Qlib：面向人工智能的量化投资平台

Microsoft Research
作者：Xiao Yang、Weiqing Liu、Dong Zhou、Jiang Bian、Tie-Yan Liu
联系邮箱：Xiao.Yang、Weiqing.Liu、Zhou.Dong、Jiang.Bian、Tie-Yan.Liug@microsoft.com

量化投资旨在在一段连续的交易期内，通过一组金融工具最大化回报并最小化风险。近年来，受人工智能技术在量化投资领域中的快速发展和巨大潜力的启发，越来越多的人开始采用基于人工智能的工作流程进行量化研究和实际投资。在丰富量化投资方法的同时，人工智能技术也给量化投资系统带来了新的挑战。特别是，针对量化投资的新学习范式需要升级基础设施以适应改进后的工作流程；此外，人工智能技术的数据驱动特性确实需要更强大性能的基础设施；另外，在金融场景中应用人工智能技术解决不同任务存在一些独特的挑战。为了应对这些挑战，弥合人工智能技术与量化投资之间的差距，我们设计和开发了 Qlib，旨在实现人工智能技术在量化投资中的潜力，赋予研究力量，并创造价值。

1 引言

量化投资作为研究领域中备受关注的一个方向，吸引了许多来自学术界和金融行业的杰出人才。在过去几十年中，通过不断优化量化方法论，专业投资者社区总结出了一个成熟但不完美的量化研究工作流程。最近，新兴的人工智能技术在这个研究领域掀起了一股新的趋势。随着对探索人工智能在量化投资中巨大潜力的关注不断增加，量化研究人员已经广泛采用人工智能技术进行实际投资。

虽然人工智能技术丰富了量化投资的方法论，但也对量化投资系统提出了多重挑战。首先，由于人工智能技术的灵活性，量化投资工作流程正在经历一场技术革命，这趋向于需要新的支持性基础设施。例如，传统的量化投资通常将整个工作流程分为几个子任务，包括股票趋势预测、组合优化等，而人工智能技术使得建立一个直接生成最终投资组合的端到端解决方案成为可能。为了支持这样的端到端解决方案，有必要升级当前基础设施，因为它具有数据驱动的特性。

同时，人工智能技术在一些新场景中面临独特的问题，这需要金融领域的丰富领域知识和数据科学的丰富经验。在没有任何领域适应的情况下将解决方案应用于量化研究任务很少有效。这种情况导致迫切需要一个平台来容纳在人工智能时代中这种现代化的量化研究工作流程，并为人工智能技术在金融场景中的应用提供指导。

因此，我们提出了一个名为 Qlib 的面向人工智能的量化投资平台。它旨在协助探索人工智能技术在量化投资中的巨大潜力的研究工作，并赋予量化研究人员在基于人工智能的量化投资中创造更重要价值的能力。具体而言，Qlib 的面向人工智能的框架旨在容纳基于人工智能的解决方案。此外，它提供了专为量化投资场景设计的高性能基础设施，使得许多

人工智能研究课题成为可能。此外，Qlib 集成了一批针对量化投资场景中的机器学习而设计的工具，以帮助用户充分利用人工智能技术。

最后，我们通过比较量化投资中一个典型任务的几种解决方案，展示了 Qlib 基础设施的一些应用案例并对其性能进行了评估。结果显示，专为量化投资定制的 Qlib 基础设施在这个任务上表现优于大多数现有解决方案。

2 背景和相关工作

在本节中，我们首先展示了现代量化研究人员在应用人工智能技术时面临的主要实际问题，这也是 Qlib 诞生的动因。然后，我们将简要介绍相关工作。 2.1 实际问题 量化研究工作流程的革命 在传统的投资研究工作流程中，研究人员通常基于几个因素（因子类似于机器学习中的特征）和基础金融数据，通过线性模型[Petkova, 2006]或手动设计的规则[Murphy, 1999]来开发交易信号。然后，他们会遵循一个交易策略（通常是 Barra[Sheikh, 1996]）来生成目标投资组合。最后，研究人员会通过回测函数评估交易信号和投资组合。 随着人工智能技术的崛起，它引发了传统量化投资的技术革命。传统的量化研究工作流程对于灵活的技术来说太过原始。为了更直观地展示差异，我们将展示一个基于人工智能技术的典型现代研究工作流程。它以一个具有大量特征的数据集为起点（通常超过几百维度）。手动设计如此多的特征需要大量时间。通常会利用机器学习算法自动生成这些特征[Potvin 等，2004; Neely 等，1997; Allen 和 Karjalainen，1999; Kakushadze，2016]。构建数据集的另一种选择是生成数据[Feng 等，2019]。基于多样的数据集，研究人员已经提供了数百种机器学习方法来挖掘交易信号[Sezer 等，2019]。研究人员可以基于这些交易信号生成目标投资组合。但是，这并不是唯一的工作流程选择。与将任务分成几个阶段不同，强化学习（RL）提供了一个从数据到最终交易行动的端到端解决方案[Deng 等，2016]。RL 通过与环境（金融场景中的交易模拟器）交互来优化交易策略。RL 需要一个响应灵敏的模拟器，而不是传统研究工作流程中的回测函数。此外，大多数人工智能算法具有复杂的超参数，需要进行精心调整。 人工智能技术如此灵活，已经超出了为传统方法论设计的现有工具的范畴。从头开始构建基于人工智能技术的研究工作流程需要很长时间。 对基础设施的高性能要求 随着人工智能技术的出现，对基础设施的要求发生了变化。这种数据驱动的方法可以利用大量数据。在高频交易场景中，数据量可能达到 TB 数量级。此外，通常会从基本的价格和成交量数据中衍生出成千上万个新特征（例如 Alpha101 [Kakushadze, 2016]），这些数据总共只有五个维度。一些研究人员甚至尝试通过搜索表达式[Allen 和 Karjalainen，1999; Neely 等，1997; Potvin 等，2004]来创建新的因子或特征。这种繁重的数据处理工作使得研究人员负担过重，甚至使一些研究无法顺利进行。

