ZhMaR documentation:
Library for Variational inference

Anton Zhevnerchuk, Sergey Makarychev,
Aleksandr Rozhnov

May 12, 2018

1 Introduction

In this section we will discuss our model and approaches that we implemented.

We have a probabilistic model $p_\theta(x, z)$, where x is a vector of observed values, z - latent variables of this model, θ - vector of parameters. Using Bayes formula, we can calculate the posterior over the latent variable z as

$$p(z|x) = \frac{p(x, z)}{\int p(x, z)dz}. \tag{1}$$

Unfortunately, the denominator of this previous expression is the (usually intractable) evidence. Thus, the task is to approximate a posterior distribution $p(z|x)$. The first simplification of the model is to search for approximation of posterior distribution in a ”good”family of functions. Let us denote by $q(z|\psi)$ such an approximation.

Consider the following evident equations:

$$\log p_\theta(x) = \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x) = \mathbb{E}_{z\sim q_\phi(z|x)} \log \frac{p_\theta(x, z)q_\phi(z|x)}{q_\phi(z|x)p_\theta(z|x)} = \mathbb{E}_{z\sim q_\phi(z|x)} \log \frac{p_\theta(x, z)}{q_\phi(z|x)} + KL(q_\phi(z|x)||p_\theta(z|x))$$

$$\log p_\theta(x) \geqslant \mathbb{E}_{z\sim q_\phi(z|x)} \log \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} = \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) - KL(q_\phi(z|x)||p(z)) = L(x; \phi, \theta) \to \max_{\phi,\theta}$$

The last low bound for $\log p_\theta(x)$ is called ELBO. It has several equivalent forms, such as:

$$L_{KL}(x; \phi, \theta) = \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x|z) - KL(q_\phi(z|x)||p(z))$$
$$L_{ent}(x; \phi, \theta) = \mathbb{E}_{z\sim q_\phi(z|x)} \log p_\theta(x, z) + H(q_\phi(z)),$$

called ELBO in KL and Entropy form.

We use Variational Inference (VI) for solving this optimization task. Ideas, behind this method, are well studied and described in the article ([RR13]).

2 Notation

In this documentation we will be supported by the following notation.

- beta : global latent variable
- z_i : local latent variable
- x_i : observed values

The following methods for prior distribution will denote:

- **log_likelihood_global(beta)**: $\log p(\beta)$
- **log_likelihood_local(z, beta)**: $\log p(z|\beta)$
- **log_likelihood_joint(x, z, beta)** : $\log p(x, z|\beta)$

The following methods for variational distribution will denote:

- **log_likelihood_global(beta)**: $\log q(\beta)$
- **log_likelihood_local(z, beta)**: $\log q(z|\beta)$
- **log_likelihood_joint(x, z, beta)** : $\log q(x, z|\beta)$

3 class SVI:

3.1 *check_methods*

Check if provided model supports input loss. For now we support losses in the form 'bb1', 'bb2, 'kl', and 'entropy'.

- **Args**:
  - **loss:** string, name of loss
- **Returns**:
  - **flag:** boolean, if model supports loss
  - **message:** string, error source if model does not support loss
  - **methods_to_implement:** dict, methods to implement if model does not support loss

3.1.1 Example of usage

```
from BBSVI import SVI
import torch
import test_models

prior = test_models.ToyPrior()
var = test_models.ToyVariationalDistribution()

loss = 'entropy'

opt = torch.optim.Adam(var.parameters, lr=1e-3)
BBSVI.SVI(data, prior, var, opt)

svi.check_methods(loss)
>>> (True, 'OK', defaultdict(list, {}))
```

3.2 *make_inference*

This function performs SVI inference.

- **Args**:
  - **num_steps:** int, maximum number of epoches
  - **tol:** required tolerance
  - **num_samples:** int, number of samples used for ELBO approximation
  - **batch_size:** int, size of one batch
  - **loss:** string, loss function
  - **discounter_schedule:** used only for 'entropy' loss, None or torch tensor of size num_steps, discounter_schedule[i] is a discounter for an analytically-computed term at step i
  - **kl:** None or callable, compute KL divergency between variational and prior distributions, required only for 'kl' loss
  - **shuffle:** boolean, if batch is shuffled every epoch or not
  - **print_progress:** boolean, if True then progress bar is printed
  - **callback:** None or callable, if not None, applied to loss after every iteration
  - **retain_graph:** boolean, passed to loss.backward()

