通货膨胀率预测建模

2024 年 5 月

目录

第一章　通胀率预测文献综述

1.1　通胀解释变量

哪些变量可以辅助预测通胀率？大量学者从经济含义上进行考量，挖掘通胀率和相关经济变量的关系，借此进行预测。

基于菲利普斯曲线，失业率和产出类/经济活动类指标对通胀率具有解释作用被广泛认可。这类指标包括就业率、失业率、GDP 以及房屋开工率、房地产投资额等。如，Stock et al. (2003)、Bai et al. (2008) 都在研究中引入了此类指标。

大宗商品等供给侧因素同样会影响通胀率。大宗商品价格上涨，将导致投入成本上升，进而影响商品和服务的供应。**Chen et al. (2014)** 基于 GSCI 大宗商品价格分类指数数据，研究了 1983Q1-2010Q3 期间，大宗商品对五个大宗商品出口经济体（澳大利亚、加拿大、智利、新西兰和南非）的通胀率预测效果。发现在使用大宗商品价格预测时将通胀率的结构性断点纳入考虑，效果更好，特别是智利和南非。作者使用 Bai 和 Perron(1998,2003) 的多重断点检验方法，对均值断点进行检验，并在误差修正模型中引入与结构断点日期相关的虚拟项进行研究。**Salisu et al. (2018)** 将反应供给侧冲击的国际石油价格纳入传统的菲利普斯曲线，并将油价分解为正变化和负变化两部分，探究了油价对美国通胀率的非对称影响。基于 1957-2017 年数据，发现模型改善了对美国通胀的预测效果。

大量学者在预测通胀率时纳入了金融变量。这一动机来源于，通胀预期有自我实现功能，对实际通胀有很大影响。而金融变量一定程度上可以体现通胀预期。以利率期限结构为例，费雪效应指出，名义利率是实际利率与预期通胀率之和。预期通胀率上升，一方面会引致名义利率上升，导致债券市场参与者使用长期债券代替短期债券，长期债券利率上升。另一方面也可能会导致预期实际利率下降，使得短期债券需求上升，从而使得短期债券利率上升。

Fama（1975）关于利率对通胀预测作用的研究表明，名义利率的变动反应了通胀预期的变化，而不是实际利率的变化。Mishkin（1990）进一步考察了利率期限结构对未来通胀的影响，明确了预期通胀对长短期债券价格的影响。指出：向下倾斜的收益率曲线，即利率随期限长度降低的收益率曲线，反映了对通货膨胀率下降的预期，而陡峭向上倾斜的收益率曲线则表明对通货膨胀率上升的预期。之后有大量学者将利率与利率期限结构纳入通胀率预测框架。

Stock et al. (1999, 2002) 使用 100 多个经济指标，包括利率和收益率差等，对通胀率进行了预测。**Stock et al. (2003)** 系统回顾了资产价格对通胀率和实际产出增速预测的研究。并对 1959-1999 年期间，38 个季度预测变量，包括短期政府票据利率、短期中期长期政府债券利率、期限价差、股票价格指数、股息价格指数等，和货币总量、实体经济活动以及大宗商品价格指标，在 37 个经合国家的预测效果进行了实证研究。发现不同资产价格的预测效果因国家而异，通过计算平均值/中位数的方法汇总预测变量进行预测，比单个辅助变量预测效果更好。**Forni et al. (2003)** 基于广义动态因子模型

（Forni 等，2000）和近似动态因子模型（SW,2002）研究了在 1987M2-2001M3 期间，447 个变量（118 个金融变量、42 个货币变量、46 个产出变量、139 个价格变量、62 个欧盟委员会调查和价格预期变量和 40 个其他变量）对欧元区 6 个国家的预测效果，发现利率、名义于实际汇率、利差等金融变量的汇总信息有助于预测通胀，但对预测工业生产没有作用。但**Ang et al. (2007)** 基于 1952Q2-2002Q4 美国数据，对 ARIMA 模型、菲利普斯曲线、无套利期限结构模型和通胀预期调查数据预测四种方法的通胀率预测效果比较研究表明，短期利率和期限价差等信息对于预测未来的通胀率的作用较小，不如调查数据。**Hillebrand et al. (2018)**，基于大量债券收益率曲线，构造有监督的因子模型，基于 1970-2010 年月度数据，对美国产出和通胀率进行了预测。

调查数据同样可以体现通胀预期，如**Groen et al. (2013)** 同时使用路透社和密歇根大学消费者调查的通胀预期和利率期限结构衡量通胀预期。**何启志 等 (2017)** 结合中国通胀预期问卷调查系统，测度了公众通胀预期，并将其应用于菲利普斯曲线，构建了时变系数的菲利普斯曲线。

还有学者考虑了实时数据的预测作用，如**郑挺国 等 (2012)** 基于实时数据，研究了基于产出缺口的菲利普斯曲线在我国通胀预测中的适用性。**林玉婷 等 (2023)** 根据 GDP 的实时数据和最终数据，采用 GMM 方法、状态空间模型和卡尔曼滤波方法，估计了后顾式、前瞻式和混合型三种菲利普斯曲线。发现基于最终数据的混合型时变菲利普斯曲线更能拟合我国通胀运行特征。无论是常系数还是时变系数模型，实时数据都无法提高曲线对通胀的预测能力。

Cepni et al. (2024) 主要研究了对于欧洲新兴市场国家的通胀率与全球通胀因子的关系。他们通过使用偏最小二乘法 PLS 以及主成分分析法 PCA 来估计全球通胀因子和局部通胀因子，并构建了因子增广模型对各个国家的通胀进行预测，结果发现，在这些新兴的市场国家中，全球的通胀因子的方差占其他国家的方差一半以上，说明全球通胀因子对国内的通胀率具有重大的影响。

近几年，在线价格数据、搜索数据、新闻文本数据成为热门解释变量。

在线价格数据方面：**Aparicio et al. (2020)** 基于 PriceStats 公司统计的在线价格数据，构建 VAR 模型，预测了澳大利亚、加拿大、法国等 10 个国家一个月后的通胀率。**姜婷凤 等 (2022)** 引入基于互联网在线大数据的居民消费价格指数，采用 LASSO 降维法和混频数据抽样模型 MIDAS，结合 iCPI 总类和 iCPI 八大类指标与传统预测指标，对通胀（CPI 月同比变化率），进行了现时预测，并对不同频率指标的预测效果进行了对比。

网络搜索数据方面：**唐晓彬 等 (2020)** 使用机器学习长短期记忆（LSTM）神经网络模型，基于 2011 年 1 月-2019 年 7 月消费者信心指数相关网络搜索数据，对我国消费者信心指数进行了长期、中期和短期预测研究。**Zhang et al. (2023)** 在卷积神经网络（CNN）框架中融合 MADLMIDAS 模型，建立异步混频卷积神经网络模型，基于 2016 年 1 月至 2019 年 12 月线上消费者价格指数和网络搜索价格指数对中国 CPI 进行了预测。**Borup et al. (2023)** 构建了一种混频的机器学习模型 MFML，基于谷歌每日搜索大量的有关与失业的信息对美国的每周失业保险初次领取人数进行实时预测和追溯，该模型结合了线性模型 LASSO、弹性网络以及非线性模型如机器学习等模型，通过模型结合来进行预测。结果证明，包含谷歌每日数据的模型预测要比不包含每日数据的模型预测精度高。

新闻文本数据方面：**Kalamara et al. (2022)** 基于 1990 年-2019 年三家英国日报的文本信息，使用机器学习方法对英国通胀率进行了预测。**Ellingsen et al. (2022)** 使用 1985M1-2020M4 期间，道琼斯通讯社档案中的 2250 万篇新闻，使用狄利克雷分配（LDA）方法进行文本处理，基于 LASSO、主成分分析、随机森林方法对美国实际 GDP、私人消费支出和私人非住宅投资总额进行了预测，发现预测效果优于 FRED-MD 数据集。**Kelly et al. (2021)** 使用 1990M1-2010M12 华尔街日报新闻数据，使用机器学习方法提取文本信息对美国宏观经济指标进行了预测。**Zheng et al. (2023)** 从 2006.6.1-2021.6.60

期间三份国内财经报纸和百度指数中提取文本信息，结合宏观预测变量，使用岭回归、LASSO 模型和弹性网模型与混频数据采用方法（MIDAS）对中国消费价格指数进行了预测。

1.2 传统计量方法

菲利普斯曲线在通胀预测中有重要地位。这类预测将通胀率作为被解释变量，将产出缺口、通胀率滞后项、货币供应量等作为解释变量，通过回归方法拟合方程，找出通胀率的关键驱动因素后，基于拟合参数进行预测。**Gordon (1981)** 最早将菲利普斯曲线用于通胀率预测。作者运用通胀率三角模型，即通胀惯性、需求冲击和供给冲击三个因素，对美国 20 世纪 80 年代早期的通胀率进行了预测。**Stock et al. (1999)** 将失业率以及衡量总体经济活动的指标的滞后项纳入传统菲利普斯曲线，对 1959 年至 1997 年美国的月度的 PCE 价格平减指数和 CPI 进行了预测，预测步长为一个月。

然而**Atkeson et al. (2001)** 实证发现使用美国 1984Q1-1999Q3 数据进行研究时，发现传统菲利普斯曲线和 Stock 和 Waston(1999) 提出的曲线的预测效果较差，都不如简单的随机游走模型，即不如将预测季度相对当前年份这一季度的价格水平的季度增长率（预测季度的通胀率），表示为前四个季度的通货膨胀率。

此外，**Hubrich (2005)** 使用自回归（AR）模型预测了欧元区 1992-2001 年间的月度消费价格协调总指数（HICP）和其子类成分，发现 AR 模型预测效果优于随机游走模型，Stock 和 Waston(1999) 的菲利普斯曲线和几种 VAR 模型。**Stock et al. (2007)** 开创性地提出了包含随机波动的不可观测成分模型（UC-SV 模型），将实际通胀率分解为了周期成分（暂时性冲击）和不可观测趋势成分，允许暂时性冲击方差时变，并将不可观测趋势成分建模为扰动项方差时变的随机游走过程。使用该方法对美国 1970-2004 年间的实际通胀率进行了预测。发现由于允许周期和趋势成分时变，UC-SV 模型有良好的预测效果。另一种时变参数模型，IMA 模型，即允许 MA 系数的大小与永久扰动方差和暂时扰动方差的比值成反比的模型，也具有良好的预测效果。将趋势成分纳入预测模型打破了以往通胀率稳态水平为零的假设，进一步提高了单变量预测模型的预测能力。**Stock et al. (2009)** 研究指出，在通胀波动率较小、失业率接近 NAIRU（非加速通胀率的失业率）时，AR 等单变量模型预测效果较好，而在通胀波动率较高时，引入失业率变量有助于预测。

不少学者基于 VAR 模型等使用少量变量辅助通胀率预测。例如，**许永洪 等 (2022)** 使用 1995 年 1 月至 2018 年 9 月 GDP 与 CPI 的环比和累计同比的月度季度混频数据，建立了 Sims-Zha 先验分布的 BVAR 模型, 检测了社会消费品零售总额增速、新增固定资产投资完成额、进出口增速和月无风险收益率几个变量的预测效果。发现相比 AR、向量自回归、随机游走模型、GLP 和基于明尼苏达先验分布的 BVAR 模型，该模型在 GDP 上的短期预测效果更精准，在 CPI 的短期预测上预测效果差于 GLP 模型，但是优于其他模型。**张劲帆 等 (2018)** 构建了基于贝叶斯估计的混频向量自回归模型（MF-BVAR）。发现房地产投资对于模型预测能力有重要作用。

使用少数变量进行辅助预测的模型依赖于变量的选择。大量学者使用因子模型，基于高维变量，汇集大量信息进行预测。**Stock et al. (2002)** 基于近似动态因子模型（The approximate dynamic factor model），使用大量的时间序列估计因子后，直接基于因子预测关键宏观变量。以往的研究集中在对协变指标的估计上，作者直接使用因子进行预测，避免了因子模型中固有的识别性 (或旋转性) 困难的问题。作者基于美国 1959M1 至 1998M12 数据，根据 NBER 商业周期分析师开发的“扩散指数”（215 个时间序列）构建因子，预测了 6 个月、12 个月和 24 个月后的宏观变量。经检验，方法优于单变量自回归、小向量自回归 (VAR) 和领先指标模型。**Bai et al. (2008)** 改进了因子预测方法。一是考虑二

次主成分分析，即从水平数据和水平数据的二次项中提取因子，允许预测变量和因子之间呈现非线性关系。二是基于硬阈值和软阈值构建“目标预测器”，确定用于提取因子的预测变量，以减少预测变量噪音。作者基于美国 1960M1 至 2003M12 数据，考虑了 132 个备选预测变量，使用该方法对 1 个月、6 个月、12 个月和 24 个月后的对数 CPI，进行了预测，预测准确度有明显提升。**Inoue et al. (2008)** 基于在相关模型和因子模型框架下的 bagging 变体算法，以一小组实际经济活动指标（约 30 个时间序列）来预测美国的通货膨胀。认为 bagging 方法强于等权重预测、中值预测、ARMA 预测和 AFTER 预测等。和贝叶斯收缩预测、岭回归预测、迭代 Lasso 预测和基于随机额外预测子集的贝叶斯模型平均预测方法预测精度相当。**Hillebrand et al. (2018)** 提出了一种可以监督预测目标的因子预测方法：CF 因子法。不同于直接将预测变量合成因子的传统方法，作者在逐个使用单个预测变量的滞后项拟合被预测变量后，基于所有拟合结果计算被预测变量的 PC 因子或 NS 因子，即 CF 因子，进而基于 CF 因子重新拟合被预测变量的回归曲线，并进行预测。蒙特卡洛模拟和 1970-2010 年美国产出和通胀率的预测结果显示，该方法优于 PC 因子模型和 NS 因子模型。**Hauzenberger et al. (2023)** 使用时变参数模型估计因子与通胀率的关系方程，比较了线性主成分分析、二次主成分分析、核主成分分析以及几种基于机器学习的非线性降维方法对实时通胀率的预测效果。基于美国 1980M1-1999M12 年期间 FRED-MD 数据库中的 104 个预测变量，对一个月度和一个季度的预测效果显示，Autoencoder（AE），即一种基于深度神经网络的算法，估计的因子，在点预测和密度预测方面的效果都较好，二次主成分因子也有较好的预测效果。作者同时考虑了几种方法的动态平均估计，发现预测精度与单个模型的预测效果几乎一致。预测结果还表明，在大衰退时期和 COVID-19 时期，控制估计中的非线性关系对于准确预测十分重要。**Chen et al. (2024)** 构建了一种新型的时变参数的因子模型 TVF，其中因子通过 PCA 方法计算得到，同时通过对系数构建局部光滑的核函数，使得系数可以平滑的随时间转变，并将多个不同的因子模型通过加权平均进行模型结合，使得模型在面对高维变量以及时变环境下保证模型结果的精确。

部分学者较早使用了神经网络方法进行通胀率预测。**Teräsvirta et al. (2005)** 使用 G7 国家 1960M1-2000M12 期间的工业产值、消费者价格指数、狭义货币供应量、3 个月利率、出口量、进口量和失业率，共 47 个月度变量构建了平滑过渡自回归（LSTAR）模型和带有自回归的单层神经网络模型（AR-NN），对所有变量进行了预测。指出 LSTAR 模型预测效果始终优于线性模型。但是使用 AR-NN 模型进行预测时，要谨防模型爆炸和不稳定的风险，这种风险使得长期范围内的预测可能是不可信的。其中，使用贝叶斯正则化指定的神经网络模型比使用一般方法选择的神经网络模型，在长期范围内的预测效果更准确。神经网络模型的激活函数为逻辑函数。**Nakamura (2005)**）使用通胀率的一阶和二阶滞后为输入变量，双曲正切函数作为激活函数，基于交叉验证实施停止程序，使用美国 1978-2002 季度数据，应用神经网络网络方法预测了通胀率。指出停止程序的规范化对于神经网络模型的预测效果有重要作用。认为上述神经网络方法的短期（一两个季度）预测精度优于单一变量自回归预测。但是长期（如四个季度）的预测效果仍然较差。**Choudhary et al. (2012)** 以通胀率的滞后项为输入变量，基于两种 12 层的金字塔结构的人工神经网络模型（ANN）：混合神经网络和内嵌递归行为的动态的混合神经网络，分别预测，并求取均值，对 1991M1-2008M6 期间 28 个经济合作国家的月通胀率进行了预测。发现对于百分之 45 的国家，预测较为准确。

1.3 机器学习算法

目前主流的用于宏观经济变量预测的机器学习方法主要有 Lasso 类方法、完全子集回归、循环神经网络 RNN 及其变种、回归树方法及其变种等方法。

Medeiros et al. (2016) 研究了 adaLASSO 模型在稀疏、高维线性时间序列中的渐进性质，表明该方法适用于误差项和回归变量非高斯分布和条件异方差的情形，也适用于候选变量数量大于观测值数量的情形。并基于 1960M1-2011M12 期间 131 个宏观经济变量，使用 adaLASSO 模型预测了美国月度通胀率，认为该方法优于自回归和传统因子模型等。

在使用神经网络类的机器学习方法预测的研究中，**Barkan et al. (2023)** 通过构造层次循环网络 HRNN，根据英国官方 CPI 数据，对 CPI 的各个组成成分进行预测，其中将 CPI 的成分分为多个层级，第一层次分为衣物穿着、食物、能源消耗的通胀，第二层次为各类商品的总类的通胀情况，如穿着大类中的男士穿着、鞋类等，以及食品大类中的蔬果类的通胀等，以此类推。并将此模型与传统预测模型、机器学习预测方法的预测效果相对比，结果发现 HRNN 模型的预测效果强于其他的预测模型。**Stoneman et al. (2024)** 构建了一个 6 层的 RNN 深度网络来预测美国通胀率，并将此模型的预测结果与美国的专家预测调查 SPF 的通胀核心预测结果相对比，发现该模型的预测精度要高于 SPF 的中位数预测。**Bhardwaj et al. (2022)** 通过对比多种机器学习模型对经济合作发展组织 OECD 的 33 个国家的 GDP 比资本（购买力平价）的预测，发现神经网络模型比其他的机器学习方法的预测能力更为优秀。*Šestanović et al. (2021)* 基于 1999M1-2017M1 欧元区总的劳动成本变化率、M3 扩张率、名义有效汇率变化率衡量的折旧率和通胀滞后变量，使用 JNN（前馈神经网络）和 FNN（Jordan 神经网络）方法，对通胀率进行了预测。并基于一种偏好排序组织方法（PROMETHEE），比较了最优的 JNN 模型 HNN(4,3,1）和最优的 FNN 模型 FNN（4,51）。指出，在同等条件下，需要更少隐藏元的 Jordan 神经网络（JNN）比前馈神经网络（FNNs ）有更好的预测性能，是更适用于预测通胀的神经网络结构。**唐晓彬 等 (2020)** 使用机器学习长短期记忆（LSTM）神经网络模型，基于 2011 年 1 月-2019 年 7 月期间消费者信心指数相关网络搜索数据，对我国消费者信心指数进行了长期、中期和短期预测研究。**张虎 等 (2020)** 在卷积神经网络（CNN）框架中融合 MADL MIDAS 模型，建立异步混频卷积神经网络模型，基于 2016 年 1 月至 2019 年 12 月线上消费者价格指数和网络搜索价格指数数据对中国 CPI 进行了预测。

对于回归树类方法，目前已有许多学者使用该方法预测通胀率或提取通胀因子。其中，**Yoon (2021)** 讨论了机器学习方法在预测实际 GDP 增长的预测能力，其中主要是梯度提升算法和随机森林算法这两种，并与国际货币基金组织以及日本央行预测值作为对比，文章发现这两种方法的预测效果均强于国际货币基金组织和日本央行的预测。**Borup et al. (2023)** 构造了一种目标预测变量的随机森林回归方法，该方法可以解决以往随机森林模型中因存在大量无关或若相关的预测变量导致模型预测能力下降的问题，并通过股市回报率，宏观经济变量以及债券超额回报，工业增长率等实际应用来考察模型的预测效果。**Goulet Coulombe (2020)** 使用随机森林方法构建一种广义时变参数模型预测宏观经济变量，该方法相比于传统的机器学习方法具有一定的线性结构，因此可以对宏观经济运行情况进行解释，同时又具有足够的灵活性。**Clark et al. (2023)** 使用贝叶斯累加回归树方法来预测宏观经济尾部风险，对宏观经济变量进行预测。

部分学者比较了各类机器学习方法的预测效果。**Garcia et al. (2017)** 基于 2003M1-2015M12 期间，59 个宏观预测变量和 34 个与专家预测相关的变量，比较了目标因子模型、LASSO 与自适应 LASSO、随机森林、和完全子集回归 (CSR) 方法，在预测巴西实时通胀率时的准确度。发现比巴西消费者价格指数发布时间提前五天的预测中，LASSO 方法预测结果最好，比指数发布提前一个月零五天的预测，自适应 LASSO 模型表现最好。其他情况下完全子集回归方法表现最优。作者同时基于 bootstrap 重采样进行了密度预测，预测结果与点预测一致。**Medeiros et al. (2021)** 基于 1990 年至 2015 年间，FRED-MD 数据库月度数据，比较了岭回归、Lasso、adaLasso、弹性网、目标因子模型、完全子集回

归、随机森林、混合线性随机森林、bagging 和折刀模型平均方法（JMA）等方法的预测效果，指出，随机森林方法预测效果最好。这种方法有良好表现有两个原因：一是该模型有特定的变量选择方法，二是允许以往的关键性宏观变量和通胀之间呈现非线性关系。

肖争艳 等 (2022) 将机器学习应用在解释维度，将支持向量回归、随机森林、梯度提升、极端梯度提升、K 近邻回归 5 种非线性机器学习方法与 SHAP 值解释性方法结合，对 2001-2019 年期间我国发生的五轮通胀的影响因素进行了识别。发现通胀预期和食品价格上涨是 5 轮通胀的共同驱动因素，消费和投资需求因素影响减弱，成本特别是劳动力成本因素增强，货币政策一直是影响通胀的重要因素。同时发现非线性方法预测效果更优。

1.4 模型改进方法

不少研究表明，变系数模型与模型平均方法可以提高预测精度。例如，**高维清 等 (2023)** 考虑了非平稳特征可能同时存在于 CPI 通胀率的条件均值和波动率序列中的可能性，提出了含结构性断点波动率与时变参数均值随机波动模型 (stochastic volatility in mean model with time-varying parameters and structural breaks in the volatility, SB-TVP-SVM)，基于 2012 年 1 月至 2022 年 7 月的 CPI 月度数据，对中国、美国、德国和韩国通胀率进行了预测。并与 TVP-SVM 和 MA 模型预测效果进行了对比。时变参数均值随机波动模型部分用于解决条件均值不平稳的问题。为建模非平稳波动率，作者假设波动率为分段平稳序列，基于贝叶斯方法，引入“spike-and-slab”先验得到结构性断点位置、波动率和模型参数的估计。并使用 MCMC 方法更新迭代。该方法揭示了改变波动率系统平稳性的重大事件, 刻画了高通货膨胀往往对应高波动率的规律。**Patton et al. (2023)** 基于广义自回归得分 GAS 模型，使用回归树方法，构建了时变参数回归树和回归森林，并将该模型应用到 S&P500 收益率的预测中，结果证明，该模型的预测能力要强于基准的 GAS 模型。

Koop et al. (2012) 基于 1960Q1-2008Q414 个预测指标，使用包含动态模型平均的计量经济学方法，根据广义菲利普斯曲线预测了美国季度通胀。该方法允许系数、预测模型随时间变化。与简单的基准回归和更复杂的方法（例如使用时变系数模型的方法）相比，动态模型平均可以带来显著的预测改进。

1.5 通胀率分布预测

分布预测（密度预测）是目前比较新兴的预测方式。该方法与传统的点预测方法不同，更关注预测变量的分布性质，例如尾部性质，偏度，峰度等。目前常用来进行分布预测的方法主要是贝叶斯计量方法，非参数或半参数方法等。

其中学界主要使用贝叶斯计量方法来对经济变量进行密度估计。**Tallman et al. (2020)** 构建一种混合模型，通过使用相对熵将提前一期预测与长期预测 VAR 模型预测与专家即期和长期预测进行对比。该混合模型相比于非混合模型，预测精度获得很大提升。**Geweke et al. (2010)** 通过对 5 种贝叶斯模型对 S&P500 收益率的预测比较，根据概率区间转换 PIT 指标，发现 CARCH 模型的预测效果弱于 t-GARCH 模型，随机波动模型 SV，蒙特卡洛正态混合模型 MNM，以及层次蒙特卡洛正态混合模型 HMNM，随机波动模型 SV 以及蒙特卡洛混合模型 MNM 要优于其他模型。**Amisano et al. (2007)** 构建一种针对密度预测的检验，根据对照模型的预测分布的有权重的似然得分的比率来比较模型预测效果。**Ravazzolo et al. (2014)** 构建了一种将多种预测模型集成的方法，通过将不同的预测模型的密度根据连续排列概率得分 CRPS 构建权重，将多个模型的预测结果集成，并对美国的个人消费支出通胀

进行预测，结果表明，这种方法的表现要优于基于移动平均加总的密度预测。

目前，除了使用贝叶斯方法进行密度预测外，也有学者通过非参数或半参数方法来构造预测分布。其中**Adrian et al. (2019)** 通过分位数回归，获得从 5% 到 95% 的分位数，通过插值拟合方式，使用偏态 t 分布对不同时期的美国 GDP 增长率的预测分布进行拟合。**Costa et al. (2021)** 在对油价进行预测时，使用对数正态分布作为油价增长率的预测分布。

Kynigakis et al. (2023) 对自回归模型进行增广，使用均值回归和分位数回归方法，基于 1964M12-2019M12 期间 FRED-MD 数据库数据中 126 个预测变量，对美国月度工业生产指数、城镇消费者价格指数和非农居民就业率进行了均值、分位数和密度预测。具体来说，作者首先估计预测变量的自回归模型，之后使用不同的函数将宏观经济变量映射到自回归模型的残差中，以改进自回归模型的拟合效果。并分别对上述过程进行均值和分位数估计，从而得到均值和分位数预测结果。映射函数的构成方法包括 PCA 方法、LASSO 方法、梯度增强方法（GB）、随机森林方法和人工神经网络方法。作者使用分位数分数（QS）评价分位数预测效果。使用加权分位数分数（WQS），调节权重，评价对不同分布区域的预测效果，从而达到评估密度预测效果的目的。研究发现：将经济信息纳入分位数自回归预测，可以提高对左边尾和中间部分的预测效果；以及，基于分位点估计结果的组合预测条件均值，比直接生成点预测更准确。同时探究了哪些变量会导致远离中心的条件分布。

Wochner (2020) 将 Schlosser 等 (2019) 的新型分布森林算法从点预测拓展到密度预测，基于 1960-2019 年美国 FRED-MD 数据库数据对 1 个月、2 个月、3 个月后的通胀率进行了预测。该方法可以考虑均值的局部演变、时变波动性、非线性和分布的厚尾特征。实证结果表明其预测效果优于随机游走、自回归和随机波动模型。

康宁 等 (2016) 使用门限分位数自回归（TQAR），将通胀划分为低通胀和高通胀两个阶段，基于中国 2001 年 1 月至 2014 年 12 月数据，对 CPI 的月度环比增长率进行了条件分位数预测和条件密度预测，并对比了 TAR 和 QAR 方法的预测效果。

参考文献

STOCK J H, W. WATSON M. Forecasting Output and Inflation: The Role of Asset Prices[J/OL]. Journal of Economic Literature, 2003, 41(3): 788-829. DOI: 10.1257/002205103322436197.

BAI J, NG S. Forecasting economic time series using targeted predictors[J/OL]. Journal of Econometrics, 2008, 146(2): 304-317. DOI: 10.1016/j.jeconom.2008.08.010.

CHEN Y C, TURNOVSKY S J, ZIVOT E. Forecasting inflation using commodity price aggregates [J/OL]. Journal of Econometrics, 2014, 183(1): 117-134. DOI: 10.1016/j.jeconom.2014.06.013.

SALISU A A, ISAH K O. Predicting US inflation: Evidence from a new approach[J/OL]. Economic Modelling, 2018, 71: 134-158. DOI: 10.1016/j.econmod.2017.12.008.

STOCK J H, WATSON M W. Forecasting inflation[J/OL]. Journal of Monetary Economics, 1999, 44 (2): 293-335. DOI: 10.1016/S0304-3932(99)00027-6.

STOCK J H, WATSON M W. Macroeconomic Forecasting Using Diffusion Indexes[J/OL]. Journal of Business & Economic Statistics, 2002, 20(2): 147-162. DOI: 10.1198/073500102317351921.

FORNI M, HALLIN M, LIPPI M, et al. Do financial variables help forecasting inflation and real activity in the euro area?[J/OL]. Journal of Monetary Economics, 2003, 50(6): 1243-1255. DOI: 10.1016/S0304-3932(03)00079-5.

ANG A, BEKAERT G, WEI M. Do macro variables, asset markets, or surveys forecast inflation better? [J/OL]. Journal of Monetary Economics, 2007, 54(4): 1163-1212. DOI: 10.1016/j.jmoneco.2006.04.006.

HILLEBRAND E, HUANG H, LEE T H, et al. Using the Entire Yield Curve in Forecasting Output and Inflation[J/OL]. Econometrics, 2018, 6(3): 40. DOI: 10.3390/econometrics6030040.

GROEN J J J, PAAP R, RAVAZZOLO F. Real-Time Inflation Forecasting in a Changing World[J/OL]. Journal of Business & Economic Statistics, 2013, 31(1): 29-44. DOI: 10.1080/07350015.2012.727718.

何启志, 姚梦雨. 中国通胀预期测度及时变系数的菲利普斯曲线[J/OL]. 管理世界, 2017(5): 66-78. DOI: 10.19744/j.cnki.11-1235/f.2017.05.007.

郑挺国, 王霞, 苏娜. 通货膨胀实时预测及菲利普斯曲线的适用性[J]. 经济研究, 2012, 47(3): 88-101.

林玉婷, 孙坚强, 陈创练. 实时数据、动态演变与菲利普斯曲线甄别[J/OL]. 数理统计与管理, 2023, 42 (6): 1113-1126. DOI: 10.13860/j.cnki.sltj.20231030-006.

CEPNI O, CLEMENTS M P. How local is the local inflation factor? evidence from emerging european countries[J]. International Journal of Forecasting, 2024, 40(1): 160-183.

APARICIO D, BERTOLOTTO M I. Forecasting inflation with online prices[J/OL]. International Journal of Forecasting, 2020, 36(2): 232-247. DOI: 10.1016/j.ijforecast.2019.04.018.

姜婷凤, 汤珂, 刘涛雄. 基于在线大数据的通货膨胀“现时”预测[J]. 计量经济学报, 2022, 2(3): 597-619.

唐晓彬, 董曼茹, 张瑞. 基于机器学习 LSTM&US 模型的消费者信心指数预测研究[J/OL]. 统计研究, 2020, 37(7): 104-115. DOI: 10.19343/j.cnki.11-1302/c.2020.07.009.

ZHANG Q, NI H, XU H. Forecasting models for the Chinese macroeconomy in a data-rich environment: Evidence from large dimensional approximate factor models with mixed-frequency data[J/OL]. Accounting & Finance, 2023, 63(1): 719-767. DOI: 10.1111/acfi.13003.

BORUP D, RAPACH D E, SCHÜTTE E C M. Mixed-frequency machine learning: Nowcasting and backcasting weekly initial claims with daily internet search volume data[J]. International Journal of Forecasting, 2023, 39(3): 1122-1144.

KALAMARA E, TURRELL A, REDL C, et al. Making text count: Economic forecasting using newspaper text[J/OL]. Journal of Applied Econometrics, 2022, 37(5): 896-919. DOI: 10.1002/jae.2907.

ELLINGSEN J, LARSEN V H, THORSRUD L A. News media versus FRED-MD for macroeconomic forecasting[J/OL]. Journal of Applied Econometrics, 2022, 37(1): 63-81. DOI: 10.1002/jae.2859.

KELLY B, MANELA A, MOREIRA A. Text Selection[J/OL]. Journal of Business & Economic Statistics, 2021, 39(4): 859-879. DOI: 10.1080/07350015.2021.1947843.

ZHENG T, FAN X, JIN W, et al. Forecasting cpi with multisource data: The value of media and internet information[J]. Journal of Forecasting, 2023.

GORDON R J. Inflation, flexible exchange rates, and the natural rate of unemployment[R]. National Bureau of Economic Research, 1981.

ATKESON A, OHANIAN L E. Are Phillips Curves Useful for Forecasting Inflation?[J/OL]. Quarterly Review, 2001, 25(1). DOI: 10.21034/qr.2511.

HUBRICH K. Forecasting euro area inflation: Does aggregating forecasts by HICP component improve forecast accuracy?[J/OL]. International Journal of Forecasting, 2005, 21(1): 119-136. DOI: 10.1016/j.ijforecast.2004.04.005.

STOCK J H, WATSON M W. Why Has U.S. Inflation Become Harder to Forecast?[J/OL]. Journal of Money, Credit and Banking, 2007, 39(s1): 3-33. DOI: 10.1111/j.1538-4616.2007.00014.x.

STOCK J H, WATSON M W. Phillips Curve Inflation Forecasts[J/OL]. Understanding Inflation and the Implications for Monetary Policy, 2009: 101-186. DOI: 10.7551/mitpress/9780262013635.003.0018.

许永洪, 殷路皓, 朱建平. 中国经济增长与通胀的混频预测——基于 Sims-Zha 先验分布的 BVAR 模型[J/OL]. 数理统计与管理, 2022, 41(2): 225-238. DOI: 10.13860/j.cnki.sltj.20220302-001.

张劲帆, 刚健华, 钱宗鑫, 等. 基于混频向量自回归模型的宏观经济预测[J]. 金融研究, 2018(7): 34-48.

INOUE A, KILIAN L. How Useful Is Bagging in Forecasting Economic Time Series? A Case Study of U.S. Consumer Price Inflation[J/OL]. Journal of the American Statistical Association, 2008, 103 (482): 511-522. DOI: 10.1198/016214507000000473.

HAUZENBERGER N, HUBER F, KLIEBER K. Real-time inflation forecasting using non-linear dimension reduction techniques[J/OL]. International Journal of Forecasting, 2023, 39(2): 901-921. DOI: 10.1016/j.ijforecast.2022.03.002.

CHEN Q, HONG Y, LI H. Time-varying forecast combination for factor-augmented regressions with smooth structural changes[J]. Journal of Econometrics, 2024, 240(1): 105693.

TERÄSVIRTA T, VAN DIJK D, MEDEIROS M C. Linear models, smooth transition autoregressions, and neural networks for forecasting macroeconomic time series: A re-examination[J]. International Journal of Forecasting, 2005, 21(4): 755-774.

NAKAMURA E. Inflation forecasting using a neural network[J/OL]. Economics Letters, 2005, 86(3): 373-378. DOI: 10.1016/j.econlet.2004.09.003.

CHOUDHARY M A, HAIDER A. Neural network models for inflation forecasting: An appraisal[J/OL]. Applied Economics, 2012, 44(20): 2631-2635. DOI: 10.1080/00036846.2011.566190.

MEDEIROS M C, MENDES E F. 1-regularization of high-dimensional time-series models with non-gaussian and heteroskedastic errors[J]. Journal of Econometrics, 2016, 191(1): 255-271.

BARKAN O, BENCHIMOL J, CASPI I, et al. Forecasting cpi inflation components with hierarchical recurrent neural networks[J]. International Journal of Forecasting, 2023, 39(3): 1145-1162.

STONEMAN D, DUCA J V. Using deep (machine) learning to forecast us inflation in the covid-19 era [J]. Journal of Forecasting, 2024.

BHARDWAJ V, BHAVSAR P, PATNAIK D. Forecasting gdp per capita of oecd countries using machine learning and deep learning models[C]//2022 Interdisciplinary Research in Technology and Management (IRTM). IEEE, 2022: 1-6.

ŠESTANOVIĆ T, ARNERIĆ J. Neural network structure identification in inflation forecasting[J/OL]. Journal of Forecasting, 2021, 40(1): 62-79. DOI: 10.1002/for.2698.

张虎, 沈寒蕾, 夏伦. 基于多源异步混频 CPI 数据的预测方法研究[J/OL]. 数量经济技术经济研究, 2020, 37(10): 149-168. DOI: 10.13653/j.cnki.jqte.2020.10.009.

YOON J. Forecasting of real gdp growth using machine learning models: Gradient boosting and random forest approach[J]. Computational Economics, 2021, 57(1): 247-265.

BORUP D, CHRISTENSEN B J, MÜHLBACH N S, et al. Targeting predictors in random forest regression[J]. International Journal of Forecasting, 2023, 39(2): 841-868.

GOULET COULOMBE P. The macroeconomy as a random forest[J]. Journal of Applied Econometrics, 2020.

CLARK T E, HUBER F, KOOP G, et al. Tail forecasting with multivariate bayesian additive regression trees[J]. International Economic Review, 2023, 64(3): 979-1022.

GARCIA M G P, MEDEIROS M C, VASCONCELOS G F R. Real-time inflation forecasting with high-dimensional models: The case of Brazil[J/OL]. International Journal of Forecasting, 2017, 33 (3): 679-693. DOI: 10.1016/j.ijforecast.2017.02.002.

MEDEIROS M C, VASCONCELOS G F R, VEIGA Á, et al. Forecasting Inflation in a Data-Rich Environment: The Benefits of Machine Learning Methods[J/OL]. Journal of Business & Economic Statistics, 2021, 39(1): 98-119. DOI: 10.1080/07350015.2019.1637745.

肖争艳, 陈衍, 陈小亮, 等. 通货膨胀影响因素识别——基于机器学习方法的再检验[J/OL]. 统计研究, 2022, 39(6): 132-147. DOI: 10.19343/j.cnki.11-1302/c.2022.06.009.

高维清, 吴奔, 张波. 通货膨胀率的非平稳时间序列预测和结构性断点诊断: 以中美等国为例[J]. 计量经济学报, 2023, 3(1): 108-127.

PATTON A J, SIMSEK Y. Generalized autoregressive score trees and forests[A]. 2023.

KOOP G, KOROBILIS D. Forecasting Inflation Using Dynamic Model Averaging*[J/OL]. International Economic Review, 2012, 53(3): 867-886. DOI: 10.1111/j.1468-2354.2012.00704.x.

TALLMAN E W, ZAMAN S. Combining survey long-run forecasts and nowcasts with bvar forecasts using relative entropy[J]. International Journal of Forecasting, 2020, 36(2): 373-398.

GEWEKE J, AMISANO G. Comparing and evaluating bayesian predictive distributions of asset returns [J]. International Journal of Forecasting, 2010, 26(2): 216-230.

AMISANO G, GIACOMINI R. Comparing density forecasts via weighted likelihood ratio tests[J]. Journal of Business & Economic Statistics, 2007, 25(2): 177-190.

RAVAZZOLO F, VAHEY S P. Forecast densities for economic aggregates from disaggregate ensembles [J]. Studies in Nonlinear Dynamics & Econometrics, 2014, 18(4): 367-381.

ADRIAN T, BOYARCHENKO N, GIANNONE D. Vulnerable growth[J]. American Economic Review, 2019, 109(4): 1263-1289.

COSTA A B R, FERREIRA P C G, GAGLIANONE W P, et al. Machine learning and oil price point and density forecasting[J]. Energy Economics, 2021, 102: 105494.

KYNIGAKIS I, PANOPOULOU E. Modeling the distribution of key economic indicators in a data-rich environment: New empirical evidence[J]. Available at SSRN 4606867, 2023.

WOCHNER D. Distributional Forests for Short-Term Density Predictions of U.S. Inflation[J/OL]. SSRN Electronic Journal, 2020. DOI: 10.2139/ssrn.3594103.

康宁, 荆科. 门限分位数自回归模型的预测方法及应用[J/OL]. 数量经济技术经济研究, 2016, 33(3): 146-161. DOI: 10.13653/j.cnki.jqte.2016.03.010.

