TensorFlow Requirement 1.x TensorFlow 2 Not Supported ✗

REINFORCing Concrete with REBAR

Implemention of REBAR (and other closely related methods) as described in "REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models" by George Tucker, Andriy Mnih, Chris J. Maddison, Dieterich Lawson, Jascha Sohl-Dickstein (https://arxiv.org/abs/1703.07370).

Learning in models with discrete latent variables is challenging due to high variance gradient estimators. Generally, approaches have relied on control variates to reduce the variance of the REINFORCE estimator. Recent work (Jang et al. 2016; Maddison et al. 2016) has taken a different approach, introducing a continuous relaxation of discrete variables to produce low-variance, but biased, gradient estimates. In this work, we combine the two approaches through a novel control variate that produces low-variance, unbiased gradient estimates. Then, we introduce a novel continuous relaxation and show that the tightness of the relaxation can be adapted online, removing it as a hyperparameter. We show state-of-the-art variance reduction on several benchmark generative modeling tasks, generally leading to faster convergence to a better final log likelihood.

REBAR applied to multilayer sigmoid belief networks is implemented in rebar.py and rebar_train.py provides a training/evaluation setup. As a comparison, we also implemented the following methods:

- NVIL
- MuProp
- Gumbel-Softmax

The code is not optimized and some computation is repeated for ease of implementation. We hope that this code will be a useful starting point for future research in this area.

Errata

11/27/2019

The _generator_network function has separate paths for the unconditional and conditional generative models. In the conditional generative models code path, the generative model does not have multiple stochastic layers even when n_layers is > 1. My intention was to have multiple stochastic layers in the conditional generative model, however, due to a bug this is not how it was implemented. As the code is currently, with the conditional generative model and n_layers > 1, the recognition network has multiple stochastic layers, but the generative model has a single stochastic layer.

Hai-Tao Yu (yuhaitao@slis.tsukuba.ac.jp) discovered this issue.

Quick Start:

Requirements:

- TensorFlow (see tensorflow.org for how to install)
- MNIST dataset
- Omniglot dataset

First download datasets by selecting URLs to download the data from. Then fill in the download_data.py script like so:

```
MNIST_URL = 'http://yann.lecun.com/exdb/mnist'
MNIST_BINARIZED_URL =
'http://www.cs.toronto.edu/~larocheh/public/datasets/binarized_mnist'
OMNIGLOT_URL =
'https://github.com/yburda/iwae/raw/master/datasets/OMNIGLOT/chardata.mat'
```

Then run the script to download the data:

```
python download_data.py
```

Then run the training script:

```
python rebar_train.py --
hparams="model=SBNDynamicRebar,learning_rate=0.0003,n_layer=2,task=sbn"
```

and you should see something like:

```
Step 2084: [-231.026474      0.3711713     1.            1.06934261    1.07023323
    1.02173257    1.02171052    1.            1.            1.            1.        ]
-3.6465678215
Step 4168: [-156.86795044    0.3097114     1.            1.03964758    1.03936625
    1.02627242    1.02629256    1.            1.            1.            1.        ]
-4.42727231979
Step 6252: [-143.4650116     0.26153237    1.            1.03633797    1.03600132
    1.02639604    1.02639794    1.            1.            1.            1.        ]
-4.85577583313
Step 8336: [-137.65275574    0.22313026    1.            1.03467286    1.03428006
    1.02336085    1.02335203    0.99999988    1.            0.99999988
    1.        ]
-4.95563364029
```

The first number in the list is the log likelihood lower bound and the number after the list is the log of the variance of the gradient estimator. The rest of the numbers are for debugging.

We can also compare the variance between methods:

```
python rebar_train.py \
  --hparams="model=SBNTrackGradVariances,learning_rate=0.0003,n_layer=2,task=omni"
```

and you should see something like:

```
Step 959: [ -2.60478699e+02   3.84281784e-01   6.31126612e-02   3.27319391e-02
   6.13379292e-03   1.98278503e-04   1.96425783e-04   8.83973844e-04
   8.70995224e-04             -inf]
('DynamicREBAR', -3.725339889526367)
('MuProp', -0.033569782972335815)
('NVIL', 2.7640280723571777)
('REBAR', -3.539274215698242)
('SimpleMuProp', -0.040744658559560776)
Step 1918: [ -2.06948471e+02   3.35904926e-01   5.20901568e-03   7.81541676e-05
   2.06885766e-03   1.08521657e-04   1.07351625e-04   2.30646547e-04
   2.26554010e-04  -8.22885323e+00]
```

```
('DynamicREBAR', -3.864381790161133)
('MuProp', -0.7183765172958374)
('NVIL', 2.266523599624634)
('REBAR', -3.662022113800049)
('SimpleMuProp', -0.7071359157562256)
```

where the tuples show the log of the variance of the gradient estimators.

The training script has a number of hyperparameter configuration flags:

- task (sbn): one of {sbn, sp, omni} which correspond to MNIST generative modeling, structured prediction on MNIST, and Omniglot generative modeling, respectively
- model (SBNGumbel) : one of {SBN, SBNNVIL, SBNMuProp, SBNSimpleMuProp, SBNRebar, SBNDynamicRebar, SBNGumbel SBNTrackGradVariances}. DynamicRebar automatically adjusts the temperature, whereas Rebar and Gumbel-Softmax require tuning the temperature. The ones named after methods uses that method to estimate the gradients (SBN refers to REINFORCE). SBNTrackGradVariances runs multiple methods and follows a single optimization trajectory
- n_hidden (200): number of hidden nodes per layer
- n_layer (1): number of layers in the model
- nonlinear (false): if true use 2 x tanh layers between each stochastic layer, otherwise use a linear layer
- learning_rate (0.001): learning rate
- temperature (0.5): temperature hyperparameter (for DynamicRebar, this is the initial value of the temperature)
- n_samples (1): number of samples used to compute the gradient estimator (for the experiments in the paper, set to 1)
- batch_size (24): batch size
- muprop_relaxation (true): if true use the new relaxation described in the paper, otherwise use the Concrete/Gumbel softmax relaxation
- dynamic_b (false): if true dynamically binarize the training set. This increases the effective training dataset size and reduces overfitting, though it is not a standard dataset

Maintained by George Tucker (gjt@google.com, github user: gjtucker).