使用 LOB 信息预测股价涨跌的深度学习模型初步调研

摘要

根据上次 deepLOB 文章的讨论，先调研了一下 deepLOB 文章里提到的其他使用机器学习方法预测股价涨跌的文章，这次调研聚焦在使用的模型和数据上。主要的模型依次有 RR，SLFN，MDA，WMTR，MCSD，T-BoF，TABL，CNN，LSTM，CNN-LSTM，MLP，SVM。其中，除了 SVM，其他文献都来自 Alexandros Iosifidis 团队的文章。

Ridge regression and Single Layer Forward Network[1]

1. 目的
   对于使用高频数据预测股价涨跌，在使用机器学习方法时还没有一个标准的数据集和 baselines，该文章提供了这么一个数据集和 baselines。
2. 数据
   1) 原始数据：来自由 NASDAQ OMX Nordic 提供的 ITCH feed。该原始数据格式是一天一个文件，一个文件包含当天所有股票的数据。其中信息是按时间顺序发生的事件信息，事件信息包含订单提交、成交、订单取消这 3 类的信息。这 3 类信息的字段包含时间戳（精确到毫秒）、方向（买或卖）、价格、量。该文件还包含其他信息：交易的开始、结束、中间停止等。
   2) 中间数据：对上述数据做处理，获取了 5 只股票的 2010-6-1 到 2010-6-14 之间 10 个交易日的数据。这 5 只股票的信息为表 1。对于每只股票，得到两种信息，分别为 message book 和 limit book，表 2 和表 3 分别展示了股票 FI0009002422 在 2010-6-1 的这两种数据的一段样本，limit book 中的行对应 message book 中同一行事件刚发生后 limit book 的状态。
   3) 最终数据:由于上述中间数据是按照 event time 排列的，如果按照等时间间隔，同样的时间间隔里包含的 event 数目相差很大。所以，对于上述中间数据，每隔 10 个 event 生成原始特征。原始特征的定义见表 4。这样，最终的样本集（5 只股票合起来）包含的样本点就有 394,337 个，144 个自变量。

表 1

Table 1
Stocks used in the analysis

Id	ISIN Code	Company	Sector	Industry
KESBV	FI0009000202	Kesko Oyj	Consumer Defensive	Grocery Stores
OUT1V	FI0009002422	Outokumpu Oyj	Basic Materials	Steel
SAMPO	FI0009003305	Sampo Oyj	Financial Services	Insurance
RTRKS	FI0009003552	Rautaruukki Oyj	Basic Materials	Steel
WRT1V	FI0009000727	Wärtsilä Oyj	Industrials	Diversified Industrials

表 2

Table 2
Message list example

Timestamp	Id	Price	Quantity	Event	Side
1275386347944	6505727	126200	400	Cancellation	Ask
1275386347981	6505741	126500	300	Submission	Ask
1275386347981	6505741	126500	300	Cancellation	Ask
1275386348070	6511439	126100	17	Execution	Bid
1275386348070	6511439	126100	17	Submission	Bid
1275386348101	6511469	126600	300	Cancellation	Ask

表 3

Table 3
Order book example

			Level 1				Level 2				...
			Ask		Bid		Ask		Bid		
Timestamp	Mid-price	Spread	Price	Quantity	Price	Quantity	Price	Quantity	Price	Quantity	
1275386347944	126200	200	126300	300	126100	17	126400	4765	126000	2800	...
1275386347981	126200	200	126300	300	126100	17	126400	4765	126000	2800	...
1275386347981	126200	200	126300	300	126100	17	126400	4765	126000	2800	...
1275386348070	126050	100	126100	291	126000	2800	126200	300	125900	1120	...
1275386348070	126050	100	126100	291	126000	2800	126200	300	125900	1120	...
1275386348101	126050	100	126100	291	126000	2800	126200	300	125900	1120	...

表 4

Table 4
Feature Sets

Feature Set	Description	Details
Basic	$u_1 = \{P_i^{ask}, V_i^{ask}, P_i^{bid}, V_i^{bid}\}_{i=1}^{n}$	10(=n)-level LOB Data
Time-Insensitive	$u_2 = \{(P_i^{ask} - P_i^{bid}), (P_i^{ask} + P_i^{bid})/2\}_{i=1}^{n}$	Spread & Mid-Price
	$u_3 = \{P_n^{ask} - P_1^{ask}, P_1^{bid} - P_n^{bid}, \lvert P_{i+1}^{ask} - P_i^{ask}\rvert, \lvert P_{i+1}^{bid} - P_i^{bid}\rvert\}_{i+1}^{n}$	Price Differences
	$u_4 = \left\{\frac{1}{n}\sum_{i=1}^{n} P_i^{ask}, \frac{1}{n}\sum_{i=1}^{n} P_i^{bid}, \frac{1}{n}\sum_{i=1}^{n} V_i^{ask}, \frac{1}{n}\sum_{i=1}^{n} V_i^{bid}\right\}$	Price & Volume Means
	$u_5 = \left\{\sum_{i=1}^{n}(P_i^{ask} - P_i^{bid}), \sum_{i=1}^{n}(V_i^{ask} - V_i^{bid})\right\}$	Accumulated Differences
Time-Sensitive	$u_6 = \left\{dP_i^{ask}/dt, dP_i^{bid}/dt, dV_i^{ask}/dt, dV_i^{bid}/dt\right\}_{i=1}^{n}$	Price & Volume Derivation
	$u_7 = \left\{\lambda_{\Delta t}^1, \lambda_{\Delta t}^2, \lambda_{\Delta t}^3, \lambda_{\Delta t}^4, \lambda_{\Delta t}^5, \lambda_{\Delta t}^6\right\}$	Average Intensity per Type
	$u_8 = \left\{\mathbf{1}_{\lambda_{\Delta_t}^1 > \lambda_{\Delta_T}^1}, \mathbf{1}_{\lambda_{\Delta_t}^2 > \lambda_{\Delta_T}^2}, \mathbf{1}_{\lambda_{\Delta_t}^3 > \lambda_{\Delta_T}^3}, \mathbf{1}_{\lambda_{\Delta_t}^4 > \lambda_{\Delta_T}^4}, \mathbf{1}_{\lambda_{\Delta_t}^5 > \lambda_{\Delta_T}^5}, \mathbf{1}_{\lambda_{\Delta_t}^6 > \lambda_{\Delta_T}^6}\right\}$	Relative Intensity Comparison
	$u_9 = \{d\lambda^1/dt, d\lambda^2/dt, d\lambda^3/dt, d\lambda^4/dt, d\lambda^5/dt, d\lambda^6/dt\}$	Limit Activity Accelaration

