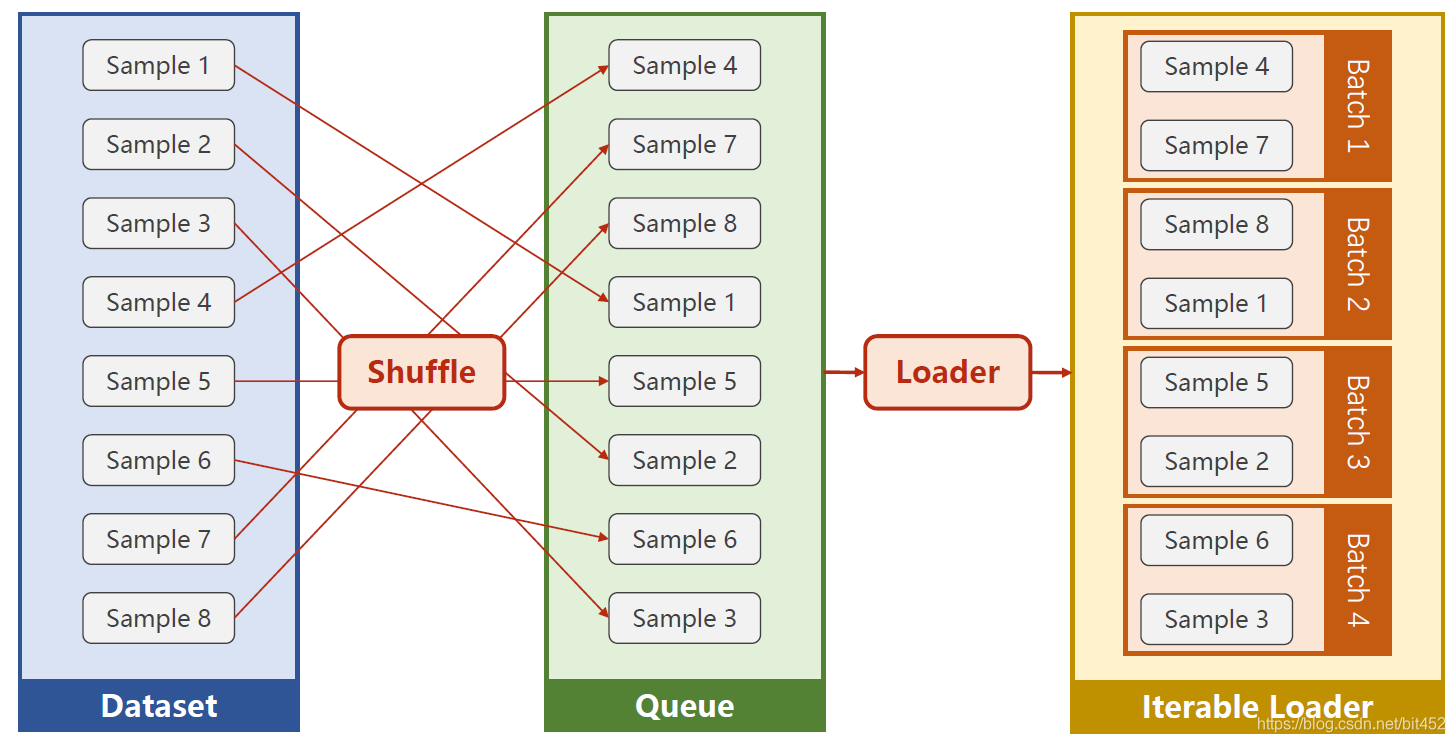
PyTorch 深度学习实践 第8讲

第8讲  加载数据集 源代码

B站 刘二大人 ，传送门PyTorch深度学习实践——加载数据集

说明：1、DataSet 是抽象类，不能实例化对象，主要是用于构造我们的数据集

          2、DataLoader 需要获取DataSet提供的索引[i]和len;用来帮助我们加载数据，比如说做shuffle(提高数据集的随机性)，batch\_size,能拿出Mini-Batch进行训练。它帮我们自动完成这些工作。DataLoader可实例化对象。



代码说明：

1、需要mini\_batch 就需要import DataSet和DataLoader

2、继承DataSet的类需要重写init，getitem,len魔法函数。分别是为了加载数据集，获取数据索引，获取数据总量。

3、DataLoader对数据集先打乱(shuffle)，然后划分成mini\_batch。

4、len函数的返回值 除以 batch\_size 的结果就是每一轮epoch中需要迭代的次数。

5、inputs, labels = data中的inputs的shape是[32,8],labels 的shape是[32,1]。也就是说mini\_batch在这个地方体现的

6、diabetes.csv数据集老师给了下载地址，该数据集需和源代码放在同一个文件夹内。

|  |
| --- |
| import torch  import numpy as np  from torch.utils.data import Dataset  from torch.utils.data import DataLoader    # prepare dataset      class DiabetesDataset(Dataset):  def \_\_init\_\_(self, filepath):  xy = np.loadtxt(filepath, delimiter=',', dtype=np.float32)  self.len = xy.shape[0] # shape(多少行，多少列)  self.x\_data = torch.from\_numpy(xy[:, :-1])  self.y\_data = torch.from\_numpy(xy[:, [-1]])    def \_\_getitem\_\_(self, index):  return self.x\_data[index], self.y\_data[index]    def \_\_len\_\_(self):  return self.len      dataset = DiabetesDataset('diabetes.csv')  train\_loader = DataLoader(dataset=dataset, batch\_size=32, shuffle=True, num\_workers=2) #num\_workers 多线程      # design model using class      class Model(torch.nn.Module):  def \_\_init\_\_(self):  super(Model, self).\_\_init\_\_()  self.linear1 = torch.nn.Linear(8, 6)  self.linear2 = torch.nn.Linear(6, 4)  self.linear3 = torch.nn.Linear(4, 1)  self.sigmoid = torch.nn.Sigmoid()    def forward(self, x):  x = self.sigmoid(self.linear1(x))  x = self.sigmoid(self.linear2(x))  x = self.sigmoid(self.linear3(x))  return x      model = Model()    # construct loss and optimizer  criterion = torch.nn.BCELoss(reduction='mean')  optimizer = torch.optim.SGD(model.parameters(), lr=0.01)    # training cycle forward, backward, update  if \_\_name\_\_ == '\_\_main\_\_':  for epoch in range(100):  for i, data in enumerate(train\_loader, 0): # train\_loader 是先shuffle后mini\_batch  inputs, labels = data  y\_pred = model(inputs)  loss = criterion(y\_pred, labels)  print(epoch, i, loss.item())    optimizer.zero\_grad()  loss.backward()    optimizer.step() |

————————————————

版权声明：本文为CSDN博主「错错莫」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/bit452/article/details/109686474