长江证券 CHANGJIANG SECURITIES

机器学习

机器学习白皮书系列之四：机器学习流程和算法介绍及金融领域应用实例

2018-2-7

金融工程 | 专题报告

报告要点

■ 机器学习问题和其流程

机器学习问题本质上在于找出使得经验风险泛函（样本误差）最小的建模流程，基本的流程可以分为特征工程、模型训练和模型融合。本篇就上述三个过程，给出相关算法的介绍，并补充了之前系列报告中未详细介绍的内容。

■ 机器学习三大步骤

特征工程包含特征构建、特征提取和特征选择三个过程，以选择相对最优的特征空间。特征工程往往会采用无监督和有监督的机器学习算法。机器学习模型可以分为线性模型、树模型和深度学习模型。线性模型主要体现了数据中的线性关系，如输入与输出的线性关系，点集的线性可分；树模型可以很好的捕捉输入与输出的非线性关系，和线性模型相辅相成。一些改进的随机梯度下降法可以很好地训练深度学习模型。模型融合有横向拼接和纵向拼接两种方式，在模型融合过程中，往往要求模型表现好，且之间的相关性小。深度学习可以将模型融合通过网络结构的设计在模型训练中完成。

■ 多因子选股案例

目前多因子选股模型多以单个模型在整个特征空间上的预测构建策略，往往很难保证函数空间上的一致性，而机器学习流程选股通过在大的函数空间中选择多个小的函数空间进行合并，得到更为完善的模型，可以在估计函数空间上更加逼近实际函数空间。本文以提升树模型、ExtraTrees 模型和深度神经网络模型进行横向拼接，加权平均模型输出的伪概率，构建投资组合。融合模型超额年化收益为 25.91%，夏普比为 1.06，信息比为 2.26，月度超额胜率为 0.76，在超额收益、夏普比、信息比及月度超额胜率上表现略优于单个模型，且在分组投资组合的区分上更为明显。

分析师 **覃川桃**
☎ （8621）61118766
✉ qinct@cjsc.com.cn
执业证书编号：S0490513030001

联系人 **郑起**
☎ （8621）61118706
✉ zhengqi2@cjsc.com

联系人 **杨靖凤**
☎ （8621）61118736
✉ yangjf@cjsc.com.cn

相关研究

《机器学习白皮书系列之三：深度学习的方法介绍及金融领域应用实例》2018-1-22

《红利策略：展望 2018》2018-1-14

《鹏华基金指数型产品的投资价值分析》2018-1-2

风险提示：
1. 模型在使用中存在建模风险；
2. 本文举例均是基于历史数据不保证其未来收益。

目录

机器学习流程介绍 4
机器学习问题 4
“方差”与“偏差” 4
机器学习三大步骤 5
特征工程 5
特征构建 6
特征提取 6
主成分分析 6
独立成分分析 7
限制波尔兹曼机 7
特征选择 7
过滤式 8
封装式 8
嵌入式 8
深度学习的特征工程 9
模型训练 9
线性模型 9
广义线性回归模型 9
支持向量机 10
线性模型比较 12
树模型 13
决策树模型 13
集成学习算法 14
Bagging 与随机森林模型 14
Boosting 与提升树模型 15
树模型比较 18
深度学习模型 18
最优化算法 18
训练模型参数 20
交叉验证 21
模型融合 21

复杂的模型融合.....22
深度学习的模型融合.....23
多因子选股案例.....24
建模流程.....25
数据与特征工程.....25
模型与回测相关.....27
相关结果.....27
总结.....30

图表目录

图 1：“方差”与“偏差”制衡.....5
图 2：机器学习流程.....5
图 3：支持向量机运作示意图.....10
图 4：噪音对支持向量机的影响.....11
图 5：支持向量机惩罚项对拟合效果影响.....11
图 6：决策树预测流程.....13
图 7：Blending 流程图.....22
图 8：Stacking 流程图.....23
图 9：模型融合网络图.....24
图 10：融合模型分组表现.....28
图 11：各个模型第一组表现.....29
图 12：各个模型超额收益.....29
图 13：各个模型多空表现.....30

表 1：常用特征构建过程.....6
表 2：常用特征组合过程.....6
表 3：特征选择方法比较.....8
表 4：线性模型比较.....12
表 5：树模型比较.....18
表 6：简单的模型融合.....22
表 7：因子列表.....26
表 8：模型 f1score/模型融合权重.....27
表 9：融合模型分组表现.....28
表 10：融合模型第一组分年表现.....28
表 11：模型分组指标比较.....29
表 12：模型风险评价指标.....30

机器学习流程介绍

在之前的报告中，我们介绍了一些常用的传统机器学习算法和深度学习算法，本文将对机器学习在特定场景中应用的建模流程，给出相关介绍。

机器学习问题

机器学习问题可以抽象为一个寻找当前数据集分布的问题，具体来看，给定训练样本$D = \{(x_1, y_1), (x_2, y_2), .., (x_n, y_n)\}$（训练样本是根据联合分布$F(x, y) = F(x)F(y|x)$抽取的$n$个独立同分布的观测），机器学习问题就是从给定的函数空间$f(x, \lambda), \lambda \in \Omega$中找出最优的逼近函数，以达到预测$y$的目的，其中$\Omega$为参数空间。

对于一个函数逼近问题，一般会使用损失函数$L(y, f(x, \lambda))$来衡量在寻找逼近函数中存在的误差，其期望也被称作风险泛函：

$$R(\lambda) = \int L\big(y, f(x, \lambda)\big)\, dF(x, y), \lambda \in \Omega$$

函数逼近中最优化损失函数本质上是在已有的样本空间上，使得估计函数有最小化风险泛函。由于联合分布$F(x, y)$未知，风险泛函并不可求，但是可以利用算数平均来近似代替风险泛函，从而得到了经验风险泛函，也即样本误差：

$$R_{exp}(D, \lambda) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i, \lambda))$$

其中D为样本空间。所以机器学习问题就变成了如何在当前样本下得到最优的样本误差。由上式可以看出，样本误差是一个关于样本和参数的函数，在样本一定的情况下，函数就变成了一个参数空间到样本误差空间的函数，这个参数空间可以是狭义上的机器学习模型的超参数，广义上来说，是建模流程中的每个步骤，如在特征工程中采用的特征选择算法、降维算法，特征子空间，模型训练过程中选择的模型簇，模型的超参数，模型融合的方式等等。所以**机器学习本质上是找出使得经验风险泛函最小的建模流程。**

“方差”与“偏差”

机器学习中的样本误差，以平方误差为例，可以对其进行分解：

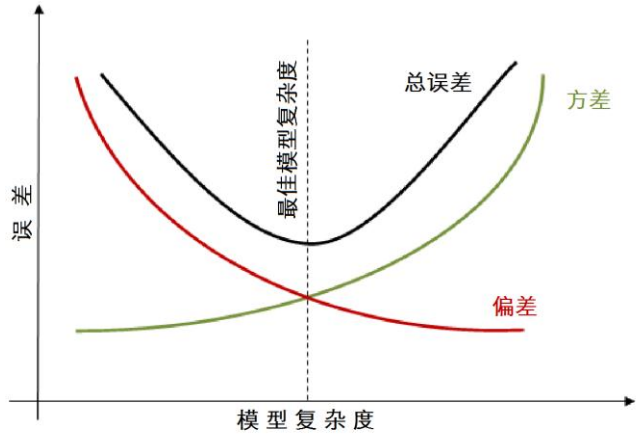
$$Err(\boldsymbol{x}) = E_D\left[\left(f(\boldsymbol{x}; D) - \bar{f}(\boldsymbol{x})\right)^2\right] + \left(\bar{f}(\boldsymbol{x}) - y\right)^2 + E_D[(y_D - y)^2]$$

$$= variance + bias^2 + noise$$

其中$f(\boldsymbol{x}; D)$是训练集D下得到的模型，$\bar{f}(\boldsymbol{x})$是模型的期望预测，y是真实值，y_D为观测值。

直观上来说，“方差”反映的是模型每次输出结果和模型输出期望之间的误差，即模型在不同数据集上表现的稳定性，一般模型复杂度越低“方差”越小；“偏差”反映的是模型在样本上的输出与真实值之间的误差，即所用模型和真实模型的误差，一般模型复杂度越高“偏差”越小。为了使得我们建立的模型在做预测的时候误差尽可能小，我们需要减小“方差”和“偏差”。但是两者之间存在一个制衡，一般情况下不能同时减小，如下图：

图 1：“方差”与“偏差”制衡



资料来源：长江证券研究所

如果训练不足，学习器未能很好的逼近真实情况，“偏差”较大，同时因为学习器能力不强，导致数据集的扰动无法使学习器表现产生变化，“方差”较小，即**欠拟合情况**。

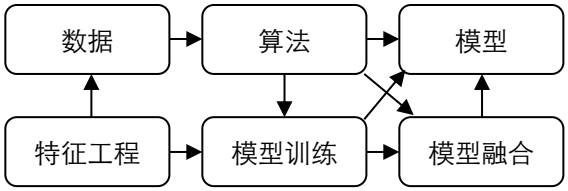
如果充分训练，学习器可以很好的逼近当前情况，“偏差”较小，但是数据集的轻微扰动都会对学习器表现造成较大影响，“方差”较大，即**过拟合情况**。

机器学习中的很多流程都是为了寻找使得“偏差”和“方差”相对平衡的点，保证整体的误差最小。

机器学习三大步骤

一般来说，机器学习有三个要素：数据、算法和模型。数据是场景的描述，包括输入和输出。算法是得到模型的过程，狭义上说，特指机器学习算法，如传统线性回归、树和支持向量机以及深度学习；广义上说，从最初得到数据到最终确定模型中间的所有过程，即建模流程都可以看作算法，如分类、回归模型，搜索最优参数算法。模型是输入到输出的映射，即我们最后需要得到的特定法则，对场景给出相关预测。对应这三个要素，机器学习又可以分为三个步骤：特征工程、模型训练和模型融合，如下图所示：

图 2：机器学习流程



资料来源：长江证券研究所

通过特征工程，确定需要预测的**数据**，如回归值或分类标签，以及体现分析问题的特征；经过特征处理的数据使用特定**算法**，得到**部分模型**；得到的部分模型使用特定**算法**进行融合，得到**整体模型**。

特征工程

在很长一段时间内，特征工程都是机器学习的核心部分。机器学习中有一句非常有名的话“garbage in, garbage out”，数据和特征往往决定了机器学习的上限，而模型和算法

只是逼近这个上限，所以很多数据科学家在建模过程中，对特征的处理可能会花费整个建模 80%的时间，直到深度学习的兴起，从某种意义上很大程度的解放了数据科学家在这一过程中的工作量，但是特征工程仍然重要。

