CS410课程大作业报告

CONTENTS

I. 问题背景

股指期货价格的波动是盈利的根本来源，人们通常通过对规则的学习预测股指期货走势，调整投资策略。影响股指价格变动的因素有很多，有长期的也有短期的，最终都反映在买卖博弈中。因此，从这个角度上来说，买卖博弈是影响价格变动最直接的因素。

但是，庞大的股票市场每时每刻都产生成大量的数据，如何利用这些数据从而做出好的决策是不少投资公司所关注的。在对于订单簿（Order Book）的动态建模，有传统的计量经济学研究方法，经典的例如：研究价差分析的MRR模型 [1]、Huang和Stoll分解模型 [2]，研究订单持续期的ACD模型 [3]。随着机器学习领域的不断发展，近年来，人们越来越开始关注引入机器学习算法来进行股价预测 [4] [5],。

本次课程设计内容为，通过对订单簿中数据的学习，预测下20个时间点中间价（MidPrice）的均值。本次课程设计的重点在于，巩固实践课堂所学内容，拓展阅读前沿科研成果，探索机器学习在股指期货价格预测中的应用。

我们所做工作的流程图如Fig. 3

II. 原始数据分析

在这一步要做的基本就是EDA (Exploratory Data Analysis)，也就是对数据进行探索性的分析，从而为之后的处理和建模提供必要的结论。

本次课程中，我们使用pandas 来载入数据，并做一些简单的可视化来理解数据。

首先，是一些订单簿数据的类型：

- Date - 日期
- Time - 时间
- MidPrice - 中间价（买入价与卖出价的平均值）
- LastPirce - 最新成交价
- Volume - 当日累计成交数量
- BidPrice1 - 申买最高价
- BidVolume1 - 申买最高价对应的量
- AskPrice1 - 申卖最高价
- AskVolume1 - 申卖最高价对应的量

我们的任务是给定连续十个时间点的以上特征，预测下20个时间点MidPrice的均值。

此外，还有其他一些有关订单簿的特性:

- 交易时间为工作日9:30-11:30, 13:00-15:00
- 股价的形成：有9:15-9:25的集合竞价（本次任务中不考虑），连续竞价：9:30之后，根据买卖双方的委托形成价格。
- 竞价原则：价格优先、时间优先
- 确立成交价原则，申买价>=申卖价时成交，最新成交价更新。申买量>申卖量，价格走高，反之价格走低。

接下来是数据的基本可视化：

由Fig. 1，Fig. 1以及通过对数据中日期时间的观察，我有以下几点结论：

- 对于Volume相关的特征，训练数据和测试数据的分布类似，而且都很像长尾分布，可以考虑使用log(x+1)这种机器学习中处理长尾分布常见的变换来让其变成近似正态分布

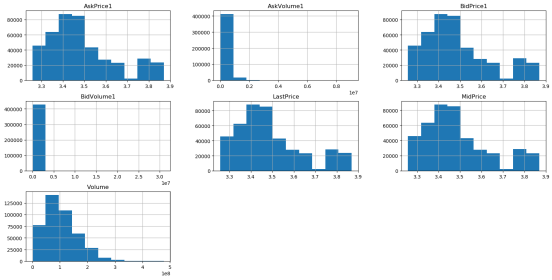

Fig. 1. 训练数据直方图

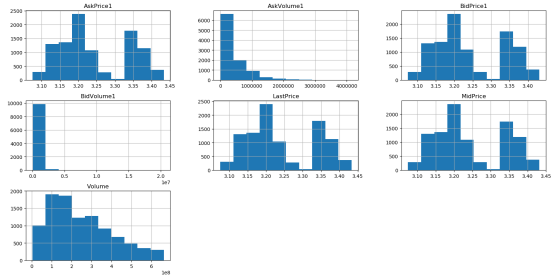

Fig. 2. 测试数据直方图

- 对于日期特征，训练数据是2018.6.1-2018.9.28（不包括周六日共84天）全天的数据，测试数据是2018.10.8-2018.10.19（不包括周六日共10天）全天的数据。由于涉及到的时间区间很短，因此考虑增添日期相关变量时，我只提取出该数据是周一到周五哪一天的特征（独热编码），没有设计月和年相关的变量
- 对于时间特征，都是每天9:30-11:30 13:00-15:00的数据，但是数据是Dirty的，比如有连续两条一模一样的数据（包括日期和时间），比如有12:00的数据（这时候是休市的），这都需要我们去处理，不然的话相当于添加了大量噪音。
- 对于价格特征，可以观察到训练数据和测试数据的分布类型是相似的，但是它们的值是非常不同的，而且由于价格的特殊性，如果直接归一化，其实会导致：面对没有出现过的值，模型难以做出很好的预测，这一点在 [6]有更详细的描述。这也需要我们去设计方法来进行归一化。

III. 数据清洗

Garbage in, garbage out! - 计算机领域经典谚语

在初步了解了数据的概况之后，面对Sec. II中提到的问题，我们对数据进行了清洗：

1) 对于日期中异常的时间，比如11:30:xx, 12:00:xx, 12:59:xx，我们选择了丢弃训练数据中的这些数据。这时候想到问题就是测试数据中，这种异常的数据我们该怎么预测呢，通过一个Python小脚本检查，我们发现只有一组预测12:00:xx的数据，我们采取的策略是输出前强制把该组的预测结果改为给定数据的最后