2.2 高性能基础设施需求

随着人工智能技术的兴起，对基础设施的需求发生了变化。这种数据驱动的方法可以利用大量数据。在高频交易场景中，数据量可能达到 TB 数量级。此外，通常会从基本的价格

和成交量数据中衍生出成千上万个新特征（例如 Alpha101 [Kakushadze, 2016]），总共只有五个维度。一些研究人员甚至尝试通过搜索表达式[Allen 和 Karjalainen，1999; Neely 等，1997; Potvin 等，2004]来创建新的因子或特征。这种繁重的数据处理工作使得研究人员负担过重，甚至使一些研究无法顺利进行。这些情况对基础设施提出了更严格的性能要求。

应用机器学习解决方案的障碍

金融数据和任务具有其独特性和挑战性。

在没有任何适应性的情况下将机器学习解决方案应用于量化研究任务很少成功。由于金融数据中信噪比（Signal to Noise Ratio）极低，很难在金融市场上建立一个成功的数据驱动策略。大多数机器学习算法是数据驱动的，必须应对这些困难。如果不仔细处理细节，机器学习模型很难达到令人满意的性能。即使是一个小错误也会导致模型过度拟合噪声而无法学习有效的模式。正确处理细节需要对金融行业具有很多领域知识。此外，典型的目标，如年化回报率，通常是不可微分的，这使得直接为机器学习方法训练模型变得困难。定义一个合理的任务，具有适当的监督目标，对于建模金融数据非常重要。这些障碍使得许多没有太多金融行业领域知识的数据科学家望而却步。

构建机器学习应用的另一个必要步骤是超参数优化。不同的机器学习算法具有不同的超参数搜索空间，每个空间都有多个维度，具有不同的含义和优先级。一些量化研究人员来自传统金融行业，对机器学习了解不多。这种巨大的学习成本使许多用户望而却步。2.2 相关工作在金融行业中，随着越来越多的投资者追随一个投资策略，该策略的盈利能力将会降低。因此，金融从业者，尤其是量化研究人员，很少愿意分享自己的算法和工具。

OLPS [Li 等，2016] 是第一个开源的投资组合选择工具箱。它由一系列基于机器学习算法的经典策略组成，作为基准和工具包，以便开发新的学习方法。该工具箱仅支持 Matlab 和 Octave，不兼容当前科学主流语言 Python，因此对于现代机器学习算法不太友好。它的框架相当简单，而基于 AI 技术的现代量化研究工作流程要复杂得多。近年来出现了其他量化工具。QuantLib [Firth, 2004] 只关注现代量化研究工作流程的一部分。QUANTAXIS 2 更注重 IT 基础设施，而不是研究工作流程。Quantopian 发布了一系列开源工具：1）Alphalens：用于预测性（alpha）股票因子性能分析的 Python 库；2）Zipline：用于事件驱动的回测系统；3）Pyfolio：用于金融投资组合的性能和风险分析的 Python 库。所有这些工具都只关注交易信号分析或投资组合分析。总的来说，Qlib 是第一个开源平台，适应了在人工智能时代的现代量化研究人员的工作流程。它旨在赋予每个量化研究人员发掘人工智能技术在量化投资中的巨大潜力。

3 AI 导向的量化投资平台

3.1 总体设计

在与具有多年金融市场实践经验的量化研究人员合作中，我们遇到了以上所有问题，并探索了各种解决方案。在当前形势的推动下，我们实现了 Qlib，将人工智能技术应用于量化投资。

AI 导向的框架 Qlib 基于现代研究工作流程以模块化的方式设计，以提供最大的灵活性来适应人工智能技术。量化研究人员可以扩展模块并构建工作流程以高效地尝试他们的想法。在每个模块中，Qlib 提供了几种默认的实现选择，在实际投资中表现良好。有了这些

现成的模块，量化研究人员可以专注于他们在特定模块中感兴趣的问题，而不会被其他琐碎的细节所分散注意力。除了代码外，计算和数据也可以在某些模块中共享，因此 Qlib 被设计为服务用户的平台，而不仅仅是一个工具箱。

高性能基础设施 数据处理的性能对于像 AI 技术这样的数据驱动方法非常重要。作为一个 AI 导向的平台，Qlib 提供了高性能的数据基础设施。Qlib 提供了一个时间序列的平面文件数据库。这样的数据库专门用于金融数据的科学计算。在量化投资研究中的一些典型数据处理任务上，它比当前流行的通用数据库和时间序列数据库有着更高的性能。此外，数据库提供了一个表达式引擎，可以加速因子/特征的实现和计算，使依赖于表达式计算的研究课题成为可能。

