Structural Machine Learning: Final Project

陳慧如7108053117

Department of Applied Mathematics, National Chung Hsing University

I. 實作PAPER

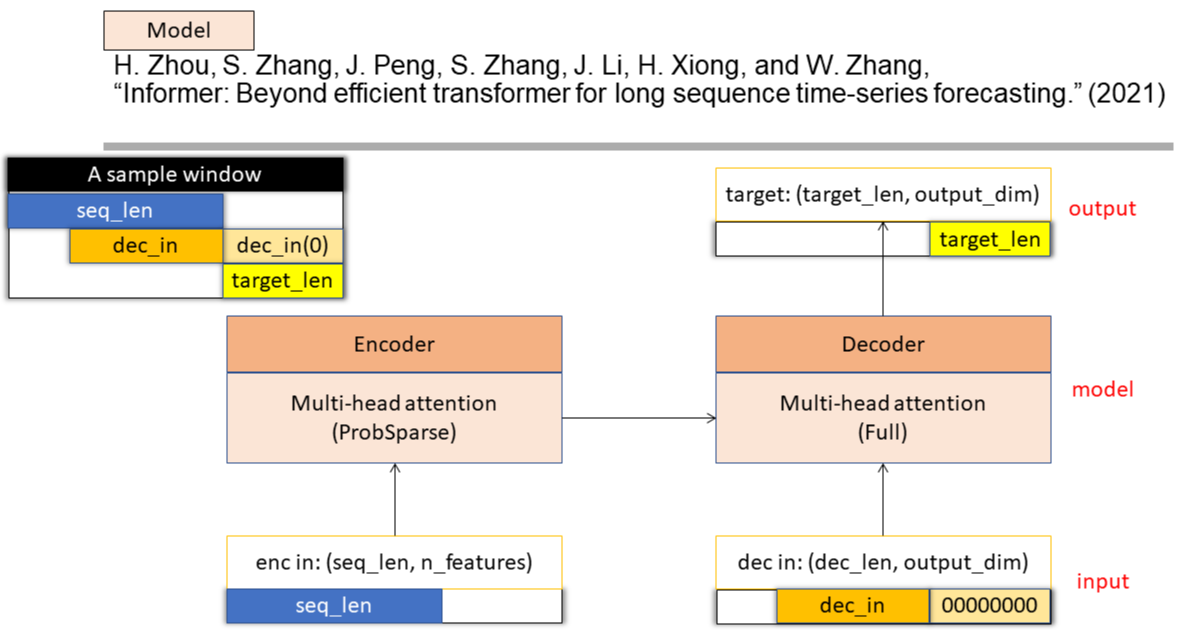

Fig. 1: model

實驗用的是Informer[1]，是Transformer的encoder跟decoder架構，另外把時間資訊用Bert[2] (segment and position embedding) 的方式加上原序列，以此對時間序列資料做處理。

A. *Actor with Multi-head Attention Mechanism*

After using dictionary learning to reconstruct each feature respectively, we concatenate the the atoms in $\mathbf{D}^c$ correspond to the top N_{nz} max value in $\mathbf{A}^c$ into a vetor as the c-th element in the input sequence of the actor with Attention mechanism[3]:

- D_t: Vector size of each element in the input sequence.
- $D_t = N_{wi} \times N_{nz}$.
- The c-th element at time t in the input sequence for the actor: $\mathbf{f^t_c} \in \mathbb{R}^{D_t}$.
- Input sequence of the actor at time t: $\mathbf{F^t} = [\mathbf{f^t_1},\ \mathbf{f^t_2},\ \cdots,\ \mathbf{f^t_{N_f}}] \in \mathbb{R}^{N_f \times D_t}$
- D_e: Embedding size.
- N_h: Number of heads.
- ϕ^Q, ϕ^K and ϕ^V: The same linear transformation with different weights to let the input sequence play the different roles in Attention mechanism by itself.
- ϕ^O: The linear transformation to integrate the information from each head and output the final policy.
- ϕ^M: The actor with Multi-head Attention mechanism.