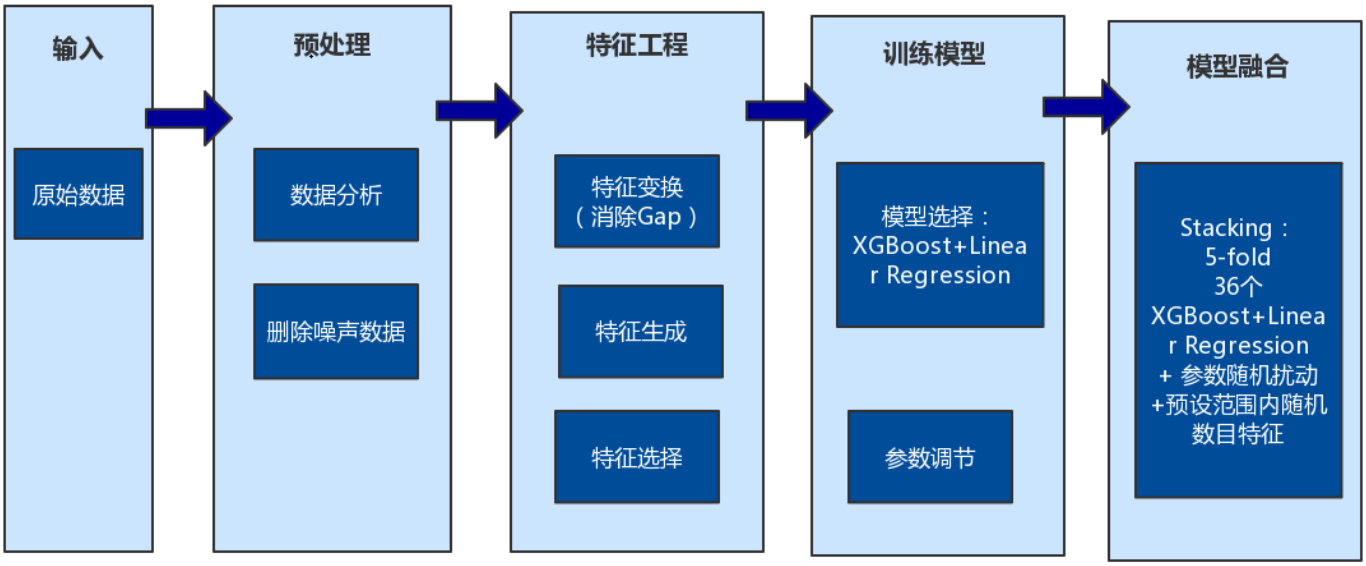

Fig. 3. Kaggle比赛工作流程图

一个MidPrice，因为通过检查训练数据中这些异常时间的数据，发现由于是休市，各个Feature是不变的。

2) 对于前后两行完全一样的问题，我们选择去掉了训练数据中所有的重复数据。有人可能会问：这样做会不会导致模型无法学会处理重复数据，导致预测测试数据时误差很大呢？经过仔细检查测试数据，我们发现重复状况只发生在测试数据的11:29:xx，也就是中午休市前，我们采取的策略是输出前强制把中午休市前最后一组的预测结果改为给定数据的最后一个MidPrice，因为通过检查训练数据中中午休市前的数据，发现由于是休市前，各个特征变化极小。
3) 除此之外，由于午间休市和第二天早晨公开竞价的存在，我们认为让时间窗口跨过中午\下午会增添很多噪声，而且测试数据中也没有跨中午\下午预测的组，因此我们在划分训练数据的组时，确保时间窗口不穿过中午\下午。

以上数据清洗的流程减少了大量噪声，实战过程中，他有效的降低了本地的cross validation loss（由于该工作是最早期进行的，没有提交到Public Board上对比测试）。

IV. 特征工程

有人总结Kaggle比赛是特征为主，调参和集成为辅。（我个人感觉在不大幅改变模型结构、不增加或删除特征、不改进任务的实现方式的情况下，调参能做的是找到当前设计下的一个较优解，继续疯狂调参的行为没有什么意义）。特征工程能做到什么程度，取决于对数据领域的了解程度。怎么构造有用的特征，是一个不断学习和提高的过程。一般来说，当一个变量从直觉上来说对所要完成的目标有帮助，就可以将作为特征。总的来说，我们应该生成尽量多的特征，相信模型能够挑出最有用的特征。

因此我们组也花费大量时间做了特征工程，下面我们将一一介绍：

1) 对于日期特征，如Sec. II 中所提，没有必要生成月和年的特征，因此只生成了这是一周的周几的特征(1,2,3,4,5)。
2) 对于时间特征，为了减少变量个数，我没有直使用Unix时间或这是一天的第多少秒，而是使用这是一天第几个小时(9,10,11,13,14)，特别的，对于训练和测试数据中12点和15点的数据令其该特征值为11和14以尽可能减少类别数量
3) 对于上述两种离散的类别特征，我们开始可以用(1,2,3,4,5)或(9,10,11,13,14)来表示，但这样的特征处理并不能直接放入机器学习算法中，因为类别之间是无序的（例如周五到下周一和到这周二相隔的天数一样，因此我们认为这是类别的）。因此我们需要**独热编码**来表示。

   独热编码，又称为一位有效编码，主要是采用N位状态寄存器来对N个状态进行编码，每个状态都由他独立的寄存器位，并且在任意时候只有一位有效，它是分类变量作为二进制向量的表示。这首先要求将分类值映射到整数值。然后，每个整数值被表示为二进制向量，除了整数的索引之外，它都是零值，它被标记为1。

   独热编码作用有：将离散特征的取值扩展到了欧式空间，离散特征的某个取值就对应欧式空间的某个点，让特征之间的距离计算更加合理。我们常用的距离或相似度的计算都是在欧式空间的相似度计算，计算余弦相似性，基于的是欧式空间，而在回归，分类，聚类等机器学习算法中，特征之间距离的计算或相似度的计算是非常重要的。

   实验证明这样的类别特征，对cross validation loss有不小的降低，对比而言，在我们试过的模型中，将Unix时间直接放入没有什么效果。

   此外，我们自己设计了以下交叉项
4) BidTotal = BidPrice1 * BidVolume1 反映了买一总金额数。
5) AskTotal = AskPrice1 * AskVolume1 反映了卖一总金额数。

Basic Set	Description(i = *level index*, $n = 10$)
$v_1 = \{P_i^{ask},\ V_i^{ask},\ P_i^{bid},\ V_i^{bid}\}_{i=1}^{n}$,	*price and volume (n levels)*