第二章　线性框架下的变量选择

2.1　引言

高斯马尔科夫定理：LS 估计量在所有的线性无偏估计量中有最小均方误差，因为有最小方差。其中均方误差与方差的关系式如下：

$$MSE(\hat{\theta}) = E(\hat{\theta} - \theta)^2 = Var(\hat{\theta}) + [E(\hat{\theta}) - \theta]^2$$

bias 和 viarance 之间平衡的问题本质可以理解为如何减小估计系数均方误的问题。

引发一个问题：既然 LS 是无偏的，为什么要引入 LASSO 等，这类有偏估计。原因如下：（1）LS 估计目的是让拟合曲线尽可能多的经过拟合点，侧重于无偏估计，LS 估计的方差较大。这会导致模型的泛化能力较差，即只对特定样本有效，对随机噪声较为敏感，进而预测效果较差。（2）另一个是，当变量存在多重共线性时，LS 估计结果不稳定，方差更大。（3）为了利用更多的信息，也侧重于使用更多的预测变量，在这种情况下，更需要对变量进行选择。

在实践中，一个小的偏差可以换取更小的方差和更好的预测性能。部分估计虽然是有偏的，方差却较小，总体上能够减小均方误。例如：岭回归、LASSO 等。这类方法对系数施加惩罚项来减小系数的影响力，是一种连续的选择方法，可以平滑模型的预测，减小估计方差。

2.2　岭回归

先来讨论一般情况下的模型估计方法（最小二乘法 OLS 估计）一般线性模型

$$y_t = \beta_0 + x'\beta + \epsilon_t \tag{2.1}$$

则估计量 $\hat{\beta} = (X'X)^{-1}(X'Y)$，其中 $X = (x_1, ..., x_T), Y = (y_1, ..., y_T)'$。当模型中的解释变量存在高度相关性时，$(X'X)^{-1}$ 将会变成病态矩阵，导致模型估计结果对于数据的扰动异常敏感，使得模型的估计结果产生极大的偏误。为了解决这种问题，岭回归就此诞生。

岭回归的一般形式：

$$min\{\sum_i (y_t - \beta x_t)\} \tag{2.2}$$

$$s.t. \|\beta\|^2 \leq m \tag{2.3}$$

其等价形式为：

$$min\{\sum_t (y_t - \beta x_t)^2 + \lambda\|\beta\|^2\} \tag{2.4}$$

对上述目标函数求解之后，可得，

$$\hat{\beta_{ridge}} = (X'X + \lambda I)^{-1}(X'Y) \tag{2.5}$$

当 $X'X$ 可逆时，则岭回归的解如上所示，其中，I 为单位矩阵，$\hat{\beta_{ridge}}$ 中由于增加了一项 λI，使得相比于 OLS 估计的绝对值要小一些。

但是当 $X`X$ 为奇异矩阵时，需要换种方式进行分析。

奇异值分解：对于任意的 $n \times p$ 的矩阵 X 均可分解为以下形式：

$$X = UDV^T \tag{2.6}$$

其中，U 为 $n \times p$ 的列正交矩阵,D 为 $p \times p$ 对角矩阵，且对角元素由大到小排列,V 为 $p \times p$ 正交矩阵。当 X 奇异时，有 D 的某一个或多个对角元素为 0。

根据奇异值分解定理，我们可以知道

$$X'X = VD^2V' \tag{2.7}$$

根据奇异值分解，最小二乘估计量下的拟合值可以写作：

$$X\beta^{ls} = X(X'X)^{-1}Xy \tag{2.8}$$

$$= UU'y \tag{2.9}$$

注：根据最小二乘法的估计方式可以得到这样的结果。

（补充）最小二乘法估计：

$$X = Z\Gamma \tag{2.10}$$

其中 Z 为列正交的 $n \times p$ 矩阵,Γ 为 $p \times p$ 矩阵，我们可以对 Z 进行列单位化，取 G 使得 $Q = ZG^{-1}$ 满足 $QQ^T = I$, 此时可得

$$X = ZG^{-1}G\Gamma \tag{2.11}$$

$$= QR \tag{2.12}$$

因此可以获得最小二乘法估计量以及其拟合值：

$$\hat{\beta_l}s = R^{-1}Q^Ty \tag{2.13}$$

$$\hat{y_{ls}} = QQ^Ty \tag{2.14}$$

因此，利用最小二乘估计量的特性，可以得到如下结论：

$$X\hat{\beta^{ridge}} = X(X'X + \lambda I)^{-1}X'y \tag{2.15}$$

$$= UD(D^2 + \lambda I)^{-1}DU'y \tag{2.16}$$

$$= \sum_{j=1}^{p} u_j \frac{d_j^2}{d_j^2 + \lambda} u_j'y \tag{2.17}$$

根据最小二乘法的性质，我们可以得到：

$$X'X = VD^2V' \tag{2.18}$$

此时令 $z_i = Xv_i$，v_i 为 V 的第 i 列向量，则有

$$Var(z_i) = Var(Xv_i) = \frac{d_i^2}{N} \tag{2.19}$$

其中有 $z_i = Xv_i = u_i d_i$ 为第 i 个主成分，u_i 是标准化后的第 i 个主成分。

$$df(\lambda) = tr(X(X'X + \lambda I)^{-1}X') \tag{2.20}$$

$$= tr(H_\lambda) \tag{2.21}$$

$$= \sum_{j=1}^{p} \frac{d_j^2}{d_j^2 + \lambda} \tag{2.22}$$

其中 $df(\lambda)$ 称为岭回归的有效自由度。当 $df(\lambda) = p$ 时，可以得知 $\lambda = 0$，当 $\lambda \to \infty$，$df(\lambda) = 0$。

2.3 LASSO

LASSO 方法的目标函数：

$$\hat{\beta}^{lasso} = \arg\min_{\beta} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 \tag{2.23}$$

$$subject \quad to \sum_{j=1}^{p} |\beta_j| \le t$$

当 $X^T X$ 是正交矩阵，可以发现对变量系数压缩程度由 t，特别是 t 与 $t_0 = \sum_{j=1}^{p} |\hat{\beta}_j|$ 的相对大小决定，其中 $\hat{\beta}$ 为 LS 估计系数。例如：如果 t 比 t_0 大，那么使用 LASSO 方法会得到 LS 估计值。如果 $t = 1/2t_0$，那么系数会被压缩至 LS 估计值的 50%。可以使用交叉验证方法选择 t 的值。

拉格朗日形式为：

$$\hat{\beta}^{lasso} = \arg\min_{\beta} \{\sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j|\} \tag{2.24}$$

2.3.1 LASSO 数值解特征

上述函数的矩阵表达如下：

$$\hat{\beta}^{lasso} = \arg\min_{\beta} 1/2n||y - X\beta||^2 + \lambda||\beta||_1 \tag{2.25}$$

对上述拉格朗日函数求解次梯度条件如下：

$$-1/n x_j^T (y - X\hat{\beta}) + \lambda s_j = 0, j = 1, \ldots, p \tag{2.26}$$

其中，$s_j = sign(\hat{\beta}_j)$, 当 $\hat{\beta}_j = 0, s_j \in [-1, 1]$。可以发现：

a. 对于有效集，模型中的变量与残差有相同的协方差。$1/n|x_j^T(y - X\hat{\beta})| = \lambda$。

b. 对于不在有效集中的变量，$1/n|x_k^T(y - X\hat{\beta})| \le \lambda$。

c.LASSO 的系数在 λ 的变化范围内是连续和分段线性的。分段节点是有效集变化或系数符号变化时。例如: 假设调节参数从 λ_1 变到一个更小值 λ_2, 且有效集没有发生变化，始终为 $\mathcal{A}$, 令 $s_{\mathcal{A}}$ 为 $\hat{\beta}$ 的符号向量，则：

$$X_{\mathcal{A}}^T(y - X\hat{\beta}(\lambda_1)) = s_{\mathcal{A}}\lambda_1$$

$$X_{\mathcal{A}}^T(y - X\hat{\beta}(\lambda_2)) = s_{\mathcal{A}}\lambda_2$$

则 $\hat{\beta}(\lambda_2) - \hat{\beta}(\lambda_1) = (\lambda_1 - \lambda_2)(X_{\mathcal{A}}^T X_{\mathcal{A}})^{-1} s_{\mathcal{A}}$

d. 使得 $\hat{\beta}(\lambda_{max}) = 0$ 的最小的,λ 的最大值为 $\max_j 1/n|x_j^T y|$

2.4 弹性网

$$\hat{\beta} = \arg\min_{\beta}\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}(\alpha\beta_j^2 + (1-\alpha)|\beta_j|)\} \tag{2.27}$$

优势：可以把部分变量系数压缩至 0，进行变量选择。也可以把相关性较强的系数进行同等程度压缩。并且比 L_q 惩罚项的方法有计算优势。

这类方法的一般形式及其贝叶斯先验解释如下：

$$\hat{\beta} = \arg\min_{\beta}\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|\beta_j|^q\} \tag{2.28}$$

q=2：LASSO，高斯分布。

q=1: 岭回归，Laplace 分布，密度函数为 $1/2\lambda\exp(-|\beta|\lambda)$

q=0：子集选择

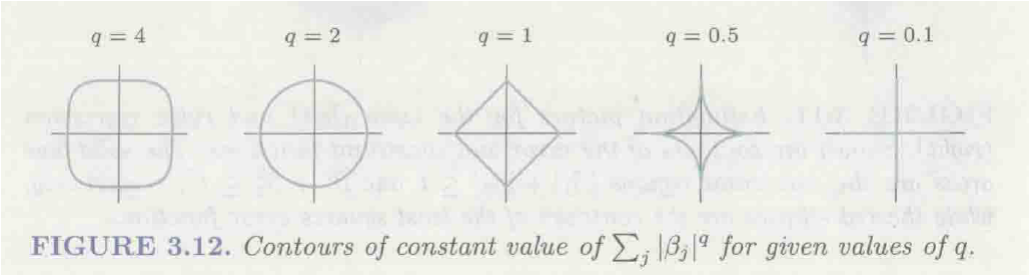


FIGURE 3.12. *Contours of constant value of* $\sum_j |\beta_j|^q$ *for given values of q.*

图 2.1: 包含两个参数时的不同惩罚项示意图

2.5 最小角回归

最小角度回归算法（Least angle regression）每一步沿最小角度方向前进，允许多个变量同时进入模型。基本步骤如下：

（1）挑选出与预测变量相关性最大的变量，得到活跃集 x_k。

（2）将 x_k 乘以一个权重向量 w_k，得到一个与 x_k 中所有变量相关性相同，且为 A_k 的等角向量 u_k，将其作为移动方向。w_k 的构成协方差矩阵有关。

（3）选择一个最小移动距离 γ。即不直接选择向移动方向移动最大距离的 C，而是逐渐增大移动距离，

直至存在一个不在活跃集中的变量的预测相关性达到当前水平即可，该距离即为移动距离 。

（4）使用 γ 乘以这个等角向量/移动方向向量，得到新的移动路径

LARS 算法的正式表达如下：假设 $\hat{\mu}_0 = 0$。

- 设 $\hat{\mu}$ 为当前的估计值，$\hat{c} = X'(y - \hat{\mu})$。
- 定义 K 为对应于具有最大绝对相关性的变量的索引集合：

$$\hat{C} = \max_j |\hat{c}_j| \quad K = \{j : |\hat{c}_j| = \hat{C}\}.$$

- 令 $s_j = \text{sign}(\hat{c}_j)$，并定义对应于 K 的活动矩阵为：

$$X_K = (s_j x_j)_{j \in K}.$$

- 令 $G_K = X_K' X_K$ 和 $A_K = (1_K' G_K^{-1} 1_K)^{-1/2}$，其中 1_K 是大小为 K 的全 1 向量。一个与活动集矩阵 X_K 列数相同的单位等角向量可以定义为：

$$u_K = X_K w_K, \quad w_K = A_K G_K^{-1} 1_K, \quad a_K = X' u_K,$$

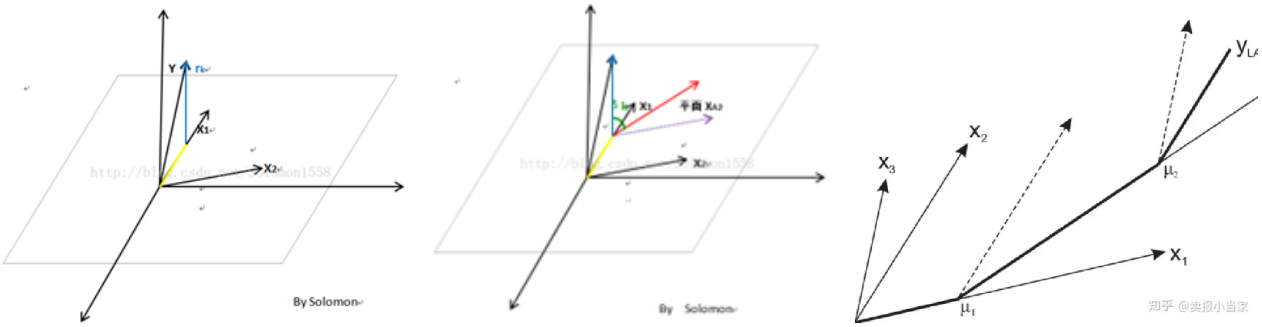
使得 $X_K' u_K = A_K 1_K$。然后 LARS 算法更新 $\hat{\mu}$ 为：

$$\hat{\mu}^{\text{new}} = \hat{\mu} + \hat{\gamma} u_K,$$

其中：

$$\hat{\gamma} = \min_{j \in K^c} \left(\frac{\hat{C} - \hat{c}_j}{A_K - a_j}, \frac{\hat{C} + \hat{c}_j}{A_K + a_j} \right),$$

最小值是在所有正分量上取得。



(a) LARS 图解 1 (b) LARS 图解 2 (c) LARS 图解 3

图 2.2: LARS 的图解说明

2.5.1 使用坐标下降法求解 LASSO-以 OLS 估计问题为例

坐标下降法的基本思想如下：

a. 对于每一个 λ_k, 只解决一个 β_j 的 LASSO 问题，保持其他固定，循环该过程，直至估计值稳定。

b. 从最大的 λ 开始，逐渐减小调节参数，直至有效集增长变缓，对有效集进行猜测和验证。并重新基于该有效集和调节参数值，迭代坐标下降过程，直至收敛。

以加权最小二乘 LASSO 问题的求解为例进行说明，如下：

R 软件包 glmnet 对每个 λ_k 都采用临近牛顿策略。

1. 对解向量 $\hat{\beta}(\lambda_k)$ 计算当前估计的对数似然 L 的加权最小二乘 (二次) 近似; 这产生了一个工作响应和观测权重, 就像常规 GLM 一样。

2. 通过坐标下降在 λ_k 处求解加权最小二乘 LASSO, 使用热启动和有效集迭代。

2.6 adaLASSO

adaLASSO 方法的估计目标函数如下：

$$\hat{\beta} = \arg\min_{\beta}\{\sum_{i=1}^{N}(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda\sum_{j=1}^{p}|w_j\beta_j|\} \tag{2.29}$$

（1）首先，只有在一定的条件下，LASSO 估计才有 oracle 性质（Zou，2006），该条件要求预测变量的协方差矩阵之间需要满足一定的关系，特别是有效预测变量的协方差矩阵和有效与无效预测变量间的协方差矩阵之间需要满足一定的关系。

定理 1（必要条件）：假设 $\lim_n P(A_n = A) = 1$。那么存在某个符号向量 $\mathbf{s} = (s_1, \ldots, s_{p_0})^T$，其中 $s_j = 1$ 或 -1，使得

$$|\mathbf{C}_{21}\mathbf{C}_{11}^{-1}\mathbf{s}| \leq 1.$$

也就是说，如果上述条件不成立，那么 lasso 变量选择是不一致的。

其中各变量含义如下，假设

1. $y_i = \mathbf{x}_i\beta^* + \epsilon_i$，其中 $\epsilon_1, \ldots, \epsilon_n$ 是独立同分布（iid）的随机变量，均值为 0，方差为 σ^2。

2. $\frac{1}{n}\mathbf{X}^T\mathbf{X} \to \mathbf{C}$，其中 $\mathbf{C}$ 是一个正定矩阵。

考虑 lasso 估计，$\hat{\beta}^{(n)}$，

$$\hat{\beta}^{(n)} = \arg\min_{\beta}\left\|\mathbf{y} - \sum_{j=1}^{p}\mathbf{x}_j\beta_j\right\|^2 + \lambda_n\sum_{j=1}^{p}|\beta_j|,$$

其中 λ_n 随 n 变化。令 $\mathcal{A}_n = \{j : \hat{\beta}_j^{(n)} \neq 0\}$。当且仅当 $\lim_n P(\mathcal{A}_n = \mathcal{A}) = 1$ 时，lasso 变量选择是一致的。

不失一般性，假设 $\mathcal{A} = \{1, 2, \ldots, p_0\}$。令

$$\mathbf{C} = \begin{bmatrix}\mathbf{C}_{11} & \mathbf{C}_{12}\\ \mathbf{C}_{21} & \mathbf{C}_{22}\end{bmatrix},$$

其中 $\mathbf{C}_{11}$ 是一个 $p_0 \times p_0$ 的矩阵。

（2）可以证明，adaLASSO 在正确设定权重矩阵和选择调节参数的情况下，有 Oracle 性质，adaLASSO 的权重可设定为如下形式，即可以由一个 $\sqrt{n}$ 一致的估计量构成。该权重设计使得对接近零的系数的惩罚加大，对比较大的系数的惩罚减小。

例如，我们可以使用 $\hat{\beta}(\text{ols})$。选择一个 $\gamma > 0$，并定义权重向量 $\mathbf{w} = \frac{1}{|\hat{\beta}|^\gamma}$。自适应 LASSO 估计 $\hat{\beta}^{*(n)}$ 定义为：

$$\hat{\beta}^{*(n)} = \arg\min_{\beta} \left\| \mathbf{y} - \sum_{j=1}^{p} \mathbf{x}_j \beta_j \right\|^2 + \lambda_n \sum_{j=1}^{p} \mathbf{w}_j |\beta_j|.$$

类似地，令 $\mathcal{A}_n^* = \{j : \hat{\beta}_j^{*(n)} \neq 0\}$。

（3）adaLASSO 的 Oracle 性质完整表述如下：

通过适当选择 λ_n，自适应 LASSO 可以具有 oracle 性质。

定理 2（Oracle 性质）

假设 $\lambda_n/\sqrt{n} \to 0$ 并且 $\lambda_n n^{(\gamma-1)/2} \to \infty$。那么，自适应 LASSO 估计必须满足以下条件：

1. 变量选择的一致性：$\lim_n P(\mathcal{A}_n^* = \mathcal{A}) = 1$
2. $\sqrt{n}(\hat{\beta}_A^{*(n)} - \beta_A^*) \xrightarrow{d} N(0, \sigma^2 \mathbf{C}_{11}^{-1})$

定理 2 表明，ℓ_1 惩罚至少与其他任何“oracle”惩罚一样好。

备注：对自适应 LASSO 来说，$\hat{\beta}$ 不需要是根 $\sqrt{n}$ 一致的。这个条件可以大大放松。假设存在一个序列 $\{a_n\}$，使得 $a_n \to \infty$ 且 $a_n(\hat{\beta} - \beta^*) = O_P(1)$。那么，如果令 $\lambda_n = o(a_n/\sqrt{n})$ 并且 $a_n^\gamma \lambda_n/\sqrt{n} \to \infty$，上述 oracle 性质仍然成立。

（4）adaLASSO 的求解过程如下。

用于自适应 LASSO 的 LARS 算法：

1. 定义 $\mathbf{x}_j^{**} = \mathbf{x}_j/\hat{w}_j$，其中 $j = 1, 2, \ldots, p$。
2. 对所有 λ_n 求解 LASSO 问题：

$$\hat{\beta}^{**} = \arg\min_{\beta} \left\| \mathbf{y} - \sum_{j=1}^{p} \mathbf{x}_j^{**} \beta_j \right\|^2 + \lambda_n \sum_{j=1}^{p} |\beta_j|.$$

3. 输出 $\hat{\beta}_j^{*(n)} = \hat{\beta}_j^{**}/\hat{w}_j$，其中 $j = 1, 2, \ldots, p$。

2.7 SCAD 惩罚项

Fan 和 Li（2001）SCAD 的目标函数设定为：

$$\min_{\beta} \left\{ \frac{1}{2n} \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 + \sum_{j=1}^{p} p_\lambda(\beta_j) \right\} \tag{2.30}$$

其中 y_i 是响应变量，x_i 是预测变量，n 是样本数，β 是系数向量，p 是预测变量的数量，而 $p_\lambda(\beta_j)$ 是对系数 β_j 应用的 SCAD 惩罚函数，定义为：

$$p_\lambda(\beta_j) = \lambda \left\{ I(|\beta_j| \leq \lambda) + \frac{(a\lambda - |\beta_j|)_+}{(a-1)\lambda} I(|\beta_j| > \lambda) \right\} \tag{2.31}$$

对于 $a > 2$ 和 $\lambda > 0$，其中 I 是指示函数，$(x)_+$ 表示 $\max(x, 0)$。在系数的绝对值小于或等于 λ 时起到类似 Lasso 的线性惩罚作用，而当系数的绝对值超过 λ 时，惩罚增长的速度会减慢，并且当系数的绝对值大于 $a\lambda$ 时，额外的增量会停止，使得大系数受到的惩罚有上限。

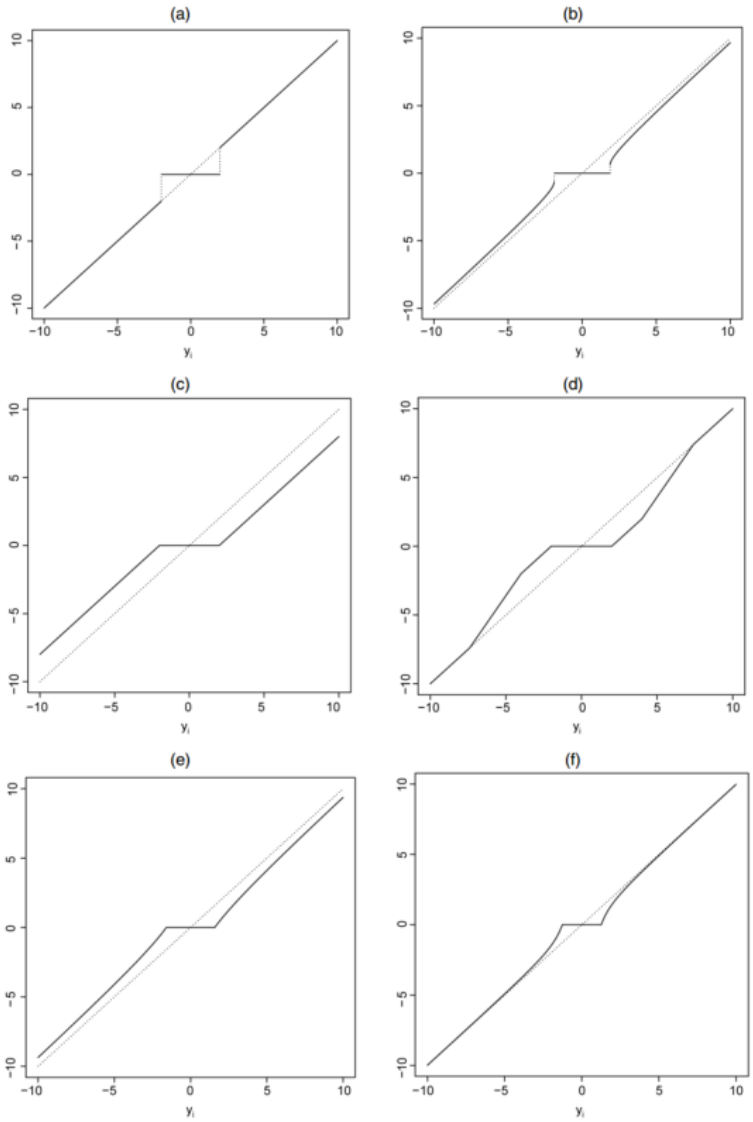


Figure 1. Plot of Thresholding Functions With $\lambda = 2$ for (a) the Hard; (b) Bridge $L_{.5}$; (c) the Lasso; (d) the SCAD; (e) the Adaptive Lasso $\gamma = .5$; and (f) the Adaptive Lasso, $\gamma = 2$.

图 2.3: 不同类型 LASSO 模型比较

第三章　因子模型

May 2024

3.1　动态因子模型整体框架

此处介绍的是具有发散指标的因子模型的一般框架。

其中 y_{t-1} 表示被预测的序列，X_t 表示 N 维的用来预测的时间序列，时间从 $t = 1, ..., T$，y_t 以及 X_t 均值均为 0。此时构建一个具有 r 维共同因子 f_t 的动态因子模型

$$y_{t+1} = \beta(L)f_t + \gamma(L)y_t + \epsilon_{t+1} \tag{3.1}$$

$$X_{it} = \lambda(L)_i f_t + e_{it} \tag{3.2}$$

其中 $e_t = (e_{1t}, ..., e_{Nt})'$，$\lambda(L), \beta(L)$ 是非负权重。假设 $E(\epsilon_{t+1}|f_t, y_t, X_t, f_{t-1}\ , y_{t-1}, X_{t-1}) = 0$，当 $f_t, \beta(L), \gamma(L)$ 已知条件下，则可以根据估计结果来预测 $y_{T+1} = \beta(L)f_T + \gamma(L)y_T$。

$$y_{t+1} = \beta' F_t + \gamma(L)y_t + \epsilon_{t+1} \tag{3.3}$$

$$X_t = \Lambda F_t + e_t \tag{3.4}$$

其中，$F_t = (f_t', ..f_{t-q}')'$ 是 $r \times 1$ 其中 $r \leq (q+1)\overline{r}$，Λ 的第 i 行为 $(\lambda_{i0}, ..., \lambda_{iq})$,$\beta = (\beta_0, ..., \beta_q)'$。则预测 h 期可以写为

$$y_{t+h}^h = \alpha_h + \beta_h(L)F_t + \gamma_h(L)y_t + \epsilon_{t+h}^h \tag{3.5}$$

模型估计方法

其中由于 $\{F_t\}, \alpha_h, \beta_h(L), \gamma_h(L)$ 未知，估计时采用两步法来进行估计。

第一步，用样本数据 $\{X_t\}_{t=1}^T$ 估计发散指标因子的时间序列 $\{\hat{F}_t\}_{t=1}^T$

第二部，根据已知的 f_t 和 y_t 可以对 $\hat{\alpha_h}, \hat{\beta}_h(L), \hat{\gamma_h}(L)$ 进行估计，此时 $y_{T+h}^h = \hat{\alpha_h} + \hat{\beta}_h(L)\hat{F}_T + \hat{\gamma_h}(L)y_T$

目前因子估计方法大多采用的是主成分分析法。原因在于第一，主成分分析方法非常成熟，不仅可以处理高维数据，对于不规整的数据（数据缺失）也可以处理。同时，当 (ϵ, e, F) 满足一定的矩条件以及 Λ 满足渐进秩条件时，它是渐进一阶有效的，这意味着当不管 N, T 以及他们的比值有多大，主成分分析方法估计的结果均足够接近最优的值。

3.2 主成分分析估计方法

当数据不规整或者使用非平衡面板数据时，标准的主成分分析方法难以应用，因此需要对标准的主成分分析方法进行改进。文章使用 EM 算法来迭代求解最优值。

$$V(F,\Lambda)=\sum_{i=1}^{N}\sum_{t=1}^{T}(X_{it}-\lambda_i'F_t)^2 \tag{3.6}$$

其中 λ_i 是 Λ 第 i 行。F 可有特征矩阵计算出。

当数据为非平衡面板数据或者有缺失值时，

$$V^{+}(F,\Lambda)=\sum_{i=1}^{N}\sum_{t=1}^{T}\mathbf{1}_{it}(X_{it}-\lambda_i'F_t)^2 \tag{3.7}$$

其中 $\mathbf{1}_{it}$ 为示性函数，当 X_{it} 未缺失时为 1，否则为 0。

注意到 $V(F,\Lambda)$ 在 X_{it} 服从 $i.i.dN(\lambda_iF_t,1)$ 时是与极大似然函数成比例的，因此可以直接通过该损失函数来使用 EM 算法。

当算法迭代到第 j 代时，令 $\hat{\Lambda}$ 以及 $\hat{F}$ 是 Λ,F 的估计，令

$$Q(X^{+},\hat{F},\hat{\Lambda},F,\Lambda)=E_{\hat{F},\hat{\Lambda}}[V(F,\Lambda)|X^{+}] \tag{3.8}$$

其中 X^{+} 表示所有可观测到数据，$E_{\hat{F},\hat{\Lambda}}[V(F,\Lambda)|X^{+}]$ 是 $V(F,\Lambda)$ 似然对数的期望值。对 Λ 和 F 在第 j 代的估计就是解 $Min_{F,\Lambda}Q(X^{+},\hat{F},\hat{\Lambda},F,\Lambda)$。

$$Q(X^{+},\hat{F},\hat{\Lambda},F,\Lambda)=\sum_{i}\sum_{t}\{E_{\hat{F},\hat{\Lambda}}(X_{it}^{2}|X^{+})+(\lambda_i'F_t)^2-2\hat{X_{it}}(\lambda_i'F_t)\} \tag{3.9}$$

其中 $\hat{X_{it}}=E_{\hat{F},\hat{\Lambda}}(X_{it}|X^{+})$, 因此，可以通过最小化 $\hat{V}(F,\Lambda)=\sum_i\sum_t(\hat{X_i}t-\lambda_i'F_t)^2$ 获取 F,Λ 的值。

下面文章对 X^{+} 进行了详细讨论。为了计算 $\hat{X_{it}}$ 令 $\underline{X_i}=(X_i,...,X_{iT})'$, 令 $\underline{X_i^{+}}$ 为第 i 行的可观测变量，假设 $\underline{X_i^{+}}=A_i\underline{X_i}$，其中 A_i 已知，此时有 $E(\underline{X_i}|X^{+})=E(\underline{X_i}|X_i^{+})=F\lambda_i+A_i'(A_iA_i')^{-}(\underline{X_i^{+}}-A_iF\lambda_i)$，其中 $(A_iA_i')^{-}$ 是 (A_iA_i') 的广义逆。

A. 缺失值

若某些观测值缺失，在第 j 次迭代时，待估计平衡面板的元素，当元素可观测时，$\hat{X_{it}}=X_{it}$，当元素不可观测时有 $\hat{X_{it}}=\hat{\lambda_i'}\hat{F_t}$, 其中 F 的估计是从 $N^{-1}\sum_i\hat{X}_i\hat{X}_i$ 挑选特征值最大的前 r 个特征向量的成分，该估计使用最小二乘法进行估计。

B. 月度数据与季度数据混频——I(0) 序列的存量数据

若序列服从 I(0) 过程，则可以按照 A. 中的处理方式进行处理。

C. 月度数据与季度数据混频——I(0) 序列的流量数据

季度的流量数据是对月度流量数据的平均（加总）。当序列是 I(0) 过程的，则可以按照以下处理方式：

$$X_{it}^{q}=\frac{1}{3}(X_{i,t-2}+X_{i,t-1}+X_{i,t}),t=3,6,9,12,... \tag{3.10}$$

$$\hat{X_{it}}=\lambda_i'F_t+\hat{e_{it}} \tag{3.11}$$

$$\hat{e_{it}}=X_{it}^{q}-\lambda_i'(F_{\tau-2}+F_{\tau-1}+F_{\tau})/3 \tag{3.12}$$

其中，$\tau=3$ 当 $t=1,2,3,$ 以此类推。

D. 月度数据和季度数据混频——I(1) 序列的存量数据

令 X_{it}^1 表示月度数据的一阶差分，对于季度数据每个月度数据的一阶差分 X_{it}, 有

$$X_{it}^q = (X_{i,t-2} + X_{i,t-1} + X_{i,t}) \tag{3.13}$$

对于任意的 $t = 3, 6, 9, ...$, 估计过程与 A. 相同，注意其中 $\hat{X_{it}} = \lambda_i' F_t + \frac{1}{3}\hat{e_{it}}$，且 $\hat{e_{it}} = X_{it}^q - \lambda_i'(F_{\tau-2} + F_{\tau-1} + F_\tau)$

E. 月度数据与季度数据混频——I(1) 序列的流量数据

此时重构 $\hat{X_{it}}$ 之前的例子更加困难，因此此处提供一个总体的框架。季度数据的一阶差分记作 X_{it}^q，对于可观测的数据 $\underline{X_i^+} = (X_{i3}^q, X_{i6}^q, ..., X_{i\tau}^q)$, 将月度数据的一阶差分记作 X_{it}，此时有

$$X_{it}^q = \frac{1}{3}(X_{i,t} + 2X_{i,t-1} + 3X_{i,t-2} + 2X_{i,t-3} + X_{i,t-4}) \tag{3.14}$$

此时可以得到 A_i 的具体形式，则有：

$$\hat{X_i} = F\lambda_i + A_i'(A_i A_i')^-(\underline{X_i^+} - A_i F\lambda_i) \tag{3.15}$$

3.3 EM 算法简介

EM 算法即最大期望算法（Expectation-Maximization algorithm, EM），是一类通过迭代进行极大似然估计（Maximum Likelihood Estimation, MLE）的优化算法，通常作为牛顿迭代法（Newton-Raphson method）的替代用于对包含隐变量（latent variable）或缺失数据（incomplete-data）的概率模型进行参数估计。

当计算似然函数时有无法去除的隐变量时，则需要使用 EM 算法来求解最优值。我们设数据中存在一个隐变量 z。

$$l(\theta) = \sum_{i=1}^{N} log(p(x_i;\theta)) = \sum_{i=1}^{N} log(\sum_z p(x_i, z;\theta)) \tag{3.16}$$

EM 算法的原理为先根据假设的 θ，估计出 z；然后根据 z 利用最大似然函数更新 θ，之后循环迭代直至收敛。因此 EM 算法求解的关键是“根据 θ 求出 z”和“根据 z 求出 θ”，关于后者通过最大化似然函数即可求解，难点主要在于“根据 θ 求出 z”，因此本节主要对该部分的推导与求解过程进行介绍。

$$\begin{aligned}
\sum_i logp(x_i;\theta) &= \sum_i log \sum_{z_i} p(x_i, z_i;\theta) \\
&= \sum_i log \sum_{z_i} Q_i(z_i)\frac{p(x_i, z_i;\theta)}{Q_i(z_i)} \\
&\geq \sum_i \sum_{z_i} Q_i(z_i) log\frac{p(x_i, z_i;\theta)}{Q_i(z_i)}
\end{aligned} \tag{3.17}$$

其中，令 $\sum_{z_i} Q_i(z_i) log\frac{p(x_i,z_i;\theta)}{Q_i(z_i)}$ 为 $\frac{p(x_i,z_i;\theta)}{Q_i(z_i)}$ 的期望，其中 log 函数为凹函数，为使不等式符号满足，得有；

$$\frac{p(x_i, z_i;\theta)}{Q_i(z_i)} = c \tag{3.18}$$

$$Q_i(z_i) = \frac{p(x_i, z_i;\theta)}{c} = \frac{p(x_i, z_i;\theta)}{\sum_z p(x_i, z_i;\theta)} = \frac{p(x_i, z_i;\theta)}{p(x_i;\theta)} = p(z_i|x_i;\theta) \tag{3.19}$$

于是在固定参数 θ 后，通过下界拉升的 $Q_i(z_i)$ 的计算公式就是条件概率公式，在确定 $Q_i(z_i)$ 来调整 θ

$$argmax_\theta \sum_i \sum_{z_i} Q_i(z_i) log(\frac{p(x_i, z_i; \theta)}{Q_i(z_i)}) \tag{3.20}$$

去掉常数部分可得：

$$argmax_\theta \sum_i \sum_{z_i} Q_i(z_i) log(p(x_i, z_i; \theta)) \tag{3.21}$$

因此，只要函数是凹函数，就可以保证算法最终收敛到最小值。当函数在全局中不是全局凹的，EM 算法仅能保证收敛到局部最优值。

3.4 二次因子模型（Bai 和 Ng，2008）

Bai 和 Ng（2008）研究了因子预测方法的两种改进方法。一是，使用二次主成分分析方法，允许因子和预测变量之间呈非线性关系。二是，引入预测变量筛选机制，更高效地选取目标预测变量。

3.4.1 模型主框架-二次主成分

定义 $X_t = (X_{1t}, \ldots, X_{Nt})', \quad i = 1, \ldots, N, \quad t = 1, \ldots, T$ 为预测变量集。y_{T+h} 为 y_t 在 $T+h$ 期的变化值。W_t 为前定变量向量，如常数项和 y_{t+h} 的滞后项。

为允许预测变量与因子之间呈现非线性关系，可以引入非线性函数 $g(\cdot)$，对预测变量进行变换。即：

$$g(X_{it}) = \theta'_{I_t} I_t + e_{it}$$

一种可能性是引入二次形式，具体形式如下：

$$X_t^* = \Phi J_t + e_t$$

当 $X_t^* = \{X_{it}, X_{it}^2\}, N^* = 2N$，为平方主成分分析（SPC，squared principal components）。当引入的变换形式为交乘项 $X_{it}X_{jt}, i \neq j$，即为二次主成分分析（QPC，quadratic principal components）。

注意：这种方法不同于直接基于 X_{it}^2 提取因子，也不同于基于因子的平方进行预测。

3.4.2 预测变量选择

二次项的存在使得有噪音的预测变量的干扰作用被放大，因此需要进行目标预测变量筛选，这是这篇文章的研究重点之一。

变量筛选方法包括“硬阈值”和“软阈值”两种方法。“硬阈值”基于测试过程，直接决定预测变量是否纳入目标预测变量集，主要指对所有备选预测变量逐个进行回归，计算预测变量 t 统计量，基于 t 统计量选择边际影响在 显著性水平下显著的预测变量，作为目标预测变量，构建预测模型。

“软阈值”将预测变量的预测能力排序，基于所设定的软阈值规则，选取预测变量，包括 LASSO、岭回归、最小角度回归等。

3.4.3 预测函数

在该文中，预测函数为

$$\hat{y}_{t+h}^{h} = \hat{\alpha}_0 + \hat{\alpha}_1'(L)z_t + \hat{\beta}_1'(L)\hat{t}_t$$

其中:

$$y_{t+h}^{h} = \frac{1200}{h} \cdot (y_{t+h} - y_t) - 1200 \cdot (y_t - y_{t-1})$$

$$z_t = 1200 \cdot (y_t - y_{t-1}) - 1200 \cdot (y_{t-1} - y_{t-2})$$

第四章　回归树、经典随机森林模型、目标随机森林模型、贝叶斯累加回归树

4.1　回归树模型

4.1.1　回归树

决策树是一种基本的分类与回归方法。决策树由结点 (node) 和有向边 (diredcted edge) 组成。结点有两种类型：内部结点 (internal node) 和叶结点 (leaf node)。内部结点表示一个特征或属性，叶结点表示一个类别或者某个值。

用决策树做分类或回归任务时，从根节点开始，对样本的某一特征进行测试，根据测试结果，将样本分配到其子结点；这时，每一个子节点对应着该特征的一个取值。如此递归地对样本进行测试并分配，直至到达叶结点。

其实，决策树是将空间用超平面进行划分的一种方法，每次分割的时候，都将当前的空间根据特征的取值进行划分，这样使得每一个叶子节点都是在空间中的一个不相交的区域，在进行决策的时候，会根据输入样本每一维特征的值，一步一步往下，最后使得样本落入 N 个区域中的一个（假设有 N 个叶子节点）。

4.1.2　CART 算法

分类与回归树（classification and regression tree，CART）模型由 Breiman 等人在 1984 年提出，是应用广泛的决策树学习方法。CART 同样由特征选择、树的生成及剪枝组成，既可以用于分类也可以用于回归。以下将用于分类与回归的树统称为决策树。CART 是在给定输入随机变量 X 条件下输出随机变量 Y 的条件概率分布的学习方法。CART 假设决策树是二叉树，内部结点特征的取值为“是”和“否”，左分支是取值为“是”的分支，右分支是取值为“否”的分支。这样的决策树等价于递归地二分每个特征，将输入空间即特征空间划分为有限个单元，并在这些单元上确定预测的概率分布，也就是在输入给定的条件下输出的条件概率分布。

CART 算法由以下两步组成：

（1）决策树生成：基于训练数据集生成决策树，生成的决策树要尽量大；

（2）决策树剪枝：用验证数据集对已生成的树进行剪枝并选择最优子树，这时用损失函数最小作为剪枝的标准。

决策树的生成就是递归地构建二叉决策树的过程。对回归树用平方误差最小化准则，对分类树用基尼指数（Gini index）最小化准则，进行特征选择，生成二叉树。

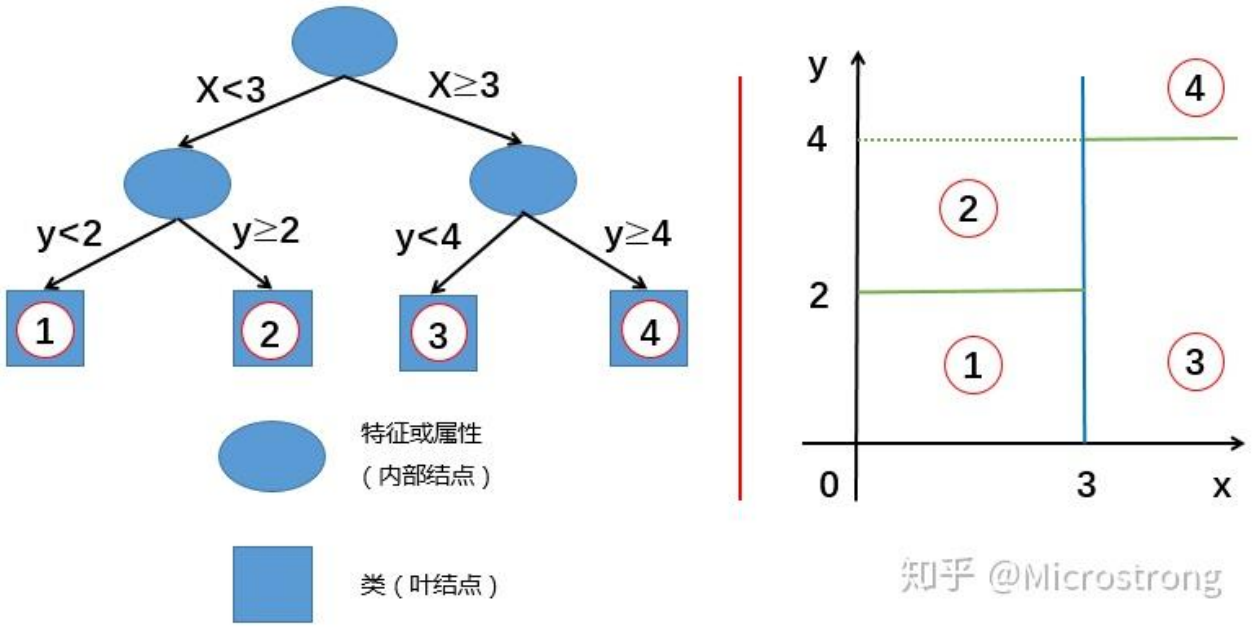


图 4.1: 回归树

CART 生成

假设 X 与 Y 分别为输入和输出变量，并且 Y 是连续变量，给定训练数据集考虑如何生成回归树。

$$D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$$

一个回归树对应着输入空间（即特征空间）的一个划分以及在划分的单元上的输出值。假设已将输入空间划分为 M 个单元 $R_1, R_2, ..., R_M$，并且在每个单元 R_m 上有一个固定的输出值 c_m，于是回归树模型可表示为

$$f(x) = \sum_{m=1}^{M} c_m I(x \in R_m) \tag{4.1}$$

当输入空间的划分确定时，可以用平方误差 $\sum_{x \in R_m}(y_i - f(x_i))^2$ 来表示回归树对于训练数据的预测误差，用平方误差最小的准则求解每个单元上的最优输出值。易知，单元 R_m 上的 c_m 的最优值 $\hat{c_m}$ 是 R_m 上的所有输入实例 x_i 对应的输出 y_i 的均值，即

$$\hat{c}_m = ave(y_i | x_i \in R_m) \tag{4.2}$$

问题是怎样对输入空间进行划分。这里采用启发式的方法，选择第 j 个变量 x(j) 和它取的值 s，作为切分变量（splitting variable）和切分点（splitting point），并定义两个区域：

$$R_1(j, s) = \{x | x^{(i)} \leq s\} \,\&\, R_2 = \{x | x^{(j)} > s\} \tag{4.3}$$

然后寻找最优切分变量 j 和最优切分点 s。具体地，求解

$$min_{j,s}\{min \sum_{x_i \in R_1} (y_i - c_1)^2 + min \sum_{x_i \in R_2} (y_i - c_2)^2\} \tag{4.4}$$

对固定输入变量 j 可以找到最优切分点 s。遍历所有输入变量，找到最优的切分变量 j，构成一个对 (j,s)。依此将输入空间划分为两个区域。接着，对每个区域重复上述划分过程，直到满足停止条件为止。这样就生成一棵回归树。这样的回归树通常称为最小二乘回归树（least squares regression tree），现将算法叙述如下：

