Asynchronous Sparse MiG

A demo for Asynchronous Sparse MiG (Comparing to KroMagnon and ASAGA).

Method "Asynchronous Sparse MiG" is described in the paper: "A Simple Stochastic Variance Reduced Algorithm with Fast Convergence Rates".

Submitted to ICML2018.

Usage

All algorithms are implemented in C++ including MiG, KroMagnon and ASAGA and all parameters can be passed through MATLAB.

To run the demo in MATLAB, first run `mex_all` in the MATLAB terminal to generate the mex file. (Note that the compiler should support at least `c++11`)

Determine all parameters in a MATLAB file and run the algorithms implemented in C++ by passing parameters through `Interface`, here is a piece of sample code:

```
% load dataset variable X,y

algorithm = 'AMiG'; % AMiG / KroMagnon / ASAGA
loop = int64(passes / 3); % loop count for AMiG

passes = 300; % total passes of train set, must be a multiple of 3

model = 'logistic'; % least_square / logistic

regularizer = 'L2';
lambda1 = 10^(-6); % L2 parameter

L = (0.25 * max(sum(X.^2, 1)) + lambda1); % logistic regression

theta = 0.3;
step_size = 1 / (0.5 * theta * L);

init_weight = zeros(Dim, 1); % Initial weight

thread_no = 8; % Thread Number

is_sparse = issparse(X);
result = Interface(X, y, algorithm, model, regularizer, init_weight, lambda1,...
    L, step_size, loop, is_sparse, sigma, lambda2, thread_no, theta);
```

Demo

One can run `Async_test` in the MATLAB terminal, a small demo using sparse dataset `rcv1_train.binary` from LIBSVM Data, to generate a plot shown as below.

Test environment: HP Z440 machine with single Intel Xeon E5-1630v4 with 3.70GHz cores, 16GB RAM, Ubuntu 16.04 LTS with GCC 4.9.0, MATLAB R2017b.

(Note that the performance for each algorithm is best-tuned on the test machine.)

```
>> Async_test
Building with 'g++'.
MEX completed successfully.
Model: L2-logistic
Algorithm: AMiG
Time: 4.247145 seconds
Algorithm: KroMagnon
Time: 2.912665 seconds
Algorithm: ASAGA
Time: 4.794878 seconds
```