Re-using gradient computations in automatic variational inference

Joseph Sakaya
Department of Computer Science
University of Helsinki, Finland
joseph.sakaya@cs.helsinki.fi

Arto Klami
Department of Computer Science
University of Helsinki, Finland
arto.klami@cs.helsinki.fi

Abstract

Automatic variational inference has recently become feasible as a scalable inference tool for probabilistic programming. The state-of-the-art algorithms are stochastic in two respects: they use stochastic gradient descent to optimize an expectation that is estimated with stochastic approximation. The core computation of such algorithms involves evaluating the loss and its automatically differentiated gradient for random parameters sampled from the approximation. We study ways of re-using some of the gradient computations during optimization, to speed up learning for large-scale applications. We present a stochastic average gradient algorithm that uses gradients computed for past mini-batches to reduce noise, and explain how importance sampling allows re-using gradients computed for data samples visited again during the optimization loop.

1 Introduction

Variational approximation has long promised a scalable alternative to MCMC for approximate posterior analysis, but the learning task that involves optimizing an expectation over high-dimensional parameter space has required extensive model-specific derivations and often resorting to conjugate priors. This has recently changed with the introduction of algorithms that use stochastic estimates of the loss and its automatically differentiated gradients.

The learning goal in variational inference is provided by the evidence lower bound (ELBO)

$$\mathcal{L}(x, \lambda) = \int q(\theta|\lambda) \log \frac{p(x, \theta)}{q(\theta|\lambda)} d\theta, \tag{1}$$

which is maximized over the parameters λ of the approximation $q(\theta|\lambda)$. The key finding enabling the use of variational inference for generic models is that we can compute stochastic estimates of the gradients of (1) using either the reparametrization method [2, 10, 12] or the log-derivative trick [8]. The former requires a differentiable probability model, whereas for the latter it is sufficient to use a differentiable approximation. The reparametrization method generally results in considerably more efficient algorithms. We build here on the reparametrization method, but note that the basic idea of re-using gradient computations could be useful also for algorithms using the log-derivative trick.

The standard reparametrization algorithm uses gradients of the form

$$\nabla_\lambda \mathcal{L}(x, \lambda) \approx \frac{1}{M} \sum_{m=1}^{M} \nabla_\theta \log p(x, \theta_m) \nabla_\lambda f(z_m, \lambda), \tag{2}$$

where θ_m are drawn from the approximating distribution $q(\theta|\lambda)$ using the reparametrization $\theta_m = f(z_m, \lambda)$ and z_m are draws from some standard distribution that does not depend on λ. The notation $\nabla_\theta \log p(x, \theta_m)$ is a shorthand for the gradient w.r.t. θ evaluated at θ_m. The key observation is that the gradient w.r.t. the parameters λ of the approximation can be computed using the gradients w.r.t. the parameters θ using the chain rule.

Most of the practical algorithms [2, 3, 12] use normal approximations, possibly combined with transformations for the model parameters. This makes the reparametrization simple location-scale transformation $f(z, \{\mu, C\}) = \mu + Cz$, where z are drawn from the standard normal distribution. The main research focus in automatic variation inference has lately been in generalizations suitable for other distributions: Maddison et al. [4] presented a differentiable reparametrization for categorical variables, Ruiz et al. [9] provided approximative reparametrization by drawing samples from a distribution that is almost independent of λ, and Naesseth et al. [6] provided even more flexible and accurate technique based on rejection sampling.

The optimization problem itself is typically solved by straightforward stochastic gradient descent (SGD). Practically all of the computation for such an algorithm goes into evaluating the expected logarithmic probability for each data point in a mini-batch for samples θ_m drawn from the approximation, and for evaluating the gradient for each of those. In this work we consider alternatives to SGD, and in particular look into ways of re-using already performed gradient computations during the optimization loop to speed up the overall convergence. First we apply the concept of stochastic average gradients (SAG) [11] for automatic variational inference. At the expense of some storage space, SAG enables emulating full-batch gradients with the computational cost of stochastic gradients by storing the gradients computed for each mini-batch and using their mean for updates.