Time-insensitive Set	Description(i = *level index*)
$v_2 = \{(P_i^{ask} - P_i^{bid}), (P_i^{ask} + P_i^{bid})/2\}_{i=1}^{n}$,	*bid-ask spreads and mid-prices*
$v_3 = \{P_n^{ask} - P_1^{ask}, P_1^{bid} - P_n^{bid}, \lvert P_{i+1}^{ask} - P_i^{ask}\rvert, \lvert P_{i+1}^{bid} - P_i^{bid}\rvert\}_{i=1}^{n}$,	*price differences*
$v_4 = \{\frac{1}{n}\sum_{i=1}^{n} P_i^{ask},\ \frac{1}{n}\sum_{i=1}^{n} P_i^{bid},\ \frac{1}{n}\sum_{i=1}^{n} V_i^{ask},\ \frac{1}{n}\sum_{i=1}^{n} V_i^{bid}\}$,	*mean prices and volumes*
$v_5 = \{\sum_{i=1}^{n}(P_i^{ask} - P_i^{bid}),\ \sum_{i=1}^{n}(V_i^{ask} - V_i^{bid})\}$,	*accumulated differences*

Time-sensitive Set	Description(i = *level index*)
$v_6 = \{dP_i^{ask}/dt,\ dP_i^{bid}/dt,\ dV_i^{ask}/dt,\ dV_i^{bid}/dt\}_{i=1}^{n}$,	*price and volume derivatives*
$v_7 = \{\lambda_{\Delta t}^{la},\ \lambda_{\Delta t}^{lb},\ \lambda_{\Delta t}^{ma},\ \lambda_{\Delta t}^{mb},\ \lambda_{\Delta t}^{ca},\ \lambda_{\Delta t}^{cb}\}$	*average intensity of each type*
$v_8 = \{\mathbf{1}_{\{\lambda_{\Delta t}^{la} > \lambda_{\Delta T}^{la}\}},\ \mathbf{1}_{\{\lambda_{\Delta t}^{lb} > \lambda_{\Delta T}^{lb}\}},\ \mathbf{1}_{\{\lambda_{\Delta t}^{ma} > \lambda_{\Delta T}^{ma}\}},\ \mathbf{1}_{\{\lambda_{\Delta t}^{mb} > \lambda_{\Delta T}^{mb}\}}\}$,	*relative intensity indicators*
$v_9 = \{d\lambda^{ma}/dt,\ d\lambda^{lb}/dt,\ d\lambda^{mb}/dt,\ d\lambda^{la}/dt\}$,	*accelerations(market/limit)*

Fig. 4. [8]中的特征表。

6) BidAskDif = BidPrice1 - AskPrice1 反映了买一卖一价差
7) VolDif = BidVolume1 - AskVolume1 反映了买一卖一数目差
8) TotalDif = BidTotal - AskTotal 反映了买一卖一总金额之差

接下来的特征主要参考了 [7]，这篇文章中特征工程部分提到都是一些结合金融领域知识的特征，其中我们采用的有:

9) RelaPriceDif = 2 * (BidPrice1-AskPrice1) / (BidPrice1+AskPrice1) 反映了买一卖一相对价差
10) Depth = (BidVolume1+AskVolume1) / 2 反映了当前的买卖深度。
11) K = BidAskDif / Depth 反映了价格变化的斜率

接下来的特征主要参考了 [6]，这篇文章主要的特征工程部分主要针对的就是价格相关变量的在训练和实际测试中分布不同的问题，其中我们采用的有:

12) RelBid = BidPrice1 / MidPrice - 1 表示相对买价
13) RelAsk = AskPrice1 / AskPrice - 1 表示相对卖价
14) RelMid = MidPrice(t)/MidePrice(t-1) - 1 中间价相对变化率

接下来的特征主要参考了 [8]，这篇文章的特征工程比较详尽，见Fig. 4, 但是由于本次任务给的特征比较少（买卖只给了一阶，数据总量也很少），因此我们采用的只有一阶数据的时间差分特征。由于我们是10个一组的数据，因此求差分时，我只求后9个的差分特征，这样就避免了第一个时间点的差分特征缺失（跨天跨中午的第一个）的麻烦。

15) 具体而言，我们对除了MidPrice之外的其他feature，都求了差分。

此外，为了进一步解决价格分布的Gap，我们采用了：对于每一组数据，求出和价格有关变量的均值(BidPrice1，AskPrice1，LastPrice，MidPrice)，然后将每个时间点的每个价格特征减去对应的均值。并且，我将我们预测的任务也转换为：预测后20个时间点MidPrice的均值相对前10个时间点MidPrice均值的变化量。其好处有：

- 原本的价格变量，在每个时间窗中的分布其实也存在不小的Gap，如Fig. 6上方的两幅图片。转换之后，分布是非常接近的正态分布，如Fig. 6中下方的两幅图片。而要求训练和测试数据服从相近的分布，是很多监督学习算法的基本假设，如果盲目的使用Z-score或Min-Max标准化，那么就难以应对价格这个性质特殊的变量，尤其是预测的目标变量也是价格，造成面对未出现过的价格时，模型发挥不稳定的情况（如果你的训练测试不同分布，那么你的结果很大程度受到运气的影响 [9]。）。
- 将任务变为预测相对前10个时间点MidPrice均值的变化量，可以有效提高模型的表现，因为要预测的目标变量变化范围有限且小，而不是：预测的目标分布在一个很大的范围内并且测试数据集的目标可能与训练数据集中的目标差距很大，这一点可以从Fig. 6与Fig. 5 可以看出。相比于其他小组分享的：预测相对于最后一个时间点的变化量，这样做让预测目标更稳定，因为是求相对于均值的变化，就减少了各种极端情况，可以理解为预测一种趋势，减小了复杂度。这一个trick，我们在第二次经验分享中，分享给了大家。