特征工程是利用数据领域的相关知识来创建能够使机器学习算法达到最佳性能的特征的过程。简单来说，特征工程就是人为的设计可以输入的变量，如多因子模型中的因子。

特征工程包含三个过程——特征构建、特征提取、特征选择，三个过程不是独立存在的，结果往往需要不断迭代，和模型训练、模型融合过程结合，得到相对最优的特征。

特征构建

特征构建就是把已有的数据转换为特征，这种特征可以直接作为输入训练模型，下表展示了一些常用的特征构建过程：

表 1：常用特征构建过程

数据类型	转化	算法	应用场景举例
文本数据	向量	分词、word2vec	情感分析中的文本处理
图像数据	矩阵	模糊、卷积、池化	图像识别中的图像处理
日期数据	整数	年月日的提取	周期性分析
地理位置	距离	欧氏距离的转化、坐标的表示	房价预测中的地理位置表示
类别变量	向量	哑变量处理	多因子选股中的行业表示
噪音数据	向量	去噪处理	时间序列中的小波变换

资料来源：长江证券研究所

特征提取

特征提取很是像特征构建的“升级”，通过“低阶”特征之间的组合、变换，从中找出有效的、可以体现问题分析特点的“高阶”特征。一般来说，会对截面上的特征进行加法、减法、乘法、除法、平方、立方、取对数等处理，也会对有时间序列性质的特征进行取平均值、最大值处理，这里列举一些多因子模型中常用的组合特征及逻辑：

表 2：常用特征组合过程

因子名称	因子描述	算法	逻辑
EP	净利润/总市值	除法	公司估值
Capital	价格×总股本	乘法	公司规模
Ln_price	价格对数	取对数	平滑价格影响
MA	价格移动平均	取平均数	光滑处理价格
Std_Nm	波动率	取标准差	体现收益率波动
Beta	与上证综指回归的	回归系数	个股的 Beta

资料来源：长江证券研究所

特征提取也会采用一些无监督的方法，以达到降低特征维度，提取主要特征的作用，主要的算法有：主成分分析、独立成分分析、限制波尔兹曼机。

主成分分析

主成分分析（Principal Component Analysis，PCA）可以对高维数据进行降维，去除数据噪声，保留数据中的主要模式。PCA 以最大化样本方差为标准，对原始特征进行线性

组合得到新的特征，并通过提取样本方差贡献较大的组合特征，达到降维效果。对于一个n维样本数据，若需要将维度降到k维，首先得到其协方差矩阵及其对应的特征值、特征向量：

$$C = \begin{pmatrix} cov(x_1, x_1) & \cdots & cov(x_1, x_n) \\ \vdots & \ddots & \vdots \\ cov(x_n, x_1) & \cdots & cov(x_n, x_n) \end{pmatrix}$$

$$CX = \lambda X$$

将特征向量对应特征值大小从上到下排列成矩阵，取前k行组成矩阵P，得到新的特征：

$$Y = PX$$

独立成分分析

独立成分分析最经典的应用之一为盲源分析，将混合的信号分离成潜在的信息成分。独立成分分析为了保证分解的解，一般要求原数据为非正态分布，但是一些分析方法可以通过某种“非高斯性”的度量最大化实现分解，如比较常用的四阶矩，通过迭代的方式，找出在每个样本空间内四阶矩最大的分解权重，得到分解矩阵。

独立成分分析很多时候更像是主成分分析的一个补充，主成分分析在数据为正态分布的假设下有着很好的效果，当数据不服从正态分布时，独立成分分析就可以作为另一种降维手段对特征进行提取。

限制波尔兹曼机

限制波尔兹曼机由两层网络结构组成：输入层和隐含层，其中隐含层可以像深度神经网络一样，由多个隐含层组成。限制波尔兹曼机可以分为两个过程，第一个过程为正向传播过程，通过现有数据的输入，得到经过隐含层处理的值；第二个阶段为反向传播过程，也可以称为重构阶段，隐含层的激活值成为反向传递中的输入，这些输入值与之前正向传播用同样的权重相乘，得到的结果再与每个可见层的偏差相加，就得到了重构值，也即原始输入的近似值。通过不断的修正偏差值，来得到最终稳定的模型。

在正向传递的过程中，限制波尔兹曼机用输入值来预测节点的激活值，输出的是一个概率，即$p(a|x;w)$；在反向传递时，激活值成为输入，输出对于原始数据的重构值，即概率$p(x|a;w)$；通过上述两种预测值的结合，就可以得到输入x和激活a的联合概率分布，即$p(x,a)$。最终限制波尔兹曼机期望达到的结果是估计原始数据的近似值，即输入数据的一种重构。

特征选择

特征选择的目的是寻找最优特征子空间，剔除不相关或冗余特征，从而达到减少特征个数，提高模型精度的目的。特征选择一般分为三个过程：

1. 特征子空间的确定：一般将特征构建和提取得到的集合作为特征全集，不断从全集中寻找子集，作为当前特征选择的搜索范围；
2. 选择算法：选择算法一般可分为两个部分。一是评价函数，通过评价函数来评价当前特征子空间的表现，二是筛选准则，当评价函数达到某个阈值后就得到了相对满意的特征集合；
3. 验证过程：在验证集上评价特征子空间的有效性。

特征选择算法中的相关的算法又可以分为三个大类。

过滤式

过滤式特征选择的评价标准从数据集本身的内在性质获得，与特定的学习算法无关，因此具有较好的普适性，也是目前特征选择过程中都会做的一步。通常会选择和目标之间相关性高或信息增益大的变量加入到之后的模型中。

过滤式算法的优点是显而易见的，即适用性强、算法复杂度低，可以高效的筛选掉大量不相关的特征，缺点是由于算法的简单导致选择的特征并非最优。针对分类和回归两个不同问题，有不同的评价函数，分类一般采用卡方检验、f 值和信息熵，回归一般采用皮尔森相关系数、f 值。

实际上，正是由于过滤式算法的高速度低精确性，往往只将其结果作为特征筛选的辅助手段，一般会采用可视化形式展示每个特征和目标之间的相关性，对集合内的特征有一个大概的了解，而不将其结果作为最终的选择。

封装式

封装式方法通过不断搜索特征子空间上模型的表现，确定最优的特征子空间，常用的算法有向前搜索法、向后搜索法。封装式方法在选择特征的过程中实际上已经用到了之后可能会用到的模型，这些模型可以是决策树、神经网络、贝叶斯分类器、近邻法以及支持向量机等等，即有监督的机器学习模型都可以作为封装式算法的基模型。

相比于过滤式方法，封装式方法找到的特征子空间性能通常更好，但是封装式方法选出的特征通用性不强，当基模型改变时，选择的特征往往也会产生变化，且由于每次评价特征子空间都要进行模型的训练，计算复杂度高，时间上不够高效。所以封装式方法往往只会采用一些简单的模型初步选择一遍特征，作为最后的特征子空间。

嵌入式

嵌入式特征选择中，特征选择算法本身作为模型训练的一部分嵌入到学习算法里，如最典型的 Lasso 回归，在回归的过程中，筛选出主要变量，降低特征空间的维数；再如树模型，根据剪枝或是一定的限制，选择出较小的特征空间。

但是由于嵌入式特征选择输入的特征信息表现较为普通，模型的表现往往一般，并不能直接使用，所以这一过程多用来有目标的选择特征空间。如利用 Lasso 回归筛选变量，供线性回归所用。

上述三种方法的相关比较如下表：

表 3：特征选择方法比较

方法	评价函数	优点	缺点	用途
筛选式	相关系数、信息熵、基尼指数、卡方检验	耗时少，适用性强	精确度低，倾向于冗余特征，未考虑变量之间的相互影响	快速找出输入和输出之
封装式	在子空间上的交叉验证模型表现	考虑到特征之间的相互影响	搜索算法耗时较长	筛选主要特征子空间
嵌入式	模型表现	较为准确	较为依赖模型	选择特定特征空间

资料来源：长江证券研究所

深度学习的特征工程

相比于传统机器学习模型，深度学习的一个巨大的优势在于有着较为有效的特征提取过程（具体讨论见《机器学习白皮书系列之三：深度学习的方法介绍及金融领域应用实例》的“为什么要用深度学习”一节），在特征提取之后，得到的特征可以直接输入模型进行训练。这一优势集中体现在两点上：

1. 可以有效处理非结构数据，如卷积神经网络处理图像数据；
2. 对于一阶特征，不需要人为构造，一定层数的神经网络都可以通过非线性激活函数逼近最佳函数。

但是深度学习的特征过程并不是万能的，简单的特征输入会有两点较为明显的局限性：

1. 在低阶特征上表现好，但是如果输入的特征涉及高阶特征，或是一些离散、稀疏的特征，特征提取效果会变差，且在训练过程中还会出现不收敛问题。针对这一点，需要人为去做一些简单的特征处理；
2. 特征往往不具有相同的阈值，当特征之间的阈值相差较大时，在优化参数的训练过程中也会使得收敛速度不一致，影响训练效率。针对这一点，可以通过归一化、标准化过程处理解决。

模型训练

机器学习模型可以简单分为传统机器学习模型和深度学习模型，传统的机器学习又可以根据模型的表达式分为树模型和线性模型，本节就常用的模型给出相关的介绍，并给出参数优化和超参数选取方面的相关细节。

线性模型

线性模型一定程度上都可以表达为特征之间的线性组合与目标之间的关系，常用的线性模型有：广义线性回归模型（包括普通线性回归、Lasso 回归、Ridge 回归、逻辑回归、弹性网络回归）和支持向量机。

广义线性回归模型

广义线性回归模型基于指数分布族：

$$P(y;\eta) = b(y) \cdot exp(\eta^T T(y) - a(\eta))$$

其中η为自然参数，$T(y)$为充分统计量。通常来说$T(y) = y$。

从广义线性回归模型的角度来看，普通线性模型是当y服从正态分布的情形；逻辑回归是当y服从伯努利分布的情形；Lasso 回归是y服从正态分布，且系数服从拉普拉斯分布的情形；Ridge 回归是y服从正态分布，且系数服从正态分布的情形。

在《机器学习白皮书系列一：监督学习的方法介绍及金融领域应用实例》中我们介绍了普通线性回归、Lasso 回归、Ridge 回归和弹性网络回归。

逻辑回归主要解决的任务是分类，通过 Sigmoid 函数，将实数范围内所有值映射到 0、1 之间，做从数值到标签的映射，以此建立回归任务，解决二分类问题。Sigmoid 函数的表达式和回归问题如下：

$$f(x) = \frac{1}{(1+e^{-x})}$$

$$P(y=1|\boldsymbol{x};\boldsymbol{\theta}) = \frac{1}{(1+e^{-\boldsymbol{\theta}^T\boldsymbol{x}})} \Leftrightarrow \ln\left(\frac{P(y=1|\boldsymbol{x};\boldsymbol{\theta})}{1-P(y=1|\boldsymbol{x};\boldsymbol{\theta})}\right) = \boldsymbol{\theta}^T\boldsymbol{x}$$

