浙江大学

本科实验报告

姓名：严轶凡

学院：控制科学与工程学院

系：自动化

专业：自动化（控制）

学号：3200100917

指导教师：周建光

年　　月　　日

浙江大学 实验报告

一、实验目的和要求（必填）

对数据进行预处理，接着使用数据分析方法，建立基础特征，进一步构建线性回归模型，且基于新数据验证模型效果。

二、实验内容和原理（必填）

实验内容：

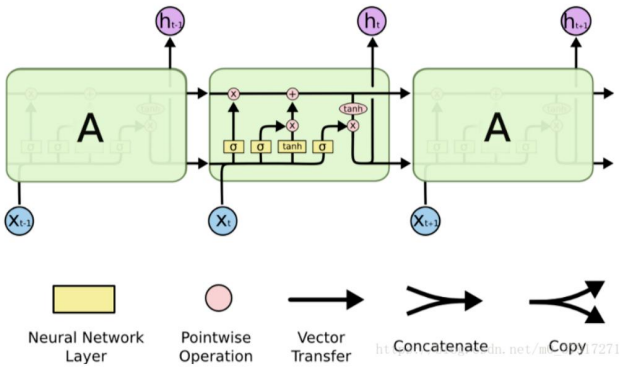
输入某股票前 14 个交易日的收盘价，预测下一个交易日的收盘价。

实验指标为平均绝对百分比误差（MAPE）和平均绝对误差（MAE）。

实验原理：

为了实现题目要求中的时间序列分析，本实验采用 LSTM 神经网路进行时间序列预测。

LSTM 具有神经网络的重复模块链的形式。只是在 RNN 的基础上，每个重复模块增加了三个神经网络层，如下图所示：



三个神经网络层分别代表 LSTM 的三个门（遗忘门、记忆门、输出门）。

遗忘门（forget gate）：决定了上一时刻的单元状态 c_t-1 有多少保留到当前时刻 c_t；

输入门（input gate）：决定了当前时刻网络的输入 x_t 有多少保存到单元状态 c_t；

输出门（output gate）：控制单元状态 c_t 有多少输出到 LSTM 的当前输出值 h_t；

使用 LSTM 神经网络能够较好的对时间序列进行分析和预测，故本题中采用 LSTM 神经网络进行预测。

三、实验环境

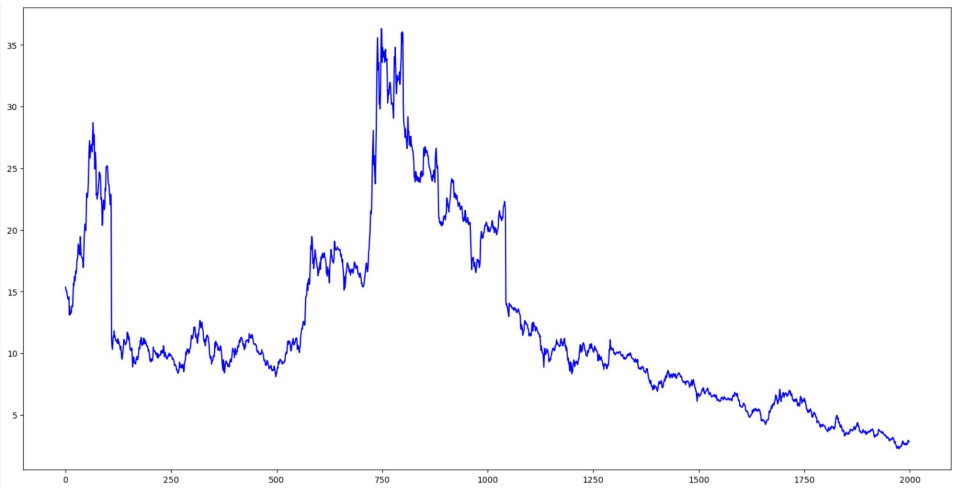
基于 Python 的 Pandas、Numpy、Scikit-learn 等库进行相关特征处理，使用 Pytorch 框架建立深度学习模型。