机器学习指导 Qlib 已经集成了一些典型的量化投资数据集，其中典型的机器学习算法可以成功地学习具有泛化能力的模式。Qlib 为机器学习用户提供了一些基本指导，并集成了一些合理的任务，包括合理的特征空间和目标标签。还提供了一些典型的超参数优化工具。在指导和合理的设置下，机器学习模型可以学习具有更好泛化能力的模式，而不仅仅是过度拟合噪声。

3.2 AI 导向的框架

图 1 显示了 Qlib 的整体框架。该框架旨在：1）适应现代 AI 技术；2）

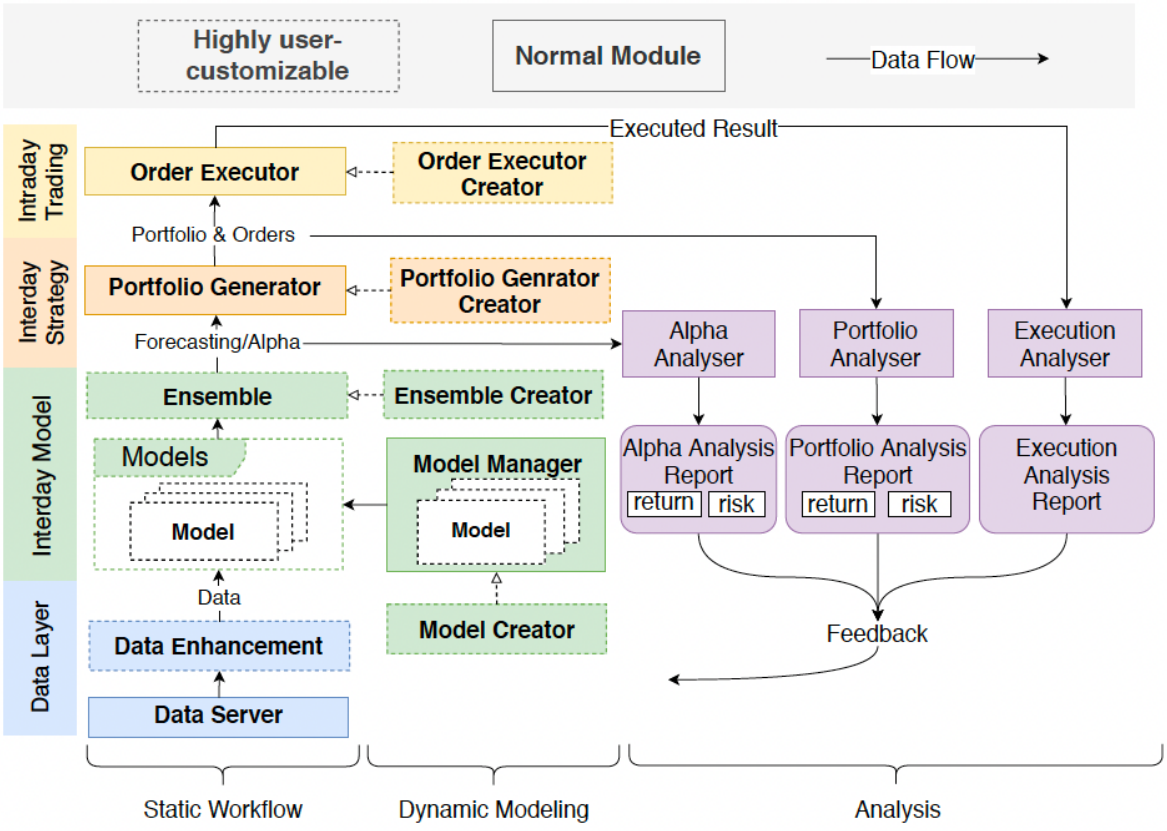


Figure 1: modules and a typical workflow built with Qlib

帮助量化研究人员以最小的努力构建完整的研究工作流程，并在不受其他部分干扰的同时保留最大的灵活性来探索他们感兴趣的问题。

这样的目标从系统设计的角度导致了模块化设计。系统基于现代实际研究工作流程被分为多个独立的模块。无论是传统的量化投资研究还是基于人工智能的研究，大多数都可以看作是一个或多个模块接口的实现。Qlib 为每个模块提供了几种在实际投资中表现良好的典型实现。此外，模块为研究人员提供了灵活性，可以覆盖现有方法以探索新思路。通过这样的框架，研究人员可以以最小的成本尝试新想法，并与其他模块一起测试整体性能。