4) 标准化：尝试了 3 种标准化。分别为 z-score，max-min，decimal precision。具体公式分别为

$$x_i^{(Z_{score})} = \frac{x_i - \frac{1}{N}\sum_{j=1}^{N} x_j}{\sqrt{\frac{1}{N}\sum_{j=1}^{N}\left(x_j - \bar{x}\right)^2}} \tag{0.1}$$

$$x_i^{(MM)} = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}} \tag{0.2}$$

$$x_i^{(DP)} = \frac{x_i}{10^k} \tag{0.3}$$

其中，k 是使得$\left|x^{(DP)}\right|<1$的最大整数。

用了多长的数据做标准化（用多长的数据计算均值、标准差、最大值等），文章未提。

5) 因变量（标签）：对于上述最终数据的一个样本，其因变量为接下来k 个（不是公式(0.3)中的k ，k 取 1，2，3，5，10。）event 处的中间价的均值相对于该样本最后一个 event 处的中间价的涨跌幅。公式为

$$l_i^{(k)}=\frac{\frac{1}{k}\sum_{j=i+1}^{i+k}m_j-m_i}{m_i} \tag{0.4}$$

。然后，对于涨跌幅大于等于 0.002 的，标记为 1，对于涨跌幅小于等于-0.002 的，标记为 3，其他的标记为 2。

3. 实验框架

总共有 10 个交易日数据。计算某一模型的表现时，将第 1 个交易日作为训练集，第 2 个交易日作为测试集；第 1 到 2 个交易日作为训练集，第 3 个交易日作为测试集；第 1 到 3 个交易日作为训练集，第 4 个交易日作为测试集；...；将第 1 到 9 个交易日作为训练集，第 10 个交易日作为测试集。如图 1。这样总共有 9 个测试集，然后在每个测试集上使用其对应的训练集上训练出的模型计算评估指标，然后用着 9 个测试集上的评估指标计算评估指标的均值和标准差。

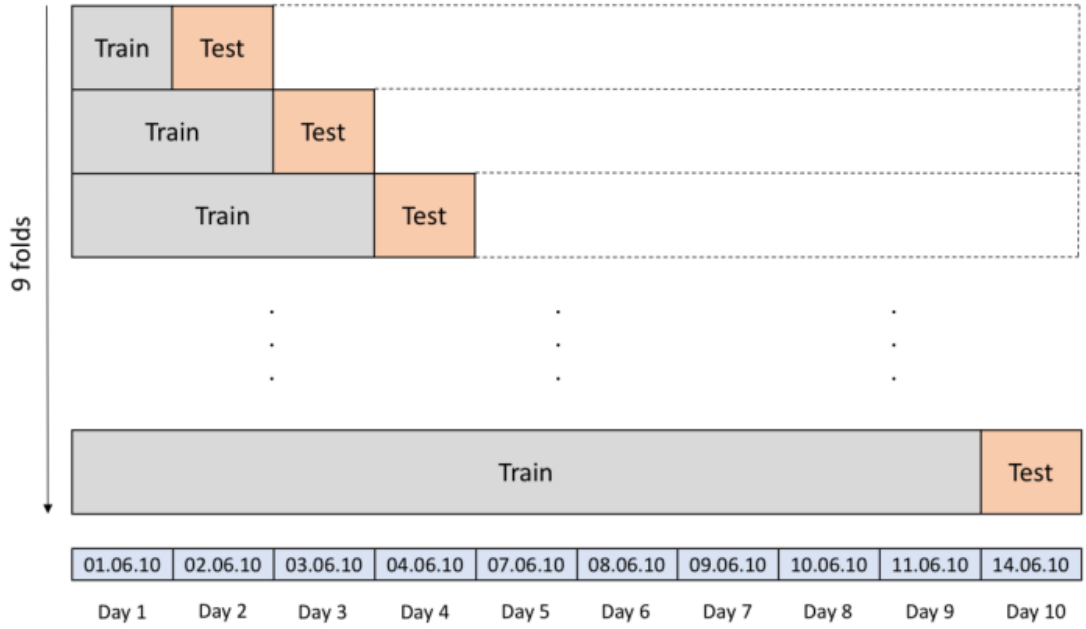

Fig. 2. Experimental Setup Framework

图 1

4. 模型

有 2 个模型，该文章将这两个模型作为 baselines，供以后的模型与其比较。这两个模型分别为线性和非线性。线性模型为岭回归（Ridge Regression，RR），非线性模型为单隐层前馈神经网络（Single Layer Forward Network，SLFN）。对于这两个模型，都是将因变量转换成 one-hot 的 3 维向量，然后对该因变量使用回归，然后分类的时候选取对这 3 个因变量的预测值最大的成分作为预测的类型。

