训练模型

超参数

训练过程中不会改变的值。用于定义训练模式时的参数。

```
# 纪元
EPOCH = 5
# 学习率
LEARNING_RATE = 0.03
# 批次大小
BATCH_SIZE = 10
```

- EPOCH：纪元
  - 将所有采集到的数据，全部通过模型训练，所有数据被便利完一次，称为一个纪元
- LEARNING_RATE：学习率
  - 决定训练时权重改变的幅度(梯度下降幅度)。
  - 学习率越大，改变幅度变大，训练速度变快，准确率变低
  - 学习率越小，改变幅度变小，训练速度变慢，准确率变高
  - 通常在0.001~1之间
- BATCH_SIZE：批次大小
  - 训练模型时，以多少个样本为一组进行训练

```
# 超参数
import torch.optim

EPOCH = 3
LEARNING_RATE = 0.001
BATCH_SIZE = 10

# 加载数据集
import torchvision
from torch.utils.data import DataLoader
from torchvision.transforms import ToTensor

# 创建张量转换器对象
to_tensor = ToTensor()
# 加载训练集，同时转换为张量格式
train_set = torchvision.datasets.MNIST("", transform=to_tensor, train=True, download=False)
# 转换为PyTorch训练的特定数据集格式
# 这里每10张图片作为一个批次，每次打乱
train_loader = DataLoader(train_set, batch_size=BATCH_SIZE, shuffle=True)

# iterator = iter(train_loader)
# images, labels = next(iterator)
# print(images, labels)
```

```python
# 创建模型、优化器、损失函数
from FNN_MODEL import MyNet

# 创建模型
model = MyNet()
# 创建优化器
optimizer = torch.optim.Adam(model.parameters(), lr=LEARNING_RATE)
# 损失函数
loss_fn = torch.nn.NLLLoss()

# 训练训练
for epoch in range(EPOCH):
    print(f"当前纪元：{epoch + 1}/{EPOCH}")
    # 开启训练
    model.train(True)
    # 遍历一个批次的10张图片
    for i, (images, labels) in enumerate(train_loader):
        # 调整权重
        # 梯度清零
        optimizer.zero_grad(),
        # 前向传播计算损失
        outputs = model.forward(images)
        # 通过损失函数计算损失
        loss = loss_fn(outputs, labels)
        # 反向传播调节参数
        loss.backward()
        # 更新参数
        optimizer.step()
        print(f"当前批次：{i + 1}/{len(train_loader)}，当前损失值：{loss}")

# 保存模型参数
torch.save(model.state_dict(), "FNN_MODEL_PARARM.pt")
```