GAN模型学习与复现

- 时间：2020.10.13-2020.10.18
- 语言：Python 3.6.8
  - 机器学习库：**PyTorch 1.6.1**
  - 绘图库：**Matplotlib**

GAN模型

生成式对抗网络（GAN, Generative Adversarial Networks ）是一种深度学习模型，是近年来复杂分布上**无监督学习**最具前景的方法之一。

模型框架中的两个核心模块：

- 判别模型D（Discriminative Model）
  - 需要输入变量，通过某种模型来进行预测
- 生成模型G（Generative Model）
  - 给定某种隐含信息，来随机产生观测数据

在训练过程中，通过两个模型的互相学习，从而产生产生越来越好的输出。生成模型G的目标是尽量产生真实的数据来欺骗判别模型D，而D的目标就是尽量把G生成的数据与真实数据区分开来。这样G和D就构成了一个动态的**博弈过程**。

最终在理想的情况下，博弈的结果是得到了一个生成式的模型G，它可以用来生成相应的数据。

模型实现

准备阶段

此模型是一个简单的GAN模型，在准备阶段需要考虑四个部分，包括数据集、随机噪音、生成器和判别器。

数据集R

本模型中采用自动生成的数据集R——高斯分布，将平均值和标准差作为输入，输出提供正确样本数据的图形，例子中使用4作为平均数和1.25作为标准差。

```
def get_distribution_sampler(mu, sigma):
    return lambda n: torch.Tensor(np.random.normal(mu, sigma, (1, n)))  # Gaussian
```

随机噪音I

它是生成器G的输入，我们设定是随机的，不过使用的是均匀分布，也就是G需要用非线性的方式来改造数据。

```
def get_generator_input_sampler():
    return lambda m, n: torch.rand(m, n)  # Uniform-dist data into generator
```

生成器G

生成器使用了前馈图来进行实现——两层隐层，三个线性映射。

```
class Generator(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(Generator, self).__init__()
        self.map1 = nn.Linear(input_size, hidden_size)
        self.map2 = nn.Linear(hidden_size, hidden_size)
        self.map3 = nn.Linear(hidden_size, output_size)

    def forward(self, x):
        x = F.elu(self.map1(x))
        x = F.sigmoid(self.map2(x))
        return self.map3(x)
```

判别器D

方法与生成器类似，这个例子中只会输出0或1的判别值，对应于真实数据和虚假数据。

训练阶段

参数设置

对于生成器G：

- 输入数据大小：1
- （？）复杂度：5
- 输出数据大小：1
- 学习率：1e-3
- 每一个epoch训练次数：20
- 激活函数：torch.sigmoid

对于判断器D：

- 输入数据大小：1
- （？）复杂度：10
- 输出数据：1或0
- 学习率：1e-3
- 每一个epoch训练次数：20
- 激活函数：torch.tanh

训练判别器D

- 首先使用真实的数据对判别器D进行训练，使用损失函数对判别器的预测值准确度进行估计，再使用pytorch中的bakward()函数进行反向传播
- 然后使用生成器G生成的假数据对判别器D进行训练，方法和上一步类似，但这里要注意的是，生成假数据要进行分离`G(d_gen_input).detach()`，防止使用了这次的假数据来训练G
- 每一遍epoch对D训练20次

训练生成器G

- 首先生成噪声数据l作为输入，根据输入生成一个虚假数据
- 将生成的虚假数据通过D进行检测，并使用损失函数进行判断，同样的使用backward()函数进行反向传播
- 每一遍epoch对G训练20次

测试结果

将训练遍数`num_epochs`设置为20000进行测试，引用matplotlib包将生成器G的生成结果以直方图的结果呈现，由图可见，G生成的数据已经很接近平均值为4的高斯分布了。