一、研报解读

- **目的**：基于779个时序因子，采用稀疏自编码器层+GRU层+全连接层输出层的神经网络框架，预测未来N日利率/国债期货的涨跌，作为日频多空决策的择时依据，并在测试集查看模型收益表现。
- **参考文献**：
  - 东方证券——宏观固收量化研究系列之（九）：基于神经网络模型的利率择时-东方证券
  - 东证期货——国债期货量化系列四：基于多种深度学习模型的策略框架探讨
  - Bao W, Yue J, Rao Y. A deep learning framework for financial time series using stacked autoencoders and long-short term memory[J]. PloS one, 2017, 12(7): e0180944.
- 笔记作者：Ting

二、具体模型设定以及部分代码复现

- **预测标的（输出）**：国债期货和利率，具体标的如下：

  - 10 年期国债期货主力合约(T)
  - 5 年期国债期货主力合约(TF)
  - 10Y 国开活跃券利率
  - 5Y 国开活跃券利率

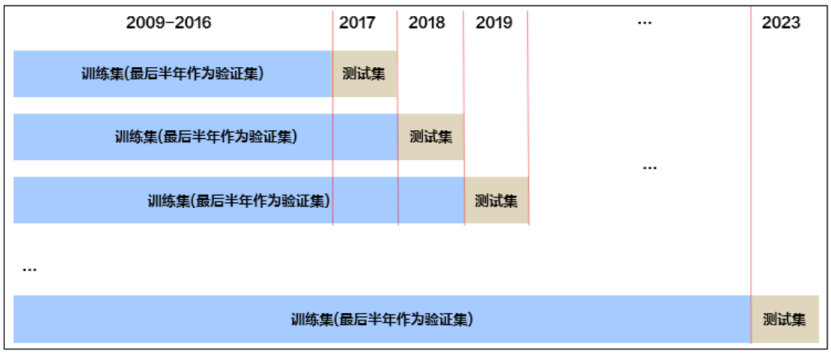
会对测试集中的每日都计算“未来N日收益”，因此是日频预测。

- **输入**：一共779个时序因子，但对于不同预测标的，会使用不同的因子和时序长度：（表格中的标签就是预测标的）

数据集	时间跨度	因子数量	标签
含国债期货	2016.05至今	779	未来N日国债期货涨跌幅
不含国债期货	2007.11至今	471	未来N日国开利率涨跌

以下分析中均以预测利率涨跌为例。

此时输入的特征维度`input_dim=471`，根据以下滚动划分的形式来确定训练集和测试集

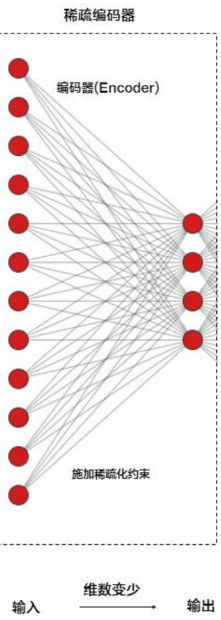


也就是说，共训练6次模型，每次训练时输入的时序长度`T`是训练集的长度(蓝色区域)，因此我们可以将输入表示为

$$X_i = (x_i^0, x_i^1, ..., x_i^T), i = 1, ..., 6$$

其中x_i^t是t时刻维度为`K=471`的因子特征向量， i是指第i次重新训练模型。

- 模型结构：
  - 稀疏自编码器层（SAE）：将高维(K=471)因子$x = (x^0, x^1, ..., x^T)$特征映射到一个相对低维(`hidden_num=d`)的空间，通过L1范数来控制稀疏约束，得到低维特征表示$z = (z^0, z^1, ..., z^T)$, 其中z^t是d维向量；**这一步的本质是降维。**



- 循环神经网络GRU层：将上步得到的低维表示用GRU unit处理，得到隐藏层序列$h = (h^0, h^1, ..., h^T)$, 其中h^t的维度可以由Pytorch函数自定义，但我根据上下文推断，这里的`GRU_hidden_num`等于SAE中的`hidden_num`. **这一步的本质是学习时序相关关系。**

