Rapid Prototyping of Probabilistic Models: Emerging Challenges in Variational Inference

Yarin Gal
University of Cambridge
yg279@cam.ac.uk

Abstract

Perhaps ironically, the deep learning community is far closer to our vision of "automated modelling" than the probabilistic modelling community. Many complex models in deep learning can be easily implemented and tested, while variational inference (VI) techniques require specialised knowledge and long development cycles, making them extremely challenging for non-experts. A possible solution, lifted from manufacturing, is to create a Rapid Prototyping pipeline by combining recent advances in stochastic VI and developments in symbolic differentiation. This seemingly naive combination of existing techniques opens the door to a series of exciting new research problems in VI and engineering. I discuss the limitations of current tools and propose novel ideas to research such as black-box variance reduction, VI model compositionality, and others. Similar ideas in deep learning have led to rapid development in model complexity, speeding up the innovation cycle. This pipeline forms a vital steppingstone in the road towards automated statistics, making VI accessible to larger audiences than before.

1 Introduction

Variational inference (VI, [1, 2, 3, 4]) has fundamentally changed the way by which we develop probabilistic models. Compared to cumbersome exact inference techniques, with VI we develop complex models and then approximate their posteriors with simple distributions. But VI requires specialised knowledge and long development cycles, making it extremely challenging for non-experts to develop new models. The VI pipeline requires a high degree of engineering experience, which biologists, physicists, and other potential users of the technique often lack. This is in contrast to our aim of simplifying the modelling process and automating it as far as possible.

The normal development cycle of variational inference is composed of several steps.

1. We first obtain the data $\mathbf{X}$, examine it, and design an initial probabilistic model trying to capture the distribution that generated the data,
$$p(x^*|\mathbf{X}) = \int p(x^*|\omega)p(\omega|\mathbf{X})\mathrm{d}\omega$$
with some latent random variable ω. The posterior $p(\omega|\mathbf{X})$ is often intractable.
2. We then choose an approximating variational distribution $q_\theta(\omega)$ trying to match the properties of the intractable posterior closely.
3. In the third step we implement the model: we evaluate the divergence between the approximating posterior and the true posterior obtaining a lower bound,
$$\mathcal{L}(\theta) := \int q_\theta(\omega) \log p(\mathbf{X}|\omega)\mathrm{d}\omega - \mathrm{KL}(q_\theta(\omega)||p(\omega)).$$
This bound is often still intractable, forcing us to use various techniques to bound it further. We use matrix calculus to calculate the derivatives of the bound with respect to all varia-

tional parameters θ, implement the derivatives in a programming language of choice, test the derivatives using finite differences, and optimise the computation for performance and numerical stability.

4. Lastly we evaluate the model, examining the probabilistic model's ability to capture the distribution underlying our data and adjust the model accordingly. This cycle is repeated several times until we are happy with the model's performance.

By far the most time consuming step of this pipeline is the third one – the implementation step: every small change to the model in the last step of a cycle forces us to re-compute many of the derivatives, re-implementing them, re-optimising, and testing them again. Further, coming up with good bounds often requires broad familiarity with the literature (especially for non-conjugate models). For example, the Softmax likelihood is not conjugate to a Gaussian approximating distribution, and the integral of Gaussian-distributed latent variables together with a Softmax distribution is intractable. Many bounds have been suggested for this integral, requiring familiarity with fields ranging from machine learning to statistics: Jaakkola and Jordan's bound [5], the tilted bound [6], piecewise linear and quadratic bounds [7], the Bohning bound [8], and Blei and Lafferty's bound [9], to name a few.

This long pipeline has stood in the way of many users of probabilistic modelling, and curbed its adoption by the scientific community for a long time. Probabilistic programming has attempted to answer some of these difficulties, with programming languages like BUGS [10] (followed by WinBUGS [11] and later OpenBUGS [12]), Stan [13], Church [14], and many others suggested over the past 20 years. Most of these techniques work well with small simple models, but these are still under active development and fairly impractical for real-world use. The community's concentration on theoretical soundness and Turing completeness mostly resulted in developments which leave pragmatic needs unanswered.[1] An alternative approach can be based on the fusion of VI and ideas rooted in manufacturing.

