WDS'10 Proceedings of Contributed Papers, Part I, 25–30, 2010. ISBN 978-80-7378-139-2 © MATFYZPRESS

Bayesian Methods in Artificial Intelligence

M. Kukačka

Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic.

Abstract. In many problems in the area of artificial intelligence, it is necessary to deal with uncertainty. Using probabilistic models can also improve efficiency of standard AI-based techniques. Commonly used methods for dealing with uncertainty include Bayesian models, which can be used to describe and work with probabilistic systems effectively. This article reviews several models based on the Bayesian approach and typical algorithms used to work with them, along with some examples of their application.

Introduction

When dealing with problems in the real world, outcomes of actions or the actual state of the world is often not known with certainty. This may be caused by the world not being fully observable, or because our model of the world is not complete. In order to make good decisions under such conditions, it is beneficial to introduce probability to the reasoning process.

Since working with the full probabilistic description of the world can be in many cases too difficult, the Bayesian models are used to simplify the reasoning process. In the following paragraphs, these models and algorithms are described. Applications of the Bayesian models can be found in many areas, ranging from diagnostics in medicine and engineering (e.g. the notorious MS Office Assistant [*Horvitz et al.*, 1998]) to signal processing, robotics and financial risk management [*Neil et al.*, 2005]. Use of Bayesian models can be also beneficial in solving non-probabilistic problems, such as pattern recognition in images.

Basic notation

The following notation will be used in this article. Uppercase names will be used for random variables, such as X or $Rain$, lowercase names will denote their values, e.g. *true*. Standard notation will be used to express the probability and the conditional probability of variables, such as $P(X = x_1)$ and $P(Rain = true|X = x_1)$. The probability distribution of a random variable, which is a vector of probabilities of the variable's values, will be denoted by $\mathbb{P}(Rain)$. The *joint probability distribution* of multiple variables is then denoted by $\mathbb{P}(X, Y)$. Using this notation in an expression does not mean multiplication of a vector, but it is rather a simplification of a set of equations, where specific values from the distribution are substituted.

Working with probability

When considering a probabilistic problem, we specify it by describing its *probability space*, which consists of a set Ω of *elementary events*, a set Σ of *events* which are subsets of Ω, and a probability measure $P : \Omega \to [0, 1]$ (for a formal definition, see [*Matoušek, Vondrák* , 2001]). We can then name a set of random variables which map events to values in discrete or continuous value ranges. For example, in the case of describing a toss of two coins, the elementary events would represent the four possible outcomes, events could be e.g. "no head came up", "one head" and "two heads", and we could have a random variable H with range $\{0, 1, 2\}$ which will give us the number of heads in the toss. Finally, we can describe a *full joint probability distribution*, which specifies the probabilities of all possible combinations of values of all variables. In its simplest form it is a multidimensional table, where each dimension corresponds to one variable. In case we had three random variables A,B,C, each with a range $\{0, 1, 2\}$, the full joint distribution would take a form of a table of size 3x3x3, containing values of $P(A = 0, B = 0, C = 0)$, $P(A = 0, B = 0, C = 1)$, etc.

Due to a representation, we can answer queries about the problem domain. An example of such query can be "What are the probabilities of diseases of a patient with fever", which in our notation would translate to computing the distribution $\mathbb{P}(Disease|Symptom = fever)$.

To answer queries about the problem domain, we can use the so-called *marginalization* method. This method consists of summing out all variables that are not of interest while respecting the eventual observed *evidence* values. This produces a probability distribution of the queried variable or variables. By generalizing this method, we get a general inference procedure:

$$\mathbb{P}(X|e) = \alpha \cdot \mathbb{P}(X, e) = \alpha \cdot \sum_y \mathbb{P}(X, e, y) \tag{1}$$

where X is the variable of interest, e is the observed evidence, Y is the set of remaining unobserved variables and α is the normalization constant which ensures that the resulting distribution correctly sums up to 1.

The disadvantage of this approach is the size of the full joint distribution of more complex systems, which grows exponentially with the number of variables. It may be also difficult to determine the exact probabilities of rare events (e.g. a rare combination of diseases). Fortunately, not all values contained in the full joint distribution are necessary if we have some further knowledge about the domain. This assumption is the main idea behind the Bayesian models introduced in the next section.

