Optimal Smoothing for Pathwise Adjoints

Jonathan Hüser, Shih-Te Yang, Uwe Naumann
Department of Computer Science
RWTH Aachen University
D-52056, Aachen, Germany
[hueser,naumann]@stce.rwth-aachen.de,
shih-te.yang@rwth-aachen.de

Abstract

We propose an optimal smoothing parametrization for gradient estimators of expectations of discontinuous functions. The reparametrization trick with discontinuous functions gives gradient estimators for discrete random variables [5, 8, 13] and makes smoothing applicable in the machine learning context (e.g. variational inference and stochastic neural networks [11, 1, 12, 6]). Our approach is based on an objective that can be solved simultaneously with a primal optimization task. Optimal smoothing is general purpose in the sense that it only requires an extension of the algorithmic differentiation tool without the need to rearrange the model.

1 Introduction

We consider the Monte Carlo estimation of the gradient of an expectation via adjoint algorithmic differentiation (AD [3, 9]) (also called backpropagation). For a continuously differentiable function $f : D \subseteq \mathbb{R}^{n_x} \times \mathbb{R}^{n_\theta} \to \mathbb{R}$ the gradient of the expectation can be estimated as

$$\nabla_\theta \mathbb{E}_{x\sim\mathcal{P}}[f(x,\theta)] = \nabla_\theta \int_{\mathbb{R}^{n_x}} f(x,\theta)\mathcal{P}(x)\mathrm{d}^{n_x}x \quad \approx \nabla_\theta \frac{1}{N}\sum_{i=1}^{N} f(x^i,\theta) = \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta f(x^i,\theta)$$

where x^i for $i = 1,\ldots,N$ is a Monte Carlo sample. If the gradients $\nabla_\theta f(x^i,\theta)$ are computed by adjoint AD we call the above estimator the pathwise adjoint [2].

For a class of piecewise continuously differentiable but discontinuous functions f the expectation $\mathbb{E}_{x\sim\mathcal{P}}[f(x,\theta)]$ is differentiable if $\mathcal{P}$ is sufficiently smooth. But the pathwise adjoint cannot be used as an estimator for the gradient because sensitivity with regard to the discontinuities is not captured by the point-wise gradients. In the following we present an approach to computing pathwise adjoints by smoothing.

We assume that discontinuity results from the composition of continuously differentiable elementary functions together with

$$\mathrm{step}(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} .$$

Backpropagation gradients of Monte Carlo estimators occur widely in machine learning applications for example in variational inference and stochastic neural networks [11, 1, 12, 6]. The step function together with a continuous random variable $x \sim \mathcal{P}$ can be used to model a discrete random variable, e.g. we get $y \sim \text{Bernoulli}(p(\theta))$ by drawing $x \sim \text{Uniform}(0,1)$ and setting $y = \mathrm{step}(p(\theta) - x)$. Pathwise adjoints are also commonly used in computational finance to compute the parameter sensitivities (Greeks) for option hedging [2, 10]. The step function is used to model the payoff structure of digital and barrier options [7]. Nonsmooth continuous functions can also be written as

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

e.g. $\max(x, y) = \text{step}(x - y)(x - y) + y$ and applications of smoothing pathwise adjoints extends to nonsmooth and nonconvex optimization by smoothing.

To illustrate the challenge of pathwise adjoints for discontinuous functions we consider the example of $f(x, \theta) = \text{step}(\theta - x)$ with

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}\theta}\mathbb{E}_{x\sim\mathcal{P}}[\text{step}(\theta - x)] &= \frac{\mathrm{d}}{\mathrm{d}\theta}\int_{\mathbb{R}} \text{step}(\theta - x)\mathcal{P}(x)\mathrm{d}x = \int_{\mathbb{R}} \frac{\mathrm{d}}{\mathrm{d}\theta}\,\text{step}(\theta - x)\mathcal{P}(x)\mathrm{d}x \\ &= \int_{\mathbb{R}} \delta(\theta - x)\mathcal{P}(x)\mathrm{d}x = \mathcal{P}(\theta)\end{aligned}$$

where δ is the Dirac delta. The pathwise derivative computes a wrong value

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\mathbb{E}_{x\sim\mathcal{P}}[\text{step}(\theta - x)] \overset{?}{\approx} \frac{1}{N}\sum_{i=1}^{N} \frac{\mathrm{d}}{\mathrm{d}\theta}\,\text{step}(\theta - x^i) = 0 \neq \mathcal{P}(\theta)$$

with probability 1.

Smoothing the step function allows for biased pathwise adjoints and smoothing parameters should be chosen such that the contribution to error for multiple discontinuities is balanced.