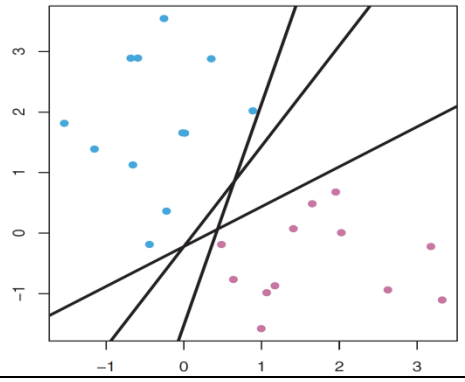
逻辑回归可以很快速的对分类问题进行建模，对连续性和类别型变量都适用，目前已经广泛的应用于工业界的很多问题。逻辑回归存在着线性回归都有的问题，如对特征中的多重共线性较为敏感，但可以通过特征提取、增加惩罚项等方法来解决这些问题。逻辑回归的主要问题在于当数据维度很大的时候，表现效果一般，同时由于预测结果呈现“S”型，中间区间的概率变化较大，很难确定一个较为合理的阈值。

支持向量机

支持向量机起源于线性超平面对当前平面的划分，所以最开始被用来解决分类问题。

对于一个简单的二分类问题，平面上两种不同颜色的点，可以通过一个超平面（直线）将两个颜色的点分开：

图 3：支持向量机运作示意图



资料来源：An Introduction to Statistical Learning，长江证券研究所

分类函数可以写成超平面表达式：

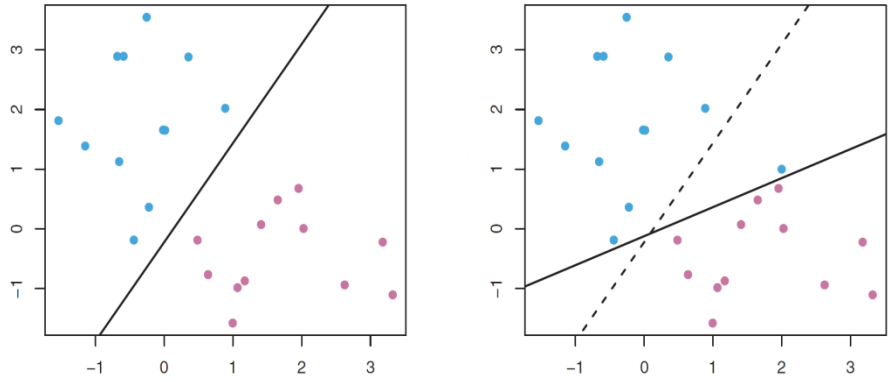
$$f(x) = \boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{b} = 0$$

空间中存在很多超平面可以划分两种颜色的点，这些超平面根据法向量可以分成一个个平面簇，如上图中即对应了多个不同的平面簇，每个簇中可以使得两个点集刚好被分开的两个超平面之间的几何距离，称为 *gap*，表示两个点集之间的距离，而落在这两个超平面上的点，被称为**支持向量**。最优的参数（$\boldsymbol{w}$和$\boldsymbol{b}$）就是使得 *gap* 值最大的超平面。为了方便求解$\boldsymbol{w}$和$\boldsymbol{b}$，限定支持向量满足$\boldsymbol{w}^T\boldsymbol{x} + \boldsymbol{b} = \pm 1$。可以证明最优化的目标为：

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2, \quad s.t.\, y_i(\boldsymbol{w}^T x_i + \boldsymbol{b}) \geq 1, i = 1, \ldots, n$$

上述情况讨论了点集可以被超平面完全分开的情况，但由于噪声的存在，绝大多数的情况下点集很难被一个超平面分开，还有些情况因为噪声的存在使得整个模型受到影响，如图：

图 4：噪音对支持向量机的影响



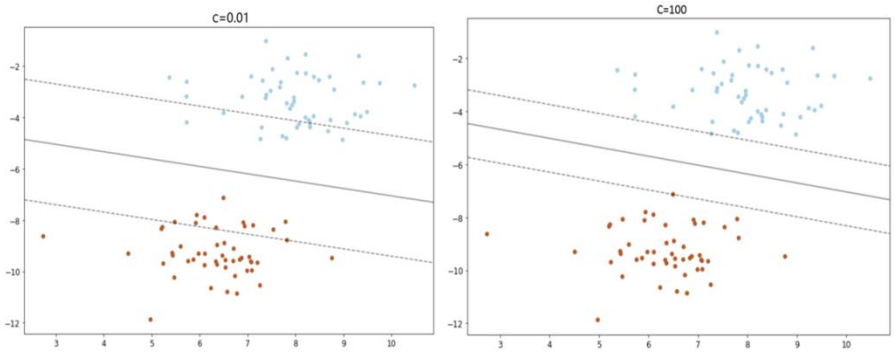
资料来源：An Introduction to Statistical Learning，长江证券研究所

上图由于异常蓝色点的存在，使得原本较优的超平面产生了偏离。为了对应这种情况，支持向量机允许数据点在一定程度上偏离超平面。原约束条件$y_i(\boldsymbol{w}^T x_i + \boldsymbol{b}) \geq 1$表明了所有点都是可以被超平面所分的，现在将约束条件变为$y_i(\boldsymbol{w}^T x_i + \boldsymbol{b}) \geq 1 - \xi_i$，表明允许一些数据点不被超平面所分，同时目标函数变为：

$$\min \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{N}\xi_i$$

C 为惩罚系数，是支持向量机的超参，值越大表明对分类错误的容忍度越低，模型方差越大，值越小表明对分类错误容忍度越高，模型方差越小。

图 5：支持向量机惩罚项对拟合效果影响



资料来源：长江证券研究所

普通的支持向量机可以处理线性可分的情况，而对于线性不可分的情况，需要通过函数，将数据映射到高维空间，在高维空间中处理线性可分的情况。理论上，任何样本都可以通过函数变换做到线性可分。函数对原数据集的处理可以表示为$\boldsymbol{\phi}(\boldsymbol{x})$，超平面变为$f(\boldsymbol{x}) = \sum_{i=1}^{N} w_i \phi_i(\boldsymbol{x}) + \boldsymbol{b}$。经过变化的问题可以通过核函数的技术进行求解。核函数用来表示两个函数在特征空间的内积：$\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{z}) = \langle \boldsymbol{\phi}(\boldsymbol{x}) \cdot \boldsymbol{\phi}(\boldsymbol{z}) \rangle$，这里不具体展开介绍。常用的核函数有：

线性核：$K(\boldsymbol{x},\boldsymbol{z})=\langle \boldsymbol{x},\boldsymbol{z}\rangle=\boldsymbol{x}^T\boldsymbol{z}$。线性核主要用于样本空间线性可分的情形，有着参数少速度快的优势。

多项式核：$K(\boldsymbol{x},\boldsymbol{z})=(\gamma\langle \boldsymbol{x},\boldsymbol{z}\rangle+1)^d=(\gamma\boldsymbol{x}^T\boldsymbol{z}+1)^d$。多项式核属于全局核函数，维数越大，映射空间越高，对样本的区分越明显，学习复杂度越高，计算量越大，也越容易过拟合。

Sigmoid 核：$K(\boldsymbol{x},\boldsymbol{z})=tanh(\gamma\langle \boldsymbol{x},\boldsymbol{z}\rangle+1)=tanh(\gamma\boldsymbol{x}^T\boldsymbol{z}+1)$。当模型采用 Sigmoid 函数作为核函数时，支持向量机本质上实现的是多层感知器神经网络。

高斯核：$K(\boldsymbol{x},\boldsymbol{z})=exp(-\gamma\frac{\|\boldsymbol{x}-\boldsymbol{z}\|^2}{2\sigma^2})$，高斯核可以将原始空间映射为无穷维，$\sigma$越大，高次特征权重越小，近似于低维空间，σ越小，实际空间维数越高，也越容易过拟合。

支持向量机回归在本质上类似分类的做法，通过一个“边界”对数据集给出划分，这个边界即为回归曲线，区别在于分类认为边界内的点是错误分类，从而进行惩罚，而回归认为在边界内的点是噪音，对回归没有影响，不对这些数据进行惩罚。支持向量机回归的形式类似 Ridge 回归，Ridge 回归是带有 L2 范数惩罚项的、以均方误差为基础的回归，支持向量机回归将均方误差替换为了 hinge loss（铰链损失），其表达形式如下：

$$\operatorname{argmin}L(\boldsymbol{w},\boldsymbol{b})=C\sum_{i}^{n}\max(0,1-y_i(\boldsymbol{w}^T\boldsymbol{\phi}(x_i)+\boldsymbol{b}))+\frac{1}{2}\sum_{j}^{d}w_j^2$$

$$=\max(0,1-\boldsymbol{y}(\boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x})+\boldsymbol{b}))+\lambda\|\boldsymbol{w}\|_2^2$$

线性模型比较

线性模型的相关比较如下表所示：

表 4：线性模型比较

	应用场景	常见超参数及调整	优点	缺点
普通线性回归	捕捉输入与输出的线性关系	无	简单高效	前提假设要求高
Lasso 回归	筛选变量，降低特征空间维数	范数 1 惩罚系数：值越大特征空间维数越低，欠拟合越严重	可以降低特征空间维数，较易校准	欠拟合
Ridge 回归	防止过拟合	范数 2 惩罚系数：值越大过拟合效果越低，欠拟合越严重	减少过拟合，较易校准	欠拟合
弹性网络回归	筛选变量，降低特征空间维数，防止过拟合	范数 1 和范数 2 惩罚系数：同 Lasso 和 Ridge 回归	同时降低特征空间和防止过拟合	欠拟合，较难校准
逻辑回归	二分类	范数 1 或范数 2 惩罚系数：同 Lasso 和 Ridge 回归	简单高效	欠拟合，精确度不高
支持向量机	二分类/回归	惩罚系数：值越大对分类错误的容忍度越低，方差越大 核函数：映射空间越高分类效果越明显，方差越大 Gamma：值越大，映射空间越稀疏，分类效果越明显，方差越大	可以处理非线性可分问题，可以解决高维空间问题	较难训练，较难校准，样本容量增加时模型表现一般，分类问题无法输出伪概率