算法 5.5（最小二乘回归树生成算法）

输入：训练数据集 D；

输出：回归树 f(x)。

在训练数据集所在的输入空间中，递归地将每个区域划分为两个子区域并决定每个子区域上的输出值，构建二叉决策树：

（1）选择最优切分变量 j 与切分点 s，求解

$$min_{j,s}\{min\sum_{x_i\in R_1}(y_i-c_1)^2+min\sum_{x_i\in R_2}(y_i-c_2)^2\} \tag{4.5}$$

遍历变量 j，对固定的切分变量 j 扫描切分点 s，选择使式（5.21）达到最小值的对（j,s）。

（2）用选定的对（j,s）划分区域并决定相应的输出值：

$$R_1(j,s)=\{x|x^{(j)}\le s\},\ R_2(j,s)=\{x|x^{(j)}>s\}$$

$$\hat{c}_m\frac{1}{N_m}\sum_{x_j\in R_m(j,s)}y_i,\ x\in R_m, m=1,2$$

（3）继续对两个子区域调用步骤（1），（2），直至满足停止条件。

（4）将输入空间划分为 M 个区域 R1,R2,…Rm，生成决策树：

$$f(x)=\sum_{m=1}^{M}\hat{c}_m I(x\in R_m)$$

分类树生成

分类树用基尼指数选择最优特征，同时决定该特征的最优二值切分点。

定义 5.4（基尼指数）分类问题中，假设有 K 个类，样本点属于第 k 类的概率为 pk，则概率分布的基尼指数定义为

$$Gini(p)=\sum_{k=1}^{K}p_k(1-p_k)=1-\sum_{k=1}^{K}p_k^2 \tag{4.6}$$

对于二类分类问题，若样本点属于第 1 个类的概率是 p，则概率分布的基尼指数为

$$Gini(p)=2p(1-p) \tag{4.7}$$

对于给定的样本集合 D，其基尼指数为

$$Gini(D)=1-\sum_{k=1}^{K}\left(\frac{C_k}{|D|}\right)^2 \tag{4.8}$$

这里，Ck 是 D 中属于第 k 类的样本子集，K 是类的个数。

如果样本集合 D 根据特征 A 是否取某一可能值 a 被分割成 D1 和 D2 两部分，即

$$D_1=\{(x,y)\in D|A(x)=a\},\ \&\ D_2=D-D_1$$

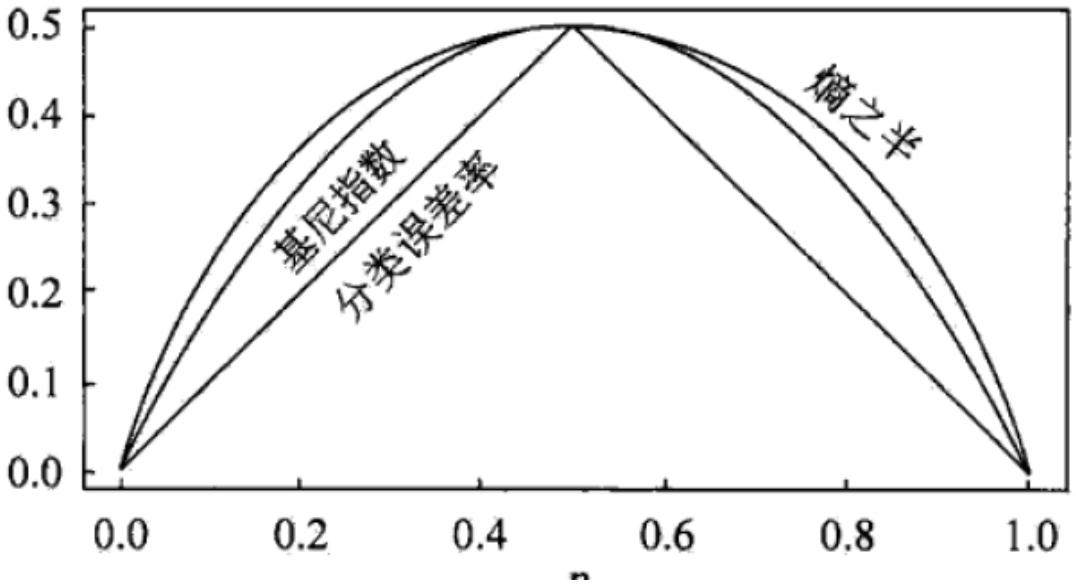


图 4.2: 二类分类问题中基尼指数 Gini(p)、熵（单位比特）之半和分类误差率的关系

则在特征 A 的条件下，集合 D 的基尼指数定义为

$$Gini(D,A)=\frac{|D_1|}{|D|}Gini(D_1)+\frac{|D_2|}{|D|}Gini(D_2) \tag{4.9}$$

基尼指数 Gini(D) 表示集合 D 的不确定性，基尼指数 Gini(D,A) 表示经 A = a 分割后集合 D 的不确定性。基尼指数值越大，样本集合的不确定性也就越大，这一点与熵相似。

图7.1显示二类分类问题中基尼指数 Gini(p)、熵（单位比特）之半 $\frac{1}{2}H(p)$ 和分类误差率的关系。横坐标表示概率 p，纵坐标表示损失。可以看出基尼指数和熵之半的曲线很接近，都可以近似地代表分类误差率。

算法 5.6（CART 生成算法）

输入：训练数据集 D，停止计算的条件；

输出：CART 决策树。

根据训练数据集，从根结点开始，递归地对每个结点进行以下操作，构建二叉决策树：

（1）设结点的训练数据集为 D，计算现有特征对该数据集的基尼指数。此时，对每一个特征 A，对其可能取的每个值 a，根据样本点对 A = a 的测试为“是”或“否”将 D 分割成 D1 和 D2 两部分，利用式（5.25）计算 A = a 时的基尼指数。

（2）在所有可能的特征 A 以及它们所有可能的切分点 a 中，选择基尼指数最小的特征及其对应的切分点作为最优特征与最优切分点。依最优特征与最优切分点，从现结点生成两个子结点，将训练数据集依特征分配到两个子结点中去。

（3）对两个子结点递归地调用（1），（2），直至满足停止条件。

（4）生成 CART 决策树。

算法停止计算的条件是结点中的样本个数小于预定阈值，或样本集的基尼指数小于预定阈值（样本基本属于同一类），或者没有更多特征。

CART 剪枝

CART 剪枝算法从“完全生长”的决策树的底端剪去一些子树，使决策树变小（模型变简单），从而能够对未知数据有更准确的预测。CART 剪枝算法由两步组成：首先从生成算法产生的决策树 T0 底端开始不断剪枝，直到 T0 的根结点，形成一个子树序列 $\{T_0\ T_1, ..., T_n\}$；然后通过交叉验证法在独立的验证数据集上对子树序列进行测试，从中选择最优子树。

1．剪枝，形成一个子树序列在剪枝过程中，计算子树的损失函数：

$$C_a(T) = C(T) + a|T| \tag{4.10}$$

其中，T 为任意子树，C(T) 为对训练数据的预测误差（如基尼指数），|T| 为子树的叶结点个数，$a \geq 0$ 为参数，$C_a(T)$ 为参数是 a 时的子树 T 的整体损失。参数 a 权衡训练数据的拟合程度与模型的复杂度。

对固定的 a，一定存在使损失函数 $C_a(T)$ 最小的子树，将其表示为 T_a。T_a 在损失函数 Ca(T) 最小的意义下是最优的。容易验证这样的最优子树是唯一的。当 a 大的时候，最优子树 T_a 偏小；当 a 小的时候，最优子树 Ta 偏大。极端情况，当 a = 0 时，整体树是最优的。当 $a \to \infty$ 时，根结点组成的单结点树是最优的。

Breiman 等人证明：可以用递归的方法对树进行剪枝。将 a 从小增大，$0 = a_0 < a_1 < ... < a_n < +\infty$，产生一系列的区间 $[a_i, a_{i+1}), i = 0, 1, ..., n$ 剪枝得到的子树序列对应着区间 $a\ [a_i, a_{i+1}), i = 0, 1, ..., n$ 的最优子树序列 $\{T_0\ T_1, ..., T_n\}$，序列中的子树是嵌套的。

具体地，从整体树 T_0 开始剪枝。对 T_0 的任意内部结点 t，以 t 为单结点树的损失函数是

$$C_a(t) = C(t) + a \tag{4.11}$$

以 t 为根结点的子树 T_t 的损失函数是

$$C_a(T) = C(T_t) + a|T_t| \tag{4.12}$$

当 a = 0 及 a 充分小时，有不等式

$$C_a(T_t) < C_a(t) \tag{4.13}$$

当 a 增大时，在某一 a 有

$$C_a(T_t) = C_a(t) \tag{4.14}$$

当 a 再增大时，不等式（16）反向。只要 $a = \frac{C(t)-C(T_t)}{|T_t|-1}$，$T_t$ 与 t 有相同的损失函数值，而 t 的结点少，因此 t 比 T_t 更可取，对 T_t 进行剪枝。

为此，对 T_0 中每一内部结点 t，计算

$$g(t) = \frac{C(t) - C(T_t)}{|T_t| - 1} \tag{4.15}$$

它表示剪枝后整体损失函数减少的程度。在 T_0 中剪去 g(t) 最小的 T_t，将得到的子树作为 T_1，同时将最小的 g(t) 设为 a_1。T_1 为区间 $[a_1, a_2)$ 的最优子树。

如此剪枝下去，直至得到根结点。在这一过程中，不断地增加 a 的值，产生新的区间。

2．在剪枝得到的子树序列 $T_0, T_1, ..., T_n$ 中通过交叉验证选取最优子树 T_a

具体地，利用独立的验证数据集，测试子树序列 $T_0, T_1, ..., T_n$ 中各棵子树的平方误差或基尼指数。平方误差或基尼指数最小的决策树被认为是最优的决策树。在子树序列中，每棵子树 $T_1, T_2, ..., T_n$ 都对应于一个参数 $a_1, a_2, ..., a_n$。所以，当最优子树 T_k 确定时，对应的 a_k 也确定了，即得到最优决策树 T_a。

现在写出 CART 剪枝算法。

算法 5.7（CART 剪枝算法）

输入：CART 算法生成的决策树 T_0；

输出：最优决策树 T_a。

（1）设 k = 0，$T = T_0$。

（2）设 $a = +\infty$ 。

（3）自下而上地对各内部结点 t 计算 $C(T_t)$，$|T_t|$ 以及

$$g(t) = \frac{C(t) - C(T_t)}{|T_t| - 1}$$

$$a = min(a, g(t))$$

这里，T_t 表示以 t 为根结点的子树，$C(T_t)$ 是对训练数据的预测误差，$|T_t|$ 是 T_t 的叶结点个数。

（4）自上而下地访问内部结点 t，如果有 g(t) = a，进行剪枝，并对叶结点 t 以多数表决法决定其类，得到树 T。

（5）设 $k = k + 1, a_k = a\ T_k = T$。

（6）如果 T 不是由根结点单独构成的树，则回到步骤（4）。（7）采用交叉验证法在子树序列 $T_0, T_1, ..., T_n$ 中选取最优子树 T_a。

4.2 随机森林模型

随机森林建立一个大量的“de correlated trees”，然后将这些模型平均。运用了 bagging 算法的思想。最终集成的模型的偏误和每个单独的树模型的偏误是一致的，是 i.d.(同分布，但不一定是独立分布）的。但是误差会降低。

4.2.1 随机森林模型算法

随机森林算法：

1. 假设树的数量/重复训练的次数为 B, 对于 $b = 1$ 到 B

(a) 从训练集中**随机**提取容量为 N 的 boostraped 样本 Z^*

(b) 在每一个节点，重复以下过程，构造一个随机森林树 T_b，直至使节点包含的样本数达到最小值 n_{min}。

- **随机**从 p 个特征变量选取 m 个特征变量。
- 从 m 个变量中选取**最佳的变量/分裂点**（split-point）
- 将每个节点拆分为两个子节点

2. 输出形成的树 $\{T_b\}_1^B$。

计算新的点 x 处的预测值

回归:$\hat{f}_{rf}^{B}(x) = 1/B \sum_{b=1}^{B} T_b(x)$

分类：$\hat{C}_{rf}^{B}(x) = majority \quad vote\{\hat{C}_b(x)\}_1^B$

---

m 决定了降维的力度，典型的设置标准是，在回归树中 $m = \lfloor p/3 \rfloor$，在分类树中，$m = \lfloor \sqrt{p} \rfloor$ 。n_{min} 保证每个叶子下的样本数不超过 n_{min}。

确定节点分裂参数方法的简要解释：将一棵树表示为

$$T(x;\theta) = \sum_{j=1}^{J} \gamma_j I(x \in R_j)$$

其中参数 $\theta = \{R_j, \gamma_j\}$。$J$ 通常被视为一个元参数。最终目标是通过最小化经验风险

$$\hat{\theta} = \arg\min_{\theta} \sum_{i=1}^{J} L(y_i, T(x_i;\theta))$$

来找到这些参数。这是一个复杂的组合优化问题，通常寻求近似的次优解。优化问题分为两部分：

1. 给定 R_j，寻找 γ_j：鉴于 R_j，通常设定 $\hat{\gamma}_j = \bar{y}_j$，即落在区域 R_j 中的 y 的平均值。对于分类树，$\hat{\gamma}_j$ 是落在区域 R_j 中的观测的模型类别。

2. 寻找 R_j：这部分较为复杂，我们通常找到近似解。请注意，找到 R_j 也涉及估计 γ_j。一个典型的策略是使用贪婪的自上向下递归划分算法来找到 R_j 有时还需要使用一个更平滑、更方便的标准来近似优化 R_j。

$$\tilde{\Theta} = \arg\min_{\Theta} \sum_{i=1}^{N} \tilde{L}(y_i, T(x_i;\Theta)).$$

N 是分裂节点下的所有样本个数，接着给定 $\hat{R}_j = \tilde{R}_j$，更精确地使用原始标准来估计 γ_j。

总之，在回归树的随机森林模型中，最终的预测值表示为：

$$\hat{f}_{rf}^{B}(x) = 1/B \sum_{b=1}^{B} T(x;\Theta(b)) \tag{4.16}$$

θ_b 表征了随机森林中第 b 棵树，包括分割变量、每个节点的分割点和终端节点的值。

说明 1：样本间是同分布的，因此在理论上，样本拟合值的均值与每次拟合值相同。

说明 2：B 个同分布，方差为 σ^2，但不独立，且两两相关性为 ρ 的样本均值的方差为

$$\rho\sigma^2 + (1-\rho)/B\rho^2 \tag{4.17}$$

因此，当 B 足够大，减少样本间相关性越小，可以减少变量均值方差。随机森林通过 bootstrap 抽样和随机选择分裂变量减小不同树模型间的相关性。

说明 3：直观上，减少 m 将会降低集合中任意两棵树之间的相关性，并因此减少平均值的方差。

Proof：假设 $X_1, X_2, \ldots, X_B$ 是同分布的变量，方差 $\mathrm{Var}(X_i) = \sigma^2$ ，任意两个变量 $i \neq j$ 的相关性

为 ρ，则任意两个变量间的协方差为 $\mathrm{Cov}(X_i, X_j) = \rho\sigma^2$.，B 个变量的均值的方差为:

$$\begin{aligned}
\mathrm{Var}\left(\frac{1}{B}\sum_{i=1}^{B} X_i\right) &= \frac{1}{B^2}\mathrm{Var}\left(\sum_{i=1}^{B} X_i\right) \\
&= \frac{1}{B^2}\left(\sum_{i=1}^{B}\mathrm{Var}(X_i) + \sum_{i\neq j}\mathrm{Cov}(X_i, X_j)\right) \\
&= \frac{1}{B^2}\left(B\sigma^2 + B(B-1)\rho\sigma^2\right) \\
&= \frac{1}{B}\sigma^2 + \frac{B-1}{B}\rho\sigma^2 \\
&= \rho\sigma^2 + \frac{1-\rho}{B}\sigma^2.
\end{aligned}$$

4.2.2 随机森林的细节

袋外样本（out-of-bag samples）

袋外样本指不在抽样样本中的样本。在随机森林中，可以计算这些样本基于当前树模型的预测误差，即 OOB 损失。该估计约等于 N 折交叉验证误差，因此在随机森林中，无需再进行交叉验证或选择单独的测试集进行测试，就可以获得预测/测试误差的无偏估计量。即：

$$err_{OOB} = 1/n\sum_{i=1}^{n} L[y_i, \hat{f}_{rf}^{(i)}(x_i)] \tag{4.18}$$

$$\hat{f}_{rf}^{(i)}(x_i) = 1/B_i\sum_{b\in\Theta_i} T_b(x_i) \tag{4.19}$$

其中 (x_i, y_i) 为不在 bootstrap 样本中的点,B_i 为不包含该样本的抽样集合的总个数，即不包含该样本的树模型的总个数，Θ_i 为不包含该样本的抽样序数的集合，n 为样本总数，或不在所有抽样样本中的样本点的总数。

注：每个样本不被取到的概率为 $((N-1)/N)^N$，当样本量等于 1000，这个概率约等于 %36.8，因此对于每棵树，大约 1/3 样本没有选到。因此每个样本大约不在 1/3 的树中。

算法详细说明如下：

- 对于每个样本 x_i，找出没有使用 x_i 作为训练数据的树的集合，记为 Θ_i。
- 使用集合 Θ_i 中的每棵树对样本 x_i 进行预测，并取这些预测的平均值（对于回归问题）作为样本 x_i 的 OOB 预测。
- 计算每个样本 x_i 的真实值 y_i 与 OOB 预测值之间的误差。
- 将所有样本的误差平均（或求和）得到 OOB 误差。

变量重要性图

方法 1：计算 split-criterion 的改进值

方法 2：使用 OOB 样本计算第 j 个样本对预测准确度的改进度。具体来说，记录 OOB 样本在第 b_i 个树下的预测误差，然后在保持其他变量不变的情况下，随机置换第 j 个变量在不同样本的值，重新计算预测精度，并计算预测精度的下降程度。对每一个树模型都重复该过程，计算第 j 个变量导致的平均下降精度。基于每个变量的计算结果，衡量变量重要性。

相比较第一种，第二种种方法计算出的样本重要性分布较为平均。

邻近性图 (Proximity Plots)

随机森林的变量邻近性（Proximity）衡量随机森林中两个样本在同一决策树叶节点（即决策树的终点）上出现的频率。如果两个样本在同一决策树的叶节点上结束得更频繁，它们的邻近性就更高。这项功能被认为是随机森林的“亮点”。

步骤如下：

- 训练随机森林模型。
- 对于每个决策树，在每个叶节点上，记录结束于该节点的所有训练样本。
- 初始化一个邻近性矩阵，其大小为训练样本数乘以自身，矩阵的初始值为零。
- 对于每个决策树中的每个叶节点，更新邻近性矩阵。对于结束于同一叶节点的样本对 i 与 j，将邻近性矩阵中的元素 $Proximity(i,j)$ 和 $Proximity(j,i)$ 的值增加 1。
- 最后，将邻近性矩阵的每个元素除以决策树的数量，以获得平均邻近性。

$$\text{Proximity}(i,j) = \text{Proximity}(j,i) = \frac{1}{|\text{Trees}|} \sum_{t \in \text{Trees}} [\text{leaf}(i,t) = \text{leaf}(j,t)] \tag{4.20}$$

$leaf(i,t), leaf(j,t)$ 表示样本 i 与样本 j 在决策树 t 中最终归于的叶节点。

使用多维尺度分析方法等可以将多维的邻近性矩阵降维至 2 维或 3 维，进而将其映射在坐标系中，进行可视性观测。测度变量间的邻近性可以用于数据降维、异常值检测、缺失值填补等。

随机森林和过拟合

1）随机森林的鲁棒性

在有效变量较少时，在每个分裂点，有效变量被选择的机会会变小，随机森林表现会变差，但不会太差。并且，对于存在较多噪声变量的情况，随机森林方法仍然有强健的效果，因为即使在噪声变量较多的情况下，有效变量个数增加带来的效果增益仍然较高。整体上有较好的性能。

2）随机森林的过拟合

随机森林一般不会过拟合，增加 B 也不会过拟合。完全生长的树可能会导致过度拟合和不必要的方差。适度控制树的深度可以避免这一点。

4.2.3 随机森林的分析

分析模型性能需要分析两个方面，一是模型的偏误，也就是使用模型进行多次拟合，拟合效果的均值与真实值的差距，衡量拟合效果的准确性。二是模型的方差，也就是使用模型进行多次拟合，拟合效果的方差，衡量拟合的稳定性。此外，随机森林模型的核心是去相关，因此分析随机森林的去相关效应和方差与偏误的特征。

方差和去相关效应

随机森林回归树的极限形式为：

$$\hat{f}_{rf}(x) = \mathbb{E}_{\Theta|Z}[T(x;\Theta(Z))] \tag{4.21}$$

则对于点 x 的预测结果的方差为：

$$\mathrm{Var}\hat{f}_{rf}(x) = \rho(x)\sigma^2(x) \tag{4.22}$$

其中，$\rho(x)$ 是平均过程中使用的任意一对树的抽样相关性为

$$\rho(x) = \mathrm{corr}[T(x;\Theta_1(Z)), T(x;\Theta_2(Z))] \tag{4.23}$$

$\Theta_1(Z), \Theta_2(Z)$ 是从随机样本 Z 生成的森林树中随机抽取的一对树。

$\sigma^2(x)$ 是任意一个随机树的抽样方差，**对相同的点使用不同的训练数据集来训练模型，预测会有多少变异**。衡量聚合中用到的所有单个模型的点预测的不确定性。

$$\sigma^2(x) = \mathrm{Var}[T(x;\Theta(Z))] \tag{4.24}$$

对于相关性，如图7.2，**特征变量个数越多，不同树模型的相关性越大**[1]。

该方差可以分解为训练样本 Z 的组间方差和组内方差：

$$\mathrm{Var}_{\Theta,Z}[T(x;\Theta(Z))] = \mathrm{Var}_Z[\mathbb{E}_{\Theta|Z}[T(x;\Theta(Z))]] + \mathbb{E}_Z[\mathrm{Var}_{\Theta|Z}[T(x;\Theta(Z))]] \tag{4.25}$$

$$\text{Total Variance} = \mathrm{Var}_Z[\hat{f}_{rf}(x)] + \text{within-}Z\text{ Variance}$$

组间方差 $\mathrm{Var}_Z[\mathbb{E}_{\Theta|Z}[T(x;\Theta(Z))]]$ 表示在创建随机森林聚合模型过程中由于抽样不同导致的方差，也是随机森林模型整体的方差。可以理解为对于每个训练样本，将其基于不同特征变量预测得到预测均值看作该样本预测值时，不同样本预测值的方差。**组间方差随着 m 的减小而减小**。一种理解是 m 减小，分裂深度减小，不同样本的分裂差异减小。

组内方差 $\mathbb{E}_Z[\mathrm{Var}_{\Theta|Z}[T(x;\Theta(Z))]]$ 表示构建随机森林时固有的随机化导致的方差，也就是即使是相同的样本，也会由于分裂时的特征变量不同，导致预测结果不同，产生的预测方差。组内方差随着 m 减小而增大。一种理解是，随着 m 的减小，样本可利用信息减小，分裂结果的方差增大。重点考虑特征变量抽样。

由图5.21左侧，**单个树模型的预测方差随 m 增大的变化程度较小**。由图5.21右侧，由于降低了不同树模型间的相关性，聚合模型的方差比单个模型的方差小。

[1]图7.2是基于模拟方程 $Y = \frac{1}{\sqrt{50}}\sum_{j=1}^{50} X_j + \epsilon$（其中 X_j, ϵ 独立且服从高斯分布），生成的 500 个容量为 100 的训练样本集，和一个容量为 600 的测试集得到的结果，相关性代表了 600 个测试点的相关性统计情况，单个模型方差代表了 600 个点的平均预测方差。注意：这部分提到的相关性是理论上的相关性，是从总体反复抽样得到的不同样本和其产生的分裂特征估计的模型间的相关性，不是一次计算中，不同 bootstrap 模型间的相关性

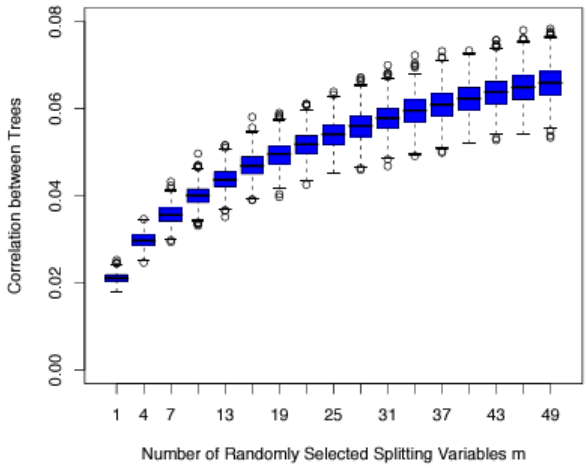


图 4.3: 随机森林回归算法的树间相关性与特征变量个数关系图

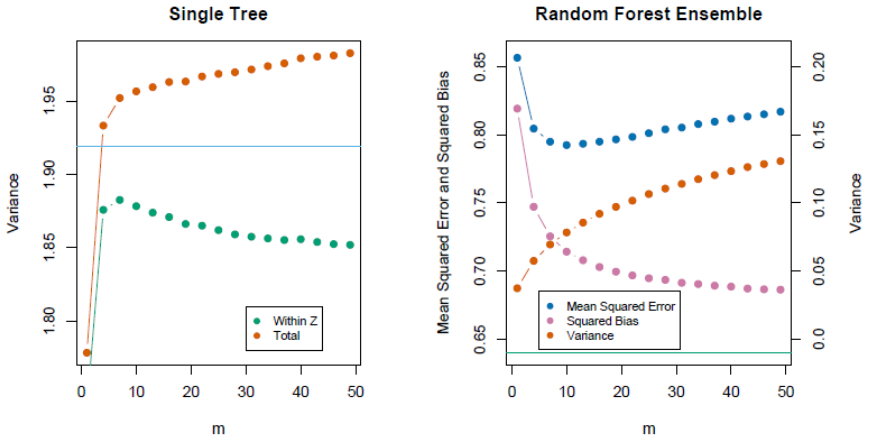


FIGURE 15.10. *Simulation results. The left panel shows the average variance of a single random forest tree, as a function of m. "Within* **Z***" refers to the average within-sample contribution to the variance, resulting from the bootstrap sampling and split-variable sampling (15.9). "Total" includes the sampling variability of* **Z***. The horizontal line is the average variance of a single fully grown tree (without bootstrap sampling). The right panel shows the average mean-squared error, squared bias and variance of the ensemble, as a function of m. Note that the variance axis is on the right (same scale, different level). The horizontal line is the average squared-bias of a fully grown tree.*

图 4.4: 随机森林模型方差与偏误解析图

PS1：由方差分解公式，一个随机变量的总方差可以分解为给定另一个随机变量时的条件方差的期望，加上这个条件期望本身的方差，即：$\mathrm{Var}(X) = \mathrm{E}[\mathrm{Var}(X|Z)] + \mathrm{Var}[\mathrm{E}(X|Z)]$

proof：

$\mathrm{Var}(X) = \mathrm{E}[(X - \mathrm{E}[X])^2]$

$= \mathrm{E}[(X - \mathrm{E}[X|Z] + \mathrm{E}[X|Z] - \mathrm{E}[X])^2]$

$= \mathrm{E}[(X - \mathrm{E}[X|Z])^2] + \mathrm{E}[(\mathrm{E}[X|Z] - \mathrm{E}[X])^2] + 2\mathrm{E}[(X - \mathrm{E}[X|Z])(\mathrm{E}[X|Z] - \mathrm{E}[X])]$

$= \mathrm{E}[(X - \mathrm{E}[X|Z])^2] + \mathrm{E}[(\mathrm{E}[X|Z] - \mathrm{E}[X])^2]$

$= \mathrm{Var}[\mathrm{E}(X|Z)] + \mathrm{E}[\mathrm{Var}(X|Z)]$

注意，$\mathrm{E}[(X - \mathrm{E}[X \mid Z])(\mathrm{E}[X \mid Z] - \mathrm{E}[X])]$ 关于 Z 的条件期望为 0，因为 $\mathrm{E}[X \mid Z] - \mathrm{E}[X]$ 关于 Z 的期望值是 0。

PS2：此外，可以证明成对随机森林树在点 x 的采样相关性由下式给出：

$$\rho(x) = \frac{\mathrm{Var}_Z[\mathbb{E}_{\Theta|Z}[T(x;\Theta(Z))]]}{\mathrm{Var}_Z[\mathbb{E}_{\Theta|Z}[T(x;\Theta(Z))]] + \mathbb{E}_Z[\mathrm{Var}_{\Theta|z}[T(x;\Theta(Z))]]} \tag{4.26}$$

证明过程如下：

$$\begin{aligned}
\rho &= \mathrm{corr}\left[T\left(x;\Theta_1(\mathbf{Z})\right), T\left(x;\Theta_2(\mathbf{Z})\right)\right] \\
&= \frac{\mathrm{Cov}\left(T\left(x;\Theta_1(\mathbf{Z})\right), T\left(x;\Theta_2(\mathbf{Z})\right)\right)}{\sqrt{\mathrm{Var}\left(T\left(x;\Theta_1(\mathbf{Z})\right)\right)\mathrm{Var}\left(T\left(x;\Theta_2(\mathbf{Z})\right)\right)}} \\
&= \frac{E_{\mathbf{Z}}\left[\mathrm{Cov}_{\Theta_i|\mathbf{Z}}\left(T\left(x;\Theta_1(\mathbf{Z})\right), T\left(x;\Theta_2(\mathbf{Z})\right)\right)\right] + \mathrm{Cov}_{\mathbf{Z}}\left[E_{\Theta_1|\mathbf{Z}}T\left(x;\Theta_1(\mathbf{Z})\right), E_{\Theta_2|\mathbf{Z}}T\left(x;\Theta_2(\mathbf{Z})\right)\right]}{\sqrt{E_{\mathbf{Z}}\left[\mathrm{Var}_{\Theta_1|\mathbf{Z}}T\left(x;\Theta_1(\mathbf{Z})\right)\right] + \mathrm{Var}_{\mathbf{Z}}\left[E_{\Theta_1|\mathbf{Z}}T\left(x;\Theta_1(\mathbf{Z})\right)\right]}\sqrt{E_{\mathbf{Z}}\left[\mathrm{Var}_{\Theta_2|\mathbf{Z}}T\left(x;\Theta_2(\mathbf{Z})\right)\right] + \mathrm{Var}_{\mathbf{Z}}\left[E_{\Theta_2|\mathbf{Z}}T\left(x;\Theta_2(\mathbf{Z})\right)\right]}}
\end{aligned}$$

因为 $T\left(x;\Theta_i(Z)\right)$ 的自助采样和特征采样分别是独立同分布的，则

$$\mathrm{Cov}_{\Theta_i|\mathbf{Z}}\left[T\left(x;\Theta_1(\mathbf{Z})\right), T\left(x;\Theta_2(\mathbf{Z})\right)\right] = 0$$

且

$$E_{\Theta_1|\mathbf{Z}}T\left(x;\Theta_1(\mathbf{Z}) = E_{\Theta_2|\mathbf{Z}}T\left(x;\Theta_2(\mathbf{Z})\right)\right.$$

$$\mathrm{Var}_{\Theta_1|\mathbf{Z}}T\left(x;\Theta_1(\mathbf{Z})\right) = \mathrm{Var}_{\Theta_2|\mathbf{Z}}T\left(x;\Theta_2(\mathbf{Z})\right)$$

则分子只剩下第二项，分母的两个因子实际上相等，于是

$$\rho = \frac{\mathrm{Var}_{\mathbf{Z}}\left[E_{\Theta|\mathbf{Z}}T(x;\Theta(\mathbf{Z}))\right]}{\mathrm{Var}_{\mathbf{Z}}\left[E_{\Theta|\mathbf{Z}}T(x;\Theta(\mathbf{Z}))\right] + E_{\mathbf{Z}}\,\mathrm{Var}_{\Theta|\mathbf{Z}}[T(x;\Theta(\mathbf{Z}))]}$$

证毕.

偏误

在随机森林中，预测偏误和任意一个单独模型的预测偏误相同：

$$\mathrm{Bias}(x) = \mu(x) - \mathbb{E}_Z\hat{f}_{rf}(x) = \mu(x) - \mathbb{E}_Z\mathbb{E}_{\Theta|Z}[T(x;\Theta(Z))] \tag{4.27}$$

由图5.21，**预测偏误随特征个数的增加而减小**。一种理解是，单棵树的分裂点越多，分支越精细，预测偏误越小。最优的结果是在减小偏误和减小预测误差之间的平衡。

总之：减小特征个数 m，树的相关性降低，预测方差降低，树的分类能力也会相应的降低，导致预测偏误增加；增大 m，则相反。关键问题是如何选择最优的 m（或者是范围），这也是随机森林唯一的一个参数。

4.3 强预测变量与目标随机森林 TRF

此部分主要讲解的是《Targeting predictors in random forest regression》文章的内容。该文章 2023 年发表于期刊 International Journay of Forecasting，主要说明了对于随机森林方法，使用强预测变量进行预测的最终的预测效果要强于不做变量选择的模型。这篇文章通过从概率数学理论方面进行推导，得到了对该命题理论上的证明。

4.3.1 前提

假设样本空间 $X = [0,1]^p$, 回归函数 $f(\cdot)$ 形式为：

$$f(x) = g(x_{su}), x \in [0,1]^p \tag{4.28}$$

其中 $g(x_{su})$ 为可测函数 $g : [0,1]^s \to \mathbf{R}$，子集 $su \subseteq [p]$,$x_{su} = (x^{(i)})_{i \in su}$，$s = |su|$ 且远小于 p。

假设：

(A1) 数据 $S_n = \{(X_1, Y_1), ..., (X_n, Y_n)\}$ 是遍历的序列；

(A2) 输入向量 X 在 $[0,1]^p$ 上是均匀分布；

(A3) 回归函数是线性的 $f(x) = \beta_0 + \sum_{i \in su} \beta_i x^{(i)}$

假设 (A1) 确保实证均值收敛到理论，该假设条件普遍在大部分的统计过程中成立。假设 (A2) 在非参数模型中比较经典。假设 (A3) 是为了简化说明。

在以上假设中，$su = \{i : \beta_i \neq 0\}$，其中，$f(\cdot)$ 是可加的，即 $f(x) = \sum_{i \in su} f_i(x^{(i)})$。累加回归树 CART 框架在分类与回归标准下是十分普遍的。给定一个节点 A 的树，当样本足够大时，节点 A 中的杂质将会渐进减少，除非 $f(\cdot)$ 为常数。

4.3.2 对强预测元分离的概率

考虑初始时是否对变量筛选对分离概率的影响。设一个初始集 $A \subseteq [p]$，且 $a := |A|$ 作为树的先验设置。$su \subseteq A \subset [p]$。给定一个树，固定一个节点 $A \subseteq [0,1]^a$，挑选随机变量 $m(a) \leq a$ 来对 A 的随机子集 M_{try}^A 进行分割。，令 $\rho_n(A)$ 为从 A 中区分出强预测元的概率。令 $s(a) = |A \cap su|$ 为 A 中的强预测元数量，当 $s(a) \ll a$，此时 $m(a)$ 可能被调整的足够小以保证 $\rho_n(A) < 1$ 并降低了树的多样性，导致了 RF 估计的方差下降。

另一方面，为了避免严格有偏的树，在设置分离方向时不能简单地随机选择，强预测元应当在绝大部分时间被选择，即 $\rho_n(A) \gg \frac{s(a)}{a}$，这意味着 $m(a)$ 不应该被挑选的太低。

此处要证明 (1) 仅仅提高 $m(a)$ 并不能确保 $\rho_n(A)$ 变得足够大；(2) 使用合适的目标可以增加 $\rho_n(A)$。

当 $\rho_n(A)$ 很难直接确定时，可以根据它的边界来确定。令 L^* 表示 CART 目标函数的全体对象，表示为：

$$L^*(i, \tau, A) = Var_A(Y) - \mathbf{P}_A(X^{(i)} \leq \tau) Var_A(Y | X^{(i)} \leq \tau) - \mathbf{P}_A(X^{(i)} > \tau) Var_A(Y | X^{(i)} > \tau) \tag{4.29}$$

其中 A 表示 $\{X \in A\}$，并定义：

$$\delta_n(A) = sup_{i \in A, \tau \in A^{(i)}} |L_n(i, \tau) - L*(i, \tau)| \tag{4.30}$$

$$C^*(A) = sup_{i \in M_{try}^A \cap su, \tau \in A^{(i)}} L^*(i, \tau) \tag{4.31}$$

其中，函数 $\delta_n(A)$ 表示降除杂质的有限样本分布，$C^*(A)$ 为在 M_{try}^A 中强预测元的最大的信号（与 CART 准则相关）。

定理 1 强预测元分离的概率 $\rho_n(A)$ 满足：

$$\mathbf{P}(2\delta_n(A) < C^*(A)) \leq \rho_n(A) \leq \mathbf{P}(M_{try}^A \cap su \neq \emptyset) \tag{4.32}$$

在假设 (A1) 以及 $E(|Y|^\gamma) < \infty$，其中 $\gamma > 2$，则除杂估计的误差可以渐进忽略。并且 $\delta_n(A) \to_p 0, as\ n \to \infty$，此外，当假设 (A2)，(A3) 满足时，有 $\rho_n(A) \to \mathbf{P}(M_{try}^A \cap su \neq \emptyset), n \to \infty$。

其中，$\delta_n(A)$ 足够小或者 $C^*(A)$ 足够大时，则 $rho_n(A)$ 下界就足够紧，而设置一个关于 Y 的较为温和的矩条件是为了保证上界可以被取到。实际上，在证明 $\rho_n(A) \to \mathbf{P}(M_{try}^A \cap su \neq \emptyset), n \to \infty$ 上，假设 (A2)(A3) 并不是严格需要的条件，因为 $\delta_n(A) \to_p 0$，只需要：

$$sup_{\tau \in A^{(i)}} L^*(\tau, i) > 0, \forall i \in su \tag{4.33}$$

控制上界

RF 与 TRF 模型均可以通过选择函数 $m(\cdot)$ 来调节，使得定理 1 中的上界降低。使用超几何分布，上界可以间接的被计算出来：

$$\mathbf{P}(M_{try}^A \cap su \neq \emptyset) = 1 - \mathbf{1}_{m(a)<a-s(a)} \begin{pmatrix} a-s(a) \\ m(a) \end{pmatrix} / \begin{pmatrix} a \\ m(a) \end{pmatrix} \tag{4.34}$$

其中 a 相比与 s(a) 足够大，合理选择 m(a) 可以确保上界不会太大。然而，在小样本条件下，$\rho_n(A)$ 显著小于这个上界，因此，除了选择 m(a) 函数，仍需要采取其他的措施（目标选择）。

控制下界

目标预测元也可用于提升定理 1 中的下界。给定 M_{try}^A，$C^*(A)$ 确定，且对于某些 $f(\cdot)$ 形式是可以被间接计算的。例如，当满足假设 (A3) 时，可以证明：

$$C^*(A) = \frac{1}{16} sup_{i \in M_{try}^A \cap su} \beta_i^2 Leb(A^{(i)})^2 \tag{4.35}$$

其中 $A = A^{(1)} \times ... \times A^{(p)}$ 是分离的节点，且 $A^{(i)}$ 是相应的第 i 个预测元可行的区间。总体上，有 $C^*(a) \leq sup_{i \in A, \tau \in A^{(i)}} L^*(i, \tau)$, 等式右边独立于 $m(a)$，且对于某些回归方程是任意小的，即使固定 A 和 Var(f(X))，这意味着对于下界，$m(a)$ 的作用有限。此时，$\delta_n(A)$ 不得不考虑进来。这个函数可以通过缩小样本集 B 来降低数值。此外，对于 A, B,C^* 是一样的，因此这个下界就提升了。这意味着 $\rho_n(B)$ 相比于 $\rho_n(A)$ 有更小的边界。因此，具有更小的条件集 B 表现是优于集 A 的。

观点 1 令 $A, B \subseteq [p]$ 有 $\delta_n(B) \leq_{st} \delta_n(A)$ 且 $C^*(A) \leq_{st} C^*(B)$，有：

$$\mathbf{P}(2\delta_n(A) < C^*(A)) \leq \mathbf{P}(2\delta_n(B) < C^*(B)) \tag{4.36}$$

特别地，当使用 A 作为目标集时，有 $C^*([p]) \leq C^*(A)$。

条件 $\delta_n(B) \leq_{st} \delta_n(A)$ 意味着 $B \subseteq A$，在合适的假设下，可以使用极值理论的近似关系：

$$\mathbf{P}(\delta_n(A) \geq x) \approx a\mathbf{P}(Z_n \geq x) \tag{4.37}$$

其中，$s(a) \ll a \ll n$，且某些随机变量 Z_n 并不依赖于 A，

考虑一种情况，当所有的方向都是可行的（$m(\cdot)$ 是恒等映射），且 B 包含与 A 同等的强方向，这说明 $C^*(A)$、$C^*(B)$ 被确定且有 $C^*(A) = C^*(B) = C^*$ 且 A 代表的是一个没有从原始集 $[p]$ 中消除非目标集的目标集，此时目标增加 ρ_n 的下界。若使用逼近方法，则有：

$$\mathbf{P}(2\delta_n(A) < C^*) - \mathbf{P}(2\delta_n([p]) < C^*) \approx (p-a)\mathbf{P}(2Z_n \geq C^*) \tag{4.38}$$

因此，目标预测元的随机森林方法提升了分离强预测变量的概率的下界，近似线性与抛弃的弱预测元的数量。因此，这表明目标随机森林 TRF 在高维的设定中要优于一般的随机森林方法。

4.3.3 树的强度

给定数据 S_n, 一般的具有 L 叶的预测树 $\hat{f_{L,n}}(x)$ 的形式为：

$$\hat{f_{L,n}}(x) = \sum_{i=1}^{L} \overline{Y_n}(A_{i,n})\mathbf{1}_{A_{i,n}}(x),\ x \in [0,1]^p \tag{4.39}$$

同时，考虑另一个集合 B 为 $[0,1]^p$ 的子集，相比 A 仅包含强预测元。此时，记为：

$$\tilde{f_{L,n}}(x) = \sum_{i=1}^{L} \overline{Y_n}(B_{i,n})\mathbf{1}_{B_{i,n}}(x^{(1)}),\ x \in [0,1]^p \tag{4.40}$$

文章评价了 $\tilde{f_{L,n}}$ 以及 $\hat{f_{L,n}}$，其中 $L \ll n$, 文章假设 $(A_{i,n})_{i=1}^{L}$ 可以进行理论最优化。$(A_{i,n})_{i=1}^{L} = (A_{i,n}^{L})_{i=1}^{L}$ 可以从起始步骤 $A_1^1 = [0,1]^p$, 并通过以下步骤进行循环：

(1) 对于 $k < L$，令 $M_{try}^{(k)}$ 作为 $[p]$ 由规模 m 的随机选择的子集

(2) 选择一个 $(A_i^k)_{i=1}^k$ 节点 A，若 $M_{try}^{(k)} \cap su \neq \emptyset$，第 k 个分割 (i^*, τ^*) 是由优化式 (3) 来获得。

(3) 分割 $(A_i^{k+1})_{i=1}^{k+1}$ 与 $(A_i^k)_{i=1}^k$ 的做法相同，而 A 不同，在给定 (i^*, τ^*)，A 是从 $A \cap \{x : x^{(i^*)} \leq \tau^*\}$ 以及 $A \cap \{x : x^{(i^*)} > \tau^*\}$ 分割.

