# 基于 iTransformer 的高频股价的趋势预测模型

黄春阳、夏泽洲

2024 年 6 月 14 日

**摘要**

近年来，人工智能技术已经被广泛应用于量化交易领域。金融市场的特点是噪声严重，难以从历史数据中得到未来的市场走势。为解决此问题，本文将价格走势预测问题建模为时序序列分类问题，利用 iTransformer[1] 模型，并使用 tick 级快照数据来进行训练和验证，模型表现优异。

## 1 问题定义

基于过去 100 个 ticks 的快照数据, 预测一定时间后股票的走势. 这个问题等效于一个三分类问题.

## 2 建模思路

目标问题是以分类问题为形式的预测问题，同时由于小数据集，模型容量不易过大，避免过拟合。由于这两个原因，本文选择将 iTransformer 应用至此问题上。iTransformer [1] 在预测任务中是最先进的模型。Transformer[2] 在计算机视觉、自然语言处理等领域具有广泛应用，而在多变量时序预测任务中通常不如线性模型。iTransformer 解释了这一问题的原因，如图1所示，传统的 Transformer 将一个时间戳下的多个变量映射成一个词向量，由于这些变量具有的物理含义完全不同，抹去了多变量之间的相关性。iTransformer 改变了注意力机制的维度，将每个变量在时间维度上映射成一个词向量，获取不同变量之间的注意力，可以大大提高多变量时序预测的能力。

模型的整体架构如图2所示，伪代码如图所示，与 Transformer 的 encoder 结构相同，仅在注意力方向进行修改。

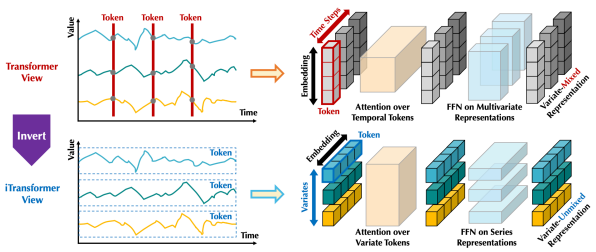



图 1: Transformer 和 iTransformer 的对比图

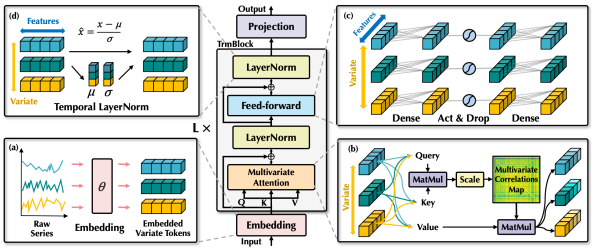



图 2: Overall structure of iTransformer

**Algorithm 1** iTransformer - Overall Architecture.

**Require:** Input lookback time series $\mathbf{X} \in \mathbb{R}^{T \times N}$; input Length $T$; predicted length $S$; variates number $N$; token dimension $D$; iTransformer block number $L$.

$\mathbf{X} = \mathbf{X}.\texttt{transpose}$ ▷ $\mathbf{X} \in \mathbb{R}^{N \times T}$

▷ Multi-layer Perceptron works on the last dimension to embed series into variate tokens.

$\mathbf{H}^0 = \texttt{MLP}(\mathbf{X})$ ▷ $\mathbf{H}^0 \in \mathbb{R}^{N \times D}$

**for** $l$ **in** $\{1, \cdots, L\}$**:** ▷ Run through iTransformer blocks.

  ▷ Self-attention layer is applied on variate tokens.

  $\mathbf{H}^{l-1} = \texttt{LayerNorm}(\mathbf{H}^{l-1} + \texttt{Self-Attn}(\mathbf{H}^{l-1}))$ ▷ $\mathbf{H}^{l-1} \in \mathbb{R}^{N \times D}$

  ▷ Feed-forward network is utilized for series representations, broadcasting to each token.

