Depth First Learning: SVGD

Week 4: Stein Variational Gradient Descent

June 7, 2019

Stein Variational Gradient Descent (SVGD) is a popular, non-parametric Bayesian Inference algorithm thats been applied to Variational Inference, Reinforcement Learning, GANs, and much more. This week, we study the algorithm in its entirety, building off of last weeks work on KSD, and seeing how viewing KSD from a KL-Divergence-minimization lens induces a powerful, practical algorithm. We discuss the benefits of SVGD over other similar approximators, and look at a practical implementation of the algorithm.

1 Table of Contents

2 Variational Inference

Last week, we saw an efficient test to test if two distributions, p and q, were the same in *Kernelized Stein Discrepancy*. This week, we're looking into subsequent step: given some ground truth distribution p, and some initial representation of q, how can we *transform* q into p?

This week, we'll be studying the Stein Variational Gradient Descent paper, which casts the above problem in the framework of variational inference: given some p, and some parameterized family of (simpler) distributions $\mathcal{Q}$, how do we find some $q^* \in \mathcal{Q}$ that approximates p? In this sense, variational inference casts the problem as an optimization of some "cost" function $\mathcal{J}(q)$, which is a function of the chosen distribution.

Alternative methods to approximate p, most commonly sampling-based variants of MCMC, provide strong guarantees in infinite time, but not many in the finite sample regime. While still an

exciting research area, there are some issues with scaling these methods up, and accelerating them with common hardware like GPUs, both of which are positive charateristics of variational inference.

2.1 The KL-Divergence

In order to frame the variational inference problem as an optimization problem, we need a method of comparing our two distributions. For this, we turn to the KL-Divergence. This function, formally defined in the equation below, measures differences in information content between two distributions.

$$KL(q||p) = \mathbf{E}_q[\log q(x) - \log p(x)] \tag{1}$$

When working with some unnormalized distribution $\bar{p}$, the optimization, which is searching for some distribution $q^* \in \mathcal{Q}$, becomes:

$$KL(q||p) = \arg\min_{q \in \mathcal{Q}} \{\mathbf{E}_q[\log q(x)] - \mathbf{E}_q[\log \bar{p}(x)] + logZ\} \tag{2}$$

The normalization constant Z drops out of the optimization here. It's important to note that the KL Divergence is a *divergence*, not a distance: this means, in general, $KL(q||p) \neq KL(p||q)$. In fact, working out the gradients for each direction gives different benefits (and issues), but in general, we focus on the former as the latter requires evaluation on the normalization constant of p. Optimizing the equation above will - in theory; in practice, they will almost never find the globally optimal solution - find a distribution q^* that minimizes the KL Divergence.

3 Variational Inference Using Smooth Transforms

Of course, after choosing to optimize the KL Divergence, we still have to choose the family of distributions. Choices often center around parameterized functions classes, or normal distributions, but for SVGD, we are going to focus on the distributions obtained by *smooth transforms* from a tractable reference distribution.

$\mathcal{Q}$ takes the form of $z = \mathbf{T}(x)$, where $\mathbf{T} : \mathcal{X} \to \mathcal{X}$ is a smooth one-to-one transform, with $x \sim q_0(x)$. It can be shown (from optimal transport theory) that these are expressive distributions, that in theory can approximate almost any other distribution.

While we've constrained our distribution family, in practice, we *also* need to worry about the class of transforms: balancing accuracy, tractability, and computational constraints, along with the injectiveness required by $\mathbf{T}$ provide a lot of requirements, many of which cannot be achieved in parallel. In the following two sections, we'll see the choices that SVGD makes to achieve these constraints.

4 Stein Operators and KL Divergence

The authors describe a small perturbation of the identity map: $\mathbf{T} = x + \epsilon\phi(x)$, which, given a small enough ϵ, guarantees $\mathbf{T}$ to be a injective map.

Note: Theorem 3.1 and Lemma 3.2 will be discussed qualitatively (with focus on their implications), as the paper's appendix does a good job of walking through the steps.