There are two principles which allow us to improve the efficiency of general inference in probabilistic systems. The first one is *independence* between variables: variables X and Y are said to be independent if $\mathbb{P}(X|Y) = \mathbb{P}(X)$ (or, equivalently, $\mathbb{P}(Y|X) = \mathbb{P}(Y)$ or $\mathbb{P}(X, Y) = \mathbb{P}(X) \cdot \mathbb{P}(Y)$). Knowing the dependence relations between variables, we can factor the variable set into dependent subsets, which reduces the full joint distribution into separate joint distributions on these subsets. We will also use the notion of *conditional independence*: $\mathbb{P}(X, Y|Z) = \mathbb{P}(X|Z) \cdot \mathbb{P}(Y|Z)$, which specifies that variables X and Y are independent of each other given the value of variable Z.

The other important tool is the *Bayes' rule*, which allows us to work with distributions of dependent variables. It is specified by the following formula:

$$\mathbb{P}(X|Y) = \frac{\mathbb{P}(Y|X) \cdot \mathbb{P}(X)}{\mathbb{P}(Y)} \tag{2}$$

Bayesian networks

The so-called *Bayesian network*, as described e.g. in Chapter 14 of [*Russel,Norvig*, 2003], is a structure specifying dependence relations between variables and their conditional probability distributions, providing a compact representation of the full joint distribution of the whole system. It is defined by the following elements:

- a set of nodes, where each node represents a single variable
- a set of directed connections on these nodes, forming a directed acyclic graph, where a link specifies a dependence relationship between variables
- a conditional probability table (CPT) for each node in the graph, specifying a probability distribution of the corresponding variable conditioned by its parents in the graph (i.e. $\mathbb{P}(X_i|Parents(X_i))$).

We can retrieve the probability of any event in a system described by a Bayesian network using the following formula:

$$P(x_1, ..., x_n) = \prod_{i=1}^{n} P(x_i|parents(X_i)) \tag{3}$$

where $parents(X_i)$ denotes the specific values of the X_i's parent variables. This implies that the Bayesian network fully describes the full joint distribution of the system.

Exact inference

Because a Bayesian network describes the full joint probability distribution, the general inference method can be used for inference in Bayesian networks. The formula of general inference (see Eq. 1) only needs to be adjusted to work with the CPT tables representation. The expression is evaluated by looping over values of the variables that are summed out, which unfortunately causes the complexity of this algorithm to be exponential in the number of variables.

There are methods for improving the complexity of this approach. The *variable elimination* algorithm attempts to evaluate the equation in an optimal manner, by ordering the variables, storing reusable intermediate results and removing unnecessary variables. This algorithm is efficient for answering individual queries. For computing the posterior probabilities of all variables in the network, however, the *clustering* algorithm is better. It transforms the Bayesian network graph in order to make it *singly connected* by merging nodes that violate the single connectivity condition. In a singly connected network, the exact inference is only linear in the size of the network, the complexity of the inference is however only moved to the merged CPT tables.

Approximate inference

The problem of exact inference in Bayesian networks is *NP-hard*, because it contains inference in propositional logic as a special case. Because of this, approximate inference algorithms have been designed, which can give good estimations of the results in reasonable time. The general approach to approximation of posterior probabilities of queried propositions is *sampling*: a large number of samples is generated according to the distributions in the network, and the posterior probability is then estimated by the ratio of samples that fit the query to the total number of samples.

The basic algorithm, called *direct sampling*, is applicable when there is no observed evidence. The nodes of the network are traversed in topological order (i.e. parent is always visited before its children) and their values are chosen randomly from their distribution, conditioned by values of their parents. A sample consists of such an assignment of all variables in the network. Accuracy of this method improves with growing numbers of samples.

In case we have observed some evidence values, we can proceed in a manner similar to direct sampling and simply discard samples that do not conform with the evidence. This approach constitutes the *rejection sampling* algorithm. It is obvious that the more evidence we have, and the less probable it is to appear in a sample, the more samples are rejected, which makes it harder to obtain results with reasonable accuracy. The fraction of usable samples drops exponentially with growing number of evidence variables.

The problem of extensive rejections can be solved by using algorithms which produce only samples that agree with the observed evidence. The *likelihood sampling* algorithm weights each sample by the probability the evidence values would be produced as a part of it. The query is then answered by a weighted sum of the samples that conform to it. Its disadvantage is its low accuracy in case the evidence has a low probability of occurring or in case it appears late during the sampling process, in which case it does not influence the sampling process.

The *Markov chain Monte Carlo* (MCMC) algorithm, described in [*Pearl*, 1987], also generates only samples that conform to the evidence. Furthermore, the samples are generated with respect to the evidence regardless of where it is in the network. The network is considered to be in a particular state with all its variables assigned, and each such state constitutes a sample. Next network state is generated by randomly choosing one of the non-evidence variables and choosing its new value according to its distribution, conditioned by its Markov blanket - the parents of the node, its children, and parents of the children. The fraction of samples with the network in a certain state is then proportional to the posterior probability of this state. A query can be answered by summing the fractions of states conforming to the query.

