精品：Pytorch深度学习建模流程总结

小数志 2022-03-18 12:00

编者荐语：

分享一篇PyTorch学习干货！

以下文章来源于AI有温度 ，作者AI工程师Tiger

AI有温度

人工智能爱好者的聚集地。专注于Python、机器学习、深度学习、CV、NLP的AI技术干货分享。

> AI因你而升温，记得加个星标哦！

大家好，我是泰哥。之前我已经讲解了 `Pytorch` 深度学习构建神经网络需要的所有知识点，本节就将所有的零碎知识点串联起来，帮助大家梳理神经网络训练的架构。

一般我们训练神经网络有以下步骤：

1. 导入库
2. 设置训练参数的初始值
3. 导入数据集并制作数据集
4. 定义神经网络架构
5. 定义训练流程
6. 训练模型

以下，我就将上述步骤使用代码进行注释讲解：

1 导入库

```
import torch
from torch import nn
from torch.nn import functional as F
from torch import optim
from torch.utils.data import DataLoader, DataLoader
import torchvision
import torchvision.transforms as transforms
```

2 设置初始值

```
# 学习率
lr = 0.15
# 优化算法参数
gamma = 0.8
# 每次小批次训练个数
bs = 128
# 整体数据循环次数
epochs = 10
```

3 导入并制作数据集

本次我们使用 `FashionMNIST` 图像数据集，每个图像是一个 `28*28` 的像素数组，共有10个衣物类别，比如连衣裙、运动鞋、包等。

注：初次运行下载需要等待较长时间。可直接在公众号后台回复【**潮装数据集**】直接获取。

```
# 导入数据集
mnist = torchvision.datasets.FashionMNIST(
    root = './Datastes'
    , train = True
    , download = True
    , transform = transforms.ToTensor())

# 制作数据集
batchdata = DataLoader(mnist
                       , batch_size = bs
                       , shuffle = True
                       , drop_last = False)
```

我们可以对数据进行检查：

```
for x, y in batchdata:
    print(x.shape)
    print(y.shape)
    break

# torch.Size([128, 1, 28, 28])
# torch.Size([128])
```

可以看到一个 `batch` 中有128个样本，每个样本的维度是 `1*28*28` 。

之后我们确定模型的输入维度与输出维度：

```
# 输入的维度
input_ = mnist.data[0].numel()
# 784

# 输出的维度
output_ = len(mnist.targets.unique())
# 10
```

4 定义神经网络架构

先使用一个128个神经元的全连接层，然后用 `relu` 激活函数，再将其结果映射到标签的维度，并使用 `softmax` 进行激活。

```
# 定义神经网络架构
class Model(nn.Module):
    def __init__(self, in_features, out_features):
        super().__init__()
        self.linear1 = nn.Linear(in_features, 128, bias = True)
        self.output = nn.Linear(128, out_features, bias = True)

    def forward(self, x):
        x = x.view(-1, 28*28)
        sigma1 = torch.relu(self.linear1(x))
        sigma2 = F.log_softmax(self.output(sigma1), dim = -1)
        return sigma2
```

5 定义训练流程

在实际应用中，我们一般会将训练模型部分封装成一个函数，而这个函数可以继续细分为以下几步：

1. 定义损失函数与优化器

2. 完成向前传播
3. 计算损失
4. 反向传播
5. 梯度更新
6. 梯度清零

在此六步核心操作的基础上，我们通常还需要对模型的**训练进度、损失值与准确度**进行监视。注释代码如下：

```
# 封装训练模型的函数
def fit(net, batchdata, lr, gamma, epochs):
# 参数：模型架构、数据、学习率、优化算法参数、遍历数据次数

    # 5.1 定义损失函数
    criterion = nn.NLLLoss()
    # 5.1 定义优化算法
    opt = optim.SGD(net.parameters(), lr = lr, momentum = gamma)

    # 监视进度：循环之前，一个样本都没有看过
    samples = 0
    # 监视准确度：循环之前，预测正确的个数为0
    corrects = 0

    # 全数据训练几次
    for epoch in range(epochs):
        # 对每个batch进行训练
        for batch_idx, (x, y) in enumerate(batchdata):
            # 保险起见，将标签转为1维，与样本对齐
            y = y.view(x.shape[0])

            # 5.2 正向传播
            sigma = net.forward(x)
            # 5.3 计算损失
            loss = criterion(sigma, y)
            # 5.4 反向传播
            loss.backward()
            # 5.5 更新梯度
            opt.step()
            # 5.6 梯度清零
            opt.zero_grad()

            # 监视进度：每训练一个batch，模型见过的数据就会增加x.shape[0]
            samples += x.shape[0]

            # 求解准确度：全部判断正确的样本量/已经看过的总样本量
            # 得到预测标签
            yhat = torch.max(sigma, -1)[1]
            # 将正确的加起来
            corrects += torch.sum(yhat == y)

            # 每200个batch和最后结束时，打印模型的进度
            if (batch_idx + 1) % 200 == 0 or batch_idx == (len(batchdata) - 1):
                # 监督模型进度
                print("Epoch{}:[{}/{} {: .0f}%], Loss:{:.6f}, Accuracy:{:.6f}".format(
                    epoch + 1
                    , samples
                    , epochs*len(batchdata.dataset)
                    , 100*samples/(epochs*len(batchdata.dataset))
                    , loss.data.item()
                    , float(100.0*corrects/samples)))
```

6 训练模型

```
# 设置随机种子
torch.manual_seed(51)

# 实例化模型
net = Model(input_, output_)

# 训练模型
fit(net, batchdata, lr, gamma, epochs)
# Epoch1:[25600/600000  4%], Loss:0.524430, Accuracy:69.570312
# Epoch1:[51200/600000  9%], Loss:0.363422, Accuracy:74.984375
# ......
# Epoch10:[600000/600000  100%], Loss:0.284664, Accuracy:85.771835
```

现在我们已经用 `Pytorch` 训练了最基础的神经网络，并且可以查看其训练成果。大家可以将代码复制进行运行！

虽然没有用到复杂的模型，但是我们在每次建模时的基本思想都是一致的。

END

相关阅读：
- 写在1024：一名数据分析师的修炼之路
- 数据科学系列：sklearn库主要模块功能简介
- 数据科学系列：seaborn入门详细教程
- 数据科学系列：pandas入门详细教程
- 数据科学系列：matplotlib入门详细教程
- 数据科学系列：numpy入门详细教程

喜欢此内容的人还喜欢

简单实现三维卷积动作识别
我不爱机器学习

【任务分配】基于蚁群算法实现无人机任务分配附matlab代码
天天Matlab

三维卷积如何将时序和空间信息融合?
我不爱机器学习