Lecture 5: ML Frameworks

CS4787/5777 — Principles of Large-Scale Machine Learning Systems

Continuing from last time: Reverse-Mode AD

- Fix one output ℓ over $\mathbb{R}$
- Compute partial derivatives $\frac{\partial \ell}{\partial y}$ for each value y
- Need to do this by going *backward* through the computation

"Deep Learning" ML Frameworks

Classical Core Components:

- Numerical linear algebra library
- Hardware support (e.g. GPU)
- Backpropagation engine
- Library for expressing deep neural networks

All embedded in a high-level language

- Usually python.

Numerical Linear Algebra

You've already seen and used this sort of thing: NumPy.

- Arrays are objects "owned" by the library
- Any arithmetic operation on these objects goes through the library
  - The library calls an optimized function to compute the operation
  - This happens outside the python interpreter
  - Control is returned to python when the function finishes
  - By default you're only going to be running one such function at a time.

Numerical Linear Algebra: More Details

- Arrays are mutable
- Multiple references can exist!

Numerical Linear Algebra On-Device

- The simplest version of this is essentially a "copy" of NumPy for each sort of hardware we want to run on. This contains a copy of every function we want to support for each type of hardware.
  - e.g. one copy that runs on the CPU, one copy that runs on the GPU
- Arrays are located explicitly on one device
  - in PyTorch, you move them with `x.to("device_name")`
- When we try to call a function, the library checks where the inputs are located
  - if they're all on one device, it calls that device's version of the function
  - if they're not all on the same device, it raises an exception

Eager Execution vs Graph Execution

When we manifest a node of the compute graph, we can either:

- (eager) compute and manifest the value at that node immediately
- (graph) just manifest the node
  - need to call some function to compute the forward pass later

This was the classic distinction between TensorFlow and PyTorch

In [1]:
```
import torch
```

In [3]:
```
x = torch.ones((1,3,4))
```

In [6]:
```
x = torch.ones(())
```

In [7]:
```
x
```

Out[7]:
```
tensor(1.)
```

In [46]:
```
x = torch.ones(())
x.requires_grad = True
u = (x + 2)
y = u.square()  # (x + 2)^2 --> 2 * (1 + 2) = 6
```

Advantages of eager mode (compute values & manifest graph at the same time):

- much better for value-based debugging!
- varying shapes
- less complicated
- condition on values

Advantages of lazy mode/graph mode (manifest graph first, then compute values):

- heavier static optimization
- better for shape-based debugging

- could use less memory
- graph overhead less/amortized

In [65]:
```
import time
```

In [86]:
```
N**2 / (1024**3)
```

Out[86]:
```
0.25
```

In [84]:
```
N = 1024 * 16
X = torch.randn(N,N)
Y = torch.randn(N,N)
```

In [90]:
```
begin = time.time()
Z = X + Y
print(Z[0,0])
end = time.time()
print(f"elapsed: {(end - begin) * 1000} ms")
```

```
tensor(-1.4525)
elapsed: 60.443878173828125 ms
```

In [91]:
```
X_mps = X.to("mps")
Y_mps = Y.to("mps")
begin = time.time()
Z_mps = X_mps + Y_mps
print(Z_mps[0,0])
end = time.time()
print(f"elapsed: {(end - begin) * 1000} ms")
```

```
tensor(-1.4525, device='mps:0')
elapsed: 56.93316459655762 ms
```

In [83]:
```
554.5499324798584/ 118.25203895568848
```

Out[83]:
```
4.689559160055244
```

In [89]:
```
4210.312843322754/985.3289127349854
```

Out[89]:
```
4.273002434929221
```

In [98]:
```
U = torch.ones((1))
```

In [99]:
```
U_mps = U.to("mps")
```

In [100…
```
U
```

Out[100]:
```
tensor([1.])
```

In [101…
```
U_mps[0] = 3
```

In [102…
```
U
```

Out[102]:
```
tensor([1.])
```

In [103…
```
U_mps
```
Out[103]:
```
tensor([3.], device='mps:0')
```

In [104…
```
U_mps.cpu()
```
Out[104]:
```
tensor([3.])
```

In [105…
```
U_mps.to("cpu")
```
Out[105]:
```
tensor([3.])
```

In [106…
```
U = torch.ones((5,5,5),device="mps")
```

In [111…
```
(N ** 3)/(1024**3)
```
Out[111]:
```
4096.0
```

In [113…
```
U = torch.ones((N,N,N),device="meta")
```

In [115…
```
torch.ones((10000,20000),device="meta") @ torch.ones((20000,30000),device="meta
```
Out[115]:
```
tensor(..., device='meta', size=(10000, 30000))
```