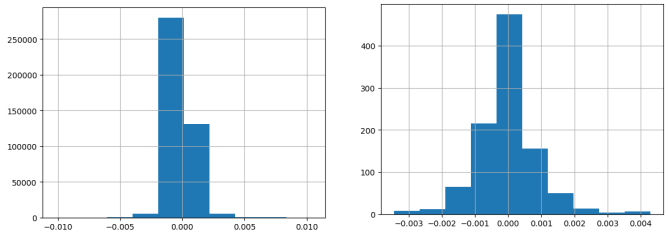

Fig. 5. 左-训练数据中target的分布直方图，右-训练数据中新target的分布直方图

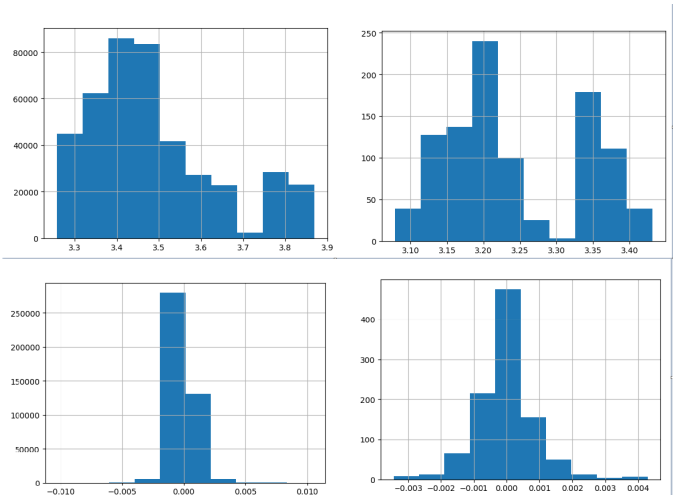

Fig. 6. MidPrice分布直方图，左上-训练数据，右上-测试数据。减去均值的MidPrice分布直方图，左下-训练数据，右下-测试数据

V. 特征选择

在上一节中，我们增添了大量的特征，但是这些特征并不全是有效的，并且针对不同模型，不同特征的效果也不同。

例如：对于我们开始尝试的神经网络，理论上是不需要特征组合的（省时省力，巨大优势！），也就是说，上

述对特征的组合，反而会影响神经网络的表现，因此我们只使用做差的trick来保证训练测试同分布。除此之外，由于神经网络的训练采用的是梯度下降，因此各个特征的scale相近有利于神经网络的快速收敛，原理如图7。原始特征空间中，由于各个特征的scale不同，在梯度下降的过程中，对于一个特征合适的learn rate，对于另一个特征可能太大或太小，导致梯度下降的过程很曲折，虽然两幅图中最终都达到了最优解，但在实际操作过程，用不标准化的数据来训练出一个收敛的神经网络难度极大。常见的标准化方法有两种Z-score (1) 和Min-Max (2)

$$x' = \frac{x - \mu}{\delta} \tag{1}$$

$$x' = \frac{x - x_{min}}{x_{max} - x_{min}} \tag{2}$$

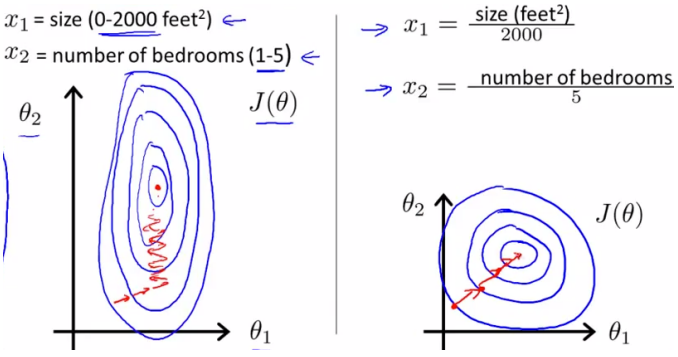

Fig. 7. 左-原始特征空间梯度下降的过程，右-Sclae后梯度下降的过程

对于我们最终使用的XGBoost而言，其作为一种树模型，对于数据的线性变化是没有意义的，但是对于数据交叉组合，是我们需要重点做的工作，这一点已经在上一节完成。

我们虽然生成了大量特征，而XGBoost理论上有自动特征选择的功能，但是太多的特征也会导致我们单次训练变慢很多，从而导致后续的工作效率很低，因此我们需要做一些特征选择，此外，减少特征数量，在一定程度上可以使模型泛化能力更强，减少过拟合。

常见的特征选择方法有:

1) 过滤法，按照发散性或者相关性对各个特征进行评分，设定阈值或者待选择阈值的个数，选择特征。
   例如：移除低方差的特征，单变量特征选择(皮尔森相关系数、最大信息系数等)
2) 包装法，根据目标函数（通常是预测效果评分），每次选择若干特征，或者排除若干特征。
   例如：递归特征消除
3) 嵌入法，先使用某些机器学习的算法和模型进行训练，得到各个特征的权值系数，根据系数从大到小选择特征。类似于Filter方法，但是是通过训练来确定特征的优劣。
   例如:基于L1的特征选择, 随机稀疏模型, 基于树的特征选择等

经过反复实验比较效果，加上XGBoost本身对特征的评分的功能就比较强大，我最终选择了XGBoost给出的feature importance作为特征的评分，来进行后续的操作。

计算好的各个特征的评分见Github中的feature importance文件（注：30个数据点一组，t-20表示第十个时间点，t-29表示第一个时间点）。

从中可以看出:

- 第十个时间点的特征最重要，越靠前的越相对不重要
- 组合特征中，相对价差、总金额差等也发挥了比较重要的作用

VI. 最终模型

这里我们主要讲一下我们最后提交的模型XGBoost+Linear Regression+ Stacking

A. *XGBoost*模型

XGBoost是boosting算法的其中一种。Boosting算法的思想是将许多弱分类器集成在一起形成一个强分类器。因为XGBoost是一种提升树模型，所以它是将许多树模型集成在一起，形成一个很强的分类器。而所用到的树模型则是CART回归树模型。