1) RR

就是带对系数L^2惩罚项的线性回归。其中涉及到超参数惩罚系数λ 。文章没有提怎么选取。

2) SLFN

该网络的第一层为 RBF 网络

$$h_{ik} = \exp\left(\frac{\|\mathbf{x}_i - \mathbf{v}_k\|_2^2}{2\sigma^2}\right), k = 1,...,K \tag{0.5}$$

，其中，i 表示第i 个样本；k 表示第k 个隐藏单元，总共有K 个隐藏单元；$\mathbf{v}_k$表示第k 个隐藏单元的均值参数，可以通过$k-means$ 聚类得到。σ^2为超参。超参K和σ^2的选取，文章未提。

该网络的第二层为隐藏单元与输出层的线性全连接，没有激活函数，输出层有三个单元，隐藏层与输出层的系数的估计使用岭回归方式，只不过自变量从原始自变量变成了隐藏单元。同样λ 的选取文章未提。

Multilinear Discriminant Analysis[2]

1. 目的

使用上一篇文章的数据，比较 tensor 方法（使用的自变量不只有当期的，还有最近多期的）和向量方法（使用的自变量只有当期的）优劣。

2. 数据

同第一篇文章的最终数据，不过自变量用到了第一篇文章中最终数据最近 10 个样本点，即，自变量的维度为144×10 ，相当于用到了最近 100 个 event 的信息（因为上文的最终数据一个点包含了 10 个 event 的信息）。

另外，数据标准化，只用了 zscore，因变量只用了未来 10 个 event 中间价均值相对于当前均值的涨跌幅。

3. 实验框架

同第一篇文章。

4. 模型

主要使用了两个模型，分别为 Multilinear Discriminant Analysis（MDA）和 Weighted Multichannel Time-series Regression（WMTR）。也顺便用了 LDA 和 MTR。

1) MDA

MDA 是 LDA 的扩展。LDA 的自变量是向量，MDA 的自变量是 tensor。

记我们的自变量为 tensor $X \in R^{I_1\times I_2\times...\times I_K}$ ，其中，K 为 tensor 的 mode 数目，$I_k, k = 1,...,K$ 为第k 个 mode 的维度。对于该文章的自变量为，$X \in R^{D\times T}$ ，$K = 2, D = 144, T = 10$，144 表示 144 个特征，10 表示 10 个时间点。记属于第i 类

的第 j 个样本的自变量为 $X_{i,j}$ ，属于第 i 类的样本有 n_i 个，总共有 C 类。MDA 是寻找一组投影矩阵 $\mathbf{W}_k \in R^{I_k \times I_k'}, I_k' < I_k, k = 1,...,K$ ，该组投影矩阵将样本点 $X_{i,j}$ 映射到 $Y_{i,j} \in R^{I_1' \times I_2' \times ... \times I_K'}$ ：

$$Y_{i,j} = X_{i,j} \prod_{k=1}^{K} \times_k \mathbf{W}_k^{\mathrm{T}} \tag{0.6}$$

，其中，

$$X_{i,j} \prod_{k=1}^{K} \times_k \mathbf{W}_k^{\mathrm{T}} = X_{i,j} \times_1 \mathbf{W}_1^{\mathrm{T}} ... \times_K \mathbf{W}_K^{\mathrm{T}} \tag{0.7}$$。

其中，

$$\left(X \times_k W \right)_{(k)} = W \times X_{(k)} \tag{0.8}$$，

$X_{(k)}$ 表示 tensor X 按照第 k 个 mode 展开（即，一个矩阵，其行数为 I_k 行，列数为 $\prod_{j \neq k} I_j$ ）。

投影矩阵的估计通过最大化类间距离与类内距离的比值来得到，即最小化

$$J\left(\mathbf{W}_1,..., \mathbf{W}_K \right) = \frac{D_b}{D_w} \tag{0.9}$$，

其中，

$$D_b = \sum_{i=1}^{C} n_i \left\| M_i \prod_{k=1}^{K} \times_k \mathbf{W}_k - M \prod_{k=1}^{K} \times_k \mathbf{W}_k \right\|_F^2 \tag{0.10}$$

$$D_w = \sum_{i=1}^{C} \sum_{j=1}^{n_i} \left\| X_{i,j} \prod_{k=1}^{K} \times_k \mathbf{W}_k - M_i \prod_{k=1}^{K} \times_k \mathbf{W}_k \right\|_F^2 \tag{0.11}$$