  $\mathbf{H}^{l} = \texttt{LayerNorm}(\mathbf{H}^{l-1} + \texttt{Feed-Forward}(\mathbf{H}^{l-1}))$ ▷ $\mathbf{H}^{l} \in \mathbb{R}^{N \times D}$

  ▷ LayerNorm is adopted on series representations to reduce variates discrepancies.

**End for**

$\hat{\mathbf{Y}} = \texttt{MLP}(\mathbf{H}^{L})$ ▷ Project tokens back to predicted series, $\hat{\mathbf{Y}} \in \mathbb{R}^{N \times S}$

$\hat{\mathbf{Y}} = \hat{\mathbf{Y}}.\texttt{transpose}$ ▷ $\hat{\mathbf{Y}} \in \mathbb{R}^{S \times N}$

**Return** $\hat{\mathbf{Y}}$ ▷ Return the prediction result $\hat{\mathbf{Y}}$

图 3: The pseudocode of overall structure

# 3 数据预处理

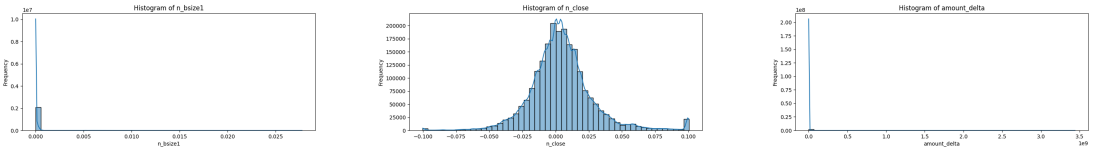
我们首先对数据进行预处理，包括检查数据完整性、观察数据分布并进行必要的特征工程。

## 3.1 数据完整性检查

我们检查了数据集中是否存在缺失值或异常值，并确保所有日期和股票的数据文件都存在。

## 3.2 数据分布观察

我们观察了与价格相关的特征，例如收盘价、中间价和买卖五档价格，并分析了它们在训练集中的分布情况，如图4。

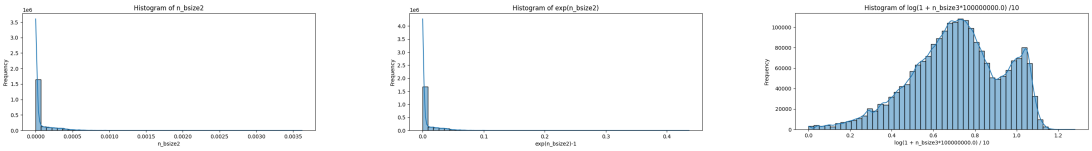



买卖量的数值分布　　价格的数值分布　　delta_amount 的数值分布

图 4: 三种特征的数值分布

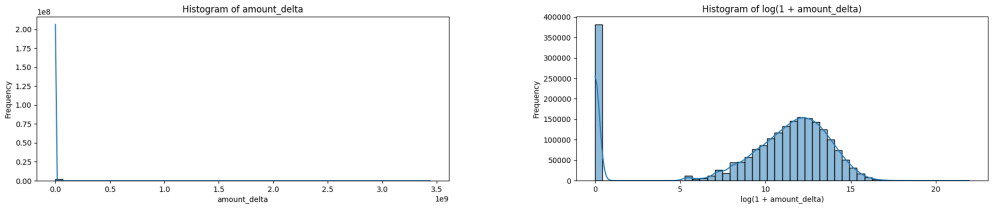
## 3.3 特征工程

我们对部分特征进行了变换，例如对成交金额进行对数变换，以改善其分布特性。




买卖量的数值分布　　exp 处理的买卖量的数值分布　　log 处理的买卖量的数值分布

图 5: 买卖量及其变换后的分布




delta_amount 的数值分布　　log 处理的 delta_amount 的数值分布

图 6: delta_amount 及其变换后的分布

# 4 模型训练

，并使用 Adam 优化器进行训练。我们还检查了模型的梯度性质，确保其收敛。

## 4.1 训练过程

我们将数据集按照时间维度划分为训练集、验证集和测试集，训练过程中通过详尽的日志记录来监控模型在各个训练周期的表现：

- 模型使用配置中指定的学习率，可以选择 Adam 或 SGD 优化器进行训练。
- 记录每个周期的训练损失、验证损失和测试损失，以监控过拟合和欠拟合情况。
- 在验证和测试阶段，记录性能指标如 F0.5 分数和准确率。
- 实现早停机制，如果验证性能没有改善，则停止训练，以防过拟合。