首先，介绍下CART回归树模型 [10]，假设树为二叉树，通过不断将特征进行分裂。比如当前树结点是基于第j个特征值进行分裂的，设该特征值小于s的样本划分为左子树，大于s的样本划分为右子树。

而CART回归树实质上就是在该特征维度对样本空间进行划分，而这种空间划分的优化是一种NP难问题，因此，在决策树模型中是使用启发式方法解决。我们用到的CART回归树产生的目标函数为:

$$\sum_{x_i}(y_i - f(x_i))^2$$

当我们为了求解最优的切分特征j和最优的切分点s，就转化为求解这么一个目标函数(3):

$$\min_{j,s}[\min_{c_1}\sum_{x_i \in R_1(j,s)}(y_i - c_1)^2 + \min_{c_2}\sum_{x_i \in R_2(j,s)}(y_i - c_2)^2] \tag{3}$$

所以我们只要遍历所有特征的的所有切分点，就能找到最优的切分特征和切分点。最终得到一棵回归树。

了解了CART回归树以后，下面简单介绍下XGBoost的思想，细节见 [11]。

XGBoost的核心想法就是不断地添加树，不断地进行特征分裂来生长一棵树，每次添加一个树，其实是学习一个新函数，去拟合上次预测的残差。当我们训练完成得到k棵树，我们要预测一个样本的分数，其实就是根据这个样本的特征，在每棵树中会落到对应的一个叶子节点，每个叶子节点就对应一个分数，最后只需要将每棵树对应的分数加起来就是该样本的预测值。

如Fig. 8中的例子训练出了2棵决策树，小孩的预测分数就是两棵树中小孩所落到的结点的分数相加。爷爷的预测分数同理。

作为今年来在Kaggle比赛中表现良好的模型（很多冠军都用到了XGBoost [12] [13]），XGBoost的主要优点有:

- 它是一种高度可扩展的end-to-end tree boosting系统，非常便于使用。
- 它具有高效的预计算算法以及新的稀疏感知算法，用于并行化tree learning，计算速度非常快

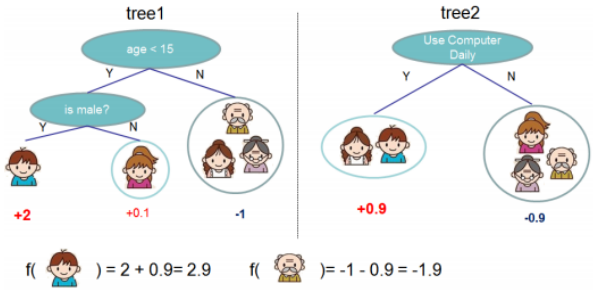

Fig. 8. XGBoost原理示意图

- 它具有高效的内存感知块结构，用于核外tree learning，节约内存与时间。

B. XGBoost参数调整

XGBoost有上百个参数 [14]，由于算力与时间的限制，我只挑选了9个参数进行调整。

调参的过程中，我使用的是sklearn中的GridSearchCV函数，但是仍然不能直接就把9个参数直接GridSearch，因为假设一个参数3个值，5折交叉检验，那么需要拟合模型$3^9 * 5 = 98415$次！这时候也体现出前面特征选择的必要性，实践表明减小特征数目，对XGBoost拟合一次的速度提升是非常明显的。

总之，我们需要按一定的顺序来调整参数，虽然这样调参得到不一定是最优参数组合（因为加了一个类似马尔科夫假设的东西），但却让运行时间变得可以接受，我们调参的流程如下:

1) 调整最佳迭代次数：n_estimators（暂时固定学习率:learning_rate为0.1）。第一个参数决定了算法跑的轮数，也可以理解成生成的决策树的数目，太小的话模型未收敛，太大的话单次运行时间过长。第二个参数决定了梯度下降的幅度。把这两个参数放在首先调整，就是为了确保模型大体上是收敛的，这样比较其他参数的效果才有意义。
2) 调整max_depth及min_child_weight。第一个参数决定了树的最大深度，这个值是用来避免过拟合的，值越大，模型会学到更具体更局部的样本。第二个参数决定最小叶子节点样本权重和，该数也用于避免过拟合。当它的值较大时，可以避免模型学习到局部的特殊样本。但是如果这个值过高，会导致欠拟合。这两个参数的值是对XGBoost模型结构影响比较大的参数，因此放在第二个调整，有一种先粗调再微调的思想。
3) 调整gamma。其指定了节点分裂所需的最小损失函数下降值。这个参数的值越大，算法越保守。这个参数的值和损失函数关联很大。因此单独调整。
4) 调整subsample及colsample_bytree。第一个参数控制对于每棵树，随机采样的比例。减小这个参数的值，算法会更加保守，避免过拟合。但是，如果这个值设置得过小，它可能会导致欠拟合。第二个参数用来控制每棵随机采样的列数的占比，类似于第一个参数。由于这两个参数效果相近，而且对模型的影响不是特别巨大，因此一起调整。
5) 调整reg_alpha及reg_lambda。二者分别代表权重的L1及L2正则化项的大小。因为都是正则化项，所以放在最后调整。
6) 最后，再精调最佳迭代次数：n_estimators和学习率:learning_rate，确保模型收敛到一个较优解。

C. XGBoost+Linear Regression

XGBoost+LR模型融合方法用于分类或者回归的思想最早由Facebook在广告ctr预测中提出 [15]。在这篇论文中他们提出了一种将XGBoost作为feature transform的方法。大概的思想可以描述为如下：先用已有特征训练XGBoost模型，然后利用XGBoost模型学习到的树来构造新特征，最后把这些新特征加入原有特征一起训练模型。构造的新特征向量是取值0/1的，向量的每个元素对应于XGBoost模型中树的叶子结点。当一个样本点通过某棵树最终落在这棵树的一个叶子结点上，那么在新特征向量中这个叶子结点对应的元素值为1，而这棵树的其他叶子结点对应的元素值为0。新特征向量的长度等于XGBoost模型里所有树包含的叶子结点数之和。最后将新的特征扔到LR模型进行训练。实验结果表明XGBoost+LR能取得比单独使用两个模型都好的效果。其原理如Fig. 9

