

《Charting by Machines》实验报告

王云志 PB21020537

中国科学技术大学，物理学院，物理系
《量化金融工程：从理论到实践》实验报告

Journal of Financial Economics 153 (2024) 103791



Contents lists available at ScienceDirect

Journal of Financial Economics

journal homepage: www.elsevier.com/locate/jfec



Charting by machines

Scott Murray*, Yusen Xia, Houping Xiao

Georgia State University, Robinson College of Business, 35 Broad Street, Atlanta, GA, 30303, United States



原文摘要（翻译）：

我们通过使用机器学习来预测股票历史表现的回报，从而检验有效市场假说。这些预测强烈预测了未来股票回报的横截面。预测能力在大多数子时期都保持不变，并且在最大的500支股票中很强。预测函数具有重要的非线性和交互作用，随着时间的推移非常稳定，并捕获与动量、反转和现有技术信号不同的影响。这些发现质疑了有效市场假说，并表明技术分析和图表是有道理的。我们还证明，在优化中表现良好的机器学习模型在样本外继续表现良好。

重点：

- 本文质疑“有效市场假说”（弱式），可以根据历史预测未来股票收益；
- 本文的模型效果优于传统，能捕捉到复杂非线性影响完成稳定预测，且泛化能力很好。

实验验证：

本实验报告采用文中的核心方法——横截面收益来选股。数据集为S&P 500在2013年2月到2018年2月的股价数据。经测试，即使数据集较小，依然可以按照模型的选股推荐获得超额收益。

数据集预处理方法：

本实验使用的数据集来自：[S&P 500 stock data](#)；数据集中有五年内的S&P 500日频股价数据。由于文中是月度的低频数据，本实验将每个月的第一天（数据有残缺，取有数据的第一天）作为月度数据。

每一天的特征在原始数据集中有五个：开盘、最低、最高、收盘、交易量。由于交易量和其他几个特征难以统一，本实验直接舍去。

预处理阶段，对每支股票的月度数据使用长度为13的窗口扫描，前面12个月的数据作为样本数据部分，然后计算第12个月收盘价买入、第13个月开盘价卖出的收益率。

这个收益率并不是标签,因为按照论文中的方法,需要根据收益率计算横截面收益排位,所以每一个样本计算完收益率后,先存入该日期的收益率表,收益率全部计算完成后,根据每天的收益率表排序生成样本的标签。

在预处理阶段根据日期标记该样本是训练集还是测试集,第13个月晚于2017-07-01的记为测试集,前面的记为训练集。

代码:

```
# 遍历 CSV 文件
for filename in os.listdir(data_dir):
    if not filename.endswith('.csv'):
        continue
    print(f"Processing {filename}...")
    filepath = os.path.join(data_dir, filename)
    df = pd.read_csv(filepath, parse_dates=["date"])
    df = df.sort_values(by="date").reset_index(drop=True)

    if len(df) < 13:
        continue
    # 提取基础特征
    features = df[["open", "high", "low", "close"]].values
    dates = df["date"].values

    N = len(df)
    print(f"Number of data points in {filename}: {N}")
    for i in range(N - 12): # 保证有12个月特征 + 1个月标签
        idx = num_data
        num_data += 1

        # 日期判断: 用于训练/测试划分
        predict_date = dates[i + 12] # 第13个月的日期
        print(f"Processing sample {idx} for date {predict_date}...")
        data_type[idx] = 1 if predict_date >= np.datetime64("2017-07-01") else 0
        print(f"Data type for sample {idx}: {'Test' if data_type[idx] == 1 else 'Train'}")
        # 特征数据存入 data: 12x5 矩阵
        data[idx] = features[i : i + 12] # shape (12, 5)
        data_date[idx] = predict_date
        # 标签收益率 r (第13月开盘 - 第12月收盘) / 第12月收盘
        close_prev = features[i + 11][3] # 第12月收盘价
        open_next = features[i + 12][0] # 第13月开盘价
        r = (open_next - close_prev) / close_prev
        returns[idx] = r

    windows_by_date[predict_date].append((r, idx))
```

```

print("☑ 数据处理完成，开始生成样本...")

# 横截面排序并分配标签
for date, lst in windows_by_date.items():
    print(f"Processing date {date} with {len(lst)} samples...")
    lst_sorted = sorted(lst, key=lambda x: x[0], reverse=True)
    for rank, (_, idx) in enumerate(lst_sorted):
        labels[idx] = rank/len(lst_sorted) # 标签为排序后的百分比排名

