On the Reparameterisation Gradient for Non-Differentiable but Continuous Models

Dominik Wagner Luke Ong

1 Background

Variational inference is a very successful approach to Bayesian statistics, which frames posterior inference in terms of an optimisation problem. The idea is to approximate the posterior probability $p(\mathbf{z} \mid \mathbf{x})$ using a family of "simpler" densities $q_{\boldsymbol{\theta}}(\mathbf{z})$ over the latent variables $\mathbf{z}$, parameterised by $\boldsymbol{\theta}$. The optimisation problem is then to find the parameter $\boldsymbol{\theta}^*$ such that $q_{\boldsymbol{\theta}^*}(\mathbf{z})$ is "closest" to the true posterior $p(\mathbf{z} \mid \mathbf{x})$.

A reduction of variance, which is essential in practice, can often be achieved by a *reparameterisation* of the distribution $q_{\boldsymbol{\theta}}$ via a diffeomorphic transformation $\boldsymbol{\phi}_{\boldsymbol{\theta}}$ of a base (or noise) distribution q, which is independent of the parameters.

Abstractly, we seek to minimise expectations of the form $\mathbb{E}_{\mathbf{s}\sim q}[f(\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{s}))]$, where f is expressed in a programming language. In practice, gradient based algorithms are used to address the optimisation problem for which it is essential to estimate gradients of the expectation unbiasedly. Unfortunately, [8] have demonstrated that the gradient estimation may be biased for non-differentiable f readily expressible in programming languages, which can result in incorrect results.

Contributions. We study the continuous but possibly non-differentiable setting: we provide categorical models, prove unbiasedness of the reparameterisation gradient estimator and demonstrate how to establish continuity in a language with conditionals compositionally. Abstractly, this provides a foundation for *fast yet correct* inference for non-differentiable continuous models.

Example 1. We model a temperature regulation system using a probabilistic program. Without intervention, the temperature fluctuates randomly. If the temperature drops below a threshold of 18 degrees centigrade the heating is engaged and the power is proportional to the deviation from the threshold. Time is discretised and after one time unit we measure a temperature of 21 degrees. We are interested in the distribution of the original temperature.

```
let  t0 = sample normal(20,σ0)
     mu = t0 + if t0 < 18 then c * (18−t0)
                              else 0
     observe 21 from normal(mu, σ)
in t0
```

($\sigma_0, \sigma, c > 0$ are constants.) Note that the joint density is not differentiable (yet continuous) at $t_0 = 18$.

2 An Idealised Programming Language

As our starting point we take a variant of the simply-typed lambda calculus with reals, primitive operations, conditionals, and statistical constructs for sampling and observations:

$$\begin{aligned} M ::= {} & x \mid \underline{r} \mid \underline{f} \mid \lambda x.\, M \mid M\, M \\ & \mid \mathbf{sample}\ \mathcal{D}(M, \dots, M) \\ & \mid \mathbf{observe}\, M\, \mathbf{from}\, \mathcal{D}(M, \dots, M) \\ & \mid \mathbf{if}\, M < 0\, \mathbf{then}\, M\, \mathbf{else}\, M \\ & \mid \mathbf{sif}\, M < 0\, \mathbf{then}\, M\, \mathbf{else}\, M \end{aligned}$$

where $r \in \mathbb{R}$, $f : \mathbb{R}^\ell \to \mathbb{R}$ and $\mathcal{D}$ is a continuous probability distribution (potentially with parameters). For instance samples from a normal distribution with mean 20 and standard deviation $\sigma_0 \in \mathbb{R}_{>0}$ (as in the example) can be obtained by $\mathbf{sample}\ \mathcal{N}(\underline{20}, \underline{\sigma_0})$. Our language has two kinds of conditionals: **if** with a standard semantics and a smooth approximation thereof, **sif**. The latter recovers expressivity since we need to restrict the use of standard (potentially discontinuous) conditionals below to guarantee unbiasedness.

