Qlib：面向 AI 的量化投资平台

Qlib: An AI-oriented Quantitative Investment Platform

Xiao Yang, **Weiqing Liu**, **Dong Zhou**, **Jiang Bian** and **Tie-Yan Liu**
微软研究院（Microsoft Research）
{Xiao.Yang, Weiqing.Liu, Zhou.Dong, Jiang.Bian, Tie-Yan.Liu}@microsoft.com

摘要

量化投资要在一段连续的交易周期内，围绕一组金融标的，把收益做到最大、把风险压到最低。近年来，AI 技术发展迅速，在量化投资中孕育重大创新的潜力也颇被看好；受此驱动，量化研究与实盘投资越来越多地采用 AI 驱动的工作流。AI 技术在丰富量化投资方法论的同时，也给量化投资系统带来新的挑战。尤其是，量化投资的新学习范式要求基础设施随之升级，以承载改造后的工作流；再者，AI 技术以数据驱动为本，这本身就意味着基础设施需要更强的性能；另外，把 AI 技术用到金融场景的各类任务上，还会遇到一些独有的难题。为应对这些挑战、填平 AI 技术与量化投资之间的鸿沟，我们设计并实现了 Qlib，用于释放 AI 技术在量化投资中的潜力、支撑相关研究并创造价值。

1 引言

量化投资是当前最热门的研究方向之一，吸引了大批学界与业界的优秀人才。过去几十年里，专业投资者群体持续打磨量化方法论，逐步沉淀出一套成熟却仍不完美的量化研究工作流。近来，新兴的 AI 技术在这一领域掀起新的风潮。随着 AI 在量化投资中的潜力越来越受关注，量化研究员已把 AI 技术广泛用于实盘投资。

AI 技术在丰富量化投资方法论的同时，也从多个方面给量化投资系统提出了新挑战。一方面，AI 技术足够灵活，由此引发的工作流变革往往需要新的基础设施来支撑。举例来说，传统量化投资通常把整条工作流拆成若干子任务，如股票趋势预测、组合优化等；AI 技术则让端到端方案成为可能，直接产出最终的投资组合。这类方案以数据驱动，要支撑它，现有基础设施必须升级。

另一方面，AI 技术还要应对一些新场景下的特有问题，这既需要充足的金融领域知识，也需要丰富的数据科学经验。不做任何领域适配就把现成方案套到量化研究任务上，几乎行不通。如此一来，业界急需这样一个平台：既能承载 AI 时代的现代量化研究工作流，又能为 AI 技术在金融场景中的落地给出指引。

为此，我们提出面向 AI 的量化投资平台 Qlib[1]。它既服务于探索 AI 技术在量化投资中巨大潜力的研究工作，也帮助量化研究员在 AI 驱动的投资实践中创造更大价值。具体来说，Qlib 的框架面向 AI 设计，可以容纳各类基于 AI 的方案。它还为量化投资场景提供高性能基础设施，让许多 AI 研究课题得以开展。平台内另集成了一批工具，专为量化投资场景下的机器学习而设计，帮助用户把 AI 技术的能力发挥充分。

最后，我们给出若干用例，并围绕一项典型的量化投资任务对比多种方案，以此评测 Qlib 基础设施的性能。结果表明，Qlib 为量化投资定制的基础设施在该任务上胜过多数现有方案。

2 背景与相关工作

本节先说明量化研究员把 AI 技术用于量化投资时遇到的几类主要实际问题，Qlib 正是为解决它们而生。之后再简要介绍相关工作。

2.1 实践中的问题

量化研究工作流的变革

传统投资研究工作流里，研究员拿基础金融数据和为数不多的几个因子作输入（因子相当于机器学习里的特征），

本文为 AI 辅助翻译版本，仅供学习交流；引用请以英文原文为准：arXiv:2009.11189。

[1]代码见 https://github.com/microsoft/qlib

用线性模型[Petkova, 2006]或人工设计的规则[Murphy, 1999]产出交易信号。随后套一层交易策略（典型的是Barra[Sheikh, 1996]）生成目标组合。最后用回测函数评估交易信号与组合。