四、操作方法与实验步骤

装订线

（一）数据集

数据集由网上的相关平台获取，训练集给出了五十几支股票的情况。数据以 `npy` 格式给出，名称为`train_data.npy` 。



（二）数据处理

首先对数据集合进行分割，其中训练集用于训练，校验集用于检验模型训练情况，测试集用于测试模型效果。本次实验中采用 7: 2: 1 的比例划分训练集、校验集和测试集。

将数据转换为`Pytorch`数据集的形式，并对 x、y 进行归一化：

```
    x_scaled = scaler.transform(x.reshape(-1,1)).reshape(-1,14)
    y_scaled = scaler.transform(y)

    x_scaled = torch.tensor(x_scaled, dtype=torch.float32)
    y_scaled = torch.tensor(y_scaled, dtype=torch.float32)
```

使用`DataLoader`加载数据集：

```
    train_data = MyDataset(x_train,y_train)
    valid_data = MyDataset(x_val,y_val)
    test_data = MyDataset(x_test,y_test)
```

（三）建立模型

```
class LSTMNet(torch.nn.Module):
    def __init__(self, num_inputs, num_hiddens, num_outputs):
        super(LSTMNet, self).__init__()
        self.hidden_size = num_hiddens
        # RNN 层
        self.rnn = torch.nn.LSTM(
        input_size=num_inputs,
        hidden_size=num_hiddens,
        batch_first=True
        )
    # 线性层
    self.dense = torch.nn.Linear(self.hidden_size, 256)
    self.dense2 = torch.nn.Linear(256, num_outputs)
    # dropout 层，这里的参数指 dropout 的概率
    self.dropout = torch.nn.Dropout(0.3)
    self.dropout2 = torch.nn.Dropout(0.5)
    # ReLU 层
    self.relu = torch.nn.ReLU()
```

```
    # 前向传播函数，这是一个拼接的过程，使用大量变量是为了避免混淆，不做过多讲解
    def forward(self, x):
        x = x.view(1, len(x), -1)
        print(x.shape)
        # LSTM 层会传出其参数，这里用 _ 将其舍弃
        h, _ = self.rnn(x)
        h_r = h.reshape(-1, self.hidden_size)
        h_d = self.dropout(h_r)
        y = self.dense(h_d)
        drop_y = self.dropout2(y)
        a = self.relu(drop_y)
        y2 = self.dense2(a)
        return y2
```

（四）训练模型

```
# 判断 gpu 是否可用
use_gpu = torch.cuda.is_available()
# 使用均方根误差
loss = torch.nn.MSELoss()
model = LSTMNet(num_inputs=14, hidden_size=128, num_layers=2, num_outputs=1)
model.to('cuda')
optimizer = torch.optim.Adam(model.parameters(), lr=1e-3)
# 训练模型
model, (train_losses, valid_losses, train_maes, train_mapes, valid_maes, valid_mapes) = \
    train_model(model, train_iter, valid_iter, loss, 1000, model.parameters(), optimizer)
```

本实验中 LSTM 神经网络的输入维度为 14，输出维度为 1，隐藏层数量设置为 2，隐藏层神经元个数设置为 128，进行 1000 轮训练，使用 MSE 误差作为损失函数，并且使用 Adam 优化器进行优化。