资料来源：长江证券研究所

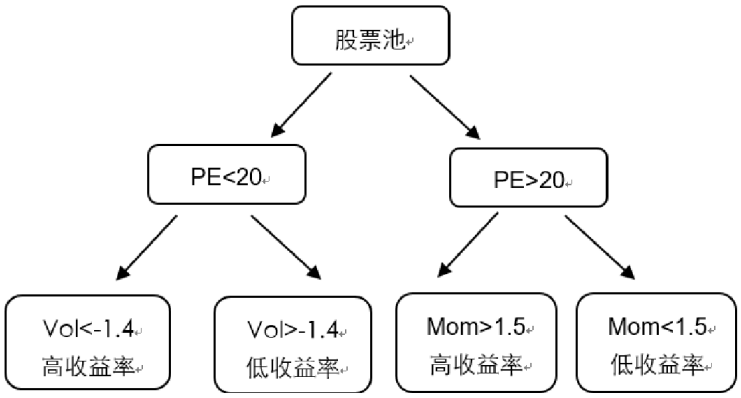
树模型

树模型以决策树为基础，在之上衍生出了各种算法，从集成学习的角度考虑，树模型可以分为 Bagging 和 Boosting 模型，下面从这个角度，介绍经典的树模型算法。

决策树模型

决策树最初被用来设计解决分类问题，每个内部节点代表对某一属性的一次测试，每条边代表一个测试结果，叶节点代表某个类，决策树的决策过程需要从决策树的根节点开始，待测数据与决策树中的特征节点进行比较，并按照比较结果选择下一比较分支，直到叶子节点作为最终决策结果，一个简单的流程如下图所示：

图 6：决策树预测流程



资料来源：长江证券研究所

决策树生成的核心在于每个节点的判断规则的生成，比较著名的算法有 ID3 和 CART 算法，其主要不同在于判断节点分裂时的标准。

ID3 算法的核心思想是以信息增益作为节点判断的标准，选择分裂后信息增益最大的属性进行分裂。首先是信息熵的概念，信息熵是用来衡量一个随机变量出现的期望值，信息不确定性越大，熵越大，定义如下：

$$Info(D) = \sum_{i=1}^{m} -p_i log_2 p_i$$

其中D为所有事件集合，p为事件发生概率，m为特征总数。信息增益是指根据节点的划分熵的变化，即根据属性分裂后导致的确定程度的增加，当确定了当前属性后，得到的信息增益为：

$$Gain(D, A) = Info(D) - \sum_{j=1}^{v} \frac{D_j}{D} Info(D_j)$$

其中A为属性，根据**信息增益最大**的属性进行节点的分裂。

CART 使用基尼指数作为判断节点的标准，基尼指数定义为：

$$Gini(D) = 1 - \sum_{i=1}^{m} (\frac{N_i}{N})^2$$

其中N和N_i分别表示集合D中样本数据总数和第i个分类的样本数量。则分类后的基尼指数为：

$$Gain(D,A)=\sum_{j=1}^{v}\frac{D_j}{D}Gini(D_j)$$

根据**基尼系数最小**的属性进行节点分裂。

决策树的算法流程可以简单描述为：

1. 初始化属性集合和数据集合；
2. 根据判断准则，确定作为当前决策节点的属性，更新数据集合和属性集合；
3. 重复第 2 步直到满足某些条件，如子集只包含单一属性或完成了对整个数据集的划分。

决策树也可以用来做回归，只需要将信息熵和基尼指数改变为体现回归效果的标准，如最常用的平方误差。

集成学习算法

对于一个机器学习问题来说，给定训练数据集，求一个弱学习算法要比求一个强学习算法要容易的多。集成学习算法的核心在于通过组合一系列的弱学习器得到一个强学习器，起到更好预测的作用。

在弱学习器的选择上，一般选择可以捕捉到输入与输出关系的、较为简单的、普适性较强的模型，如树、线性模型、逻辑回归模型等。弱学习器的种类可以随着训练的进行改变，但是目前常用算法都采用一个种类的弱学习器进行完整训练。集成学习算法根据弱学习器之间的关系，可以分成 Bagging 系列算法和 Boosting 系列算法。

Bagging 与随机森林模型

Bagging 算法是 bootstrap aggregation 的缩写，其核心思想是通过随机有放回的抽样构建训练数据集训练模型，最后组合。在**“方差”和“偏差”**一节中，我们介绍了机器学习建模中，样本误差可以分解为“方差”和“偏差”两个部分，所以降低样本误差的目标可以通过降低“方差”或降低“偏差”两个方面做到。Bagging 方法就是通过组合多个在不同样本子空间上的数据集的弱学习器，以投票或加权平均的方式得到预测结果，减少模型在不同数据集上预测的波动，从而以牺牲了部分“偏差”的代价，降低预测的“方差”，最终降低总的预测误差。随机森林算法就是 Bagging 算法中的代表。

随机森林算法主要分为两步，数据集的生成和模型的训练，具体过程如下：

1. 使用 bootstrap 的方法抽取子集作为当前模型的训练集；
2. 训练决策树；
3. 重复 1、2 步骤直到满足某些条件，如达到树的数目。

随机森林可以有效防止过拟合，并在建模过程中选择出有效特征，目前在随机森林之上衍生出了许多算法，如改变子数据集的选取方式，将有放回的随机抽样变为无放回的随机抽样，增加列维度上的数据随机抽样，但模型的训练部分并无改变。

近些年来使用的越来越多的 Extremely Randomized Trees（简称 Extra Trees）则在之上开辟了一个新的思路，通过在模型的训练部分的改变，衍生出了新的组合树的形式，其模型的建立流程如下：

1. 使用 bootstrap 的方法抽取子集作为当前模型的训练集；
2. 对每个特征，**随机选择**分裂节点，进行比较，选择最优的特征和其分裂节点；
3. 重复 1、2 步骤直到满足某些条件，如达到树的数目。

可以看出 Extra Trees 和随机森林的本质区别在于第二步，随机森林以训练模型的方式得到每一棵树，Extra Trees 以随机分裂的方式得到每一棵树。目前 Extra Trees 已经在很多地方逐渐取代随机森林成为树模型中最常用的模型，主要有以下三点原因：

1. 表现上并不逊于随机森林，且在噪声大的数据中表现更好；
2. 训练速度更快；
3. 和大部分模型的相关性低，在模型融合上有很大优势。

第三点是 Extra Trees 逐渐取代随机森林的主要原因，尤其在当前大部分机器学习问题上，GBDT 作为主要的模型，Extra Trees 的表现和其相关性较低，可以有效在降低模型的整体“方差”上面做出贡献。

Boosting 与提升树模型

Boosting 方法是另一种通过弱学习器提高准确度的方法，和 Bagging 方法不同的是，Boosting 每次根据之前模型的表现，进行新的模型的训练，以改变训练数据的权值和弱分类器的组合方式，得到最后的强学习器。Adaboost 和 GBDT 是其中的代表。

Adaboost

Adaboost 通过提高被前一轮弱分类器错误分类样本的权值，降低那些被正确分类样本的权值，同时加大误差小的学习器的权值，减小误差大的学习器的权值，组合成强学习器。Adaboost 的算法流程如下：

1. 初始化，$D_1 = (w_{11}, w_{12}, \dots, w_{1M}),\quad w_{1i} = \frac{1}{M}, i = 1,2, \dots M$
2. 训练 K 个弱分类器$k = 1,2, \dots, K$

   （1） 训练有权值的弱学习器$G_k(x)$。

   （2） 计算$G_k(x)$在训练集上的分类误差率e_k。

   （3） 计算弱学习器的系数$\alpha_k = \frac{1}{2} log \frac{1-e_k}{e_k}$。

   （4） 更新数据集的权值分布：

$$D_{k+1} = \left(w_{k+1,1}, w_{k+1,2}, \dots, w_{k+1,M}\right)$$

$$w_{k+1,i} = \frac{w_{k,i}\, exp\left(-\alpha_k y_i G_k(x_i)\right)}{\sum_{i=1}^{M} w_{k,i} exp(-\alpha_k y_i G_k(x_i))}, \qquad i = 1,2, \dots, M$$

3. 构建基本分类器加权线性组合

$$f(x)=\sum_{k=1}^{K}\alpha_k G_k(x)$$

Adaboost 可以自适应的调整样本权重进行训练，调整模型权重进行合成，算法实现简单，应用广泛。

GBDT

对于简单的损失函数，如指数损失和平方损失，每一次提升都较为简单，但是对于一般的损失函数（如绝对损失），优化难度大大增加。因此 GBDT 利用损失函数负梯度在当前模型的值作为提升树中残差的近似值，拟合一个回归树。GBDT 的具体算法如下：

1. 初始化模型

$$f_0(x)=\arg\min_c\sum_{i=1}^{M}L(y^{(i)},c)$$

2. 循环训练 K 个模型$k=1,2,\dots,K$

（1） 计算负梯度：对于$i=1,2,\dots,M$

$$r_{ki}=-\left[\frac{\partial L(y^{(i)},f(x^{(i)}))}{\partial f(x^{(i)})}\right]_{f(x)=f_{k-1}(x)}$$

（2） 以负梯度r_{ki}训练模型，得到第 k 颗树的叶结点区域R_{kj}，$j=1,2,\dots,J$

（3） 对$j=1,2,\dots,J$，计算：

$$c_{kj}=\arg\min_c\sum_{x^{(i)}\in R_{kj}}L(y^{(i)},f_{k-1}(x^{(i)})+c)$$

（4） 更新模型：

$$f_k(x)=f_{k-1}(x)+\sum_{j=1}^{J}c_{kj}I(x\in R_{kj})$$

3. 得到最终模型：

$$\hat{f}(x)=f_K(x)=\sum_{k=1}^{K}\sum_{j=1}^{J}c_{kj}I(x\in R_{kj})$$

XGBoost 是一种高效的 Boosting 训练器，可以实现 GBDT 的功能。且不同于一般的 GBDT，XGBoost 采用损失函数的二阶泰勒展开来近似原损失函数，同时在损失函数后加入惩罚项：

$$\Gamma^m=\sum_{i=1}^{n}L\left(y_i,\hat{y}^{m-1}+f_m(x_i)\right)+\Omega(f_m)$$

$$\approx\sum_{i=1}^{n}\left[L(y_i,\hat{y}^{m-1})+g_i f_m(x_i)+\frac{1}{2}h_i f_m^2(x_i)\right]+\Omega(f_m)$$

$$\tilde{\Gamma}^{(m)}=\sum_{i=1}^{n}\left[g_i f_m(x_i)+\frac{1}{2}h_i f_m^2(x_i)\right]+\Omega(f_m)$$

以树模型作为弱分类器为例，得到区别于 GBDT 构造树的过程。

首先将f_t和Ω的表达式带入目标函数中，得到目标函数的如下形式：

$$\tilde{\Gamma}^{(m)} = \sum_{i=1}^{n}\left[g_i f_m(x_i) + \frac{1}{2}h_i f_m^2(x_i)\right] + \gamma T + \frac{1}{2}\lambda\sum_{j=1}^{T} w_j^2$$

$$= \sum_{j=1}^{T}[(\sum_{i\in I_j} g_i)w_j + \frac{1}{2}(\sum_{i\in I_j} h_i + \lambda)w_j^2] + \gamma T$$