根据定理 1，可知分离强预测元的概率 ρ 为：

$$\rho = 1 - \mathbf{1}_{m+s<p} \begin{pmatrix} p-s \\ m \end{pmatrix} / \begin{pmatrix} p \\ m \end{pmatrix} \tag{4.41}$$

其中最优的分割 (i^*, τ^*) 并不唯一，我们决定有一项具体的平局决胜机制生效。我们使用 $L^*(\cdot)$ 来进行最优化，而不是 $L_n(\cdot)$，并按照以上步骤进行求解，同时我们规定：

规则 (R) 若 (i_j, τ_j) 是 A_j^k 的最优分割，此时节点的分割为 $A = A_{j^*}^k$，其中 $j^* = argmax_j L^*(i_j, \tau_j, A_j^k)$

同理，可以获得 $(B_i)_{i=1}^{L}$。其中所有的分割都是位于 su 中。定义最优部分树 $\hat{f_L}, \tilde{f_L}$:

$$\hat{f_L}(x) = \sum_{i=1}^{L} E[Y|X \in A_i]\mathbf{1}_{A_i}(x)$$

$$\tilde{f_L}(x) = \sum_{i=1}^{L} E[Y|X^{(1)} \in B_i]\mathbf{1}_{B_i}(x^{(1)}) \tag{4.42}$$

定理 2 若假设 (A1)(A3) 成立且分离规则 (R) 成立，则有：

$$MSE(\tilde{f_L}) = \beta_1^2 \frac{7\underline{\iota}(L) - 7L}{48\underline{\iota}(L)^3},\ L \geq 1 \tag{4.43}$$

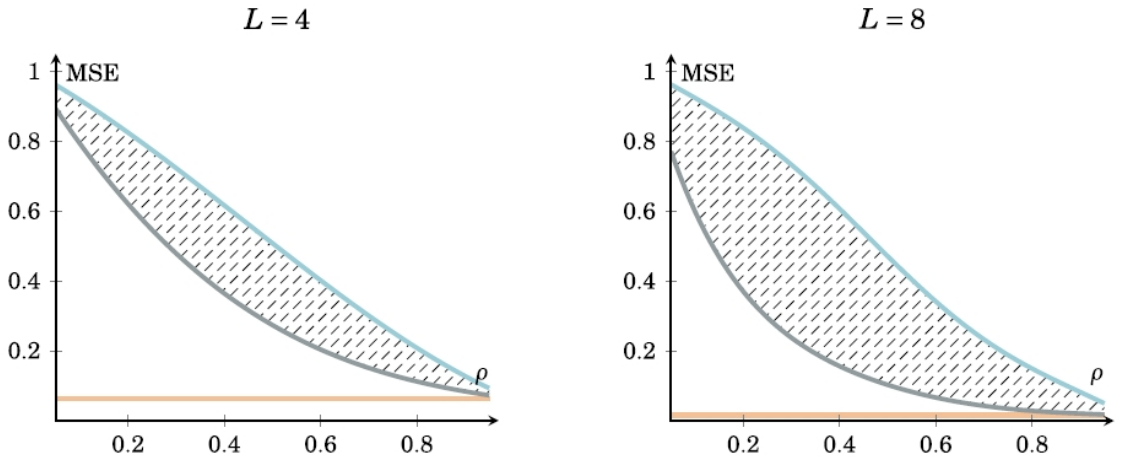


Fig. 6. Comparison of MSE to upper and lower bounds.
This figure shows a comparison of MSE($\tilde{f}_L$) (orange) to the upper (blue) and lower (gray) bounds of MSE($\hat{f}_L$) from Corollary 1 as functions of ρ, with $\beta_1 = \sqrt{12}$, for two values of L. The graph of MSE($\tilde{f}_L$) is located somewhere in the shaded region. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

图 4.5: 不同叶片数量 L 下 MSE 与 ρ 的关系

$$MSE_\Theta(\tilde{f}_{\underline{l}_0+\frac{N-l_0+1}{l_0(L_N)}}) \geq MSE_\Theta(\hat{f}_L) \geq MSE_\Theta(\tilde{f}_{\bar{\iota}(1+\frac{N}{l_1})}), L \geq 1 \tag{4.44}$$

$$\underline{\iota}(x) = 2^{\lfloor log_2(x) \rfloor}$$

$$\bar{\iota}(x) = 2^{\lceil log_2(x) \rceil}$$

其中 $\Theta = (M_{try}^{(1)}, ..., M_{try}^{(L-1)})$,$N = \sum_{k=1}^{L-1} \mathbf{1}_{\{M_{try}^{(k)} \cap su \neq \emptyset\}}$ 以及 $l_i = min\{k : \mathbf{1}_{\{M_{try}^{(k)} \cap su \neq \emptyset\}} = i\}$
令 $g_0(x,y) = MSE(\tilde{f}_{\underline{\iota}(y+\frac{x-y+1}{y(L-x)})})$，以及 $g_1(x,y) = MSE(\tilde{f}_{\bar{\iota}(1+\frac{x}{y})})$，根据定理 2，可得推论 1：
推论 1 在定理 2 的条件下，

$$E[g_0(N,l_0)] \geq MSE(\hat{f}_L) \geq E[g_1(N,l_1)] \tag{4.45}$$

其中 (N,l_0)，(N,l_1) 的概率质量函数可得：

$$\mathbf{P}(N=n,l_0=k) = \left\{ \begin{array}{c} \rho^{k-1}(1-\rho)Bin(n+1-k;L-1-k,\rho)\ if\ n \in (0,L-1) \& k \in [1,n+1] \\ \rho^n(1-\rho)^{L-1-n},\ if\ n \in [0,L-1] \& k = 1 \\ 0,\ otherwise \end{array} \right\}$$

$$\mathbf{P}(N=n,l_1=k) = \left\{ \begin{array}{c} \rho(1-\rho)^{k-1}Bin(n-1;L-1-k,\rho)\ if\ n \in (0,L-1) \& k \in [1,L-n] \\ \rho^n(1-\rho)^{L-1-n},\ if\ n \in [0,L-1] \& k = 1 \\ 0,\ otherwise \end{array} \right\}$$

其中 $Bin(k:n,\rho) = \left(\begin{array}{c} n \\ k \end{array} \right) \rho^k(1-\rho)^{n-k}$ 表示伯努利分布

可以得到 ρ 与 MSE 之间的关系。如图5.20。

其中 MSE 是位于上界与下界包围的灰色区域内部的一点，由图5.20可知，叶片数量 L 对不同预测强度的变量的 MSE 上界和下界有着非常明显的影响。当 $L = 4$ 时，MSE 的边界非常紧凑，当 $L = 8$

时 MSE 的整体范围发生较大改变。因此，对树的叶片数量的设置将会影响随机森林模型 MSE 的整体区域的边幅，但是不会影响文章的关键结论，即变量选择（筛选出强预测元）将会提升随机森林方法的预测能力。

4.4 贝叶斯累加回归树 BART

4.4.1 简介

BART 是一种决策树模型，和梯度提升树相似，BART 模型是很多 CART 树之和，但不同之处在于：

BART 是通过一个正则化先验来弱化个体决策树 BART 属于概率模型，模型的训练使用传统的参数估计方法迭代方式不同，BART 使用了贝叶斯 backfitting 方法 BART 的每棵树只负责预测“一小部分”，且只对自己负责的“部分”进行残差拟合，而 boosting 算法中每棵树负责拟合整体的残差。模型基本形式 BART 模型有两个要点，一是模型的基本形式是多个 CART 决策树之和 (sum-of-trees)，二是模型对参数的先验进行了正则化（regularization prior），下面分别给出解释。

4.4.2 树的汇总

考虑一个回归问题。我们有 n 个样本数据，每个数据的维度是 p，用 nxp 矩阵 X 表示，nx1 向量 Y 为因变量。

先给出 BART 模型的具体形式：

$$Y = f(X) + \varepsilon \approx \mathcal{T}_1^{\mathcal{M}_1}(X) + ... + \mathcal{T}_1 m^{\mathcal{M}_m}(X),\ \varepsilon \sim N_n(0, \sigma^2 I_n) \tag{4.46}$$

其中，ε 是 nx1 向量表示高斯噪声，m 是 CART 树的个数，$\mathcal{T}_i^{\mathcal{M}_i}(X)$ 表示第 i 个 CART 树的结构（也就是树的形状，以及所有内部节点的划分维度和划分规则），$\mathcal{M}_i$ 表示第 i 个 CART 树的所有叶子结点的参数（也就是每个叶子结点上的预测值），b_i 表示第 i 个 CART 树叶子结点的个数。

由此对每个 i，$\mathcal{T}_i^{\mathcal{M}_i}$ 就代表了一个 CART 树预测器，我们的 BART 其实就是用这 m 个 CART 树分别作完预测后，再求和得到最终的预测，这就是所谓的 sum-of-tree 模型。BART 的关键就是如何构造这些 CART 树。

说 CART 树略有不妥，准确说应该是更一般的二叉回归树，后面可以看到，两者在生成方式上有很大的区别。

4.4.3 正则化先验

我们的模型参数是 $\mathcal{T}_1, \mathcal{M}_1, ..., \mathcal{T}_m, \mathcal{M}_m, \sigma^2$

对参数做正则化的先验，一方面是为了防止某棵树生长过大而占据预测的主导地位，从而失去了建立这种可加树模型的意义。（可加树模型的初衷就是因为单棵树很容易出现过拟合的现象），另一方面是便于计算，这在后面的推导中可以看出。

正则化先验分为 5 个部分：

独立性和对称性

首先我们假设每棵树之间是相互独立的，且均与方差 σ 独立。

$$\mathbb{P}(\mathcal{T}_1, \mathcal{M}_1, ..., \mathcal{T}_m, \mathcal{M}_m, \sigma^2) = [\prod \mathbb{P}(\mathcal{T}_t, \mathcal{M}_t)]\mathbb{P}(\sigma^2) \tag{4.47}$$

同时假设，在给定树的具体结构的条件下，数的叶子结点的预测值之间是相互独立的。

$$\mathbb{P}(\mathcal{M}_t|\mathcal{T}_t) = \prod_l \mathbb{P}(\mu_{t,l}|\mathcal{T}_t) \tag{4.48}$$

这样我们就可以把参数的先验分布改写成

$$\mathbb{P}(\mathcal{T}_1, \mathcal{M}_1, ..., \mathcal{T}_m, \mathcal{M}_m, \sigma^2) = [\prod \mathbb{P}(\mathcal{M}_t|\mathcal{T}_t)]\mathbb{P}(\sigma^2) = \prod_t \prod_l \mathbb{P}(\mu_{t,l}|\mathcal{T}_t)\mathcal{P}(\mathcal{T}_t)\mathbb{P}(\sigma^2) \tag{4.49}$$

通过这个独立性假设，我们可以把问题简化为只需分别设计 $\mathbb{P}(\mathcal{T}_t)\mathbb{P}(\mathcal{M}_t)\mathbb{P}(\sigma^2)$ 的先验即可。

$\mathbb{P}(\mathcal{T}_t)$ 的先验

这个先验指的是对树的形状以及树的划分准则，其中划分准则包括内部节点的划分变量（选择哪个特征做划分）和划分值（切在哪个位置）的先验。注意在讨论 $\mathbb{P}(\mathcal{T}_t)$ 时，不考虑树的叶子结点上的预测值，那是 $\mathbb{P}(\mathcal{M}_t|\mathcal{T}_t)$ 要干的事，我们后面会另外给出。

首先是树的形状的先验。我们考虑二叉树 $\mathcal{T}_t$ 的生成方式，是从根结点出发，对每个非叶子结点 η，每次按某个的概率 $p_{split}(\eta_t, \mathcal{T}_t)$ 判断是否要继续向下生长出左右两个叶子结点。

一个很自然的想法是，对于所有的节点 η，都取成同一个值，比如 α，也就是不管哪个节点，向下继续划分的概率都是相同的。这当然是可以的，毕竟我们现在在做的事是确定参数的先验，这可以是非常主观的。但是这样的先验其实并不是很好，因为这隐含了一个不太合理的假设，就是所有拥有 n+1 个叶子结点的二叉树出现的概率是一样的（一共有 C_n 种），都是 $\alpha^n(1-\alpha)^{n+1}$（注意，有 n+1 个叶子结点的二叉树一定有 n 个内部节点，可以用归纳法证明）。这导致了我们生成的二叉树的形状不太好控制，但我们其实是希望生成的二叉树尽量的" 平衡" 一些，也就是叶子结点所在的层数不要差太大。

根据 CGM98 这篇文章，我们定义每个结点向下生长的概率为该结点所在深度 d_η 的函数（比如根节点对应的深度就是 0，根结点的孩子结点对应的深度是 1，以此类推）。具体的，

$$p_{split}(\eta_t, \mathcal{T}_t) = \alpha(1 + d_\eta)^{-\beta} \tag{4.50}$$

其中 $\alpha \in [0, 1]$,$\beta \in [0, +\infty]$

在这个先验假设下，通过设置合理的超参数 α, β，我们既可以保证各个叶子结点的深度不会相差太多，同时又能控制树的深度不会太深。注意是模型的超参数，是可以根据问题自行调整的。

在 CGM98 文章里，作者给出了超参数 α, β 取不同值的时候，叶子结点数的分布，如下图6.2所示。

因为 BART 模型希望每棵树都是一个弱学习器，也就是不希望单棵树生长得太深，或者说不希望叶子结点数量太多，所以第四幅图是比较符合我们的要求的（对应参数 $(\alpha, \beta) = (0.95, 1.5)$，叶子结点个数的均值为 2.9）。在 BART 的原始论文 Chipman et al. (2010) 中，作者选取了参数默认值为 $(\alpha, \beta) = (0.95, 2)$。

在给定了 $p_{split}(\eta, T)$ 之后，我们就能得到生成出各种形状的二叉树的概率分布了。

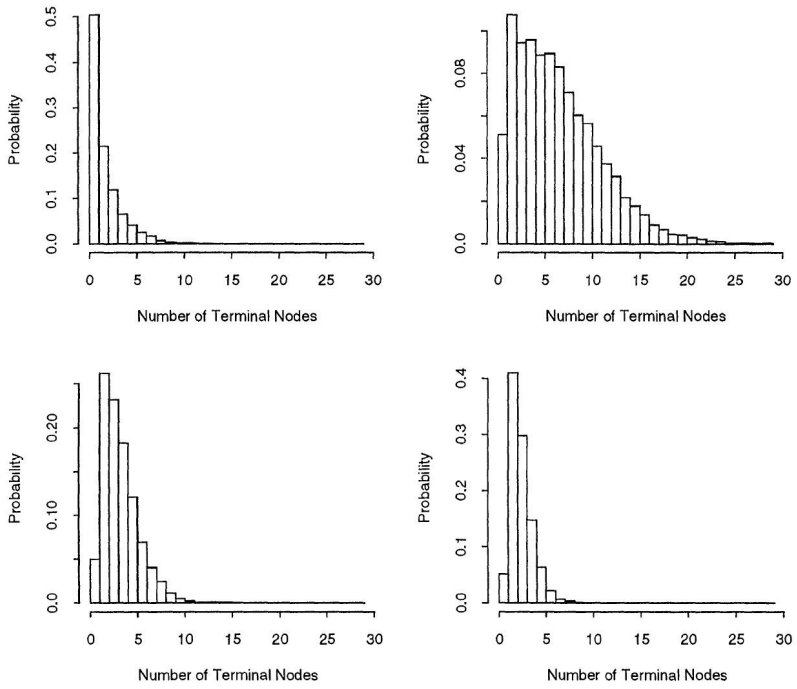


Figure 3. Prior Distribution on Number of Terminal Nodes. (a) α = .5, β = .5, with prior mean 2.1; (b) α = .95, β = .5, with prior mean 7; (c) α = .95, β = 1, with prior mean 3.7; (d) α = .95, β = 1.5, with prior mean 2.9.

图 4.6: α, β 对叶片数量的影响

然后是树的划分准则的先验。所谓树的划分准则，其实是指树的每个内部结点（非叶子结点）的划分准则。划分准则包括两个部分，划分变量和划分值。

划分变量的选择。我们的数据是 p 维的，所以在为一开始的根节点选择划分特征时，我们一共有 p 种特征可以选择。对于每个特征被选到的先验概率，我们一般就选择均匀分布，也就是说，在为根节点选择划分特征的时候，每个特征被选到的概率都是 $\frac{1}{p}$ 。对于其他内部节点，可划分的特征数可能会小于 p，这时需要将这些特征排除，然后再按均匀分布选取特征即可。当然，如果你有自己的先验判断，也可以为每个特征指定不同的被选中概率。比如你事先认为某个特征对最终的预测不太重要，那么你就可以给这个特征分配一个较低的先验概率，这样的话在为根节点选择划分特征的时候就不太可能选择这个你认为无关紧要特征。划分值的选择。划分值的先验也是均匀的，也就是在样本对应特征上的取值里均匀地采样（不去重）。

4.4.4 $\mathbb{P}(\mathcal{M}_t|\mathcal{T}_t)$ 的先验

在给定树的结构 $\mathcal{T}_t$ 后，树的每个叶子结点就代表了特征所在的 p 维空间中的一块区域，我们现在要做的就是为每一个区域（叶子结点）分配一个预测值。

由于 $\mathbb{P}(\mathcal{M}_t|\mathcal{T}_t) = \prod_l \mathbb{P}(\mu_{t,l}|\mathcal{T}_t)$，同时我们假设一开始所有叶子结点上的预测值都是相同的，由此我们只需指定 $\mathbb{P}(\mu_{t,l}|\mathcal{T}_t)$。

我们的做法是对所有的 t 和 l，节点上预测值的先验都取为：$\mu_{t,l}|\mathcal{T}_t \sim N(\mu_\mu, \sigma_\mu)$，其中 μ_μ, σ_μ 是待定的，稍后会给出。

我们来看一下这样的先验假设意味着什么。由于我们的模型得到的最终预测是将 m 棵树的预测值

相加，那么在这个先验假设下，模型的预测值服从 $N(m\mu_\mu, m\sigma_\mu)$。注意到，合理的模型的预测应该有很大的概率落在区间 $[y_{min}, y_{max}]$ 内，其中 y_{min}, y_{max} 分别表示样本的最小和最大值。由此，我们的参数 μ_μ, σ_μ 可以这样选取：

$$m\mu_\mu - k\sqrt{m}\sigma_\mu = y_{min} \tag{4.51}$$

$$m\mu_\mu + k\sqrt{m}\sigma_\mu = y_{max} \tag{4.52}$$

其中 k 用来衡量置信度，是模型的超参数。

比如取 k=2，那么上面的式子就表示我们的预测有 95% 的概率落在区间 $[y_{min}, y_{max}]$ 内。

通过求解上式，我们得到：

$$\mu_\mu = \frac{y_{min} + y_{max}}{2m} \tag{4.53}$$

$$\sigma_\mu = \frac{y_{max} - y_{min}}{2k\sqrt{m}} \tag{4.54}$$

从而最终得到我们对每棵树、每个叶子结点上预测值的先验都是同一个正态分布：

$$\mu_{t,l}|\mathcal{T}_t \sim N(\frac{y_{min} + y_{max}}{2m}, (\frac{y_{max} - y_{min}}{2k\sqrt{m}})^2) \tag{4.55}$$

为了减小原始数据中的极端值所带来的影响，我们需要对原始数据做一些标准化处理，比如 BART 的原作者就建议把按最大、最小值做线性变换，变换到区间 [-0.5,0.5] 之内，这样的话预测值就服从零均值的正态分布。在后续推导中，我们都默认已经对 y 做了上述的变换，且：

$$\mu_{t,l}|\mathcal{T}_t \sim N(0, \sigma_\mu^2),\ \sigma_\mu = \frac{1}{2k\sqrt{m}} \tag{4.56}$$

我们一般只对 y 做标准化处理，决策树的特点使得我们无需对 x 做标准化处理，这一点需要区别于神经网络等其他模型。

4.4.5 σ 的先验

由于我们假设了模型残差独立且同分布于零均值，方差 σ 未知的正态分布。我们可以取其共轭先验分布，即逆伽马分布：

$$\sigma^2 \sim InvGamma(\frac{v}{2}, \frac{v\lambda}{2}) \tag{4.57}$$

其中 v 一般取在区间 [3,10] 内，在给定 v 之后，取 λ 使得下面的式子成立：

$$\mathbb{P}(\sigma^2 < \hat{\sigma}^2) = q \tag{4.58}$$

其中 q 一般选取为 0.75,0.90 或 0.99。一般可以直接取为因变量 y 的方差（也可以取为用 x 对 y 做线性回归后的残差方差）。上面不等式的意义就是让我们的模型方差有比较高的概率能低于原始数据的方差（或者低于用线性回归拟合之后的方差）

注意，在 σ 的先验分布里，我们的超参数其实是 v 和 q ，因为 λ 是由 v 和 q 唯一确定的。

BART 的作者给出了三组建议的参数超参数取值,如下图所示。图中我们假设 Y 本身的方差 $\hat{\sigma}^2 = 2$。

其中 $(v, q) = (10, 0.75)$ 是比较保守的选择，此时 σ 分布对应的均值大约在 1.5 附近，而且不超过 $\hat{\sigma}^2$ 的概率只有 75%，也就是我们的先验给模型赋予了一个比较大的方差，说明我们一开始对模型预测结果不是太确信，是比较保守的选择。$(v, q) = (3, 0.99)$ 是比较激进的选择，此时 σ^2 分布对应的均值大约在 0.5 附近，说明我们对模型的预测比较自信。作者建议的默认取值是 $(v, q) = (3, 0.9)$，介于保守和激进之间。

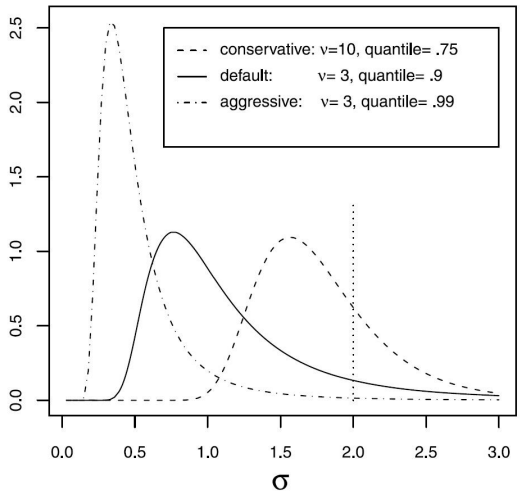


FIG. 1. *Three priors on* σ *based on* df $= \nu$ *and* quantile $= q$ *when* $\hat{\sigma} = 2$.

图 4.7: 不同 v, q 设置时 σ 的先验分布

4.4.6 m 的确定

与 boosting 方法不同，BART 模型是确定数量的树之和，因此这个数量也需要提前确定。我们把 m 视为超参数，参数和超参数的区别在于，超参数一旦确定，在模型训练的过程中就不会再改变了。超参数一般可以通过交叉验证法来确定，也可以直接使用软件包里的默认参数。

这里作者建议 m=200。另外作者还通过实验发现，当 m 从 1 开始逐渐变大时，模型的表现会快速上升，然后就趋于平缓，但是当 m 大到一定程度后，模型的表现会缓慢地下降。

以上我们给出了模型各个参数的先验分布，同时给出了模型的超参数为 $(\alpha, \beta, k, v, q, m)$

4.4.7 算法流程

在给出具体算法流程之前，我们先给出参数更新的方法。

我们需要从后验分布 $\mathbb{P}(\mathcal{T}_1, \mathcal{M}_1, ..., \mathcal{T}_m, \mathcal{M}_m, \sigma^2|Y)$ 中进行采样来作为模型的参数估计，使用的是 MCMC 中的 Gibbs 算法，这是一个迭代的过程。对于 BART 这一类加性模型（additive model）的训练，我们还要引入贝叶斯 backfitting 的技巧，这一技巧的核心在于，在循环迭代过程中，我们每步只训练一棵树，每棵树训练时使用的因变量"y" 不再是原来的数据 y，而用 y 减去之前已经采样好的其他 m-1 棵树预测值之和后得到的残差 R。具体地，在训练第 j 棵树时，我们拟合的目标为：

$$R_{-j} := y - \sum_{t \in \{1,2,..,m\}, t \neq j} \mathcal{T}_t^{\mathcal{M}_t(X)} \tag{4.59}$$

由此，我们可以给出 Gibbs 算法的参数更新步骤。假设我们共迭代 K 轮，对于第 k 轮迭代，我们的参数更新顺序如下（为了美观，省去参数所在轮次的角标）：如图6.3

简单给出一些解释。在第 k 轮迭代中，第一步，我们需要在给定滚动残差 R_{-1} 和模型方差 σ 的条件下来生成 $\mathcal{T}_1$，也就是对条件分布 $\mathbb{P}(\mathcal{T}_1|R_{-1}, \sigma^2)$ 进行采样。

这一步采样需要用到 MCMC 中的 MH 算法，因为我们无法直接从后验密度中采样。这里有很多

$$
\begin{aligned}
1: &\quad \mathcal{T}_1 \mid \boldsymbol{R}_{-1}, \sigma^2 \\
2: &\quad \mathcal{M}_1 \mid \mathcal{T}_1, \boldsymbol{R}_{-1}, \sigma^2 \\
3: &\quad \mathcal{T}_2 \mid \boldsymbol{R}_{-2}, \sigma^2 \\
4: &\quad \mathcal{M}_2 \mid \mathcal{T}_2, \boldsymbol{R}_{-2}, \sigma^2 \\
&\quad \vdots \\
2m-1: &\quad \mathcal{T}_m \mid \boldsymbol{R}_{-m}, \sigma^2 \\
2m: &\quad \mathcal{M}_m \mid \mathcal{T}_m, \boldsymbol{R}_{-m}, \sigma^2 \\
2m+1: &\quad \sigma^2 \mid \mathcal{T}_1, \mathcal{M}_1, \ldots, \mathcal{T}_m, \mathcal{M}_m, \mathcal{E}
\end{aligned}
$$

图 4.8: 参数更新顺序

细节，我们知道 MH 算法需要指定一个转移概率，同时需要计算接收概率

$$r = \frac{\mathbb{P}(\mathcal{T}|\mathcal{T}*)}{\mathcal{T}*|\mathcal{T})} \frac{\mathbb{P}(\mathcal{T}|R,\sigma)}{\mathcal{T}*|R,\sigma)} \tag{4.60}$$

转移的具体方法其实就是在上一轮迭代的树 $\mathcal{T}_1$ 的基础上进行一次修改，可以是下面三种的某一个（作者设置的默认概率分别为 28%、28%、44%，这是超参数。另外，当树生长到无可划分特征的情况下，GROW 的提议一定被拒绝；当树只剩根节点时，PRUNE 和 CHANGE 的提议一定被拒绝）：

生长（GROW）：将某个叶子结点分裂为两个新的叶子结点

剪枝（PRUNE）：剪去某个叶子结点（包括他的兄弟节点），同时将他的父亲节点置为叶子结点。

改变内部节点划分准则（CHANGE）：对内部节点进行划分准则的修改，作者这里限制只能对那些孩子结点为叶子结点的内部节点（称为 singly internal nodes）进行划分准则的修改。

作者对这三种转移方式分别进行了计算。这部分推导较为繁琐，详细请参考 bartMachine 的附录。

第二步，我们需要在给定滚动残差 R_{-1}、模型方差 σ^2 和在第一步生成的的条件下来生成，这一步是可以直接对后验采样的，因为我们的似然函数和先验都是正态的，因此后验也是正态分布，均值就是似然和先验均值的加权平均。

第三步到第 2m 步同理。

最后一步，我们需要在给定模型所有其他参数的条件下来对模型方差 σ^2 采样，由于先验是逆伽马分布，

根据 $\varepsilon := y - \sum_{t=1}^{m} \mathcal{T}_t^{\mathcal{M}_t(X)}$ 容易写出似然函数为方差未知的正态密度的乘积，这样后验也是逆伽马分布，这一步也是可以直接采样的。

补充一个小细节，实际编程的时候，我们所有的概率计算都要在 log 环境下进行，原因是似然函数是概率的乘积，很多个概率乘积必然会非常接近 0，因此必须对所有概率取对数。注意：图6.5

最后，我们给出 BART 的算法流程，完整的细节可在 Bayesian Data Analysis、bartMachine 中找到。如图6.6：

1. 迭代轮数 K 一般可以取为1000。每轮迭代中，我们需要进行 $2m+1$ 次参数更新（从后验采样）。
2. 第1轮迭代的参数我们需要事先给定，这也就是前面我们花了很多篇幅讨论的正则化先验的部分。
3. MCMC采样算法必须进行“预热”，通常可以设定预热的轮数 L 为100。因此，前100轮采样的参数需要抛弃。
4. 在第 k 轮$(k=2,3,\ldots,K)$ 参数更新完成后，即可得到对应的模型：
   $\hat{f}^{(k)}(\mathbf{x})=\sum_{t=1}^{m}(\mathcal{T}_t^{(k)})^{\mathcal{M}_t^{(k)}}(\mathbf{x})$ 。
5. 我们最终的模型是对预热之后的所有模型取平均：

$$\hat{f}(\mathbf{x})=\frac{1}{K-L}\sum_{k=L+1}^{K}\hat{f}^{(t)}(\mathbf{x})$$

图 4.9: 树的采样

1. 首先初始化模型参数：
   用树根（单结点树）做为初始，其中每棵树的测值都相同，最终预测值为样本均值：
   $(\mathcal{T}_1^{(1)})^{\mathcal{M}_1^{(1)}}(\boldsymbol{x})=(\mathcal{T}_2^{(1)})^{\mathcal{M}_2^{(1)}}(\mathbf{x})=\ldots=(\mathcal{T}_m^{(1)})^{\mathcal{M}_m^{(1)}}(\mathbf{x})=\frac{1}{nm}\sum_{i=1}^{n}y_i$
2. 计算
   $\hat{f}^{(1)}(\mathbf{x})=\sum_{t=1}^{m}(\mathcal{T}_t^{(1)})^{\mathcal{M}_t^{(1)}}(\mathbf{x})$
3. For $k=2,3,\ldots,K$:

- For $j=1,2,\ldots,m$:
  - 计算:
    $\boldsymbol{R}_{-j}^{(k)}=\boldsymbol{y}-\sum_{t<j}(\mathcal{T}_t^{(k)})^{\mathcal{M}_t^{(k)}}(\boldsymbol{X})-\sum_{t>j}(\mathcal{T}_t^{(k-1)})^{\mathcal{M}_t^{(k-1)}}(\boldsymbol{X})$
  - 对$\mathbb{P}(\mathcal{T}_j\mid\boldsymbol{R}_{-j}^{(k)},(\sigma^{(k-1)})^2)$ 进行采样，得到 $\mathcal{T}_j^{(k)}$
  - 对$\mathbb{P}(\mathcal{M}_j\mid\mathcal{T}_j^{(k)},\boldsymbol{R}_{-j}^{(k)},(\sigma^{(k-1)})^2)$ 进行采样，得到 $\mathcal{M}_j{}^{(k)}$
- 得到模型:
  $\hat{f}^{(k)}(\mathbf{x})=\sum_{t=1}^{m}(\mathcal{T}_t^{(k)})^{\mathcal{M}_t^{(k)}}(\mathbf{x})$
- 计算:
  $\mathcal{E}^{(k)}=\boldsymbol{y}-\hat{f}^{(k)}(\mathbf{X})$
- 对 $\mathbb{P}(\sigma^2\mid\mathcal{T}_1^{(k)},\mathcal{M}_1^{(k)},\ldots,\mathcal{T}_m^{(k)},\mathcal{M}_m^{(k)},\mathcal{E}^{(k)})$ 进行采样，得到 $(\sigma^{(k)})^2$

4. 对预热之后的所有模型取平均，得到最终的BART模型：

$$\hat{f}(\mathbf{x})=\frac{1}{K-L}\sum_{k=L+1}^{K}\hat{f}^{(t)}(\mathbf{x})$$

图 4.10: 贝叶斯累加回归树 BART 的一般流程

4. 对预热之后的所有模型取平均，得到最终的 BART 模型：

$$\hat{f}(X) = \frac{1}{K-L}\sum_{k=L+1}^{K}\hat{f}^{(t)}(X) \tag{4.61}$$

4.4.8 案例：贝叶斯累加回归树 BART 方法对非参数 VAR 模型的估计

此处以发表在期刊 International Economic Review 的一篇文章《Tail Forecasting With Multivariate Bayesian Additive Regression Tree》为例，详细介绍贝叶斯累加回归树如何进行非参数 VAR 模型进行估计。

令 $\{y_t\}_{t=1}^{T}$ 为 M 维宏观经济和金融序列。假设 y_t 收到它前 p 期影响，我们获得一个 $K = Mp$ 维向量记为 $x_t = (y'_{t-1}, ..., y'_{t-p})'$。假设 y_t 与 x_t 的关系未知。将模型记为：

$$y_t = F(x_t) + \eta_t \tag{4.62}$$

$$\eta_t = G(z_t) + \varepsilon_t,\ \varepsilon_t \sim N(0_M, \Sigma_t) \tag{4.63}$$

其中，$F : \mathbf{R}^K \to \mathbf{R}^M$, 以及 $G : \mathbf{R}^N \to \mathbf{R}^M$ 为两个未知的函数 $F(x_t) = (f_1(x_t), ..., f_M(x_t))'$ 以及 $G(z_t) = (g_1(z_t), ..., g_M(z_t))'$。

文章主要介绍了异方差情况（同方差情况可以视为特殊的异方差情况）。时变协方差矩阵 Σ_t，其中，假设协方差矩阵 Σ_t 具有因子随机波动（FSV：可参考 Aguilar 和 West(2000)）：

$$\Sigma_t = \Lambda\Omega_t\Lambda' + H_t \iff \varepsilon_t = \Lambda\delta_t + e_t,\ \delta_t \sim N(0_Q, \Omega_t), e_t \sim N(0_M, H_t) \tag{4.64}$$

其中 Λ 为 $M \times Q$ 因子载荷矩阵 $(Q \ll M)$ 以及对角协方差矩阵 $\Omega_t = diag(e^{u_1(w_t)}, ...$
$, e^{u_Q(w_t)})$ 以及 $H_t = diag(e^{v1(w_t)}, ..., e^{v_M(w_t)})$，其中 $u_i, v_j : \mathbf{R}^R \to \mathbf{R}$ 为未知的函数，表示误差协方差矩阵与 R 维协变量 w_t，e_t 在 Q 维因子 δ_t 条件下可以在 equation-by-equation 估计。这个因子具体来说，没有被进一步约束（没有识别）。文章主要焦点在于尾部，因此没有将 Λ 和 δ_t 分开讨论而是从 Σ_t 排除。

使用贝叶斯累加回归树进行函数学习

令 $y_{\cdot i}, x_{\cdot i}, z_{\cdot i}$ 表示 $Y = (y_1, ...y_T)', X = (x_1, ..., x_T)', Z = (z_1, ..., z_T)'$ 的第 i 列，$\varepsilon_{\cdot i} = (\varepsilon_{i1}, ..., \varepsilon_{iT})'$，则非线性模型可写为：

$$y_{\cdot i} = f_i(X) + g_i(Z) + \varepsilon_{\cdot i} \tag{4.65}$$

BART 通过树的加总来对 f_i, g_i 进行近似：

$$f_i(X) \approx \sum_{s=1}^{S} l_{is}(X|\mathcal{T}_{is}^{f},\ \mu_{is}^{f}), g_i(Z) \approx \sum_{s=1}^{S} l_{is}(Z|\mathcal{T}_{is}^{g}, \mu_{is}^{g}) \tag{4.66}$$

其中

$$l_{is}(X|\mathcal{T}_{is}^{\mathcal{S}}, \mu_{is}^{\mathcal{S}}) = \sum_{n=1}^{b_{is}^{\mathcal{S}}} \mu_{is,n}^{\mathcal{S}} \mathbb{I}(X \in S_{is,n}^{\mathcal{S}}) \tag{4.67}$$

其中 $\mathbb{I}(X \in S_{is,n}^{\mathcal{S}})$ 为示性函数。$S_{is}^{\mathcal{S}}$ 表示与 $\{f, g\}$ 相关的集合，$\mu_{is}^{\mathcal{S}} = (\mu_{is,1}^{\mathcal{S}}, ..., \mu_{is,b_{is}^{\mathcal{S}}}^{\mathcal{S}})'$ 维度为 $b_{is}^{\mathcal{S}}$

树函数是由终节点 μ_{is}^{j} 和树结构 $\mathcal{T}_{is}^{\mathcal{S}} = \{S_{is,n}^{\mathcal{S}}\}_{n=1}^{b_{is}^{\mathcal{S}}}$。分割为二元分割，$\{x_{\cdot j} \le c\}$ 或 $\{x_{\cdot j} \ge c\}$ 对于 $j = 1, ..., K$，门槛值是从观测值 X 的范围进行选取。

树的数量 S_{num} 一般要足够大可以使模型具有一定程度的灵活性，贝叶斯下沉先验可以用来避免过拟合。

为了解释贝叶斯累加回归树的原理，考虑仅有一个树的模型。模型可写为：

$$y = l(X|\mathcal{T}, \mu) + \varepsilon. \tag{4.68}$$

模型的条件均值为：

$$E(y|x) = l(X|\mathcal{T}, \mu) = \sum_{n=1}^{b} \mu_n \mathbb{I}(X \in S_n). \tag{4.69}$$

当模型设置较为简单时，说明条件均值反应出的变异较少，对 y 的波动解释性较差，因此这是一个弱学习器。因此，可以将多个简单树进行汇总（是由贝叶斯下沉可证）。同时，使用正则化来避免过拟合。

嵌套模型规范

文章主要聚焦于不同的模型规范在选择 F, G, z_t 是有差异的。第一个模型为多变量的非参数 VAR 模型，假设 $G(z_t) = 0_M$，模型可缩减为：

$$y_t = F(x_t) + \varepsilon_t. \tag{4.70}$$

这表示 y_t, x_t 之间的非参数关系，且 z_t 对 y_t 没有影响。

第二个模型假设 $z_t = x_t$,$G(x_t)$ 未知且非线性，且 $F(x_t)$ 线性且取决于 $M \times K$ 的系数矩阵 A。模型为：

$$y_t = Ax_t + g(x_t) + \varepsilon_t \tag{4.71}$$

其中 $G(x_t)$ 使用 BART 进行估计。该模型是假设波动率服从非线性关系，文章使用混合 BART 模型。

若设置 $z_t = (\eta'_{t-1}, ..., \eta'_{t-p})'$ 以及 $F(x_t) = Ax_t$, 这说明简约形式的波动 η_t 于其滞后项存在非线性关系。

最后一个模型，文章假设 $\Sigma_t = H_t$ 为对角矩阵（其中 $0_{M\times Q}$），这说明波动 ε_t 为相互独立。则模型可写为：第一个方程：

$$y_{1t} = f_1(x_t) + \varepsilon_{1t} \tag{4.72}$$

$$y_{2t} = f_2(x_t) + g_2(\varepsilon_{1t}) + \varepsilon_{2t} \tag{4.73}$$

$$y_{it} = f_i(x_t) + g_i(r_{it}) + \varepsilon_{it} \tag{4.74}$$

其中 $r_{it} = (\varepsilon_{1t}, ..., \varepsilon_{i-1,t})$ 为 $(i-1)$ 维波动向量。

模型的可加异方差性

之前并未谈论 u_i, v_j 的具体形式以及 w_t 的选择，实际上是将 $w_t = 1$ 以及 u_i, v_j 为线性函数，即同方差模型。文章考虑使用 hBART 模型来构造条件异方差特征，函数 u_i, v_j 使用 BART 进行近似：

$$u_i(w) \approx \sum_{s=1}^{S} l_{is}(w|\mathcal{T}_{is}^u, \mu_{is}^u) \ for \ i = 1, ..., Q \tag{4.75}$$

$$v_j(w) \approx \sum_{s=1}^{S} l_{is}(w|\mathcal{T}_{is}^v, \mu_{is}^v) \ for \ i = 1, ..., M \tag{4.76}$$

称模型为模型规范（因子）hBART 模型，它假设潜在因子 δ_t 和测量误差 e_t 根据 BART 规范是条件异方差。相比 SV 模型是由对数正态分布控制的序列，该方法更接近（非线性）广义自回归条件异方差模型 (GARCH)。

选择合适的预测元 w_t 是必要的。本文使用 $w_t = (t, x_t')'$，该设置不仅可以使模型具有非线性特征，还使得滞后项不仅影响其条件均值，同时影响其波动率。同时，我们选择的 w_t 使得我们可以进行多步预测误差协方差矩阵。更精确的是，可以通过式（1）式（2）提取提前一步预测分布 y_{T+1}，并计算 H_{T+2} 和 Ω_{T+2}, 根据 $w_{T+2} = (T+2, \hat{y}_{T+1}', y_T', ..., y_{T-p+1}').H_{T+2}$ 同样。

贝叶斯推断

文章使用贝叶斯方法来估计模型。尽管传统方法对于某些非参数方法是有效的，但下沉算法因有助于进行宏观经济预测更为人所知。文章先验设置与 Huber(2023) 相近。先验设置与树结构 $\mathcal{T}_{is}^j$ 和终点参数 μ_{is}^j 相关。Chipman et al. (2010) 与 Chipman et al. (1998) 提供了一种对树结构和终节点参数的基准先验，该先验对于高维数据以及样本外预测有非常不错的效果。

对树和终结点参数的先验 该先验不是直接作用在树上，而是作用在树产生的随机过程上。这个先验有三个特点，第一，该先验与非终结点的深度为 $d = 1, ...$ 的节点的生成两个子节点的概率相关。

$$\frac{\alpha}{(1+d)^\beta} \tag{4.77}$$

其中 α, β 为超参数。

文章设置 $\alpha = 0.95$, $\beta = 2$。有学者推荐该值，认为这种设置可以有效处理高维数据。

第二个特点为该先验与使用分离定律的变量选择相关。此处使用离散均匀分布先验，即暗示了对变量选择没有先验的偏好。第三点是对分离定律的具体门槛值相关。对此，文章同样使用均匀先验分布设置分离值的范围。