2 Rapid Prototyping of Probabilistic Models

Similar problems to the above were encountered in manufacturing in the 1980s. The long and costly development cycle of mechanical parts significantly hurt the process of innovation. This problem of long development cycles was answered with Rapid Prototyping (RP), in which an engineer would quickly fabricate scale models using 3D printing before building physical parts at scale. The process of fabrication allows for quick development cycles and reduces development time and costs.

The equivalent of RP in probabilistic modelling can be built using the combination of recent advances in VI, stochastic optimisation, and advances in computer engineering. Exciting initial ideas proposed in [16, 17] suggested the use of sampling-based variational inference to approximate the intractable divergence in the implementation step of the pipeline above instead of bounding it. In sampling-based variational inference we generate samples from the approximate posterior and evaluate the log likelihood at these samples, obtaining an unbiased estimator to the log evidence which we optimise. Adaptive learning-rate stochastic optimisation is then used to optimise the objective in a black-box setting, requiring no parameter tuning.

Sampling-based VI requires a lower level of familiarity with the literature but doesn't answer some of the key issues with the implementation step. Each change to the model at the end of each development cycle still requires the calculation, implementation, optimisation, and testing of the objective and its derivatives. This process is extremely time-wasteful and requires a significant amount of engineering expertise. Even a tiny bug in the implementation can cost countless hours of debugging, especially for complex models. However with developments in computer engineering from the last couple of years, which have been widely adopted by the deep learning community, we can alleviate this process considerably.

[1] The probabilistic programming community has recently started shifting towards more pragmatic solutions. Exciting ideas discussed in [15] were developed in parallel to the ideas I rely upon in this paper. But the framework in [15] was sadly developed from a different stand-point which would not allow the incorporation of many of the insights below. This will be explained further in section 3.

2.1 A New Pipeline

The computation and implementation of the variational objective's derivatives is a time consuming task, but also a mechanical one, which can be easily automated. Symbolic differentiation programming languages such as Theano [18] have had tremendous success in deep learning, and have been used to create building-blocks that can be combined together to create complex models (see [19] for example). If used in VI together with sampling-based variational inference, we could simplify the development cycle to a neat pipeline, akin to RP in manufacturing.

Combining these existing techniques I thus suggest a new VI development cycle. After obtaining the data $\mathbf{X}$ and developing an initial generative model trying to capture the data's distribution, we choose approximating variational distributions trying to match the properties of the posterior as before. Instead of bounding the divergence, calculating derivatives, implementing, optimising, and testing these, we simply write down the generative model in a symbolic language,

$$\begin{aligned}&\texttt{var}\ \omega;\\ &f = ...\\ &\mathbf{X} = f(\omega);\end{aligned}$$

We simulate T samples from the approximate posterior and propagate them down the generative model,

$$\begin{aligned}\omega_t &\sim q_\theta(\omega);\\ \mathbf{X}_t &= f(\omega_t);\end{aligned}$$

We then evaluate the objective with the output of the generative model,

$$\mathcal{L}(\theta) \approx \frac{1}{T}\sum_{t=1}^{T} \log p(\mathbf{X}_t) - \mathrm{KL}(q_\theta(\omega)||p(\omega)).$$

We symbolically differentiate the objective and evaluate the derivatives with the same samples, obtaining a noisy but unbiased gradient estimate. Together with a stochastic optimiser, we are guaranteed to converge to an optimum [16, 20, 21, 22]. As before, in the fourth step we evaluate the model, examining the probabilistic model's ability to capture the distribution underlying our data and adjust it accordingly. This cycle is repeated several times until we are satisfied with the model's performance.