We formulate an objective for the hyper parameter optimization problem of optimal smoothing such that the stochastic optimization objective can be estimated with similar variance as the gradient itself (Section 2). We present how stochastic gradient descent algorithm can be used to solve the optimal smoothing problem at the same time as the optimization that uses the pathwise adjoints (e.g. learning a stochastic neural network) (Section 3.1). The stochastic gradient of the hyper parameter problem is implemented using a second-order adjoint model. We present numerical results from an applications in computational finance (Section 3.2).

2 Optimal Smoothing

In this section we motivate a smoothing approach and propose a feasible optimal smoothing objective.

In a distributions sense the derivative of step is the Dirac delta δ but that is not directly useful in finite numerical computation. The Dirac delta can be approximated by a kernel function such as the Gaussian probability density function $\delta(x) \approx \mathcal{N}(x|0, h)$ where h is the kernel bandwidth. If we replace every step in f by

$$\text{smooth\_step}_{h_i}(x) = \Phi(x/h_i)$$

where the $i = 1, \ldots, n_h$ enumerate the occurrences of step and Φ is the cumulative normal density function then we obtain a continuously differentiable approximation $\tilde{f}_h(x, \theta) \approx f(x, \theta)$. The derivative of smooth_step is the Gaussian kernel.

We have now introduced the additional hyper parameter vector h that should be chosen to make the smoothing optimal in some sense. Optimality can be defined as minimizing the mean squared error of the smoothed gradient estimator, i.e.

$$h^* = \arg\min_h \; \mathbb{E}_{x^1,\ldots,x^N\sim\mathcal{P}}[(\nabla_\theta \mathbb{E}_{y\sim\mathcal{P}}[f(y, \theta)] - \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \tilde{f}_h(x^i, \theta))^2] \quad .$$

Because the step function can generally occur in a nested manner and at different scales in the function the naive approach of letting $h_1 = h_2 = \cdots = h_{n_h}$ cannot be assumed to give optimal bias variance trade-off. Mean squared error can be decomposed [4] into squared bias

$$(\nabla_\theta \mathbb{E}_{y\sim\mathcal{P}}[f(y, \theta)] - \mathbb{E}_{x^1,\ldots,x^N\sim\mathcal{P}}[\frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \tilde{f}_h(x^i, \theta)])^2$$

plus variance

$$\mathbb{E}_{x^1,\ldots,x^N\sim\mathcal{P}}[(\frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \tilde{f}_h(x^i, \theta) - \mathbb{E}_{y^1,\ldots,y^N}[\frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \tilde{f}_h(y^i, \theta)])^2] \approx$$

$$\frac{1}{N(N-1)}\sum_{j=1}^{N}(\nabla_\theta \tilde{f}_h(x^j, \theta) - \frac{1}{N}\sum_{i=1}^{N} \nabla_\theta \tilde{f}_h(x^i, \theta))^2 \quad .$$

The bias cannot be estimated because it requires the expectation $\mathbb{E}_{y\sim\mathcal{P}}[f(y,\theta)]$ we are trying to compute in the first place. The variance estimator also suffers from high variance if the variance of the gradient estimator is high.

Instead of trying to minimize mean squared error directly we propose a choice of smoothing parameters h that balances their contribution to squared bias $(f(x,\theta) - \tilde{f}_h(x,\theta))^2$ in the original function. The proposed objective is to minimize the norm of the variance of the gradient estimator while bounding the expectation of the squared bias in the original function by a constant C

$$h^* = \arg\min_h \|\mathbb{E}_{x\sim\mathcal{P}}[(\nabla_\theta \tilde{f}_h(x,\theta) - \mathbb{E}_{y\sim\mathcal{P}}[\nabla_\theta \tilde{f}_h(y,\theta)])^2]\|_2^2$$
$$\text{s.t. } \mathbb{E}_{x\sim\mathcal{P}}[(f(x,\theta) - \tilde{f}_h(x,\theta))^2] < C \text{ and } h > 0 \quad .$$

Using Monte Carlo estimators and a penalty method for the first inequality constraint we obtain the following stochastic objective for h to minimize

$$g(h, x^1, \ldots, x^N) = \|\frac{1}{N-1}\sum_{i=1}^{N}(\nabla_\theta \tilde{f}_h(x^i,\theta) - \frac{1}{N}\sum_{j=1}^{N}\nabla_\theta \tilde{f}_h(x^j,\theta))^2\|_2^2 +$$
$$\lambda \max(0, \frac{1}{N}\sum_{i=1}^{N}(f(x^i,\theta) - \tilde{f}_h(x^i,\theta))^2 - C)$$

hence

$$h^* = \arg\min_h \mathbb{E}_{x^1,\ldots,x^N\sim\mathcal{P}}[g(h, x^1, \ldots, x^N)]$$

where λ is the parameter that controls the strictness of the inequality constraint. For now we assume that a numerical optimizer can be parameterized such that h stays positive. In some applications additional regularization is required to prevent the variance estimator from pushing the smoothing parameters towards 0.