```

模型介绍：

本实验实验 LSTM 模型和 MSE 损失函数。样本的 shape 为 (batch_size=64, 12, 4)，首先输入 LSTM 提取特征，经实验测试，LSTM 的层数为 3 是最合适的。LSTM 提取特征后，进入全连接层，使用 ReLu 激活函数，先升维再降维，最后输出预测值。

为了使特征良好的被利用，这里采用层归一化方法（LayerNorm）对特征做归一化，这可以减少不同样本之间的差异，使训练更加稳定；还能减少内部协变量偏移加速收敛和增强泛化能力。此处的模型初始化使用 Kaiming 初始化，对权重做了一定的调整，这可以使模型避免梯度爆炸或梯度消失，并更好地适配 ReLu 激活函数。

代码：

```

class LSTMModel(nn.Module):
    def __init__(self, input_dim=4, hidden_dim=128, num_layers=3, dropout=0.3):
        super(LSTMModel, self).__init__()
        self.input_norm = nn.LayerNorm(input_dim)
        self.lstm = nn.LSTM(
            input_size=input_dim,
            hidden_size=hidden_dim,
            num_layers=num_layers,
            batch_first=True
        )
        self.regressor = nn.Sequential(
            nn.ReLU(),
            nn.Linear(hidden_dim, 256),
            nn.ReLU(),
            nn.Linear(256, 64),
            nn.ReLU(),
            nn.Linear(64, 32),
            nn.ReLU(),
            nn.Linear(32, 1) # 输出一个值
        )
        self._init_weights()

```

```

def forward(self, x):
    x = self.input_norm(x)
    out, _ = self.lstm(x) # shape: (batch, seq, hidden_dim)
    out = out[:, -1, :] # 取最后一个时间步的输出
    out = self.regressor(out)
    return out

def _init_weights(self):
    for m in self.modules():
        if isinstance(m, nn.Linear):
            nn.init.kaiming_uniform_(m.weight, nonlinearity='relu')
            if m.bias is not None:
                nn.init.constant_(m.bias, 0)

model = LSTMModel().to(device)
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=0.002)

