must be designed for specific problem where all *priori* probabilities are known in advance. Root of the network or Hypothesis nodes (nodes *B* and *E* in Fig. 1) have unconditional probabilities that are also must be known before the network is built. Probabilities are stored in probability table *P(A)* at that node. Additionally, nodes feature dynamic values *Bel (a)* that represent posterior probability and reflects the overall belief in the proposition $A = a$ given all the evidence, that is *Bel (a) = p(b|e).* To compute joint probabilities and assign belief values to nodes, network needs to be propagated.

Propagation

Propagation of BBN is a traversal of network in order to assign certain values to nodes. There are two types of propagation: forward and backward. During forward propagation, joint probabilities are being assigned as discussed above, while during back propagation, belief value are being computed and priori probabilities are modified given all the evidence *(e)* and using famous Bayesian theorem.

Consider another example (Fig. 2). Nodes W, X, Y are random variables and edges captures conditional probabilities as evidences (e). Edges with e^+ denote conditions on which X is dependent, while e^- denote conditions on which X's children are dependent. Now the value of belief of X is computed using Bayesian Theorem and the following formula:

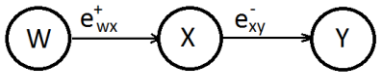

Fig. 2

$$\mathrm{Bel}(x) = \mathrm{p}\left(x \mid \mathbf{e}_X^-, \mathbf{e}_X^+\right) = \frac{\mathrm{p}\left(\mathbf{e}_X^- \mid x, \mathbf{e}_X^+\right)\mathrm{p}\left(x \mid \mathbf{e}_X^+\right)}{\mathrm{p}\left(\mathbf{e}_X^- \mid \mathbf{e}_X^+\right)} = \frac{\mathrm{p}\left(\mathbf{e}_X^- \mid x\right)\mathrm{p}\left(x \mid \mathbf{e}_X^+\right)}{\mathrm{p}\left(\mathbf{e}_X^-\right)}.$$

Defining $\pi(x) = p(x|e_x^+)$, $\lambda(x) = p(e_x^-|x)$ and $\alpha = 1/p(e_x^-)$ we have

$$\boldsymbol{Bel(x) = \alpha\pi(x)\lambda(x)}$$

Since α is total probability which is the same for all variables, its computation is omitted and it serves as normalization factor. The computation of $bel(x)$ is then completed using following formulas and illustrated on fig 3.

$$\pi(x) = \mathrm{p}\left(x \mid e_X^+\right) = \sum_w \mathrm{p}\left(x \mid w\right) \mathrm{p}\left(w \mid \mathbf{e}_W^+\right) \Rightarrow \boldsymbol{\pi}(X) = \boldsymbol{\pi}(W) \cdot \mathbf{P}\left(X \mid W\right)$$

$$\lambda(x) = \mathrm{p}\left(\mathbf{e}_X^- \mid x\right) = \sum_y \mathrm{p}\left(y \mid x\right) \mathrm{p}\left(\mathbf{e}_Y^- \mid y\right) \Rightarrow \boldsymbol{\lambda}(X) = \boldsymbol{\lambda}(Y) \cdot \mathbf{P}\left(Y \mid X\right)$$

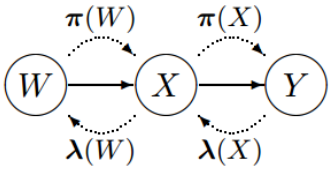

Fig 3

BBNs are widely used today in all kinds of applications. Their use varies from simple systems with few variables such as weather or traffic prediction to complex projects where BBN is a small part of inference and learning process.

III. Learning in Complex systems

One example of complex learning systems is BPL (Bayesian Program Learning). BPL was designed by researchers from CMU in 2015, it is complex learning/inference system that competes with most sophisticated deep neural networks setups for character recognition application, a common benchmark used to compare performance of various intelligent systems.

Idea behind BPL is to challenge human ability of inference, learning new concepts, and creating new exemplars based on sparse amount of data. The crucial difference of BPL is unlike connectionist systems that require large amount of data to train, BPL uses generative approach to object recognition, it tries to reconstruct objects from smaller parts and therefore requires much sparse training data.

Let us use the following definitions: **primitive** – the most basic part of character; **stroke** - part of character that starts with pressing the pen against the paper and terminated by lifting it up; **substrokes** - are more primitive movements separated by brief pauses of pen. See figure 4.

BPL generates concepts or **types**, that is simple probabilistic programs, then uses two-step Learning and apply Bayesian posterior inference to match these concepts against original data. It unites three features: compositionality, causality and learning-to-learn. Learning-to-learn uses hierarchical Bayesian modeling that allows construction of new concepts using existing ones and maintaining causal and compositional properties. BPL operates using two-step generation: **(i)** generation of **types** and **(ii)** generation of **tokens**.