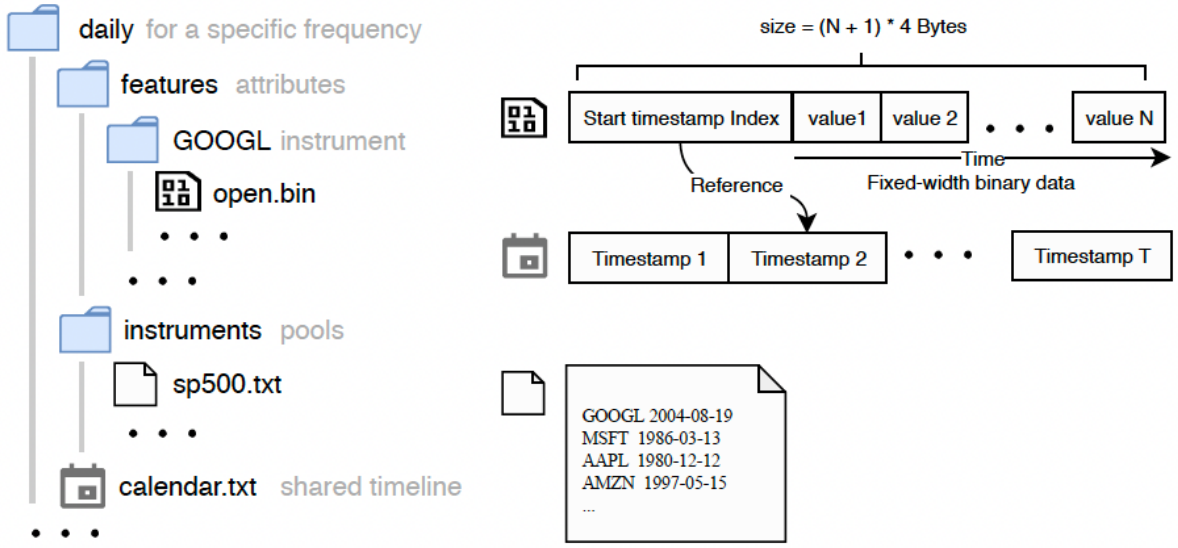
Qlib 的模块在图 1 中列出，并按照典型的工作流程连接在一起。每个模块对应量化投资中的一个典型子任务。模块中的一个实现可以被视为该任务的解决方案。我们将介绍每个模块，并给出一些相关的现有量化研究示例，以展示 Qlib 如何适应它们。

它从左下角的数据服务器模块开始，该模块提供了一个数据引擎来查询和处理原始数据。研究人员可以在数据增强模块中构建自己的数据集，通过探索和构建有效的因子/特征[Potvin et al., 2004; Neely et al., 1997; Allen and Karjalainen, 1999; Kakushadze, 2016]来构建更好的数据集已经是一种尝试。为训练生成数据集[Feng et al., 2019]是提供数据集解决方案的另一个研究方向。模型创建器模块基于数据集学习模型。近年来，众多研究人员已经探索了各种模型来从金融数据集中挖掘交易信号[Sezer et al., 2019]。此外，元学习[Vilalta and Drissi, 2002]尝试学习学习为模型创建器模块提供了一种新的学习范式。在现代研究工作流程中有很多方法可以对金融数据进行建模，模型管理系统已经成为一个必要的部分。模型管理模块旨在解决现代量化研究人员面临的这些问题。通过多样化的模型，集成学习是提高机器学习模型性能和鲁棒性的一种有效方式，在金融领域广泛应用[Qiu 等，2014；Yang 等，2017；Zhao 等，2017]。模型集成模块支持集成学习。

投资组合生成模块旨在根据模型产生的交易信号生成投资组合，这被称为投资组合管理[Qian 等，2007]。Barra [Sheikh，1996] 提供了这一任务的最流行解决方案。通过目标投资组合，我们提供了一个高保真度的交易模拟器，即订单执行模块，用于检查策略的绩效，并提供分析模块以自动分析交易信号、投资组合和执行结果。订单执行模块被设计为响应式模拟器，而不是回测功能，可以为一些需要分析模块产生的环境反馈的学习范式（例如，强化学习）提供基础设施。

量化投资中的数据以时间序列格式存在，并随时间更新。样本内数据集的大小会随时间增加。利用新数据的典型做法是定期更新我们的模型[Wang 等，2019b]。除了更好地利用不断增长的样本内数据外，动态更新模型[Yang 等，2019]和交易策略[Wang 等，2019a]将进一步提高性能，因为股票市场具有动态性[Adam 等，2016]。因此，在静态工作流程中使用一组静态模型和交易策略显然不是最佳解决方案。动态更新模型和策略是量化投资中的

一个重要研究方向。动态建模模块提供接口和基础设施，以适应这些解决方案。



3.3 高性能基础设施

金融数据

本节中，我们将总结量化研究中的数据要求。在量化研究中，最常用的数据格式遵循以下格式：

$$BasicData_T = \{x_{i,t,a}\}, i \in Inst, t \in Time, a \in Attr$$

其中，xi;t;a 是基本类型的值（例如浮点数、整数），Inst 表示金融工具集合（例如股票、期权等），Time 表示时间戳集合（例如股票市场的交易日），Attr 表示金融工具可能的属性集合（例如开盘价、成交量、市值），T 表示数据的最新时间戳（例如最新交易日期）。xi;t;a 表示时间 t 时刻工具 i 的属性 a 的值。