其中， M_i 为第 i 类样本自变量的均值， M 为所有样本均值。

具体的迭代估计方式见原文。最后分类的时候，对于一个新的样本，根据(0.6)生成其在子空间中的投影，然后比较该投影与各类型的样本均值在子空间中的投影的距离，距离谁最近，就归为哪类。

尝试的超参：最优化的迭代数，50；最优化的收敛阈值，1e-6；投影矩阵的第一个 mode（即 I_1' ）为 5 到 60 之间以 5 为步长的数；投影矩阵的第二个 mode（即 I_2' ）为 1 到 8 之间以 1 为步长的数；最优化时某一步中的正则化常数 $\lambda \in \{0.01, 0.1, 1, 10, 100\}$ 。

2) WMTR

WMTR 学习下面的函数

$$f\left(X_i\right)=\mathbf{W}_1^{\mathrm{T}}X_i\mathbf{w}_2 \tag{0.12}$$，

其中，$\mathbf{W}_1\in R^{D\times C}$ ，$\mathbf{w}_2\in R^T$ 。其系数的学习通过最小化下面的目标函数得到：

$$J\left(\mathbf{W}_1,\mathbf{w}_2\right)=\sum_{i=1}^{N}s_i\left\|\mathbf{W}_1^{\mathrm{T}}X_i\mathbf{w}_2-\mathbf{y}_i\right\|_F^2+\lambda_1\left\|\mathbf{W}_1\right\|_F^2+\lambda_2\left\|\mathbf{w}_2\right\|_F^2 \tag{0.13}$$

其中，$\mathbf{y}_i$ 是 one-hot 因变量（但是原文说是如果是该类型，则取 1，非该类型则取 -1）。

另外，

$$s_i=1/\sqrt[r]{N_{c_i}},r>0 \tag{0.14}$$

表示第 i 个样本的权重与该样本所属类型的样本个数开 r 次方成反比，r 越小，样本数越小的类型权重越高，该权重用来处理分类问题中的 unbalance 问题。最后分类的时候，对于一个新样本，根据(0.12)计算特征，然后该向量的元素哪个最大就归为哪一类。具体最优化过程见原文。

尝试的超参：最优化的迭代数，50；最优化的收敛阈值，1e-6；$\lambda\in\{0.01,0.1,1,10,100\}$；$r\in\{2,3,4\}$。

3) LDA

就是普通的 LDA，自变量为当期的 144 维向量。

4) MTR

WMTR 中的 s_i 为 1。

Multilinear Class-Specific Discriminant Analysis（MCSDA）[3]

MDA 的一个变种。

其原理如下。沿用 MDA 的符号。对于第 i 类，将标签为这一类的样本记为正例，用下标 p 表示。 寻找针对于这一类的投影矩阵 $\mathbf{W}_k\in R^{I_k\times I_k'},I_k'<I_k,k=1,...,K$，使得

$$J\left(\mathbf{W}_1,...,\mathbf{W}_K\right)=\frac{D_O}{D_I} \tag{0.15}$$

最小。其中，

$$D_O = \sum_{j,l_j \neq p} \left\| X_j \prod_{k=1}^{K} \times_k \mathbf{W}_k - M_p \prod_{k=1}^{K} \times_k \mathbf{W}_k \right\|_F^2 \quad (0.16),$$

$$D_I = \sum_{j,l_j = p} \left\| X_j \prod_{k=1}^{K} \times_k \mathbf{W}_k - M_p \prod_{k=1}^{K} \times_k \mathbf{W}_k \right\|_F^2 \quad (0.17),$$

其中，

$$M_p = \frac{1}{n_i} \sum_{j,l_i = p} X_j$$

为这一类的自变量均值。

上述表述中，$\mathbf{W}_k \in R^{I_k \times I_k'}, I_k' < I_k, k = 1,...,K$ 没有加类型下标 i ，是为了简单。根据上述原理，每一类都能得到特定的（Class-specific）该投影矩阵。最后，分类的时候，对于一个新样本，对于每一类，使用该类特定的投影矩阵计算该样本点与该类均值在子空间中的距离，然后哪一类对应的距离最小就归为哪一类。原文建议也可以将这 C 个距离作为其他分类模型的特征。

Temporal Bag-of-Features [4]