由此可得函数的极小值点：

$$w_j^* = -\frac{\sum_{i\in I_j} g_i}{\sum_{i\in I_j} h_i + \lambda}$$

每次对已有的叶子加入一个分割时，只需通过以下式子判断是否进行分割：

$$\Gamma_{split} = \frac{1}{2}\left[\frac{(\sum_{i\in I_L} g_i)^2}{\sum_{i\in I_L} h_i + \lambda} + \frac{(\sum_{i\in I_R} g_i)^2}{\sum_{i\in I_j} h_R + \lambda} - \frac{(\sum_{i\in I} g_i)^2}{\sum_{i\in I} h_i + \lambda}\right] - \gamma$$

其中I_L和I_R分别属于分裂后的左节点、右节点集合，$I = I_L \cup I_R$。

相比于一般的 GBDT，XGBoost 具有以下优点：

1. 传统 GBDT 在优化时只用到了梯度方向的信息，XGBoost 对损失函数进行二阶泰勒展开，不仅用到了二阶导数的信息，还可以根据数据的特点自定义损失函数，只要一阶导和二阶导存在。
2. XGBoost 在损失函数中加入了惩罚项，可以有效降低模型的过拟合。
3. XGBoost 不仅支持树模型作为弱学习器，也支持线性模型、Logistic 模型等作为弱学习器，可以针对数据特点选取更合适的弱学习器。
4. XGBoost 借鉴了 Bagging 中的思想，可以实现在每次 Boosting 的过程中随机抽样样本集和特征集合，有效防止过拟合。
5. XGBoost 支持在每颗树的迭代过程中子节点的并行计算，且在搜索上通过预排序更快的实现算法。

XGBoost 存在以下缺点：

1. 在选择树的分隔点时，需要遍历所有特征值。
2. 预排序结果的保存消耗大量内存。

微软推出的 LightGBM 优化了 XGBoost 的上述问题：

1. LightGBM 采用 Histogram 的决策树算法，相比于 XGBoost 的预排序算法，可以减少内存的消耗，在数据分割上，以$O(n)$的时间复杂度优于预排序的$O(n \times k)$的时间复杂度（n为数据长度，k为特征长度），并且可以大幅减少计算分割点增益的次数。
2. LightGBM 抛弃了大多数 GBDT 工具使用的按层生长，而使用带深度限制的按叶子生长的算法，在遍历一次数据的基础上，可以同时分裂同一层叶子，更容易多线程优化，降低更多的误差，得到更精确的模型。

LightGBM 也有自身的缺点：

1. Histogram 的决策树算法不能很精确的找到分割点，训练误差比不上预排序算法，虽然从大量实验结果上来看差异并不大，并且在有些情况下由于对分割点的精确度不敏感，效果会优于预排序算法（类似 Extra Trees 算法和随机森林算法）。
2. 按叶子生长的算法会产生较深的决策树，容易过拟合，但这一点可以通过超参的限制防止过拟。

在 Kaggle 的比赛中，越来越多的算法融合了 LightGBM，总体来说，LightGBM 和 XGBoost 在不同问题上表现各有优劣，但是速度上，LightGBM 很多情况下都优于 XGBoost，所以目前也成为数据科学家建立 GBDT 模型的工具。

树模型比较

树模型的相关比较如下表所示：

表 5：树模型比较

	应用场景	常见超参数及调整	优点	缺点
决策树	分类/回归	最大深度：值越大学习能力越强，越容易过拟合	简单高效，可捕捉非线性关系	过拟合
随机森林	分类/回归	最大深度：值越大学习能力越强，越容易过拟合 树的棵数：值越大，方差越小。 样本抽样率：值越小，方差越小，偏差越大 特征抽样率：值越小，方差越小，偏差越大	表现好，模型精确度高，可以捕捉非线性关系，可以得到特征重要程度，可并行，可以减小方差	噪音大的数据下过拟合，类别变量影响较大
Extra Trees	分类/回归	同随机森林	同随机森林	精确度相对低，类别变量影响较大
Adaboost	分类/回归	基模型：根据具体问题选择不同基模型，调整参数 迭代次数：值越大，偏差越小，方差越大 学习率：值越小，偏差越小，方差越大，时间越长	表现好，可以捕捉非线性关系，弱学习器有权重合成，可以得到特征重要程度	过拟合，无法并行实现
GBDT	分类/回归	最大深度：值越大学习能力越强，越容易过拟合。 树的棵数：值越大，方差越小。 样本抽样率：值越小，方差越小，偏差越大 特征抽样率：值越小，方差越小，偏差越大 学习率：值越小，偏差越小，方差越大，时间越长	表现很好，可以捕捉非线性关系，可以得到特征重要程度	过拟合，无法并行实现

资料来源：长江证券研究所

深度学习模型

在《机器学习白皮书系列之三：深度学习的方法介绍及金融领域应用实例》中我们对深度神经网络（DNN）、卷积神经网络（CNN）和循环神经网络（RNN）这些目前较为常用且表现较好的模型给出了较为详细的介绍。深度学习算法除了网络结构的实现上之外，还有一个很重要的部分就是参数的优化。在深度神经网络一节中，我们简单的介绍了在参数求解的过程中可能存在的问题，本节将具体介绍深度学习应用中参数求解的优化算法，并给出深度学习模型在训练时的一些技巧。

最优化算法

对于任何凸优化问题，都可以通过梯度下降法找到局部最优数值解：

$$g_t = \nabla_{\theta_{t-1}} f(\theta_{t-1})$$

$$\Delta\theta_t = -\eta * g_t$$

其中η是学习率，g_t是梯度。

全局梯度下降法用样本内全数据进行训练，但是当数据量很大时，一次性载入所有数据效率低下，由此引进了随机梯度下降法（stochastic gradient descent，SGD），即在梯度下降法的基础上，每一次迭代计算一部分数据的梯度。但是仍存在以下两个缺点：

1. 选择合适的学习率比较困难，因为不同的子数据集适应的 learning rate 不同，而在更新过程中赋予同样的权重，尤其存在稀疏特征的情况下，会使得对一些特征更新速度偏慢。

2. SGD 很容易收敛到局部最优。

为了解决随机梯度下降法的权重问题，Adagrad（Adaptive Subgradient）提出了自适应调整学习率的算法：

$$\Delta\theta_t = -\frac{\eta}{\sqrt{\sum_{r=1}^{t}(g_r)^2 + \epsilon}} * g_t$$

通过一个递推形成的约束项$\frac{1}{\sqrt{\sum_{r=1}^{t}(g_r)^2+\epsilon}}$，在训练前期$g_t$较小的时候，约束项较大，能够放大梯度，更快的朝极值方向，在训练后期g_t较大的时候，约束项较小，可以减小梯度。

但是 Adagrad 并未完全摆脱人为控制学习率的限制，式子中的η仍需要人为设置，初始化会有影响，且后期由于惩罚项的减小，使得训练提前结束。针对上述问题，Adadelta 进行了计算上的简化。区别于 Adagrad 累加之前所有的梯度平方，Adadelta 只累加固定大小的项，同时为了摆脱全局学习率的依赖，经过近似牛顿迭代法处理：

$$E|g^2|_t = \rho * E|g^2|_{t-1} + (1-\rho) * g_t^2$$

$$\Delta x_t = -\frac{\sqrt{\sum_{r=1}^{t-1}\Delta x_r}}{\sqrt{E|g^2|_t + \epsilon}}$$

Adadelta 可以在训练中期同样保证优化效果，训练后期在局部最小值附近波动。

RMSprop 作为 Adadelta $\rho = 0.5$的一个特例的改进，介于 Adagrad 和 Adadelta 之间：

$$RMS|g|_t = \sqrt{E|g^2|_t + \epsilon}$$

$$\Delta x_t = -\frac{\eta}{RMS|g|_t} * g_t$$

之所以单独把 RMSprop 提出来介绍，是因为经验上 RMSprop 很适合处理非平稳目标，尤其对于 RNN 模型的训练效果很好。

Adam（Adaptive Moment Estimation）是目前最为流行的一种求最优解的优化算法，本质上是带有动量项的 RMSprop，它利用梯度的一阶矩估计和二阶矩估计动态调整每个参数的学习率。Adam 的优点主要在于经过偏置校正后，每一次迭代学习率都有个确定范围，使得参数比较平稳，其优化流程如下：

$$m_t = \mu * m_{t-1} + (1-\mu) * g_t$$

$$n_t = v * n_{t-1} + (1 - v) * g_t^2$$

$$\hat{m}_t = \frac{m_t}{1 - \mu^t}$$

$$\hat{n}_t = \frac{n_t}{1 - v^t}$$

$$\Delta\theta_t = -\frac{\hat{m}_t}{\sqrt{\hat{n}_t} + \epsilon} * \eta$$

Adam 结合了 Adagrad 善于处理稀疏矩阵梯度和 RMSprop 善于处理非平稳目标的优点，自适应的调整学习率，可以很好的适用于大数据集和高维空间。

针对不同的问题和应用场景，在选择优化算法上一般遵循以下规则：

1. SGD 仍然是目前最常用的优化器，在好的初始化和学习率下，结果最为可靠；
2. 对于一些特定的数据集，如带有稀疏变量的数据或者非平稳的数据，尽量选择自适应的优化方法，如 Adam；
3. 对于一些特定结构，如 RNN，尽量选择自适应的优化方法，如 RMSprop；
4. 从表现上来看，Adadelta、RMSprop 和 Adam 表现差别不大。

训练模型参数

由于随机梯度法及其改进在深度学习训练中的广泛应用，优化算法的参数选择显得尤为重要，随机梯度法通过在子样本上的梯度方向上的下降，不断遍历整个数据集，最终找到参数稳定的最优值。优化算法一般有两个需要调整的参数：batch 和 epoch。

batch 是指每次计算梯度时用到的数据集大小，epoch 是指整个训练过程遍历数据集的次数，batch 和 epoch 一般来说存在一个制衡的关系，即 batch 越大，优化达到稳定所需要遍历数据集的次数越少，batch 越小，优化达到稳定所需要遍历数据集的次数越多。但是由于 batch 大小会影响整个梯度下降过程和最终收敛的结果，所以 batch 大小较为重要。

当 batch 较小时（如 batch 大小为 1 时），每次梯度下降的方向朝着子样本的梯度方向修正，很难达到收敛，精确度较低；随着 batch 的增加，处理整个数据集的速度变快，同时达到相同精度要遍历数据集的次数减小；当 batch 大小接近整个数据集时，每次梯度下降都会向着样本内最快的下降方向，最终得到的参数也会更加精确，但会因为数据量太大导致每个回合训练的效率低下。所以在一定范围内适当增加 batch 大小时，不仅可以提高内存使用率，减少每次遍历数据集所需次数，同时可以增加梯度下降的准确性，使算法更快的收敛，达到时间上和精度上的最优。

