Automatic Differentiation Equipped Variable Elimination for Sensitivity Analysis on Probabilistic Inference Queries

Anonymous Author(s)
Affiliation
Address
email

Abstract

Probabilistic Models are a natural framework for describing the stochastic relationships between variables in a system to perform inference tasks, such as estimating the probability of a specific set of conditions or events. In application it is often appropriate to perform sensitivity analysis on a model, for example, to assess the stability of analytical results with respect to the governing parameters. However, typical programming language are cumbersome for encoding and reasoning with complex models and current approaches to sensitivity analysis on probabilistic models are not scalable, as they require repeated computation or estimation of the derivatives of complex functions. To overcome these limitations, and to enable efficient sensitivity analysis with respect to arbitrary model queries, e.g., $P(X|Y = y)$, we propose to use Automatic Differentiation to extend the Probabilistic Programming Language Figaro.

1 Introduction

In reasoning about an uncertain system, Probabilistic Models (PMs) can help understand how the system will behave even though aspects of it are stochastic or unknown. For example, a Bayesian network is a directed acyclic graph, which encodes local probability relationships, through the graph's structure [4]. Variable's relationships are defined through their probability density functions (pdfs), and the parameters that define them. Once the pdfs are in place, it is natural to ask questions on the model such as: the probability of a specific set of conditions of the system, the most probable state of variables, generating the likelihood of events, or asking these queries with evidence asserted. For instance, an analyst could wish to perform a query on the system such as: what is the probability X is true given we observe y (this can be written as $P(X|Y = y)$)? For a concrete case which will serve as a running example for this paper, consider the system graphically shown in Fig 1(a). It has random variables representation the occurrence of an earthquake and a burglary. These variables influence whether a burglar alarm sounds, which in turn influences whether a neighbor calls. In this example all the variables are Boolean. A query on this model could be: what is the probability that the alarm is tripped, given a call was received? Solving these inference tasks can be done by exact methods such as variable elimination, or approximate methods such as belief propagation, Monte Carlo, Gibbs Sampling, etc.). We envision supporting a wide range of diverse and complex PM so we encode our models via a probabilistic programming language. Probabilistic programing uses concepts from programming languages to compactly encode complex PM [2]. Specifically, we use an open source probabilistic programming language, Figaro [6].

In computing inference queries, there is no information embedded in the answer that provides insight as to how *stable* the solution is. E.g., if a parameter that defines the network is changed slightly, will the answer to the query change substantially? Questions of this type can be classified as *sensitivity analysis*, which can be roughly described as the study of how the variation in the output of a mathematical model or system can be affected by variation in its inputs [8]. Suppose it is known to the analyst that there is significant uncertainty associated with a parameter x for this a probabilisitic model; it reasonable to ask: how far off must the true value of x be from our estimated value, to change the query output by 5%? Referencing the running example, the probabilities that describe the relationship between variables are encoded via a conditional probability table. These probabilities *parametrize* the model, giving numerical values which are used in performing queries on the it. For example, suppose the parameter of interest is the probability that alarm is true, given earthquake is true, and burglary is false: $x = P(alarm = true|earthquake = true, burglary = false)$. Traditional analysis of changes in the output of a model with respect to a specific parameter is possible [5], but manually repeating this analysis for all parameters is slow and laborious. More recent efforts explore means to compute node-to-node derivatives [1], but do not scale to more general inference tasks.

At the core of sensitivity analysis is the question of how much a function is changing with respect to changes to its input, captured by the mathematical notion of a gradient. Once gradients are obtained, they can be used to search for optimal parameter values which answer the sensitivity queries posited by the user. In our example the inference queries act as the function and the input are the parameters that define the PM. Computing these queries is often computationally expensive, and can be subject to variation due to approximations made to render calculations computationally feasible, even for parameters with constant value. A variety of methods exist to compute gradients, and we consider dual number enabled Automatic Differentiation (AD) [7] for its ability to compute exact derivatives in a computationally efficient manner [3]. There are several different mechanisms for AD, which compute gradients distinctly (e.g., forward accumulation, reverse mode), and we adopted a pure dual number approach for our prototypes. The semi-ring they form is analogous to the semi-ring used in the Variable Elimination solver used Figaro, allowing for a more straightforward implementation.

