Re-using gradient computations in automatic variational inference

Joseph Sakaya **Arto Klami**

Department of Computer Science, Helsinki Institute for Information Technology HIIT, University of Helsinki

Gradient-based variational inference

Given a probabilistic model $p(x, \theta)$, the variational approximation $q(\theta|\lambda)$ for the posterior $p(\theta|x)$ is obtained by maximizing the evidence lower bound (ELBO)

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

Direct optimization of the loss is difficult, but re-parameterizing the approximation using

$$q(\theta|\lambda) = \frac{1}{|C|}\phi(C^{-1}(\theta - \mu))$$

allows re-writing the ELBO as

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

where the integral is now over a standard distribution $\phi(z)$ that does not depend on the parameters λ [1]. Monte Carlo approximation now makes gradient-based optimization possible.

A practical learning algorithm:

1. Take a mini-batch of data points x
2. Randomly sample M values z_m from $\phi(z)$
3. Convert z_m into θ_m using $\theta_m = f(z_m) = Cz_m + \mu$
4. Automatically differentiate $\nabla_\theta \log p(x, \theta)$ and evaluate it at θ_m
5. Update the variational parameters:

$$\mu_{t+1} = \mu_t + \gamma \frac{1}{M} \sum_{m=1}^{M} \nabla_\theta \log p(x, \theta_m)$$

$$C_{t+1} = C_t + \gamma \frac{1}{M} \sum_{m=1}^{M} (\nabla_\theta \log p(x, \theta_m) \times z_m + {}^1/c)$$

Beyond the basic algorithm:

- Constrained parameters handled via transformations
- Re-parameterization can be generalized for other distributions
- Stochastic average gradients (SAG) [2] instead of SGD; compute running average of gradients over the mini-batches

Importance-Sampled Stochastic Average Gradient (I-SAG)

Core idea:

During optimization, we iteratively draw samples θ_m from the approximation and evaluate the gradient of $\log p(x, \theta)$ for those. This is where most of the computation goes.

When visiting a data point (or mini-batch) again, we already know the gradient computed for some θ_m, which could have been drawn from the current approximation as well. **Can we re-use the gradients somehow?**

Requires storing the gradients for each mini-batch visited, similar to what is needed to compute stochastic average gradients [2]. This means we can just as well combine both techniques.

Importance sampling: Tells how an expectation, here the gradient, can be estimated with samples drawn from a different distribution:

$$\mathbb{E}[\nabla_\lambda \log p(x, \theta)] \approx \sum_{m=1}^{M} w_m \nabla_{\theta_m} \log p(x, \theta_m) \nabla_\lambda f(z_m, \lambda)$$

$$w_m = \frac{1}{M} \times \frac{q(\theta_m|\lambda)}{q(\theta_m|\lambda_0)}$$

Problem:
High variance when $q(\theta_m|\lambda)$ and $q(\theta_m|\lambda_0)$ are very different.
Solutions:

- Use self-normalized importance sampling, replacing w_m with $\hat{w}_m = \frac{w_m}{\sum_j w_j}$
- Estimate the individual elements of the whole gradient independently, using separate set of w_m for each

Experiments

Implementation details:

- We re-use the old gradients with probability ρ, and otherwise compute the gradient again
- SAG implemented so that gradients of other mini-batches are multiplied with α before each update
- α ramps up from 0 to 1 during the first few passes through the data
- Standard SAG obtained when $\rho = 0$ and standard SGD when also $\alpha = 0$
- Automatic differentiation with *autograd*, working on Edward implementation

Demonstration on a Gaussian mixture model on the MNIST data, simply to study the basic behavior of the algorithm.

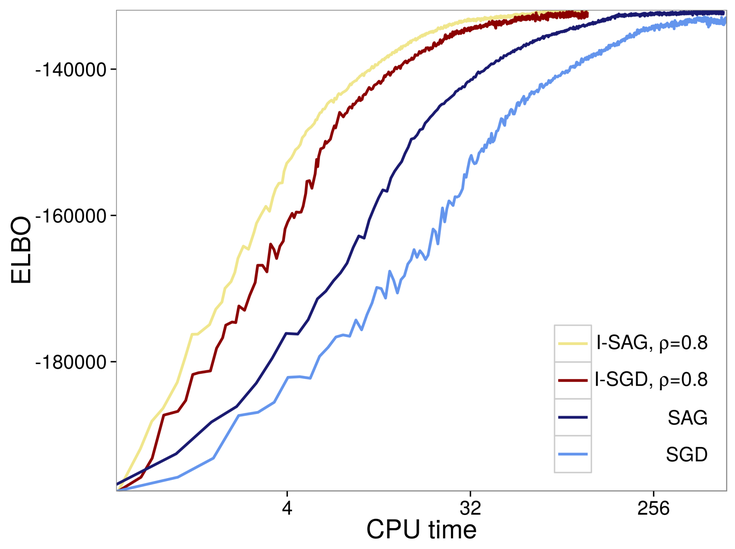

Fig 1. Re-using gradients speeds up learning compared to pure SGD. Switching to SAG helps as does re-using the previous gradients, and combining both elements provides the fastest convergence. Here self-normalized individual weights for each gradient element were used.

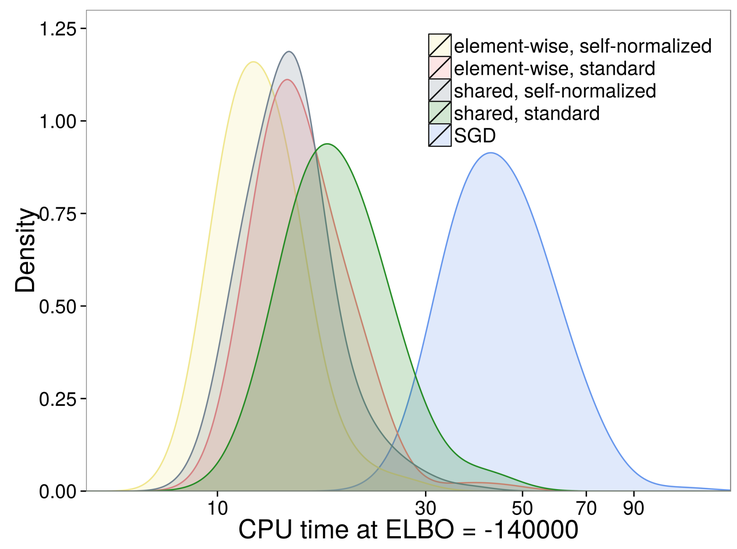

Fig 2. Distribution of convergence times over 200 random initializations. Our algorithm outperforms SGD with any of the weighting schemes, and the best variant for this model used self-normalization applied for individual weights for each element of the gradient.

Acknowledgments

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 and the Finnish Center of Excellence in Computational Inference Research (COIN).

References

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

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