除了上述介绍的线性模型、树模型和深度学习模型之外，对应一些特殊的建模问题，有一些特定的方法：K 邻近算法的实现简单，逻辑清晰，可以很好的捕捉数据中的非线性关系，在很多工业问题中都有着很好的应用；局部线性回归 LOESS 根据数据的距离分配权重，可以提高回归的精确性；卡曼滤波、小波变换可以很好的过滤数据中的噪音；朴素贝叶斯模型在数据量有限的情况下，在回归和分类上的表现都很好，并且衍生出了

贝叶斯 ARD 回归、贝叶斯 Ridge 回归；高斯混合模型在监督和无监督领域都有很好的应用，如在一些网格搜索较难进行的超参空间中寻找最优超参组合。

交叉验证

在机器学习问题一节中，我们通过泛函中的概念，将建模流程和经验风险泛函（样本误差）通过映射联系起来，但是并没有说明这个学习过程是有效的，事实上，学习过程有效性的前提在于我们可以用经验风险泛函去近似代替风险泛函，其中数据集的独立同分布就是这一前提成立的一个必要条件（另一必要条件在函数上也要求有一致性，这里不展开介绍）。所以机器学习模型目前多用于模式识别，因为样本内外数据分布很少产生变化。而对于很多非模式识别问题，为了保证样本内外数据尽量在某一维度上的独立同分布特性，我们会采用交叉验证的方法。

交叉验证是机器学习中很重要的一部分，其应用体现在建模方方面面，目的是为了降低模型的“方差”，提高模型的泛化能力，得到稳定的模型。交叉验证一般遵循以下要点：

1. 只有训练集可以用在模型的建立过程中，包括特征的提取、模型簇的选择、超参的确定、模型融合的标准，测试集只在完成模型之后用来评估**建模过程**的优劣。
2. 训练集中样本要足够多，一般大于样本总数的 50%。
3. 训练集和测试集通过均匀取样的方式得到，以保证分布的统一。

交叉验证的具体流程如下：

1. 将所有数据分成若干份。
2. 每次取其中一份作为验证集，其余数据作为训练集，训练模型，以验证集的标准作为模型表现的评价。
3. 重复第 2 步，直到遍历所有数据，并以所有验证集的评价标准的平均值，作为当前模型表现的评价。

交叉验证对数据集的划分的个数会影响模型的泛化能力，当划分数据集的个数过少时（如未作划分），模型根据全部数据进行训练，“偏差”较小，但是由于模型缺乏在不同数据上的多次训练，“方差”较大；反之，当划分数据集的个数过大时，模型的“方差”会减小但“偏差”会增大。一般来说，划分数据集的个数选取在 5 到 10 之间。

交叉验证的评价作为样本误差的一个变形，可以和建模流程一一对应起来，通过选取最优的交叉验证评价，确定相对最优的建模流程。一个最简单的建模流程就是从一个模型簇中选择最优的模型，即模型超参的选择。在模型训练一章中，我们介绍了常用机器学习模型中的超参数，并给出了超参对模型表现的影响，但是如何确定一个相对最优的超参，很难通过人为的方式直接确定。不同的超参得到的模型为一个模型簇，模型簇中每个模型经过交叉验证都可以得到一个模型的评价，通过列举一系列参数的组合，比较得到的模型评价，选择相对最优的超参组合。

模型融合

在交叉验证一节中，我们从学习的本质引出了做交叉验证的必要性，即尽量保证样本内外的数据同分布，而经验风险泛函能够代替风险泛函的另一个必要条件是要求在函数空

间上也满足一致性，模型融合最初的想法便来源于此，即通过在函数维度上，保证机器学习的一致性。

当一个函数空间中有很多函数都建立了从样本空间到样本误差的映射时，函数的不同会使得映射不同，从而使得样本误差产生变化。当选取的函数较少的时候，随数据集的改变样本误差变化较大，即“方差”较大，模型融合即通过增加选取用来逼近最优函数的个数，以“偏差”的略微增大换取“方差”的极大减小。

模型融合的成功一般有两个必要条件：

1. 被用来融合的模型表现都较好；
2. 模型之间的相关性低。

以上两点分别保证了模型“偏差”增加不大以及“方差”极大减小。在树模型中我们简单的介绍了集成学习算法，集成学习算法也是最简单的模型融合算法，和一些简单的模型融合方式共同组成了目前模型融合中效率最高的一些算法，具体描述如下表：

表 6：简单的模型融合

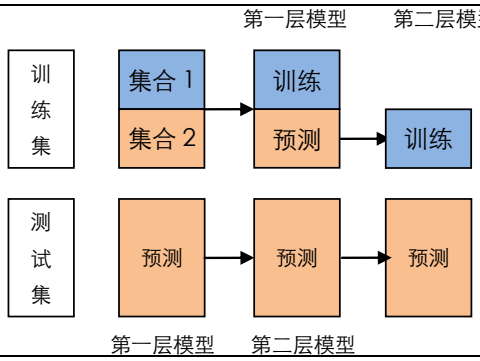
融合方法	机器学习任务	训练集合	模型	方法
Voting	分类	同一训练集	不同模型	加权投票
Averaging	回归/分类	同一训练集	不同模型	加权平均
Ranking	排序	同一训练集	不同模型	加权排序
Bagging	分类/回归/排序	不同训练集	不同模型	Voting/Averaging/Ranking
Boosting	分类/回归/排序	不同训练集	同一模型	以上一步结果更新当前模型

资料来源：长江证券研究所

复杂的模型融合

上述介绍的较为简单的模型融合方式都是横向拼接模型，Boosting 串行进行模型训练的融合方法也是直接对模型结果进行一定程度上的加权合成。Blending 融合的方法借鉴了**神经网络**的结构，纵向的融合模型，前一阶段模型的输出作为下一阶段模型的输入，建立多层的模型融合层次。以两层的 Blending 结构为例，流程如下图所示：

图 7：Blending 流程图



资料来源：长江证券研究所

训练部分：

1. 将训练集分成两份；
2. 根据第一部分数据，训练第一层模型；

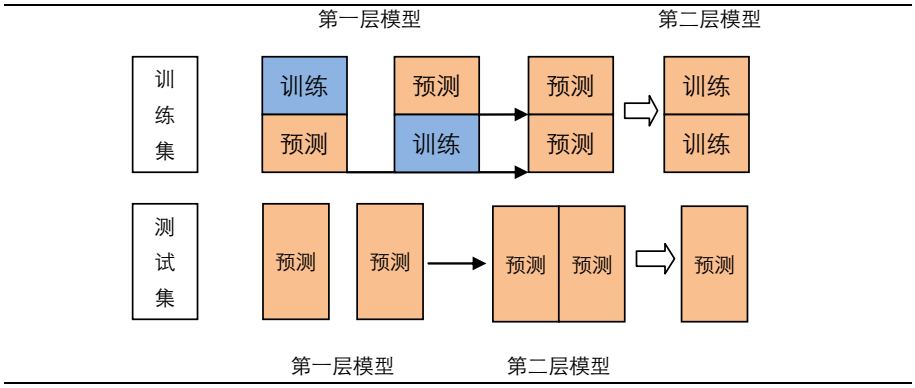
3. 由第一层模型，对第二部分数据给出预测，会得到一个第一层模型个数维数的向量，作为第二层模型的特征，训练第二层模型。

预测部分：

1. 根据第一层模型预测得到第二层模型的特征；
2. 根据第二层模型得到最终预测结果。

Blending 构建了多层模型的融合，但是在数据利用上存在缺陷：在每一阶段模型的训练上，都只用到了部分数据。Stacking 在数据利用上对 Blending 模型给出了改进，引进了**交叉验证**的思想，充分利用数据。以两层的 Stacking 为例，这里对数据集做两份划分，流程如下：

图 8：Stacking 流程图



资料来源：长江证券研究所

训练部分：

1. 将训练集分成两份；
2. 分别用两份数据，对第一层模型给出训练，并在另一份数据上给出预测；
3. 由第一层模型在另一份数据上给出的预测做一个简单的“堆叠”，作为第二层模型的特征，训练第二层模型。

预测部分：

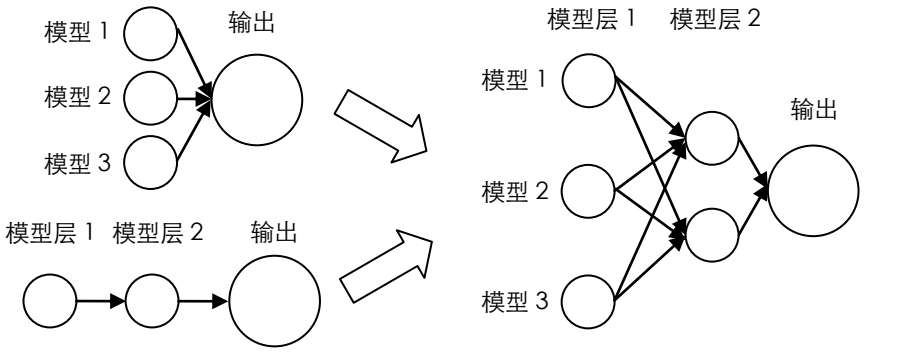
1. 根据第一层模型预测得到第二层模型的特征；
2. 根据第二层模型得到最终预测结果。

区别于 Blending 确切的模型，Stacking 更像是建立了一个从输出到输入的流程，在不同数据集上，模型不一样，而建立模型的流程是一样的，通过网络对流程进行组合。

深度学习的模型融合

模型融合的两种方式，横向和纵向，又可以结合成一个整体，如下图所示：

图 9：模型融合网络图



资料来源：长江证券研究所

不难看出，模型融合的结构类似于神经网络，其中每个模型对应了神经网络中的“神经元”， 每个横向模型融合即在同一层的模型之间通过参数分配权重得到下一层模型的输入，每个纵向模型融合即在不同层之间通过输入到输出的形式传递模型结果，每一层“神经元”个数的增加即增加了横向融合的模型簇，增加网络的层数即增加了纵向融合的模型簇。

事实上，深度学习可以将模型融合过程通过特定网络结构的设计和模型参数的优化，在模型训练的过程中完成。在《机器学习白皮书系列之三：深度学习的方法介绍及金融领域应用实例》中的总结部分，我们给出了深度学习可以在选股中应用的一个整体框架，结合了 DNN、LSTM 和 CNN 网络结构，拼接成一个大的模型。模型有三个应用场景，分别是结构数据的多因子选股、文本分析的情感分类和图像识别的技术分析，三个应用场景可以建立三个不同的模型，最后通过模型融合的方式得到整体的模型。而深度学习网络结构的存在使得我们可以将三个模型搭建在一个大的网络结构之下，作为一个整体进行训练。

虽然深度学习有着上述的模型融合的优势，但是当前设计复杂的网络结构并不是一个好的方法，主要有两个原因：

1. 大部分的深度学习结构都有着自身特定的应用场景，如 CNN 的图像识别，RNN 音频识别，并没有将各种网络结构结合的需求；
2. 过于复杂的网络结构在优化参数的问题上较难找到最优解，导致模型本身失效。