We then consider techniques to re-use computations performed the last time a particular mini-batch was seen. As shown in (2), the full gradient is obtained by combining two separate terms. The first term, the gradient of the logarithmic probability with respect to the model parameters, does not depend on the approximation and hence need not be computed again. However, it has been evaluated at θ_m drawn from some old approximation. We present an importance sampling technique to correct for that, resulting in speedup that skips the most time-consuming part of the gradient estimate at the expense of higher variance.

2 Background: Automatic variational inference

Here we briefly recap the standard procedure for gradient-based learning of variational approximations for arbitrary differentiable probability models, following roughly the presentation by Titsias and Lázaro-Gredilla [12] and Kucukelbir et al. [3]. Given a model $p(x, \theta)$ defined over some parameters θ, the goal is to learn an approximation $q(\theta|\lambda)$ for the posterior distribution $p(\theta|x)$ by optimizing a lower bound (1) for the marginal likelihood $p(x)$. This requires both specifying the family of the approximation $q(\theta|\lambda)$ and finding the optimal parameters λ.

A practical choice for approximating arbitrary differentiable models is to posit a normal distribution $q(\hat{\theta}|\lambda) = N(\hat{\theta}|\mu, CC^T)$ over parameters $\hat{\theta}$ transformed to lie in the full real line, proposed by Kucukelbir et al. [3] as a general procedure. This choice combines easy reparametrization with flexibility of modeling also constrained parameters. For the following presentation we do not explicitly present the transformations, but present the inference algorithm for the special case that approximates the posterior over the parameters θ directly with a normal distribution. This corresponds to the Gaussian variational approximation [7, 12] and can be extended to the transformed case.

The reparametrization for the Gaussian approximation is done by writing $\theta = Cz + \mu$, where C is a cholesky factor, and using $q(\theta|\lambda) = \frac{1}{|C|}\phi(C^{-1}(\theta - \mu))$ as the approximation, where $\phi(\cdot)$ is the standard normal distribution. Given this choice, a change of variables simplifies the ELBO into

$$\mathcal{L}(x, \mu, C) = \int \phi(z) \log \frac{p(x, Cz + \mu)|C|}{\phi(z)} \delta z$$

where the integral is over the standard normal distribution that does not depend on λ. Consequently, it can be replaced by a stochastic approximation

$$\mathcal{L}(x, \mu, C) \approx \frac{1}{M} \sum_{m=1}^{M} \log p(x, Cz_m + \mu) + \log |C| + \mathcal{H}_\phi$$

where z_m are drawn from $N(0, I)$. We can now easily compute the gradients using automatic differentiation of the model and the chain rule, by first differentiating $\log p(x, Cz + \mu)$ with respect to θ and then $\theta = Cz + \mu$ w.r.t. μ and C, resulting in

$$\nabla_\mu \mathcal{L}(x, \mu, C) \approx \frac{1}{M} \sum_{m=1}^{M} \nabla_\theta \log p(x, \theta_m) \tag{3}$$

$$\nabla_C \mathcal{L}(x, \mu, C) \approx \frac{1}{M} \sum_{m=1}^{M} \nabla_\theta \log p(x, \theta_m) \times (\theta - \mu)^T C^{-T} + \Delta_C,$$

where Δ_C is the derivative of $\log |C|$. Here C can either be a diagonal matrix to correspond to a mean-field approximation or cholesky factor of an arbitrary covariance matrix over the parameters.

3 Stochastic Average Gradients

We extend the idea of stochastic average gradients (SAG) [11] to automatic variational inference, by replacing the stochastic gradient with the stochastic average gradient computed by combining the most recent mini-batch with gradients computed for earlier mini-batches: $\nabla \mathcal{L}(X, \lambda) \approx \sum_{b=1}^{B} \nabla \mathcal{L}(X_b, \lambda)$, where X_b denotes the samples in the bth mini-batch. SAG is straightforward to implement, requiring only storage of the previously computed gradients, and provides an intermediate technique between full-batch gradients and pure stochastic gradients. The theoretical guarantees of SAG providing efficient convergence for losses consisting of finite sums are not directly applicable because of the stochastic estimates, but the motivation of reducing the gradient noise still applies.