## 4.2 评估指标

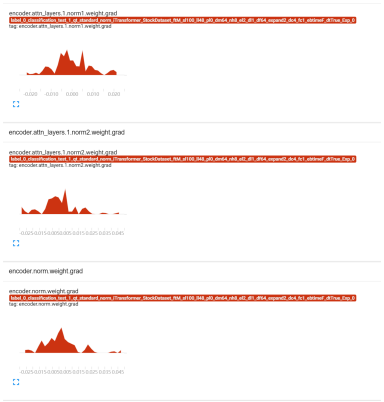
模型的表现通过多个指标进行评估：

- **准确率：**模型预测标签的整体正确性。
- **F0.5 score：**一个平衡精确度和召回率的指标，偏重于精确度。
- **confusion matrix：**提供分类问题上预测结果的概览。
- **classfication report：**包括每个类别的精确度、召回率和 F1 分数。

## 4.3 梯度监测

为了确保梯度的稳定的传播，我们使用 tensorboard 对训练过程的梯度进行统计。图 X 显示，模型的梯度传播稳定。

Encoder 的梯度分布



Layernormalization 的梯度分布

Decoder 的梯度分布

图 7: 买卖量及其变换后的分布

# 5 实验结果

我们对比了 iTransformer 模型与其他几种时序序列预测模型，例如 Transformer[2]、TimesNet[3] 和 Deeplob[4]，并分析了不同超参数对模型性能的影响。

## 5.1 实验探究

为了优化我们的模型, 我们做了许多的探究性实验. 并用本地测试来考察模型的性能. 测试时使用训练未用到的 70-78 日的全部十只股票的数据, 并选取不同预测窗口的最佳结果, 作为模型的表现.

### 5.1.1 预处理方法的影响

如图4 所示，在原始数据中，部分特征的数量级过大或过小，不利于神经网络的梯度传播。为此我们使用一些单调增函数对这些特征进行预处理，使其既保持数值的相对大小，又使其数量级接近 1。

我们控制模型的其他参数不变, 测试不同预处理方法的影响, 结果如表1所示.

**exp_log 的处理方法：**所有的买卖量 (*size)：$\exp(100 \times x) - 1$；成交量 ('amount_delta'): $\log(1 + x)$; 报价不做处理。

```
df[size_columns] = np.exp(100*df[size_columns]) - 1
 df['amount_delta'] = np.log1p(df['amount_delta'])
```

**log_log 的处理方法：**所有的买卖量 (*size)：$\log(1 + 10^8 \times x)$；成交量 ('amount_delta'): $\log(1 + x)$; 报价不做处理。

```
df[size_columns] = np.log1p(1e8 * df[size_columns])
 df['amount_delta'] = np.log1p(df['amount_delta'])
```

**times_log 的处理方法：**所有的买卖量 (*size)：$100 \times x$ ; 成交量 ('amount_delta'): $\log(1+x)$; 报价不做处理。

```
df[size_columns] = df[size_columns] * 100
df['amount_delta'] = np.log1p(df['amount_delta'])
```

| 预处理方法 | F0.5 | pnl |
|---|---|---|
| exp_log | 0.3892 | 52.43 |
| log_log | 0.3465 | 30.50 |
| times_log | 0.3772 | 41.77 |

表 1: 预处理方法

实验结果表明, 对买卖量数据的预处理方法中, exp 表现最好, times 次之, log1p 最差. 我们猜测这可能是因为 exp 函数为凸函数, times 为线性函数, log1p 为凹函数. 函数的凹凸属性影响了模型的预测能力. 一个可能的解释如下, exp 函数对较大的数值更敏感, 而较大的买卖报单对价格的影响更敏感.