$$\begin{aligned} \mathbf{Q}_i &= \phi_i^Q(\mathbf{F}) \\ \mathbf{K}_i &= \phi_i^K(\mathbf{F}) \\ \mathbf{V}_i &= \phi_i^V(\mathbf{F}) \\ \textit{where } i &= 1, 2, \ldots, N_h \end{aligned} \tag{1}$$

$$\textit{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) = \textit{Softmax}(\frac{\mathbf{Q}_i \mathbf{K}_i^{\mathbf{T}}}{\sqrt{D_e}})\mathbf{V}_i \tag{2}$$

$$\begin{aligned} \textit{Multi-head Attention} &= \phi^O(\textit{Concat}(\textit{head}_1, \textit{head}_2, \ldots, \textit{head}_h)) \\ \textit{where } \textit{head}_i &= \textit{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i) \end{aligned} \tag{3}$$

B. Encoder

用的是Probsparse的attention，作法是當Q attention完K之後進softmax，取機率最大的數個值再去乘相對應的V。

C. Decoder

用的是原來的attention[3]，取一段時間的label跟要預測的一段時間補零concat在一起當作encoder的input，想讓encoder參考前面label的樣子去預測未來。

II. DATA

原paper用的是ETT, ECL以及Weather dataset，實驗改成用yfinance的SP500股價資料當作投資標的。

一個標的的股價資料含有Open, High, Low, Close, Adj. close and Volume，因為影響股價的因素不只有標的的歷史資料本身，還有許多社會經濟現象影響投資人的決策進而影響未來的股價波動，實驗簡單取yahoo finance首頁上置頂的指標標的視作state供actor參考做決策。

III. REINFORCEMENT LEARNING

- N_{wo}: Output window size.
- The actor will output 3 probabilities of each action at time t: (1) holding one unit short position, (2) not holding any position and (3) holding one unit long position:
  $\mathbf{o}_t = [prob_t^s,\ prob_t^n,\ prob_t^l]^T$.
  $\mathbf{O}_t = [\mathbf{o}_t,\ \mathbf{o}_{t+1},\ \ldots,\ \mathbf{o}_{t+N_{ow}-1}]^T$.
- The label at time t is the return will earn at time $t+1$ in different states that actor take different actions at time t:
  $\mathbf{l}_t = [-price_{t+1},\ 0,\ price_{t+1}]^T$
  $\mathbf{L}_t = [\mathbf{l}_t,\ \mathbf{l}_{t+1},\ \ldots,\ \mathbf{l}_{t+N_{ow}-1}]^T$.
- N_b: Batch size.
- Output of the actor: $\mathbf{O} = \phi^M(\mathbf{F}) \in \mathbb{R}^{N_b \times N_{wo} \times 3}$
- Label: $\mathbf{L} \in \mathbb{R}^{N_b \times N_{wo} \times 3}$

The object of training is let the actor make the decision that can earn the most return:

$$\underset{\{\phi_i^Q\},\{\phi_i^K\},\{\phi_i^V\},\phi^O}{\arg\min} -\sum \mathbf{O} \odot \mathbf{L} \tag{4}$$

IV. EXPERIMENT

實驗將一個sample window切成如圖1左上角所示，用一個段時間的data預測未來一段時間的data。

金融市場一直在變，太久遠以前的data可能較不fit現在的市場特性，所以實驗參考[4]的方法，將資料切成很多training window，一個window切成pretraining, validation and online training data，一次移動online training data數量的步長，如圖2。