3.2.1 Example of usage

```
from BBSVI import SVI
import torch
import test_models

prior = test_models.ToyPrior()
var = test_models.ToyVariationalDistribution()

loss = 'entropy'

opt = torch.optim.Adam(var.parameters, lr=1e-3)
svi = BBSVI.SVI(data, prior, var, opt)

svi.make_inference(loss)
```

3.3 *bb1_loss_*

This function performs bb1 loss. This is private method, not needed to be called explicitly.
bb1_loss_(self, num_samples, batch_indices)

- **Args**:
  - **num_samples:** int, number of samples used for approximation
  - **batch_indices:** array, indices of batch.
- **Returns**:
  - **loss:** float, Black Box loss function; computed loss for model

The following methods are required to be implemented by user:

- **Prior_distr required methods**:
  - **log_likelihood_global(beta)**
  - **log_likelihood_local(z, beta)**
  - **log_likelihood_joint(x, z, beta)**
- **Var_distr required methods**:
  - **log_likelihood_global(beta)**
  - **log_likelihood_local(z, idx)**
  - **sample_global()**
  - **sample_local(beta, idx)**

3.4 *bb2_loss_*

Computing loss of BB SVI 2, which has lower variance compare to BB SVI 1
bb2_loss_(self, num_samples, batch_indices)

- **Args**:
  - **num_samples:** int, number of samples used for approximation
  - **batch_indices:** array, indices of batch.
- **Returns**:
  - **loss:** float, Black Box loss function; computed loss for model

The following methods are required to be implemented by user:

- **Prior_distr required methods**:
  - **log_likelihood_global(beta)**
  - **log_likelihood_local(z, beta)**
  - **log_likelihood_joint(x, z, beta)**
- **Var_distr required methods**:
  - **log_likelihood_global(beta)**
  - **log_likelihood_local(z, idx)**
  - **sample_global()**
  - **sample_local(beta, idx)**

The following attributes are required to be assigned by user:

- **Var_distr required attributes**:
  - **global_parameters**: list of global parameters
  - **local_parameters**: list of local parameters, i-th entry corresponds to i-th latent variable

3.5 *entropy_form_loss_*

Computing ELBO estimator in entropy form
entropy_form_loss_(self, num_samples, batch_indices, discounter=1)

- **Args**:
  - **num_samples:** int, number of samples used for approximation
  - **batch_indices:** array, indices of batch.
  - **discounter:** coefficient of entropy term
- **Returns**:
  - **loss:** float, ELBO in entropy form estimator

The following methods are required to be implemented by user:

- **Prior_distr required methods**:
  - **log_likelihood_global(beta)**
  - **log_likelihood_joint(x, z, beta):**
- **Var_distr required methods**:
  - **entropy(batch_indices)**
  - **sample_global()**
  - **sample_local(beta, idx)**

3.6 *kl_form_loss_*

Computing ELBO estimator in Kullback–Leibler divergence form
kl_form_loss_(self, num_samples, batch_indices, kl, discounter=1)

- **Args**:
  - **num_samples:** int, number of samples used for approximation
  - **batch_indices:** array, indices of batch.
  - **kl:** callable, function which computes KL divergence between variational and prior based on based indices
  - **discounter:** coefficient of entropy term
- **Returns**:
  - **loss:** float, ELBO in KL form estimator

The following methods are required to be implemented by user:

- **Prior_distr required methods**:
  - **log_likelihood_cond(x, z, beta):**
- **Var_distr required methods**:
  - **sample_global()**
  - **sample_local(beta, idx)**

3.7 *handle_nones*

Replace all Nones with torch tensors containing one zero
handle_nones(container)

- **Args**:
  - **container**: tuple
- **Returns**:
  - **handled_container**: container with torch tensor containing one zero instead of Nones.

4 *class HistoryCollector*

A simple class which is helpful to collect loss history of SVI

- **Args**:
  - **data_size**: int, number of data points
  - **batch_size**: : int, batch_size for SVI

Methods:

4.1 *collect_history*:

collect_history(self, loss)
Saving loss, can be passed as callback arg to SVI.make_inference

- **Args**:
  - **loss**: : torch.tensor, loss

5 References

[RR13] Sean Gerrish Rajesh Ranganath David M. Blei, *Black Box Variational Inference* (2013), available at http://www.cs.columbia.edu/~blei/papers/RanganathGerrishBlei2014.pdf.