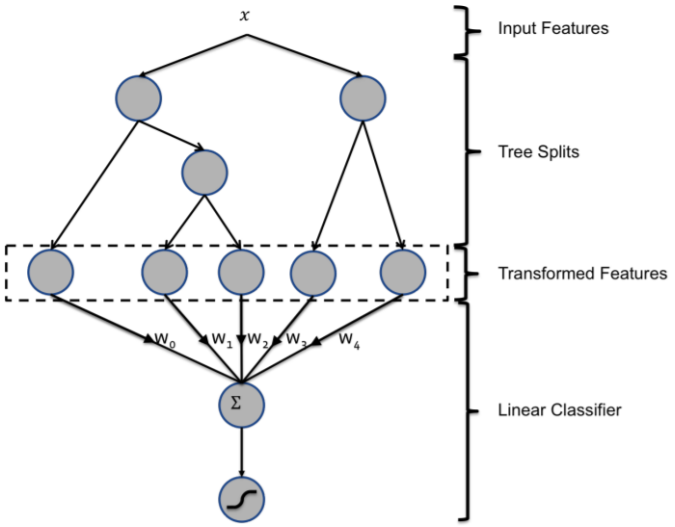

Fig. 9. XGBoost+LR的原理图 [15]

在原论文中的LR指Logistic Regression来解决分类问题，在工业和竞赛实践中，对于回归问题，LR就指代Linear Regression，在使用Xgboost+LR(我们将新特征和原特征一起放入Linear Regression模型)之后,我们的模型准确度有了很大的提升。

D. Stacking

最后一步，我们使用了集成学习。集成学习是指将多个不同的Base Model组合成一个Ensemble Model的方法。它可以同时降低最终模型的Bias和Variance [16]，从而在降低Loss的同时又降低Overfitting的风险，因此也成为Kaggle比赛中很多获奖选手用到的方法 [17]。

集成学习对基模型有两个要求:

- Base Model之间的相关性要尽可能的小。Ensemble的Diversity越大，最终Model的Bias就越低。
- Base Model之间的性能表现不能差距太大。这其实是一个Trade-off，在实际中很有可能表现相近的Model只有寥寥几个而且它们之间相关性还不低。但是实践告诉我们使在这种情况下Ensemble还是能大幅提高成绩。

我们最终使用的集成学习方法是Stacking [18]，其原理图如Fig. 10，其流程为:

1) 将数据集划为k部分，共有n个模型。我们设置为k=5折，模型我们设置为36个XGBoost+LR，每一个模型的超参数都加了随机扰动+每一个模型得到数据的特征为评分top m（预设范围内的随机数）的特征,这样设置XGBoost模型的原理及具体步骤参考 [19]。
2) 接下来k次迭代，每次迭代时，将k-1份数据作为Training Set对每个Base Model进行训练，然后在剩下一份Hold-out Set上进行预测。同时也要将其在测试数据上的预测保存下来。
3) 这样,每个Base Model在每次迭代时会对训练数据的其中1份做出预测，对测试数据的全部做出预测。k个迭代都完成以后我们就获得了一个大小为:训练数据行数xBase Model数量的矩阵，这个矩阵接下来就作为第二层的Model 的训练数据。
4) 当第二层的Model训练完以后，将之前保存的Base Model对测试数据的预测（因为每个Base Model被训练了k次，对测试数据的全体做了k次预测，所以对这k次求一个平均值，从而得到一个形状与第二层训练数据相同的矩阵）拿出来让它进行预测，就得到最后的输出。

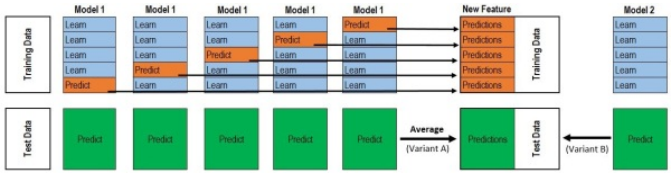

Fig. 10. Stacking原理图

值得一提的是，在开始做Stacking以后，我的cross validation loss提高不大，Leader Board成绩的提高明显，可见Stacking有效增强了模型的泛化能力（最后能够在Privete Board取得比Leader Board上更高的成绩）。

最后，我们提交的就是两次XGBoost+Linear Regression+Stacking的结果，Leader Board上的成绩为0.00132和0.00131，Private Board上的成绩为0.00125（Lucky Gap）。

VII. 其他模型

其实我们组开始时候尝试了大量的神经网络结构LSTM [20], GRU [21], CNN+LSTM [6]，TreNet [22]。

但是使用结构简单的网络遇到性能瓶颈，使用复杂的网络感觉难以训练。因此我们改用了XGBoost模型，下面是我们的一些思考：

- 树模型训练更快速，训练好一个最简单的LSTM也要耗费大量的时间，随着模型结构复杂度的上升，神经网络变得越来越难以训练，考虑到有限的时间，我们决定尝试树模型
- 神经网络相比树模型的巨大优势在于捕捉结构信息（图片、音频、文本），而本次的预测任务比较简单，用树模型一样可以取得优异的表现(No free lunch)。
- 神经网络在学术界的火热，确实是由于在经过精心的结构设计后，其能够从海量的数据中学习而不会像树模型一样早早遭遇学习能力瓶颈，而在比赛中，我们既没有海量数据（40万组，9维原始特征），也没有充足的时间、算力与精力去完成一个专门针对该任务的神经网络设计，因此树模型完全有可能取得比使用一些泛用古老的神经网络结构（LSTM-1997!）更好的结果
- 神经网络由于不需要数据组合的特征工程，我们的绝大数精力只能放在调整网络结构和调参的工作上，在这个过程中我们尝试了一星期，收效甚微（可能因为缺少调节神经网络的经验），而对于树模型，有比较多的工作可以去提升其表现，我们决定转向树模型。