Temporal models

Many problems involve reasoning about a probabilistic system that changes with time, for example when controlling a mobile robot. The uncertainty in such system can be caused by noisy sensors or a complex dynamic environment. Bayesian models described in the previous chapter can be extended to be applicable in such cases. Several of these models are described in this chapter. For more detailed descriptions, see Chapter 15 of [*Russel,Norvig*, 2003].

One of the main concepts in this chapter is that of a *time slice*, a moment in time in which the system is in a defined observable state. We will denote the hidden variables at time t as X_t, the observable variables as E_t and their particular observed values as e_t. We will also use $X_{a:b}$ to denote the sequence of variables from X_a to X_b inclusive.

Several assumptions are necessary to make the problem computationally feasible. First, we assume that the changes in the system are caused by a *stationary process*, which means that the rules of state transitions do not change over time. Next, we make the so-called *Markov assumption* - that the current state of the system depends only on a finite number of previous states. Then, we can call the system a *Markov process*. In a *first-order* Markov process, the current state depends only on the immediately preceding state, which, in our notation, can be written as $\mathbb{P}(X_t|X_{0:t-1}) = \mathbb{P}(X_t|X_{t-1})$. The conditional distribution $\mathbb{P}(X_t|X_{t-1})$ is called the *transition model*, which is common for all time slices t because of the stationary system assumption. Beside the transition model, we need to specify the *sensor model*, which influences the way in which the state described by X_t manifests itself in the values of evidence variables E_t. It is specified by a conditional distribution $\mathbb{P}(E_t|X_t)$.

Inference in temporal models

There are several basic inference tasks performed with the temporal models:

- *Filtering*: computing the probability distribution of the current state of the system, based on the evidence observed so far. It can be expressed as computing $\mathbb{P}(X_t|e_{1:t})$.
- *Prediction*: finding the probability of states of the system in the future, based on the evidence observed up to now. It consists of computing the distribution $\mathbb{P}(X_{t+k}|e_{1:t})$ for some $k > 0$.
- *Smoothing*: computing the probability of a state the system was in in the past, based on the evidence observed before and after the relevant time slice. It consists of computing the distribution $\mathbb{P}(X_k|e_{1:t})$ for some $0 \leq k < t$. The new evidence $e_{k+1:t}$, unavailable at time k, helps to estimate the state of the hidden variables more accurately.
- *Finding the most likely explanation*: finding the sequence of values of variables $X_{1:t}$ which is most likely the cause of the observed evidence $e_{1:t}$. It can be also written as finding the value of $argmax_{x_{1:t}} P(x_{1:t}|e_{1:t})$. The *Viterbi* algorithm can be used to find this sequence.

Hidden Markov model

In this and the following section, two special cases of the above described model are introduced, which are often used in practice because of their superior properties. The *Hidden Markov model* is a restriction of the above described temporal Bayesian model in the sense that the set of hidden variables X_t consists of only a single discrete variable. The transition model and the sensory model can then be specified in the form of matrices, which allows the basic tasks such as filtering to be expressed in terms of matrix operations. Systems with multiple hidden variables can be turned into a Hidden Markov model by merging the variables and their CPT tables.

The matrix description of the transition and sensor models allows the basic algorithms to be formulated in terms of matrix operations. This formulation makes it possible for some of these algorithms to be performed more efficiently. For example the smoothing algorithm can be modified to require only constant space, in the following way: the forward message is first propagated through the sequence without storing the results in each step, then it is propagated backward from $f_{t:t}$ along with the backward message by computing the forward message from the previous step via inverting the equation for conventional computation of the next forward message.

Kalman filter

The *Kalman filter*, introduced in [*Kalman*, 1960], is used to describe systems consisting of continuous variables, such as objects (robots, airplanes) moving through space. The transition and sensory models are defined by *conditional densities*, which are usually represented by some form of Gaussian distribution function (e.g. linear Gaussian or multivariate Gaussian distribution). The advantage of such representation is that the family of Gaussian distributions is closed under the standard Bayesian network operations, such as one step prediction or conditioning on new evidence, which are frequently used in the inference algorithms.

The disadvantage of Kalman filters is the failure to model systems which are not linear, e.g. in the presence of an obstacle in the space or presence of non-continuous conditions in the system. Such systems can be modeled with the use of *switching Kalman filter*, which consists of several Kalman filters modeling different variants of the system with different weights. A weighed sum of their predictions is then used.