对于均匀值参数，文章使用独立高斯先验：

$$\mu_{is}^j \sim N(0, \phi_{is}^j), \ for \ k = 1, ..., b_{is}^j. \tag{4.78}$$

其中先验协方差 ϕ_{is}^j 基于数据来设置。具体形式为：

$$\sqrt{\phi_{is,k}^j} = \frac{max(z_i^j) - min(z_i^j)}{2\gamma\sqrt{S_{num}}} \tag{4.79}$$

文章对于所有的先验超参数设置都是相同的。

全条件后验模拟

后验与预测推断使用 MCMC 方法进行抽样。模型参数的后验分布可以（近似）通过 Metropolis-Hastings (MH) 方法进行获取。

第 i 个方程为：

$$y_{it} = f_i(x_t) + g_i(z_t) + \gamma_i\delta_t + e_{it}, \ e_{it} \sim N(0, e^{v_j(w_t)}) \tag{4.80}$$

升级树 文章使用贝叶斯后向拟合策略（参考 Chipman et al. (2010)），这一步对每个树抽样是取决于在保持其他 S-1 树的条件下进行的。其中令 $\widetilde{z}_{in} = z_i^f - \sum_{n\neq s} l_{is}(X|\mathcal{T}_{is}^f, \mu_{is}^f)$，$\widetilde{z}_{in}^g = z_i^g - \sum_{n\neq s} l_{is}(Z|\mathcal{T}_{is}^g, \mu_{is}^g)$，$\widetilde{z}_{in}^v = z_i^v - sum_{n\neq s} l_{is}(w|\mathcal{T}_{is}^v, \mu_{is}^v)$ 以及 $\widetilde{z}_{kn}^u = z_k^u -$
$\sum_{n\neq s} l_{ks}(w|\mathcal{T}_{ks}, \mu_{ks}^u)$ 表示排除第 n 个树的偏残差向量。

在偏残差向量 $\widetilde{z}_{in}^{\int}$ 的条件下 ($\int \in \{f, g\}$) 以及潜在误差协方差 $h_i = (v_i(w_1), ..., v_i(w_T))$ 全历史的条件下，从终节点参数 $\mu_{is}^{\int}$ 的边际分布中构造树结构 $\mathcal{T}_{in}^{\int}$。

$$p(\mathcal{T}_{in}^{\int}|\widetilde{z}_{in}^{\int}, h_i) \propto p(\mathcal{T}_{in}^{\int}) \int p(\widetilde{z}_{in}^{\int}|\mathcal{T}_{in}^{\int}, \mu_{in}^{\int}, h_i) p(\mu_{in}^{\int}|\mathcal{T}_{in}^{\int}\ h_i) d\mu_{in}^{\int}. \tag{4.81}$$

MH 算法指定了一个转换内核 $q(\mathcal{T}_{in}^{\int(a)})$，以先前接受的树结构为条件，用于生成新树，从四个指定的不同的概率分布中选择一个进行下一步行动：

生长：生长一个终端节点。这一步随机选择一个终端节点，并将这个节点基于随即分离定律分离成两个终端节点。这一步被选择的概率为 0.25。

修剪：修建一个终端节点。选择两个终端节点并将它们融合为一个，这一步被选择的概率为 0.25。

改变：随机选择一个内节点，并通过使用分离定律进行分离新的节点。该分离定律随即从分离变量的先验分布中（离散均匀分布）以及相应的门槛值。这一步被选择的概率为 0.4。

交换：交换一组母节点和子节点。这一步被选择的概率为 0.1。

这四步产生一个树 $\mathcal{T}_{in}^{\int *}$ 且其接受概率为：

$$min(\frac{p(\mathcal{T}_{in}^{\int *}|\widetilde{z}_{in}^{\int}, h_i)}{p(\mathcal{T}_{in}^{\int(a))}|\widetilde{z}_{in}^{\int}, h_i)} \frac{q(\mathcal{T}_{in}^{\int *}, \mathcal{T}_{in}^{\int(a)})}{q(\mathcal{T}_{in}^{\int(a)}, \mathcal{T}_{in}^{\int *)})}, 1) \tag{4.82}$$

MH 算法具有终端节点参数的独立性的优势，同时又避免可逆跳跃 MCMC 方法的计算问题。

升级误差协方差

为了升级误差协方差，文章使用了 Omori et al. (2007). 逼近的条件高斯模型。对 e_t 的第 i 个元素进行平方以及对数，可得：

$$log(e_{it}^2) = \sum_{is}^{S_{num}} l_i(w_t|\mathcal{T}_{is}^v, \mu_{is}^v) + \overline{w}_{it},\ \overline{w}_{it} \sim log(\chi_1^2). \tag{4.83}$$

$$log(e_{it}^2|\xi_t) = j \sim N(\sum_{is}^{S_{num}} l_i(w_t|\mathcal{T}_{is}^v, \mu_{is}^v) + m_j, \sigma_j^2) \tag{4.84}$$

其中，$\overline{w}_{it}$ 服从 $log(\chi_1^2)$ 分布。其中 $log(\chi_1^2)$ 分布，大多文献使用混合高斯分布来模拟该分布。具体做法是，将一些相互独立，且具有不同均值和方差的正态分布以不同概率叠加为 $log(\chi_1^2)$ 分布，具体参考 Omori（2007）的《Stochastic Volatility with Leverage: Fast and Efficient Likelihood Inference》。

在树形结构的条件下，文章考虑的所有不同类型 BART 模型的终端节点参数都很容易从独立的高斯分布中模拟出来。它们采用标准形式，类似于一个简单的截距模型树结构用于将观测值分配到不同的终端节点，然后将这些观测值用于计算后验矩。如果它包含严重的异常值 (例如在大流行期间观察到的异常值)，BART 很可能将它们分组在一起，相应的终端节点参数将具有后验方差，该方差等于异常值数量的倒数加上先验精度 (这将很低; 见式6.5)。因此，相应的后验方差会很大，导致预测区间更宽，从而在后验预测分布下观察到异常值的概率更高。

第五章　机器学习模型：深度神经网络

May 2024

神经网络通过层层特征变换，可以使变换后的特征包含与任务较相关的信息，从而实现分类、预测等目的。

5.1　单层神经网络

单层神经网络是最简单的神经网络模型。线性回归和 softmax 回归都是单层神经网络。

其中 softmax 回归适用于分类问题，输出变量 Y 代表样本属于某一类的概率。由于直接进行线性回归可能会出现负值、各类的概率和不等于 1 等情况.softmax 回归对线性回归结果 O 进行 softmax 运算，得到最终分类概率 Y。含义如下：

$$O = XW + b$$

$$\hat{Y} = \text{softmax}(O)$$

$$\hat{Y}_{ij} = \frac{\exp(O_{ij})}{\sum_k \exp(O_i k)}$$

其中 X 是输入元素，每行代表一个数据样本。O 和 Y 是 $n \times q$ 的矩阵，n 是样本数量，q 是类别数量。回归问题中，输出层的输出个数设为 1，并使用平方损失函数作为拟合标准。分类问题对输出层做 softmax 运算，使用交叉熵损失函数作为拟合标准。

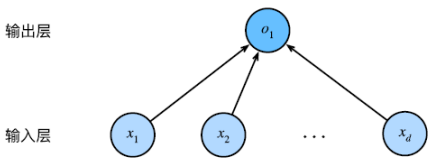


图 5.1: 单层神经网络线性回归

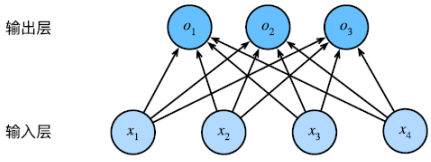


图 5.2: 单层神经网络 softmax 回归

5.2　多层感知机

5.2.1　基本构成

多层感知机（multilayer perceptron，MLP）在单层的基础上引入了一个到多个**隐藏层**（hidden layer），每个隐藏层可能包含多个隐藏单元（hidden unit）。并且多层感知机的隐藏层中的神经元和输入层各个

输入、输出层各个神经元完全连接，形成**全连接层**。

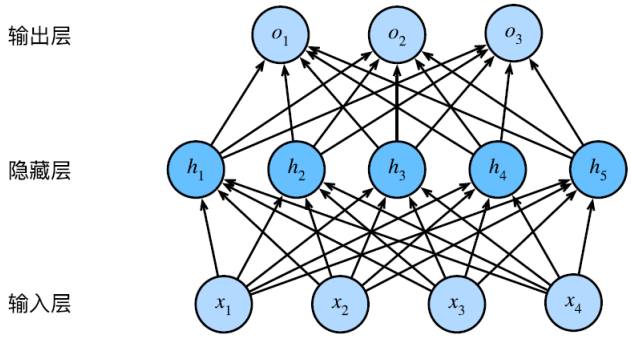


图 5.3: 具有 5 个隐藏单元的，单隐藏层的多层感知机

为发挥多层架构潜力，多层感知机的每个隐藏层的输出通过非线性的**激活函数**（activation function）进行变换。输出方式如下：

$$H = \phi(XW_h + b_h) \tag{5.1}$$

$$O = HW_o + b_o \tag{5.2}$$

其中 X 为输入元，是长度为 m 的向量。W_h 为权值，b_h 为偏置（bias）向量，H 为活性值（Activation），即隐藏元取值，O 为输出元取值。一个简单的二分类多层感知器机算法流程如下：

Algorithm 1 感知机学习算法,Frank Rosenblatt（1957）

初始化权重 $w = 0$, 偏置 $b = 0$
repeat
  从训练集随机选择一个样本 (x_i, y_i)
  计算感知机的输出 $a = \text{sign}(w \cdot x_i + b)$
  if $a \neq y_i$ **then**
    更新权重 $w' \leftarrow w + \eta \cdot y_i \cdot x_i$
    更新偏置 $b' \leftarrow b + \eta \cdot y_i$
  end if
until 训练次数达到要求
输出：分类网络参数 w 和 b
其中 η 为学习类率

通过层层堆叠全连接层，保证前一层的输出层节点数与当前层输入节点数匹配，可以得到任意层数的网络，如图5.4。深度理解神经网络离不开激活函数、过拟合、模型训练、模型稳定性和初始化模型参数等问题。

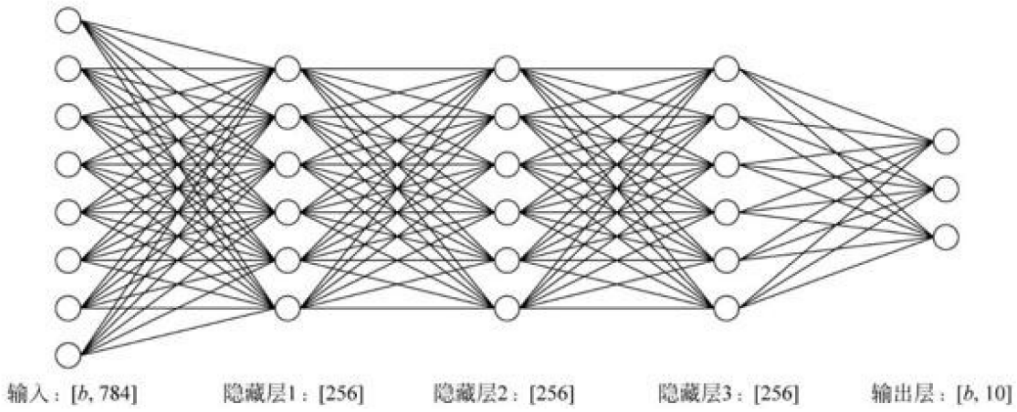


图 5.4: 四层全连接神经网络

5.2.2 激活函数

ϕ 为激活函数。常用的激活函数包含：

（1）ReLU 函数

ReLU 函数即修正线性单元函数（Rectified linear unit），该函数将小于 0 的值全部转为 0，只保留正值。该函数在求导时，要么让参数消失（导数为 0），要么让参数通过（导数为 1），减轻了梯度消失问题（gradient vanishing）的困扰，是最受欢迎的激活函数。

$$\mathrm{ReLU}(x) = \max(x, 0). \tag{5.3}$$

该函数有许多变体，如参数化 ReLU（parameterized ReLU，pReLU）函数。该函数使得某些参数值尽管是负的，也仍然可以通过。

$$\mathrm{pReLU}(x) = \max(0, x) + \alpha \min(0, x).$$

（2）sigmod 函数

sigmoid 函数将输　变换为区间 (0, 1) 上的输出，也被称为挤压函数 (squashing function)，经常被用于概率分布和信号强度类计算。且 sigmoid 函数连续可导，可以直接使用梯度下降算法进行优化。但是该函数在输入值较大或较小时容易出现梯度值接近于 0 的现象，即梯度弥散/消失，使参数将长时间得不到更新或更新过小。

$$\mathrm{sigmoid}(x) = \frac{1}{1 + \exp(-x)} \tag{5.4}$$

$$\frac{d}{dx}\mathrm{sigmoid}(x) = \frac{\exp(-x)}{(1 + \exp(-x))^2} = \mathrm{sigmoid}(x) \cdot (1 - \mathrm{sigmoid}(x))$$

（3）tanh 函数

与 sigmoid 函数类似，tanh(双曲正切) 函数的输出值也被转换到特定区间，不同的是，该函数将输出转换到 $(-1, 1)$ 区间。当输入在 0 附近时，接近线性变换。tanh 函数的公式如下：

$$\tanh(z) = \frac{1 - \exp(-2z)}{1 + \exp(-2z)} \tag{5.5}$$

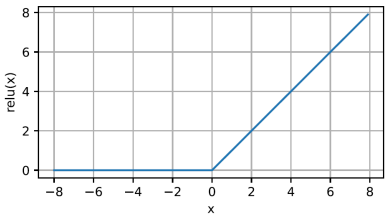


图 5.5: ReLU 函数

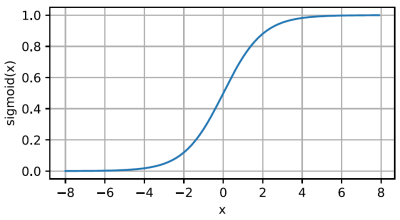


图 5.6: sigmoid 函数

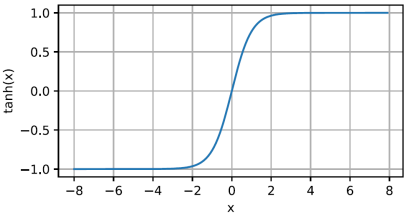


图 5.7: tanh 函数

5.2.3 过拟合

深度神经网络很容易出现过拟合问题，当出现过拟合问题，可以重新设计模型的容量，如降幅网络层数、减少网络参数量、添加正则化手段等。

常用的**正则化手段**是权重衰减（weight decay）法，指在原损失函数上添加正则项，如 L2 范数惩罚项。

还可以使用**丢弃法**（dropout）控制神经网络容量，即以一定的概率“丢弃”部分隐藏单元。以倒置丢弃法（inverted dropout）为例：

在该方法中，新的隐藏单元为

$$h_i^{'} = \frac{\delta_i}{1-p} h_i \tag{5.6}$$

其中 $p(\delta_i = 0) = p, p(\delta_i = 1) = 1 - p$ 且 $E(h_i^{'}) = h_i$。有 p 的概率隐藏元 h_i 会被清零，有 $1 - p$ 的概率会除以 $1 - p$ 做拉伸。该方法可以使模型不依赖于特定的神经元。不过，在测试模型时为了得到更加确定性的结果一般不使用丢弃法。

避免出现过拟合，还可以**提前停止训练**（early stopping）：具体来说，记录模型的验证准确率，并监控验证准确率的变化，当发现验证准确率连续多次不能提升时，可以预测可能已经达到了最适合的训练次数附近，从而提前终止训练。

5.2.4 模型训练

训练深度模型的基本方法是正向传播（forward-Propagation）和反向传播（back-propagation）。掌握算法的核心是理解反向传播和梯度下降方法。

(1) 正向传播

正向传播（Forward propagation）沿着从输入层到输出层的顺序，依次计算模型的中间变量和输出，最后一步是计算误差。假设输入样本是 $\mathbf{x} \in \mathbb{R}^d$, 并且假设隐藏层不包括偏置项。中间变量为：

$$\mathbf{z} = \mathbf{W}^{(1)}\mathbf{x}$$

其中 $\mathbf{W}^{(1)} \in \mathbb{R}^{h \times d}$ 是隐藏层的权重参数。将中间变量 $\mathbf{z} \in \mathbb{R}^h$ 通过激活函数 ϕ 后，我们得到长度为 h 的隐藏激活向量:

$$\mathbf{h} = \phi(\mathbf{z})$$

隐藏变量 $\mathbf{h}$ 也是一个中间变量。假设输出层的参数只有权重 $\mathbf{W}^{(2)} \in \mathbb{R}^{q \times h}$, 我们可以得到输出层变量, 它是一个长度为 q 的向量:

$$\mathbf{o} = \mathbf{W}^{(2)}\mathbf{h}$$

假设损失函数为 l ，样本标签为 y ，我们可以计算单个数据样本的损失项，

$$L = l(\mathbf{o}, y)$$

根据 L_2 正则化的定义，给定超参数 λ ，正则化项为

$$s = \frac{\lambda}{2}\left(\left\|\mathbf{W}^{(1)}\right\|_F^2 + \left\|\mathbf{W}^{(2)}\right\|_F^2\right)$$

其中矩阵的 Frobenius 范数是将矩阵展平为向量后应用的 L_2 范数。最后, 模型在给定数据样本上的正则化损失为:

$$J = L + s$$

J 为目标函数 (objective function)。

（2）反向传播

反向传播（Backward Propogation）指计算神经网络参数梯度的方法。该方法依据微积分中的链式法则，沿着输出层到输入层的顺序，依次计算并存储目标函数关于神经网络各层的中间变量和参数的梯度。基于反向传播，可以使用梯度下降（gradient descent，GD）算法更新参数。

以前文正向传播描述的框架为例，待估的权重矩阵为输入层到隐藏层的权重矩阵 W^1 和隐藏层到输出层的权重矩阵 W^2。J 为目标函数，L 为均方误差损失函数，s 为 L2 惩罚项。o 为输出层变量，z 为激活前中间变量，h 为激活后中间变量，即隐藏层取值。则可以算得，关于两个权重矩阵的梯度为：

$$\frac{\partial J}{\partial \mathbf{W}^{(1)}} = \operatorname{prod}\left(\frac{\partial J}{\partial \mathbf{z}}, \frac{\partial \mathbf{z}}{\partial \mathbf{W}^{(1)}}\right) + \operatorname{prod}\left(\frac{\partial J}{\partial s}, \frac{\partial s}{\partial \mathbf{W}^{(1)}}\right) = \frac{\partial J}{\partial \mathbf{z}}\mathbf{x}^\top + \lambda \mathbf{W}^{(1)} \tag{5.7}$$

$$\frac{\partial J}{\partial \mathbf{W}^{(2)}} = \operatorname{prod}\left(\frac{\partial J}{\partial \mathbf{o}}, \frac{\partial \mathbf{o}}{\partial \mathbf{W}^{(2)}}\right) + \operatorname{prod}\left(\frac{\partial J}{\partial s}, \frac{\partial s}{\partial \mathbf{W}^{(2)}}\right) = \frac{\partial J}{\partial \mathbf{o}}\mathbf{h}^\top + \lambda \mathbf{W}^{(2)} \tag{5.8}$$

其中

$$\frac{\partial J}{\partial \mathbf{o}} = \frac{\partial L}{\partial \mathbf{o}}$$

$$\frac{\partial J}{\partial \mathbf{z}} = \frac{\partial J}{\partial \mathbf{h}} \odot \phi'(z) = \frac{\partial J}{\partial \mathbf{o}}\frac{\partial o}{\partial \mathbf{h}} \odot \phi'(z) = W^{(2)T}\frac{\partial L}{\partial \mathbf{o}} \odot \phi'(z)$$

prod 指在进行必要的操作，如转置和互换位置后对输入做乘法运算。可以发现，计算出后一层的梯度核心变量 $\frac{\partial L}{\partial \mathbf{o}}$ 后，可以计算前一层的梯度。反向传播算法使拟合一个复杂模型在计算上可行，在早期被认为是神经网络的一次突破。

（3）梯度下降算法

在神经网络方法中，由于有 n 个训练样本，目标函数为 $f(x) = \sum_{i=1}^n f_i(x)$，其中 $\nabla f_i(x) = \frac{\partial J}{\partial \mathbf{W}^{(k)}}$。因此梯度为

$$\nabla f(x) = \frac{1}{n}\sum_{i=1}^n \nabla f_i(x)$$

梯度下降即不断通过下列方式迭代 x，直至梯度值已经足够小，或已经达到预先设定的迭代次数等。

$$x \leftarrow x - \eta \nabla f(x)$$

在大样本中，为减少计算开销，经常会使用**随机梯度下降**、**小批量随机梯度下降**等，这类方法只使用一小部分样本来估计梯度，并更新模型参数，可以显著降低单次迭代的计算成本。（但可能会增加收敛到最小损失的总迭代次数）。其中，随机梯度下降在每次迭代中，随机均匀采用一个样本索引 $i \in \{1, \ldots, n\}$, 计算梯度 $\nabla f_i(x)$ 并对 x 进行迭代, 降低了计算开销：$x \leftarrow x - \eta \nabla f_i(x)$. 小批量随机梯度下降方法使用小批量的样本，对参数进行更新。

Algorithm 2 一个简单的神经网络优化流程

初始化
repeat
　反向计算每层的梯度
　使用梯度下降算法对每层权重完成一次参数更新
　计算均方误
until 训练次数达到要求

5.2.5 深度模型的稳定性问题

深度模型面临衰减（vanishing）和爆炸（explosion）的稳定性问题。即当权重参数设置不当时，随着层数的增多，对后面层的计算结果会出现衰减/爆炸，包括当前层的结果的值和梯度的值，导致模型的稳定性变差。

5.2.6 初始化模型参数

神经网络常常需要随机初始化模型参数，特别是权重参数初始化。原因是，若每个隐藏单元的权重参数相同，并且激活函数相同，那么每个隐藏单元计算的输出值将相同，本质上将只有一个隐藏单元在发挥作用。常见的随机初始化方法包括：假设权重参数服从正态分布；假设权重参数服从均匀分布；每个元素都从相应分布中随机采用。还有一种常用的随机化初始方法为 Xavier 随机初始化方法，该方法从以下均匀分布中对权重参数中的每个元素进行随机采用：

$$U\left(-\sqrt{\frac{6}{a+b}}, \sqrt{\frac{6}{a+b}}\right) \tag{5.9}$$

其中 a 是全连接层的输入个数，b 是输出个数。可以使得每层输出的方差不受该层输入个数的影响，每层梯度的方差也不受该层输出个数的影响。证明过程如下：

假设一个没有非线性元素的全连接层输出（例如, 没有隐藏变量）o_i 的分布特征。对于该层 n_{in} 输入 x_j 及其相关权重 w_{ij}, 输出由下式给出

$$o_i = \sum_{j=1}^{n_{\text{t}}} w_{ij} x_j.$$

权重 w_{ij} 都是从同一分布中独立抽取的。假设该分布具有零均值和方差 σ^2 。并假设层 x_j 的输入也具有零均值和方差 γ^2, 并且它们独立于 w_{ij} 且彼此独立。在这种情况下, 我们可以按如下方式计算 o_i

的平均值和方差:

$$\begin{aligned} E\left[o_i\right] &= \sum_{j=1}^{n_{\text{in}}} E\left[w_{ij} x_j\right] \\ &= \sum_{j=1}^{n_{\text{m}}} E\left[w_{ij}\right] E\left[x_j\right] \\ &= 0, \\ \operatorname{Var}\left[o_i\right] &= E\left[o_i^2\right] - \left(E\left[o_i\right]\right)^2 \\ &= \sum_{j=1}^{n_{\text{m}}} E\left[w_{ij}^2 x_j^2\right] - 0 \\ &= \sum_{j=1}^{n_{\text{in}}} E\left[w_{ij}^2\right] E\left[x_j^2\right] \\ &= n_{\text{in}}\, \sigma^2 \gamma^2. \end{aligned}$$

保持方差不变的一种方法是设置 $n_{\text{in}}\sigma^2 = 1$ 。考虑反向传播过程，面临的问题类似，尽管梯度是从更靠近输出的层传播的。使用与前向传播相同的推断, 可以看到, 除非 $n_{\text{out}}\,\sigma^2 = 1$, 否则梯度的方差可能会增大, 其中 n_{out} 是该层的输出的数量。这使得我们进退两难：我们不可能同时满足这两个条件。相反，我们可以满足:

$$\frac{1}{2}\left(n_{\text{in}} + n_{\text{out}}\right)\sigma^2 = 1 \text{ 或等价于} \sigma = \sqrt{\frac{2}{n_{\text{in}} + n_{\text{out}}}}.$$

这就是现在标准且实用的 Xavier 初始化的基础，它以其提出者 (Glorot and Bengio, 2010) 第一作者的名字命名。通常, Xavier 初始化从均值为零, 方差 $\sigma^2 = \frac{2}{n_{\text{min}} + n_{\text{out}}}$ 的高斯分布中采样权重。也可以按照上文所述的均匀分布进行采用。

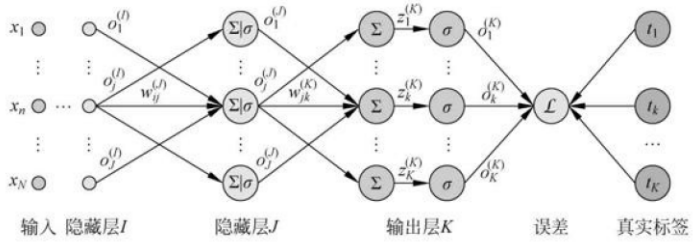


图 5.8: 多层反向传播算法示意图

5.2.7 一个较详细的反向传播计算示例

以图5.8为例推导隐藏层的梯度传播规律。可以算出输出层的偏导数公式:

$$\frac{\partial \mathcal{L}}{\partial w_{jk}} = \left(o_k - t_k\right)\frac{\partial o_k}{\partial w_{jk}} = \left(o_k - t_k\right)\frac{\partial \sigma(z_k)}{\partial z_k}\frac{\partial z_k}{\partial w_{jk}} = \left(o_k - t_k\right) o_k \left(1 - o_k\right) o_j = \delta_k o_j$$

其中 o_k 是输出值，z_k 是未激活值，o_j 是上一层的激活值，t_k 是真实值，$\mathcal{L} = \frac{1}{2}\sum_{i=1}^{K}(o_i^{(K)} - t_i)^2$。使用的激活函数为 sigmoid 函数，因此激活函数导数为：$\sigma(z_k)(1 - \sigma(z_k)) = o_k\left(1 - o_k\right)$。

考虑倒数第二层的偏导数 $\frac{\partial \mathcal{L}}{\partial w_{ij}}$, 如图5.8所示, 输出层节点数为 K, 输出为 $\boldsymbol{o}^{(K)}=\left[o_1^{(K)}, o_2^{(K)}, \cdots, o_K^{(K)}\right]$; 倒数第二层节点数为 J, 输出为 $\boldsymbol{o}^{(J)}=\left[o_1^{(J)}, o_2^{(J)}, \cdots, o_J^{(J)}\right]$; 倒数第三层的节点数为 I, 输出为

$$\boldsymbol{o}^{(I)}=\left[o_1^{(I)}, o_2^{(I)}, \cdots, o_I^{(I)}\right]$$

为了表达简洁, 部分变量的上标有时会省略掉。首先将均方误差函数展开:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}}=\frac{\partial}{\partial w_{ij}} \frac{1}{2} \sum_k\left(o_k-t_k\right)^2$$

由于 $\mathcal{L}$ 通过每个输出节点 $\mathrm{o_k}$ 与 $\mathrm{w_{ij}}$ 相关联, 故此处不能去掉求和符号, 运用链式法则将均方差函数拆解:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}}=\sum_k\left(o_k-t_k\right) \frac{\partial}{\partial w_{ij}} o_k$$

将 $\mathrm{o_k}=\sigma\left(\mathrm{z_k}\right)$ 代入可得:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}}=\sum_k\left(o_k-t_k\right) \frac{\partial}{\partial w_i} \sigma\left(z_k\right)$$

利用 Sigmoid 函数的导数 $\sigma'=\sigma(1-\sigma)$ 进一步分解为:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}}=\sum_t\left(o_i-t_t\right) \sigma\left(z_i\right)\left(1-\sigma\left(z_t\right)\right) \frac{\partial z_i}{\partial w_{ij}}$$

将 $\sigma\left(z_{\mathrm{k}}\right)$ 写回 $\mathrm{o_k}$ 形式, 并利用链式法则, 将 $\frac{\partial z_k}{\partial w_{ij}}$ 分解为:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}}=\sum_k\left(o_k-t_k\right) o_k\left(1-o_k\right) \frac{\partial z_k}{\partial o_j} \cdot \frac{\partial o_j}{\partial w_{ij}}$$

其中 $\frac{\partial z_k}{\partial o_j}=\mathrm{w_{jk}}$, 因此:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}}=\sum_k\left(o_k-t_k\right) o_k\left(1-o_k\right) w_{jk} \frac{\partial o_j}{\partial w_{ij}}$$

考虑到 $\frac{\partial o_j}{\partial w_{ij}}$ 与 k 无关, 可提取公共项为:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}}=\frac{\partial o_j}{\partial w_{i_j}} \sum_k\left(o_k-t_k\right) o_k\left(1-o_k\right) w_{jk}$$

进一步利用 $\mathrm{O_j}=\sigma\left(\mathrm{z_j}\right)$, 并利用 Sigmoid 导数 $\sigma'=\sigma(1-\sigma)$, 将 $\frac{\partial o_j}{\partial w_{ij}}$ 拆分为:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}}=o_j\left(1-o_j\right) \frac{\partial z_j}{\partial w_{ij}} \sum_k\left(o_k-t_k\right) o_k\left(1-o_k\right) w_{jk}$$

其中 $\frac{\partial z_j}{\partial w_{ij}}$ 的导数可直接推导出为 $\mathrm{o_i}$, 上式可写为:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}}=o_j\left(1-o_j\right) o_i \sum_k \underbrace{\left(o_k-t_k\right) o_k\left(1-o_k\right)}_{\delta_k^{(K)}} w_{jk}$$

其中 $\delta_k^{(K)}=\left(o_{\mathrm{k}}-\mathrm{t_k}\right) \mathrm{o_k}\left(1-\mathrm{o_k}\right)$, 则 $\frac{\partial \mathcal{L}}{\partial w_{ij}}$ 的表达式可简写为:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}}=o_j\left(1-o_j\right) o_i \sum_k^K \delta_k^{(K)} w_{jk}$$

类似地，仿照输出层 $\frac{\partial \mathcal{L}}{\partial w_{jk}} = \delta_k^{(K)} x_j$ 的书写方式，将 δ_j^J 定义为:

$$\delta_j^J \triangleq o_j \left(1 - o_j\right) \sum_k \delta_k^{(K)} w_{jk}$$

此时 $\frac{\partial \mathcal{L}}{\partial w_{ij}}$ 可以写为当前连接的起始节点的输出值 o_i 与终止节点 j 的梯度变量信息 $\delta_j^{(J)}$ 的简单相乘运算:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \delta_j^{(J)} o_i^{(I)}$$

可以看到, 通过定义 δ 变量, 每一层的梯度表达式变得更加清晰简洁, 其中 δ 可以简单理解为当前连接 $\mathrm{w_{i}}_j$ 对误差函数的贡献值。下面来小结每层的偏导数的传播规律。输出层:

$$\frac{\partial \mathcal{L}}{\partial w_{jk}} = \hat{\delta}_k^{(K)} o_j$$
$$\delta_k^{(K)} = o_k \left(1 - o_k\right) \left(o_k - t_k\right)$$

倒数第二层:

$$\frac{\partial \mathcal{L}}{\partial w_{ij}} = \delta_j^{(j)} o_i$$
$$\delta_j^{(j)} = o_j \left(1 - o_j\right) \sum_k \delta_k^{(K)} w_{jk}$$

倒数第三层:

$$\frac{\partial \mathcal{L}}{\partial w_{ni}} = \hat{\delta}_i^{(I)} o_n$$
$$\delta_i^I = o_i \left(1 - o_i\right) \sum_j \delta_j^{(j)} w_{ij}$$

其中 o_{n} 为倒数第三层的输入, 即倒数第四层的输出。依照此规律, 只需要循环迭代计算每一层每个节点的 $\delta_k^{(K)}$ $\delta_j^{(J)}$ $\delta_i^{(l)}$: 等值，即可求得当前层的偏导数, 从而得到每层权值矩阵 W 的梯度, 再通过梯度下降算法迭代优化网络参数即可。至此, 反向传播算法介绍完毕。

5.3 卷积神经网络

卷积神经网络（convolution neural network）指使用卷积对特征进行提取，含有卷积层（convolution layer）的网络。

全连接层**较高的内存占用量**严重限制了神经网络朝着更大规模、更深层数方向的发展。卷积神经网络将卷积核在输入数据上进行滑动，每次计算与卷积核重叠部分的点的乘和。这种方法利用了数据的**局部相关性**，同时基于**权值共享**思想，对特征进行抽象，将参数量减少，也使得网络对输入数据的变化更加鲁棒和准确。具体来说，卷积网络将参数量从 $I \times J$ 个，减小到了 $K \times K$ 个。其中 I 是全连接神经网络的输入层单元个数，J 是全连接层神经网络的输出层单元个数，K 是卷积神经网络的权值矩阵维度。

- 局部相关性理解：例如股票数据相邻日期的数据相关性强且近段数据趋势预测能力强、图片上距离近的像素点关联度大等。利用局部相关性指关注和自己距离较近的部分节点，忽略较远的节点。

- 权值共享理解：即对于每个输出节点，都使用相同的权值矩阵。

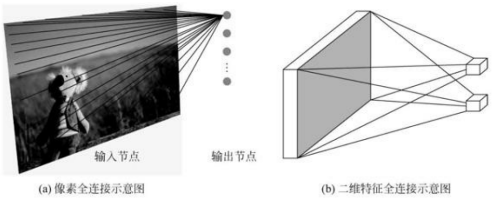


图 5.9: 二维特征全连接

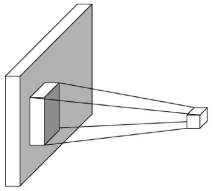


图 5.10: 局部连接的网络层

5.3.1 卷积运算（convolution）

一维连续卷积运算：

$$(f \otimes g)(n) = \int_{-\infty}^{\infty} f(\tau)g(n-\tau)d\tau$$

一维离散卷积运算：

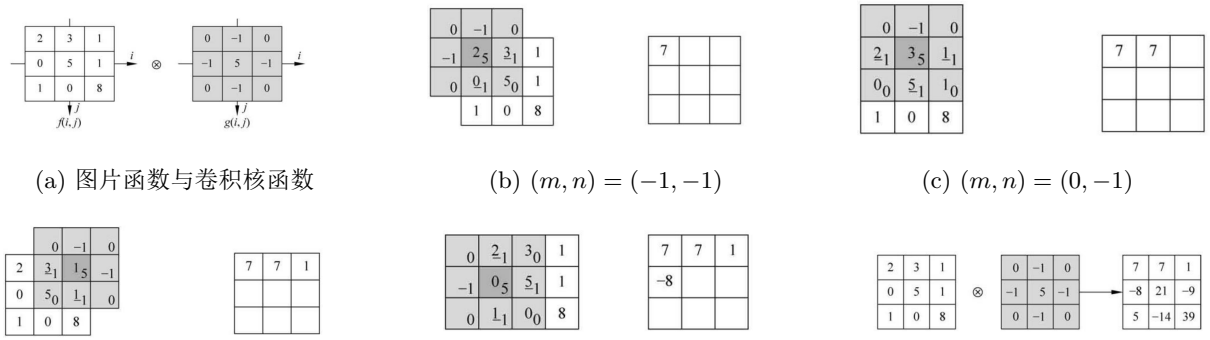
$$(f \otimes g)(n) = \sum_{\tau=-\infty}^{\infty} f(\tau)g(n-\tau)$$

二维离散卷积运算：

$$[f \otimes g](m,n) = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} f(i,j)g(m-i,n-j)$$

其中函数 g 被称为**卷积核**（Kerner），卷积运算即指包含了平移操作“卷”和积分操作“积”的运算。在上述定义中，对卷积核首先进行翻转运算，变为 $g(-\tau)$ 或 $g(-i\ -j)$, 再进行平移运算，变为 $g(n-\tau)$ 或 $g(m-i\ n-j)$。对该运算的详细理解可以参考二维离散卷积运算图解或单通道卷积运算示意图。

二维离散卷积运算核心：从左上方逐步向右、向下移动**卷积核**，并与图片对应位置处的**局部区域**（**感受野**，ReceptiveField）像素相乘累加，得到此位置的输出值，从而**提取每个位置的像素特征**。卷积核（Kerner）是行、列大小为 k 的权值矩阵，也称为 Fliter、Weight 等。感受野是输入矩阵/特征图上的大小为 k 的窗口。直至卷积核移动至最右下方，完成卷积运算。



(a) 图片函数与卷积核函数　(b) $(m,n)=(-1,-1)$　(c) $(m,n)=(0,-1)$

(d) $(m,n)=(1,-1)$　(e) $(m,n)=(-1,0)$　(f) 计算结果

图 5.11: 二维离散卷积运算图解

原图效果	锐化效果	模糊效果	边缘提取效果
			
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0.0625 & 0.125 & 0.0625 \\ 0.125 & 0.25 & 0.125 \\ 0.0625 & 0.125 & 0.0625 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

图 5.12: 常见卷积核及其效果

5.3.2 步长与填充

（1）步长

步长（Stride）指感受野窗口/卷积窗口每次滑动的行数和列数。通过设定步长 s，可以有效控制信息密度的提取。步长越小，越有利于提取更多特征信息，输出特征的尺寸越大。步长越大，越有利于减少计算代价，过滤多余信息。

（2）填充

填充（padding）指在输入数组的高和宽两侧填充元素（通常是 0 元素）。主要目的包括：a. 保持输出的特征图（feature map）与输入数组的空间维度相同，避免在进行多层次卷积时迅速丢失空间信息。b. 允许边缘像素在卷积中被频繁使用，避免丢失信息。c. 便于维度的处理，例如输入数组为奇数维，移动步长为 2，不填充会难以进行。

输入数组维度为 $n_h \times n_w$，在高的两侧一共填充 p_h 行，在宽的两侧一共填充 p_w 列，并设高上移动步长为 s_h, 宽上的移动步长为 s_w，卷积核的维度为 $k_h \times k_w$, 则输出形状为。

$$n'_h = \left\lfloor \frac{n_h + p_h - k_h + s_h}{s_h} \right\rfloor \tag{5.10}$$

$$n'_w = \left\lfloor \frac{n_w + p_w - k_w + s_w}{s_w} \right\rfloor \tag{5.11}$$

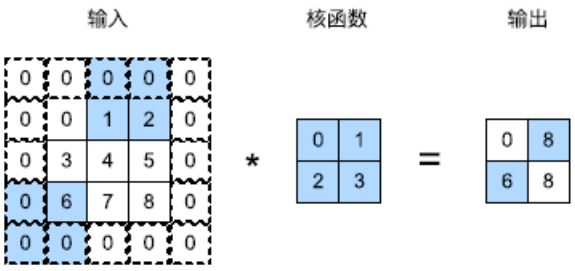


图 5.13: 垂直步幅为 3， 平步幅为 2 的运算

5.3.3 池化层

池化层（PoolingLayer）可以实现尺寸压缩功能，也称汇聚层、向下采样层等。池化层直接计算感受野窗口/池化窗口内元素的最大值（最大池化层，MaxPooling）或平均值（平均池化层，AveragePooling）。

- 池化层同样可以设定步长和进行填充
- 对于多通道输入数据，池化层对每个通道分别池化后直接输出，即池化层的输出通道与输入通道相等。
- 池化层是确定的，没有需要学习的参数，计算简单，应用广泛。
- 池化层可以降低卷积层对位置的敏感性，同时降低对空间降采样表 的敏感性。

5.3.4 CNN 的梯度传播

以一个输入为 3×3 的单通道，卷积核为 2×2 的单卷积核为例。说明卷积网络依然有网络可导性，且梯度推导并不复杂。

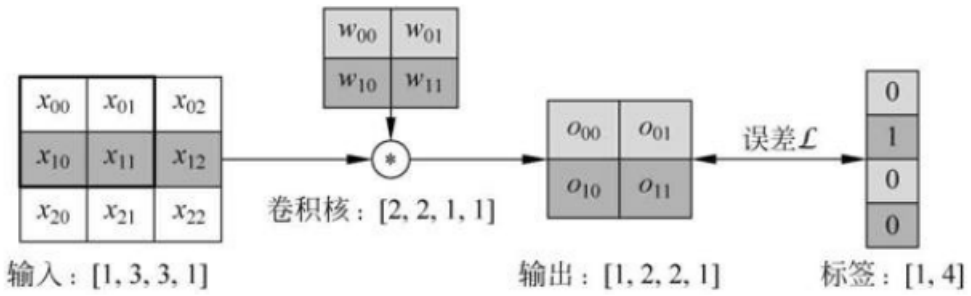


图 5.14: 卷积层梯度传播举例

输出元素（张量）O 的表达形式为:

$$\begin{aligned}
\mathrm{o}_{00} &= \mathrm{x}_{00}\mathrm{w}_{00} + \mathrm{x}_{01}\mathrm{w}_{01} + \mathrm{x}_{10}\mathrm{w}_{10} + \mathrm{x}_{11}\mathrm{w}_{11} + \mathrm{b} \\
\mathrm{o}_{01} &= \mathrm{x}_{01}\mathrm{w}_{00} + \mathrm{x}_{02}\mathrm{w}_{01} + \mathrm{x}_{11}\mathrm{w}_{10} + \mathrm{x}_{12}\mathrm{w}_{11} + \mathrm{b} \\
\mathrm{o}_{10} &= \mathrm{x}_{10}\mathrm{w}_{00} + \mathrm{x}_{11}\mathrm{w}_{01} + \mathrm{x}_{20}\mathrm{w}_{10} + \mathrm{x}_{21}\mathrm{w}_{11} + \mathrm{b} \\
\mathrm{o}_{11} &= \mathrm{x}_{11}\ \mathrm{W}_{00} + \mathrm{x}_{12}\mathrm{w}_{01} + \mathrm{x}_{21}\mathrm{w}_{10} + \mathrm{x}_{22}\mathrm{w}_{11} + \mathrm{b}
\end{aligned}$$

以 w_{00} 的梯度计算为例, 通过链式法则分解:

$$\frac{\partial \mathcal{L}}{\partial w_{00}} = \sum_{i \in \{00,01,10,11\}} \frac{\partial \mathcal{L}}{\partial o_i} \frac{\partial o_i}{\partial w_{00}}$$