综合上述原因，在最初调试一周的神经网络之后，我们尝试XGBoost，发现有更好的表现以及其他可提升的工作，因此就将精力主要转向了XGBoost。

下面是我们尝试过的神经网络，都没有超过XGBoost的表现。

A. LSTM

Long short term memory(LSTM)是一种RNN神经网络结构，由 [20]提出，为了解决RNN难以学习长的依赖关系的问题而提出来的神经网络结构，其示意图如Fig. 11。其核心在于使用门机制以及隐状态，来达到更好的捕捉长信息的同时避免梯度消失的效果，成为捕捉序列信息的一个业界广泛使用的模型。

本次任务中，我们就可以使用一个序列长度为10的LSTM来捕捉序列信息，可以通过Stacking LSTM来捕捉高阶时序信息提高模型表现 [23]，最后过线性层即可得到回归目标。

我们在使用过程LSTM中，在Public Board上的最好成绩为0.00154。

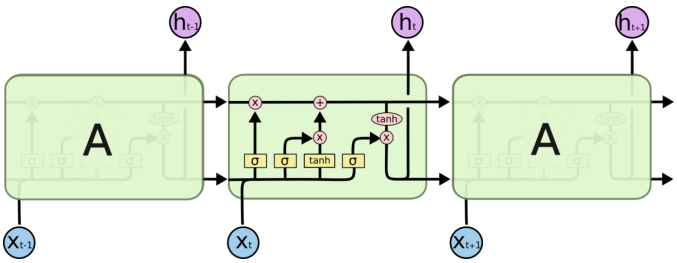

Fig. 11. LSTM示意图

B. GRU

Gated Recurrent Unit (GRU)是由 [21]提出的LSTM的变体，它在保持了LSTM的效果的同时简化了结构，从而减少了计算量，使得模型的训练变得容易。其与LSTM的不同见Fig. 12，可以看出结构简化了很多。

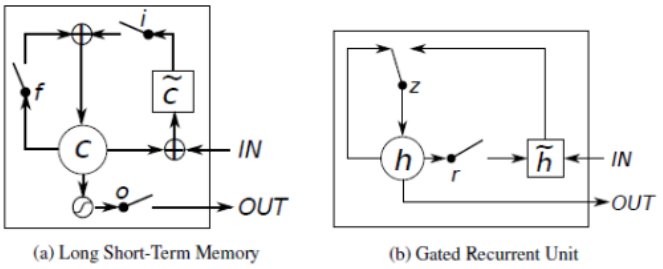

Fig. 12. LSTM与GRU对比图 [24]

在比赛中，GRU的收敛确实要快一点，但是模型表现上相对LSTM并没有提高，因此我们很快放弃了尝试GRU，因为这时候LSTM的计算复杂度并不是瓶颈。

C. CNN+LSTM

该架构的思想是是开始时使用CNN来捕捉序列特征，然后将提取出来的高阶序列特征放入LSTM进行训练。

Convolutional Neural Network (CNN) 开始时用来提取图像的特征的一种神经网络结构，后来被迁移到捕捉序列数据的特征上来，它的主要好处包括:局部感知（捕捉高阶特征），参数共享（减少计算量），多核（捕捉不同的特征）等，对于序列数据，一般都采用1d卷积，如Fig. 13是NLP领域的1d卷积示意图。

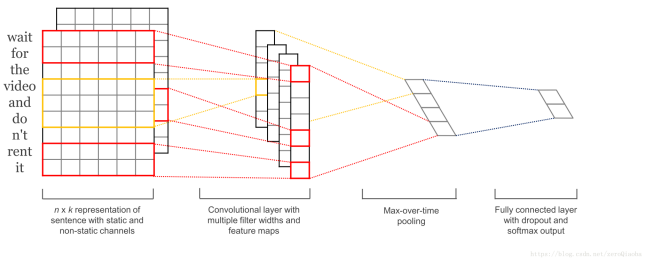

Fig. 13. 1d卷积示意图

其主要特点在于卷积核的长与输入特征等长，宽决定了捕捉几个时间步之间的互信息。

我们搭建模型时主要参考了 [6]的架构，如Fig. 14所示，先Stack卷积层提取高阶时序信息，然后用LSTM来处理这些高阶时序信息。

在比赛中，当CNN层数较少时，效果有所提升(Public Board 0.00148)，当我们将CNN层数Stack的比较深时，遭遇了梯度消失的问题，模型的训练loss降不下去，因为时间有限，我们放弃了这个结构，后来和同学交流，了解到可以考虑使用ResNet的思想来尝试解决这个问题。

D. TreNet

这个模型是我们查时间序列预测模型中，比较新的一篇论文 [22](IJCAI 2017)。它提出了一种混合神经网络的结构，同时用RNN与CNN提取序列特征，然后将输出拼接作为全连接层的输入，最后输出最终的预测结果，其结构如Fig. 15。

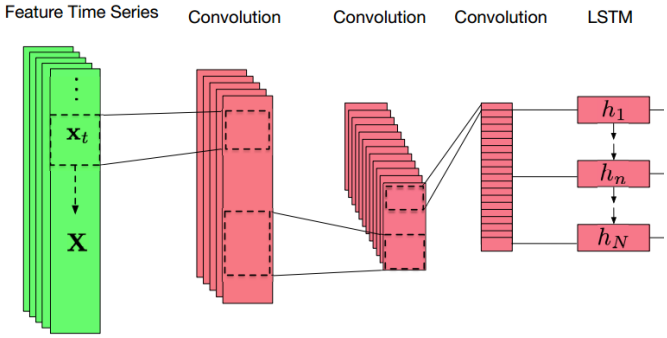

Fig. 14. CNN+LSTM示意图

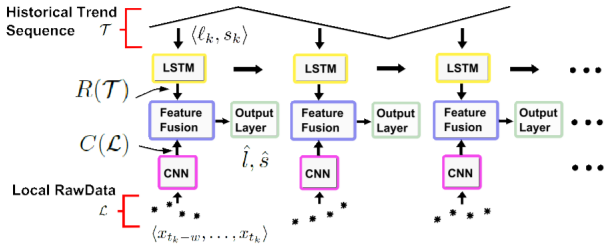

Fig. 15. TreNet示意图 [22]

