Cheatography

Pytorch入门 Cheat Sheet

by Yuxuejie0912 via cheatography.com/123584/cs/23342/

Pytorch是什么？

PyTorch是一个基于 Torch 的 Python 开源机器学习库。具有强大的 GPU 加速的张量计算（运算机制类似于 NumPy），包含自动求导系统，简洁优雅，大佬支持，新手友好。

Pytorch神经网络示例

```
import torch
import torch.nn as nn
from torch.autograd import Variable
import torch.nn.functional as F
```

```
model = nn.Sequential(nn.Linear(2, 10),
nn.ReLU(),
nn.Linear(10, 1), nn.Sigmoid() ) # 搭建网络
```

```
if torch.cuda.is_available():
    model = model.cuda() #判断GPU的cuda计算是否可用
```

```
optimzer = torch.optim.SGD(model.parameters(), lr=0.05) # 优化器
loss_func = nn.MSELoss() # 损失函数
```

```
out = model(x)
loss = loss_func(out, y)
optimzer.zero_grad() # 清除梯度
loss.backward() # 反向传播
optimzer.step() # 梯度更新
```

Tensor and Variable

Tensor运算机制类似于Numpy，支持GPU加速

Variable是对Tensor的封装，支持神经网络图计算，具备三个属性 .data, .grad, .grad_fn

```
x_tensor = torch.randn(5, 5)
x_var_regular = Variable(x_tensor, requires_grad=True)
x_var_volatile = Variable(x_tensor, volatile=True)
```

自动求导

自动构造反向求导流图，通过requires_grad或volatile判断Variable参数是否需要更新，常用于冻结模型或inference阶段，节省内存

torch.nn.functional

torch.nn：可看作对nn.functional的类包装，同时继承了nn.Module相关属性和方法，适合构建模型；

torch.nn.functional：直接使用def function()定义，使用灵活，但需要手动传入weight，不利于代码复用。

```
conv = nn.Conv2d(3, 64, 3, 2, 1)
```

```
output = nn.functional.conv2d(inputs, weight, bias, padding=1)
```

torch.nn

专为神经网络设计的模块化接口

激活函数 torch.nn.ReLU(inplace=False)

损失函数 torch.nn.BCELoss(weight=None, size_average=True, reduce=None, reduction='mean')

全连接层 torch.nn.Linear(in_features, out_features, bias=True)

卷积层 torch.nn.Conv2d(in_channels, out_channels, kernel_size, stride=1, padding=0, dilation=1, groups=1, bias=True)

卷积层代码示例：

```
input = Variable(torch.randn(32, 3, 28, 28))
conv1 = nn.Conv2d(in_channels=3, out_channels=10, kernel_size=3, stride=1, padding=1)
conv2 = nn.Conv2d(in_channels=10, out_channels=128, kernel_size=3, stride=1, padding=1)
```

标准化层 torch.nn.BatchNorm2d(num_features, eps=1e-05, momentum=0.1, affine=True)

池化层 torch.nn.MaxPool2d(kernel_size, stride=None, padding=0, dilation=1, return_indices=False, ceil_mode=False)

torch.nn.AvgPool2d(kernel_size, stride=None, padding=0, ceil_mode=False, count_include_pad=True)

Dropout层 torch.nn.Dropout(p=0.5, inplace=False)

torch.nn (cont)

插值函数 torch.nn.Upsample(size=None, scale_factor=None, mode='nearest', align_corners=None)

容器 torch.nn.Module torch.nn.Sequential(* args)

nn.Module是所有神经网络的基类，定义任一网络应继承该类。

```
model.modules #返回一个包含当前模型所有模块的迭代器
model.state_dict() #返回字典，保存module的所有状态
model.forward() #前向计算，自动调用，所有子类必须重写
model.train() #模型在train和evaluation模式的切换
model.eval() #仅当存在BN层和dropout层时有影响
```

nn.Sequential(* args) 是一个时序容器，modules 会以他们传入的顺序被添加到容器中。

数据预处理

```
class torchvision.transforms #实现数据增广变换
```

```
# 使用Compose将变换操作串联起来
transforms.Compose([ transforms.CenterCrop(224),
transforms.RandomHorizontalFlip(),
transforms.ToTensor(),
transforms.Normalize(0.5, 0.2) ])
```

模型的保存和加载

```
torch.save({ 'epoch': epoch,
'model_state_dict': model.state_dict(),
'optimizer_state_dict': optimizer.state_dict(),
'loss': loss, ... }, PATH)
```

```
checkpoint = torch.load(PATH)
model.load_state_dict(checkpoint['model_state
optimizer.load_state_dict(checkpoint['optimizer
epoch = checkpoint['epoch'] loss = checkpoint[
```

By **Yuxuejie0912**

cheatography.com/yuxuejie0912/

Published 18th June, 2020.
Last updated 18th June, 2020.

模型优化 torch.optim

torch.optim.SGD(model.parameters(), lr = 0.01, momentum=0.9)

torch.optim.RMSprop(model.parameters(), lr=0.01, alpha=0.99, eps=1e-08, weight_decay=0, momentum=0, centered=False)

torch.optim.Adam(model.parameters(), lr=0.001, betas=(0.9, 0.999), eps=1e-08, weight_decay=0, amsgrad=False)

optimizer.zero_grad() # pytorch梯度backward累积而不是替换，每批次清零

optimizer.step() # 模型更新

数据集

```
# 使用pytorch内置数据集
import torchvision.datasets as datasets
cifar10 = datasets.CIFAR10()
torch.utils.data.DataLoader(cifar10, batch_size=args.batchSize, shuffle=True, num_workers=args.nThreads)
```

```
# 封装自定义数据集必须继承Dataset类
from torch.utils.data.dataset import Dataset
```

预训练模型

```
import torchvision.models as models
resnet18 = models.resnet18( )
alexnet = models.alexnet(pretrained=True)
vgg19 = models.vgg19(pretrained=True)
```

可视化

TensorboardX

```
from tensorboardX import SummaryWriter
with SummaryWriter(comment='LeNet') as w:
    w.add_graph(model, (input, ))
tensorboard --logdir runs
```

pytorchviz github项目

https://github.com/szagoruyko/pytorchviz