1. 数据

   上述最终数据。

2. 网络结构

T-BoF 的网络整体见图 2。

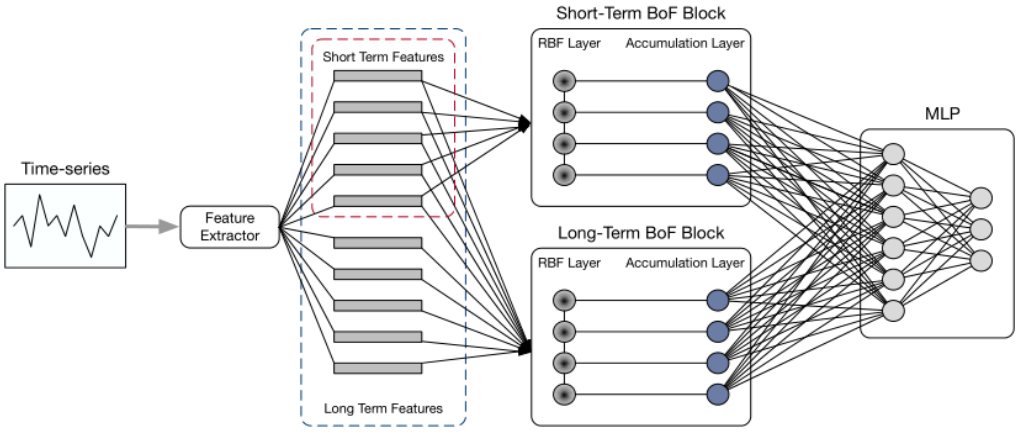

Fig. 2: The architecture of the proposed Temporal BoF model

图 2

记第 i 个自变量为 $\left\{\mathbf{x}_{ij} \in R^D, j = 1,...,T_i\right\}$ ，在该应用中 D=144 ，T_i 是不随 i 变化的，比如取 20，表示使用最近 20 个最终特征。j 在这里表示时间（离当前时间的距离），j 越小表示时间越靠后。图 2 中的 feature extractor 后面的方框中就是将 $\left\{\mathbf{x}_{ij} \in R^D, j = 1,...,T_i\right\}$ 的

成分按 j 由小到大排下来。 然后，对每一个 $\mathbf{x}_{ij}$ 应用 RBF，这里用了两个 RBF，一个是短期的，一个是长期的，短期的只应用在前 $N_1 < T_i$ 个 $\mathbf{x}_{ij}$ 上，长期的应用在前 $N_0 \le T_i$ 个 $\mathbf{x}_{ij}$ 上，该应用设定 N_0=T_i 。这个 RBF 的输出为

$$\phi_{mk}\left(\mathbf{x}_{ij}\right)=\frac{\exp\left(-\left\|\left(\mathbf{x}_{ij}-\mathbf{v}_{mk}\right)\odot\mathbf{w}_{mk}\right\|_2^2\right)}{\sum_{\kappa=1}^{N_K}\exp\left(-\left\|\left(\mathbf{x}_{ij}-\mathbf{v}_{m\kappa}\right)\odot\mathbf{w}_{m\kappa}\right\|_2^2\right)} \quad (0.18),$$

其中，$\odot$ 为 element-wise 乘法，$\mathbf{x}\in R^D$，$m=0$ 表示长期，$m=1$表示短期，k 表示 RBF 输出的第 k 神经元，RBF 总共输出 N_K 个神经元（长期和短期的个数一样），分母用来将 RBF 的输出做归一化，即这 N_K 个输出加和是 1。

在 RBF 层之上，是累加层，其作用如下，对于 $m=0$ ，将前 N_0 个 $\mathbf{x}_{ij}$ 输出的 $\phi_0\left(\mathbf{x}_{ij}\right)$ 相加；对于 $m=1$ ，将前 N_1 个 $\mathbf{x}_{ij}$ 输出的 $\phi_1\left(\mathbf{x}_{ij}\right)$ 相加，记

$$\phi_m\left(\mathbf{x}_{ij}\right)=\left(\phi_{mk}\left(\mathbf{x}_{ij}\right)\right)_{k=1}^{N_K} \quad (0.19),$$

则

$$\mathbf{h}_{mi}=\frac{1}{N_m}\sum_{j=1}^{N_m}\phi_m\left(\mathbf{x}_{ij}\right) \quad (0.20)。$$

可以看到，$\mathbf{x}_{ij}$ 与 $\mathbf{v}_{mk}$ 越接近，$\phi_{mk}\left(\mathbf{x}_{ij}\right)$越大。$\left\{\mathbf{v}_{mk}\right\}_{k=1}^{N_K}$ 就像 N_K 个词，$\mathbf{h}_{mi}$ 就像在计算这 N_K 个词各自在 $\left\{\mathbf{x}_{ij}\in R^D, j=1,...,T_i\right\}$ 中出现的比例，所以叫做 bag-of-features。

然后，将长期 $\mathbf{h}_{0i}$ 和短期 $\mathbf{h}_{1i}$ 并起来，作为特征

$$\mathbf{h}_i=\begin{pmatrix}\mathbf{h}_{0i}\\ \mathbf{h}_{1i}\end{pmatrix} \quad (0.21)。$$

然后，将 $\mathbf{h}_i$ 上叠加单隐层分类器，即

$$\mathbf{p}_i=\phi^{(elu)}\left(\mathbf{W}_H\mathbf{h}_i+\mathbf{b}_H\right) \quad (0.22)$$

$$\mathbf{y}_i=\phi^{(elu)}\left(\mathbf{W}_O\mathbf{p}_i+\mathbf{b}_O\right) \quad (0.23)$$

$\mathbf{y}_i$ 为三种类型的概率，其中，

$$\phi^{(elu)}(x)=\begin{cases} x, x>0 \\ \alpha_{elu}\left(\exp(x)-1\right), x\le 0 \end{cases} \quad (0.24)。$$

损失函数使用交叉熵。

Temporal Attention augmented Bilinear Network（TABL）[5]

1. 数据
   最终数据。
2. 网络结构
   1) Bilinear Layer（BL）

   自变量记为 $X_i \in R^{D\times T}$ ，第一个 mode 是特征维度，第二个维度是时间维度。BL 层将 $D\times T$ 的输入转换为 $D'\times T'$ 的输出，

   $$\mathbf{Y}=\phi\left(\mathbf{W}_1\mathbf{X}\mathbf{W}_2+\mathbf{B}\right) \quad (0.25)，$$

