Variational Inference and Dropout

Anonymous Author(s)
Affiliation
Address
email

Abstract

1 Variational Inference for Latent Variable Model

Let $x = (x_i, i \in [\![1, n]\!])$ and $y = (y_i, i \in [\![1, n]\!])$ be i.i.d. input-output pairs and $w \in \mathsf{W} \subseteq \mathbb{R}^J$ be a latent variable. The joint distribution of y, w can be written as:

$$p(y, w|x) = p(w) \prod_{i=1}^n p_i(y_i|x_i, w) \tag{1}$$

Our goal is to compute the posterior distribution $p(w|y, x)$ given the input-output pairs x, y. The variational Inference (VI) algorithm [?] consists of minimizing the KL divergence between a candidate family of parametric distributions $\{q(w, \boldsymbol{\theta}), \boldsymbol{\theta} \in \Theta\}$ and the posterior distribution $p(w|y, x)$ of the global latent variable w. For example, $q(w; \boldsymbol{\theta})$ belongs to a simple family of distributions such as the multivariate Gaussian family with mean ρ and covariance matrix $\sigma^2\mathbf{I}$, where we have $\boldsymbol{\theta} = (\rho, \sigma^2) \in \Theta = \mathbb{R} \times \mathbb{R}_+^*$. VI can be framed as an optimization problem, usually in terms of KL divergence, of the following form:

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta} \in \Theta}{\arg\min} \;\; \mathrm{KL}\left(q(w; \boldsymbol{\theta}) \,||\, p(w|y, x)\right) = \underset{\boldsymbol{\theta} \in \Theta}{\arg\min} \;\; \mathcal{L}(\boldsymbol{\theta}) \tag{2}$$

where $\mathcal{L}(\boldsymbol{\theta}) := n^{-1} \sum_{i=1}^n \mathcal{L}_i(\boldsymbol{\theta})$ for $\boldsymbol{\theta} \in \Theta$ with :

$$\mathcal{L}_i(\boldsymbol{\theta}) := -\int_{\mathsf{W}} q(w; \boldsymbol{\theta}) \log p_i(y_i|x_i, w) \mathrm{d}w + \frac{1}{n} \mathrm{KL}\left(q(w; \boldsymbol{\theta}) \,||\, p(w)\right) = r_i(\boldsymbol{\theta}) + d(\boldsymbol{\theta}), \tag{3}$$

Directly optimizing the finite sum objective (2) can be infeasible especially when $n \gg 1$. This is because evaluating the objective function $\mathcal{L}(\boldsymbol{\theta})$ requires a full pass of computation over the entire dataset, and this approach does not adapt to complex models when the last integral cannot be evaluated analytically. For instance, such optimization problem is notoriously hard for Bayesian neural networks [?].

To apply the MISSO method with a stochastic surrogate model (??), we consider the following quadratic surrogate function. For any $i \in [\![1, n]\!]$, we take:

$$\widehat{\mathcal{L}}_i(\boldsymbol{\theta}; \overline{\boldsymbol{\theta}}) := \mathcal{L}_i(\overline{\boldsymbol{\theta}}) + \left\langle \nabla \mathcal{L}_i(\overline{\boldsymbol{\theta}}) \,|\, \boldsymbol{\theta} - \overline{\boldsymbol{\theta}} \right\rangle + \frac{L}{2} \|\overline{\boldsymbol{\theta}} - \boldsymbol{\theta}\|^2 \tag{4}$$

where L is the smoothness modulus of $\mathcal{L}_i$ at $\overline{\boldsymbol{\theta}}$. To compute the gradient $\nabla \mathcal{L}_i(\overline{\boldsymbol{\theta}})$, we apply the re-parametrization technique suggested in [???]. Let $t : \Theta \times \mathbb{R}^d \mapsto \mathbb{R}^d$ be a measurable function and ϕ be the density of the standard normal distribution $\mathcal{N}_d(0, \mathbf{I})$. The function t is designed such that for all $\overline{\boldsymbol{\theta}} \in \Theta$ and for $\epsilon \sim \phi(\cdot)$, the distribution of the random vector $W = t(\overline{\boldsymbol{\theta}}, \epsilon)$ is the same as $q(\cdot, \overline{\boldsymbol{\theta}})$. It follows from [?, Proposition 1] that:

$$\nabla \int_{\mathsf{W}} \log p_i(y_i|x_i, w) q(w, \overline{\boldsymbol{\theta}}) \mathrm{d}w = \int_{\mathbb{R}^d} \mathrm{J}_t^{\boldsymbol{\theta}}(\overline{\boldsymbol{\theta}}; e) \nabla \log p_i(y_i|x_i, t(\overline{\boldsymbol{\theta}}, e)) \phi(e) \mathrm{d}e \tag{5}$$

where for each $e \in \mathbb{R}^d$, $\mathrm{J}_t^{\boldsymbol{\theta}}(\overline{\boldsymbol{\theta}}; e)$ is the Jacobian of the function $t(\cdot, e)$ with respect to $\boldsymbol{\theta}$ evaluated at $\overline{\boldsymbol{\theta}}$. Consequently, we can apply the MISSO method to tackle the VI problem with the pair $\big(r_i(\boldsymbol{\theta}; \overline{\boldsymbol{\theta}}, e), \phi(e)\big)$ where $r_i(\boldsymbol{\theta}; \overline{\boldsymbol{\theta}}, e)$ reads:

$$
\begin{aligned}
r_i(\boldsymbol{\theta}; \overline{\boldsymbol{\theta}}, e) := &\big(-\log p_i(y_i | x_i, t(\overline{\boldsymbol{\theta}}, e)) + d(\overline{\boldsymbol{\theta}})\big) \\
&+ \Big(-\mathrm{J}_t^{\boldsymbol{\theta}}(\overline{\boldsymbol{\theta}}; e) \nabla \log p_i(y_i | x_i, t(\overline{\boldsymbol{\theta}}, e)) + \nabla d(\overline{\boldsymbol{\theta}})\Big)^{\top} (\boldsymbol{\theta} - \overline{\boldsymbol{\theta}}) + \frac{L}{2} \|\boldsymbol{\theta} - \overline{\boldsymbol{\theta}}\|^2
\end{aligned}
\tag{6}
$$

