PyTorch Tutorial Notes

Sunday, July 17, 2022 1:51 PM

Pytorch makes it easy to utilize GPU

Tensors

- identical to numpy arrays
  - generic tool for scientific computing
  - but also can utilize GPU

Autograd

- automatic differentiation to automate backward passes
- forward pass defines a computational graph
  - nodes are tensors, connecting edges are functions

New Autograd Functions

- to define a new operation that we can do backprop on, just need to define forward pass function, and backward pass function (see class torch.autograd.Function)

PyTorch vs TensorFlow

TensorFlow - static graphs
- graph is defined once and executed over multiple times
- advantage: can optimize graph up front

PyTorch - Dynamic Graphs
- graph is built repeatedly whenever we need it
- allows for simpler control flow in model

Pytorch: nn

- like Keras for TF, nn has a set of modules, roughly the same as neural network layers

example layers (Ex.):
- torch.nn.Sequential()
- torch.nn.Linear()
- torch.nn.ReLU()

- also defines a bunch of useful loss functions for neural networks

Pytorch: optim

- this package abstracts the idea of an optimization algorithm
- Ex. Common optimization functions such as Adam, RMSProp, AdaGrad

Pytorch: Custom nn Modules

- can define your own modules in addition to using predefined ones
- subclass nn.Module and defining forward function

Pytorch: Control Flow + Weight Sharing

- can have network that changes on each forward step, but we can still conduct backprop after each step