AI 技术兴起，给传统量化投资掀起一场技术变革。这类技术很灵活，而传统量化研究工作流过于原始，承载不了。为了把差别摆得更直观，下面给出一条建立在 AI 技术之上的典型现代研究工作流。起点是特征众多的数据集，维度通常上百。人工设计这么多特征太耗时间，常见做法是让机器学习算法自动生成[Potvin 等, 2004; Neely 等, 1997; Allen and Karjalainen, 1999; Kakushadze, 2016]。构造数据集还有另一条路，就是直接生成数据[Feng 等, 2019]。数据集形态各异，研究者已在其上提出数百种机器学习方法来挖掘交易信号[Sezer 等, 2019]。拿到交易信号，即可据此生成目标组合。但这条路并非唯一选择。强化学习（reinforcement learning, RL）不把任务切成若干阶段，而是从数据直通最终交易动作，给出端到端方案[Deng 等, 2016]。RL 靠与环境交互来优化交易策略，金融场景下的环境就是交易模拟器。它要的是一个响应迅速的模拟器，而不是传统研究工作流里那个回测函数。多数 AI 算法还带着一批复杂的超参数，需要仔细调优。

AI 技术如此灵活，早已越出现有工具的能力边界，那些工具原本是为传统方法论设计的。而从零搭一条基于 AI 技术的研究工作流，又相当费时间。

基础设施的高性能要求

AI 技术出现之后，对基础设施的要求也随之改变。数据驱动的方法吃得下海量数据，高频交易场景中数据量可达 TB 量级。基础量价数据总共才五个维度，从中衍生出上千个新特征却是常事，Alpha101[Kakushadze, 2016]就是一例。还有研究者尝试用搜索表达式的办法构造新因子与新特征[Allen and Karjalainen, 1999; Neely 等, 1997; Potvin 等, 2004]。数据处理的负担如此之重，研究员疲于应付，某些研究课题甚至因此无从做起。这种局面对基础设施提出了更严苛的性能要求。

应用机器学习方案的障碍

金融数据与金融任务自有其特殊之处，也各带难点。把机器学习方案原样搬到量化研究任务上，不作任何适配，很少能奏效。金融数据的信噪比（Signal to Noise Ratio, SNR）极低，要在金融市场上做出一套真正管用的数据驱动策略非常难。多数机器学习算法都是数据驱动的，绕不开这一困难。细节处理得不够仔细，模型表现就很难令人满意。哪怕一处小失误，也足以让模型把噪声拟合进去，学不到有效模式。要把细节处理对，需要大量金融行业的领域知识。再者，年化收益率之类的典型目标往往不可导，机器学习方法难以照着它直接训练模型。所以，为金融数据建模时，把任务定得合理、把监督目标选得恰当，尤为要紧。这些门槛吓退了不少数据科学家，他们所缺的恰恰是金融行业的领域知识。

搭一套机器学习应用还有一步躲不掉：超参数优化。算法不同，超参数搜索空间也不同，每个空间又含多个维度，各维度的含义与优先级互不相同。有些量化研究员出身传统金融行业，对机器学习所知不多。学习成本这样高，许多人便无法把机器学习的价值用足。

2.2 相关工作

金融行业有一条规律：一套投资策略的跟随者越多，能赚的利润就越薄。所以金融从业者、尤其是量化研究员，从来不愿把自己的算法和工具拿出来共享。OLPS[Li 等, 2016]是第一个面向组合选择的开源工具箱。它收录了一批由机器学习算法驱动的经典策略充当基准，并配了一套工具包，方便开发新的学习方法。这套工具箱只支持 Matlab 和 Octave，与当下科学计算的主流语言 Python 不兼容，对现代机器学习算法很不友好。它的框架也相当简单，而基于 AI 技术的现代量化研究工作流要复杂得多。近年还出现了其他量化工具。QuantLib[Firth, 2004]只覆盖现代量化研究工作流的一部分。QUANTAXIS[2]更偏 IT 基础设施，而不是研究工作流。Quantopian 开源了一系列工具：1) Alphalens，用于分析 alpha 因子表现的 Python 库；2) Zipline，事件驱动的回测系统；3) Pyfolio，用于分析组合业绩与风险的 Python 库。它们都只盯着交易信号或投资组合的分析这一环。