此外，工具池（instruments pools）是指随时间变化的金融工具集合，它们是必要的信息。

$$Pool_T = \{pool_t\}, t \in Time, pool_t \subseteq Inst$$

标普 500 指数（S&P 500 Index）是一个典型的工具池示例。

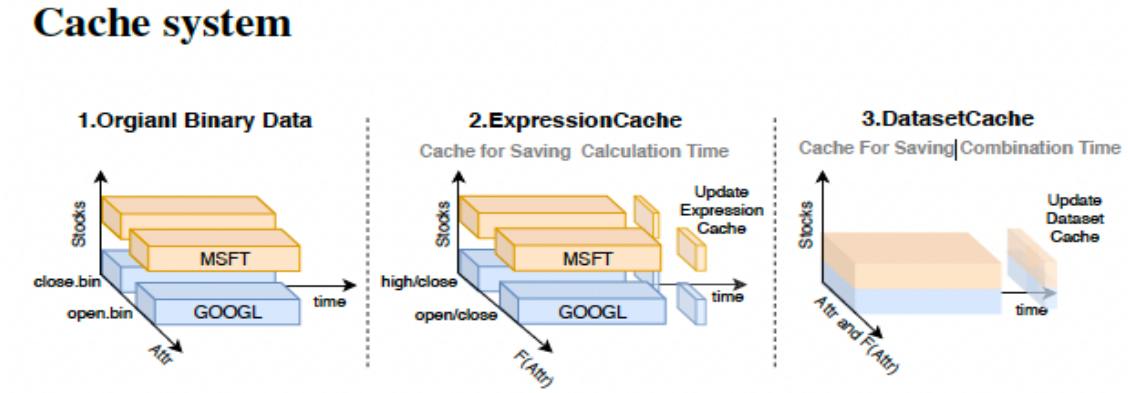
数据更新是一个重要特性。现有的历史数据不会随时间变化。只有对新数据进行附加操作，才能实现数据的更新。

$$\begin{aligned} BasicData_T &= OldBasicData_T \cup \{x_{i,t,a_{new}}\} \\ BasicData_{T+1} &= BasicData_T \cup \{x_{i,T+1,a}\} \\ Pool_{T+1} &= Pool_T \cup \{pool_{t+1}\} \end{aligned}$$

$$\begin{aligned} Data_{Query} = \{x_{i,t,a} | & i_t \in pool_t, pool_t \in Pool_{query} \\ & a \in Attr_{query}, time_{start} \le t \le time_{end}\} \end{aligned}$$

在量化研究中，最常用的数据格式遵循以下更新操作形式：
BasicDataT =OldBasicDataT [fxi;t;anewg
BasicDataT+1 =BasicDataT [fxi;T+1;ag
PoolT+1 =PoolT [fpoolt+1g
用户查询可以形式化表示为：
DataQuery = fxi;t;ajit 2 poolt; poolt 2 Poolquery
a 2 Attrquery; timestart　t　timeendg
它表示在特定池中的特定时间范围内查询工具的某些属性的数据。
这些需求非常简单。许多现成的开源解决方案支持此类操作。我们将它们分为三类，并列出每个类别中流行的实现。
通用数据库：MySQL[MySQL, 2001]，MongoDB[Chodorow, 2013]
时序数据库：InfluxDB [Naqvi et al., 2017]
科学计算数据文件：使用 numpy[Oliphant, 2006]数组或 pandas[McKinney, 2011]数据框组织的数据
通用数据库支持多种格式和结构的数据。此外，它们提供许多复杂的机制，如索引、事务、实体关系模型等。大多数通用数据库为特定任务增加了繁重的依赖和不必要的复杂性，而并没有解决特定场景中的关键问题。时序数据库优化了时序数据的数据结构和查询方式。但是它们仍然不适用于量化研究，量化研究中的数据通常以紧凑的基于数组的格式存储，以便进行科学计算以利用硬件加速。如果数据从磁盘到客户端的过程中保持紧凑的基于数组的格式而不进行格式转换，将节省大量时间。然而，通用数据库和时序数据库都以不同的格式存储和传输数据，这对科学计算来说是低效的。
由于数据库的低效性，基于数组的数据在科学界变得流行起来。Numpy 数组和 pandas 数据框是科学计算中的主流实现，通常以 HDF5 或 pickle6 的形式存储在磁盘上。这种格式的数据具有轻量级的依赖关系，并且非常适合科学计算。然而，这种数据存储在单个文件中，很难进行更新或查询。
经过对上述存储解决方案的调查，我们发现没有一种完全适用于量化研究场景。因此，有必要为量化研究设计一种定制化的解决方案。
文件存储设计
图 2 展示了文件存储的设计。如图左侧所示，Qlib 将文件组织成树形结构。数据按照不同的特性分隔为文件夹和文件。
频率、工具和属性。所有属性的值都以紧凑的固定宽度格式存储在二进制数据中，以便能够进行字节索引。共享的时间轴单独存储在名为"calendar.txt"的文件中。属性值的数据文件将其前 4 个字节设置为时间轴的索引值，以指示数据系列的起始时间戳。通过起始时间索引，Qlib 可以在时间维度上对齐所有的值。
数据以紧凑的格式存储，便于合并为科学计算的数组。虽然它在科学计算中具有基于数组的数据的高性能，但它也满足量化投资场景中的数据更新要求。所有的数据都按时间顺序排列。新数据可以通过追加的方式进行更新，效率非常高。添加和删除属性或工具非常直观和高效，因为它们存储在单独的文件中。这种设计非常轻量级。在没有数据库的开销下，Qlib 实现了高性能。
表达式引擎