Motivated by variational inference, we are primarily interested in joint densities. Thus, we endow our language with a *denotational weight semantics*[1] $[\![(-)]\!]$, which is in the spirit of the operational versions in [1, 10]. At ground type, it assigns a *weight* to each valuation of the free variables and the trace of samples. Whilst quasi-Borel spaces [4] are a well-established categorical model for probabilistic programs, we need to capture further properties than "just" measurability.

3 A General Categorical Model

[7] use *piecewise analytic functions under analytic partitions (PAPs)* as a natural semantic framework for programs with branching. Recall that a function f is PAP if it has the form $f(\mathbf{x}) = \sum_{i=1}^{\ell} [\mathbf{x} \in U_i] \cdot f_i(\mathbf{x})$, where $U_1, \dots, U_\ell \subseteq \mathbb{R}^n$ is a partition of analytic sets and each f_i is an analytic function[2]. [9] extend it to higher-order recursive programs.

We propose a general categorical model generalising the cartesian closed category of Frölicher spaces [3, 13], replacing smoothness with arbitrary sets of functions $\mathbb{R} \to \mathbb{R}$ satisfying mild closure properties.

We wish to interpret smoothed conditionals $\mathbf{sif}\, L < 0\, \mathbf{then}\, M\, \mathbf{else}\, N$ as smoothly weighted convex combinations of the branches: $(\sigma \circ (-[\![L]\!])) \cdot [\![M]\!] + (\sigma \circ [\![L]\!]) \cdot [\![N]\!]$, where σ is a logistic sigmoid. For this to be well-defined (in particular for higher-order branches) we show how to enrich the category over vector spaces to ensure that if $\alpha : X \to \mathbb{R}$ and $f : X \to Y$ are morphisms, so is $\alpha \cdot f$ (defined pointwise).

As a special case we obtain a cartesian closed category **VectPAP** to interpret our language (including smoothed conditionals) in, and ground terms denote PAP functions (as long as the primitives are PAPs as well).

4 Unbiasedness for Continuous PAPs

Now, we return to investigating continuity. For a *continuous PAP* (CPAP) the piecewise definitions agree on the boundaries. More formally this means it has the above form and for each $\mathbf{x} \in \overline{U_i} \cap \overline{U_j}$, $f_i(\mathbf{x}) = f_j(\mathbf{x})$. CPAPs have the great advantage of avoiding bias:

[1] E.g. $[\![\mathbf{sample}\ \mathcal{N}(\,\underline{20}, \underline{\sigma_0}) + x \cdot (\mathbf{observe}\, 2\, \mathbf{from}\, \mathcal{N}(0,1))]\!](x, [s]) = \mathrm{pdf}_{\mathcal{N}}(s \mid 20, \sigma_0) \cdot \mathrm{pdf}_{\mathcal{N}}(2 \mid 0, 1)$

[2] Recall that a function $f : \mathbb{R}^n \to \mathbb{R}$ is analytic if it is infinitely differentiable and its multivariate Taylor expansion at every point $\mathbf{x}_0 \in \mathbb{R}^n$ converges pointwise to f in a neighbourhood of $\mathbf{x}_0$. A set $U \subseteq \mathbb{R}^n$ is analytic if it is the finite intersection of sets of the form $f^{-1}(-\infty, 0)$ or $f^{-1}[0, \infty)$ for analytic $f : \mathbb{R}^n \to \mathbb{R}$.

Theorem 1 (Unbiasedness)**.** *If $f \circ \boldsymbol{\phi}_{\boldsymbol{\theta}}$ is a continuous PAP with partial derivatives which are uniformly dominated by an integrable function*[3] *then*

$$\nabla_{\boldsymbol{\theta}}\, \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}[f(\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{s}))] = \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}[\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{s}))]$$

(Note that $f \circ \boldsymbol{\phi}_{\boldsymbol{\theta}}(-)$ is a.e. differentiable.) The proof makes use of the dominated convergence theorem [6, Theorem 6.28] and exploits that due to continuity, branches do not deviate significantly near boundaries of the piecewise definition.