Qlib 是 AI 时代第一个能承载量化研究员整套工作流的开源平台。它的目标是让每一位量化研究员都能把 AI 技术在量化投资中的巨大潜力兑现出来。

3 面向 AI 的量化投资平台

3.1 总体设计

我们与在金融市场有多年一线实战经验的量化研究员合作，上面这些问题一一遇到过，各种解法也都试过。正是这样的现实处境，促使我们做出 Qlib ，把 AI 技术用到量化投资里。

[2]https://github.com/QUANTAXIS/QUANTAXIS

面向 AI 的框架 Qlib 依照现代研究工作流做模块化设计，为容纳 AI 技术留出最大的灵活性。量化研究员可以扩展其中的模块，再拼出一条工作流，高效验证自己的想法。每个模块里，Qlib 都给了若干默认实现，它们在实盘投资中表现良好。有了这些现成模块，研究员只需盯住自己关心的那一个模块，不必被其他琐碎细节牵扯精力。除代码之外，部分模块的计算与数据同样可以共享。因此 Qlib 面向用户的定位是平台，而不是工具箱。

高性能基础设施 数据处理的性能，对 AI 这类数据驱动的方法至关重要。作为面向 AI 的平台，Qlib 配了一套高性能数据基础设施。Qlib 自带时间序列平铺文件数据库[3]，专为金融数据的科学计算而设。在量化投资研究的若干典型数据处理任务上，它比通用数据库、时间序列数据库这些当下流行的存储方案快得多。该数据库还带有表达式引擎，因子与特征的编写和计算都因此加快，一批依赖表达式计算的研究课题才得以开展。

机器学习的上手指引 Qlib 已内置若干量化投资的典型数据集；在这些数据集上，常见机器学习算法能学到有泛化能力的模式。面向机器学习用户，Qlib 给出基本的上手指引，并内置了一批设定合理的任务，其特征空间与目标标签都已配好；常用的超参数优化工具也一并提供。有了指引和合理的设定，模型学到的是泛化更好的模式，而不是把噪声拟合进去。

3.2 面向 AI 的框架

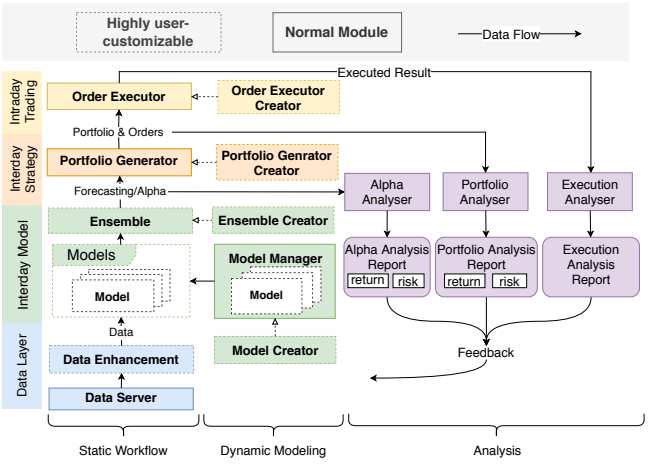

图 1: Qlib 的各功能模块，以及用这些模块串起来的一条典型量化研究工作流

图 1 给出 Qlib 的整体框架。该框架有三个目标：1）容纳现代 AI 技术；2）让量化研究员以最小代价搭起一整条研究工作流；3）给研究员留足灵活性，让他们专心探索感兴趣的问题，不被其余环节分心。

从系统设计的角度看，这样的目标自然导向模块化。系统按当下的实战研究工作流拆成若干独立模块。量化投资的研究方向，无论走传统路线还是基于 AI，大多可以看作对一个或多个模块接口的实现。每个模块里，Qlib 都为用户备了几种在实盘投资中好用的典型实现。模块同时允许研究员覆写已有方法，用来试新想法。有了这套框架，研究员提出一个新点子，再连同其余模块一起测整体表现，代价都很低。