In [116…
```
torch.ones((10000,20000),device="meta") @ torch.ones((30000,30000),device="meta
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Cell In[116], line 1
----> 1 torch.ones((10000,20000),device="meta") @ torch.ones((30000,30000),dev
ice="meta")

File /opt/anaconda3/lib/python3.9/site-packages/torch/_meta_registrations.py:4
48, in meta_mm(a, b)
    446 N, M1 = a.shape
    447 M2, P = b.shape
--> 448 check(M1 == M2, lambda: "a and b must have same reduction dim")
    449 return a.new_empty(N, P)

File /opt/anaconda3/lib/python3.9/site-packages/torch/_prims_common/__init__.p
y:1563, in check(b, s, exc_type)
   1556 """
   1557 Helper function for raising an error_type (default: RuntimeError) if a
boolean condition fails.
   1558 Error message is a callable producing a string (to avoid wasting time
   1559 string formatting in non-error case, and also to make it easier for to
rchdynamo
   1560 to trace.)
   1561 """
   1562 if not b:
-> 1563     raise exc_type(s())

RuntimeError: a and b must have same reduction dim
```

In [119…
```
torch.ones((1000000,2000000),device="meta")
```

Out[119]:
```
tensor(..., device='meta', size=(1000000, 2000000))
```

In [ ]:
```
torch.ones((1000000,2000000)).to("meta")
```

In [1]:
```
import torch
```

In [3]:
```
X = torch.ones((1000,2000),device="meta")
```

In [4]:
```
X.nonzero()
```

```
---------------------------------------------------------------------------
NotImplementedError                       Traceback (most recent call last)
Cell In[4], line 1
----> 1 X.nonzero()

NotImplementedError: Could not run 'aten::nonzero' with arguments from the 'Me
ta' backend. This could be because the operator doesn't exist for this backen
d, or was omitted during the selective/custom build process (if using custom b
uild). If you are a Facebook employee using PyTorch on mobile, please visit ht
tps://fburl.com/ptmfixes for possible resolutions. 'aten::nonzero' is only ava
ilable for these backends: [CPU, MPS, BackendSelect, Python, FuncTorchDynamicL
ayerBackMode, Functionalize, Named, Conjugate, Negative, ZeroTensor, ADInplace
OrView, AutogradOther, AutogradCPU, AutogradCUDA, AutogradHIP, AutogradXLA, Au
togradMPS, AutogradIPU, AutogradXPU, AutogradHPU, AutogradVE, AutogradLazy, Au
togradMeta, AutogradMTIA, AutogradPrivateUse1, AutogradPrivateUse2, AutogradPr
ivateUse3, AutogradNestedTensor, Tracer, AutocastCPU, AutocastCUDA, FuncTorchB
atched, FuncTorchVmapMode, Batched, VmapMode, FuncTorchGradWrapper, PythonTLSS
napshot, FuncTorchDynamicLayerFrontMode, PythonDispatcher].

CPU: registered at /Users/runner/work/pytorch/pytorch/pytorch/build/aten/src/A
Ten/RegisterCPU.cpp:31034 [kernel]
MPS: registered at /Users/runner/work/pytorch/pytorch/pytorch/build/aten/src/A
Ten/RegisterMPS.cpp:22748 [kernel]
BackendSelect: fallthrough registered at /Users/runner/work/pytorch/pytorch/py
torch/aten/src/ATen/core/BackendSelectFallbackKernel.cpp:3 [backend fallback]
Python: registered at /Users/runner/work/pytorch/pytorch/pytorch/aten/src/ATe
n/core/PythonFallbackKernel.cpp:144 [backend fallback]
FuncTorchDynamicLayerBackMode: registered at /Users/runner/work/pytorch/pytorc
h/pytorch/aten/src/ATen/functorch/DynamicLayer.cpp:491 [backend fallback]
Functionalize: registered at /Users/runner/work/pytorch/pytorch/pytorch/aten/s
rc/ATen/FunctionalizeFallbackKernel.cpp:280 [backend fallback]
Named: registered at /Users/runner/work/pytorch/pytorch/pytorch/aten/src/ATen/
core/NamedRegistrations.cpp:7 [backend fallback]
Conjugate: registered at /Users/runner/work/pytorch/pytorch/pytorch/aten/src/A
Ten/ConjugateFallback.cpp:17 [backend fallback]
Negative: registered at /Users/runner/work/pytorch/pytorch/pytorch/aten/src/AT
en/native/NegateFallback.cpp:19 [backend fallback]
ZeroTensor: registered at /Users/runner/work/pytorch/pytorch/pytorch/aten/src/
ATen/ZeroTensorFallback.cpp:86 [backend fallback]