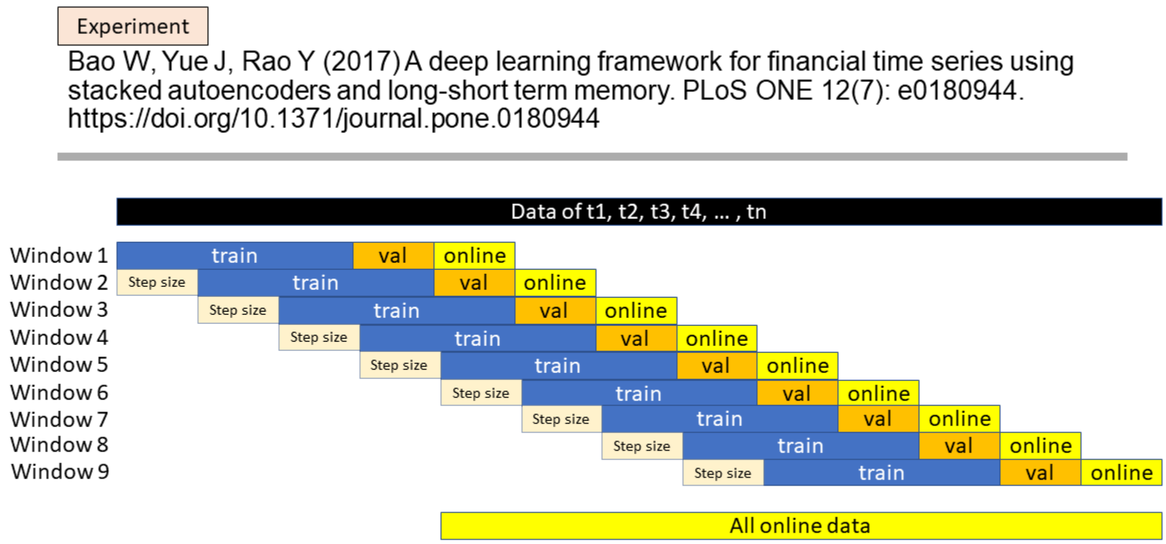

Fig. 2: experience design

設state包含很多個金融標的，用的是1分鐘間隔以及2分鐘間隔的data，每個標的的開盤收盤時間不一樣，實驗將每筆target的encoder input取最後一天當基準，去抓state中其他標的以時間點基準往前取sequence length的data與target合併當作一筆data的input。

總結用以下方式整理data (圖3) ，algorithm1：

V. Result

實驗用SP500當作投資目標，試了區間一分鐘跟兩分鐘的，epochs試了1個，4個還有64個，feature extractor用的是Informer的encoder-decoder架構，online test的累積報酬結果如圖4。每個時刻actor出的動作如圖5，每個時刻t固定持有一單位position，黑色（no position）、紅色（long position）以及綠色（short position）。每個時刻經過attention架構取到的feature經過PCA再用K-means分成3群的結果如圖6。

看結果會發現train 64個epochs的動作幾乎都會出一樣的，取到的feature也發現很大一部份集中在同一群，這部分還沒找到原因，再來會先把其他比較實驗做起來，比較結果看看問題出在哪邊。