图 1 列出 Qlib 的各个模块，并按一条典型工作流把它们连起来。每个模块对应量化投资中的一个典型子任务，模块里的一份实现就是该任务的一个解法。下面逐个介绍模块，并举出已有量化研究的例子，说明 Qlib 如何把它们纳进来。

起点是左下角的**数据服务器（Data Server）模块**，它提供一个数据引擎，负责查询并处理原始数据。取到数据后，研究员可在**数据增强（Data Enhancement）模块**里搭自己的数据集。为了得到更好的数据集，研究者尝试过许多方案，核心是挖掘并构造有效的因子与特征[Potvin 等, 2004; Neely 等, 1997; Allen and Karjalainen, 1999; Kakushadze, 2016]。直接生成训练用的数据集[Feng 等, 2019]是另一条给出数据集方案的研究路线。**模型构建（Model Creator）模块**在数据集上学出模型。近年来，大量研究者尝试用各类模型从金融数据集中挖掘交易信号[Sezer 等, 2019]。元学习[Vilalta and Drissi, 2002]要学的是如何学习，它为模型构建模块带来一种新的学习范式。现代研究工作流里，给金融数据建模的方法众多，模型管理系统于是成了工作流中不可缺的一环，**模型管理（Model Manager）模块**正是为量化研究员处理这类问题而设。模型一多，集成学习就成了提升机器学习模型表现与鲁棒性的有效手段，金融领域用得很频繁[Qiu 等, 2014; Yang 等, 2017; Zhao 等, 2017]，这由**模型集成（Model Ensemble）模块**承担。**组合生成器（Portfolio Generator）模块**把模型输出的交易信号转成一个投资组合，这件事通常称为组合管理[Qian 等, 2007]，其中最流行的方案来自 Barra [Sheikh, 1996]。有了目标组合，我们再用高保真交易模拟器**订单执行器（Order Executor）模块**检验策略表现，并用**分析器（Analyser）模块**自动分析交易信号、投资组合与执行结果。订单执行器模块的定位是可响应的模拟器，而不是一个回测函数；这样它就能为需要环

[3]https://en.wikipedia.org/wiki/Flat-file_database

境反馈的学习范式（如强化学习）提供底座，而反馈正由分析器模块产生。

量化投资的数据是时间序列，随时间不断更新，样本内数据集也随之变大。用上新数据的常见做法是定期更新模型[Wang 等, 2019b]。除了更充分地利用增长中的样本内数据，动态更新模型[Yang 等, 2019]与交易策略[Wang 等, 2019a]还能让表现更进一步，原因在于股票市场本身是动态的[Adam 等, 2016]。可见，像**静态工作流（Static Workflow）**那样固定一套模型和交易策略，显然不是最优解。模型与策略的动态更新，是量化投资的一个重要研究方向；**动态建模（Dynamic Modeling）**中的各模块为这类方案提供接口与基础设施。

3.3 高性能基础设施

金融数据

本节归纳量化研究对数据的要求。量化研究中最常用的数据都是如下形式

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

其中 $x_{i,t,a}$ 是基本类型（如 float、int）的取值，$Inst$ 是金融标的集合（如股票、期权等），$Time$ 是时间戳集合（如股票市场的交易日），$Attr$ 是标的可能具有的属性集合（如开盘价、成交量、市值），T 是数据的最新时间戳（如最近一个交易日）。$x_{i,t,a}$ 表示标的 i 在时刻 t 的属性 a 之值。

要指定一组随时间变化的金融标的，还需要标的池信息

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

标普 500 指数 [4] 就是 $Pool$ 的典型例子。

数据更新是必备能力。已有的历史数据不随时间改变，只需追加新数据。形式化的更新操作为

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

用户查询可形式化为

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

它表示：在给定的标的池与时间范围内，取一批标的的若干属性。

[4] https://en.wikipedia.org/wiki/S%26P_500_Index

这类需求相当简单，不少现成的开源方案都支持这些操作。我们把它们归为三类，每类列出常见实现。

- 通用数据库：MySQL[MySQL, 2001]、MongoDB[Chodorow, 2013]
- 时间序列数据库：InfluxDB [Naqvi 等, 2017]
- 科学计算数据文件：以 numpy[Oliphant, 2006] 数组或 pandas[McKinney, 2011] dataframe 组织的数据