As a concrete example we can look at [23], a probabilistic model of sequences of discrete variables with non-linear embeddings. In Gal et al. [23] we used a Softmax likelihood and a Gaussian prior, resulting in intractable model evidence. The original implementation using the Bohning bound required thousands of lines of Python code, with 4 pages of derivations. Using the techniques above

```
import theano.tensor as T
m = T.dmatrix('m') # ... and other variational parameters
X = m + s * randn(N, Q) # these are the generative model's variables
U = mu + L.dot(randn(M, K))
Kmm = RBF(sf2, l, Z)
Kmn = RBF(sf2, l, Z, X)
Knn = RBFnn(sf2, l, X)
KmmInv = T.matrix_inverse(Kmm)
A = KmmInv.dot(Kmn)
B = Knn - T.sum(Kmn * KmmInv.dot(Kmn), 0)
F = A.T.dot(U)+B[:,None]**0.5 * randn(N,K)
S = T.nnet.softmax(F) # model's output - Softmax probabilities
KL_U, KL_X = get_KL_U(), get_KL_X() # these are the KL terms
LS = T.sum(T.log(T.sum(Y * S, 1)))
            - KL_U - KL_X # and this is the lower bound we optimise
LS_func = theano.function(['''inputs'''], # compile the model
            LS)
dLS_dm = theano.function(['''inputs'''], # and the derivatives
            T.grad(LS, m))
# ... and optimise LS with RMS-PROP
```

Figure 1: **Example Python code resulting from the new pipeline.** Here, `m`, `s`, `mu`, and `L` are the variational parameters, and the generative model `S` (the probabilities of the discrete variables) is a function of latents `X`, `U`, and `F`. Our objective is `LS`.

we managed to implement the generative model and inference in Theano [18] with the simplest version implemented fully with 20 lines of code (fig. 1).

By almost completely avoiding the implementation step in the original development cycle, this new pipeline makes VI more accessible to non-experts, requiring far less expertise and time. The pipeline thus forms an invaluable steppingstone in our vision of automated statistics and scalable probabilistic programming.

This combination of existing techniques might seem naive, but the pipeline opens the door to a series of exciting new research problems in VI and engineering, which are studied next.

3 Emerging Challenges

There are many open problems to research to make this pipeline practical both from the VI perspective and the engineering perspective, with the development of new tools being a fundamental necessity. Here I discuss the limitations of current tools and propose exciting new leads to research. I will concentrate on Theano as the underlying engine for symbolic differentiation rather than alternative ones. Theano is immensely popular within the deep learning community and considerable amounts of resources have been invested in the engine itself and the surrounding ecosystem. Many practical deep learning frameworks have been built on top of Theano which will allow us to combine these with VI seamlessly (as explored in section 3.3). This is in contrast to the framework developed in [15] which does not allow us to leverage existing developments within the deep learning community.

3.1 Existing Tools

Numerous difficulties arose during our implementation in [23], focusing our attention on the lack of various tools in the field. The most prominent difficulty is the lack of appropriate symbolic differentiation tools for many of the models often used in VI.

As a concrete example we can look at the symbolic differentiation of matrix inverses, which are common in Gaussian approximating distributions. Theano and many other symbolic differentiation frameworks build a graph of symbolic variables and operations on these, and then optimise the graph to make its computations more efficient. Theano calculates the derivatives by propagating the chain rule throughout the graph, making graph optimisation vital since naive evaluations would be computationally too expensive.

However Theano and other common tools were built within the deep learning community and mostly aimed at its needs. Even though matrix inverse differentiation is supported, current graph optimisations are not ideal for models requiring this operation. This results in slow models which are often impractical. A similar problem exists with matrix determinants.

An additional engineering-directed difficulty is the lack of numerically stable techniques and efficient approximations in the graph optimisation for such operations. Many "tricks-of-the-trade" are used in VI implementations to avoid problems of numerical instability and to avoid large matrix multiplications. These are missing from existing symbolic differentiation frameworks.

With the introduction of such tools it will become much easier to design efficient models, with much smaller, readable, and extendible code-bases.

3.2 Black-box Variance Reduction

One of the major difficulties with sampling-based VI is estimator variance. Even though sampling-based VI simplifies inference considerably, large estimator variance can make optimisation infeasible. In such cases variance reduction techniques are needed, but these often force us to re-parametrise the model. The resulting re-parametrisation often leads to complicated inference and code.

This difficulty suggests the need for black-box variance reduction techniques. Black-box variance reduction techniques will be applied automatically to the symbolic graph without needlessly complicating the generative model. This will allow users of the sampling-based VI pipeline to simplify the development process by making estimator variance concerns irrelevant. A direct result is compact and more readable generative models.

Recent techniques proposed in [17, 20] set a good starting point for this.