基于基本数据开发新的因子/特征是一项常见任务。这样的任务占据了许多量化研究人员时间的很大比例。既要通过代码实现这些因子，计算过程又很耗时。因此，Qlib 提供了一个表达式引擎来最大程度地减少这类任务的工作量。
实际上，因子/特征的本质是将基本数据转化为目标值的函数。该函数可以分解为一系列表达式的组合。表达式引擎就是基于这个思想设计的。借助这个表达式引擎，量化研究人员可以通过编写表达式而不是复杂的代码来实现新的因子/特征。例如，布林带技术指标[Bollinger, 2002]是一个广泛使用的技术因子，其上界可以通过一个简单的表达式"（MEAN（$close，N）+2*STD（$close，N）-$close）/ MEAN（$close，N）"来实现。这样的实现简单、可读、可重用和易于维护。用户可以仅仅通过一系列简单的表达式来构建数据集。搜索表达式以构建有效的交易信号是一个典型的研究课题，许多研究人员已经进行了探索[Allen 和 Karjalainen，1999；Neely 等，1997；Potvin 等，2004]。



为了避免重复计算，Qlib 拥有内置的缓存系统，包括内存缓存和磁盘缓存。

内存缓存：当 Qlib 使用表达式引擎计算因子/特征时，它将表达式解析为语法树。所有节点的计算结果都将存储在内存中的 LRU（最近最少使用）缓存中，以避免重复计算相同的（子）表达式。

磁盘缓存：量化投资中数据处理的典型工作流程可以分为三个步骤：获取原始数据、计算表达式和将数据组合成科学计算所需的数组。计算表达式和数据组合非常耗时。如果能够缓存共享的中间数据，将能节省大量时间。在实际的数据处理任务中，许多中间结果可以共享。例如，不同的数据处理任务可以共享相同的表达式计算。因此，Qlib 设计了一个两级磁盘缓存机制，如图 3 所示。左侧是我们在第 3.3 节中描述的原始数据。第一级是表达式缓存，它将所有计算过的表达式保存在磁盘缓存中。表达式缓存的数据结构与原始数据相同。通过表达式缓存，相同的表达式只会计算一次。在表达式缓存之后是数据集缓存，它存储了组合后的数据以节省组合时间。两个级别的缓存数据都按时间排列，并且在时间维度上可进行索引，因此即使查询时间发生变化，磁盘缓存仍然可以共享。此外，由于数据按时间排列，Qlib 支持通过追加新数据进行数据更新。使用这样的机制，数据的维护更加容易。

3.4 机器学习指导

正如我们在第 2 节中讨论的那样，对于机器学习算法的指导非常重要。Qlib 为机器学习算法提供了典型的数据集。在 Qlib 中可以找到一些典型的任务设置，如数据预处理、学习目标等。研究人员无需从零开始探索一切。这些指导为研究人员提供了大量的领域知识，帮助他们开始在这个研究领域中的探索之旅。

对于大多数机器学习算法来说，超参数优化是实现更好泛化性能的必要步骤。尽管它非常重要，但需要花费很多精力和时间。为了减轻研究人员的负担，Qlib 提供了一些典型的超参数优化工具。通过这些工具的指导和合理的设置，机器学习模型可以学习到更好的泛化能力，而不仅仅过拟合噪声。

Qlib 还提供了一些常用的机器学习算法的默认实现，研究人员可以根据自己的需求选择适合的算法并进行相应的调整和优化。这样的指导和实现选择可以加速研究人员的实验和验证过程，提高研究效率。

	HDF5	MySQL	MongoDB	InfluxDB	Qlib -E -D	Qlib +E -D	Qlib +E +D
Storage(MB)	**287**	1,332	911	**394**	**303**	802	1,000
Load Data(s)	**0.80±0.22**	182.5±4.2	70.3±4.9	186.5±1.5	**0.95±0.05**	4.9±0.07	7.4±0.3
Compute Expr.(s)	179.8±4.4				137.7±7.6	**35.3±2.3**	-
Convert Index(s)	-				3.6±0.1		-
Filter by Pool(s)	3.39 ±0.24						-
Combine data(s)	1.19±0.30						-
Total (1CPU) (s)	184.4±3.7	365.3±7.5	253.6±6.7	368.2±3.6	147.0±8.8	47.6±1.0	**7.4±0.3**
Total(64CPUs) (s)	-				8.8±0.6	**4.2±0.2**	-

Table 1: Performance comparison of different storage solutions

因此，Qlib 提供了一个超参数调优引擎（HTE），使这样的任务变得更加容易。HTE 提供了一个接口，用于定义超参数搜索空间，并自动搜索最佳超参数。在典型的金融时间序列数据建模任务中，新数据按时间顺序到达。为了利用新数据，模型必须定期在新数据上重新训练。新的最佳超参数会发生变化，但通常与之前的最佳超参数接近。HTE 提供了一个专门用于金融任务的超参数优化机制。它为超参数搜索空间生成一个新的分布，以更小的尝试次数获得更好的结果。搜索的分布可以形式化为：

$$p_{new}(x) = \frac{p_{prior}(x)\varphi_{\theta_{prev},\sigma^2}(x)}{\mathbb{E}_{x\sim p_{prior}}[\varphi_{\theta_{prev},\sigma^2}(x)]}$$

