Guidelines for Santa

Changyou Chen
Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708
cchangyou@gmail.com

This document provides guidelines to use the code for the Santa algorithm [1]. There are three versions, a MATLAB version for FNN and CNN (under MATLAB folder), a python version for RNN (under PYTHON folder), and a scalable version implemented in Caffe [2] (under Caffe folder). While the MATLAB version has been carefully tuned and used for the experiments in the paper, I will explain how to use the Caffe version because of its scalability and generazation.

Santa is implemented as a *solver* in Caffe, see *Santa_e_solver.cpp* in *solvers* folder. I use the Euler integrator because of its simplicity. To use the solver, specify a solver profile as other algorithms, but with additional hyparameters, summarized in Table 1.

Table 1: Additional hyper parameters for Santa

nD	#training samples
sigma	weight proportion when constructing the preconditioner, same as in the paper (default 0.999)
lambda	smooth parameter when constructing the preconditioner, same as in the paper (default 1e-8)
explore	#iterations used for exploration (default 5000)
C	parameter used for initializing $\mathbf{u}$, same as in the algorithm (default 100)
anneal_a anneal_b anneal_c	annealing temperature $=$ anneal_a $\times$ $(t +$ anneal_b$)^{\text{anneal_c}}$ with t being the iteration number default: anneal_a=1, anneal_b=0, anneal_c= 2
approx_g	whether consider using numerical approximation for $\nabla_\theta G$ as described in the paper (default 0)
pc	whether use preconditioner (default 1)
update_u	whether update α in the *refinement* stage (default 0)

Below are some practical guidelines for parameter setting.

- C is important for fast convergence, experiences suggest $C = 100$ for MNIST-size datasets, and setting it larger when running on larger datasets.
- When using preconditioners, it scales the learning rate, making the learning rates different from other algorithms. A good learning rate is 1e-7 in this case. When not using preconditioners, set the learning rate to the one similar to SGD with momentum.
- The approximate calculation for $\nabla_\theta G$ described in the paper sometimes explodes with inappropriate learning rates because the dimensions of the preconditioner mightt be uneven (some dimensions are highly peak). Empirically, we found it beneficial to drop out the approximation term, *i.e.*, set approx_g to the default value. This is reasonable because the term only affects the accuracy of the samples to the true posterior. Our final goal is to find a (global) optimal point, which is not affected much by the intermedia samples.
- When running on big datasets such as the ImageNet dataset, it is found that all the precondition-based algorithms such as Adam and RMSProp failed. Possible reasons might be that the preconditioners constructed from a small minibatch does not reflect well the global geometry when the dataset is large, or because of inappropriate hyparameter settings which is hard to tune. In such case, I recommend to use no preconditioners in Santa, *i.e.*, setting pc to 0.

- Another strategy to deal with big data is to set update_u to 1, so that the thermostat variable α would be updated in both the *exploration* and *refinement* stages.
- When setting *explore* to 0, *i.e.*, no *exploration* stage, set update_u to 1 so that α could be updated.

For running examples, please see *examples/mnist/train_lenet_santae.sh* for MNIST, *examples/cifar10/train_full_santa.sh* for CIFAT10, and *models/bvlc_alexnet/train_santa_alexnet.sh* for ImageNet, under the Caffe folder, as well as the corresponding ".log" files in the same folders for running examples.

References

[1] C. Chen, D. Carlson, Z. Gan, and C. Li. Bridging the gap between stochastic gradient mcmc and stochastic optimization. In *International Conference on Artificial Intelligence and Statistics*, 2016.

[2] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.