其中 $\frac{\partial \mathcal{L}}{\partial o_i}$ 可直接由误差函数推导出来, 直接来考虑 $\frac{\partial o_i}{\partial w_i}$, 例如:

$$\frac{\partial o_{00}}{\partial w_{00}} = \frac{\partial \left(x_{00}w_{00} + x_{01}w_{01} + x_{10}w_{10} + x_{11}w_{11} + b\right)}{w_{00}} = x_{00}$$

同样的方法, 可以推导出:

$$\begin{aligned}
\frac{\partial o_{01}}{\partial w_{00}} &= \frac{\partial \left(x_{11}w_{e0} + x_{02}w_{01} + x_{11}w_{10} + x_{12}w_{11} + b\right)}{w_{00}} = x_{01} \\
\frac{\partial o_{10}}{\partial w_{00}} &= \frac{\partial \left(x_{10}w_{00} + x_{11}w_{01} + x_{20}w_{10} + x_{21}w_{11} + b\right)}{w_{00}} = x_{10} \\
\frac{\partial o_{11}}{\partial w_{00}} &= \frac{\partial \left(x_{11}w_{00} + x_{12}w_{01} + x_{21}w_{10} + x_{22}w_{11} + b\right)}{w_{00}} = x_{11}
\end{aligned}$$

5.3.5 一个经典卷积网络：LeNet 网络

该网络是最早发布的卷积神经网络之一，被用于识别手写数字图像的识别，也是第一篇通过反向传播训练卷积神经网络的研究。由 Yann LeCun 等人在 1989 年提出，因而命名为 LeNet。

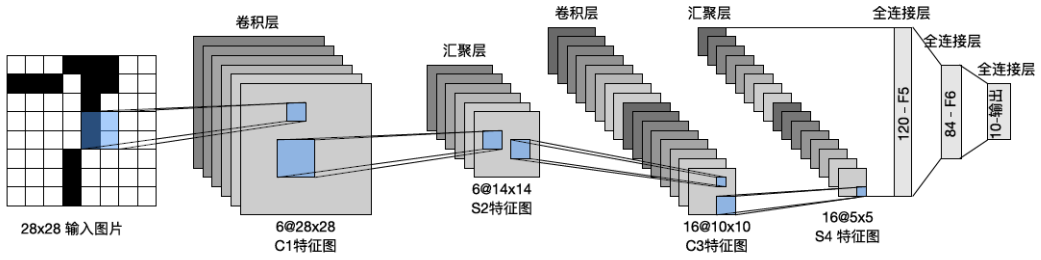


图 5.15: LeNet 数据流，输入为手写数字，输出为 10 种可能结果的概率

该网络由两个卷积层快和三个全连接层组成,每个卷积块中的基本单元是 个卷积层、 个 sigmoid 激活函数和平均池化层。每个卷积层使 5×5 卷积核计算，并使用 sigmoid 激活函数进行输出。第 卷积层有 6 个输出通道， 第 个卷积层有 16 个输出通道，增加输出通道是为使两个卷积层的参数尺

寸类似。池化层使用 2×2 的窗口，且步幅为 2，将维数减少 4 倍。卷积的输出形状由批量 、通道数、 度、宽度决定。

当卷积层传入全连接层时，将每个样本/小批量样本的特征展平（flatten）后进行计算。三个全连接层，分别有 120、84 和 10 个输出，10 对应了输出类别个数。

表 5.1: LeNet 网络参数量

层	卷积层 1	卷积层 2	全连接层 1	全连接层 2	全连接层 3
参数量	156	2416	48120	10164	850

* 参数总数 =(滤波器高度 × 滤波器宽度 × 输入通道数 + 偏置项数) × 输出通道数

5.4 循环神经网络 RNN

5.4.1 基本循环神经网络

传统的神经网络模型是从输入层到隐含层再到输出层的全连接，且同层的节点之间是无连接，网络的传播也是顺序的，但这种普通的网络结构对于许多问题却显得无能为力。例如，在自然语言处理中，如果要预测下一个单词，就需要知道前面的部分单词，因为一个句子中的单词之间是相互联系的，即有语义。这就需要一种新的神经网络，即循环神经网络 RNN，循环神经网络对于序列化的数据有很强的模型拟合能力。具体的结构为：循环神经网络在隐含层会对之前的信息进行存储记忆，然后输入到当前计算的隐含层单元中，也就是隐含层的内部节点不再是相互独立的，而是互相有消息传递。隐含层的输入不仅可以由两部分组成，输入层的输出和隐含层上一时刻的输出，即隐含层内的节点自连；隐含层的输入还可以由三部分组成，输入层的输出、隐含层上一时刻的输出、上一隐含层的状态，即隐含层内的节点不仅自连还互连。结构如图5.16所示。

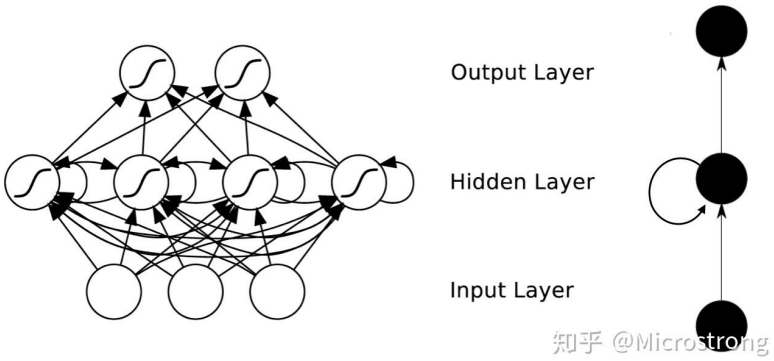


图 5.16: 循环神经网络结构图

在图 1 中，可以看到隐含层节点间有消息的相互传递。为了更简单的理解，现在我们将 RNN 在时间坐标轴上展开成一个全神经网络，如图5.17所示。例如，对一个包含 3 个单词的语句，那么展开的网络便是一个有 3 层的神经网络，每一层代表一个单词。

对于图5.17的网络，计算过程如下：

- x_t 表示第 t 步 (step) 的输入。比如 x_1 为第二个词的词向量 (x_0 为第一个词)；

- s_t 为隐藏层的第 t 步的状态，它是网络的记忆单元。s_t 根据当前输入层的输出与上一时刻隐藏层的状态进行计算，如公式 1 所示。其中，U 是输入层的连接矩阵，W 是上一时刻隐含层到下一时刻隐含层的权重矩阵，f(x) 一般是非线性的激活函数，如 tanh 或 ReLU。

$$s_t = f(Ux_t + Ws_{t-1}) \tag{5.12}$$

- o_t 是是第 t 步的输出。输出层是全连接层，即它的每个节点和隐含层的每个节点都互相连接，V 是输出层的连接矩阵，g(x) 是激活函数。

$$o_t = g(V * s_t) \tag{5.13}$$

如果将 (1) 式循环带入 (2) 式可得：

$$o_t = g(Vs_t)$$

$$= Vf(Ux_t + Ws_{t-1})$$

$$= Vf(Ux_t + Wf(Ux_{t-1} + s_{t-2}))$$

$$= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Us_{t-2} + Ws_{t-3})))$$

$$= Vf(Ux_t + Wf(Ux_{t-1} + Wf(Us_{t-2} + Wf(Ux_{t-3} + ...))))$$

由式 (3) 可以看出，循环神经网络的输出值与前面多个时刻的历史输入值有关，这就是为何循环神经网络能够往前看任意多个输入值的原因，也就是为何循环神经网络能够对序列数据建模的原因。

在图 2 中，我们展示了一个单向循环神经网络，但是单向循环神经网络也有不足之处。从单向的结构中可以知道它的下一时刻预测输出是根据前面多个时刻的输入共同影响的，而有些时候预测可能需要由前面若干输入和后面若干输出共同决定，这样才会更加准确。

5.4.2 RNN 的几种架构

由于 RNN 对于序列化的数据有很强的模型拟合能力，因此在不同的应用场景中可以对 RNN 组合形成不同的网络架构。

1 to N

这种情况有两种方式，一种是只在序列开始进行输入计算, 如图5.18图5.19

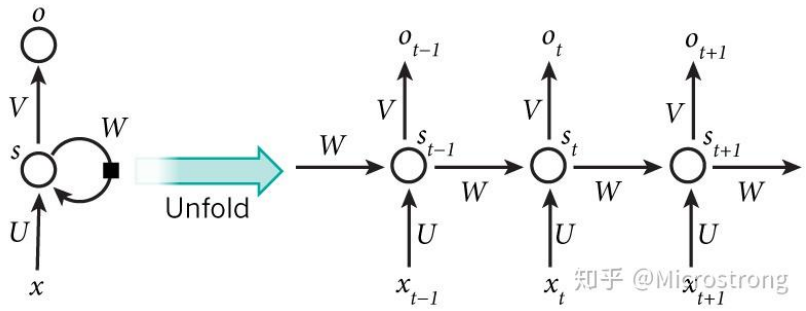


图 5.17: 循环神经网络展开图

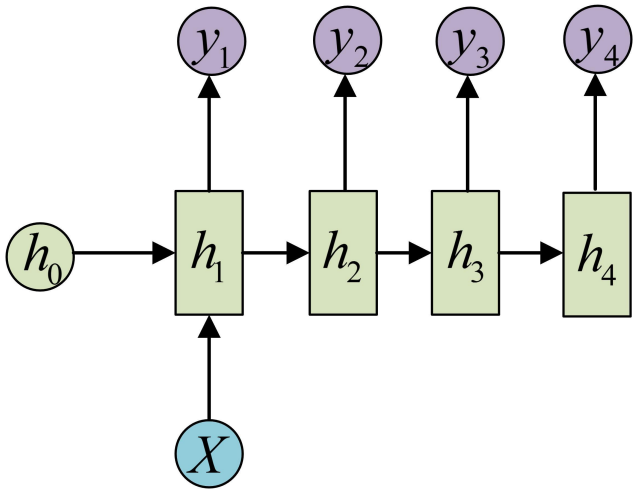


图 5.18: 1 to N（1）

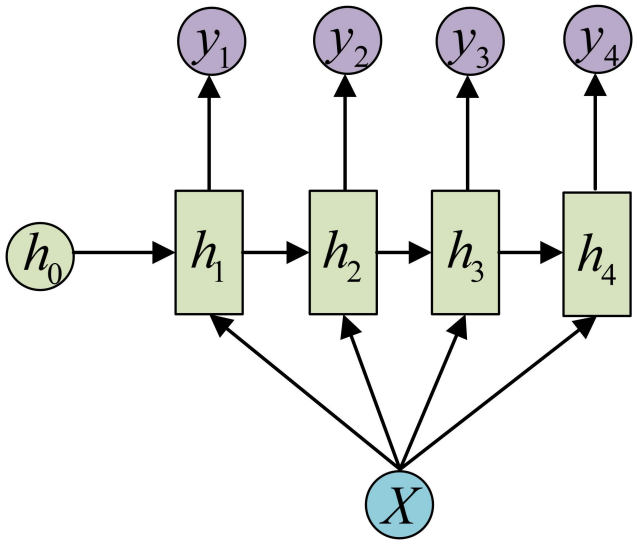


图 5.19: 1 to N（2）

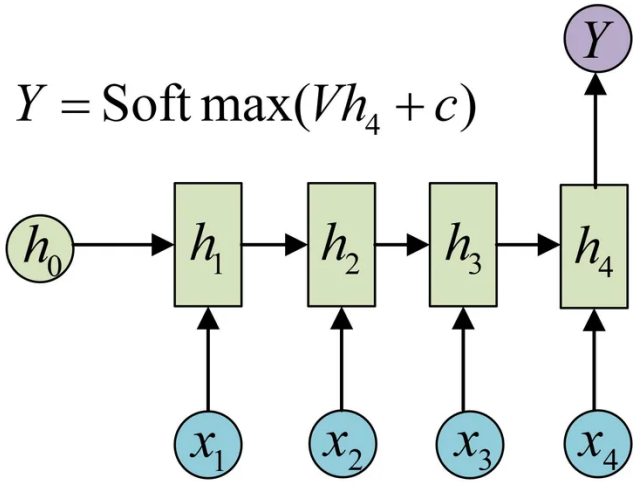


图 5.20: N to 1

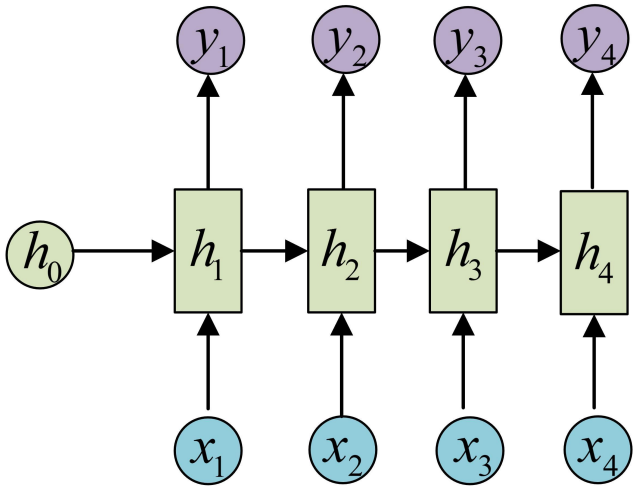


图 5.21: N to N

N to 1

输入是一个序列，输出是一个单独的值而不是序列。这种结构通常用来处理序列分类问题。如输入一段文字判别它所属的类别，输入一个句子判断其情感倾向，输入一段文档并判断它的类别等等。具体如图5.20：

N to N

如图5.21

输入和输出序列是等长的。这种可以作为简单的 Char RNN 可以用来生成文章，诗歌，甚至是代码，非常有意思。

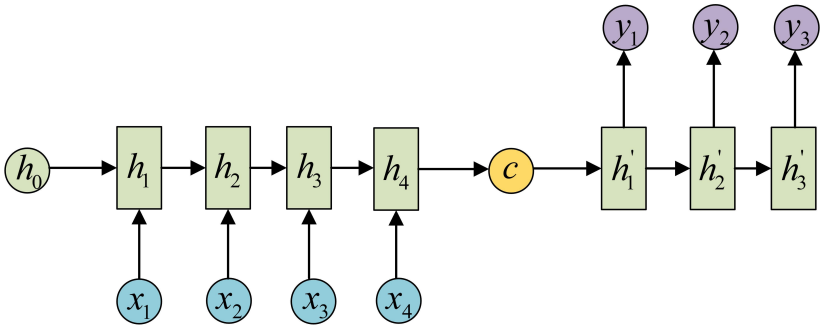


图 5.22: N to M

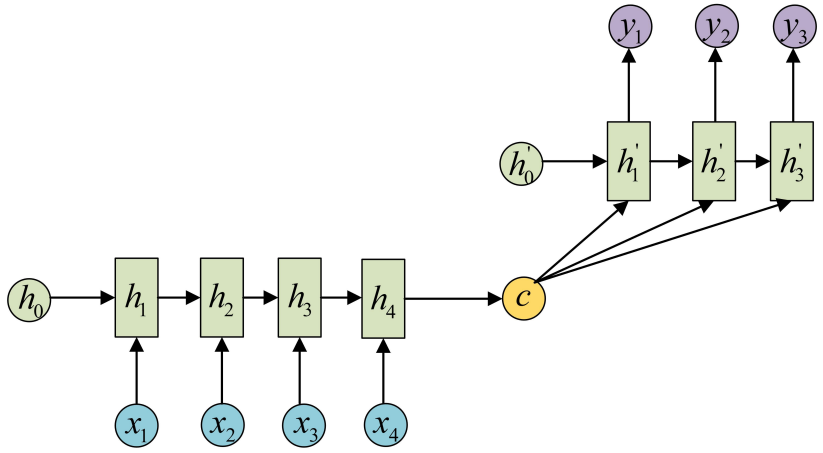


图 5.23: 带有 Attention 机制的 N to M

N to M

这种结构又叫 Encoder-Decoder 模型，也称之为 Seq2Seq 模型。在现实问题中，我们遇到的大部分序列都是不等长的。如机器翻译中，源语言和目标语言的句子往往并没有相同的长度。而 Encoder-Decoder 结构先将输入数据编码成一个上下文向量 c，之后在通过这个上下文向量输出预测序列。如图5.22, 图5.23。

5.4.3 双向循环神经网络

对于语言模型来说，很多时候单向循环神经网络表现是不好的，比如下面这句话：

我的手机坏了，我打算 ___ 一部新手机。

可以想象，如果我们只看横线前面的词，手机坏了，那么我是打算修一修？换一部新的？还是大哭一场？这些都是无法确定的。但如果我们也看到了横线后面的词是“一部新手机”，那么，横线上的词填『买』的概率就大得多了。

对于上面的语言模型，单向循环神经网络是无法对此进行建模的。因此，我们需要用双向循环神经网络，如图 3 所示。

从图 3 中可以看出，双向循环神经网络的隐藏层要保存两个值，一个 A 参与正向计算，另一个值 A' 参与反向计算。我们以 y_2 的计算为例，推出循环神经网络的一般规律。最终的输出值 y_2 取决于 A_2

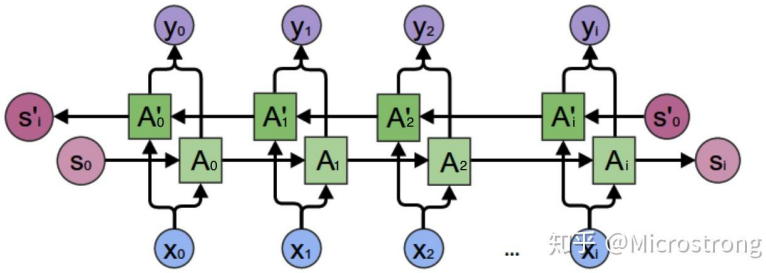


图 5.24: 双向循环神经网络

和 A_2' 。其计算方式为公式 4：

$$y_2 = g(VA_2 + V'A_2') \tag{5.14}$$

A_2 和 A_2' 分别为：

$$A_2 = f(WA_1 + Ux_2) \tag{5.15}$$

$$A_2' = f(W'A_3' + U'x_2) \tag{5.16}$$

现在，我们已经可以看出一般的规律：正向计算时，隐藏层的值 s_t 与 s_{t-1} 有关；反向计算时，隐藏层的值 s_t' 与 s_{t+1} 有关；最终的输出取决于正向和反向计算的加和。现在，我们仿照式 5 和式 6，写出双向循环神经网络的计算方法：

$$o_t = g(Vs_t + V's_t') \tag{5.17}$$

$$s_t = f(Ux_t + Ws_{t-1}) \tag{5.18}$$

$$s_t' = f(U'x_t + W's_{t+1}') \tag{5.19}$$

从上面三个公式我们可以看到，正向计算和反向计算不共享权重，也就是说 U 和 U'、W 和 W'、V 和 V' 都是不同的权重矩阵。

前向传播：

- 沿着时刻 0 到时刻 i 正向计算一遍，得到并保存每个时刻向前隐含层的输出。
- 沿着时刻 i 到时刻 0 反向计算一遍，得到并保存每个时刻向后隐含层的输出。
- 正向和反向都计算完所有输入时刻后，每个时刻根据向前向后隐含层得到最终输出。

反向传播：

- 计算所有时刻输出层的损失函数项。
- 根据所有输出层的损失函数项，使用 BPTT 算法更新向前层。
- 根据所有输出层的损失函数项，使用 BPTT 算法更新向后层。

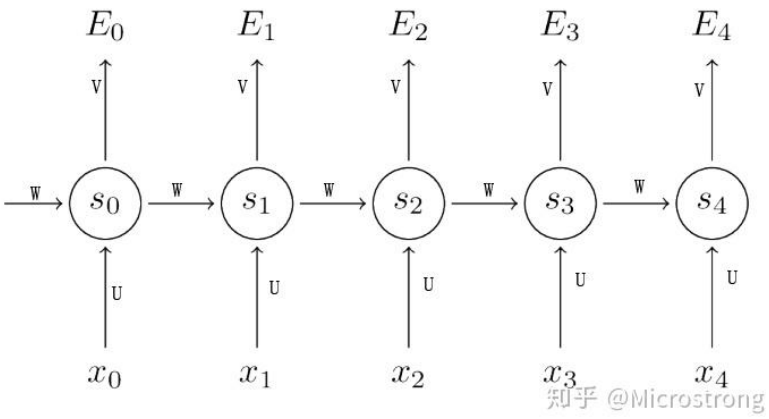


图 5.25: 循环神经网络的展开图

5.4.4 循环神经网络反向传播算法 (BPTT)

RNN 的反向传播算法为 Backpropagation Through Time (BPTT)。让我们先回忆一下 RNN 的基本公式，注意到这里在符号上稍稍做了改变（o 变成 $\hat{y}$），这只是为了和我参考的一些资料保持一致。

$$s_t = tanh(Ux_t + Ws_{t-1})$$

$$\hat{y} = softmax(Vs_t)$$

同样把损失值定义为交叉熵损失，如下：

$$E_t(y_t, \hat{y}_t) = -y_t log(\hat{y}_t)$$

$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t) = -\sum_t y_t log\hat{y}_t$$

这里，y_t 表示时刻 t 正确的词，$\hat{y}_t$ 是我们的预测。通常我们会把整个句子作为一个训练样本，所以总体误差是每一时刻的误差的累加和。

我们的目标是计算误差值相对于参数 U,V,W 的梯度以及用随机梯度下降学习好的参数。就像我们要把所有误差累加一样，我们同样会把每一时刻针对每个训练样本的梯度值相加：

$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$

为了计算梯度，我们使用链式求导法则，主要是用反向传播算法往后传播误差。下面我们使用 E_3 作为例子，主要是为了描述方便。

$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial z_3}\frac{\partial z_3}{\partial V} = (\hat{y}_3 - y_3) \otimes s_3 \tag{5.20}$$

上面 $z_3 = Vs_3$，$\otimes$ 是向量的外积。如果你不理解上面的公式，不要担心，我在这里对上面的推导过程进行补充。这里我想说明的一点是梯度值只依赖于当前时刻的结果 $\hat{y}_3, y_3, s_3$。根据这些，计算 V 的梯度就只剩下简单的矩阵乘积了。

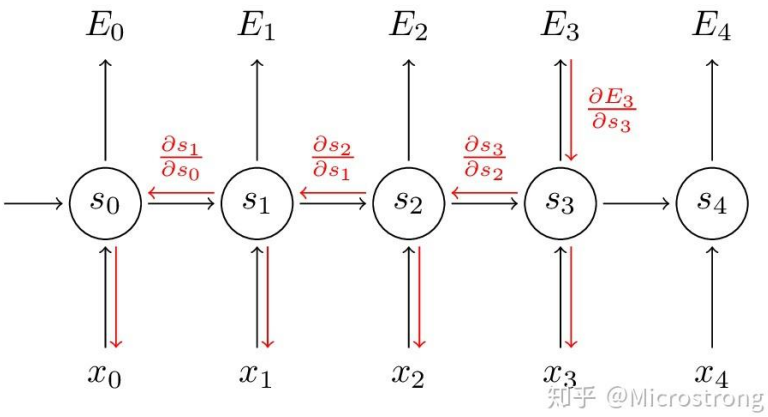


图 5.26: 循环神经网络的 BPTT 训练算法

但是对于梯度 $\frac{\partial E_3}{\partial W}$ 情况就不同了，我们可以像上面一样写出链式法则。

$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\frac{\partial s_3}{\partial W} \tag{5.21}$$

我们把每一时刻得到的梯度值累加，换句话说，W 在计算输出的每一步中都使用了。我们需要通过将 $t=3$ 时刻的梯度反向传播至 $t=0$ 时刻。

请注意，这与我们在深层前馈神经网络中使用的标准反向传播算法是完全相同的。关键的区别是，我们把每一时刻针对 W 的不同梯度做了累加。在传统的神经网络中，我们不需要跨层共享权重，所以不需要求和。但在我看来，BPTT 是应用于展开的 RNN 上的标准反向传播的另一个名字。就像反向传播一样，你也可以定义一个反向传递的 δ 向量，例如，$\delta_2^{(3)} = \frac{\partial E_3}{\partial z_2} = \frac{\partial E_3}{\partial s_3}\frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial z_2}$，其中 $z_2 = Ux_2 + Ws_1$。

通过以上的推导，我们可以得出结论：

RNN 很难训练的原因是：序列 (句子) 可能很长，可能是 20 个单词或更多，因此需要在许多层中进行反向传播。在实践中，许多人限定反向传播为有限的步骤（也是因为有梯度消失的问题存在）。

5.4.5 深度循环神经网络

上面介绍的基本循环神经网络和双向循环神经网络只有单个隐藏层，有时不能很好的学习数据内部关系，可以通过叠加多个隐藏层，形成深度循环神经网络结构。

Deep（Bidirectional）RNN 和 Bidirectional RNN 相似，只是在每个时刻会有多个隐藏层。在实际中会有更强的学习能力，但也需要更多的训练数据。

5.4.6 循环神经网络的时间步长和参数共享

time steps 就是循环神经网络认为每个输入数据与前多少个陆续输入的数据有联系。例如具有这样一段序列数据“…ABCDBCEDF…”，当 time steps 为 3 时，在模型预测中如果输入数据为“D”，那么之前接收的数据如果为“B”和“C”则此时的预测输出为 B 的概率更大，之前接收的数据如果为“C”和“E”，则此时的预测输出为 F 的概率更大。

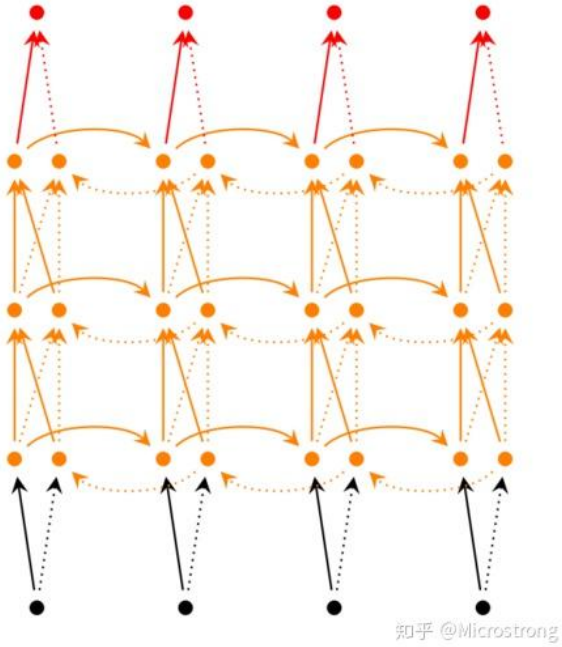


图 5.27: 深度循环神经网络

循环神经网络参数共享指的是：在每一个时间步上，所对应的参数是共享的。参数共享的目的有两个：一、用这些参数来捕获序列上的特征；二、共享参数减少模型的复杂度。

如图5.27所示，在 RNN 中每输入一步，每一层各自都共享参数 U、V、W。其反映着 RNN 中每一时刻都在做相同的事情，只是输入不同，因此大大减少了网络中需要学习的参数。

对于 RNN 的参数共享，我们可以理解为对于一个句子或者文本，参数 U 可以看成是语法结构、参数 W 是一般规律，而下一个单词的预测必须是上一个单词和一般规律 W 与语法结构 U 共同作用的结果。我们知道，语法结构和一般规律在语言当中是共享的。所以，参数自然就是共享的！

CNN 和 RNN 参数共享的区别：

我们需要记住的是，深度学习是怎么减少参数的，很大原因就是参数共享，而 CNN 是在空间上共享参数，RNN 是在时间序列上共享参数。

5.4.7 RNN 的梯度消失和梯度爆炸

上面我们讲到的单向循环神经网络、双向循环神经网络和深度循环神经网络都是传统的循环神经网络。这些传统的循环神经网络在学习过程中容易出现梯度消失和梯度爆炸的现象。所谓的梯度消失是指：信息在反向传播时误差减小为 0；而梯度爆炸是指：信息在反向传播时误差呈指数增长。下面，我们来深入的了解一下循环神经网络中产生梯度消失和梯度爆炸的本质原因。

RNN 很难学到长范围的依赖-即相隔几步的词之间的联系。传统的 RNN 不能捕获长距离词之间的关系。要理解为什么，让我们先仔细看一下下面计算的梯度：

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_3}{\partial W} \tag{5.22}$$

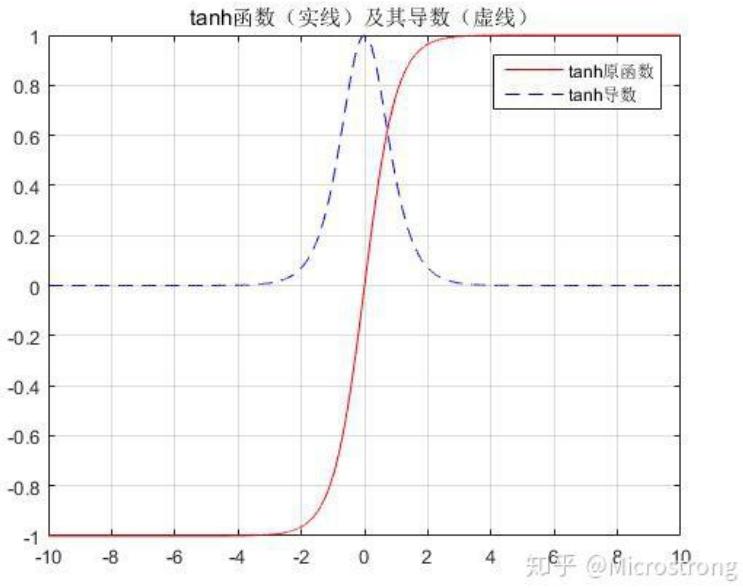


图 5.28: tanh 函数及其导数

注意到 $\frac{\partial s_3}{\partial s_k}$ 也需要使用链式法则。例如，$\frac{\partial s_3}{\partial s_1} = \frac{\partial s_3}{\partial s_2}\frac{\partial s_2}{\partial s_1}$。注意到因为我们是用向量函数对向量求导数，结果是一个矩阵（称为 Jacobian Matrix），矩阵元素是每个点的导数。我们可以把上面的梯度重写成：

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3}\frac{\partial \hat{y}_3}{\partial s_3}\left(\prod_{j=k+1}^{3} \frac{\partial s_j}{\partial s_{j-1}}\right)\frac{\partial s_k}{\partial W} \tag{5.23}$$

可以证明上面的 Jacobian 矩阵的二范数（可以认为是一个绝对值）的上界是 1。这很直观，因为激活函数 *tanh* 把所有值映射到-1 和 1 之间，导数值得界限也是 1。

从图5.28中我们可以看到，tanh 函数在 x 趋近于无穷大和无穷小时梯度值都为 0。当这种情况出现时，我们就认为相应的神经元饱和。它们的梯度为 0 使得前面层的梯度也为 0，因为前面层的梯度是由后面层的梯度连乘得到的。梯度中存在比较小的值，多个梯度相乘会使梯度值以指数级速度下降，最终在反向传播几步后就完全消失。比较远的时刻的梯度值为 0，这些时刻的状态对学习过程没有帮助，导致你无法学习到长距离依赖。梯度消息问题不仅出现在 RNN 中，同样也出现在深度前向神经网中。只是 RNN 网络通常比较深，使得这个问题更加普遍。

RNN 的学习依赖于我们的激活函数和网络初始参数，如果 Jacobian 矩阵中的值太大，会产生梯度爆炸而不是梯度消失问题。梯度消失比梯度爆炸受到了更多的关注有两方面的原因。其一，梯度爆炸容易发现，梯度值会变成 NaN，导致程序崩溃。其二，用预定义的阈值裁剪梯度可以简单有效的解决梯度爆炸问题。梯度消失问题就不容易发现并且也不好处理，这也是梯度消失比梯度爆炸受到学术界更多关注的原因。

上面我们讲到的循环神经网络梯度消失问题在 RNN 中会造成不能 Long-Term 依赖问题。

例如，考虑一个语言模型试图基于先前的词预测下一个词。如果我们要预测“the clouds are in the sky”，我们不需要其它更遥远的上下文信息就能预测出下一个词应该是 sky。在这样的例子中，句子长度很小，相关信息和预测单词的距离很小，RNN 可以学着去使用过去的信息。

但是，在大部分情况下我们需要更多的上下文信息。考虑预测这个句子中最后一个词：“I grew up in France…I speak fluent French.”最近的信息表明下一个词可能是一种语言的名字，但如果我们想要找出是哪种语言，我们需要从更久远的地方获取 France 的上下文。相关信息和预测单词之间的距离完全可能是非常巨大的。而不幸的是，随着距离的增大，RNN 会产生梯度消失的问题，造成不能够连接更长的上下文信息

5.4.8 RNN 的 Long-Term 依赖问题

上面我们讲到的循环神经网络梯度消失问题在 RNN 中会造成不能 Long-Term 依赖问题。

例如，考虑一个语言模型试图基于先前的词预测下一个词。如果我们要预测“the clouds are in the sky”，我们不需要其它更遥远的上下文信息就能预测出下一个词应该是 sky。在这样的例子中，句子长度很小，相关信息和预测单词的距离很小，RNN 可以学着去使用过去的信息。

但是，在大部分情况下我们需要更多的上下文信息。考虑预测这个句子中最后一个词：“I grew up in France…I speak fluent French.”最近的信息表明下一个词可能是一种语言的名字，但如果我们想要找出是哪种语言，我们需要从更久远的地方获取 France 的上下文。相关信息和预测单词之间的距离完全可能是非常巨大的。而不幸的是，随着距离的增大，RNN 会产生梯度消失的问题，造成不能够连接更长的上下文信息。

5.4.9 长短期记忆模型 LSTM

循环神经网络除了训练困难，还有一个更严重的问题，那就是短时记忆 (Short-term memory)。考虑一个长句子：今天天气太美好了，尽管路上发生了一件不愉快的事情……我马上调整好状态，开开心心地准备迎接美好的一天。根据理解，之所以能够“开开心心地准备迎接美好的一天”，在于句子最开始处点名了“今天天气太美好了”。可见人类是能够很好地理解长句子的，但是循环神经网络却不一定。研究人员发现，循环神经网络在处理较长的句子时，往往只能够理解有限长度内的信息，而对于较长范围类的有用信息往往不能够很好地利用起来。把这种现象称为短时记忆。

那么，能否延长这种短时记忆，使得循环神经网络可以有效利用较大范围内的训练数据，从而提升性能？1997 年，瑞士人工智能科学家 Jürgen Schmidhuber 提出了长短时记忆网络 (LongShort-TermMemory，LSTM)。LSTM 相对于基础的 RNN 网络来说，记忆能力更强，更擅长处理较长的序列信号数据，LSTM 提出后，被广泛应用在序列预测、自然语言处理等任务中，几乎取代了基础的 RNN 模型。接下来将介绍更加流行、更强大的 LSTM 网络。

基础的 RNN 网络结构如图5.29所示，上一个时间戳的状态向量 h_{t-1} 与当前时间戳的输入 x_t 经过线性变换后，通过激活函数 tanh 后得到新的状态向量 h_t。相对于基础的 RNN 网络只有一个状态向量 h_t，LSTM 新增了一个状态向量 C_t，同时引入了门控 (Gate) 机制，通过门控单元来控制信息的遗忘和刷新，如图5.29所示。

在 LSTM 中，有两个状态向量 c 和 h，其中 c 作为 LSTM 的内部状态向量，可以理解为 LSTM 的内存状态向量 Memory，而 h 表示 LSTM 的输出向量。相对于基础的 RNN 来说，LSTM 把内部 Memory 和输出分开为两个变量，同时利用 3 个门控：输入门（Input Gate）、遗忘门（ForgetGate）和输出门（Output Gate）来控制内部信息的流动。

门控机制可以理解为控制数据流通量的一种手段，类比于水阀门：当水阀门全部打开时，水流畅通无阻地通过；当水阀门全部关闭时，水流完全被隔断。在 LSTM 中，阀门开和程度利用门控值向量 g

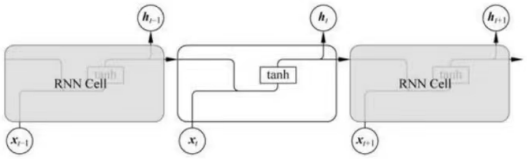


图11.13 基础RNN结构

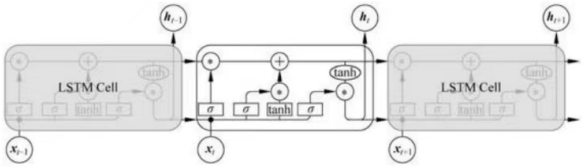


图11.14 LSTM结构

图 5.29: LSTM 结构

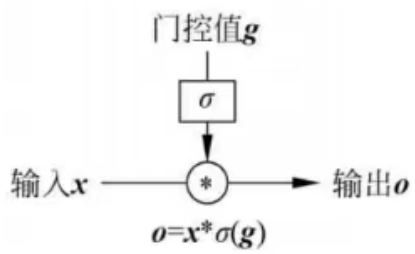


图11.15 门控机制

图 5.30: 门控机制

表示，如图5.30所示，通过 $\sigma(g)$ 激活函数将门控制压缩到 [0，1] 区间，当 $\sigma(g) = 0$ 时，门控全部关闭，输出 o=0；当 $\sigma(g) = 1$ 时，门控全部打开，输出 o=x。通过门控机制可以较好地控制数据的流量程度。下面分别来介绍三个门控的原理及其作用。

遗忘门作用于 LSTM 状态向量 c 上面，用于控制上一个时间戳的记忆 c_{t-1} 对当前时间戳的影响。遗忘门的控制变量 g_f 由

$$g_f = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{5.24}$$

产生，如图5.31所示，其中 W_f 和 b_f 为遗忘门的参数张量，可由反向传播算法自动优化，σ 为激活函数，一般使用 Sigmoid 函数。当门控 $g_f = 1$ 时，遗忘门全部打开，LSTM 接受上一个状态 c_{t-1} 的所有信息；当门控 $g_f = 0$ 时，遗忘门关闭，LSTM 直接忽略 c_{t-1}，输出为 0 的向量。这也是遗忘门的名字由来。

经过遗忘门后，LSTM 的状态向量变为 $g_f c_{t-1}$。

输入门用于控制 LSTM 对输入的接收程度。首先通过对当前时间戳的输入 x_t 和上一个时间戳的输

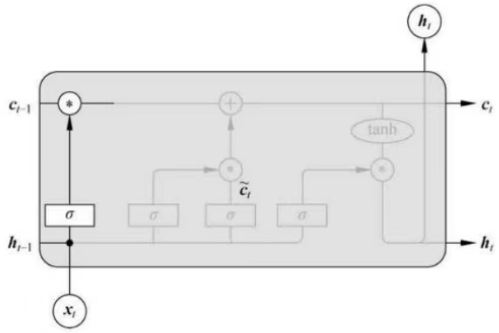


图11.16 遗忘门

图 5.31: 遗忘门

出 h_{t-1} 做非线性变换得到新的输入向量 $\widetilde{c}_t$：

$$\widetilde{c}_t = tanh(W_c[h_{t-1}, x_t] + b_c) \tag{5.25}$$

其中 W_c 和 b_c 为输入门的参数，需要通过反向传播算法自动优化，tanh 为激活函数，用于将输入标准化到 [-1，1] 区间。并不会全部刷新进入 LSTM 的 Memory，而是通过输入门控制接受输入的量。输入门的控制变量同样来自于输入 x_t 和输出 h_{t-1}：

$$g_i = \sigma(W_i[h_{t-1}\ x_t] + b_i) \tag{5.26}$$

其中 W_i 和 b_i 为输入门的参数，需要通过反向传播算法自动优化，σ 为激活函数，一般使用 Sigmoid 函数。输入门控制变量 g_i 决定了 LSTM 对当前时间戳的新输入的接受程度：当 $g_i = 0$ 时，LSTM 不接受任何的新输入；当 $g_i = 1$ 时，LSTM 全部接受新输入，如图5.32所示。

经过输入门后，待写入 Memory 的向量为 $g_i\widetilde{c}_t$。

在遗忘门和输入门的控制下，LSTM 有选择地读取了上一个时间戳的记忆 c_{t-1} 和当前时间戳的新输入 $\widetilde{c}_t$，状态向量 c_t 的刷新方式为：

$$c_t = g_i\widetilde{c}_t + g_f c_{t-1} \tag{5.27}$$

得到新状态向量 c_t 即为当前时间戳的状态向量，如图5.32

LSTM 的内部状态向量 c_t 并不会直接用于输出，这一点和基础的 RNN 不一样。基础的 RNN 网络的状态向量 h 既用于记忆，又用于输出，所以基础的 RNN 可以理解为状态向量 c 和输出向量 h 是同一个对象。在 LSTM 内部，状态向量并不会全部输出，而是在输出门的作用下有选择地输出。输出门的门控变量 g_o 为：

$$g_o = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{5.28}$$

其中 W_o 和 b_o 为输出门的参数，同样需要通过反向传播算法自动优化，σ 为激活函数，一般使用 Sigma 函数，当输出门 $g_o = 0$ 时，输出关闭，LSTM 内部记忆完全被隔断，无法用作输出，此时输出为 01 的向量；当输出门 $g_o = 1$ 时，输出完全打开，LSTM 的状态向量全部用于输出。LSTM 的输出由：

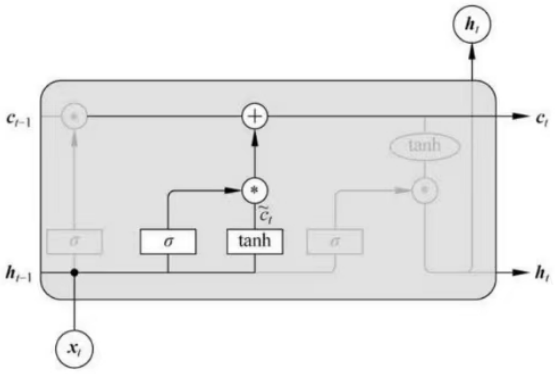


图11.17 输入门

图 5.32: 输入门

$$h_t = g_o \cdot tanh(c_t) \tag{5.29}$$

产生,即内存向量 c_t 经过 $tanh$ 激活函数后与输出门作用, 得到 LSTM 的输出。由于 $g_o \in [0,1], tanh(c_t) \in [-1,1]$，因此 LSTM 的输出 $h_t \in [-1,1]$, 如图5.33。

相比于传统 RNN 模型，长短期记忆模型 LSTM 在处理长序列数据上具有更优越的表现。原因在于，RNN 模型中仅通过构造潜在因子来进行预测，对于因子的记忆能力保持不变，这可能导致，潜在因子对于之前更长远期的数据记忆能力不佳。而长短期记忆模型 LSTM 考虑到这种记忆因素，通过构造记忆因子（记录当期数据）与遗忘因子（记录前期数据）来保证序列的长期记忆能力，使得长短期记忆模型 LSTM 方法处理长序列数据有着不错的表现。