Due to our general construction, there is a cartesian closed category **VectCPAP** capturing continuity in which we can interpret the fragment of the language without standard conditionals (but with smoothed conditionals). The conditional in the above example can be rephrased via a non-differentiable primitive as $c \cdot (\underline{\mathrm{ReLU}}\,(18 - t_0))$. Consequently, the reparameterisation gradient is unbiased for such terms. (The uniform domination premise of Theorem 1 can be guaranteed by restricting distributions to have densities which are Schwartz functions [5, 12] and primitive operations to have partial derivatives bounded by polynomials.)

5 Continuity for Terms with Conditionals

We now turn to the task of establishing continuity via a type system for the full language language including (standard) conditionals. In principle, continuity could be added as side condition of a typing rule for conditionals:

$$\frac{\begin{array}{c}\Gamma \vdash_{\mathrm{cont}} L : R \\ \Gamma \vdash_{\mathrm{cont}} M : \tau \qquad \Gamma \vdash_{\mathrm{cont}} N : \tau\end{array}}{\Gamma \vdash_{\mathrm{cont}} \mathbf{if}\, L < 0\, \mathbf{then}\, M\, \mathbf{else}\, N : \tau} \quad \begin{array}{l}\forall \gamma \in [\![\Gamma]\!].\, [\![L]\!](\gamma) = 0 \\ \to [\![M]\!](\gamma) = [\![N]\!](\gamma)\end{array}$$

However, type checking this is generally *not* tractable.

Our approach is to restrict the conditionals of the language to *affine* guards, and to use a randomised check for continuity. Affine guards are beneficial in a twofold respect: to efficiently sample from the boundary of guards [8] and to efficiently check their consistency using off-the-shelf linear arithmetic solvers [2]. This keeps the computational burden of type checking very low, whilst we can still benefit from the added expressivity due to conditionals.

In general we restrict conditionals to having affine guards, to be purely 1st-order and, for the sake of simplicity, to only use analytic primitives in the branches:

$$\begin{aligned} F ::= x \mid \underline{r} \mid \underline{f}\, F \cdots F \mid \mathbf{sif}\, F < 0\, \mathbf{then}\, F\, \mathbf{else}\, F \\ \mid \mathbf{if}\, \underline{\mathbf{a}}^T \mathbf{x} + \underline{c} < 0\, \mathbf{then}\, F\, \mathbf{else}\, F \end{aligned}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is analytic. Conditionals *can* be nested and outside of conditionals we may use abstractions, applications and (non-differentiable) PAP primitives. E.g. we can rephrase the above example using a rectified linear unit primitive as the following typable term:

$$(\lambda f.\, \underline{c} \cdot (f\, (\underline{18} - t_0)))\, (\lambda x.\, \mathbf{if}\, x < 0\, \mathbf{then}\, \underline{0}\, \mathbf{else}\, x)$$

Randomised Continuity Check

Example 2. Consider $\mathbf{if}\, x - y < 0\, \mathbf{then}\, \underline{f}\, x\, y\, \mathbf{else}\, \underline{g}\, x\, y$, where f and g are analytic primitives. For continuity we need to check that $x = y$ implies $f(x, y) = g(x, y)$. In other words for $U := \{(x, y) \mid x = y\}$ the restriction of $f - g$ to U must be constant 0. On the other hand, since the restriction of $f - g$ to U is analytic, either $(f - g)_{|U} = 0$ or $(f - g)_{|U} \neq 0$ a.e. [11] Therefore, with probability 1, $(f - g)_{|U} = 0$ if $f(x, x) = g(x, x)$ for a random x (e.g. $x \sim \mathcal{N}(0, 1)$).