Theorem 3.1 relates the derivative of KLD to the negative trace of the Stein Operator, and because of Kernelized Stein Discrepancy, we can relate $\phi^*_{q,p}$ as the optimal perturbation direction (i.e steepest descent on KL) in the RKHS. Lemma 3.2 shows us that this generates a iterative procedure that allows us to start with *any* q_0 that will, under the smooth transform assumptions above, converge to ground truth distribution p. In short, we apply iterative transforms, $\mathbf{T}^*_i = x + \epsilon_i \cdot \phi^*_{q_i,p}(x)$, to each distribution q_i (starting from tractable q_0), until we converge ($\phi^*_{q_\infty,p} = 0$, which means the transform becomes the identity map).

The paper includes an interpretation from a functional gradient perspective, but as that won't be important until next week, we skip directly to the practical algorithm proposed.

5 Stein Variational Gradient Descent

Now, we're ready to set the stage for SVGD. From the paper, we know that the optimal direction is:

$$\phi^*_{q,p}(\cdot) = \mathbf{E}_{x\sim q}[k(x,\cdot)\nabla_x \log p(x) + \nabla_x k(x,\cdot)] \tag{3}$$

To approximate the expectation, we average over an initial set of particles and use an empirical version of the transform (averaged over the particles) to move our particles to match our target distribution $p(x)$.

The entire algorithm can be condensed into a single line, which repeats until convergence given an initial set of n particles $\{x_i\}$ and target distribution $p(x)$.

$$\hat{\phi}^*(x_i) = \frac{1}{n}\sum_{j=1}^{n}[k(x_j,x_i)\nabla_{x_j} \log p(x_j) + \nabla_{x_j} k(x_j,x_i)] \tag{4}$$

This is calculated individually for each particle, and then used with some step size to "move" the particle.

5.1 Benefits of SVGD

1. With more particles (a larger n), the approximation for each q_i gets better.

2. With a single particle, this reduces into a MAP estimate.

3. SVGD, unlike other methods, is expressive without requiring the inversion of a Jacobian matrix (a very expensive operation).

4. The two terms in the above equation help introduce accuracy - the first term maximizes log probability of the samples - and diversity - the second term repulses similar particles - into the approximation.

5. With automatic differention packages, many parts of SVGD can be parallelized and offloaded to efficient hardware implementations.

6. The final form of the update is incredibly similar to standard gradient descent, making implementation and understanding extremely easy.

7. (A pro, but not from this paper) SVGD has strong guarantees for particular settings, and can be seen from various different lenses (one of which we'll see next week).

5.2 Implementations

Here are some implementations of SVGD in:

1. Matlab and Python (original authors)

2. Autograd (Sanyam Kapoor)

3. Pytorch (Calvin Woo)

4. Jax (Bhairav Mehta)

References

[1] Y. Feng, D. Wang, and Q. Liu. Learning to draw samples with amortized stein variational gradient descent, 2017.

[2] T. Gangwani, Q. Liu, and J. Peng. Learning self-imitating diverse policies, 2018.

[3] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies, 2017.

[4] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[5] Y. Liu, P. Ramachandran, Q. Liu, and J. Peng. Stein variational policy gradient, 2017.

[6] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. Active domain randomization, 2019.

[7] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning, 2016.

[8] J. Oh, Y. Guo, S. Singh, and H. Lee. Self-imitation learning, 2018.

[9] R. Ranganath, J. Altosaar, D. Tran, and D. M. Blei. Operator variational inference, 2016.

[10] J. Schulman, X. Chen, and P. Abbeel. Equivalence between policy gradients and soft q-learning, 2017.

[11] H. Tang and T. Haarnoja. Soft q learning. *https://bair.berkeley.edu/blog*, 2017.

[12] L. Weng. Policy gradient algorithms. *lilianweng.github.io/lil-log*, 2018.

[13] R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(34):229256, May 1992.

[14] B. D. Ziebart, A. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI08, page 14331438. AAAI Press, 2008.