Project: Making the MATLAB Implementation Competitive with Tensorflow

Last updated: May 25, 2020

Goal

- Using the Matlab-C interface to improve the running speed of our MATLAB implementation

Project Contents I

- From project 3 we know that the MATLAB implementation is slower than Tensorflow
- The main issue is on index manipulation
- In project 4 we have seen that at least for one place (matrix expansion), our multi-core C code can be faster than MATLAB's implementations
- If we can integrate such implementations to the simpleNN MATLAB code, then the overall training time can be reduced
- To do so we should use the MATLAB-C interface

Project Contents II

- Besides the matrix expansion, we want to develop C code for other bottlenecks as well
- We hope that eventually the MATLAB code can be as fast as Tensorflow
- Not clear if we can really reach this goal, but let's try the best

MATLAB-C Interface I

- Say we would like to replace

```
phiZ = phiZ(net.idx_phiZ{m}, :);
```

  with our own implementation
- We write a special interface file `matrixExpansion.cpp`
- It's a MATLAB `mexFunction` and the format must be like

MATLAB-C Interface II

```
/* The gateway function */
void mexFunction(int nlhs, mxArray *plhs[],
                 int nrhs, const mxArray *prhs[])
{
/* variable declarations here */

/* code here */
}
```

- See more information at
  https://www.mathworks.com/help/matlab/matlab_external/standalone-example.html

MATLAB-C Interface III

- Here we have four arguments
- `nlhs`: Number of output (left-side) arguments, or the size of the plhs array.
- `plhs`: Array of output arguments.
- `nrhs`: Number of input (right-side) arguments, or the size of the prhs array.
- `prhs`: Array of input arguments.
- Thus `prhs[0]` can be for example the input array for expansion
- We will show a real example of matrix expansion after project 4 presentation

An Example on Matrix Expansion I

- The .cpp code

```
#include <omp.h>

#include "mex.h"

extern "C" void mexFunction(int nlhs,
    mxArray* plhs[], int nrhs, const mxArray*
{
auto& matrix = prhs[0];
auto& indices = prhs[1];
auto& out = plhs[0];
```

An Example on Matrix Expansion II

```
auto l = mxGetM(indices);
auto m = mxGetM(matrix);
auto n = mxGetN(matrix);

auto A = (float*)mxGetPr(matrix);
auto a = mxGetPr(indices);

out = mxCreateNumericMatrix(l, n, mxSINGLE_C
auto B = (float*)mxGetPr(out);
```

An Example on Matrix Expansion III

```
#pragma omp parallel for schedule(static)
for(mwSize j = 0; j < n; j++)
  for(mwSize i = 0; i < l; i++)
    B[j*l+i] = A[j*m+int(a[i])-1];
}
```

- See files provided in this directory
- To build the .mex file for MATLAB, we provide two ways by using

  make.m

  or

  Makefile

An Example on Matrix Expansion IV

- Thus you can either type

  `>> make`

  under MATLAB or

  `$ make`

  under the shell
- For unknown reasons, if using

  `>> make`

  on the department's servers, MATLAB reported an error saying that the resulting file `is not a MEX file`.

An Example on Matrix Expansion V

But in fact it works

- To build the file on Octave, the only way we provided is through

```
>> make
```

However, you need to remove the line

```
#include "matrix.h"
```

in the cpp file.

- The usage can be like

```
>> A = single(rand(1000, 1000));
>> a = randi(1000, 2000, 1);
>> isequal(A(a, :), matrixExpansion(A, a))
```

An Example on Matrix Expansion VI

We provide a `test.m` for running these three lines

Presentation I

- Presentations for projects 5 and 6

```
proj ID
5 ntust_f10802006
6 b05201015
5 b05201024
6 b05201037
5 t08303135
5 b06502060
5 r08521508
6 d08525008
6 b05701231
```

Presentation II

6 b06901143
5 t08902130
5 b06902124
6 b05902035
5 b05902050
5 b05902105
5 d08921024
6 a08922103
5 a08922119
6 a08922203
6 d08922029

Presentation III

5 d08922034
5 p08922005
6 r08922019
6 r08922082
5 r08922163
5 r07922100
6 r07922154
6 r08922a07
5 d04941016
6 r08942062
6 a08946101

Presentation IV

please do a 10-minute presentation (9-minute the contents and 1-minute Q&A)