通用数据库能容纳格式与结构各异的数据，还带来索引、事务、实体关系模型等一整套成熟机制。这些机制大多只是给特定任务压上沉重的依赖和多余的复杂度，并不解决具体场景中的关键问题。时间序列数据库针对时序数据优化了数据结构与查询，但同样不是为量化研究设计的：量化研究的数据通常采用紧凑的数组格式，以便科学计算借力硬件加速。数据若能从磁盘一路到客户端始终保持紧凑的数组格式、无需格式转换，便可省下大量时间。可惜通用数据库与时间序列数据库为了通用，存储和传输数据时用的都是另一套格式，对科学计算而言效率低下。

数据库效率不足，基于数组的数据格式因此在科学界流行开来。科学计算中的主流实现是 numpy 数组和 pandas dataframe，落盘时多存成 HDF[5] 或 pickle[6]。这类格式依赖轻，科学计算效率很高，但整份数据挤在单个文件里，更新和查询都很困难。

考察过上述存储方案后我们发现，没有一种能很好契合量化研究场景，必须为它定制一套专门的设计。

文件存储设计

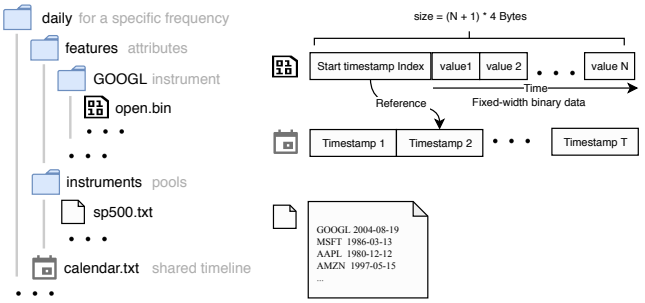

图 2: 平铺文件数据库示意；左侧是文件的结构，右侧是文件的内容

[5] https://en.wikipedia.org/wiki/Hierarchical_Data_Format

[6] https://docs.python.org/3/library/pickle.html

图 2 给出文件存储设计。如图左侧所示，Qlib 以树形结构组织文件。数据按频率、标的、属性拆分到不同的文件夹与文件中。所有属性值以紧凑的定宽格式存成二进制数据，于是按字节寻址成为可能。公共时间轴单独存放在名为“calendar.txt”的文件里。属性值数据文件的前 4 个字节写的是时间轴上的索引值，用来标明该序列数据的起始时间戳。有了这个起始时间索引，Qlib 就能把所有取值在时间维度上对齐。

数据以紧凑格式存储，拼成数组供科学计算时效率很高。这种格式既有基于数组的数据在科学计算中的高性能，又满足量化投资场景的数据更新要求。全部数据按时间排列，新数据追加即可更新，很省时；属性与标的分文件存放，增删都直接而高效。这套设计极为轻量，没有数据库的开销，Qlib 的性能因此很高。

表达式引擎

在基础数据之上开发新因子/特征是相当常见的工作，占去许多量化研究员大量时间。用代码实现这类因子费时，计算过程同样费时。为此 Qlib 提供表达式引擎，把这类工作的投入压到最低。

因子/特征本质上是把基础数据变换为目标值的函数，而这个函数可以拆成一系列表达式的组合。表达式引擎正是循此思路设计的。有了它，量化研究员写表达式就能实现新因子/特征，不必再写复杂代码。举例来说，布林带（Bollinger Band）技术指标 [Bollinger, 2002] 是应用很广的技术因子，用表达式引擎实现它的上轨只需一句简单表达式 *“(MEAN($close, N)+2*STD($close, N)-$close)/MEAN($close, N)”*。

这样的实现简洁、可读、可复用、易维护。用户只用一串简单表达式就能搭出数据集。搜索表达式以构造有效的交易信号是一类典型的研究课题，已有不少研究者做过探索 [Allen and Karjalainen, 1999; Neely 等, 1997; Potvin 等, 2004]。表达式引擎正是这类课题不可或缺的工具。