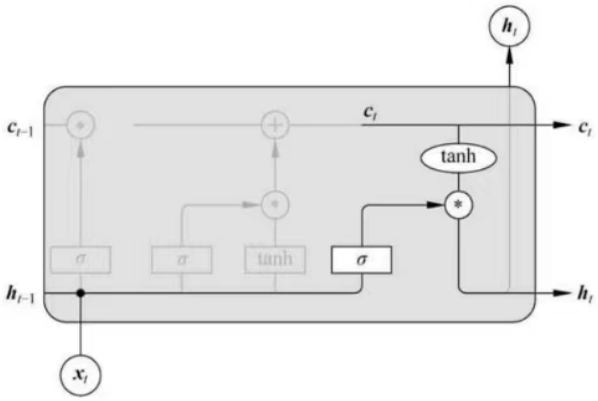


图11.18 输出门

图 5.33: 输入门

第六章　集成算法：Bagging、Boosting、stacking 及相关算法

May 2024

6.1 Bagging 算法简介

Bagging 算法的全称是 Bootstrap Aggregating，由 Leo Breiman 于 1996 年提出。它的基本思想是通过对原始数据集进行多次随机有放回抽样，生成多个不同的子数据集，并在这些子数据集上训练多个基模型，最后将这些基模型的预测结果进行组合，从而得到最终的预测结果。Bagging（Bootstrap Aggregating）是一种集成学习技术，通过结合多个基模型的预测结果来提高机器学习模型的稳定性和准确性。它主要用于减少模型的方差，增强整体性能。同时，由于多个基模型是基于不同的子样本集训练的，Bagging 能有效防止单个模型过拟合于训练数据. 此外，在 Bagging 算法中，各个基模型的训练可以并行进行，计算效率较高。Bagging 算法的主要步骤如下：

- 步骤 1：数据抽样。从原始数据集中使用有放回的抽样方法随机抽取多个子样本集。每个子样本集的大小与原始数据集相同，但是由于是有放回抽样，某些样本可能在一个子样本集中出现多次，而某些样本可能未出现。
- 步骤 2：训练基模型。在每个子样本集上训练一个基模型（例如决策树、线性回归等）。由于每个子样本集都是不同的，因此训练得到的基模型也会有所不同。
- 步骤 3：将所有训练好的基模型的预测结果进行组合。对于回归任务，可以通过平均法（averaging）来确定最终预测值。

6.1.1 应用案例：How Useful Is Bagging in Forecasting Economic Time Series? （Inoue 和 Kilian，2008）

本节基于 Inoue 和 Kilian（2008）介绍以下内容：1）怎么使用 bagging 方法处理可能具有序列相关和异方差误差的动态线性多元回归问题；2）bagging 方法在因子模型中的应用。

模型基本说明

预测模型如下：

$$y_{t+h} = \alpha' w_t + \beta' x_t + \varepsilon_{t+h}, \quad h = 1, 2, 3, \ldots \tag{6.1}$$

ε_{t+h} 是 h 期的预测误差, β 是 M 维的参数向量，x_t 是 t 期 M 个可能相关的预测变量的列向量. y_t 和 x_t 是联合协方差平稳过程。w_t 是前定向量，包含截距项和响应变量的滞后项等，α 是其系数. 在本节，均基于 t 统计量对 x_t 进行变量选择。以下三个例子将说明，如何基于 bagging 方法对不同模型进行聚合。为便于说明，后文的描述忽略 $\alpha' w_t$。

Bagging Unorthogonalized Predictors，BA

该方法不考虑预测变量间的相关性，建立 y_{T+h} 和 x_T 间的回归模型，基于 bootstrap 样本进行重复估计，实现预测。其中，T 是最近的观测期，$x_t \in \mathfrak{R}^M$. $\widehat{\beta}$ 是公式6.1β 的 OLS 估计结果。t_j 是 β_j 为 0 的 t 统计量，β_j 是 β 的第 j 个元素。$\widehat{\gamma}$ 是进行变量选择后的 OLS 估计量。无限制模型 (UR), 限制模型 (FR), 和提前测试选择变量后的模型（PT）基于 x_T 的条件函数为：

$$\begin{aligned}
&\hat{y}_{T+h}^{\mathrm{UR}}(x_T) = \widehat{\beta}' x_T, \\
&\hat{y}_{T+h}^{\mathrm{FR}}(x_T) = 0, \\
&\hat{y}_{T+h}^{\mathrm{PT}}(x_T) = 0, \quad \text{if } |t_j| < c \forall j \quad \text{and} \\
&\hat{y}_{T+h}^{\mathrm{PT}}(x_T) = \widehat{\gamma}' S_T x_T \quad \text{otherwise,}
\end{aligned}$$

S_T 是随机选择矩阵，是 (i,i) 位置处元素为 $I(|t_i| > c)$ 的对角矩阵，c 是提前预测用到的临界值。特别的，进行 PT 选择时先进行 UR 估计，得到每个变量的 t 统计量，然后基于此对选择预测变量。进行 t 检验时，要选取允许序列相关和异方差的标准差。

基于 bootstrap 抽样样本，估计 PT 方程后进行聚合。带 * 的标记表示基于 bootstrap 抽样得到的数据或估计。

BA 方法：

a. 将数据 $\{(y_{t+h}, x_t')\}, t = 1, \ldots, T-h$ 排列为下列维度为 $(T-h) \times (M+1)$ 的矩阵：

$$\begin{matrix} y_{1+h} & x_1' \\ \vdots & \vdots \\ y_T & x_{T-h}' \end{matrix}$$

对其进行块抽样，每次替换掉其中的 m 行，得到 bootstrap 抽样样本。$\left(y_{1+h}^*, x_1^{**}\right), \ldots, \left(y_T^*, x_{T-h}'^*\right)$。

b. 对于每个抽样样本，计算基于 x_T 的 PT 条件预测值：

$$\begin{aligned}
&\hat{y}_{T+h}^{*\mathrm{PT}}(x_T) = 0, \quad \text{if } \left|t_j^*\right| < c \forall j \quad \text{and} \\
&\hat{y}_{T+h}^{*\mathrm{PT}}(x_T) = \widehat{\gamma}^{*\prime} S_T^* x_T \quad \text{otherwise,}
\end{aligned}$$

在构造 $\left|t_j^*\right|$ 时, 基于下式计算 $\sqrt{T}\widehat{\beta}^*$ 的方差 $\widehat{H}^{*-1}\widehat{V}^*\widehat{H}^{*-1}$，其中

$$\begin{aligned}
\widehat{V}^* =& \frac{1}{bm}\sum_{k=1}^{b}\sum_{i=1}^{m}\sum_{j=1}^{m}\left(x_{(k-1)m+i}^* \varepsilon_{(k-1)m+i+h}^*\right) \\
&\times \left(x_{(k-1)m+j}^* \varepsilon_{(k-1)m+j+h}^*\right)', \\
\widehat{H}^* =& \frac{1}{bm}\sum_{k=1}^{b}\sum_{i=1}^{m}\left(x_{(k-1)m+i}^* x_{(k-1)m+i}^{*\prime}\right),
\end{aligned}$$

$\varepsilon^*_{t+h} = y^*_{t+h} - \widehat{\beta}^{*\prime} x^*_t$, b 是 $(T-h)/m$ 的整数值 (Inoue and Shintani 2006)。

c. baaging 估计量即 bootstrap 样本数据的 PT 估计值的均值：

$$\hat{y}^{\mathrm{BA}}_{T+h}(x_T) = \frac{1}{B}\sum_{i=1}^{B} \widehat{\gamma}^{*i\prime} S^{*i}_T x_T$$

理论上 $B=\infty$，实际应用中 B 一般在 100 以上，可以使用 data-dependent rules，如校准等选择 m（Politis, Romano 和 Wolf 1999).c 可以基于网格搜索进行选择.

Bagging Orthogonalized Predictors，CBA

当变量间存在相关性，直接使用 t 检验检验变量的预测有效性不合理，因为每个变量的有效性会相互影响。为此，可以对解释变量作正交化处理：$\widetilde{x}_t = P'^{-1}x_t$, 其中 P 是 $E(x_t x_t')$ 进行乔里斯基分解得到的 $M\times M$ 上三角矩阵，$P'P = E(x_t x_t')$。并基于正交化处理的数据进行拟合。此时预测函数如下：

$$y_{t+h} = \alpha' w_t + \beta' \tilde{x}_t + \varepsilon_{t+h}, \quad h = 1,2,3,\ldots$$

对变量的预测试过程如下：

$$\begin{aligned}
\hat{y}^{\mathrm{CPT}}_{T+h}(\tilde{x}_T) &= 0, \quad \text{if } |t_j| < c\forall j \quad \text{and}\\
\hat{y}^{\mathrm{CPT}}_{T+h}(\tilde{x}_T) &= \widehat{\gamma}' S_T \tilde{x}_T \quad \text{otherwise,}
\end{aligned}$$

CBA 算法如下：a. 将变量 $\{(y_{t+h}, x_t')\}, t = 1,\ldots,T-h$, 排列为如下形式：

$$\begin{matrix} y_{1+h} & x_1' \\ \vdots & \vdots \\ y_T & x'_{T-h} \end{matrix}$$

使用 bootstrap 块抽样抽取样本：$\left(y^*_{1+h}, x_1'^*\right), \ldots, \left(y^*_T, x'^*_{T-h}\right)$ 。

b. 将变量作正交化处理 $\widetilde{x}_t = P'^{-1}x_t$, 其中 P 是 $E(x_t x_t')$ 的乔里斯基分解矩阵。对每一个抽样样本，进行如下估计：

$$\begin{aligned}
\hat{y}^{*\mathrm{CPT}}_{T+h}(\widetilde{x}_T) &= 0, \quad \text{if } \left|t^*_j\right| < c\forall j \quad \text{and}\\
\hat{y}^{*\mathrm{CPT}}_{T+h}(\widetilde{x}_T) &= \widehat{\gamma}^{*\prime} S^*_T \widetilde{x}_T \quad \text{otherwise,}
\end{aligned}$$

构造 $\left|t^*_j\right|$ 时, 基于下式计算 $\sqrt{T}\widehat{\beta}^*$ 的方差 $\widehat{H}^{*-1}\widehat{V}^*\widehat{H}^{*-1}$：

$$\begin{aligned}
\widehat{V}^* =& \frac{1}{bm}\sum_{k=1}^{b}\sum_{i=1}^{m}\sum_{j=1}^{m}\left(\widetilde{x}^*_{(k-1)m+i}\widetilde{\varepsilon}^*_{(k-1)m+i+h}\right)\\
&\times\left(\widetilde{x}^*_{(k-1)m+j}\widetilde{\varepsilon}^*_{(k-1)m+j+h}\right)',\\
\widehat{H}^* =& \frac{1}{bm}\sum_{k=1}^{b}\sum_{i=1}^{m}\left(\widetilde{x}^*_{(k-1)m+i}\widetilde{x}^{*\prime}_{(k-1)m+i}\right),
\end{aligned}$$

$\widetilde{\varepsilon}^*_{t+h} = y^*_{t+h} - \widehat{\beta}^{*\prime}\widetilde{x}^*_t$, b 是 $(T-h)/m$ 的整数部分。

c. 最终预测值如下：

$$\hat{y}^{\mathrm{CBA}}_{T+h}(\tilde{x}_T) = \frac{1}{B}\sum_{i=1}^{B}\widehat{\gamma}^{*i\prime} S^{*i}_T \tilde{x}_T$$

Bagging Factor Predictors

上述正交化方法只适用于解释变量矩阵满秩的情形。对于解释变量矩阵非满秩的情形，可以基于因子模型进行分析。

$$y_{t+h} = \alpha' w_t + \beta' f_t + \varepsilon_{t+h}, \quad h = 1, 2, 3, \ldots \tag{3}$$

f_t 是基于主成分分析从 N 个解释变量中提取的前 r 个解释性最好的因子（Stock 和 Watson 2002a,b)

此时预检测方程 PT 如下：

$$\begin{aligned} \hat{y}_{T+h}^{\mathrm{PT}^F}\left(\widehat{f_T}\right) &= 0, \quad \text{if } |t_j| < c\forall j \quad \text{and} \\ \hat{y}_{T+h}^{\mathrm{PT}^F}\left(\widehat{f_T}\right) &= \widehat{\gamma}' S_T \widehat{f_T} \quad \text{otherwise.} \end{aligned}$$

BA ^{F} 方法如下：

a. 对 $T \times N$ 的解释变量矩阵 X 进行主成分分析，提取前 r 个公共因子，得到第 t 期的 $r \times 1$ 维的 $\widehat{f}_t$ 向量.

b. 将 $\left\{\left(y_{t+h}, \widehat{f}_t'\right)\right\}, t = 1, \ldots, T-h$ 进行排列

$$\begin{matrix} y_{1+h} & \widehat{f}_1' \\ \vdots & \vdots \\ y_T & \widehat{f}_{T-h}' \end{matrix}$$

使用 bootstrap 块抽样抽取样本 $\left(y_{1+h}^*, \widehat{f_1^*}\right), \ldots, \left\{\left(y_T^*, \widehat{f_{T-h}^*}\right)\right.$ 即每次替换 m 行，块抽样可以捕捉误差项的自相关特征。**对于提取的因子，使用主成分分析做正交化处理。**

c. 对于每次得到的 bootstrap 样本，做 PT 估计：

$$\begin{aligned} \hat{y}_{T+h}^{*\mathrm{PT}^F}\left(\widehat{f}_T\right) &= 0, \quad \text{if } \left|t_j^*\right| < c\forall j \quad \text{and} \\ \hat{y}_{T+h}^{*\mathrm{PT}^F}\left(\widehat{f}_T\right) &= \widehat{\gamma}^{*\prime} S_T^* \widehat{f_T} \quad \text{otherwise,} \end{aligned}$$

构造 $\left|t_j^*\right|$ 统计量时, 基于下式估计 $\sqrt{T}\widehat{\beta}^*$ 的方差 $\widehat{H}^{*-1}\widehat{V}^*\widehat{H}^{*-1}$, 其中

$$\begin{aligned} \widehat{V}^* =& \frac{1}{bm}\sum_{k=1}^{b}\sum_{i=1}^{m}\sum_{j=1}^{m}\left(\widehat{f}_{(k-1)m+i}^*\widehat{\varepsilon}_{(k-1)m+i+h}^*\right) \\ & \times\left(\widehat{f}_{(k-1)m+j}^*\widehat{\varepsilon}_{(k-1)m+j+h}^*\right)', \\ \widehat{H}^* =& \frac{1}{bm}\sum_{k=1}^{b}\sum_{i=1}^{m}\left(\widehat{f}_{(k-1)m+i}^*\widehat{f}_{(k-1)m+i}^{*\prime}\right), \end{aligned}$$

$\widehat{\varepsilon}_{t+h}^* = y_{t+h}^* - \widehat{\beta}^{*\prime}\widehat{f}_t^*$,$b$ 是 $(T-h)/m$ 的整数部分.

d. 求不同样本得到的点预测的均值

$$\hat{y}_{T+h}^{\mathrm{BA}^F}\left(\widehat{f}_T\right) = \frac{1}{B}\sum_{i=1}^{B}\widehat{\gamma}^{*i\prime}S_T^{*i}\widehat{f}_T$$

主要对比结果：作者发现，发现使用 bagging 可以提高上述模型的预测效果。对于因子模型，进行一个月的预测时，bagging 的改善效果更为显著，且预测改善程度与使用的因子个数有关。

6.2 boosting 算法简介

boosting 是一个将“弱学习算法”，提升为“强学习算法”的过程。在该算法中：（1）每个弱学习器依次添加到模型中，并对先前模型的残差进行拟合。（2）通过适应性地调整观测值的权重或直接最小化损失函数来改进模型。Boosting 通过顺序训练多个基模型，并调整权重，使得模型能够更好地拟合数据，可以有效减少偏差和方差，进而提高预测的准确性。

提升算法的两个关键问题: (1) 在每一轮如何改变训练数据的权值或概率分布，或如何拟合一个弱学习器 (2) 如何将弱分类器组合成一个强分类器.

6.2.1 AdaBoost

AdaBoost 是一个经典的 boosting 算法，适用于分类问题。

假设给定一个二分类训练数据集：训练数据集 $T = \{(x_1, y_i), (x_2, y_2), \cdots, (x_N, y_N)\}$, 其中 $x_i \in \mathcal{X} \subseteq \mathbf{R}^n, y_i \in \mathcal{Y} = \{-1, +1\}$；

弱学习算法：$G_m(x)$，一个弱二分类分类器，由 $x < v$ 或 $x > v$ 产生

输出: 最终分类器 $G(x)$

Algorithm 3 AdaBoost.M1 算法

初始化观测权重 $w_i = \frac{1}{N}, i = 1, 2, \ldots, N$.

for $m = 1$ **to** M **do**

(a) 使用权重 w_i 将训练数据进行加权，并使用加权后数据拟合分类器 $G_m(x)$

(b) 计算

$$err_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}$$

(c) 计算基本分类器系数 $\alpha_m = \log((1 - err_m)/err_m)$

(d) 更新数据权值 $w_i \leftarrow w_i \cdot \exp(\alpha_m \cdot I(y_i \neq G_m(x_i))), i = 1, 2, \ldots, N$

end for

输出 $G(x) = \text{sign}(\sum_{m=1}^{M} \alpha_m G_m(x))$

关于第 1 个问题,AdaBoost 提高那些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. 因此，没有得到正确分类的数据, 由于其权值的加大而受到后一轮的弱分类器的更大关注. 于是, 分类问题被一系列的弱分类器“分而治之”.

至于第 2 个问题, 即弱分类器的组合, AdaBoost 采取加权多数表决的方法，将弱分类器进行线性组合。具体地，加大分类误差率小的弱分类器的权值，使其在表决中起较大的作用, 减小分类误差率大的弱分类器的权值, 使其在表决中起较小的作用.

6.2.2 加性模型的前向分步算法

加性模型的前向分步算法是 Boosting 算法的理论基础。可以考虑一个加性模型 (additive model)（如单隐藏层神经网络的计算等）

$$f(x)=\sum_{m=1}^{\mu}\beta_m b\left(x,\gamma_n\right) \tag{6.2}$$

其中, $b\left(x;\gamma_m\right)$ 为基函数, γ_m 为基函数的参数, β_m 为基函数的加权系数.

学习加性模型 $f(x)$ 的过程即损失函数极小化问题:

$$\min_{\beta_m,m_j}\sum_{j=1}^{N}L\left(y_i,\sum_{m=1}^{N}\beta_m b\left(x_i;\gamma_n\right)\right) \tag{6.3}$$

通常这是一个复杂的优化问题. 前向分步算法 (forward stagewise algorithm) 求解这一优化问题的想法是: 将同时求解从 $m=1$ 到 M 所有参数 β_m,γ_m 的优化问题简化为逐次求解各个 β_m,γ_m 的优化问题。即，从前向后, 每步只优化如下损失函数:

$$\min_{\beta,y}\sum_{i=1}^{N}L\left(y_i,\beta b\left(x_i;\gamma\right)\right) \tag{6.4}$$

给定训练数据集 $T=\left\{\left(x_1,y_1\right),\left(x_2,y_2\right),\cdots,\left(x_N,y_N\right)\right\},x_i\in\mathcal{X}\subseteq\mathbf{R}^n,y_i\in\mathcal{Y}=\{-1,+1\}$. 损失函数 $L(y,f(x))$ 和基函数的集合 $\{b(x,y)\}$ ，学习加性模型 $f(x)$ 的前向分步算法如算法4

Algorithm 4 Forward Stagewise Additive Modeling（前向分步算法）

输入: 训练数据集 $T=\left\{\left(x_1,y_1\right),\left(x_2,y_2\right),\cdots,\left(x_N,y_N\right)\right\}$: 损失函数 $L(y,f(x))$; 基函数集 $\{b(x,\gamma)\}$;

初始化 $f_0(x)=0$

for $m=1,2,\cdots,M$ **do**

（a）极小化损失函数 $\left(\beta_m,\gamma_m\right)=\arg\min_{\beta,y}\sum_{i=1}^{N}L\left(y_i,f_{m-1}\left(x_i\right)+\beta b\left(x_i;\gamma\right)\right)$，得到参数 β_m,γ_m

(b) 更新 $f_m(x)=f_{m-1}(x)+\beta_m b\left(x;\gamma_m\right)$

end for

得到加性模型

输出: 加性模型 $f(x)$. $f(x)=f_M(x)=\sum_{m=1}^{M}\beta_m b\left(x;\gamma_m\right)$

- 说明 1：对于平方损失函数，每一步新添加的项是当前残差的最优拟合值：

$$L(y,f(x))=(y-f(x))^2,$$

则

$$\begin{aligned}L\left(y_i,f_{m-1}\left(x_i\right)+\beta b\left(x_i;\gamma\right)\right)&=\left(y_i-f_{m-1}\left(x_i\right)-\beta b\left(x_i;\gamma\right)\right)^2\\&=\left(r_{im}-\beta b\left(x_i;\gamma\right)\right)^2\end{aligned}$$

- 说明 2：AdaBoost 算法是前向分步加性模型算法的特例，是加性模型由基本分类器组成，损失函数是指数函数时的特例。证明过程如下：假设一个加性模型的损失函数是指数函数：

$$L(y,f(x))=\exp(-yf(x)).$$

弱学习器为一个基础分类器件 $G_m(x)\in\{-1,1\}$，每一步更新 G_m 与 β_m 时求解的目标函数为：

$$\left(\beta_m,G_m\right)=\arg\min_{\beta,G}\sum_{i=1}^{N}\exp\left[-y_i\left(f_{m-1}\left(x_i\right)+\beta G\left(x_i\right)\right)\right]$$

可以表示为

$$(\beta_m, G_m) = \arg\min_{\beta, G} \sum_{i=1}^{N} w_i^{(m)} \exp\left(-\beta y_i G\left(x_i\right)\right)$$

其中 $w_i^{(m)} = \exp\left(-y_i f_{m-1}\left(x_i\right)\right)$. 该项与 $f_{m-1}\left(x_i\right)$ 有关，随迭代次数 m 变化而变化.

首先，固定 β，上述目标函数为

$$e^{-\beta} \cdot \sum_{y_i = G(x_i)} w_i^{(m)} + e^{\beta} \cdot \sum_{y_i \neq G(x_i)} w_i^{(m)},$$

可转换为

$$\left(e^{\beta} - e^{-\beta}\right) \cdot \sum_{i=1}^{N} w_i^{(m)} I\left(y_i \neq G\left(x_i\right)\right) + e^{-\beta} \cdot \sum_{i=1}^{N} w_i^{(m)}.$$

将上式再次代入目标函数，求解 β, 可以得到

$$\beta_m = \frac{1}{2} \log \frac{1 - \mathrm{err}_m}{\mathrm{err}_m},$$

其中

$$\mathrm{err}_m = \frac{\sum_{i=1}^{N} w_i^{(m)} I\left(y_i \neq G_m\left(x_i\right)\right)}{\sum_{i=1}^{N} w_i^{(m)}}.$$

进行更新

$$f_m(x) = f_{m-1}(x) + \beta_m G_m(x),$$

可以发现

$$w_i^{(m+1)} = w_i^{(m)} \cdot e^{\alpha_m I(y_i \neq G_m(x_i))} \cdot e^{-\beta_m},$$

其中 $\alpha_m = 2\beta_m$，便是 AdaBoost 算法 2(c) 的系数。

6.2.3 boosting tree

Algorithm 5 回归问题的 boosting tree 算法

输入: 训练数据集 $T = \{(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)\}, x_i \in \mathcal{X} \subseteq \mathbf{R}^n, y_i \in \mathcal{Y} \subseteq \mathbf{R}$;

初始化 $f_0(x) = 0$

for $m = 1, 2, \cdots, M$ **do**

(a) 计算残差 $r_{mi} = y_i - f_{m-1}\left(x_i\right), \quad i = 1, 2, \cdots, N$

(b) 学习一个回归树，拟合残差 r_{mi} , 得到 $T\left(x; \Theta_m\right)$

(c) 更新 $f_m(x) = f_{m-1}(x) + T\left(x; \Theta_m\right)$

end for

得到回归问题提升树 $f_M(x) = \sum_{m=1}^{M} T\left(x; \Theta_m\right)$

输出: 提升树 $f_M(x)$

每一步的树模型可以表示为：

$$T(x;\Theta)=\sum_{j=1}^{J}\gamma_j I\left(x\in R_j\right),$$

$\Theta=\{R_j,\gamma_j\}_1^J$. J 是一个元参数. 需要最小化下列目标函数求解参数值：

$$\hat{\Theta}=\arg\min_{\Theta}\sum_{j=1}^{J}\sum_{x_i\in R_j}L\left(y_i,\gamma_j\right).$$

6.2.4 梯度提升

提升树利用加性模型与前向分步算法实现学习的优化过程. 当损失函数是平方损失和指数损失函数时, 每一步优化是很简单的, 但对一般损失函数而言, 往往每一步优化并不那么容易. 针对这一问题, Freidman 提出了梯度提升 (gradient boosting) 算法. 这是利用最速下降法的近似方法, 其关键是利用损失函数的负梯度在当前模型的值

$$-\left[\frac{\partial L\left(y,f\left(x_i\right)\right)}{\partial f\left(x_i\right)}\right]$$

作为回归问题提升树算法中的残差的近似值，拟合一个回归树.

Algorithm 6 回归树模型的梯度提升算法

输入: 训练数据集 $T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\},x_i\in\mathcal{X}\subseteq\mathbf{R}^n,y_i\in\mathcal{Y}\subseteq\mathbf{R}$ 损失函数 $L(y,f(x))$:

初始化

$$f_0(x)=\arg\min_{c}\sum_{i=1}^{N}L\left(y_i,c\right)$$

for $m=1,2,\cdots,M$ **do**

(a) 对 $i=1,2,\cdots,N$ ， 计算

$$r_{\mathrm{mi}}=-\left[\frac{\partial L\left(y_i,f\left(x_i\right)\right)}{\partial f\left(x_i\right)}\right]_{f(x)=f_{m-1}(x)}$$

(b) 对 r_{mi} 拟合一个回归树, 得到第 m 棵树的叶结点区域 $R_{mj},j=1,2,\cdots,J$

(c) 对 $j=1,2,\cdots,J$, 计算

$$c_{mj}=\arg\min_{c}\sum_{x_i\in R_{mj}}L\left(y_i,f_{m-1}\left(x_i\right)+c\right)$$

(d) 更新 $f_m(x)=f_{m-1}(x)+\sum_{j=1}^{J}c_{mj}I\left(x\in R_{mj}\right)$

end for

得到回归树

$$\hat{f}(x)=f_M(x)=\sum_{n=1}^{M}\sum_{j=1}^{J}c_{mj}I\left(x\in R_{mj}\right)$$

输出: 回归树 $\hat{f}(x)$

算法第 1 步初始化, 估计使损失函数极小化的常数值, 它是只有一个根结点的树. 第 2 (a) 步计算损失函数的负梯度在当前模型的值, 将它作为残差的估计. 对于平方损失函数, 它就是通常所说的残差; 对于一般损失函数, 它就是残差的近似值. 第 2 (b) 步估计回归树叶结点区域, 以拟合残差的近似值. 第 2 (c) 步利用线性搜索估计叶结点区域的值, 使损失函数极小化. 第 2 (d) 步更新回归树. 第 3 步得到输出的最终模型 $\hat{f}(x)$。

6.2.5 案例：BOOSTING DIFFUSION INDICES（Bai 和 Ng，2009）

为从 N 个变量、N 个变量的主成分因子及其各自滞后项中选择预测变量，Bai 和 Ng（2009）使用了 boosting 方法。

（1）Algorithm 1 Component-wise L2 boost W

即分量逐步 L_2 提升，该方法将每个变量的每个滞后视为一个单独的预测器。

当潜在预测变量的数量很大时，便捷的方法，是一次使用一个预测变量来拟合学习器。假设 $z_{.,i}$ 表示第 i 个变量的 T 个观测值形成的向量。$z_{.,i_m^*}$ 为在第 m 轮选中的预测变量：

$$i_m^* = \arg \min_{i=1,...n} \sum_{t=1}^{T} (u_t - \hat{\Phi}_m(z_{t,i}))^2$$

也就是说，在第 m 步中，变量 i_m^* 是在所有考虑的预测变量中，拟合得到的残差平方和最小的变量。

用线性最小二乘作为基学习器的分量逐步 L_2 提升的算法过程如下：

1. 让 $\hat{\Phi}_{t,0} = \bar{Y}$ 对每个 t。
2. 对 $m = 1, ..., M$：
   (a) 对 $t = 1, ..., T$，$u_t = Y_t - \hat{\Phi}_{t,m-1}$ 为“当前残差”；
   (b) 对每个 $i = 1, ..., n$，将当前残差向量 u 对 $z_{.,i}$（第 i 个回归量）做回归，得到 $\hat{b}_i$。计算 $e_{.,i} = u - z_{.,i}\hat{b}_i$，以及 $SSR_i = e'_{.,i}e_{.,i}$；
   (c) 求解 i_m^*，使得 $SSR_{i_m^*} = \min_{i=[1,...n]} SSR_i$；
   (d) 让 $\hat{\Phi}_{.,m} = z_{.,i_m^*}\hat{b}_{i_m^*}$。
3. 对 $t = 1, ..., T$，更新 $\hat{\Phi}_{t,m} = \hat{\Phi}_{t,m-1} + \nu\hat{\Phi}_{.,m}$，其中 $0 < \nu \leq 1$，是步长。

如果 Z_t 是 $N \times 1$ 且考虑了最大滞后数，那么将有 $n = N \times \text{pmax}$ 个元素在 z_t 中。分量逐步提升处理这些 n 个变量作为独立变量。

（2）Algorithm 2 Block-wise L2 boost

即区块逐步 L_2 提升，该方法将变量及其滞后视为一组，一同进行考虑。

对每个 t，令 $\hat{\Phi}_{t,0} = \bar{Y}$。对 $m = 1, ..., M$：

(a) 对 $t = 1, ..., T$，$u_t = y_t - \hat{\Phi}_{t,m-1}$ 为‘当前残差’；

(b) 对每个 $i = 1, ..., N$：

(i) 对 $p = 1, ..., \text{pmax}$，估计模型 $u_t = a_1 Z_{t,i} + a_2 Z_{t-1,i} + ... + a_p Z_{t-p,i} + v_{t,p}$。基于 $IC(p) = \log(\hat{\sigma}^2_{v_p}) + A_T \frac{p}{T}$ 计算滞后阶数，其中 $\hat{\sigma}^2_{v_p} = T^{-1} \sum_{t=1}^{T} v^2_{t,p}$，$A_T$ 取决于对简约模型的需求。

(ii) 让 $p^*_i = \arg\min_p IC(p)$；

(iii) 将 u_t 对 $Z_{t,i}$ 进行回归，得到 $\hat{b}_i$，其中 $Z_{t,i} = (Z_{t,i}, ..., Z_{t-p^*_i,i})'$。计算 $e_{.,i} = u - Z_{.,i}\hat{b}_i$，以及 $SSR_i = e'_{.,i} e_{.,i}$；

(c) 求解 i^*_m，使得 $SSR_{i^*_m} = \min_{i=[1,...N]} SSR_i$；

(d) 令 $\hat{\Phi}_{.,m} = Z_{.,i^*_m} \hat{b}_{i^*_m}$。

对 $t = 1, ..., T$，更新 $\hat{\Phi}_{t,m} = \hat{\Phi}_{t,m-1} + v\hat{\Phi}_{.,m}$，其中 $0 < v \leq 1$ 是步长。

通过上述两种算法，可以使用 boosting 方法实现对动态数据的变量选择。

6.3 stacking 算法原理

堆叠泛化 (Stacking) 是一个集成学习框架，它训练一个单独的 ML 算法来组合来自两个或多个集成成员的预测。它是由 Wolpert 在 1992 年引入的，用于减少机器学习问题中的泛化误差。在几个 ML 模型在特定任务上具有独特技能的情况下，堆叠是有用的; 然后，堆叠方法将使用一个单独的 ML 模型来学习何时使用来自各种模型的预测

具体来说，它涉及到使用多个基本算法 (称为 0 级模型) 和元学习算法 (训练另一个模型来组合来自基本模型的预测) 来构建模型。元模型被称为一级模型。堆叠的核心思想是使用训练数据集训练 0 级基础学习器，并提供样本外或未见过的数据; 他们在未见数据上预测的目标标签，与实际标签一起，形成用于训练元学习器的新数据集的输入和输出对。

元学习是机器学习的一部分，其中算法使用其他 ML 算法的输出进行训练，并根据其他基本分类器的预测做出更准确的预测。如图6.1所示，元学习器是堆叠框架的一个组成部分，因为它训练模型进行最终的预测。元分类器学习如何最好地结合基础学习器的预测。堆叠方法是强大的，因为它利用多个性能良好的分类器的优势来进行分类，这些分类优于组成整体的单个模型。

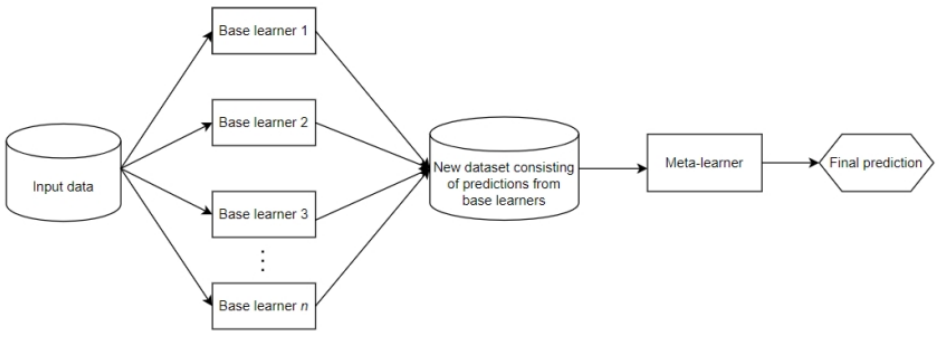


FIGURE 3. Block diagram of the Stacking framework.

图 6.1: stcaking 算法结构

此外，叠加使用不同的基本算法和相同的数据集来获得多样化的模型，并以不同的方式处理预测建模问题。与 bagging 不同，bagging 主要使用在输入数据子集上训练的决策树模型，堆叠模型使用不同

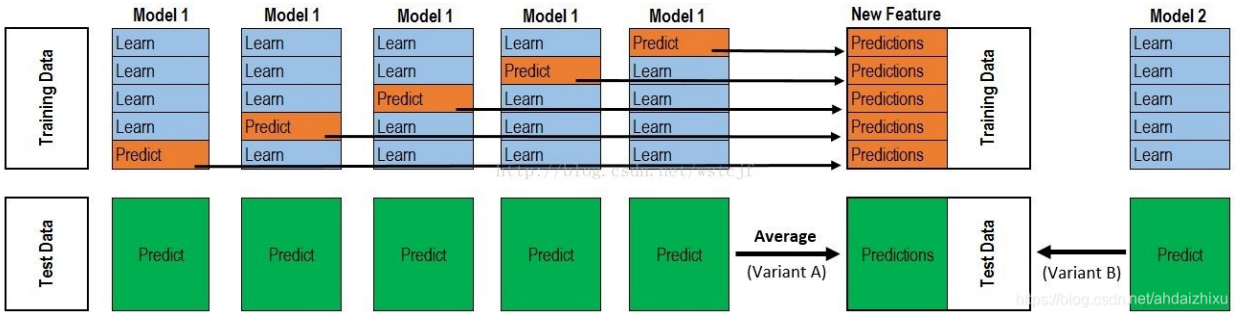


图 6.2: stacking 算法流程

的算法，并在相同的数据集上训练。此外，与增强不同，增强是通过顺序训练模型来纠正先前模型的预测，堆叠使用单个模型来学习如何最佳地组合来自基础学习器的预测。同时，元模型通常很简单，因为它从 0 级模型的预测中学习。因此，线性分类器，如逻辑回归，经常被用作元学习器。然而，在回归问题中，线性回归主要被用作元分类器。堆叠框架分为以下几个步骤:

- 步骤 1: 将选择的算法应用于使用训练数据训练基本模型。基本算法的选择取决于特定的用户和问题域。
- 步骤 2: 使用基本模型的输出来创建一个新的数据集。具体而言，将基本模型的预测目标标签作为新特征，将实际目标标签作为新数据集中的目标变量。例如，如果 S 中的每个实例都是 $\{x_i, y_i\}$，则在新数据集中获得一个等效的实例 $\{\hat{x}_i, y_i\}$，其中 $\hat{x}_i = \{h_1(x_i), h_2(x_2), ..., h_T(x_i)\}$。
- 步骤 3: 使用新数据集训练选定的元学习器。

举例说明，假设源数据总共有 12500 条，其中 10000 条作为训练集，2500 条数据作为测试集。将这 10000 条数据进行分组，例如分为 5 组，每份 2000 条数据。

取其中 4 份数据作为训练集，将剩余一份数据作为验证集，对模型进行训练，得到 2000 条验证集的结果，再将得到的模型对 2500 条数据的测试集进行测试，得到 2500 条测试集结果。

对于一个模型，将上述步骤重复进行，直至所得到的验证集结果数据包含所有的训练集数据。对这 5×2000 条验证结果，5×2500 测试集结果，记为 A_1，对 5×2500 测试集结果进行加权平均，记为 B_1

对每一个模型进行上述操作, 假设共有 M 个模型，可以得到 $A_1, A_2, ..., A_i, ..., A_M$，将这些矩阵合并可以得到新的训练集数据，以及 $B_1, B_2, ..., B_i, ..., B_M$ 进行合并作为新的测试集数据，输入到下一层的训练器中。如图6.2。

stacking 的主要好处是，堆叠模型通常利用几个性能良好的算法来进行分类，这些分类比用于构建集成的任何单个算法都要好。通常，堆叠模型具有很高的精度，这是它们被应用于多次 ML 竞赛并赢得胜利的重要原因。此外，由于使用了不同的 ML 算法来训练基础模型，因此堆叠模型提高了多样性。例如，使用分解模型 (如矩阵分解) 和基于树的模型 (如决策树和随机森林) 作为基础学习器可以提供很好的多样性，因为前者的训练方式与后者截然不同。同时，堆叠模型的主要限制是当训练数据集较大时，由于使用整个数据集来训练每个基分类器，因此计算时间较高。

Stacking 注意事项:

做 Stacking 模型融合时需要注意以下个点，我们拿 2 层 stacking 模型融合来举例子：

第一层的基模型最好是强模型，而第二层的基模型可以放一个简单的分类器，防止过拟合。

第一层基模型的个数不能太小，因为一层模型个数等于第二层分类器的特征维度。大家可以把勉强将其想象成神经网络的第一层神经元的个数，当然这个值也不是越多越好。

第一层的基模型必须" 准而不同"，如果有一个性能很差的模型出现在第一层，将会严重影响整个模型融合的效果。

6.4 人工蜂群算法 ABC

此部分选取了 2013 年发表于 Swarm and Evolutionary Computation 期刊的《Optimization of stacking ensemble configurations through Artificial Bee Colony algorithm》文章，用以介绍与 stacking 集成方法相关的算法：人工蜂群-stacking 算法。传统的 stacking 集成方法对基础级分类器的预测能力有较强的敏感性，当基础分类器的预测能力较弱时，可能导致最终的预测结果产生较大的偏误。人工蜂群 ABC-stacking 算法相比于一般 stacking 算法，它的优势在于可以通过计算不同基础级分类器对信息的预测能力，选择合适的基础分类器来改善预测性能，从而一定程度上解决传统 stacking 集成方法的弊端。这篇文章详细讲解了基础人工蜂群算法 ABC-stacking 以及对基础人工蜂群算法 ABC-stacking 的改进算法。

ABC 算法是一种基于蜂群智能觅食行为的群体智能算法。该智能算法由 Karaboga 提出，自提出以来，该算法已被用于解决大量领域的优化问题。ABC 中的觅食行为模型包括食物来源和三种蜜蜂: 雇主蜂、围观者蜂和侦察兵蜂。食物源代表优化问题的可能解。食物来源的花蜜含量与溶液的适合度相对应。ABC 算法在蜂群中使用了三种蜜蜂: 雇佣蜜蜂、围观者蜜蜂和侦察蜜蜂。最初，生成食物来源位置 (N)。受雇蜜蜂的数量等于食物来源的数量。每只被雇佣的蜜蜂被分配一个食物来源。受雇蜂利用食物来源，并将花蜜数量的信息传递给旁观的蜜蜂。受雇蜜蜂和旁观蜜蜂的数量是相等的。旁观蜜蜂根据受雇蜂提供的花蜜信息，对食物来源和周围环境进行开发，直到食物来源枯竭。耗尽食物来源的蜜蜂变成了侦察兵。然后侦察兵开始寻找新的食物来源位置。花蜜信息代表了从食物来源获得的溶液的质量。花蜜数量的增加增加了旁观蜜蜂选择特定食物来源的可能性。

6.4.1 ABC-stacking 1 方法

在本研究中，ABC 在 ABC- stacking1 中优化了基础层的堆叠集成分类器的选择，而在 ABC- stacking2 中，蜜蜂在基础层优化了分类器的选择，同时优化了元层分类器的选择。堆叠、ABC-Stacking1 和 ABC-Stacking2 的示意图如图6.3所示。

在本文提出的 ABC Stacking 算法中，ABC 算法优化了堆叠的分类器配置。给定一组由 sn 个分类器组成的基本分类器 C，蜜蜂对它们进行探索和利用，得到一个由 n 个分类器组成的子集 $S(n \leq sn; SC)$ 以达到最大可能的预测精度。

在一般问题的蜂群算法中，x_i 通常是用来作为优化目标的近似解，但是在预测模型中，由于优化目标是希望预测值能更贴近真实值，因此，x_i 在此时作为预测精度（预测误差）来表示，即预测值对真实值的偏离。在这层含义下，我们可以进行下面的讨论。

在初始阶段，生成基本级分类器池，$C = c_i : i = 1, 2, ..., sn$，决策树 J48 作为元分类器，用于构建堆叠集成。然后，生成由受雇蜂和旁观者蜂组成的蜜蜂群。蜜蜂种群由相等数量的受雇蜜蜂和旁观蜜蜂 (sn) 组成，这等于基本级分类器集中分类器的数量。每个被雇佣的蜜蜂被分配一个基本分类器。首先通过 10 折交叉验证计算分类器的预测精度 (预测误差)(x_i)。目标值 (f_i) 被赋值为预测精度 $f_i = (x_i)$，旨在通过最优配置分类器来最大化预测精度。然后，被雇佣的蜜蜂使用式6.5来评估分类器的适应度拟合度:

$$fit_i = \frac{1}{1 + f_i} \tag{6.5}$$

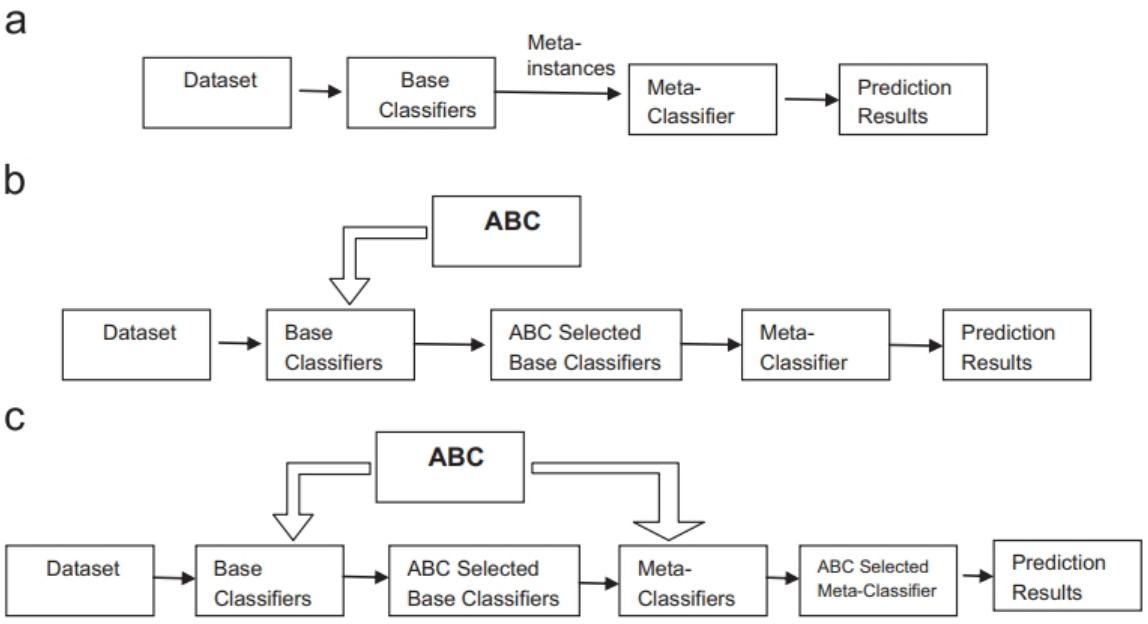


Fig. 3. Schematic diagrams of (a) Stacking, (b) ABC-Stacking1 and (c) ABC-Stacking2.