In general, continuity of $\mathbf{if}\, (\underline{\mathbf{a}}^T \mathbf{x} + \underline{c}) < 0\, \mathbf{then}\, F\, \mathbf{else}\, G$ can be checked in a finitary manner by examining all (finitely many) branches (or straightline programs) in F and G.

Consistency of Branches

Some pairs of branches may be unnecessary to check (thus unnecessarily ruling out terms) because of inconsistent guards:

Example 3. The following term implements $\max\{|x|, 1\}$, which is continuous in x:

$$\mathbf{if}\, x + 1 < 0\, \mathbf{then}\, -x\, \mathbf{else}\, (\mathbf{if}\, x - 1 < 0\, \mathbf{then}\, 1\, \mathbf{else}\, x)$$

By the method presented thus far, for the outer conditional we need to check compatibilty of $-x$ with x at the boundary of $x + 1 < 0$, i.e. $x = -1$. Obviously, for $x = -1$, $-x \neq x$ and the test would fail.

However, continuity is *not* compromised because the two branches given by the conditionals $x + 1 < 0$ and $x + 1 \geq 0 \land x - 1 \geq 0$ do not share boundary points. Phrased differently, the linear arithmetic constraints $x + 1 \leq 0 \land x + 1 \geq 0 \land x - 1 \geq 0$ (note the non-strict inequality in the first conjunct) is inconsistent.

To account for this, we collect the guards and only check pairs of branches with consistent guards. Fig. 1 defines an auxiliary function for branches and their aggregated guards.

Combining all ingredients, we can employ the following randomised check for continuity of a term $\mathbf{if}\, (\underline{\mathbf{a}}^T \mathbf{x} + \underline{c}) < 0\, \mathbf{then}\, F\, \mathbf{else}\, G$ (assuming w.l.o.g. $a_1 \neq 0$):

> For all *consistent* branches[a] (or straightline programs) S and T in F and G, respectively, it must hold
>
> $$[\![S]\!] \left(-\frac{\mathbf{a}_{-1}^T \mathbf{x}_{-1} + c}{a_1}, \mathbf{x}_{-1}\right) = [\![T]\!] \left(-\frac{\mathbf{a}_{-1}^T \mathbf{x}_{-1} + c}{a_1}, \mathbf{x}_{-1}\right)$$
>
> where $\mathbf{x}_{-1} = (x_2, \ldots, x_n)$ is sampled e.g. from the multivariate standard normal.
>
> [a] formally: $(S, \psi) \in \mathrm{br}(F)$ and $(T, \chi) \in \mathrm{br}(G)$ such that for some assignment α, $\alpha \models \chi \land \psi$

This check is sound (i.e. admits only continuous terms, typable via $\vdash_{\mathrm{cont}}$) with probability 1. Crucially, consistency can be checked very efficiently using off-the-shelf linear arithmetic solvers [2].

6 Concluding Remarks

We have demonstrated that the reparameterisation gradient estimator, which is usually superior to other estimators, can be safely applied to continuous but possibly non-differentiable programs. We have studied categorical models, which are useful for establishing continuity for programs without branching. We have also presented a randomised method to efficiently establish continuity in the presence of standard conditionals.

References

[1] Johannes Borgström, Ugo Dal Lago, Andrew D. Gordon, and Marcin Szymczak. A lambda-calculus foundation for universal probabilistic programming. In

[3] i.e. there exists $g : \mathbb{R}^n \to \mathbb{R}$ such that $\mathbb{E}[|g(\mathbf{s})|] < \infty$ and $\frac{\partial}{\partial \theta_i} f(\boldsymbol{\phi}_{\boldsymbol{\theta}}(\mathbf{s})) \leq g(\mathbf{s})$ for all $\boldsymbol{\theta} \in \boldsymbol{\Theta}$ and $\mathbf{s} \in \mathbb{R}^n$.