```

训练设置：

本实验执行 40 轮次训练，学习率每隔 5 轮缩小至 0.4 倍。

代码：

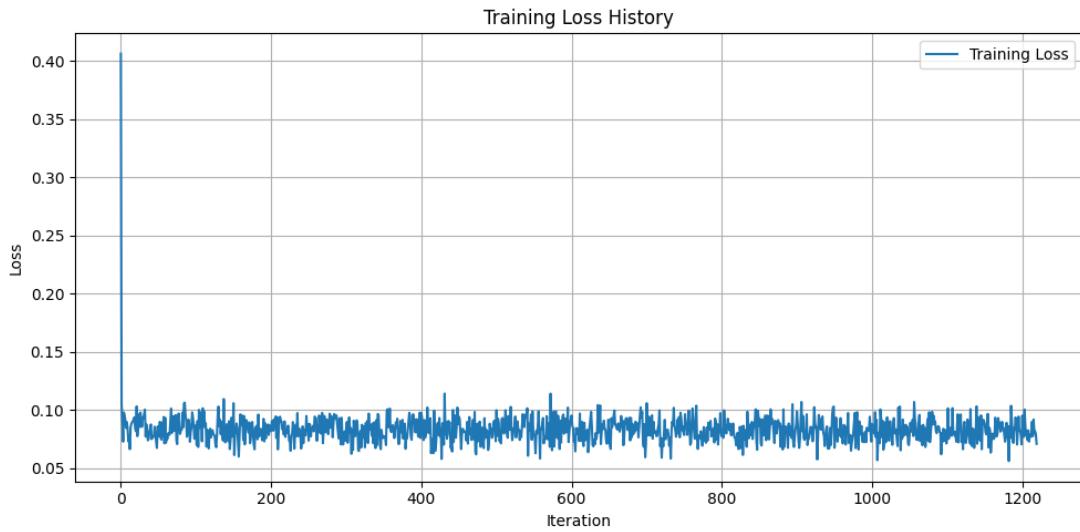
```

scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=5, gamma=0.4)
for epoch in range(40):
    model.train()
    total_loss = 0
    for X_batch, y_batch in train_loader:
        X_batch = X_batch.to(device)
        y_batch = y_batch.to(device)

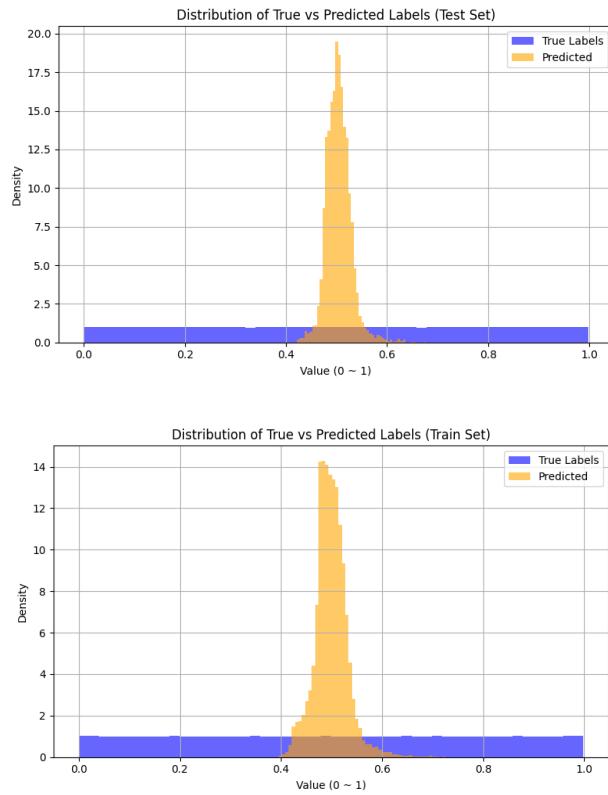
        optimizer.zero_grad()
        preds = model(X_batch)
        loss = criterion(preds, y_batch)
        loss_history.append(loss.item())
        loss.backward()
        optimizer.step()
        total_loss += loss.item()
    print(f"Epoch {epoch+1}, Loss: {total_loss / len(train_loader):.4f}")
    scheduler.step()

```

损失曲线如下：



可以看到，模型在前几个轮次的训练中快速收敛，但是在后面的轮次难以进一步降低损失。通过观察预测值和真实值的分布函数，发现模型难以良好地预测，选择了较为稳妥的均值附近的预测方法。



模型陷入这种预测方法说明特征难以学习到，这或许和我们的标签设置有关：相似的股价走势，即使未来收益率一致，其横截面收益也未必相同，这导致预测效果较差。

不过模型可能依然学习到了一定的选股方法，对同一时期的股票输入模型输入预测值，尽管预测值分布不满足 $[0, 1]$ 的均匀分布，但仍然可以按照这个预测值做排序指导选股。

测试时期实盘收益：

在测试阶段（第 13 个月晚于 2017-07-01）的几个月的数据做测试，每个时期选择最优的 1 支或 3 支股票在第 12 个月收盘时买入，第 13 个月开盘时卖出，计算其平均收益率。