2 Approach

2.1 Sensitivity Query Example Problem

The motivation for developing a tool for performing sensitivity analysis, was answering questions such as: what is the minimum amount we can change parameter x by, such that the output of a query changes by ϵ. This can be expressed in the minimization problem

$$\begin{aligned} \underset{\delta}{\text{argmin}} \quad & \delta \\ \text{subject to} \quad & |f(x_0 + \delta) - f(x_0)| > \epsilon. \end{aligned} \tag{1}$$

where $f(x)$ is the query, and x is the parameter of interest. We now extend our example with a sensitive analysis query and pose it as the minimization problem in Eq. 1. Let the query of interest be the probability of the alarm being triggered given the neighbor is calling, and the parameter of interest be the prior on an earthquake occurring: $x = P(earthquake = true), f(x) = P(alarm = true|call = true)$. Note that for this analysis we consider all other parameters constant, so that f is only a function of x. In order to solve the minimization problem, we will a Newton's line search to update x:

$$x_{i+1} = x_i - \eta \frac{f(x_i)}{f'(x_i)} \tag{2}$$

where η is the learning rate for the search. The challenge now becomes computing the derivative $f'(x)$ at each step. Symbolic methods are appealing in that they are exact, but they suffer from expression bulge as models get complex. This quickly leads to intractable calculations as models become complex. Numerical derivatives are unappealing because computing the query $f(x)$ may be expensive. Worse, approximations and sampling methods entail that the results of successive queries of $f(x)$ may vary on the same scale as the true derivative, which creates "noisy" gradient estimates, e.g., when evaluating $f(x+\delta) - f(x)$. Therefore, we use dual numbers to perform automatic differentiation to yield an exact derivative without computing the query value multiple times. [1]

2.2 Extending Figaro's Variable Elimination Algorithm with AD

Dual numbers form a semiring, extending real numbers by adjoining a new element d with the property $d^2 = 0$ (e.g., a dual number may be written as $a + bd$, where $a, b \in \mathbb{R}$). Dual numbers have the interesting property that when a dual number is passed into a function, the output contains the gradient value in its dual component: I.e.,

$$f(a + bd) = c + ed \quad \implies f'(a) = e \tag{3}$$

This result depends on the property $d^2 = 0$ and the arithmetic associated with the dual number semiring. To perform inference on this PM we use Figaro's Variable Elimination (VE) algorithm, but instead of standard arithmetic we compute over a dual number semiring. With the parameter of interest expressed as a dual number, the coefficient of the dual number in the output is the derivative of the query with respect to the parameter of interest.

To see this, consider our example (the query is the probability of the probability of the alarm being triggered given the neighbor is calling, and the parameter of interest is the prior on an earthquake occurring). Using the chain rule, we can write out the analytic expression the probability:

$$P(A^+|C^+) = \frac{1}{Z} \sum_{E,B} P(E)P(B)P(A|B,C)P(C|A) \tag{4}$$

where Z denotes $P(C^+)$, X^+ denotes X is true, and X^- denotes X is false. We can then now plug in a dual number $x + d$ for the parameter of interest.

$$P(A^+|C^+) = \frac{1}{Z} \sum_{E,B} (x+d)_E P(B)P(A|B,C)P(C|A) \tag{5}$$

where $(x + d)_E$ takes on the value of $P(E = true)$ or $P(E = false)$ depending on which value of E is being used in the summation. After the summation is performed we group terms by real components and dual components to get:

$$P(N^+) = \alpha + \beta d \tag{6}$$

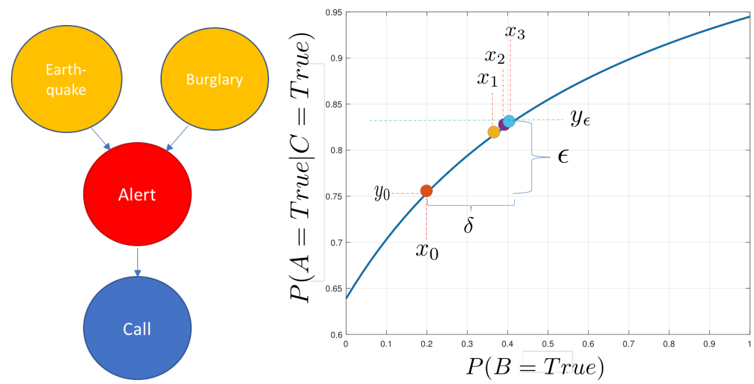
Where α is the numerical answer to the inference query, and β with derivative of the query with respect to the parameter of interest.