$$
\begin{gathered}
\mathrm{br}(x) \coloneqq \{(x, \top)\} \qquad \mathrm{br}(\underline{r}) \coloneqq \{(\underline{r}, \top)\} \\
\mathrm{br}(\underline{f}\, F_1 \cdots F_n) \coloneqq \{(\underline{f}\, S_1 \cdots S_n, \psi_1 \wedge \cdots \psi_n) \mid (S_1, \psi_1) \in \mathrm{br}(F_1), \ldots, (S_n, \psi_n) \in \mathrm{br}(F_n)\} \\
\mathrm{br}(\mathbf{sif}\, F < 0 \,\mathbf{then}\, G \,\mathbf{else}\, H) \coloneqq \{(\mathbf{sif}\, S < 0 \,\mathbf{then}\, T \,\mathbf{else}\, U, \psi_1 \wedge \psi_2 \wedge \psi_3) \mid (S, \psi_1) \in \mathrm{br}(F), (T, \psi_2) \in \mathrm{br}(G), (U, \psi_3) \in \mathrm{br}(H)\} \\
\mathrm{br}(\mathbf{if}\, \underline{\mathbf{a}}^T \mathbf{x} + \underline{c} < 0 \,\mathbf{then}\, F \,\mathbf{else}\, G) \coloneqq \{(S, \psi \wedge \mathbf{a}^T \mathbf{x} + c \leq 0) \mid (S, \psi) \in \mathrm{br}(F)\} \cup \{(T, \psi \wedge \mathbf{a}^T \mathbf{x} + c \geq 0) \mid (T, \psi) \in \mathrm{br}(G)\}
\end{gathered}
$$

Figure 1: Auxiliary function capturing branches and aggregated guards.

Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, ICFP 2016, Nara, Japan, September 18-22, 2016, pages 33–46, 2016.

[2] Bruno Dutertre and Leonardo de Moura. A fast linear-arithmetic solver for dpll(t). In Thomas Ball and Robert B. Jones, editors, *Computer Aided Verification*, pages 81–94, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[3] A. Frölicher and A. Kriegl. *Linear Spaces and Differentiation Theory.* Interscience, J. Wiley and Son, New York, 1988.

[4] Chris Heunen, Ohad Kammar, Sam Staton, and Hongseok Yang. A convenient category for higher-order probability theory. *Proc. Symposium Logic in Computer Science*, 2017.

[5] L. Hörmander. *The Analysis of Linear Partial Differential Operators I: Distribution Theory and Fourier Analysis.* Classics in Mathematics. Springer Berlin Heidelberg, 2015.

[6] Achim Klenke. *Probability Theory: A Comprehensive Course.* Universitext. Springer London, 2014.

[7] Wonyeol Lee, Hangyeol Yu, Xavier Rival, and Hongseok Yang. On correctness of automatic differentiation for non-differentiable functions. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[8] Wonyeol Lee, Hangyeol Yu, and Hongseok Yang. Reparameterization gradient for non-differentiable models. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 5558–5568, 2018.

[9] Alexander K. Lew, Mathieu Huot, and Vikash K. Mansinghka. Towards denotational semantics of AD for higher-order, recursive, probabilistic languages. *CoRR*, abs/2111.15456, 2021.

[10] Carol Mak, C.-H. Luke Ong, Hugo Paquet, and Dominik Wagner. Densities of almost surely terminating probabilistic programs are differentiable almost everywhere. In Nobuko Yoshida, editor, *Programming Languages and Systems - 30th European Symposium on Programming, ESOP 2021, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2021, Luxembourg City, Luxembourg, March 27 - April 1, 2021, Proceedings*, volume 12648 of *Lecture Notes in Computer Science*, pages 432–461. Springer, 2021.

[11] Boris Mityagin. The zero set of a real analytic function, 2015.

[12] M. Reed and B. Simon. *Methods of Modern Mathematical Physics: Functional analysis. I.* World Published Corporation, 2003.

[13] Andrew Stacey. Comparative smootheology. *Theory and Applications of Categories*, 25(4):64–117, 2011.