- 全连接输出层：将上一步中最后的隐藏层输出h^T用全连接层处理，得到最后的预测标的输出$\hat{y}^{T+N}$,其中`N`表示对未来N日标的的涨跌幅的预测（并非多步预测，输出仅为scaler，也不是直接预测涨or跌的二分类）。因此

$$h^0, h^1, ..., h^T = gru(z^0, z^1, ..., z^T)$$
$$\hat{y}^{T+N} = decoder(h^T)$$

其中`gru()`为GRU层，`decoder()`为全连接层。

- 目标损失函数：

$$L = \sum_{i=1}^{n}(\hat{y}_i^{T+N} - y_i^{T+N})^2 + \lambda \sum_{i=1}^{n}\sum_{t=0}^{T}|z_i^t|$$

其中`n`是测试集中的待预测个数，即交易日天数， 而z_i^t是对低维向量的L1范数。

> 因此我猜测输入中的$x_i = (x_i^0, x_i^1, ..., x_i^T)$应当是根据$y_i^{T+N}$滚动更新，而不是一直从0时刻开始。

SE-GRU的模型搭建复现

- 关键模型超参：

模型	超参数	默认值
稀疏自编码层SAE	feature_num	K=471
	hidden_num	d
	λ	
GRU	lookback	T+1
	input_num	d
	output_num	d
	layer_num	
	dropout_ratio	
全连接输出层	input_num	d
	output_num	1

- SE-GRU模型用Pytorch进行模型搭建：

```
# model building
class SEGRU(torch.nn.Module):
    def __init__(self, feature_num, hidden_num, layer_num, output_num=1,
                 sparsity_weight=0.2,dropout = 0.0):
        super(SEGRU, self).__init__()
        self.feature_num = feature_num
        self.hidden_num = hidden_num
        self.output_num = output_num
        self.layer_num = layer_num

        self.sparsity_weight = sparsity_weight
        ## here dropout is not the nn.Dropout unit,
        ## but is a hyper-param of GRU unit
        self.dropout = dropout

        self.build_model()



    def build_model(self):
        ## SAE
        self.encoder = torch.nn.Sequential(
            torch.nn.Linear(self.feature_num, self.hidden_num),
            torch.nn.Sigmoid())
        ## GRU
        self.gru = torch.nn.Sequential(nn.GRU(input_size=self.hidden_num,
                                hidden_size=self.hidden_num,
                                num_layers=self.layerNum,
                                dropout=self.dropout,
                                 batch_first=True, ))
        ## Full linear layer
        self.decoder = torch.nn.Sequential(
            torch.nn.Linear(self.hidden_num, self.output_num))




    def forward(self, inputs):
        ## input should be of dimension: [batchsize, T, feature_num]
        ## z should be of dimension: [batchsize, T, hidden_num]
        low_rank_z = self.encoder(inputs)


        hidden_seq,final_hidden = self.gru(low_rank_z)

        output = self.decoder(final_hidden)
        # output is a vector of length: batchsize
```

```
        sparsity = torch.sum(torch.sum(torch.abs(low_rank_z),dim=2),dim = 1)
        # sparsity loss is a vector of length: batchsize
        return output, sparsity

    def loss(self, output, sparsity,labels):
        criterion = nn.MSELoss(reduction='sum')
        loss1 = criterion(output,labels)
        loss2 = torch.sum(sparsity)

        return loss1+self.sparsity_weight*loss2
```

> 以上是简单模型搭建的思路，但由于我并没有因子库的数据，因此没有数据处理和训练部分的代码，还望参考。

```
# to use the SEGRU network
feature_num = 471
hidden_num = 128 # should select by validation or fine-tuning
layer_num = 3 # should select by validation or fine-tuning
sparsity_weight = 0.15 # should select by validation or fine-tuning

net = SEGRU(feature_num, hidden_num, layer_num, sparsity_weight)
output, sparsity = net(train_input)
```

模型输出中的`output`是所要预测的日频标的$\hat{y}^{T+N}$，根据其正负决定未来交易日的多空决策，即

$$\hat{y}^{T+N} > 0 \Rightarrow \text{利率上行，看空}$$
$$\hat{y}^{T+N} < 0 \Rightarrow \text{利率下行，看多}$$