   $\mathbf{W}_1 \in R^{D'\times T}, \mathbf{W}_2 \in R^{T\times T'}, \mathbf{B} \in R^{D'\times T'}$ 是参数。

   BL 相对于传统的全连接，参数规模要小；其相对于全连接的约束是先在第一个维度上 combine，再在第 2 个维度上 combine，2 个维度没有交叉的 combine。

   2) Temporal Attention augmented Bilinear Layer（TABL）

   在 BL 的基础上引入在时间维度的 Attention 机制。TABL 具体为：

   $$\bar{\mathbf{X}}=\mathbf{W}_1\mathbf{X} \quad (0.26)$$

   $$\mathbf{E}=\bar{\mathbf{X}}\mathbf{W} \quad (0.27)$$

   $$\alpha_{ij}=\frac{\exp\left(e_{ij}\right)}{\sum_{k=1}^{T}\exp\left(e_{ik}\right)} \quad (0.28)$$

   $$\tilde{\mathbf{X}}=\lambda\left(\bar{\mathbf{X}}\odot\mathbf{A}\right)+\left(1-\lambda\right)\bar{\mathbf{X}} \quad (0.29)$$

   $$\mathbf{Y}=\phi\left(\tilde{\mathbf{X}}\mathbf{W}_2+\mathbf{B}\right) \quad (0.30)，$$

其中，e_{ij} 为 $\mathbf{E}$ 的 ij 元，α_{ij} 为 $\mathbf{A}$ 的 ij 元，$\odot$ 为 element-wise 乘法，$\mathbf{W}$ 是 T 列的。(0.26)与 BL 的 $\mathbf{W}_1$ 作用相同，即在同一个时间点上将特征维度线性组合产生新的特征，(0.27)，(0.28)，

(0.29)表示 attention 机制，$\bar{\mathbf{X}} \odot \mathbf{A}$ 表示对于 $\bar{\mathbf{X}}$ 中的每个元素 $\bar{x}_{ij}$ 乘以一个权重 α_{ij}，而(0.28)表示这些权重在同一行加和为 1，即这些权重表示在时间维度上的权重，(0.27)表示这些权重与初步转化后的特征有关。另外，文章设定(0.27)中 $\mathbf{W}$ 的对角线元素为 $1/T$ 。

3) 使用 BL 和 TABL 搭建网络

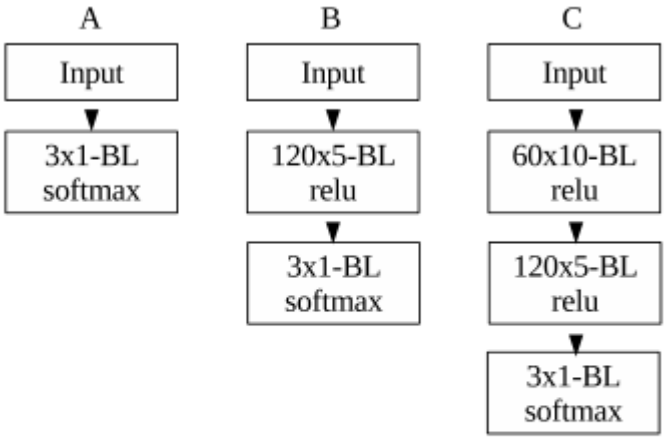

Fig. 2. Baseline Network Topologies

图 3

前面，只是介绍了 BL 和 TABL 的一层。在搭建网络是，可以使用多层 BL 和 TABL 层。在分类问题上，将最后一层的激活函数使用 sotfmax。该文章分别尝试了一层、二次、三层的 BL，如图 3，分别记为 A（BL），B（BL），C（BL）。对于 TABL，该文章只是在上述三个网络的最后一层应用了 TABL，其他层仍然是 BL，对应的模型分别记为 A（TABL），B（TABL），C（TABL）。

CNN-1[6]

1. 数据

   使用的数据不再是最终数据，而是中间数据中的 10 档 orderbook 信息。即每 10 个 event 一个点，而是一个 event 一个点。自变量为 $X_i \in R^{100\times 40}$，100 表示使用最近 100 个 event 的 LOB 信息，40 表示一个 event 时间点有 40 个变量，分别是 bid 和 ask 端各 10 档的量价数据。

   前面的最终数据可以这样理解，如果自变量是最终数据的最近 10 个点构成，则也使用了最近 100 个 event 的 LOB 信息，只不过想人工合成了 144 个特征。
2. 实验框架

   将 10 个交易日的数据划分，前 7 天为训练集，后 3 天为测试集。该框架称为 setup 2，第一篇文章的框架称为 setup 1。
3. 网络结构

   网络结构见图 4。

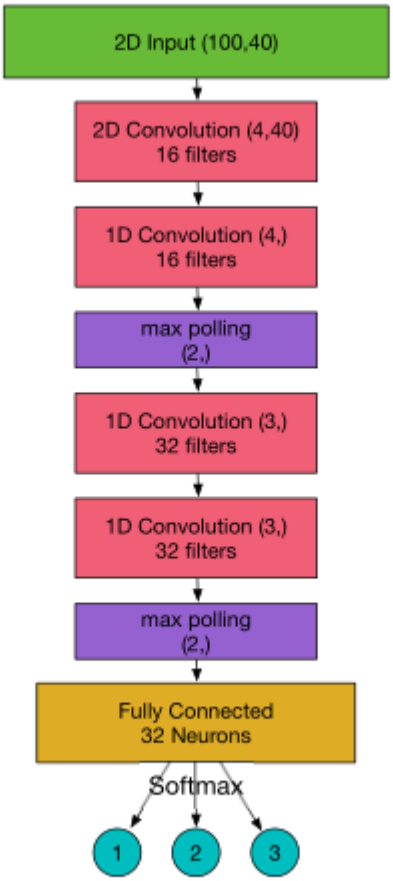

Fig. 3: A visual representation of the evaluated CNN model

图 4

LSTM[7]