缓存系统

为避免重复计算，Qlib 内置了缓存系统，由内存缓存和磁盘缓存两部分组成。

内存缓存 Qlib 用表达式引擎计算因子/特征时，会先把表达式解析成语法树。各节点算出的结果都存进内存中的 LRU（Least Recently Used，最近最少使用）缓存，相同（子）表达式的重复计算就此省去。

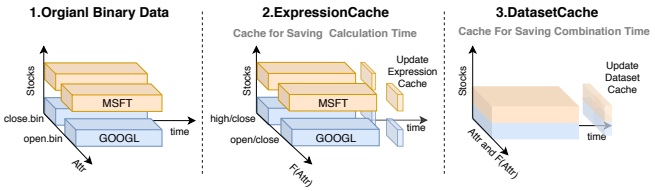

图 3: Qlib 的磁盘缓存系统；表达式缓存省去表达式的计算时间，数据集缓存省去数据的合并时间

磁盘缓存 量化投资中典型的数据处理工作流分三步：取原始数据、计算表达式、把数据合并成数组供科学计算。计算表达式与合并数据都很耗时，若能缓存共享的中间数据，便可省下不少时间。实际的数据处理任务中，可共享的中间结果很多，比如同一个表达式的计算就能被不同的数据处理任务共用。因此 Qlib 设计了两级磁盘缓存机制。缓存系统见图 3，左侧是第 3.3 节所述的原始数据。第一级是表达式缓存，把算出的表达式全部写入磁盘缓存，其数据结构与原始数据相同；有了表达式缓存，同一个表达式只会计算一次。表达式缓存之后是数据集缓存，存放合并好的数据，用来省去合并时间。两级缓存的数据都按时间排列、可在时间维度上索引，因此查询时间变化后磁盘缓存依然能复用。按时间排列还带来一个好处：Qlib 可以追加新数据来更新，数据维护因此轻松许多。

3.4 机器学习指引

正如第 2 节所述，机器学习算法的指引十分重要。Qlib 为机器学习算法提供了典型数据集，还内置了数据预处理、学习目标等典型任务设置，研究者不必事事从零摸索。这些指引为研究者提供了大量领域知识，便于他们踏入这一研究领域。

对大多数机器学习算法而言，超参数优化是取得更好泛化能力的必要步骤。这一步虽然重要，却耗时且重复。因此，Qlib 提供了**超参数调优引擎（Hyperparameters Tuning Engine, HTE）**来简化这项工作。HTE 提供了一个接口，用于定义超参数搜索空间 Θ，并自动搜索最优超参数 θ。

在时间序列建模这一典型金融任务中，新数据按时间顺序不断到来。为利用新数据，模型必须定期在新数据上重新训练。每次重新训练得到的最优超参数 θ 会发生变化，但通常与上一次的最优值接近。HTE 为金融任务的超参数优化提供了专门机制：它为搜索空间生成新的分布，使模型能以更少的试验次数逼近最优点。搜索

	HDF5	MySQL	MongoDB	InfluxDB	Qlib -E -D	Qlib +E -D	Qlib +E +D
存储占用 (MB)	**287**	1,332	911	**394**	**303**	802	1,000
加载数据 (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
计算表达式 (s)	179.8±4.4				137.7±7.6	**35.3±2.3**	-
转换索引 (s)	-				3.6±0.1		-
按标的池筛选 (s)	3.39 ±0.24						-
合并数据 (s)	1.19±0.30						-
总计 (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**
总计 (64CPUs) (s)	-				8.8±0.6	**4.2±0.2**	-

表 1: 不同存储方案的性能对比

θ 的分布可形式化为

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

其中 p_{prior} 是原始的超参数搜索空间；$\varphi_{\theta_{prev},\sigma^2}(x) \sim \mathcal{N}(\theta_{prev}, \sigma^2)$；$\theta_{prev}$ 是上一次模型训练得到的最优超参数。超参数搜索空间的定义域保持不变，但 θ_{prev} 附近的概率密度会增大。