所以将深度学习模型作为一个模型簇，人为的进行模型融合，目前是一种更好的选择。

多因子选股案例

在之前的**机器学习问题**和**模型融合**两个章节中，我们介绍了机器学习问题的中样本误差可以用来逼近风险泛函以评价当前建立的模型逼近实际函数程度的条件，简单的说，即在**数据分布**上和**建模的函数空间**上，均与未来出现的问题保持一致性。对于在数据分布上不一致的问题，目前相关研究通过如滚动更新模型、交叉验证、时序模型建立等方式，得到了一定程度上的解决。而对于函数空间上的不一致性，还没有相关的解决思路，研究多集中在比较不同函数空间上策略表现。对应到多因子选股模型上，目前多以单个模型的预测构建策略，且选取的因子为整个特征空间，往往会有以下两个问题：

1. 每个模型适用的特征空间并不一样，在训练模型的过程中会产生偏差，即函数空间本身会有误差；

2. 由于策略的建立基于单个模型，在估计函数空间的一致性上会产生偏差，即单个函数空间在逼近时会有误差。

而从特征提取到模型融合的一系列完整流程可以为函数空间逼近的一致性提供一种研究思路。本章节将简单的介绍机器学习流程在多因子选股中的具体步骤，并重点比较单个模型和融合模型的结果。

建模流程

数据与特征工程

股票池：全部 A 股；剔除 ST 以及上市不足一年的股票；剔除最近一个月停牌时间超过 10 个交易日的股票；剔除每个截面的下一个交易日停牌或涨停不能交易的股票。

数据区间：2004-12-31 到 2017-12-31。

特征构建：在每个月的最后一个交易日，提取表 7 中的因子作为初始特征空间；对下个月个股涨跌幅进行排序，取序列的前 40%作为正样本，序列的后 40%作为负样本。

特征提取：

1. 中位数去极值：对某一截面的因子序列x_i，x_m为该序列的中位数，M为序列$|x_i - x_m|$的中位数，则将序列x_i中大于$x_m + 5M$的数据设置为$x_m + 5M$，序列中小于$x_m - 5M$的数据设置为$x_m - 5M$；

2. 缺失值处理：对得到的新的因子序列，将缺失的部分设置为训练样本内申万一级行业相同行业个股因子的中位数；

3. 标准化处理：将缺失值处理后的因子序列减去因子序列的均值，除以因子序列的标准差，得到最终的因子序列。

特征选择：本文选取了三组特征，分别对应之后需要融合的三个模型：

1. 第一组特征在训练集内以全部特征训练提升树，根据提升树输出的特征重要性从大到小选取，并根据特征之间相关性排除部分特征，特征中尽量包含表 7 中的所有大类因子，作为之后提升树模型的特征空间；

2. 第二组特征在训练集内以全部特征训练随机森林，根据随机森林输出的特征重要性从大到小选取，并根据特征之间相关性排除部分特征，特征中尽量包含表 7 中的所有大类因子，作为之后 Extra Trees 模型的特征空间；

3. 第三组特征为全部特征，作为之后深度神经网络模型的特征空间。

表 7：因子列表

大类因子	具体因子	因子描述	因子方向
价值	EP	净利润(TTM)/总市值	1
	BP	净资产/总市值	1
	CFP	经营性现金流(TTM)/总市值	1
	SP	营业收入(TTM)/总市值	1
	GPE	净利润(TTM)同比增速与市盈率(TTM)比值	1
成长	Sale_G_Q	营业收入季度同比增速	1
	Profit_G_Q	净利润季度同比增速	1
	OCF_G_Q	经营性现金流季度同比增速	1
	ROE_G_Q	ROE 季度同比增值	1
	Sale_G_TTM	营业收入(TTM)同比增速	1
	Profit_G_TTM	净利润(TTM)同比增速	1
	OCF_G_TTM	经营性现金流(TTM)同比增速	1
	ROE_G_TTM	ROE(TTM)同比增值	1
盈利质量	ROE_Q	ROE(YTD)	1
	ROE_TTM	ROE(TTM)	1
	ROA_Q	ROA(YTD)	1
	ROA_TTM	ROA(TTM)	1
	Grossprofitmargin_Q	毛利率(YTD)	1
	Grossprofitmargin_TTM	毛利率(TTM)	1
	Profitmargin_Q	净利率(YTD)	1
	Profitmargin_TTM	净利率(TTM)	1
	Assetturnover_Q	资产周转率(YTD)	1
	Assetturnover_TTM	资产周转率(TTM)	1
	Operationcashflowratio_Q	经营性现金流/净利润（YTD)	1
	Operationcashflowratio_TTM	经营性现金流/净利润（TTM)	1
资产结构	Currentratio	流动比率	1
	Equitydebtratio	非流动负债/净资产	-1
	Cashratio	现金比率	1
市值	Ln_size	流通市值对数	-1
反转	Alpha	过去一年与上证综指回归的 Alpha	-1
	Return_Nm	过去 N 个月涨跌幅，N=1,3,6,12	-1
波动率	Std_Res_Nm	过去 N 个月与上证综指回归的残差波动率，N=1,3,6,12	-1
	Std_Nm	过去 N 个月的个股波动率，N=1,3,6,12	-1
Beta	Beta	过去一年与上证综指回归的 Beta	-1
股价	Ln_price	股价的对数	-1
换手率	Turn_Nm	过去 N 个月的个股换手率，N=1,3,6,12	-1
	Bias_turn_Nm	过去 N 个月与过去 24 个月的个股日均换手率比值，N=1,3,6,12	-1
股东相关	Holder_num_GN	股东户数对于前 N 季度的变化，N=1,2,3	-1
	Holder_avgpercent_GN	户均持股比例相对于前 N 季度变化率，N=1,2,3	1

资料来源：长江证券研究所

模型与回测相关

回测相关：回测区间为 2011-01-31 到 2017-12-31，月度调仓，月底产生选股信号，以下一个交易日收盘价调仓，交易成本为双边 0.15%，以全部 A 股的等权指数为标的。

模型训练：每年滚动进行模型的更新，以之前六年时间的数据作为训练集。

模型簇：本案例选取了三个模型，分别为提升树模型、Extra Trees 模型和深度神经网络模型。提升树通过筛选的第一组特征，可以很好的找出数据之间的内在关系，作为主要模型；两层隐含层的深度神经网络可以捕捉绝大多数的非线性关系，通过在整个特征空间上的训练，找出提升树特征空间之外的特征和输出之间的关系；Extra Trees 因其算法本身的泛化能力强，可以有效的降低模型的方差，作为整体模型的一种补充。

模型超参和结构：通过网格搜索的方式，以 5 份子集的交叉验证 f1score 为标准，选择提升树模型（XGBoost）、Extra Trees 模型的超参数。选择 f1score 作为标准是因为 f1score 体现了正样本预测的准确性。通过交叉验证选取提升树模型的超参有：最大深度、样本抽样率和特征抽样率；通过交叉验证选取 Extra Trees 模型的超参数有：最大深度、特征抽样率；深度神经网络为包含两层隐含层的全连接网络。

模型融合：模型融合以简单的加权平均方式，得到最终模型的伪概率，即横向对模型进行融合。通过网格搜索的方式，以 5 份子集的交叉验证 f1score 为标准，确定三个模型融合的权重。

相关结果

下表展示了每年滚动更新模型时，各个模型在样本内 5 份子集的交叉验证 f1score 值之和，以及通过交叉验证选取模型融合的权重：

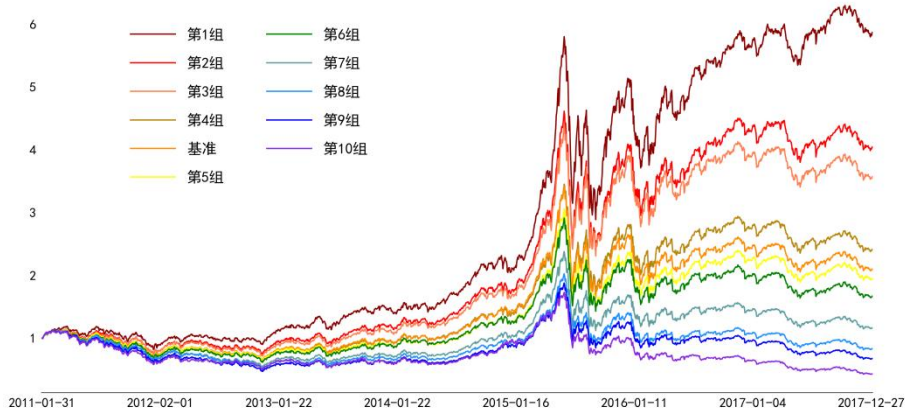
表 8：模型 f1score/模型融合权重

	提升树	Extra Trees	深度神经网络	融合模型
2011	3.00/0.75	2.94/0.00	2.77/0.25	3.02/-
2012	3.00/0.75	2.91/0.00	2.84/0.25	3.02/-
2013	2.98/0.75	2.91/0.15	2.81/0.25	3.00/-
2014	2.97/0.50	2.90/0.20	2.88/0.30	3.00/-
2015	2.99/0.60	2.91/0.00	2.94/0.40	3.02/-
2016	3.01/0.60	2.91/0.00	2.86/0.40	3.04/-
2017	3.04/0.55	2.98/0.15	2.96/0.30	3.08/-

资料来源：天软科技，长江证券研究所

本文将每期预测涨跌的伪概率分为十组进行投资组合，第 1 组为强势股，第 10 组为弱势股，下图为融合模型的分组表现：

图 10：融合模型分组表现



资料来源：天软科技，长江证券研究所

下表计算了融合模型分组的风险评价指标：

表 9：融合模型分组表现

	年化收益	超额年化	年化波	最大回撤	超额最大回	夏普比	信息比
第 1 组	72.93%	25.91%	0.289	50.24%	10.57%	1.06	2.22
第 2 组	45.62%	13.61%	0.292	49.28%	8.40%	0.86	2.07
第 3 组	38.50%	10.38%	0.294	47.65%	7.62%	0.80	2.06
第 4 组	21.17%	2.22%	0.295	51.42%	5.47%	0.60	0.63
第 5 组	14.22%	-1.08%	0.295	52.56%	13.62%	0.49	-0.32
第 6 组	10.02%	-3.05%	0.296	51.03%	21.50%	0.41	-1.11
第 7 组	2.46%	-6.61%	0.300	52.95%	44.47%	0.23	-2.86
第 8 组	-2.50%	-8.98%	0.301	59.73%	60.49%	0.06	-3.49
第 9 组	-4.86%	-10.11%	0.303	64.52%	67.83%	-0.04	-3.50
第 10	-8.54%	-11.86%	0.313	75.43%	79.34%	-0.24	-2.70