(i) At the beginning BPL is pretrained with data taken from Omniglot from where it retrieves basic parts of characters. Then using probabilistic inference programs tries to build sub-strokes and combine them into strokes using relations.

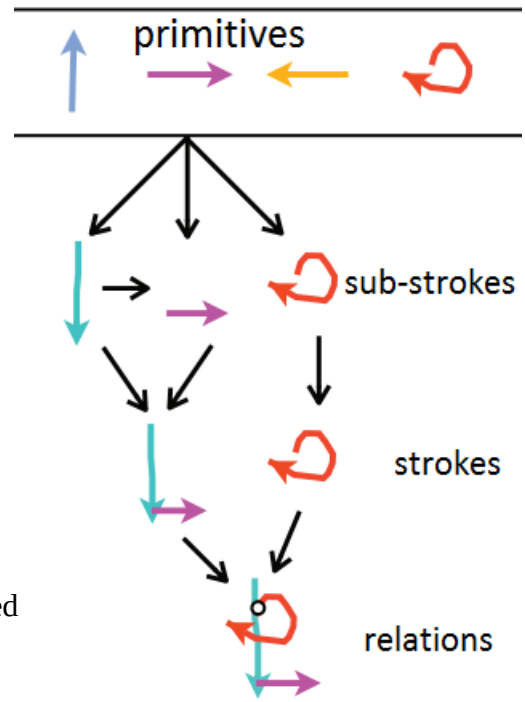

Fig. 4

Entire joint probability for **types** is as follows:

Type $\psi=\{k,S,R\}$, where S are strokes, R are relations, and $k = |S|$.

$$P(\psi) = P(k)\prod_{i}^{k} P(S_i)P(R_i|S_i,\dots,S_{i-1})$$

Each stroke S_i is composed of sub-strokes, that is $S_i = \{s_{i1},\dots,s_{in}\}$ where number of sub-strokes is sampled from frequency $P(n_i|k)$.

Sub-stroke are decomposed in three variables: $s_{ij} = \{z_{ij}, x_{ij}, y_{ij}\}$ where z_{ij} is index of primitives with distribution as follows:

$P(z_i) = P(z_{i1})\prod_{j=2}^{n_i} P(z_{ij}|z_{i,j-1})$.

Thus, **stroke** has the following joint distribution: $P(S_i) = P(z_i)\prod_{j=1}^{n} P(x_{ij}|\, z_{ij})P(y_{ij}|z_{ij})$.

Relation R_i specifies how strokes are positioned, scaled and joined. Relations come in four types: *Independent, Start, End, Along.* Each type has different sub-variables and dimensionality.

(ii) At the second stage, generation of tokens (Fig. 5), Tokens Θ_i are being generated by adding noise and variance, sampling start location L_i, composing stroke's trajectory T_i, affine transformations A_i, and second noise process that stochastically flips pixels interpreting their values as independent Bernoulli probabilities. Finally, Binary image created I^m created by stochastic function: $P(I^m\,|T^m, A^m)$. Joint distribution on types, set of tokens and images becomes very complex:

$$\boldsymbol{P(\psi|\theta^{m[i]},\, I^m) = P(\psi|\theta^{m[i]})}$$

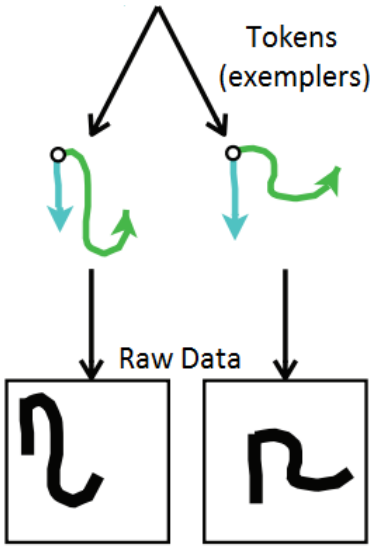

Fig. 5

Propagating such a network with generic algorithms for every **token** becomes nearly impossible task therefore the idea is selecting most successful and promising motor programs that are further refined with Markov chain Monte Carlo (MCMC) resulting in approximate posterior distribution $P(\psi, \theta^m \mid I^m)$. To save resources on computation it is possible to produce conditional samples from the first stage generation, type-level generation, that is $P(\psi|\theta^{m[i]}, I^m) = P(\psi|\theta^{m[i]})$.

At the final stage training images I^T are compared to different test images I^C to compute classification score as follows :

$$\begin{aligned}\log P(I^{(T)}|I^{(c)}) &\approx \log \int P(I^{(T)}|\theta^{(T)})P(\theta^{(T)}|\psi)Q(\theta^{(c)},\psi,I^{(c)})\,\mathrm{d}\psi\,\mathrm{d}\theta^{(c)}\,\mathrm{d}\theta^{(T)} \\ &\approx \log \sum_{i=1}^{K} w_i \max_{\theta^{(T)}} P(I^{(T)}|\theta^{(T)}) \frac{1}{N}\sum_{j=1}^{N} P(\theta^{(T)}|\psi^{[ij]})\end{aligned}$$

The highest score indicates belonging to a class.