3 Numerical Experiments

In this section we first give a short overview of an algorithm for the optimal smoothing problem. We then present some numerical results using an application from computational finance. At the workshop we will present results from optimal smoothing in the optimization of a stochastic neural network with discrete random variables.

3.1 Implementation of Stochastic Gradient Descent

If the smoothed gradients $\nabla_\theta f_h(x,\theta)$ are computed by adjoint AD then we can compute $\nabla_h g(h, x^1, \ldots, x^N)$ using an adjoint-over-adjoint model. The second-order adjoint model is an adjoint code pattern that can potentially be implemented more efficiently by using an tangent-over-adjoint model. This aspect of algorithmic second-order adjoint model implementation will be highlighted in more detail at the workshop.

The pathwise adjoint estimator of $\nabla_h \mathbb{E}_{x^1,\ldots,x^N\sim\mathcal{P}}[g(h, x^1, \ldots, x^N)]$ can be used in a stochastic gradient descent algorithm. If the gradient estimator $\nabla_\theta \mathbb{E}_{x\sim\mathcal{P}}[f_h(x,\theta)]$ is already used in the context of a gradient-based optimization for θ then we can optimize h and θ simultaneously for example by Algorithm 1.

3.2 Barrier Option Delta

We consider the computation of Delta for a barrier option, i.e. the sensitivity of the option price with respect to initial price of the underlying. The barrier option payoff is defined as 0 if the price of the underlying goes above B at any point in time and otherwise it is $\max(0, S^T - K)$ where K is the strike price and S^T is the price of the underlying at the exercise date T.

Option price is estimated using the Euler-Maruyama discretization of a geometric Brownian motion $\mathrm{d}S = rS\mathrm{d}t + \sigma S\mathrm{d}W$ where r models a risk-free rate and σ is the volatility of the underlying. The discrete time evolution with time step Δt is

$$S^{i+1} = S^i + rS^i\Delta t + \sigma S^i \Delta W^i$$

Algorithm 1 Smoothing Stochastic Gradient Descent

Initialize $h_i^0 = 1$ for all $i = 1, \ldots, n_h$, $k = 0$, θ^0 as you would.
while θ^k not optimal **do**
  Sample $x^1, \ldots, x^N \sim \mathcal{P}$.
  Register h^k in inner adjoint tape.
  Register θ^k in outer adjoint tape.
  Evaluate $\tilde{f}_h(x^i, \theta^k)$ for all $i = 1, \ldots, N$.
  Interpret outer adjoint tape to obtain $\nabla_\theta \tilde{f}_h(x^i, \theta^k)$ for all $i = 1, \ldots, N$.
  Evaluate $f(x^i, \theta^k)$ for all $i = 1, \ldots, N$.
  Evaluate $g(h^k, x^1, \ldots, x^N)$ using the previously computed values.
  Interpret the inner adjoint tape to obtain $\nabla_h g(h^k, x^1, \ldots, x^N)$.
  Let $\theta^{k+1} \coloneqq \theta^k - \alpha^k \frac{1}{N} \sum_{i=1}^N \nabla_\theta \tilde{f}_h(x^i, \theta^k)$
  Let $h^{k+1} \coloneqq h^k - \beta^k \nabla_h g(h^k, x^1, \ldots, x^N)$
  Let $k \coloneqq k + 1$
end while

where $\Delta W^i \sim \mathcal{N}(0, \sqrt{\Delta t})$ for all $i = 1, \ldots, (n_T - 1)$ and $\Delta t = T/(n_T - 1)$. The option price is the expectation of the discounted payoff that is estimated as

$$\mathbb{E}_{\Delta W^1, \ldots, \Delta W^{(n_T-1)} \sim \mathcal{N}(0, \sqrt{\Delta t})}[P(S^1, r, \sigma, B, K, \Delta W^1, \ldots, \Delta W^{(n_T-1)})] \approx$$
$$\frac{1}{N} \sum_{i=1}^N P(S^1, r, \sigma, B, K, (\Delta W^1)^i, \ldots (\Delta W^{(n_T-1)})^i)$$

with

$$P(S^1, r, \sigma, B, K, (\Delta W^1)^i, \ldots (\Delta W^{(n_T-1)})^i) =$$
$$\mathrm{step}(B - S^1) \cdot \ \ldots \ \cdot \mathrm{step}(B - S^{n_T}) \cdot \max(0, S^{n_T} - K) \cdot \exp(-rT) \quad .$$

The sensitivity we want to estimate using a pathwise derivative is

$$\frac{\mathrm{d}}{\mathrm{d}S^1} \mathbb{E}_{\Delta W^1, \ldots, \Delta W^{(n_T-1)} \sim \mathcal{N}(0, \sqrt{\Delta t})}[P(S^1, r, \sigma, B, K, \Delta W^1, \ldots, \Delta W^{(n_T-1)})] \quad .$$

From this reduced example the necessity of an adjoint model is not immediately clear but some finance models have many more parameters and in general it is desirable to compute a large number of Greeks (sensitivities) at the same time.