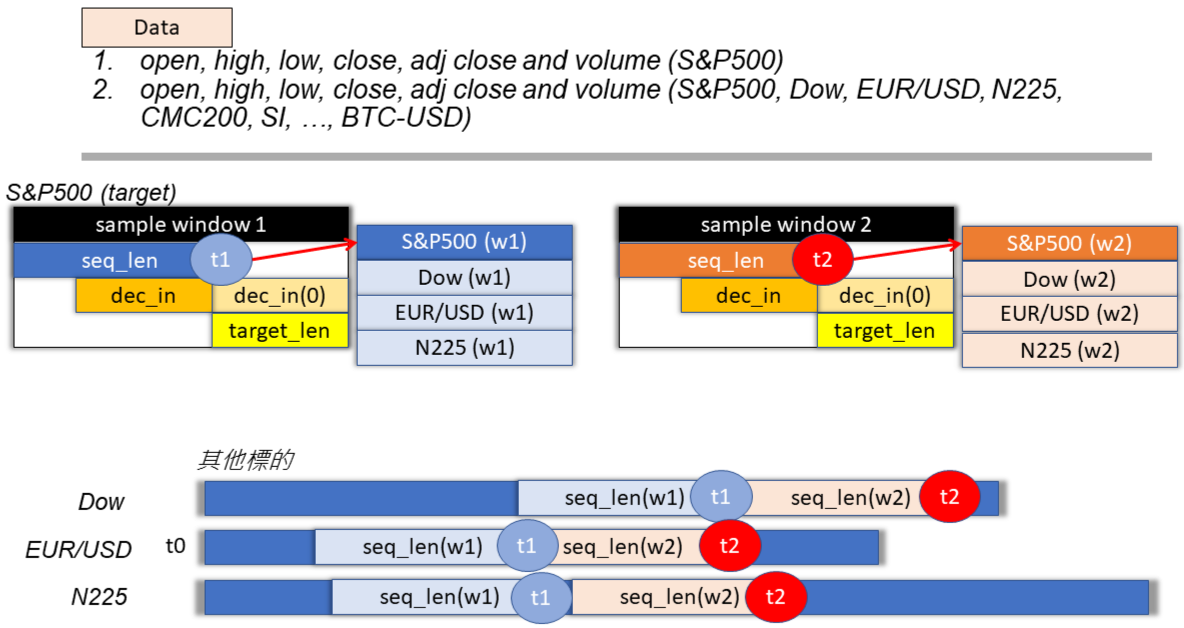

Fig. 3: generate multi-feature data

References

[1] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang, "Informer: Beyond efficient transformer for long sequence time-series forecasting," 2020.

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2017.

[4] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLOS ONE*, vol. 12, no. 7, pp. 1–24, 07 2017. [Online]. Available: https://doi.org/10.1371/journal.pone.0180944

Algorithm 1 generate multi-feature data

```
target symbol = SP500
state symbols = [SP500, DJI, N225, GBP/USD, SI, ··· , BTC/USD]
df = concatenate state symbols by time.
convert target symbol into sample windows.
for window in sample windows do
    for sample in window do
        enc in, dec in, target = sample
        idx = enc in[-1]
        for symbol in state symbols do
            data = df[symbol][:idx].dropna()
            if len(data) < seq len then
                padding zeros in upper side.
            end
        end
        concatenate all the data for each symbol as the encoder input.
    end
end
convert sample windows(contain state symbols) into training windows.
for pretrain, val, online in training windows do
    normalize encoder input of pretrain, val and online data:
    divide data in a training by max(abs(encoder input in all pretrain data)) for each column.
end
```