3.3 Model Compositionality

Current research in VI is rather fragmented, and the implementation of existing published models can take days if not weeks. In the deep learning community, however, many common models are built-in into frameworks such as [19], and a vast number of published papers can be easily built from these simpler building blocks, becoming themselves higher level building blocks. This speeds-up the innovation cycle allowing fast-evolving model complexity. The Neural Queue forms a concrete example of this. It is based on the Neural Turing Machine, which in turn is based on Recurrent Neural Networks and simple deep networks as building blocks.

Thus, a last thread of open problems to research is the design of small VI building blocks. A unified VI framework based on the tools above will become a cornerstone in the field, making VI accessible to larger audiences similar to the BUGS project in 1989. With the recent framework casting many deep learning tools as VI in Bayesian neural networks [24, 25, 26] we already have many building blocks to start this endeavour with.

References

[1] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13. ACM, 1993.

[2] Steve Waterhouse, David MacKay, and Tony Robinson. Bayesian methods for mixtures of experts. 1996.

[3] David JC MacKay. Ensemble learning for hidden Markov models. 1997.

[4] Hagai Attias. Inferring parameters and structure of latent variable models by variational Bayes. In *Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence*, pages 21–30. Morgan Kaufmann Publishers Inc., 1999.

[5] Tommi S. Jaakkola and Michael I. Jordan. A variational approach to Bayesian logistic regression models and their extensions. In *Proceedings of the Sixth International Workshop on Artificial Intelligence and Statistics*, 1997.

[6] David A Knowles and Tom Minka. Non-conjugate variational message passing for multinomial and binary regression. In *Advances in Neural Information Processing Systems*, pages 1701–1709, 2011.

[7] Benjamin M Marlin, Mohammad Emtiyaz Khan, and Kevin P Murphy. Piecewise bounds for estimating bernoulli-logistic latent Gaussian models. In *ICML*, pages 633–640, 2011.

[8] Dankmar Böhning. Multinomial logistic regression algorithm. *Annals of the Institute of Statistical Mathematics*, 44(1):197–200, 1992.

[9] David Blei and John Lafferty. Correlated topic models. *Advances in neural information processing systems*, 18:147, 2006.

[10] A Thomas. BUGS: a statistical modelling package. In *RTA/BCS Modular Languages Newsletter*, volume 2, pages 36–38, 1994.

[11] D J Spiegelhalter, A Thomas, and N G Best. WinBUGS version 1.2 user manual. In *MRC Biostatistics Unit*, 1999.

[12] David Lunn, David Spiegelhalter, Andrew Thomas, and Nicky Best. The BUGS project: Evolution, critique and future directions. 2009.

[13] Bob Carpenter Hoffman, Matthew D. and Andrew Gelman. Stan, scalable software for Bayesian modeling. In *Proceedings of the NIPS Workshop on Probabilistic Programming*, 2012.

[14] ND Goodman, VK Mansinghka, D Roy, K Bonawitz, and JB Tenenbaum. Church: A language for generative models. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence, UAI 2008*, pages 220–229, 2008.

[15] Alp Kucukelbir, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic variational inference in Stan. *arXiv preprint arXiv:1506.03431*, 2015.

[16] John W Paisley, David M Blei, and Michael I Jordan. Variational Bayesian inference with stochastic search. In *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 1367–1374, 2012.

[17] Rajesh Ranganath, Sean Gerrish, and David M Blei. Black box variational inference. *arXiv preprint arXiv:1401.0118*, 2013.

[18] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010. Oral Presentation.

[19] Franois Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

[20] Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[21] Danilo J Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1278–1286, 2014.

[22] Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational Bayes for non-conjugate inference. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1971–1979, 2014.

[23] Yarin Gal, Yutian Chen, and Zoubin Ghahramani. Latent Gaussian processes for distribution estimation of multivariate categorical data. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015.

[24] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Insights and applications. In *Deep Learning Workshop, ICML*, 2015.

[25] Yarin Gal and Zoubin Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. *arXiv:1506.02142*, 2015.

[26] Yarin Gal and Zoubin Ghahramani. Bayesian convolutional neural networks with Bernoulli approximate variational inference. *arXiv:1506.02158*, 2015.