数据同 CNN-1，使用的网络是标准的 LSTM。

但是，该文章有个问题，我们计划用最近的 100 个 event 的 LOB 信息预测接下来价格的涨跌，那么应该符合 DeepLearningBook 中 RNN 的输入为 sequence 且输出为一个大小不随输入长度变化的向量的模型；而该文章的 RNN 为输入为 sequence，输出为同样长度的 sequence，即对应于自变量的每个 event 时间点，都有一个对涨跌幅的预测，这地方应该不合理。

在训练网络的时候，该文章中有下面一段话（图 5 和图 6），也跟使用的输入为 sequence 输出为同样长度的 sequence 的 RNN 有关。

In our first attempt to train an LSTM network to predict the mid-price trend direction we noticed a very interesting pattern in the mean cost per recurrent step, as shown in Figure 1. The cost is significantly higher on the initial steps before it eventually settles. This happens because it is not possible for the network to build a correct internal representation having seen only a few samples of the depth. To avoid this unnecessary source of error and noise in the training gradients, we do not propagate the error for the first 100 recurrent steps. These steps are treated as a "burn-in" sequence, allowing the network to observe a portion of the LOB depth timeline before making an accountable prediction.

图 5

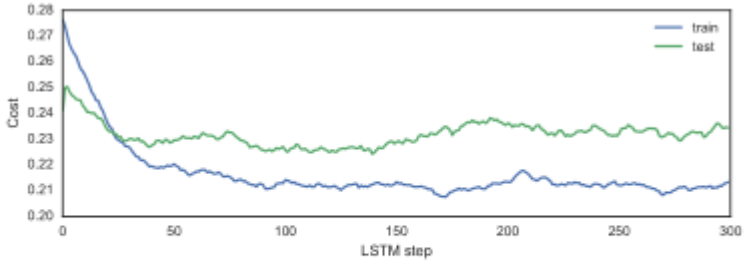

Fig. 1: Mean cost per recurrent step of the LSTM network

图 6

CNN-LSTM[8]

1. 数据

   使用的数据同 CNN-1。做了两种处理。

   1) 数据不处理。一个时间点有 40 个变量。
   2) 为了使得数据更加平稳，做了如下操作。对于 bid 端的 10 档价格，计算相对于中间价的涨跌幅来代替价格本身。为了弥补价格本身信息的消失，增加一个变量，当前 event 的中间价相对于前一个 event 的中间价的涨跌幅。另外，对于 bid 端的 10 档挂单量，也用从 bid1 档到该档的累积挂单量来代替。

      然后，用前一天这些变量的均值和标准差，来对当日的这些变量计算 z-score，作为最终的变量。

      最后，一个时间点有 41 个变量。

2. 网络结构

   网络结构见图 7。

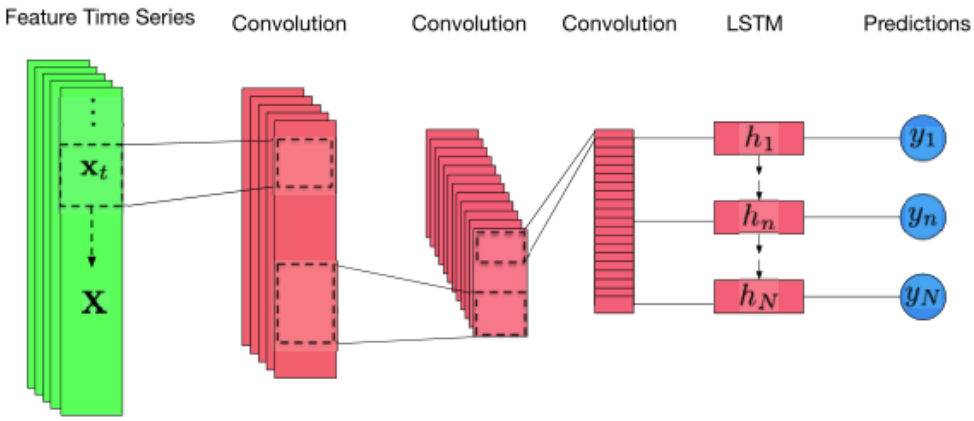

Figure 2: CNN-LSTM model

图 7

对输入数据，先在时间维度上应用几层卷积（CNN 部分），然后，将其输出作为 LSTM 的输入。

该文的 CNN 部分具体见图 8。

1. 1D Convolution with 16 filters of size $(5, 42)$
2. 1D Convolution with 16 filters of size $(5,)$
3. 1D Convolution with 32 filters of size $(5,)$
4. 1D Convolution with 32 filters of size $(5,)$

