Torch Week2 Report

蒋欣桐

Grade summary from grade_all.py

```
==================================================
all: grader summary:
score: 510/510
# passed: 53
# failed: 0
all tests passed!
==================================================
```

Questions answer

Why we cannot use while queue is not empty as exit condition directly in multithreading case?

因为多线程情况下 queue 为空不等价于所有节点的任务都已经完成。可能出现线程1中 queue 已经空了但是线程2中还有节点任务在执行的情况。应该直接统计完成节点数。

Why multithreading in backward execution matters, even if there is operator level multithreading?

操作符级别的多线程优化的是单个运算操作内部的计算。反向传播过程中层与层之间有依赖关系，但是又不一定是“直线型”的完全依赖关系，一个点可能有多个依赖，这多个依赖可能具有一定独立性。操作符级别多线程，不能同时处理不同的依赖。所以需要反向传播中的多线程。

Challenges encountered and solutions

感觉这次很难像 week1 的内容一样列出来一条条问题，加上时间原因，就暂时概括性地描述困难和解决方法了。

首先读 tutorial 之后我还是没有理解 autograd 是在做什么，不清楚整个过程的目的是什么。看了几个 Youtube 讲解反向传播和 autograd 的视频才大概理解流程。然后重新读了几遍 tutorial 。开始写之前我花了一个晚上在一个交互式 python 学习网站速通了一下 python 。

虽然对整个过程有个模糊的理解，但按照 tutorial 的顺序写 part0 部分内容的时候还是非常茫然，第一遍写的时候基本就是提示让写什么我就写什么（还不一定正确），也空了一些部分。后续调试第一个 clone 测试点的时候对 part0 进行了非常大的修改。继续往后基本就走上正轨了，part0 和其他部分对照着调试。

python 的自动 traceback 还是挺好的，至少知道具体崩溃原因和崩溃前在什么位置。不过大多数时候这些信息不够我确定代码逻辑上是哪里出错，所以还是要手动加很多输出调试。由于第一遍代码写得太 shi 了，感觉调试能力有了显著进步（也可能只是更耐心/犟了）……

btw 其实不少错误来自 tensor.cc ，第一周的测试点太弱没测出来。这周经常回去改 C++ 代码，出错的主要是 MatMul 和 squeeze。以及发现 tensor.cc 的封装实在是烂，后面有空再去调整一下。

写完这周内容我感觉对 python 语言（copy, decorator 等）、张量操作和反向传播算法等都有了更深的理解，代码能力也有所提升。

调试过程中助教们给了我很大帮助（包括解释代码逻辑、 python 的 copy 和赋值逻辑等），也很有耐心，对于我的一些病急乱投医式的问题也做出了解答。非常感谢助教们的帮助！