其中 pprior 是原始的超参数搜索空间；' prev; 2 (x) 为 N(prev; 2)的分布， prev 是上一次模型训练的最佳超参数。超参数搜索空间的定义域保持不变，但在 prev 附近的概率密度增加。

4 用例与性能评估

4.1 用例

Qlib 提供了一个配置驱动的流水线引擎（CDPE），帮助研究人员更轻松地构建图中显示的整个研究工作流。用户可以使用简单的配置文件定义工作流，类似于列表??（一些细节被替换为“...”）。这样的界面不是强制性的，我们为用户留下了最大的灵活性，可以通过代码像构建模块一样构建量化研究工作流。

4.2 性能评估

数据处理的性能对于像 AI 技术这样的数据驱动方法来说非常重要。作为一个面向 AI 的平台，Qlib 提供了数据存储和数据处理的解决方案。为了展示 Qlib 的性能，我们将 Qlib 与第 3.3 节中讨论的几种其他解决方案进行比较，包括 HDF5、MySQL、MongoDB、InfluxDb 和 Qlib。其中，Qlib + E-D 表示启用了表达式缓存但禁用了数据集缓存，依此类推。

```
model:
  class:   "qlib.model.GBDTModel"
  args:    ...
data:
  class:   "qlib.dataset.Alpha360"
learner:
  class:   "qlib.trainer.NormalTrainer"
  args:    ...
portfolio_manager:
  class:   "qlib.portfolio.TopkStrategy"
executor:
  account: ...
```

图 4：CDPE 的配置示例

任务是从股票市场的基本 OHLCV7 日线数据创建一个数据集，涉及数据查询和处理。最终数据集由从 OHLCV 数据衍生出的 14 个因子/特征组成（例如“Std($close, 5)/$close”）。数据的时间范围从 2007 年 1 月 1 日到 2020 年 1 月 1 日。股票池每天包含 800 只股票，每天都会发生变化。

除了比较每个解决方案的总时间外，我们将任务细分为以下步骤以进行更详细的分析：
- 加载数据：将 OHCLV 数据或缓存加载到 RAM 中，以进行科学计算的基于数组的格式。
- 计算衍生因子/特征：计算衍生因子/特征。
- 转换索引：仅适用于 Qlib。因为 Qlib 不存储原始数据的索引（即时间戳，股票 ID），所以需要设置数据索引。
- 过滤数据：按照特定的股票池筛选股票数据。例如，标准普尔 500 指数每天包含 500 只股票，但总共涉及 1000 多只股票。不包含在标准普尔 500 指数中的股票数据应该被过滤掉，尽管它们曾经在标准普尔 500 指数中。在加载数据时无法进行过滤，因为一些衍生特征依赖于历史的 OHLCV 数据。
- 合并数据：将不同股票的所有数据合并为一个基于数组的数据。

如表 1 所示，可以看到 Qlib 的紧凑存储方式在大小和加载速度方面与专用科学计算存储方式相当。

HDF5 数据文件。数据库在加载数据时花费了太多时间。经过研究底层实现，我们发现数据在通用数据库和时间序列数据库解决方案中经过了太多层次的接口和不必要的格式转换。这些开销极大地减慢了数据加载过程。由于 Qlib 的内存缓存，Qlib -E -D 可以节省约 24%的 Compute Expr.的时间。此外，Qlib 提供了表达式缓存和数据集缓存机制。如果启用

了表达式缓存，Qlib +E -D 可以节省 Compute Expr.的 80.4%时间，假设没有错过表达式缓存。将因子/特征组合成每只股票的一个基于数组的数据是 Qlib +E -D 的主要时间消耗，它包含在 Compute Expr.步骤中。除了计算成本，最耗时的步骤是数据合并。数据集缓存旨在减少这些开销。如表 1 中的 Qlib +E +D 列所示，时间成本进一步降低。此外，Qlib 可以利用多个 CPU 核心加速计算。从表 1 的最后一行可以看出，使用多个 CPU 的 Qlib 的时间成本显著降低。由于它只读取现有缓存并几乎不进行计算，Qlib +E +D 无法进一步加速。

4.3 关于 Qlib 的更多信息

Qlib 是一个持续开发的开源平台。更详细的文档可以在其 GitHub 存储库 8 中找到。在线存储库中还可以找到许多未在本文中详细介绍的功能（例如具有客户端-服务器架构的数据服务、分析系统、云上的自动部署）。欢迎您的贡献。

5 结论

在本文中，我们介绍了 AI 时代现代量化研究者面临的实际问题。基于这些实际问题，我们设计和实现了 Qlib，旨在使每个量化研究者能够在量化投资中充分发挥 AI 技术的巨大潜力。

引用

[Adam et al., 2016] Klaus Adam, Albert Marcet, and Juan Pablo Nicolini. "股市波动性与学习", 2016.

[Allen and Karjalainen, 1999] Franklin Allen 和 Risto Karjalainen. "使用遗传算法寻找技术交易规则". 《金融经济学杂志》, 51(2):245–271, 1999.