图 8

MLP

上述这些文章里，也有一些测试了普通的神经网络（即 MLP），用来做比较。其输入，在 10 个 event 合成一个时间点的数据情况下，使用一个时间点的变量（144）作为输入，【4】中使用了 512 个神经元的单隐层 MLP；在 1 个 event 作为 1 个时间点的数据情况下，使用多个（如 50 个）时间点的变量作为输入，【7】中使用了 128 个神经元的单隐层的 MLP，【8】中使用了有 3 个隐藏层的 MLP，隐藏层的神经元个数分别为 126,64,32。

SVM[9]

上述文献中的 144 个特征来自于该 SVM 的文章。该文章于前面的文章有一下几点区别

1. 数据因变量

   NASDAQ 的 5 只股票（SMFT,INTC,AMZN,AAPL,GOOG，5 只股票各自跑各自的模型）在某一天的数据，messagebook 和 limitbook。对于同一只股票，一个时间点的数据就包含 message book 和 limit book 在该时间点的信息，将这所有的时间点的信息作为原始数据。每个时间点定义因变量。因变量有两种定义。分别为

   1) Midprice 法：Δt 个 event 后的中间价相对于当前时刻中间价的涨幅，如果该涨幅大于 0，则标记为+1，如果该涨幅小于 0，则标记为-1，如果该涨幅等于 0，则标

记为 0。

2) Spread-cross 法：Δt 个 event 后的一档 bid 价大于当前时刻一档 ask 价，则标记为+1；Δt 个 event 后的一档 ask 价小于当前时刻一档 bid 价，则标记为-1；其他情况标记为 0。

2. 数据采样

然后对于某只股票，某个因变量，从原始数据中抽样，抽样时，保持抽取的样本当中因变量类型的比例与原始数据中的一样，然后对于抽样得到的时间点按照表 4 计算自变量。这样得到的数据用来训练与评估。

3. 变量筛选

另外，该文章没有将上述自变量全部加进去，而是做了一步筛选。具体先为单个自变量计算一个指标，该指标在该文章中为 information gain（IG），然后将指标按照该指标由大到小排列，往模型中按该顺序逐步加入自变量，直到模型的 F-beta 达到全部自变量都加入的模型的 F-beta 的 95%。该文章没有说明，我推测，计算 IG 时使用训练集，计算 F-beta 时使用验证集。

IG 的计算为

$$IG = H(Y) - H(Y|X) \tag{0.31}$$

其中

$$H(Y) = -\sum_{y \in Y} p(y) \log_2 \left(p(y) \right) \tag{0.32}$$

$$H(Y|X) = -\sum_{x \in X} \left(p(x) \sum_{y \in Y} p(y|x) \log_2 \left(p(y|x) \right) \right) \tag{0.33}$$

$H(Y)$ 表示因变量的熵，$H(Y|X)$ 表示因变量条件在自变量上的条件熵的无条件期望。

参考文献

1. Adamantios Ntakaris & Martin Magris & Juho Kanniainen & Moncef Gabbouj & Alexandros Iosifidis, 2018. "**Benchmark dataset for mid-price forecasting of limit order book data with machine learning methods**," Journal of Forecasting, John Wiley & Sons, Ltd., vol. 37(8), pages 852-866, December
2. Dat Thanh Tran & Martin Magris & Juho Kanniainen & Moncef Gabbouj & Alexandros Iosifidis, 2017. "**Tensor Representation in High-Frequency Financial Data for Price Change Prediction**," Papers 1709.01268, arXiv.org, revised Nov 2017.
3. Thanh Tran, Dat, Moncef Gabbouj, and Alexandros Iosifidis. “Multilinear Class-Specific Discriminant Analysis.” Pattern Recognition Letters 100 (2017): 131–136. Crossref. Web.
4. Nikolaos Passalis & Anastasios Tefas & Juho Kanniainen & Moncef Gabbouj & Alexandros Iosifidis, 2019. "**Temporal Logistic Neural Bag-of-Features for Financial Time series Forecasting leveraging Limit**

Order Book Data," Papers 1901.08280, arXiv.org.

5. Dat Thanh Tran & Alexandros Iosifidis & Juho Kanniainen & Moncef Gabbouj, 2017. "**Temporal Attention augmented Bilinear Network for Financial Time-Series Data Analysis**," Papers 1712.00975, arXiv.org.
6. A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj and A. Iosifidis, "Forecasting Stock Prices from the Limit Order Book Using Convolutional Neural Networks," *2017 IEEE 19th Conference on Business Informatics (CBI)*, Thessaloniki, 2017, pp. 7-12.
7. A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj and A. Iosifidis, "Using deep learning to detect price change indications in financial markets," *2017 25th European Signal Processing Conference (EUSIPCO)*, Kos, 2017, pp. 2511-2515.
8. Avraam Tsantekidis & Nikolaos Passalis & Anastasios Tefas & Juho Kanniainen & Moncef Gabbouj & Alexandros Iosifidis, 2018. "**Using Deep Learning for price prediction by exploiting stationary limit order book features**," Papers 1810.09965, arXiv.org
9. Alec N. Kercheval & Yuan Zhang, 2015. "**Modelling high-frequency limit order book dynamics with support vector machines**," Quantitative Finance, Taylor & Francis Journals, vol. 15(8), pages 1315-1329, August.

下一步工作

1. 做一个线性的
2. 做一个简单的深度学习。深度学习最好能使用别人预训练好的，因为自己训练起来难度大。