在每轮训练中，梯度下降对神经网络模型参数进行优化，并将最终训练结果的模型结构参数保存在文件中。

（五）模型预测

加载上述训练过程中训练出的神经网络模型，对输入数据进行预测：

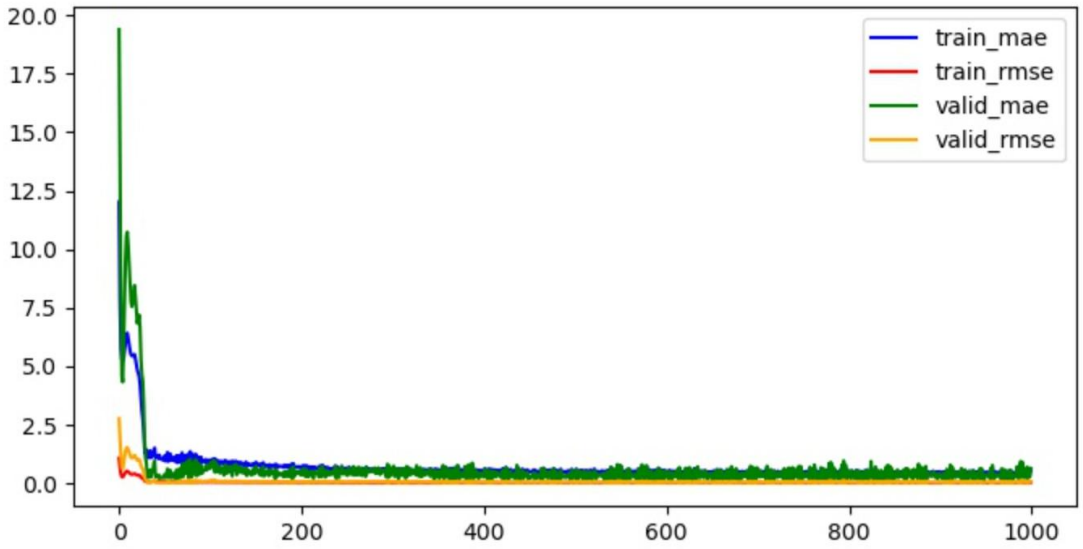
```
def predict(test_x):
    # test 的数目
    n_test = test_x.shape[0]
    test_y = None

    test_y = model(test_x)
    test_y = test_y.detach().cpu().numpy()

    assert (type(test_y) == np.ndarray)
    assert (test_y.shape == (n_test, 1))
    return test_y
```

五、实验数据记录和处理

训练过程中，训练集和校验集平均绝对误差和均方根误差的变化情况如图所示：

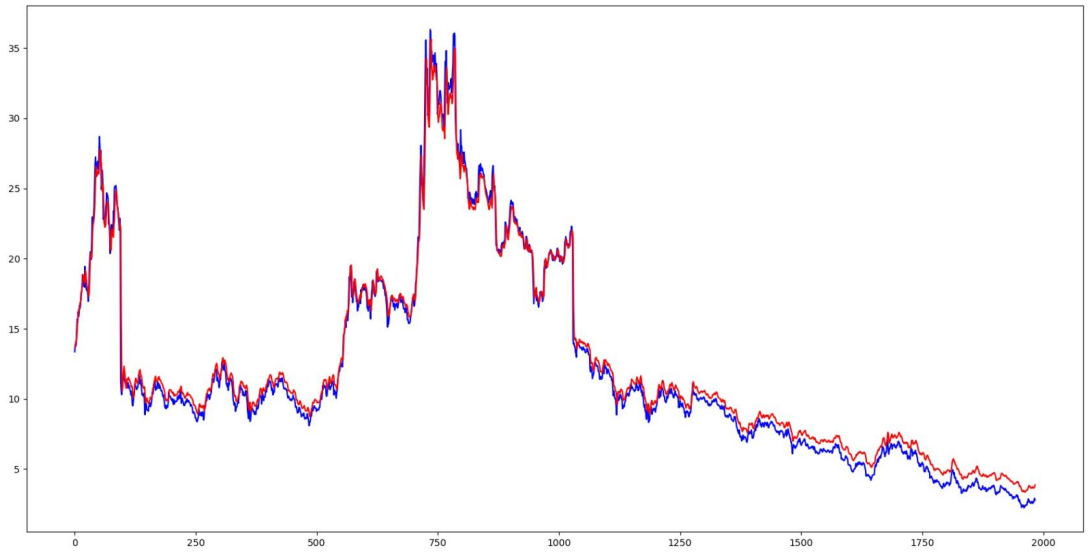


可见随着训练次数增加，训练集和校验集的误差均减小，且区域稳定。

最终将训练得到的模型在测试集上面进行测试，得到平均绝对误差为 0.865 和均方根误差为 0.216。

六、实验结果与分析（必填）

实验训练所得模型在原训练集上预测效果如下，可见拟合效果较佳。



七、讨论、心得

通过本次实验，我对机器学习训练深度模型的基本步骤有了更深的理解，同时掌握了使用 pytorch 框架进行深度学习框架搭建的流程和原理。

同时在本次股价预测实验中，我对线性回归模型和 LSTM 神经网络的原理和结构有了更进一步的了解，也积累了在深度学习代码编写中调整参数的相关经验。