Torch Week1 Report

蒋欣桐

Grade summary form `grade_all.py`

```
================================================
all: grader summary :
score: 960/960
# passed: 103
# failed: 0
all tests passed!
================================================
```

Challenges encountered and solutions

`pybind` 编译报错问题

解决

- 添加路径但没有效果
- 询问助教，发现虽然编译器报错但是可以在 `clownpiece` 目录下运行 `pip install e .`
- 怀疑是 g++ 编译路径没有加上 `pybind11`
- `.bashrc` 中加入了对应路径，没用
- 作罢，似乎也不影响什么

`python` 和 `C++` bind 起来时调用流程不清楚，输出调试非常奇怪

解决

- 发现其实是 `python` 部分的输出流没有刷新导致输出位置奇怪，加 `sys.stdout.flush()` 和 `flush=True` 就好了

仓库同步问题

解决

- 询问助教
- 尝试自己翻找其他仓库中对应文件，检查同步性

收获

版本控制和仓库同步真的很重要（

写涉及 broadcast 的二元运算符遇到的问题（以 + 为例）

问题

做 element-wise 操作时应该先把 lhs 和 rhs 广播成同一个 shape。
我会：

```
auto [l, r] = lhs.broadcast(lhs, rhs);
```

一开始的实现中我接下来就直接 return：

```
return l.apply_binary_op([](dtype a, dtype b) -> dtype {
    return a + b;
}, r);
```

然而在 Tensor apply_unary_op(std::function<dtype(dtype)> op) const; 函数中，我的 result Tensor 永远是根据 *this ，即 l 来创建。如果是 lhs 被 broadcast ，那么 result Tensor 的物理内存就会比期望的少，会导致内存读取错误。（本质上还是当时写的时候对 broadcast 理解不到位。）

解决

把 Tensor apply_unary_op(std::function<dtype(dtype)> op) const; 改成 Tensor apply_binary_op(std::function<dtype(dtype, dtype)> op, const Tensor& rhs, const shape_t& sp) const; , result Tensor 由 sp 构造。

收获

逻辑上张量的元素个数和物理内存上可以不同，broadcast 中主要体现在 stride_ 。

写 matmul 函数遇到的2个问题

问题

一开始没看懂四类分别要做什么。
两个张量都至少 2 维的情况下，没懂怎么对"每个"二维矩阵做矩阵乘法。

解决

- 先把后面的 reduction and shape manipulation 写完，可以通过调用对应函数解决除了"两个张量都至少 2 维"之外的三种情况。
- 把"两个张量都至少 2 维"情况封装。把矩阵乘法封装。每次分别从原张量中切下一块二维矩阵（通过计算 offset ），做完矩阵乘法之后"粘贴"回结果张量。不知道这样是否会比较繁琐。

收获

封装和组合的实践（包括后面把下标检查封装了一下）。

写 sum 等函数遇到的问题

问题

主要是原来下标到新下标的映射。

解决

没想到什么比较优雅的实现，直接对每个旧下标（逻辑上和物理上）进行暴力计算映射到新下标（逻辑上和物理上）。具体而言是通过类似

```
 /* Update indices */
for (int d = shape_.size() - 1; d >= 0; --d) {
      if (++indices[d] < shape_[d]) {
         break;
      } else {
         indices[d] = 0;
      }
}
```

的方法更新逻辑下标，通过类似

```
int result_index = 0;
int stride = 1;
for (int d = ndim - 1; d >= 0; --d) {
      if (d == dim) {
      continue;
      }
      int coord = indices[d];
      result_index += coord * stride;

      /* stride in new shape */
      int dim_in_new_shape = (d > dim && !keepdims) ? d - 1 : d;
      if (dim_in_new_shape > 0) {
         stride *= new_shape[dim_in_new_shape];
      }
}

result.storage_[result_index] += data_at(i);
```

的方法更新物理下标 `result_index`。