We implement SAG by multiplying the gradients for all but the latest mini-batch X_b by $\alpha \in [0, 1]$ before each gradient step, and let α grow from 0 to 1 during the optimization using $(1 - 0.9^{0.5t/B})$, where t is the current iteration and B is the number of mini-batches. During the early phases the approximation is changing rapidly and hence we want to give less weight for old mini-batches with potentially stale gradients, but eventually we like to take steps along the full batch gradient.

We are not the first ones to consider algorithms motivated by SAG for variational inference, but the earlier approaches have been provided for conjugate models and they operated on top of sufficient statistics instead of gradients. Mandt and Blei [5] extended stochastic variational inference by SAG-like elements, and Archambeau and Ermis [1] proposed an incremental variational inference algorithm and its gradient-based variant that average sufficient statistics over mini-batches similar to how SAG averages gradients.

4 Importance sampling for re-using gradients

SAG as described above helps in reducing the gradient noise by combining the current mini-batch gradient with gradients of other mini-batches. It has effectively the same computational cost as standard SGD, since computing the gradient typically clearly dominates the total computation compared to the memory management required for SAG.

An interesting question is whether we can do better. When re-visiting a mini-batch that has already been processed, we have computed the gradient w.r.t. θ and evaluated it for some set of θ_m drawn from $q(\theta|\lambda_0)$, where λ_0 refers to the parameters of the approximation when the gradient was computed. Re-using that information offers an intriguing opportunity, and it can be done via importance sampling. Importance sampling is traditionally used for estimating expectations when we cannot easily draw samples from the density of interest; here we could draw samples from the new approximation but want to avoid computation and hence choose not to.

To estimate the expectation over $q(\theta|\lambda)$ using $\theta_m \sim q(\theta|\lambda_0)$ we use the estimate

$$\mathbb{E}_{q(\theta|\lambda)}[\nabla_\lambda \mathcal{L}(x, \lambda)] \approx \frac{1}{M} \sum_{m=1}^{M} \frac{q(\theta_m|\lambda)}{q(\theta_m|\lambda_0)} \nabla_{\theta_m} \log p(x, \theta_m) \nabla_\lambda f(z_m, \lambda).$$

Here z_m refers (with a slight abuse of notation) to a draw from the standard distribution that would have been required to create θ_m under the new approximation. The term $\nabla_{\theta_m} \log p(x, \theta_m)$ has already been computed, and hence to estimate the full gradient we only need to compute the weight $\frac{q(\theta_m|\lambda)}{q(\theta_m|\lambda_0)}$ and the latter term of the gradient. For normal approximation we merely need to find $z_m = C^{-1}(\theta_m - \mu)$ and compute the gradients as in (3).

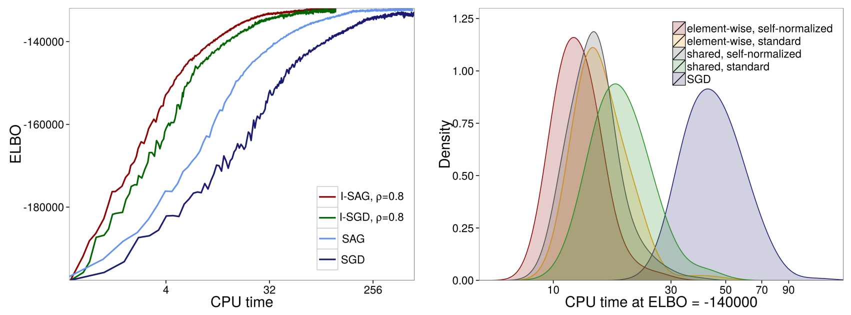

Figure 1: **Left**: Re-using gradients speeds up learning. Switching from SGD to SAG improves convergence speed as does re-using previous gradients with importance sampling, and combining both results in a total speed-up of almost an order of magnitude. Here, I-SAG and I-SGD use normalized weights computed for each element of the gradient separately. **Right**: The distribution of the convergence times over 200 random initializations for the different importance sampling variants. I-SAG outperforms SGD with any of the weighting schemes, but the naive version using a single set of weights without normalization is not as efficient as the other variants.

