1 任务描述

N3LGD对MKL多线程有了更好的支持，本篇记录了batch大小和多线程对程序速度影响，以及各种注意事项。

2 使用MKL库

我们可以使用MKL库编译出两种不同程序，一种是单线程程序，另一种是多线程程序，两种程序都有加速效果。如果编译链接mkl_sequential 则编译出单线程程序；如果编译使用mkl_intel_thread，则可编译出多线程程序。这个不同编译库的链接修改可以在CMakeList.txt中替换掉库（mkl_sequential 或者mkl_intel_thread）的名字即可。编译完成的程序在Linux使用ldd命令查看是否正确链接了库。 在运行多线程程序时，需要控制线程数量，用下述命令即可让程序启动5个线程:

export MKL_NUM_THREADS=5

3 记录时间的代码

进行这个实验需要注意的一点就是记录时间的代码。C++ 中有多种方式可以记录程序的运行时间，其中用clock()来记录多线程程序的运行时间会出现问题。 这个函数会返回程序消耗的处理器时间，如果在多线程程序中，程序占用的多个处理器，那么这个clock() 函数返回的时间会叠加在一起。这个函数一般用在单线程程序中，具体用法见:

`http://www.cplusplus.com/reference/ctime/clock/`

记录多线程程序所消耗的时间，一种保险的做法是直接记录当前时间，可以使用now()函数。具体用法见:

`http://www.cplusplus.com/reference/chrono/high_resolution_clock/now/`

另外程序在开发集和训练集的所花的时间太短，很难看出batch大小和多线程的影响，所以我们记录了训练的时间，取前5轮迭代所用时间。

4 实验设置和结果

实验使用的超参数见Table 1；准确率见Table 2；程序速度见Table 3，nomkl没有使用mkl库，seq使用mkl单线程加速，threadx是使用了x线程对程序进行加速。

hyperParams	value
hiddenSize	200
wordEmbSize	200
wordcontext	5
cnnLayerSize	25
adaEps	1e-06
adaAlpha	0.01
regParameter	1e-08
wordEmbFineTune	true

Table 1: 超参数设置

batch	1	50	100	150	200
nomkl	79.25	70.85	66.35	71.65	78.3
seqmkl	79.4	69.5	55.7	77.7	79.85
thread1	79.4	69.5	55.7	77.7	79.85
thread2	75	69.5	67.75	71.65	71.65
thread3	77.7	69.5	67.75	71.65	76.15
thread4	77.7	69.25	67.75	71.65	72.1
thread5	77.7	70.85	67.75	71.65	79.85

Table 2: 测试集上前5轮的准确率（%）

batch	1	50	100	150	200
nomkl	514.4	214.4	214.8	216.4	214.8
seq	480.2	201	199.6	199.6	202.8
thread1	477	196	196	196.6	198.2
thread2	401.4	139	136.4	138.4	139.4
thread3	368.6	119	118	119	117
thread4	354.6	107.8	106	107.4	107.4
thread5	341	97.6	100.4	101.8	101.6

Table 3: 前5轮迭代训练平均用时(s)

Layer	1	5	10	15	20	25
Acc	80.25	80.25	75.0	79.7	77.7	79.25

Table 4: 层数对性能的影响