资料来源：天软科技，长江证券研究所

下表计算了融合模型第一组的分年风险评价指标

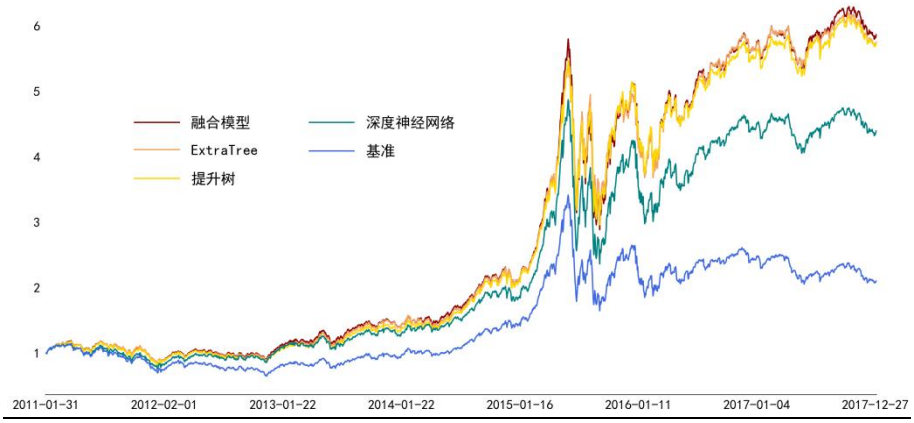
表 10：融合模型第一组分年表现

	年化收益	超额年化	年化波动	最大回撤	超额最大回	夏普比	信息比
2011	-13.50%	19.95%	0.224	27.57%	1.59%	-0.53	4.76
2012	22.61%	16.87%	0.228	15.48%	3.93%	1.00	2.54
2013	40.07%	10.52%	0.211	15.96%	2.62%	1.72	2.06
2014	40.61%	-4.28%	0.217	11.70%	9.23%	1.69	-0.63
2015	149.76%	34.67%	0.528	50.24%	9.14%	2.02	2.70
2016	12.22%	19.59%	0.307	22.88%	2.35%	0.53	3.17
2017	4.03%	19.70%	0.120	10.91%	3.18%	0.39	2.69

资料来源：天软科技，长江证券研究所

下图为各个模型第一组的多头累计净值：

图 11：各个模型第一组表现



资料来源：天软科技，长江证券研究所

下表计算了各个模型各组的年化收益和其标准差：

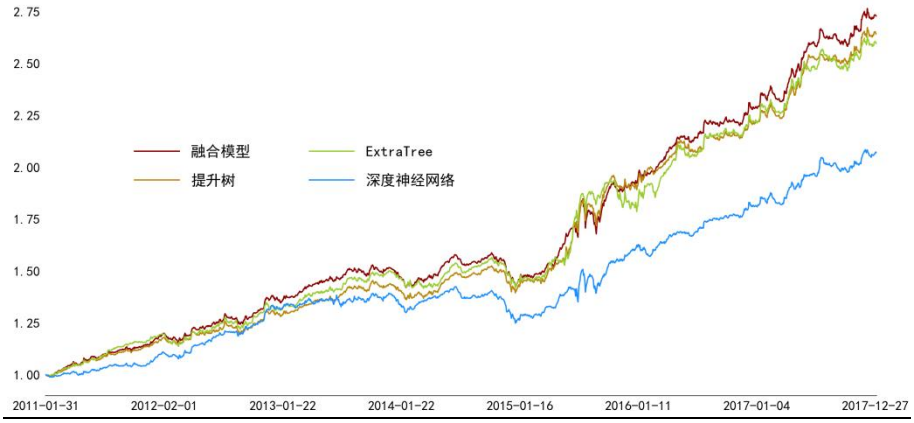
表 11：模型分组指标比较

	提升树模型	Extra Trees	深度神经网络	融合模型
年化收益标准差	22.90%	23.43%	18.36%	24.84%
超额收益标准差	11.09%	11.13%	9.14%	12.18%
夏普比标准差	0.38	0.41	0.36	0.42
Carmar 比标准差	0.51	0.55	0.37	0.52
信息比标准差	2.16	2.03	2.00	2.38
超额胜率标准差	0.240	0.236	0.255	0.267

资料来源：天软科技，长江证券研究所

下图为各个模型的超额收益表现：

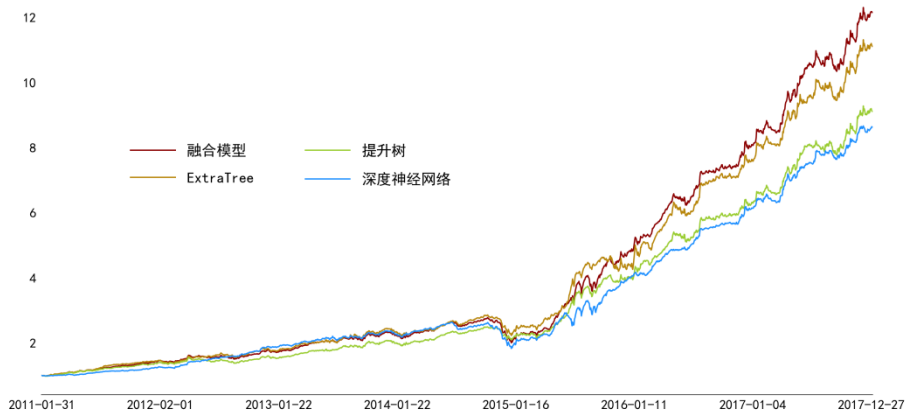
图 12：各个模型超额收益



资料来源：天软科技，长江证券研究所

下图为各个模型的多空收益表现：

图 13：各个模型多空表现



资料来源：天软科技，长江证券研究所

下表计算了各个模型超额收益与多空收益的风险指标：

表 12：模型风险评价指标

	提升树模型	Extra Trees	深度神经网络	融合模型
超额收益	24.60%	23.92%	16.05%	25.91%
超额波动	0.066	0.072	0.064	0.067
超额最大回撤	8.52%	9.33%	12.30%	10.57%
超额夏普比	2.22	2.03	1.74	2.26
超额胜率	0.73	0.70	0.75	0.76
多空收益	121.56%	151.50%	114.39%	167.11%
多空波动	0.125	0.142	0.129	0.139
多空最大回撤	16.08%	22.10%	30.56%	27.93%
多空夏普比	2.70	2.62	2.58	2.76
多空胜率	0.76	0.80	0.83	0.82

资料来源：天软科技，长江证券研究所

从模型之间的比较上来看，Extra Trees 的结果很接近融合模型的结果，其分组之间收益率的标准差更大，分组效果更明显，超额收益和多空收益都更高。但是从表 8 中给出的加权合成比例可以看出，Extra Trees 的权重只占很少的比例，其结果的优化贡献来自于 Extra Trees 的部分有限。融合模型得到了比三个模型都更好的结果，这一点尤其体现在收益和夏普比上。说明模型的融合使得整个拟合函数的函数空间更加接近实际函数空间，从一定程度上解决了函数空间的一致性问题。

总结

机器学习本质上是找出使得经验风险泛函最小的建模流程，从建模流程角度建立的模型普适性更强，精确度更高。本文从机器学习建模流程角度出发，介绍了机器学习三大步骤——特征工程、模型训练和模型融合中其中常用的机器学习算法。

特征工程的核心在于找到最优的特征空间，可以细分为特征构建、特征提取和特征选择，除了人为构造特征之外，通常还会同时用到无监督和有监督方法。线性模型捕捉数据之

间存在的线性关系，包括普通线性回归、带惩罚项的线性回归、逻辑回归和支持向量机等，用于处理分类、回归和特征选择等问题。树模型可以很好的捕捉数据之间的非线性关系，以决策树为基础，在集成方法的改进上，衍生出了随机森林、ExtraTrees、Adaboost 和 GBDT 等算法，是当前传统有监督机器学习算法中最为成功的一个分支。深度学习模型主要以多层的网络结构逐步逼近数据中存在的非线性关系，针对多层结构难以训练的问题，很多改进的随机梯度下降方法被广泛应用于深度学习的训练当中，如 Adagrad 、Adadelta、RMSprop 和 Adam 等。模型融合除了投票、加权平均这种简单的方法之外，现阶段比较流行的复杂融合方法有 Blending 和 Stacking，多层次的对模型进行融合。

针对机器学习流程如何建立，本文以多因子选股为例，在全部 A 股中进行选股测试，并得到了以下结论：

1. **以机器学习流程为基础进行多因子选股：**目前多因子选股模型多以单个模型在整个特征空间上的预测构建策略，往往很难保证函数空间上的一致性，而机器学习流程选股体系通过在大的函数空间中选择多个小的函数空间进行合并，得到更为完善的模型，降低函数逼近误差。机器学习流程选股主要通过特征工程确定特征空间，交叉验证的方法确定模型簇中的模型，横向拼接模型的融合方式，得到最终的模型。本文实例重点从特征工程和模型融合两个角度，介绍了特征空间的确定和模型融合的方法。

2. **融合模型的表现略优于单个模型，且在分组投资组合的区分上更明显：**对于全 A 股的等权重策略，融合模型超额年化收益为 25.91%，夏普比为 1.06，信息比为 2.26，月度超额胜率为 0.76，在超额年化收益、超额夏普比、超额胜率以及多空年化收益、多空夏普比、多空胜率上均优于单个模型。分 10 组的投资组合表现中，融合模型的年化收益标准差、超额收益标准差、夏普比标准差、信息比标准差和超额胜率标准差均高于单个模型。

接下来我们将进一步晚上机器学习流程选股体系，在特征空间的含义、函数簇的稳定性和多样性、融合的一致性以及融合目标方面给出更为详细的讨论。

投资评级说明

行业评级	报告发布日后的 12 个月内行业股票指数的涨跌幅度相对同期沪深 300 指数的涨跌幅为基准，投资建议的评级标准为：	
	看　　好：	相对表现优于市场
	中　　性：	相对表现与市场持平
	看　　淡：	相对表现弱于市场
公司评级	报告发布日后的 12 个月内公司的涨跌幅度相对同期沪深 300 指数的涨跌幅为基准，投资建议的评级标准为：	
	买　　入：	相对大盘涨幅大于 10%
	增　　持：	相对大盘涨幅在 5%～10%之间
	中　　性：	相对大盘涨幅在-5%～5%之间
	减　　持：	相对大盘涨幅小于-5%
	无投资评级：	由于我们无法获取必要的资料，或者公司面临无法预见结果的重大不确定性事件，或者其他原因，致使我们无法给出明确的投资评级。

联系我们

上海

浦东新区世纪大道 1198 号世纪汇广场一座 29 层（200122）

武汉

武汉市新华路特 8 号长江证券大厦 11 楼（430015）

北京

西城区金融街 33 号通泰大厦 15 层（100032）

深圳

深圳市福田区福华一路 6 号免税商务大厦 18 楼（518000）

重要声明