4 用例与性能评测

4.1 用例

Qlib 提供**配置驱动流程引擎（Config-Driven Pipeline Engine, CDPE）**，帮助研究者更轻松地搭起图 1 所示的整套研究工作流。一个简单的配置文件就能定义一条工作流，形式见图 4（其中琐碎细节以“...”代替）。这套接口并不强制使用，用户仍可像搭积木一样用代码组装量化研究工作流，灵活度不受限制。

```
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

4.2 性能评测

对 AI 这类数据驱动方法而言，数据处理的性能举足轻重。Qlib 面向 AI，为数据存储与数据处理给出了一整套方案。为展示 Qlib 的性能，我们把 Qlib 与另外几种方案作了对比，它们已在第 3.3 节讨论过，包括 *HDF5*、*MySQL*、*MongoDB*、*InfluxDb* 与 *Qlib*。其中 *Qlib +E -D* 表示开启表达式缓存、关闭数据集缓存的 Qlib，其余标记以此类推。

各方案要完成的任务是：从某股票市场的 OHLCV[7] 日频数据构建一个数据集，其中既有数据查询也有数据处理。最终数据集含 14 个由 OHLCV 数据派生的因子/特征，如“Std($close, 5)/$close”。数据时间跨度为 2007/1/1 至 2020/1/1。股票池每日 800 只，且逐日变动。

除比较各方案的总耗时外，我们还把任务拆成以下几步，以便看清细节。

- **加载数据（Load Data）**把 OHLCV 数据或缓存读入内存，转成科学计算所用的数组格式。
- **计算表达式（Compute Expr.）**算出派生的因子/特征。
- **转换索引（Convert Index）**这一步只有 Qlib 需要。Qlib 的原始数据中不保存索引，即时间戳与股票代码，因此必须另行建立数据索引。
- **过滤数据（Filter data）**按指定标的池筛选股票数据。以标普 500 为例，其成分股累计超过 1000 只，但每日只纳入 500 只。某只股票即便一度入选标普

[7]股票的开盘价、最高价、最低价、收盘价与成交量

500，只要在当天不属于标普 500 成分股，其数据就应剔除。这一步无法在加载数据时顺带完成，因为部分派生特征要用到历史 OHLCV 数据。

- **合并数据（Combine data）**把不同股票的数据拼接成一整块数组形式的数据。

结果见表 1。Qlib 的紧凑存储在体积与加载速度上都接近 HDF5 这一专用于科学计算的数据文件格式，而几种数据库在加载数据上耗时过多。查看底层实现后我们发现，通用数据库与时序数据库两条路线都让数据穿过了过多的接口层，还做了不必要的格式转换；这些开销把加载过程拖得很慢。得益于 Qlib 的内存缓存，Qlib -E -D 在 Compute Expr. 一步上省下约 24% 的时间。Qlib 还提供表达式缓存与数据集缓存两套机制。Qlib +E -D 开启表达式缓存后，若表达式缓存全部命中，Compute Expr. 的耗时可省 80.4%。此时 Qlib +E -D 的主要耗时来自把各只股票的因子/特征合并成一整块数组数据，这部分计入 Compute Expr. 步骤。除计算开销外，最耗时的一步就是数据合并。数据集缓存正是为削减这类开销而设计。Qlib +E +D 一列表明，耗时进一步下降。

Qlib 还能调用多个 CPU 核心加速计算。表 1 最后一行显示，多 CPU 下 Qlib 的耗时大幅下降。Qlib +E +D 已无进一步加速的余地，它只读取已有缓存，几乎不做计算。

4.3 Qlib 的更多内容

Qlib 是持续开发中的开源平台，更详细的文档见其 github 仓库 [8]。本文未详细介绍的许多特性，如客户端-服务器架构的数据服务、分析系统、云端自动部署，都能在线上仓库中找到。欢迎大家参与贡献。

5 结论

本文梳理了 AI 时代量化研究者面临的实际问题，并据此设计并实现了 Qlib。它的目标是让每一位量化研究者都能释放 AI 技术在量化投资中的潜力。

参考文献

[Adam 等, 2016] Klaus Adam, Albert Marcet, and Juan Pablo Nicolini. *Stock market volatility and learning*, 2016.

[Allen and Karjalainen, 1999] Franklin Allen and Risto Karjalainen. Using genetic algorithms to find technical trading rules. *Journal of financial Economics*, 51(2):245–271, 1999.

[Bollinger, 2002] John Bollinger. *Bollinger on Bollinger bands*. McGraw Hill Professional, 2002.

[Chodorow, 2013] Kristina Chodorow. *MongoDB: the definitive guide: powerful and scalable data storage*. " O'Reilly Media, Inc.", 2013.

[Deng 等, 2016] Yue Deng, Feng Bao, Youyong Kong, Zhiquan Ren, and Qionghai Dai. Deep direct reinforcement learning for financial signal representation and trading. *IEEE transactions on neural networks and learning systems*, 28(3):653–664, 2016.

[Feng 等, 2019] Fuli Feng, Huimin Chen, Xiangnan He, Ji Ding, Maosong Sun, and Tat-Seng Chua. Enhancing stock movement prediction with adversarial training. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 5843–5849. AAAI Press, 2019.

