1. **实验目标**
2. 学习pytorch的使用方法
3. 尝试运行pytorch程序
4. 分析所选程序代码的运行结果
5. **实验内容**
6. 访问Github官网并查询关键词“pytorch”
7. 在所给pytorch的文件中选择“yunjey/pytorch-tutorial”
8. 找到代码并在pycharm中运行
9. 对运行结果进行分析
10. **实验过程**

首先我在pycharm中建立一个new project，然后登录pytorch官网在python中建立pytorch运行环境。由于代码中含有matplotlib库的使用，我又将matplotlib库加入到python中。在建立好运行环境后，我就分别将“yunjey/pytorch-tutorial”中不同的代码copy到pycharm中运行，对运行结果进行观察并对参数进行调试。

1. **实验结果及分析**

线性回归代码：

**import** torch
**import** torch.nn **as** nn
**import** numpy **as** np
**import** matplotlib.pyplot **as** plt

*# Hyper-parameters*input\_size = 1
output\_size = 1
num\_epochs = 60
learning\_rate = 0.001

*# Toy dataset*x\_train = np.array([[3.3], [4.4], [5.5], [6.71], [6.93], [4.168],
 [9.779], [6.182], [7.59], [2.167], [7.042],
 [10.791], [5.313], [7.997], [3.1]], dtype=np.float32)

y\_train = np.array([[1.7], [2.76], [2.09], [3.19], [1.694], [1.573],
 [3.366], [2.596], [2.53], [1.221], [2.827],
 [3.465], [1.65], [2.904], [1.3]], dtype=np.float32)

*# Linear regression model*model = nn.Linear(input\_size, output\_size)

*# Loss and optimizer*criterion = nn.MSELoss()
optimizer = torch.optim.SGD(model.parameters(), lr=learning\_rate)

*# Train the model***for** epoch **in** range(num\_epochs):
 *# Convert numpy arrays to torch tensors* inputs = torch.from\_numpy(x\_train)
 targets = torch.from\_numpy(y\_train)

 *# Forward pass* outputs = model(inputs)
 loss = criterion(outputs, targets)

 *# Backward and optimize* optimizer.zero\_grad()
 loss.backward()
 optimizer.step()

 **if** (epoch + 1) % 5 == 0:
 print(**'Epoch [{}/{}], Loss: {:.4f}'**.format(epoch + 1, num\_epochs, loss.item()))

*# Plot the graph*predicted = model(torch.from\_numpy(x\_train)).detach().numpy()
plt.plot(x\_train, y\_train, **'ro'**, label=**'Original data'**)
plt.plot(x\_train, predicted, label=**'Fitted line'**)
plt.legend()
plt.show()

*# Save the model checkpoint*torch.save(model.state\_dict(), **'model.ckpt'**)

运行结果1：

Epoch [5/60], Loss: 0.7007

Epoch [10/60], Loss: 0.3863

Epoch [15/60], Loss: 0.2589

Epoch [20/60], Loss: 0.2073

Epoch [25/60], Loss: 0.1864

Epoch [30/60], Loss: 0.1779

Epoch [35/60], Loss: 0.1745

Epoch [40/60], Loss: 0.1731

Epoch [45/60], Loss: 0.1725

Epoch [50/60], Loss: 0.1723

Epoch [55/60], Loss: 0.1722

Epoch [60/60], Loss: 0.1721



图1 线性回归曲线1

运行结果2：

Epoch [5/60], Loss: 5.0693

Epoch [10/60], Loss: 2.3013

Epoch [15/60], Loss: 1.1796

Epoch [20/60], Loss: 0.7248

Epoch [25/60], Loss: 0.5401

Epoch [30/60], Loss: 0.4650

Epoch [35/60], Loss: 0.4341

Epoch [40/60], Loss: 0.4213

Epoch [45/60], Loss: 0.4157

Epoch [50/60], Loss: 0.4131

Epoch [55/60], Loss: 0.4116

Epoch [60/60], Loss: 0.4107



图2 线性回归曲线2

这部分代码由于需要画图，使用了matplotlib库函数，其大致思路是先设置参数，然后将数据矩阵输入进去，然后建立一个线性回归曲线，并计算损失与优化，然后画出线性回归曲线并将数据用点表示。图像中给出了优化过后的线性回归曲线。图1与图2分别是同一个代码的不同运行结果，其结果有较大的差别。

1. **实验中遇到的问题及解决方案**
2. 怎样建立pytorch运行环境

这是我遇到的第一个问题，也是一个最关键的问题。起初是在Github上搜索解决办法，但由于是英文版，比较难理解，后来便在百度上搜索，找到了解决办法。我在pytorch官网上找到下载代码，打开“cmd”，将代码复制进去便可在python中建立pytorch环境。

1. 建立matplotlib库函数运行环境。

同样在百度找到了解决办法，下载补丁后在python中建立库函数。

1. num\_epochs的疑问

我改变了num\_epochs的参数，发现所得出的线性回归曲线图像是不同的，且每次运行的图像都会发生改变。通过阅读代码，我把它理解为优化线性回归曲线的次数。每次优化后计算的损失都不一样，因此线性回归曲线也会不同。这也体现机器学习有一定的误差，不够稳定。

1. **心得体会**

通过这次大作业，我以实践的方式体会到了机器学习的奇妙，感受到了它无穷的魅力。只要敢想敢做，就能利用机器学习的方式得到创新。这次实验我尝试了许多种代码，它们的结果各不相同。很多代码的编写思路是值得我学习的。这次机器学习的课程也开阔了我的眼界，让我对人工智能有了更深的了解，受益匪浅。