If we model $\max$ by step as mentioned in the introduction then the payoff contains the step function $n_h = n_T + 1$ times. We initialize their smoothing parameters $h_1, \ldots, h_{n_h} = 0.2$ resulting in relatively large bias initially but giving lower variance to the optimal smoothing objective. Figure 1 shows the evolution of h for $N = 10000, n_T = 10, S^1 = 100, B = 110, K = 90, r = 0.05, \sigma = 0.1, T = 1, C = 0.01, \lambda = 500$ using a stochastic gradient descent algorithm with constant step size. The figure also shows the evolution of estimated variance of the derivative estimator and the estimated expected bias in the original payoff function.

4 Conclusion

We have presented a general purpose approach to the choice of smoothing parameters to get good bias variance trade-off for pathwise adjoint gradient estimators. The stochastic convergence of the proposed optimal smoothing algorithm was demonstrated in a numerical experiment on an application from computational finance. At the workshop we will present results with a machine learning specific application of stochastic neural networks and go into more detail on the theoretical aspects and issues of implementation using algorithmic differentiation tools.

References

[1] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

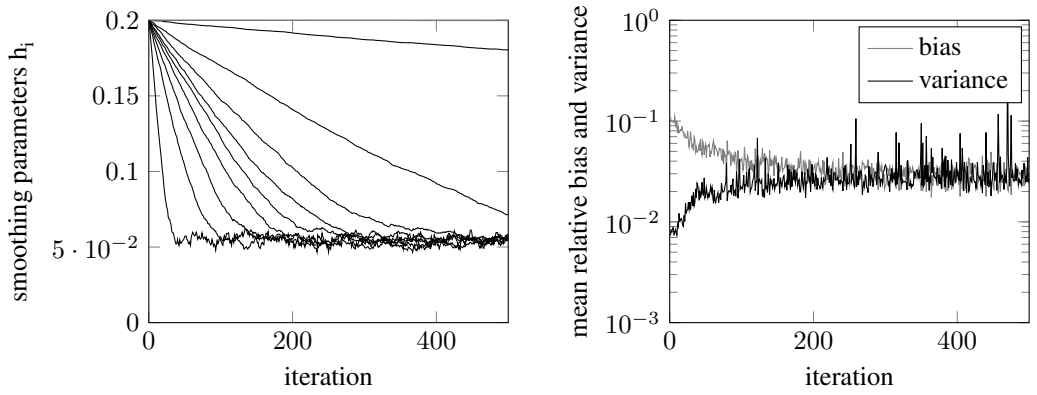

Figure 1: Optimal smoothing for barrier option Delta

[2] Mike Giles and Paul Glasserman. Smoking adjoints: Fast monte carlo greeks. *Risk*, 19(1):88–92, 2006.

[3] Andreas Griewank and Andrea Walther. *Evaluating derivatives: principles and techniques of algorithmic differentiation*. SIAM, 2008.

[4] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An introduction to statistical learning*, volume 112. Springer, 2013.

[5] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[6] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 18(14):1–45, 2017.

[7] Guangwu Liu and L Jeff Hong. Kernel estimation of the greeks for options with discontinuous payoffs. *Operations Research*, 59(1):96–108, 2011.

[8] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

[9] Uwe Naumann. *The art of differentiating computer programs: an introduction to algorithmic differentiation*. SIAM, 2011.

[10] Uwe Naumann and Jacques du Toit. Adjoint algorithmic differentiation tool support for typical numerical patterns in computational finance. Technical report, Numerical Algorithms Group. https://www.nag.co.uk/doc/techrep/pdf/tr3_14.pdf, 2014. To appear in Journal of Computational Finance.

[11] Tapani Raiko, Mathias Berglund, Guillaume Alain, and Laurent Dinh. Techniques for learning binary stochastic feedforward neural networks. *arXiv preprint arXiv:1406.2989*, 2014.

[12] John Schulman, Nicolas Heess, Theophane Weber, and Pieter Abbeel. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems*, pages 3528–3536, 2015.

[13] George Tucker, Andriy Mnih, Chris J Maddison, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *arXiv preprint arXiv:1703.07370*, 2017.