2 Numerical Experiments

2.1 Fitting Bayesian LeNet-5 on MNIST:

In this experiment, we implement the MISSO algorithm for variational inference in the Bayesian variant of LeNet-5 [?] (architecture is described in Appendix 1.1). We train this network on the MNIST dataset [?] used extensively as a benchmark example. The training set is composed of $N = 55\,000$ handwritten digits, 28×28 images. Each image is labelled with its corresponding number (from zero to nine).

Given weight matrices $(w_\ell)_{\ell=1}^L \in \mathsf{W}^L$, we put, for each layer ℓ a standard Gaussian prior distributions over those weights: $p(w_\ell) = \mathcal{N}(0, I)$, as in [?]. We denote by $f(x_i, (w_\ell)_{\ell=1}^L)$ the output of the nested function describing the neural network with weights $(w_\ell)_{\ell=1}^L \in \mathsf{W}^L$ taking as input a data sample x_i. We assume a softmax likelihood in that classification task: $p(y_i | x_i, w) = \mathrm{Softmax}(f(x_i, (w_\ell)_{\ell=1}^L))$.

The variational distribution $q(w, \theta)$ belongs to the Gaussian multivariate distribution family. Here, the MISSO algorithm coincides with a mini-batch version of the Variational Inference algorithm. At iteration k, minimizing the sum of stochastic surrogates defined as in (**??**) and by the quantities (6) yields the following MISSO update: pick a function index I_k uniformly on $[\![1, n]\!]$, sample a Monte Carlo batch $\{e_m^{(k)}\}_{m=1}^{M_{(k)}}$ from the standard Gaussian distribution and update the parameters as $\boldsymbol{\theta}^{(k)} = \frac{1}{n} \sum_{i=1}^n \boldsymbol{\theta}^{(\tau_i^k)} - \frac{1}{2\gamma} \sum_{i=1}^n \hat{\boldsymbol{m}}_i^k$ where $\hat{\boldsymbol{m}}_i^k$ are defined recursively as follows:

$$
\hat{\boldsymbol{m}}_i^{(k)} \triangleq \begin{cases} -\frac{1}{M_{(k)}} \sum_{m=0}^{M_{(k)}-1} \mathrm{J}_{\boldsymbol{\theta}}^t(e_m^{(k)}) \nabla_{\boldsymbol{\theta}} \log p_i(y_i, x_i | t(\boldsymbol{\theta}, e_m^{(k)})) + \nabla d(\boldsymbol{\theta}^{(k-1)}) & \text{if } i \in I_k \\ \hat{\boldsymbol{m}}_i^{(k-1)} & \text{otherwise} \end{cases}
\tag{7}
$$

We compare the convergence behaviors of the following state of the art optimization algorithms, using their vanilla implementations on TensorFlow [?]: the ADAM [?], the Momentum [?] the *Bayes by Back-prop* (BBB) [?] and the Dropout [?] algorithms versus our MISSO update. The loss function (3) and its gradients were computed by Monte Carlo integration using Tensorflow Probability library, based on the reparametrization trick. We use the following hyperparameters for all runs: the learning rate is set to 10^{-3}, we run 100 epochs and use a mini-batch size of 128.

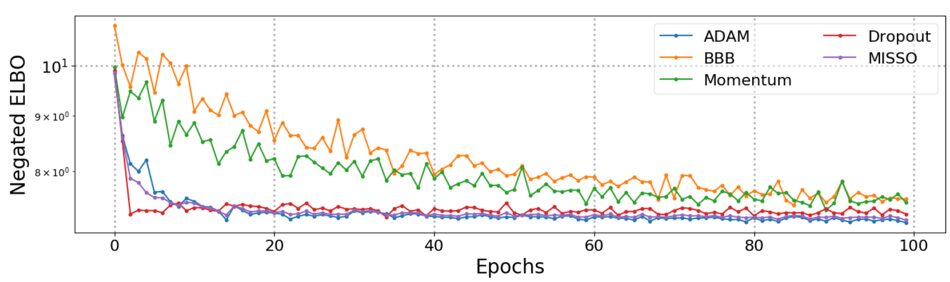

Figure 1: (Incremental Variational Inference) Convergence of the negated ELBO on MNIST.

.1 Incremental Variational Inference for MNIST

.1.1 Bayesian LeNet-5 Architecture

layer type	width	stride	padding	input shape	nonlinearity
convolution (5×5)	6	1	0	$1 \times 32 \times 32$	ReLU
max-pooling (2×2)		2	0	$6 \times 28 \times 28$	
convolution (5×5)	6	1	0	$1 \times 14 \times 14$	ReLU
max-pooling (2×2)		2	0	$16 \times 10 \times 10$	
fully-connected	120			400	ReLU
fully-connected	84			120	ReLU
fully-connected	10			84	

Table 1: LeNet-5 architecture