ADInplaceOrView: fallthrough registered at /Users/runner/work/pytorch/pytorch/
pytorch/aten/src/ATen/core/VariableFallbackKernel.cpp:63 [backend fallback]
AutogradOther: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/
csrc/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradCPU: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/cs
rc/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradCUDA: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/c
src/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradHIP: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/cs
rc/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradXLA: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/cs
rc/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradMPS: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/cs
rc/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradIPU: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/cs
rc/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradXPU: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/cs
rc/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradHPU: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/cs
rc/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradVE: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/csr
```

```
c/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradLazy: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/c
src/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradMeta: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/c
src/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradMTIA: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/c
src/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradPrivateUse1: registered at /Users/runner/work/pytorch/pytorch/pytorch/
torch/csrc/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradPrivateUse2: registered at /Users/runner/work/pytorch/pytorch/pytorch/
torch/csrc/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradPrivateUse3: registered at /Users/runner/work/pytorch/pytorch/pytorch/
torch/csrc/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
AutogradNestedTensor: registered at /Users/runner/work/pytorch/pytorch/pytorc
h/torch/csrc/autograd/generated/VariableType_0.cpp:15256 [autograd kernel]
Tracer: registered at /Users/runner/work/pytorch/pytorch/pytorch/torch/csrc/au
tograd/generated/TraceType_0.cpp:16728 [kernel]
AutocastCPU: fallthrough registered at /Users/runner/work/pytorch/pytorch/pyto
rch/aten/src/ATen/autocast_mode.cpp:487 [backend fallback]
AutocastCUDA: fallthrough registered at /Users/runner/work/pytorch/pytorch/pyt
orch/aten/src/ATen/autocast_mode.cpp:354 [backend fallback]
FuncTorchBatched: registered at /Users/runner/work/pytorch/pytorch/pytorch/ate
n/src/ATen/functorch/BatchRulesDynamic.cpp:64 [kernel]
FuncTorchVmapMode: fallthrough registered at /Users/runner/work/pytorch/pytorc
h/pytorch/aten/src/ATen/functorch/VmapModeRegistrations.cpp:28 [backend fallba
ck]
Batched: registered at /Users/runner/work/pytorch/pytorch/pytorch/aten/src/ATe
n/LegacyBatchingRegistrations.cpp:1073 [backend fallback]
VmapMode: fallthrough registered at /Users/runner/work/pytorch/pytorch/pytorc
h/aten/src/ATen/VmapModeRegistrations.cpp:33 [backend fallback]
FuncTorchGradWrapper: registered at /Users/runner/work/pytorch/pytorch/pytorc
h/aten/src/ATen/functorch/TensorWrapper.cpp:210 [backend fallback]
PythonTLSSnapshot: registered at /Users/runner/work/pytorch/pytorch/pytorch/at
en/src/ATen/core/PythonFallbackKernel.cpp:152 [backend fallback]
FuncTorchDynamicLayerFrontMode: registered at /Users/runner/work/pytorch/pytor
ch/pytorch/aten/src/ATen/functorch/DynamicLayer.cpp:487 [backend fallback]
PythonDispatcher: registered at /Users/runner/work/pytorch/pytorch/pytorch/ate
n/src/ATen/core/PythonFallbackKernel.cpp:148 [backend fallback]
```

In [ ]:

```
```