Importance sampling has high variance if $q(\theta|\lambda_0)$ and $q(\theta|\lambda)$ are very different. We propose three remedies to overcome this: (1) We use the importance sampled estimate only with probability ρ, and otherwise we reset the process by re-sampling θ_m and computing the gradient from scratch for that mini-batch, (2) We normalize the weights w_m to sum up to one, following the self-normalized importance sampling procedure typically used with unnormalized densities, and (3) We estimate each of the elements of the gradient independently using separate set of weights, instead of using the same weights for the whole gradient. The last trick allows using importance sampling in one-dimensional space instead of a high-dimensional one.

We call the algorithm I-SAG for importance-sampled SAG, and note that importance-sampled SGD can be implemented in a similar fashion by keeping $\alpha = 0$ throughout the optimization. We demonstrate I-SAG on isotropic Gaussian mixture model of the MNIST digits, using mean-field normal approximation for parameters transformed to unconstrained real values and ADAM for controlling the learning rate. Figure 1 shows that I-SAG outperforms SGD by a wide margin, and that re-using gradients help for both SAG and SGD. We also compared alternative approaches for importance sampling, observing that here element-wise normalized weights worked the best.

5 Discussion

We presented steps towards avoiding unnecessary computation during automatic variational inference, showing the derivations for the special case of normal approximations directly parametrized by μ and C. For such an approximation we can re-use previously computed gradients with importance sampling and a trivial transformation of $\nabla_\lambda \log p(x, \lambda)$. We showed that the proposed algorithm clearly outperforms standard SGD, and experimented with different strategies for controlling the variance of the importance sampling step. Further work is needed to find settings that would be robust for a wide class of models.

Models like variational autoencoder [2] use Gaussian approximation but parametrize the mean and covariance of the approximation with non-linear mappings from the observed data. For such models the full gradient consists of two computationally demanding parts, one for the decoder and one for the encoder. The first term can be re-used in our importance sampling algorithm, but the latter needs to be re-computed, which means the computational saving is smaller. The practical value of the idea for such models remains to be evaluated empirically as future work.

Acknowledgements

This work was supported by Tekes as part of the Scalable probabilistic analytics project and by Academy of Finland as part of the project number 266969.

References

[1] Cedric Archambeau and Beyza Ermis. Incremental variational inference of Latent Dirichlet Allocation. *arXiv:1507.05016*, 2015.

[2] D.P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proceedings of ICLR*, 2014.

[3] Alp Kucukelbir, Dustin Tran, Rajesh Ranganath, Andrew Gelman, and David M. Blei. Automatic differentiation variational inference. *Journal of Machine Learning Research*, 2016.

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

[5] Stephan Mandt and David Blei. Smoothed gradients for stochastic variational inference. In *Proceedings of NIPS*, 2014.

[6] Christian A. Naesseth, Francisco J. R. Ruiz, Scott W. Linderman, and David M. Blei. Rejection sampling variational inference. *arXiv:1610.05683*, 2016.

[7] Manfred Opper and Cedric Archambeau. The variational Gaussian approximation revisited. *Neural computation*, 21(3):876–792, 2009.

[8] Rajesh Ranganath, Sean Gerrish, and David M Blei. Black box variational inference. In *Proceedings of Artificial Intelligence and Statistics (AISTATS)*, 2014.

[9] Francisco J.R. Ruiz, Michalis K. Titsias, and David M. Blei. The generalized reparameterization gradient. In *Proceedings of NIPS*, 2016.

[10] Tim Salimans and David A. Knowles. Fixed-form variational posterior approximation through stochastic linear regression. *Bayesian Analysis*, 8(4):837–882, 2013.

[11] Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *arXiv:1308.2388*, 2013.

[12] Michalis Titsias and Miguel Lázaro-Gredilla. Doubly stochastic variational bayes for non-conjugate inference. In *Proceedings of International Conference on Machine Learning (ICML)*, 2014.