We implemented this in Figaro by extending the initial *factors* produced by the VE algorithm with dual numbers (i.e., the factors produced from VE are similar to the individual terms produced in Eq. 4). These factors will have numerical values associated with them; for the factors relevant to the parameter of interest we give the dual coefficients a value of 1, and all others 0. Once the terms are assigned the correct dual numbers the VE algorithm runs as usual, but with arithmetic defined by the dual number semiring. The output contains a dual number (such as in Eq 6), which will contain both the query output in the natural number component and the gradient information in the dual component. We refer to this algorithm as Variable Elimination with Automatic Differentiation (VEAD).

2.3 Results

For the example in 2.1 we used the gradients obtained by the VEAD algorithm to execute a Newton's line search as in Eq. 2. The results are depicted in Fig. 1(b), where one can see the parameter value

[1]The computational cost is roughly a small constant factor more than the cost of computing the query



(a) (b)

Figure 1: a) An example four node Bayesian model describing a scenario where a burglary or an earthquake influences whether an alarm goes off, which influences whether a neighbor calls. Each circle is a Boolean random variable characterized by conditional probability tables. b) An iterative search over the parameter space leads to the optimum value, solving the sensitivity query.

quickly converging to optimum value which caused the query to change by a target 5%. The method was also tested with a variety of PMs of varying complexity and results were verified by manually and numerically checking the gradients.

3 Conclusion and Future Work

We explored the usage of Automatic Differentiation to extend the probabilistic programming language Figaro, with a tool for efficiently calculating gradients of probabilistic inference queries. These gradients can be used to perform sensitivity analysis on these queries in order for an analyst to answer such questions as: how far off must the true value of a parameter of the system be from our estimated value, to change the query output by 5%? We have shown questions such as this can be answered with our framework utilizing a newtons line search to solve a for the optimum parameter value.

There are ample directions for future work. First, to validate the efficiency of our method, we would like to conduct a series of "wall clock tests" versus purely numerical means (even though these numerical derivatives may suffer from numerical instabilities, which the dual number approach for calculating does not). Secondly, we would like to explore augmenting powerful, approximate solvers such as Markov Chain Monte Carlo, Gibbs Sampling, or Importance Sampling, with the ability to ingest dual numbers in order to automatically compute gradients when computing queries, in the same we extended Variable Elimination to produce gradients.

References

[1] Chan, H., Darwiche, A.: Sensitivity analysis in bayesian networks: From single to multiple parameters. In: Proceedings of the 20th conference on Uncertainty in artificial intelligence. pp. 67–75. AUAI Press (2004)

[2] Goodman, N.D.: The principles and practice of probabilistic programming. In: Proceedings of the 40th annual ACM SIGPLAN-SIGACT symposium on Principles of programming languages. pp. 399–402. ACM (2013)

[3] Griewank, A., Walther, A.: Evaluating derivatives: principles and techniques of algorithmic differentiation. SIAM (2008)

[4] Koller, D., Friedman, N.: Probabilistic graphical models: principles and techniques. MIT press (2009)

[5] Laskey, K.B.: Sensitivity analysis for probability assessments in bayesian networks. IEEE Transactions on Systems, Man, and Cybernetics 25(6), 901–909 (1995)

[6] Pfeffer, A.: Creating and manipulating probabilistic programs with Figaro. In: 2nd International Workshop on Statistical Relational AI (2012)

[7] Rall, L.B.: Automatic differentiation: Techniques and applications (1981)

[8] Saltelli, A., Ratto, M., Andres, T., Campolongo, F., Cariboni, J., Gatelli, D., Saisana, M., Tarantola, S.: Global sensitivity analysis: the primer. John Wiley & Sons (2008)