### 5.1.2 模型大小的影响

我们测试了不同层数和模型大小的超参数组合，其中测试了两种不同的数据预处理方法.dm 表示模型的维度, el 表示注意力层的层数. 实验表明, 对于 exp log 的数据预处理方法, dm 和 el 增大时, 模型的表现更好. 对于 log log 的数据预处理方法, 模型的 dm 或 el 增大时会使得模型编号, 但是 dm 和 el 同时增大时导致结果变差, 这可能是因为发生了过拟合的现象.

| 预处理方法 exp_log | | | 预处理方法 log_log | | |
|---|---|---|---|---|---|
| 模型大小 | F0.5 | pnl | 模型大小 | F0.5 | pnl |
| dm=32,el=1 | nan | nan | dm=32,el=1 | 0.3342 | 19.00 |
| dm=32,el=2 | 0.3708 | 44.03 | dm=32,el=2 | 0.3774 | 46.22 |
| dm=64,el=1 | 0.3842 | 48.99 | dm=64,el=1 | 0.3804 | 44.47 |
| dm=64,el=2 | 0.3892 | 52.43 | dm=64,el=2 | 0.3465 | 30.50 |

表 2: 模型大小

### 5.1.3 优化器的影响

选用 adam 和 sgd 优化器, 在其他条件不变的情况下测试, 结果如下所示, adam 的结果更好.

| 优化器 | F0.5 | pnl |
|---|---|---|
| adam | 0.3465 | 30.50 |
| sgd | 0.3326 | 21.84 |

表 3: 优化器

#### 5.1.4 滑动平均的影响

我们尝试对数据进行滑动平均, 探究滑动窗口大小对模型预测能力的影响. 特别的, 滑动窗口为 1 时, 等效于没有进行滑动平均. 从表中可以看出, 随着滑动平均的增加, 模型的预测能力逐渐下降. 我们认为, 这是因为 itransformer 的信息利用能力较强, 所以当模型输入的信息减少时, 预测能力就会下降.

| 滑动窗口大小 | 1 | 2 | 5 | 10 | 15 | 20 |
|---|---|---|---|---|---|---|
| F0.5 | 0.3465 | 0.3459 | 0.3326 | 0.3279 | 0.3267 | 0.3243 |
| pnl | 30.50 | 29.58 | 24.53 | 21.56 | 21.90 | 20.63 |

表 4: 滑动平均

#### 5.1.5 噪声的影响

我们尝试对数据增加一些噪声, 探究噪声强度对模型预测能力的影响. 考虑到不同特征的物理含义不同，数量级存在较大差异，我们使用比例添加噪声，代码如下。

```
noise = np.random.normal(0, self.noise_amplitude, x.shape)
x = x * (1 + noise)
```

从表中可以看出, 随着噪声强度的增加, 模型的预测能力逐渐下降. 我们认为, 这是因为我们的模型没有过拟合, 因此噪声的增加使得模型获取的信噪比更低, 所以预测能力下降.

| 噪声幅度 | 0 | 0.05 | 0.1 | 0.2 | 0.3 | 0.4 |
|---|---|---|---|---|---|---|
| F0.5 | 0.3465 | 0.3401 | 0.3297 | 0.3167 | 0.3292 | 0.3383 |
| pnl | 30.50 | 28.05 | 24.37 | 21.34 | 19.91 | 18.37 |

表 5: 噪声强度

#### 5.1.6 训练 label 的影响

我们探究了不同训练指标对模型不同长度的序列预测能力的影响, 结果如下所示. 其中, label10,20,40,60 的训练参数相同, 测试参数也想通. label5 的训练参数与之前一样, 但少了一列 time 的数据, 测试数据的数量也比之前更多一点. 参数不完全一样的主要原因是时间有限.