在比赛中，由于这个模型针对的任务是实数时间序列，每一个时间步只是一个实数值，因此迁移到比赛任务中后，没有充分利用其他信息，得到的结果不太理想，也就放弃了。

VIII. 其他技巧

在这次Kaggle上比赛的过程中，我们收获了不少机器学习的实战经验，下面分享一下：

1) Leader Board (LB)上的loss和Local Cross Validation (CV)即本地测试集上的loss，我们训练时希望的是降低CV，而提交上去希望降低LB，当CV降了，LB没降时，我们就需要仔细分析：是什么导致了CV数据和LB数据存在Gap，比如本次比赛中价格特征的Gap。分析清楚Gap之后，就要想办法减小Gap，一个是像我们本次比赛中用到的对CV和LB数据作相同处理；还可以考虑采样CV数据时，不采用完全随机，而是采样与LB数据相似的样本，这样也可以有效降低Gap。
2) 当然有些时候LB上的分数并不一定是准确，因为LB和Private Board上的数据可能也有Gap，我们在确保自己的本地数据不是特别有偏的情况下，应该相信CV，专注于降低CV（也就是提高自己模型的泛化能力），靠刷提交次数来刷LB分数的行为并不可取，因为这在某种程度上算是LB数据的信息泄漏到模型中，导致模型过拟合到LB数据上，反而有可能降低Private Board分数。
3) 比赛的重心应该放在数据处理以及模型结构上，盲目耗费大量精力调参并不可取。
4) 写好PipeLine，将模型的各个步骤解耦，使得不管是增加或删除特征，调节模型的参数，更换模型都可以

写很少的代码完成，如果比较长的流程都堆在一起，反复修改造成的bug会浪费不少精力。

5) 自动记录日志，比如记录模型的参数、训练和测试的loss变化、最终的结果，尽可能让每一个模型的每一次训练都有价值，不要让“调出一个好模型，但是忘记参数”这样浪费时间的事情发生。
6) 善用工具包，比如Tensor Board监测神经网络训练状况，GridSearch用于XGBoost调参，可以节省大量人工工作

REFERENCES

[1] A. Madhavan, M. Richardson, and M. Roomans, “Why do security prices change? a transaction-level analysis of nyse stocks,” *The Review of Financial Studies*, vol. 10, no. 4, pp. 1035–1064, 1997.

[2] R. D. Huang and H. R. Stoll, “The components of the bid-ask spread: A general approach,” *The Review of Financial Studies*, vol. 10, no. 4, pp. 995–1034, 1997.

[3] R. F. Engle and J. R. Russell, “Autoregressive conditional duration: a new model for irregularly spaced transaction data,” *Econometrica*, pp. 1127–1162, 1998.

[4] M. Kearns and Y. Nevmyvaka, “Machine learning for market microstructure and high frequency trading,” *High Frequency Trading: New Realities for Traders, Markets, and Regulators*, 2013.

[5] A. A. Adebiyi, A. O. Adewumi, and C. K. Ayo, “Comparison of arima and artificial neural networks models for stock price prediction,” *Journal of Applied Mathematics*, vol. 2014, 2014.

[6] A. Tsantekidis, N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, “Using deep learning for price prediction by exploiting stationary limit order book features,” *arXiv preprint arXiv:1810.09965*, 2018.

[7] “基于机器学习的订单簿高频交易策略,” https://www.fmz.com/bbs-topic/490, accessed April 4, 2010.

[8] A. N. Kercheval and Y. Zhang, “Modelling high-frequency limit order book dynamics with support vector machines,” *Quantitative Finance*, vol. 15, no. 8, pp. 1315–1329, 2015.

[9] A. Ng, “Machine learning yearning,” 2017.

[10] R. L. Lawrence and A. Wright, “Rule-based classification systems using classification and regression tree (cart) analysis,” *Photogrammetric engineering and remote sensing*, vol. 67, no. 10, pp. 1137–1142, 2001.

[11] T. Chen and C. Guestrin, “Xgboost: A scalable tree boosting system,” in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. ACM, 2016, pp. 785–794.

[12] “1st place solution of allstate claims severity,” https://www.kaggle.com/c/allstate-claims-severity/discussion/26416.

[13] “1st place solution of home credit default risk,” https://www.kaggle.com/c/home-credit-default-risk/discussion/64821.

[14] “Xgboost documentation,” https://xgboost.readthedocs.io/en/latest/index.html.

[15] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers *et al.*, “Practical lessons from predicting clicks on ads at facebook,” in *Proceedings of the Eighth International Workshop on Data Mining for Online Advertising*. ACM, 2014, pp. 1–9.

[16] A. Chandra, H. Chen, and X. Yao, “Trade-off between diversity and accuracy in ensemble generation,” in *Multi-objective machine learning*. Springer, 2006, pp. 429–464.

[17] “1st place solution of otto group product classification challenge,” https://www.kaggle.com/c/otto-group-product-classification-challenge/discussion/14335.

[18] D. H. Wolpert, “Stacked generalization,” *Neural networks*, vol. 5, no. 2, pp. 241–259, 1992.

[19] “用户人品预测大赛-冠军分享,” http://www.dcjingsai.com/common/bbs/topicDetails.html?tid=348.

[20] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[21] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[22] T. Lin, T. Guo, and K. Aberer, “Hybrid neural networks for learning the trend in time series,” in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, 2017, pp. 2273–2279.

[23] A. Graves, A.-r. Mohamed, and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 2013, pp. 6645–6649.

[24] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.