[Bollinger, 2002] John Bollinger. "Bollinger 论 Bollinger 带". McGraw Hill Professional, 2002.

[Chodorow, 2013] Kristina Chodorow. "MongoDB: 确定性指南：强大且可扩展的数据存储". O' Reilly Media, Inc., 2013.

[Deng et al., 2016] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, 和 Qionghai Dai. "用于金融信号表示和交易的深度直接强化学习." IEEE 神经网络与学习系统交易, 28(3):653–664, 2016.

[Feng et al., 2019] Fuli Feng, Huimin Chen, Xiangnan He, Ji Ding, Maosong Sun, 和 Tat-Seng Chua. "通过对抗性训练提高股票价格预测." 在第 28 届国际人工智能联合会议论文集中发表, 页码 5843–5849. AAAI 出版社, 2019.

[Firth, 2004] N Firth. "为什么使用 Quantlib." 可在以下网址获得: http://www.quantlib.co.uk/publications/quantlib.pdf, 2004.

[Kakushadze, 2016] Zura Kakushadze. "101 个公式化的阿尔法." Wilmott, 2016(84):72–81, 2016.

[Li et al., 2016] Bin Li, Doyen Sahoo, 和 Steven CH Hoi. "Olps：在线投资组合选择工具箱." 机器学习研究杂志, 17(1):1242–1246, 2016.

[McKinney, 2011] Wes McKinney. "Pandas：数据分析和统计的基础 Python 库." Python 高性能与科学计算, 14, 2011.

[Murphy, 1999] John J Murphy. "金融市场技术分析:交易方法和应用的综合指南." 企鹅出版社, 1999.

[MySQL, 2001] AB MySQL. "MySQL," 2001.

[Naqvi et al., 2017] Syeda Noor Zehra Naqvi, Sofia Yfantidou, 和 Esteban Zim´anyi. "时间序列数据库和 InfluxDB." Studienarbeit, Universit´e Libre de Bruxelles, 2017.

[Neely et al., 1997] Christopher Neely, Paul Weller, 和 Rob Dittmar. "外汇市场技术分析是否有利可图？一种基于遗传编程的方法." 金融与数量分析杂志, 32(4):405–426, 1997.

[Oliphant, 2006] Travis E Oliphant. "NumPy 指南，卷 1." Trelgol Publishing USA, 2006.

[Petkova, 2006] Ralitsa Petkova. "Fama-French 因子是否代表预测变量的创新？" 金融学杂志, 61(2):581–612, 2006.

[Potvin et al., 2004] Jean-Yves Potvin, Patrick Soriano, 和 Maxime Vall´ee. "使用遗传编程在股票市场上生成交易规则." 计算与运营研究, 31(7):1033–1047, 2004.

[Qian et al., 2007] Edward E Qian, Ronald H Hua, 和 Eric H Sorensen. "定量股票组合管理:现代技术和应用." CRC 出版社, 2007.

[Qiu et al., 2014] Xueheng Qiu, Le Zhang, Ye Ren, Ponnuthurai N Suganthan, 和 Gehan Amaratunga. "集成深度学习用于回归和时间序列预测." 在 2014 年 IEEE 计算智能集成学习(CIEL)研讨会上, 页码 1–6. IEEE, 2014.

[Sezer et al., 2019] Omer Berat Sezer, Mehmet Ugur Gudelek, 和 Ahmet Murat Ozbayoglu. "基于深度学习的金融时间序列预测: 2005-2019 的系统文献综述." arXiv 预印本 arXiv:1911.13288, 2019.

[Sheikh, 1996] Aamir Sheikh. "Barra 的风险模型." Barra 研究见解, 页码 1–24, 1996.

[Vilalta and Drissi, 2002] Ricardo Vilalta 和 Youssef Drissi. "元学习的透视观点和调查." 人工智能综述, 18(2):77–95, 2002.

[Wang et al., 2019a] Lewen Wang, Weiqing Liu, Xiao Yang, 和 Jiang Bian. "保守还是激进？具有置信度感知的动态投资组合构建." 在 2019 年 IEEE 全球信号与信息处理会议(GlobalSIP)上发表, 页码 1–5. IEEE, 2019.

[Wang et al., 2019b] Shouxiang Wang, Xuan Wang, Shaomin Wang, 和 Dan Wang. "基于注意机制和滚动更新的双向长短期记忆方法用于短期负荷预测." 国际电力与能源系统杂志, 109:470–479, 2019.

[Yang et al., 2017] Bing Yang, Zi-Jia Gong, 和 Wenqi Yang. "使用深度神经网络集成进行股票市场指数预测." 在 2017 年第 36 届中国控制会议(CCC)上发表, 页码 3882–3887. IEEE, 2017.

[Yang et al., 2019] Xiao Yang, Weiqing Liu, Lewen Wang, Cheng Qu, 和 Jiang Bian. "基于注意力机制的多重投资策略组合的分治框架." 在 2019 年 IEEE 全球信号与信息处理会议(GlobalSIP)上发表, 页码 1–5. IEEE, 2019.

[Zhao et al., 2017] Yang Zhao, Jianping Li, 和 Lean Yu. "用于原油价格预测的深度学习集成方法." 能源经济学, 66:9–16, 2017.