图 6.3: ABC 堆叠结构

旁观蜜蜂通过与雇佣蜜蜂的交流，获得适合度和准确性信息。然后，每个分类器的围观蜂进行邻域搜索，并根据与该分类器关联的概率值 p_i 选择一个分类器，其计算公式如下:

$$p_i = \frac{fit_i}{\sum_{i=1}^{sn} fit_i} \tag{6.6}$$

然后使用已分配给受雇蜜蜂的分类器的精度 x_i 和旁观者蜜蜂当前选择的分类器 x_j，旁观者蜜蜂使用式6.7计算新的 v_i:

$$v_i = x_i + \phi_i(x_i - x_j) \tag{6.7}$$

在 $j \in 1, 2, ..., sn$ 是一个随机选择的索引，j 必须不同于 i;ϕ_i 是一个在 [−1,1] 范围内的随机数，用于控制分类器 c_i 邻域内分类器的选择。围观蜜蜂应用贪婪选择将选择的分类器包含在最佳分类器配置中。如果 v_i 大于 x_i，旁观者蜜蜂将选择的分类器添加到最佳分类器 S_i 的子集中，否则 S_i 保持不变。S_i 中的分类器是堆叠的，所使用的蜜蜂现在被分配为集成分类器，集成精度现在将作为新的 x_i。

因此，每当蜜蜂完成一个开采周期，花蜜含量就会积累到更好的分类器配置。如果一个受雇蜜蜂的解决方案在预定的循环次数 (极限) 内没有改进，那么该分类器的受雇蜜蜂将成为侦察员，并被分配一个新的分类器源。新的分类器位置使用式6.8计算。

$$x_i^j = x_{min}^j + rand[0,1](x_{max}^j - x_{min}^j) \tag{6.8}$$

在式6.8，x_{max}^j 和 x_{min}^j 分别代表参数 j 的上界和下界。蜜蜂继续探索，生成分类器子集配置，直到找到更好的配置。在整个搜索过程完成后，将得到最终的最优配置。ABCStacking1 算法的步骤如图6.4所示。

1. *Cycle* = 1
2. Initialize the classifier positions c_i and compute the accuracy of the classifiers $x_i, i = 1, 2, .., sn$
3. Evaluate the fitness fit_i of the classifiers
4. Repeat
   a) Employed Bee
      For i from 1 to sn, for each employed bee
      - Initialize its configuration $S_i = \{c_i\}$
      - Produce new classifier combinations v_i suggested by the onlooker bee
      - Evaluate the fitness fit_i of the ensemble
      - Calculate the probability p_i of the ensemble solution

   b) Onlooker Bee
      For j from 1 to sn, for each onlooker bee
      - Set the flag of adding a new classifier to TRUE
      - Select a classifier c_i depending on the value of p_i
      - Compute v_i using x_i and x_j
      - Compare v_i and x_i
      - IF v_i is greater THEN $S_i = S_i \bigcup \{c_j\}$
      - ELSE set the flag to FALSE

   C) Scout Bee
      - If there exists any employed bee, whose solution has not been improved THEN replace it with new classifier positions

   d) Nectar quantity is increased when an iteration finishes
   e) *cycle* = *cycle* + 1

   Until the maximum limit of *cycle* is reached
5. Use the same searching procedure of bees to generate the optimal classifier configuration of stacking

Fig. 4. The ABC-Stacking1 algorithm.

图 6.4: ABCStacking1 算法步骤

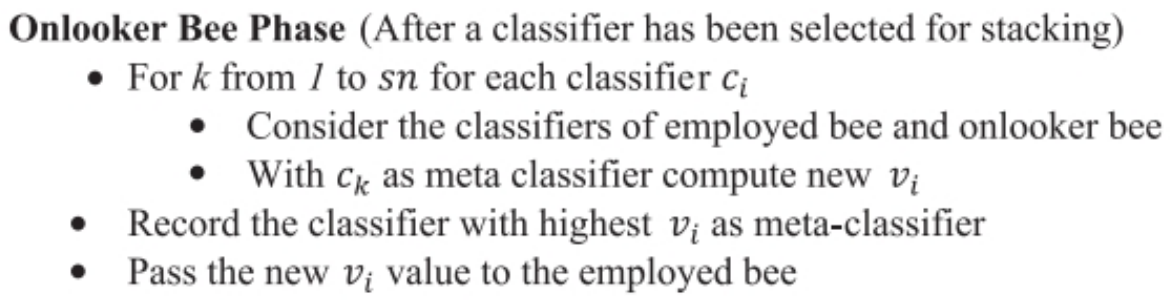


图 6.5: ABC-Stacking 2 额外算法

6.4.2 ABC-stacking 2 方法

在 ABC-Stacking2 的构造中,除了蜜蜂在每次堆叠时寻找最佳元分类器外,采用了与 ABCStacking1 相同的步骤。在 ABC-Stacking1 中，每次围观蜜蜂决定将一个分类器包含到最佳分类器配置中时，该配置的组成分类器使用固定的元分类器进行堆叠。在 ABC-Stacking2 中，一旦一个分类器被添加到配置中，蜜蜂就会搜索可用分类器的空间，并选择一个对该配置的预测准确性最大化的元分类器。因此，在 ABC-Stacking2 中，生成最优分类器配置，每个配置都有自己的最佳元分类器。

图6.5显示了在 ABC-Stacking1 中添加的额外算法步骤，以获得 Stacking2 算法。在围观蜜蜂阶段，一旦围观蜜蜂决定了要堆叠的分类器，在堆叠之前，将堆叠精度分配给被雇佣的蜜蜂，执行图6.4所示的步骤。这将导致分类器在基本级别上的最优配置，并由最佳元分类器堆叠所选的基本分类器。

6.4.3 学习算法

基分类器池是利用 WEKA 中的 10 种分类器算法生成的。我们将 Chen 和 Wong 在 ACO 堆叠中使用的 10 种分类器算法作为基分类器，将 ABC-Stacking2 中使用的 10 种分类器算法作为基分类器和元分类器。采用相同分类器算法的原因是为了比较和证明 ABC 与 ACO-Stacking 相比有所进步。这些分类器算法有 Naïve Bayes (NB)、Logistic、IB1、IBk (k=5)、KStar、OneR、PART、ZeroR、Decision Stump 和 J48 Decision Tree。它们被简单地讨论如下：

Naïve Bayes 朴素贝叶斯：它基于一个简化的假设，即给定目标值，属性值是有条件独立的。

Logistic：通过分析因变量和自变量之间的关系以及建立多项回归模型来进行预测

IB1 ：这是基于实例的最近邻分类器，它使用欧几里得距离执行搜索。测试实例的类基于与其相似的训练实例的类。

IBk ：这与 IB1 类似，但使用 k 个最近邻而不是一个。对于本次实验，IBk 的 k 值设为 5。

KStar ：这也是一个基于实例的分类器。它与其他基于实例的学习器的不同之处在于它使用基于熵的距离函数。

ZeroR：它是最简单的依赖目标值而忽略所有预测因子的分类方法。它只是预测多数类别。

OneR：它为数据中的每个预测器生成一条规则，然后选择总误差最小的规则作为其预测的“一条规则”。

Table 2
ABC parameters.

Parameters	Values
Population size	$2 \times sn$
Maximum number of iterations (*limit*)	100
No. of runs	30
φ	[−1, 1]

图 6.6: ABC 参数

PART：在每次迭代中构建局部决策树，并根据找到的最佳叶节点对实例进行分类。

Decision Stump：决策树桩——生成用于预测实例的一级决策树。

J48 Decision Tree：决策树分类是通过生成决策树来完成的，决策树的叶子表示与不同类类型相关的条件概率。

6.4.4 参数设置

ABC-Stacking 的参数初始化如图6.6所示。蜂群由数量相等的受雇蜜蜂和旁观蜜蜂组成。食物来源的数量等于受雇蜜蜂或围观者蜜蜂的数量。在 ABC-Stacking 中，分类器被视为食物来源，因此群体的种群规模被设置为分类器数量的两倍。ϕ 取 [−1,1] 范围内的随机值。当采用 ABC-Stacking ϕ 来控制分类器与分类器组合的选择时，被雇佣蜜蜂和旁观蜜蜂正在进行搜索。

通过这些参数设置，蜜蜂搜索和蜜蜂生成的最终分类器配置用于构建堆叠。ABC 最优选择基分类器构成 Stacking Ensemble 的过程如图6.7所示。在基础水平上考虑 10 个分类器，因此蜜蜂种群由 10 个受雇蜜蜂和 10 个旁观蜜蜂组成。每个被雇佣的蜜蜂从基本分类器池中分配一个分类器。在被雇佣蜂阶段和旁观者蜂阶段执行后，得到待堆叠的基级分类器的配置。“勾号”标记表示在基础级别选择了特定的分类器，“错误”标记表示未选择分类器。

在这项工作中，我们使用了分而治之的原则来预测分层类。在使用分而治之的方法时，分类器在每个层次结构级别上进行训练，生成一个分类器树。根分类器使用所有的训练实例进行训练。然后，在下一个类级别，使用属于这些类的实例子集来训练分类器。在测试阶段，以自顶向下的方式对实例进行分类。重复这个过程，直到到达一个叶节点类，并且无法进行额外的预测。这些步骤都是在不影响可靠性的情况下进行的。

计算涉及使用 10 倍交叉验证。在交叉验证过程中，每个数据集被分成 10 个大小相等的折叠，并执行 10 次。因此，当执行 ABCStacking1 算法时，在选择要放置到基分类器配置中的分类器时，比较 10 次迭代中每次迭代的预测准确性 (交叉验证)。在 ABC-Stacking2 中，执行上述步骤后，围观蜜蜂只有在比较使用各种元分类器生成的堆叠配置的预测精度后，才会将新的堆叠配置分配给被雇佣蜜蜂。基础分类器配置和 ABC 算法的元分类器选择都遵循分而治之的方法。

在我们的实验中，我们使用了两种方法来评估所提出的模型: 预测精度 (PA)，均方根误差 (RMSE)。

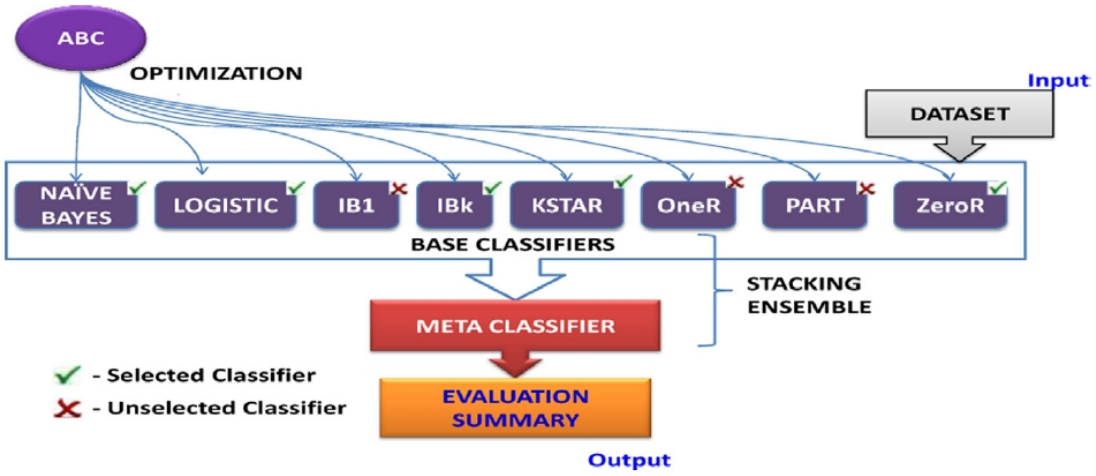


Fig. 6. Optimal selection of base-level classifiers by ABC-Stacking1.

图 6.7: 基于 ABC-stacking1 算法选择最优分类器

其中，预测精度 (PA) 由式6.9定义：

$$PA = TPR \times TNR \; where \; TPR = \frac{TP}{TP+FN} TNR = \frac{FN}{TN+FP} \tag{6.9}$$

均方误差 (RMSE) 为：

$$RMSE = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(\mu_i' - \mu)^2} \tag{6.10}$$

通过目标函数使用蜂群算法对问题进行优化，可以的到问题的一般近似解。相比于 ABC-stacking1 方法，文章提供的 ABC-stacking2 方法的预测能力有着较大的改善，具体结果在此不进行详细说明。但是，需要指出的是，蜂群算法并不是完美无瑕。对于一些问题，每次计算得到的结果可能有较大的差异，这可能是因为蜂群在选择基础分类器时的随机性导致的。因此，蜂群算法在计算的收敛性上的表现可能并不足够令人满意的。

第七章　贝叶斯模型平均与时变和断点结构

7.1　贝叶斯模型平均方法介绍

所谓贝叶斯模型平均，是当研究者不知道真正的模型，但有几个候选模型时考虑预测。可以通过对每个模型的预测赋予权重来构建预测。如果这些权重都相等，那么这就是简单的预测平均。但是，还可以从所有模型都同样好的先验出发，估计模型的后验概率，将后验概率用作权重。贝叶斯模型平均起源于 Leamer(1978). 在 2000 年左右得到统计和计量经济学界的关注。

考虑 n 个备选模型 $M_1, \dots M_n$. 每个模型的参数向量用 θ 表示。假设有一个备选模型是正确的模型，但是不清楚正确的模型到底是哪一个。[1]同时，假设我们的先验信念为：第 i 个模型是正确模型的概率为 $P(M_i)$。之后，我们可以结合样本数据 D，更新第 i 个模型是正确模型的后验概率：

$$P(M_i \mid D) = \frac{P(D \mid M_i) P(M_i)}{\sum_{j=1}^{n} P(D \mid M_j) P(M_j)} \tag{7.1}$$

其中

$$P(D \mid M_i) = \int P(D \mid \theta, M_i) P(\theta \mid M_i) \, d\theta$$

是第 i 个模型的边际概率。$P(\theta \mid M_i)$ 为模型中参数向量的先验密度，$P(D \mid \theta, M_i)$ 是似然（likelihood）。每个模型可以得到一个预测密度 $f_1, \dots f_n$。在不确定哪个模型是正确模型的情况下，预测密度为：

$$f^* = \sum_{i=1}^{n} P(M_i \mid D) f_i$$

同样，每个模型存在一个点预测，在不确定哪个模型为真实模型的情况下，通过模型的后验对每个模型的预测进行加权得到点预测值。

关键设定：模型的先验 $P(M_i)$、参数的先验 $P(\theta \mid M_i)$。其他内容通过计算得到。

假设模型为线性模型，具体来说，第 i 个模型为：

$$y = X\beta + \varepsilon$$

其中 y 是响应变量，X 是预测变量矩阵，β 是 px1 的参数向量，$\varepsilon = (\varepsilon_1, \dots \varepsilon_T)'$ 为扰动项向量，T 是样本规模。假设误差项是 MA(h-1) 过程，且方差 σ^2 满足：

$$\mathrm{Cov}(\varepsilon_t, \varepsilon_{t-j}) = \sigma^2 \frac{h-j}{h}, j \le h-1$$

[1]Bernardo and Smith (1994) and Key, Perrichi and Smith (1998) 考虑了实际上没有任何一个模型是正确的情况下的贝叶斯模型平均的理论框架

将模型先验设定为：$P(M_i) = \frac{1}{n}$. β 的先验设定为自然共轭 g 先验 (Zellner (1986)), 即 β 关于 σ^2 的条件分布为 $N\left(0, \phi\sigma^2 (X'X)^{-1}\right)$. 参考 Fernandez, Ley 和 Steel (2001) 等，假设 σ^2 的先验是与 $1/\sigma^2$ 成正比，这是一个不合适的（improper）先验. 则

$$P(D \mid M_i) = \int P(D \mid M_i, \beta, \sigma) P(\beta \mid M_i, \sigma) P(\sigma \mid M_i) d\beta d\sigma$$

$$\propto \int \frac{1}{\sigma^2} \exp\{-\frac{1}{2\sigma^2}(y - X\beta)'V^{-1}(y - X\beta) - \frac{1}{2\phi\sigma^2}\beta' X'X\beta\} d\beta d\sigma^2$$

其中

$$V = \begin{pmatrix} 1 & \frac{h-1}{h} & \cdots & \frac{1}{h} \\ \frac{h-1}{h} & 1 & \cdots & \frac{2}{h} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{1}{h} & \frac{2}{h} & \cdots & 1 \end{pmatrix}$$

进行积分，可以得到：

$$P(D \mid M_i) = \frac{\Gamma(T/2)}{\pi^{T/2}}(1 + \phi)^{-p/2} S^{-T/h}$$

其中

$$S^2 = y'y - y'X (X'X)^{-1} X'y \frac{\phi}{1 + \phi}$$

结合模型先验和参数先验，可以估计每个模型的方程 (1)，进而得到每个模型的后验概率，为预测分配权重。β 的先验以零为中心，因此在每个模型中参数都向零缩小，这对应于无可预测性。收缩的程度由 ϕ 决定，较小的 ϕ 值意味着更大的收缩，并使先验信息更丰富。也就是说，ϕ 越大，我们越愿意偏离模型先验来响应我们在数据中观察到的东西。反之，ϕ 越小，得到的后验会越接近先验。

7.2 案例：Forecasting US inflation by Bayesian model averaging（Wright，2009）

以往研究存在一个问题：当一个模型相对于原始时间序列预测具有预测能力时，它往往是不稳定的。也就是说，在一个子周期中具有良好预测能力的模型在另一个子周期中具有很少或没有良好预测能力的倾向。为此，Stock 和 Watson (2001,2002a) 认为，最好的预测性能是通过从大量模型中构建预测并简单地平均这些预测来获得的。Wright（2009）则通过研究指出贝叶斯平均模型通常比简单的等加权平均模型在预测美国通胀方面做得更好。在不同的通胀指标和数据区间之间，都是如此。在该文中：

预测函数形如：

$$\pi_{t,t+h} = \alpha + \gamma Z_t + \rho\pi_{t-h,t} + \varepsilon_t \tag{7.2}$$

其中 $\pi_{t,t+h}$ 是从 t 到 t + h, h 时刻的通胀率，Z_t 是一个预测变量[2]，ε_t 是误差项，假设误差项是 MA（h-1）过程。

则贝叶斯模型平均的预测值为：

$$\sum_{i=1}^{n} P_t(M_i \mid D)(\hat{\alpha}_t + \hat{\gamma}_t Z_t + \hat{\rho}_t \pi_{t-h,t}) \tag{7.3}$$

[2]该文需要估计 93 个模型的概率，因此计算压力较小，没有使用 MCMC 等方法，但是标准的贝叶斯模型平均可能需要对 2^{93} 个模型进行计算和组合，则可能会用到 MCMC 等方法。

其中 $P_t\left(M_i \mid D\right)$ 为基于第 t 期和第 t 前之前的数据计算的模型 i 为真实模型的后验概率。作者假设所有模型的先验概率相等，并估计了 $\phi = 20, 5, 2, 1, 0.5$ 时的取值结果。通过比较发现，贝叶斯模型平均预测结果由于以下两种基准模型：

基准模型一：AR 模型

$$\pi_{t,t+h} = \alpha + \rho\pi_{t-h,t} + \varepsilon_t \tag{7.4}$$

基准模型二：等权平均模型：

$$\frac{1}{n}\sum_{i=1}^{n} \hat{\alpha}_t + \hat{\gamma}_t Z_t + \hat{\rho}_t \pi_{t-h,t} \tag{7.5}$$

其中 $\hat{\alpha}_t, \hat{\gamma}_t$、$\hat{\rho}_t$ 是使用 t 时刻和更早时刻数据估计的公式7.2的 OLS 估计的参数。

7.3 时变与结构断点模型（Groen 等，2013）

该节以 Groen 等 2013 年发表在 Journal of Business & Economic Statistics 上的文章《Real-Time Inflation Forecasting in a Changing World》为例，介绍时变与结构断点模型的设定和估计细节。

该文在简化的菲利普斯模型中引入变量选择与结构断点，并使用贝叶斯方法进行估计。

7.3.1 广义菲利普斯曲线的简约形式

$$y_{t+h} = \beta_0 + \sum_{j=1}^{k_1} \beta_j^a a_{jt} + \sum_{j=1}^{k_2} \beta_j^e e_{jt} + \sum_{j=0}^{k_3} \beta_j^y y_{t-j} + \sigma\varepsilon_t \tag{7.6}$$

$$= \beta_0 + \sum_{j=1}^{k} \beta x_{jt} + \sigma\varepsilon_t; \quad t = 1, \ldots, T-h$$

T 是时间序列样本观测点总数。y_t 是通胀率，$y_t = 100\Delta \ln(P_t) = 100(\ln(P_t) - \ln(P_{t-1}))$，其中 P_t 是一个特定的价格指数。$h > 0$ 是预测期数，
$y_{t+h} = 100\Delta \ln(P_{t+h}) = 100(\ln(P_{t+h}) - \ln(P_{t+h-1}))$. a_{jt} 是 k_1 个实际经济活动和成本变量，e_{jt} 是 k_2 个通胀预期的代理变量，通胀率 y_t 的滞后阶数为 k_3。扰动项 $\varepsilon_t \sim \text{NID}(0,1)$ 且 $\sigma > 0$. 为表示方便，记 $(x_{1,t} \cdots x_{k,t})' = (a_{1,t} \cdots a_{k_1,t} e_{1,t} \cdots e_{k_2,t} y_t \cdots y_{t-k_3})'$，以及 $k = k_1 + k_2 + k_3$. 文章使用的解释变量包含通胀的滞后项和 14 个其他变量。

核心研究目标是：（1）确定在什么情况下应该包含预测变量的哪种组合。（2）考虑通胀率与预测变量的关系在 1960-2008 年期间可能会发生变化，因此应允许回归参数时变（2）考虑由于经济结构的变化，宏观经济时间序列可能经历方差上的突变，这种突变与均值变化无关，因此应允许误差方差时变。

7.3.2 考虑模型不确定性和结构断点

模型不确定性，反应了指标变量的哪种组合最准确地总结了实际活动、实际成本和通胀动态预期的影响的不确定性。为此对模型做以下拓展：

$$y_{t+h} = \beta_0 + \sum_{j=1}^{k} \delta_j \beta_j x_{jt} + \sigma\varepsilon_t; \quad t = 1, \ldots, T-h, \tag{7.7}$$

其中 $\varepsilon_t \sim \text{NID}(0,1)$. 向量 $D = (\delta_1, \ldots, \delta_k)'$ 是预测变量的选择向量，可以有 2^k 种不同取值。每个模型用索引 $i = (\delta_1, \ldots, \delta_k)_2$ 来表示。

为允许模型回归参数和方差的结构性变化，在模型中引入时变回归参数 β_{jt} 和方差 σ_t：

$$y_{t+h} = \beta_{0t} + \sum_{j=1}^{k} \beta_{jt} x_{jt} + \sigma_t \varepsilon_t; \quad t = 1, \ldots, T-h. \tag{7.8}$$

结构性断点结构由 $k+2$ 个随机变量 κ_{jt} 来体现[3]，当第 j 个变量在 t 时刻有结构性断点，κ_{jt} 取值为 1，否则，κ_{jt} 取值为 0，$j = 0, \ldots, k+1$，$t = 1, \ldots, T$。假设 $\kappa_t = (\kappa_{0,t}, \ldots, \kappa_{k,t}, \kappa_{k+1,t})'$ 是一个 0/1 过程，且各元素互不相关：

$$\Pr[\kappa_{jt} = 1] = \pi_j; \quad j = 0, \ldots, k+1 \tag{7.9}$$

[3]包括截距项、回归系数项和残差项的结构断点

结构断点的规模由一个独立的，均值为 0，方差为 q_j^2，$j=0,\ldots,k+1$ 的随机冲击过程 η_{jt} 体现，即时变过程被定义为：

$$\beta_{jt}=\beta_{j,t-1}+\kappa_{jt}\eta_{jt};\quad j=0,\ldots,k \tag{7.10}$$

$$\ln\sigma_t^2=\ln\sigma_{t-1}^2+\kappa_{k+1,t}\eta_{k+1,t}$$

其中 $\eta_t=(\eta_{0,t},\ldots,\eta_{k+1,t})'\sim \mathrm{NID}(0,Q)$，$Q=\mathrm{diag}\left(q_1^2,\ldots,q_{k+1}^2\right)$.

最终模型形如：

$$y_{t+h}=\beta_{0t}+\sum_{j=1}^{k}\delta_j\beta_{jt}x_{jt}+\sigma_t\varepsilon_t;\quad t=1,\ldots,T-h \tag{7.11}$$

其中 $\varepsilon_t\sim \mathrm{NID}(0,1)$

7.3.3 估计方法

估计和推断

使用贝叶斯模型平均和结构断点方法估计模型。待估参数包括：变量选择向量 $D=(\delta_1,\ldots,\delta_k)'$, 参数存在结构断点的概率 $\pi=(\pi_0,\ldots,\pi_{k+1})'$，断点规模的方差向量 $q=\left(q_0^2,\ldots,q_{k+1}^2\right)'$. 将上述参数用一个维度为 $(3k+4)$ 的向量表示：$\theta=(D',\pi',q')'$.

先验设定包括：假设第 j 个变量的变量选择参数服从伯努利分布

$$\Pr\left[\delta_j=1\right]=\lambda_j\quad \text{for } j=1,\ldots,k. \tag{7.12}$$

假设模型的第 j 个参数存在结构断点的概率服从 Beta 分布

$$\pi_j\sim \mathrm{Beta}\left(a_j,b_j\right)\quad \text{for } j=0,\ldots,k+1 \tag{7.13}$$

参数 a_j 和 b_j 需要根据先验信念进行设定。

假设结构断点规模的方差服从逆伽马-2 分布

$$q_j^2\sim \mathrm{IG}-2\left(\nu_j,\omega_j\right)\quad \text{for } j=0,\ldots,k+1 \tag{7.14}$$

参数 $\nu_j,\omega_j,j=0,\ldots,k+1$，需要根据先验信念进行选择。联合先验 $p(\theta)$ 由上述三种先验的乘积得到。

使用 Geman 和 Geman (1984) 提出的 Gibbs 抽样方法，与 Tanner 和 Wong (1987) 提出的数据增广方法计算后验结果。潜变量 $B=\{\beta_t\}_{t=1}^{T-h}$, $\beta_t=(\beta_{0t},\beta_{1t},\ldots,\beta_{kt})'$, $S=\{\sigma_t^2\}_{t=1}^{T-h}$ 和 $K=\{\kappa_t\}_{t=1}^{T-h}$ 可以根据参数 θ 估计得到.

数据的似然函数，或者说数据和潜在变量的联合分布为：

$$p(y,B,S,K\mid x,\theta)=\prod_{t=1}^{T-h}p\left(y_{t+h}\mid D,x_t,\beta_t,\sigma_t^2\right)p\left(\beta_t\mid\beta_{t-1},\kappa_t,q_0^2,\ldots,q_k^2\right) \tag{7.15}$$

$$p\left(\ln\sigma_t^2\mid\ln\sigma_{t-1}^2,\kappa_{k+1,t},q_{k+1}^2\right)\prod_{j=0}^{k+1}\pi_j^{\kappa_{jt}}\left(1-\pi_j\right)^{1-\kappa_{jt}}$$

其中 $y=(y_1,\ldots,y_T)$,且 $x=(x_1',\ldots,x_T')'$. 根据模型设定过程，$p\left(y_{t+h}\mid D,x_t,\beta_t,\sigma_t^2\right),p\left(\beta_t\mid\beta_{t-1},\kappa_t,q_0^2,\ldots,q_k^2\right)$ 和 $p\left(\ln\sigma_t\mid\ln\sigma_{t-1},\kappa_{k+1,t},q_{k+1}^2\right)$ 为正态分布密度函数。

后验密度函数可由下列公式得到：

$$p(\theta, B, S, K \mid y, x) \propto p(\theta) p(y, B, S, K \mid x, \theta) \tag{7.16}$$

结合 Kuo 和 Mallick (1998) 的算法进行变量选择，结合 Gerlach et al. (2000) 的算法控制结构断点. 定义 $\theta = \left(\bar{\theta}', D'\right)'$，其中 $\bar{\theta} = (\pi', q')'$，$K_\beta = \{\kappa_{0t}, \ldots, \kappa_{kt}\}_{t=1}^{T-h}$，$K_\sigma = \{\kappa_{k+1,t}\}_{t=1}^{T-h}$, 在每次迭代中，循环下列过程：

1. 以 $B, S, K, \bar{\theta}, y$ 和 x 为条件，采样 D
2. 以 $D, S, K_\sigma, \bar{\theta}, y$ 和 x 为条件，采样 K_β.
3. 以 $D, S, K, \bar{\theta}, y$ 和 x 为条件，采样 B.
4. 以 $D, B, K_\beta, \bar{\theta}, y$ 和 x 为条件，采样 K_σ .
5. 以 $B, D, K, \bar{\theta}, y$ 和 x，采样 S.
6. 以 D, B, S, K, y 和 x，采样 $\bar{\theta}$.

详细过程如下：

Step1: 抽样变量选择参数 D

参考 Kuo 和 Mallick (1998) 的方法，假设 $\delta_j = 0$ 和 $\delta_j = 1$ 的后验密度为 p_{j0} 和 p_{j1}，使用下列函数估计变量选择参数的全条件后验概率：

$$\Pr\left[\delta_j = 1 \mid \bar{\theta}, B, S, K, D_{-j}, y, x\right] = \frac{p_{j1}}{p_{j0} + p_{j1}},$$

其中 $j = 1, \ldots, k$, 且 $D_{-j} = (\delta_1, \ldots, \delta_{j-1}, \delta_{j+1}, \ldots, \delta_k)'$.

Step2: 抽样 K_β

使用 Gerlach et al. (2000, Section 3) 的方法，估计结构断点参数 κ_{jt} 的后验。在该方法中，为提高估计的效率，对 κ_{jt} 进行采样时，不以 β_{jt} 为条件。不以 B 为条件的，$\kappa_{jt}, t = 1, \ldots, T, j = 0, \ldots, k$ 的条件后验概率为：

$$\begin{aligned}
&p\left(\kappa_{0t}, \ldots, \kappa_{kt} \mid K_{\beta,-t}, K_\sigma, S, \theta, y, x\right) \\
&\propto p(y \mid K, S, \theta, x) p\left(\kappa_{0t}, \ldots, \kappa_{kt} \mid K_{\beta,-t}, K_\sigma, S, \theta, x\right) \\
&\propto p\left(y_{t+h+1}, \ldots, y_{T-h} \mid y_{h+1}, \ldots, y_{t+h}, K, S, \theta, x\right) \\
&\quad p\left(y_{t+h} \mid y_{h+1}, \ldots, y_{t+h-1}, \kappa_1, \ldots, \kappa_t, K_\sigma, S, \theta, x\right) p\left(\kappa_{0t}, \ldots, \kappa_{kt} \mid K_{\beta,-t}, K_\sigma, S, \theta, x\right),
\end{aligned}$$

其中 $K_{\beta,-t} = \left\{\{\kappa_{js}\}_{j=0}^{k}\right\}_{s=1, s\neq t}^{T-h}$. 由于 κ_{jt} 不取决于 δ_j，密度函数 $p\left(\kappa_{0t}, \ldots, \kappa_{kt} \mid K_{\beta,-t}, K_\sigma, S, \theta, x\right)$ 等价于 $\prod_{j=0}^{k} \pi_j^{\kappa_{jt}} (1 - \pi_j)^{1-\kappa_{jt}}$

参考 Gerlach et al. (2000, Section 3). 可以很容易地进行计算：
$p\left(y_{t+h+1}, \ldots, y_{T-h} \mid y_{h+1}, \ldots, y_{t+h}, K, S, \theta, x\right)$ 和 $p\left(y_{t+h} \mid y_{h+1}, \ldots, y_{t+h-1}, \kappa_1, \ldots, \kappa_t, K_\sigma, S, \theta, x\right)$ 。因为 κ_t 的取值是有限的，the integrating constant can easily be computed by normalization.

Step3: 抽样 B 中的回归参数

使用模拟平滑器（simulation smoother）计算潜在回归参数 B 的全条件后验概率。参考 Carter 和 Kohn (1994)，使用卡尔曼平滑器估计潜在参数的条件均值和方差，初始值设置为均值为 0 的多元正态先验。如果变量 x_j 没有被选中，对于 $t = 1, \ldots, T-h$，k_{jt} 和 β_{jt} 的全条件分布不依赖于数据 y 和 x。直接根据公式**??**和公式**??**，从无条件分布中进行抽样。

Step 4 和 Step5: 抽样方差参数 K_σ 和 S

K_σ 和 S 的抽样方式与上文类似。由于 $\ln \sigma_t^2$ 不能得到一个线性的状态空间模型，不能使用卡尔曼滤波进行计算。因此使用 Giordani 和 Kohn (2007) 进行计算，并将模型重新表述为：

$$\ln\left(y_{t+h} - \beta_{0t} - \sum_{j=1}^{k} \delta_j \beta_{jt} x_{jt}\right)^2 = \ln \sigma_t^2 + u_t$$

$$\ln \sigma_t^2 = \ln \sigma_{t-1}^2 + \kappa_{k+1,t} \eta_{k+1,t},$$

其中 $u_t = \ln \varepsilon_t^2$ ，是自由度为 1 的 $\log \chi^2$ 分布。

参考 Carter 和 Kohn (1994, 1997), Shephard (1994) ，以及 Kim et al. (1998) 使用一个有限的混合正态分布近似 $\ln \chi^2(1)$ 分布。具体来说，考虑 5 个正态分布的混合，因此 u_t 的密度函数为

$$f(u_t) = \sum_{s=1}^{5} \varphi_s \frac{1}{\omega_s} \phi\left(\left(u_t - \mu_s\right)/\omega_s\right)$$

其中 $\sum_{s=1}^{5} \varphi_s = 1$.

μ_s, ω_s^2 和 φ_s 的近似值参考 Kohn (1997, Table 1) 得到。在 Gibbs 采样的每一步中，会从混合分布中估计一部分。基于混合成分的值，可以使用 Kalman 滤波方法进行处理。因此向量 K_σ 和 S 可以以一种与 K_β 和 B 类似的方法抽样得到。

Step6：抽样 $\bar{\theta}$

基于贝叶斯推断的标准方法，对 $\bar{\theta}$ 进行抽样。也就是说，π_j 从 Beta 分布中抽样得到，q_j^2 从逆伽马-2 分布中得到。

预测

条件预测函数如下：

$$p\left(y_{T+h+1} \mid y, x, y_{T+1}, x_{T+1}\right) = \int \cdots \int \sum_{D} \sum_{K} \sum_{\kappa_{T+1}} p\left(y_{T+h+1} \mid D, x_{T+1}, \beta_{T+1}, \sigma_{T+1}^2\right) \tag{7.17}$$

$$p\left(\beta_{T+1} \mid \beta_T, \kappa_{T+1}, q\right) p\left(\sigma_{T+1}^2 \mid \sigma_T^2, \kappa_{T+1}, q\right) \prod_{j=0}^{k+1} \pi_j^{\kappa_{j,T+1}} \left(1 - \pi_j\right)^{1-\kappa_{j,T+1}}$$

$$p(\bar{\theta}, D, B, S, K \mid y) d\beta_{T+1} d\sigma_{T+1}^2 dB dS d\bar{\theta},$$

其中 $p\left(y_{T+h+1} \mid D, x_{T+1}, \beta_{T+1}, \sigma_{T+1}^2\right)$，$p\left(\beta_{T+1} \mid \beta_T, \kappa_{T+1}, q\right)$ 和 $p\left(\sigma_{T+1}^2 \mid \sigma_T^2, \kappa_{T+1}, q\right)$ 由公式（5）-公式（6）得到。$p(\bar{\theta}, D, B, S, K \mid y, x)$ 是公式（11）得到的后验密度函数。预测密度 (12) 由 (6) 中所有可能的模型规格的加权平均值组成，其权重等于后验模型概率。关于结构断裂时间的不确定性，反映在样本内断点 K 的后验分布中。预测分布直接使用前面提到的 Gibbs 抽样得到。使用公式 (4)-公式 (6) 作为数据生成过程来模拟每个 Gibbs 过程中的 y_{T+h} ，基于后验分布中的抽样来替换参数和潜在变量。使用预测的分布的后验的中位数作为点预测值。

7.3.4 实证结果

先验设定原则：

公式7.9中结构断点规模的方差的分布参数：对 $v_j, j = 0, 1, \ldots, k+1$ 赋予较大的值。其含义是, 当 $\Pr[\kappa_{jt} = 1] = 1$ 断裂的大小与平方根 ω_j 成正比。具体来说，设定 $v_j = 100\ j = 0, 1, \ldots, k+1$,$\omega_j$ 的值在 0.01 到 0.5 之间变换。**公式7.8中断点的概率参数：**将其设定为一种严格递减形式，使得全样本中出现的断点数能小于 3。**公式7.7中的变量选择参数：**设置 $\lambda_j = 0.05, j = 1, \ldots, k.$

进行 9000 次 Gibbs 抽样，其中前 1000 次抽样过程为训练过程，在后面 8000 次抽样过程，每次都选取第二次抽样结果，也就是得到 4000 次 MCMC 抽样，基于这部分抽样进行参数估计和推断。

美国通胀率的实时预测

考虑到算力限制，每次预测使用第一期至预测期前一个季度或前一年之间的样本，重新估计公式7.11。并将样本划分为了 1980Q1-2008Q4,1980Q1-1994Q4 和 1995Q1-2008Q4 三个区间进行预测。1960Q1-1979Q4 的数据被作为训练样本。和不同模型的比较结果见图7.1和图7.2。除在 1995Q1-2008Q4 期间（区间 II）的预测效果弱于 UCSV 模型外，作者提出的带有结构断点的时变菲利普斯曲线都具有更好的预测效果。

Table 5: RMSE - PCE

	Horizon: $h = 1$			Horizon: $h = 4$		
	F	I	II	F	I	II
BMASB	0.39	0.37	0.41	0.43	0.42	0.44
univariate models						
RW	1.23	1.26	1.18	1.20	1.29	1.12
AR(1)	1.19	1.20	1.17	1.12	1.14	1.10
AR(2)	1.15	1.17	1.14	1.10	1.14	1.07
AR(3)	1.10	1.12	1.09	1.10	1.13	1.07
AR(4)	1.10	1.12	1.09	1.10	1.13	1.07
TVP-AR(4)	1.14	1.16	1.12	1.20	1.29	1.11
UCSV	1.07	1.17	**0.99**	1.07	1.22	**0.95**
Linear	1.04	1.09	1.00	1.18	1.28	1.08
Ridge	1.06	1.06	1.06	1.03	1.06	1.00
BMA	1.05	1.07	1.04	1.26	1.41	1.09
multivariate models						
VAR(4)	1.08	1.10	1.07	1.05	1.03	1.07
VAR-BMA(4)	1.05	1.05	1.06	1.11	1.20	1.01

Note: The table presents root mean square prediction error (RMSE) of the BMASB Phillips curve-type model (6), the first line, as well as RMSE ratios relative to it for different univariate and multivariate models, see Table 4, for the full 1980Q1-2008Q4 evaluation sample (F) and two subsamples (I: 1980Q1-1994Q4, II: 1995Q1-2008Q4) at one-quarter ($h = 1$) and one-year ($h = 4$) ahead forecasting horizons for inflation. **Bold** indicates when our BMASB Phillips curve forecasts are outperformed by any of the forecasts from the competing models.

图 7.1: 预测性能比较 1

Table 6: RMSE - GDP deflator

	Horizon: $h = 1$			Horizon: $h = 4$		
	F	I	II	F	I	II
BMASB	0.27	0.31	0.22	0.32	0.36	0.27
univariate models						
RW	1.25	1.19	1.36	1.10	1.11	1.08
AR(1)	1.21	1.15	1.34	1.10	1.09	1.13
AR(2)	1.15	1.11	1.23	1.09	1.09	1.10
AR(3)	1.08	1.04	1.17	1.09	1.10	1.09
AR(4)	1.09	1.04	1.18	1.09	1.11	1.07
TVP-AR(4)	1.11	1.09	1.14	1.22	1.09	1.41
UCSV	1.11	1.16	1.03	1.10	1.23	**0.90**
Linear	1.09	1.04	1.19	1.23	1.26	1.18
Ridge	1.04	1.01	1.12	1.01	1.04	**0.97**
BMA	1.13	1.08	1.23	1.30	1.36	1.20
multivariate models						
VAR(4)	1.07	1.05	1.13	1.02	1.02	1.02
VAR-BMA(4)	1.14	1.11	1.21	1.16	1.17	1.13

Note: See the notes for Table 5.

图 7.2: 预测性能比较 2