Dynamic Bayesian network

The *Dynamic Bayesian network* (DBN) model is the most general of temporal models, and the two models described above can be thought of as its special cases. It can combine discrete and continuous variables in the X_t set, making the network general enough to describe even the most complex system, but also making the inference algorithms more complicated.

Exact inference can be performed in the DBN model by simply *unrolling* the sequence of time slices to form one big Bayesian network, then using one of the previously described algorithms. The obvious disadvantage of this naive approach is the computation cost, growing with each new time slice.

The sampling methods, such as likelihood weighting, are applicable to DBNs. But because the evidence variables are in this case always "downstream" from the state variables, the generation of the samples is done without any influence from the evidence whatsoever. This leads to a poor performance of the likelihood weighting algorithm. A better alternative is the *particle filtering* algorithm, which keeps the population of samples in regions of the state space with high probability. The idea behind this algorithm is to resample the population in each time slice, based on the likelihood of the samples conditioned by the evidence. This results in an efficient approximation algorithm.

Bayesian learning

Apart from finding a suitable network structure, which is still not a well solved problem, the learning of Bayesian networks can be viewed as searching in a space of hypotheses. This approach is described in more detail in Chapter 20 of [*Russel,Norvig*, 2003]. The individual hypotheses consist of a full set of network parameters. The idea of Bayesian learning is to assign probabilities to the hypotheses and update them based on the observed evidence.

Predictions can be made using all possible hypotheses, by weighting their separate predictions by the likelihood the hypothesis produced the observed data, and then summing the predictions. If the hypotheses space is too complex, approximations can be used to simplify this task. The *Maximum a posteriori* (MAP) approximation uses only the hypothesis with the highest likelihood to make the predictions.

Application of probabilistic methods in pattern recognition

The use of probabilistic models can be beneficial even for problems that are not inherently probabilistic. An example of such application is presented in an article by LeCun [*LeCun et al.*, 1998]. Here, the Viterbi algorithm is used to solve the problem of separating objects prior to their classification as a part of hand-written text recognition. The article describes two approaches to this problem: the Heuristic Over-Segmentation and the Space-Displacement Neural Networks. The former approach consists of using a heuristic to make a large number of cuts hoping that the correct segmentation will be their subset, and then using a classifier to recognize the segments and their combinations. The latter approach simply uses the classifier on neighboring locations in the presented image. Both approaches produce an *interpretation graph*, which describes all possible interpretations of the input in terms of paths traversing it from its start node to its end node. The classifier outputs can be considered to be the observed evidence variables, while the actual hand-written characters are the hidden variables. From this point of view, the output of the classifier behaves as a Hidden Markov Model. The Viterbi algorithm is then used to find the most probable sequence of characters, producing the final output of such pattern recognition model. This model is also globally trainable with a gradient descent algorithm which aims to minimize the penalty of the path with the lowest penalty and correct classification.

Conclusion

The Bayesian models represent a well-studied and effective way to describe and reason about problems that include uncertainty. They have a very large range of applications, from medicine and engineering to information processing. The temporal Bayesian models are used to describe probabilistic systems that change in time, they are applied to problems varying from signal processing to modeling physical processes. Furthermore, application of these models can be beneficial to problems which are not inherently probabilistic, such as the pattern recognition in images.

Acknowledgments. The author would like to thank his advisor doc. RNDr. Iveta Mrázová, CSc. for her advices and guidance. The present work was supported by the Czech Grant Agency under the contract no. 201/09/H057 and by the Charles University Grant Agency under the contract no. 4361/2009 (grant no. 136109).

References

LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-Based Learning Applied to Document Recognition, *Proceedings of the IEEE*, 1998, 86(11):2278-2324,

Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach, Second Edition, *Prentice Hall Series in Artificial Intelligence*, 2003

Kalman, R.: A new approach to linear filtering and prediction problems, *Journal of Basic Engineering*, 1960, 82, 35-46

Pearj, J.: Evidential reasoning using stochastic simulation of causal models, *Artificial Intelligence*, 1987, 32, 247-257

Horvitz, E. J., Breese, J. S., Heckerman, D., Hovel, D.: The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users, *Uncertainty in Artificial Intelligence, Proc. of Fourteenth Conference*, 1998, 256-265

Neil, M., Fenton, N. E., Tailor, M.: Using Bayesian Networks to Model Expected and Unexpected Operational Losses, *Risk Analysis: an International Journal (John Wiley & Sons)*, 2005, 25 (4): 963972

Matoušek, J., Vondrák, J.: The Probabilistic Method, Lecture Notes, Department of Applied Mathematics, Charles University, Prague, 2001