在测试阶段，一月期美国债券无风险收益率如下，全部小于 1.3%。



经计算，按照模型推荐选股得到的月度收益率显著大于同时期的一月期债券无风险收益率。比如如下两组模型的预测结果，仅买卖预测最佳的一个股票的收益率分别为 6% 和 4% 左右，买卖预测最佳的 3 支股票的收益率为 3% 左右。

```
PS C:\Users\ustc> & D:/miniconda3/python.exe d:/AI4Charting/大作业/testofre.py
Model loaded successfully: D:/AI4Charting/大作业/models/lstm_model_without_volume_2139.pth
Date: 2017-07-03T00:00:00.000000000, Selected 1 avg return: 0.0700
Date: 2017-08-01T00:00:00.000000000, Selected 1 avg return: 0.2255
Date: 2017-09-01T00:00:00.000000000, Selected 1 avg return: 0.0009
Date: 2017-10-02T00:00:00.000000000, Selected 1 avg return: 0.0969
Date: 2017-11-01T00:00:00.000000000, Selected 1 avg return: -0.0415
Date: 2017-12-01T00:00:00.000000000, Selected 1 avg return: 0.0835
Date: 2018-01-02T00:00:00.000000000, Selected 1 avg return: 0.0255
Date: 2018-02-01T00:00:00.000000000, Selected 1 avg return: 0.0386

Overall average return of top-1 selections per date: 0.0624

Date: 2017-07-03T00:00:00.000000000, Selected 3 avg return: 0.0221
Date: 2017-08-01T00:00:00.000000000, Selected 3 avg return: 0.0394
Date: 2017-09-01T00:00:00.000000000, Selected 3 avg return: 0.0045
Date: 2017-10-02T00:00:00.000000000, Selected 3 avg return: 0.0851
Date: 2017-11-01T00:00:00.000000000, Selected 3 avg return: 0.0304
Date: 2017-12-01T00:00:00.000000000, Selected 3 avg return: 0.0469
Date: 2018-01-02T00:00:00.000000000, Selected 3 avg return: 0.0253
Date: 2018-02-01T00:00:00.000000000, Selected 3 avg return: 0.0184

Overall average return of top-3 selections per date: 0.0340

PS C:\Users\ustc> & D:/miniconda3/python.exe d:/AI4Charting/大作业/testofre.py
Model loaded successfully: D:/AI4Charting/大作业/models/lstm_model_without_volume_2200.pth
Date: 2017-07-03T00:00:00.000000000, Selected 1 avg return: 0.1220
Date: 2017-08-01T00:00:00.000000000, Selected 1 avg return: -0.0237
Date: 2017-09-01T00:00:00.000000000, Selected 1 avg return: 0.0009
Date: 2017-10-02T00:00:00.000000000, Selected 1 avg return: 0.0743
Date: 2017-11-01T00:00:00.000000000, Selected 1 avg return: 0.0671
Date: 2017-12-01T00:00:00.000000000, Selected 1 avg return: 0.0523
Date: 2018-01-02T00:00:00.000000000, Selected 1 avg return: 0.0255
Date: 2018-02-01T00:00:00.000000000, Selected 1 avg return: 0.0158

Overall average return of top-1 selections per date: 0.0418

Date: 2017-07-03T00:00:00.000000000, Selected 3 avg return: 0.0546
Date: 2017-08-01T00:00:00.000000000, Selected 3 avg return: 0.0692
Date: 2017-09-01T00:00:00.000000000, Selected 3 avg return: 0.0135
Date: 2017-10-02T00:00:00.000000000, Selected 3 avg return: 0.0523
Date: 2017-11-01T00:00:00.000000000, Selected 3 avg return: 0.0454
Date: 2017-12-01T00:00:00.000000000, Selected 3 avg return: 0.0112
Date: 2018-01-02T00:00:00.000000000, Selected 3 avg return: -0.0009
Date: 2018-02-01T00:00:00.000000000, Selected 3 avg return: 0.0032

Overall average return of top-3 selections per date: 0.0311
```

数据提供：

本实验所有的数据和代码可以在作者的仓库中查看：[phywyz-ustc/QuantCourseReport](https://github.com/phywyz-ustc/QuantCourseReport)