从表中可以看出, 无论是用窗口为 10,20,40 还是 60 的指标来进行训练, 模型均对 60 steps 后有最强的预测能力. 这可能说明, 我们的模型擅长把握长距离的信息.

但是使用窗口为 5 的指标进行训练时, 模型在窗口为 10 的时候 F0.5 表现最佳, 窗口为 20 的时候 pnl 表现最佳, 说明此时模型对短距离的信息把握能力较强.

二者之间的差别可能是不同训练参数的区别, 也可能是引入了新的自变量'time', 或者测试量不同导致的, 因此日后还需要更详细的研究和分析.

| F0.5 | predict | | | | |
|---|---|---|---|---|---|
| train | 5 | 10 | 20 | 40 | 60 |
| 10 | 0.3083 | 0.3921 | 0.3203 | 0.3660 | 0.3787 |
| 20 | 0.2570 | 0.3492 | 0.2907 | 0.3461 | 0.3609 |
| 40 | 0.2258 | 0.3044 | 0.2679 | 0.3473 | 0.3686 |
| 60 | 0.1996 | 0.2736 | 0.2393 | 0.3275 | 0.3618 |

表 6: 训练指标和预测能力对照表 (F0.5)

| pnl | predict | | | | |
|---|---|---|---|---|---|
| train | 5 | 10 | 20 | 40 | 60 |
| 10 | 4.21 | 6.28 | 8.28 | 8.83 | 7.84 |
| 20 | 3.44 | 5.50 | 7.65 | 9.41 | 8.66 |
| 40 | 1.92 | 2.97 | 5.09 | 7.91 | 7.82 |
| 60 | 1.35 | 2.26 | 3.97 | 6.54 | 7.41 |

表 7: 训练指标和预测能力对照表 (pnl)

| | | predict | | | | |
|---|---|---|---|---|---|---|
| train | type | 5 | 10 | 20 | 40 | 60 |
| 5 | F0.5 | 0.3291 | 0.3892 | 0.3079 | 0.3395 | 0.3505 |
| 5 | pnl | 36.44 | 48.26 | 52.43 | 43.41 | 36.60 |

表 8: 训练指标和预测能力对照表 (label5)

### 5.2 表现

在本地测试中, 我们模型最好的 F0.5 为 0.3892, pnl 为 48.26. 在公榜数据上, 我们的模型也表现更好, 如表 (9) 所示. 其中排名截止到 2024.6.14 12:00, 为截止提交的 12h 前. 排名是指选手的排名.

我们的模型在公榜上的结果比本地测试集的 F0.5 结果更好, 这可能是因为我们的模型泛化能力更强, 没有对数据过拟合.

| 指标 | precision | recall | F0.5 | pnl | pnl_average |
|---|---|---|---|---|---|
| 数值 | 0.4203 | 0.5699 | 0.4265 | 87.49 | 0.0004339 |
| 排名 | 3 | 2 | 1 | 2 | 2 |

表 9: 公榜表现

## 6 结论

本实验表明，iTransformer 模型在金融中间价预测任务上具有良好的性能。未来可以进一步探索其他模型和特征工程方法，以提高预测准确率。

# 参考文献

[1] Yong Liu, Tengge Hu, Haoran Zhang, Haixu Wu, Shiyu Wang, Lintao Ma, and Mingsheng Long. itransformer: Inverted transformers are effective for time series forecasting. *arXiv preprint arXiv:2310.06625*, 2023.

[2] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. Attention is All you Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[3] Haixu Wu, Tengge Hu, Yong Liu, Hang Zhou, Jianmin Wang, and Mingsheng Long. Timesnet: Temporal 2d-variation modeling for general time series analysis. In *International Conference on Learning Representations*, 2023.

[4] Zihao Zhang, Stefan Zohren, and Stephen Roberts. Deeplob: Deep convolutional neural networks for limit order books. *IEEE Transactions on Signal Processing*, 67(11):3001–3012, June 2019.