One-shot classification

BPL was tested on one-shot image classification where it achieved excellent results. One shot classification creates new image I^1 based on a given image I^2 with the following probability distribution:

$$\begin{aligned}P(I^{(2)},\theta^{(2)}|I^{(1)}) &= \int P(I^{(2)},\theta^{(2)}|\theta^{(1)},\psi)P(\theta^{(1)},\psi|I^{(1)})\,\mathrm{d}(\psi,\theta^{(1)}) \\ &= \int P(I^{(2)}|\theta^{(2)})P(\theta^{(2)}|\psi)P(\theta^{(1)},\psi|I^{(1)})\,\mathrm{d}(\psi,\theta^{(1)}) \\ &\approx \int P(I^{(2)}|\theta^{(2)})P(\theta^{(2)}|\psi)Q(\theta^{(1)},\psi,I^{(1)})\,\mathrm{d}(\psi,\theta^{(1)}) \\ &= \textstyle\sum_{i=1}^{K}\sum_{j=1}^{N} \frac{w_i}{N} P(I^{(2)}|\theta^{(2)})P(\theta^{(2)}|\psi^{[ij]}).\end{aligned}$$

With only 10 parses through image I^1, BPL model could produce images with error margins comparable of humans and less than best deep learning algorithms: BPL – 3.3%, humans – 4.5, pre-trained Siamese convnet – 8%

IV. Conclusion and Thoughts

Intelligent systems using various statistical techniques, including Bayesian, has achieved great success in recent years. Bayesianism has deeply penetrated most intelligent systems and functions reasonably well for certain tasks. For example, BPL achieves less error rate than humans and passes “visual Turing test”.

However, BPL is far from being "rational". It remains deep and complex statistical model that is far from "how people think". BPL proved its performance on relatively simple concepts of handwritten characters, and its performance for other real life application is questionable. Moreover, choosing best motor programs and then propagating network using MCMC is extremely computation intensive task which is not appropriate for bigger domain.

One major limitation of Bayesianism is *priori* probabilities. Before creation of Bayesian Belief Network designers must know all probabilities for each variable and outcome, which is rarely the case in real world. Another issue is portability and modularity. For example, BPL operates well for character recognition, however to be useful in different domains, BPL would need to be highly modified, most of its belief networks would have to be rebuilt; and even new algorithm might need to be developed for types, token, parsing and for new concept creation.

Besides, the above technical and complexity limitations, perhaps the most crucial limitation is conceptual. Bayesian approach cannot handle uncertainty in background knowledge and prior probabilities function, it operates on given data without questioning its correctness. Furthermore, Bayesianism is always "conditional", implicitly or explicitly, there is always a relation between proposition and background knowledge. Although some techniques exist that allow flatten this issue such as Jeffrey's rule or introduction of "virtual proposition", Wang (2004), distinction between implicit and explicit conditioning is rarely made in real applications.

References

Luger, G (2002). Artificial intelligence: Structures and strategies for complex problem solving. Harlow, England: Pearson Education.

Pearl, J (2011). Bayesian networks. Department of Statistics, UCLA. UCLA: Department of Statistics, UCLA. Retrieved from:

Human-level concept learning through probabilistic program induction
BY BRENDEN M. LAKE, RUSLAN SALAKHUTDINOV, JOSHUA B. TENENBAUM
SCIENCE11 DEC 2015 : 1332-1338

Wang, P. (2004). The limitation of Bayesianism. Artificial Intelligence, 158(1), 97-106.

Pearl, J. (1988) Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann, San Francisco CA.

Braun, J. J. (2000) Dempster-Shafer theory and Bayesian reasoning in multisensor data fusion, in Sensor Fusion: Architectures, Algorithms, and Applications IV, Vol. 4051 of Proceedings of the SPIE, Orlando, FL

Hautaniemi, S. K., Korpisaari, P. T. & Saarinen, J. P. P. (2000) Target identiflcation with Bayesian networks, in Sensor Fusion: Architectures, Algorithms, and Applications IV, Vol. 4051 of Proceedings of the SPIE, Orlando, FL

Heckerman, D. (1996) A Tutorial on Learning With Bayesian Networks, Technical Report MSR-TR-95-06, Microsoft Corporation, Redmond, WA.

Suermondt, H. J. (1992) Explanation in Bayesian belief networks, PhD thesis, Stanford University, Stanford, CA.

Heckerman, D., 1999. Bayesian learning. In: Wilson, R., Keil, F. (Eds.), The MIT Encyclopedia of the Cognitive Sciences. MIT Press, Cambridge, Massachusetts