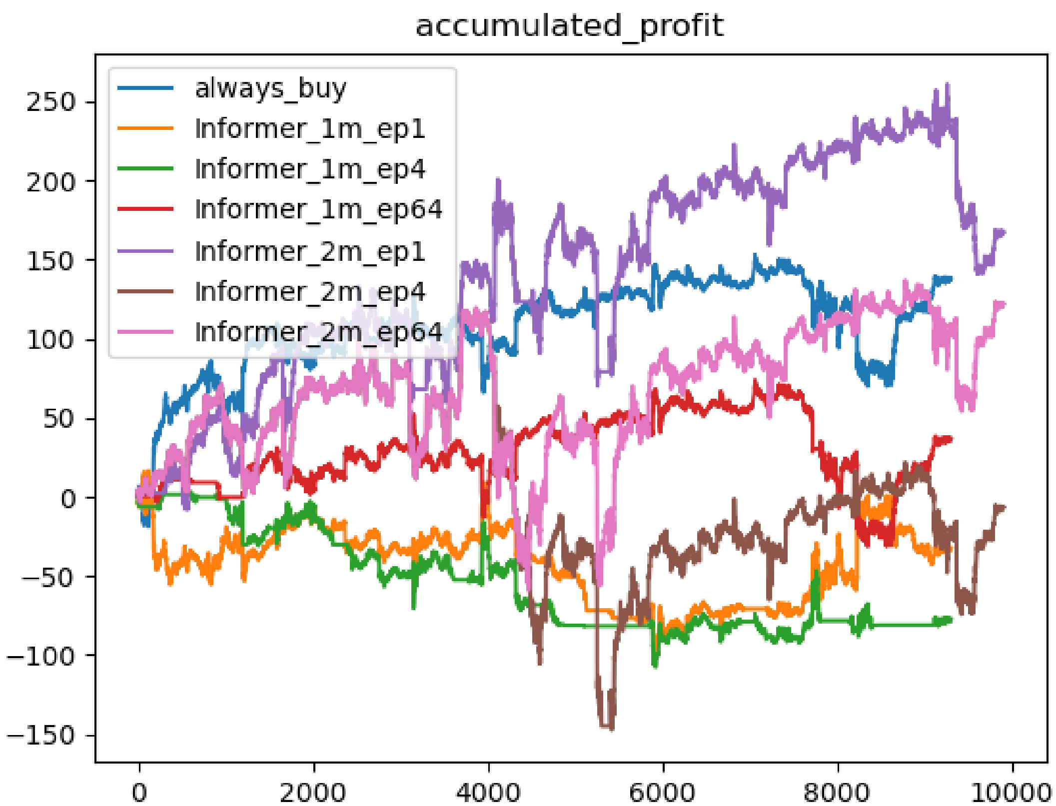

Fig. 4: accumulated profit

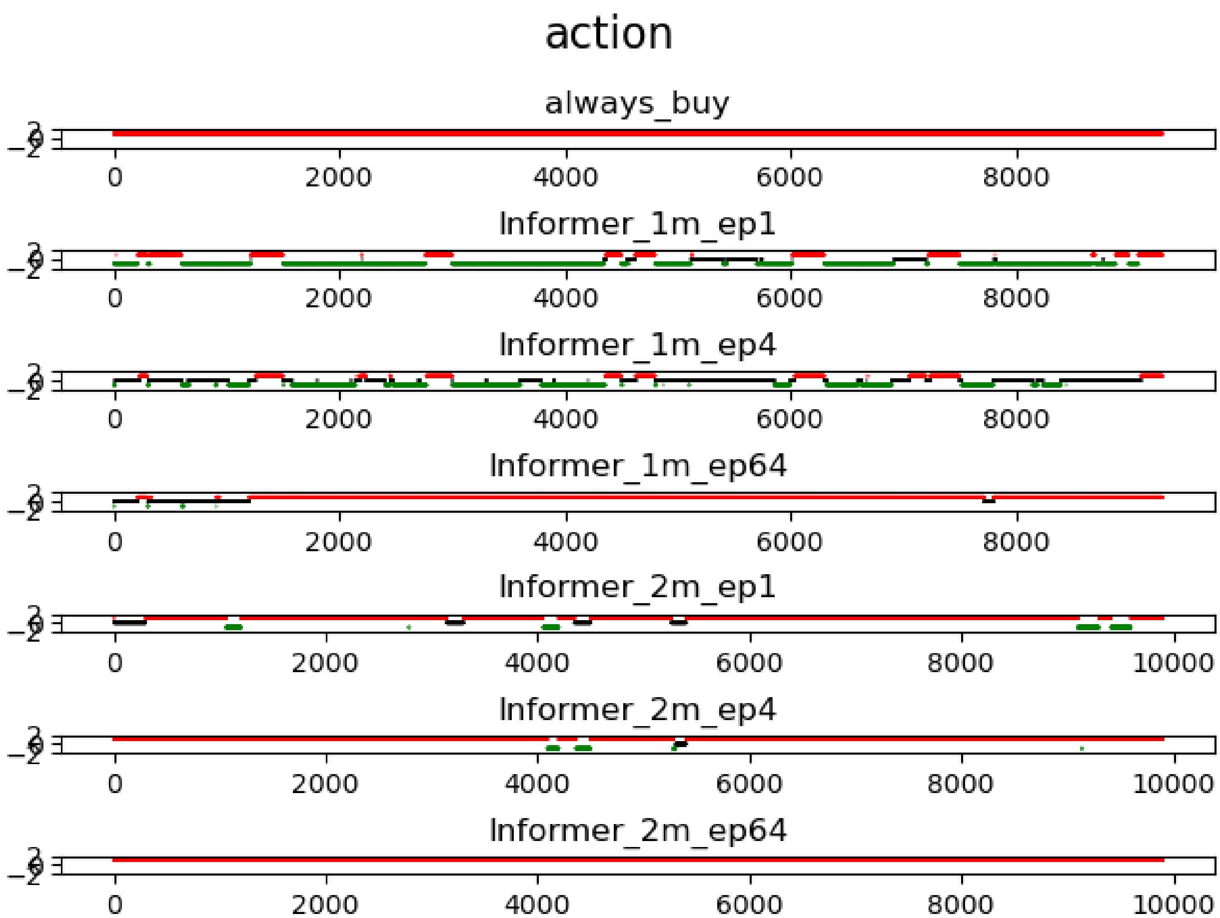
action
always_buy
Informer_1m_ep1
Informer_1m_ep4
Informer_1m_ep64
Informer_2m_ep1
Informer_2m_ep4
Informer_2m_ep64
0
2000
4000
6000
8000
10000

