torch 笔记

徐世桐

1 import

```
from torch import nn
import torch.nn.functional as F
import torch.optim as optim
```

2 tensor 数据类型

`torch.arange`

`torch.arange(a)` 得到 Tensor$[0, 1, ..., \lfloor a \rfloor]$

`torch.arange(a, b)` 得到 Tensor$[a, a+1, ..., a+n]$，n 为整数且 $a+n<b$

`torch.arange(a, b, c)` 得到 Tensor$[a, a+c, ..., a+nc]$，n 为整数且 $a+nc<b$

`torch.from_numpy(NDArray)` 从 `NDArray` 创建 Tensor

`torch.mm(Tensor, Tensor)` tensor 矩阵乘法

`+-*/` 同 NDArray 使用广播机制

`Tensor.reshape()` 改变形状，**新形状元素数必须等于输入元素数**

`torch.random(MEAN, STD, SIZE*)`

`size=`$(x_1, x_2, ...)$ 限定输出张量形状

`mean=Tensor`, `std=Tensor/const` 当没有限定 size 时 `mean` 必为 float Tensor，形状和输出形状相同。

`mean=Tensor/const`, `std=Tensor/const` 当限定 size 后 `mean, std` 可为 const 或单个值的 Tensor

`torch.rand(SIZE*)`

得到 SIZE 形状的随机数张量，每一元素 $\in [0, 1)$。SIZE 无定义则得到 const 随机数

代替 torch.uniform 功能

`dataset = torch.utils.data.TensorDataset(样本 Tensor, 标签 Tensor)`

`dataiter = torch.utils.data.DataLoader(dataset, batch_size= 批量大小, shuffle=True)`

使用 torch 进行批量迭代

3 torch 神经网络

`net = nn.Sequential()`

神经网络定义：`net.add_module(' 层名', 层)`

层定义:

nn.Linear(输入节点数, 输出节点数) 定义全连接层

可使用层直接进行前向计算，训练函数中使用 [layer.weight, net.bias] 传入参数

net.weight/bias.data.fill_(值) 对层中所有权重/偏差赋值

nn.init.xavier_uniform(net.weight/.bias) 对层中所有权重/偏差使用 xavier 初始化

loss = nn.MSELoss() 平方代价函数

trainer = optim.SGD(net.parameters(), lr= 学习率) SGD 迭代函数

trainer.step() 进行迭代

net.parameters() 得到权重