Fig. 5: action

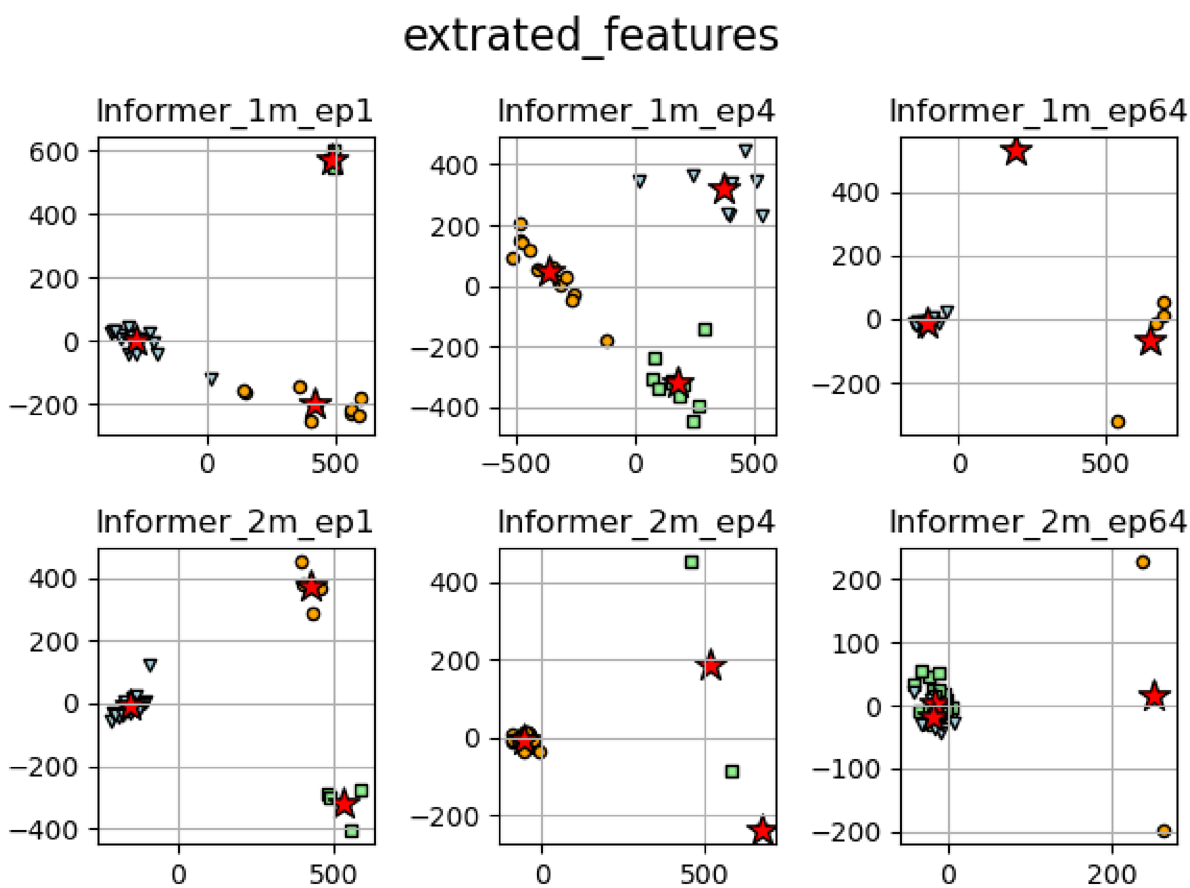

Fig. 6: feature