[Firth, 2004] N Firth. Why use quantlib. *Paper available at: http://www. quantlib. co. uk/publications/quantlib. pdf*, 2004.

[Kakushadze, 2016] Zura Kakushadze. 101 formulaic alphas. *Wilmott*, 2016(84):72–81, 2016.

[Li 等, 2016] Bin Li, Doyen Sahoo, and Steven CH Hoi. Olps: a toolbox for on-line portfolio selection. *The Journal of Machine Learning Research*, 17(1):1242–1246, 2016.

[McKinney, 2011] Wes McKinney. pandas: a foundational python library for data analysis and statistics. *Python for High Performance and Scientific Computing*, 14, 2011.

[Murphy, 1999] John J Murphy. *Technical analysis of the financial markets: A comprehensive guide to trading methods and applications*. Penguin, 1999.

[MySQL, 2001] AB MySQL. Mysql, 2001.

[Naqvi 等, 2017] Syeda Noor Zehra Naqvi, Sofia Yfantidou, and Esteban Zimányi. Time series databases and influxdb. *Studienarbeit, Université Libre de Bruxelles*, 2017.

[8] https://github.com/microsoft/qlib/

[Neely 等, 1997] Christopher Neely, Paul Weller, and Rob Dittmar. Is technical analysis in the foreign exchange market profitable? a genetic programming approach. *Journal of financial and Quantitative Analysis*, 32(4):405–426, 1997.

[Oliphant, 2006] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[Petkova, 2006] Ralitsa Petkova. Do the fama–french factors proxy for innovations in predictive variables? *The Journal of Finance*, 61(2):581–612, 2006.

[Potvin 等, 2004] Jean-Yves Potvin, Patrick Soriano, and Maxime Vallée. Generating trading rules on the stock markets with genetic programming. *Computers & Operations Research*, 31(7):1033–1047, 2004.

[Qian 等, 2007] Edward E Qian, Ronald H Hua, and Eric H Sorensen. *Quantitative equity portfolio management: modern techniques and applications*. CRC Press, 2007.

[Qiu 等, 2014] Xueheng Qiu, Le Zhang, Ye Ren, Ponnuthurai N Suganthan, and Gehan Amaratunga. Ensemble deep learning for regression and time series forecasting. In *2014 IEEE symposium on computational intelligence in ensemble learning (CIEL)*, pages 1–6. IEEE, 2014.

[Sezer 等, 2019] Omer Berat Sezer, Mehmet Ugur Gudelek, and Ahmet Murat Ozbayoglu. Financial time series forecasting with deep learning: A systematic literature review: 2005-2019. *arXiv preprint arXiv:1911.13288*, 2019.

[Sheikh, 1996] Aamir Sheikh. Barra's risk models. *Barra Research Insights*, pages 1–24, 1996.

[Vilalta and Drissi, 2002] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.

[Wang 等, 2019a] Lewen Wang, Weiqing Liu, Xiao Yang, and Jiang Bian. Conservative or aggressive? confidence-aware dynamic portfolio construction. In *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1–5. IEEE, 2019.

[Wang 等, 2019b] Shouxiang Wang, Xuan Wang, Shaomin Wang, and Dan Wang. Bi-directional long short-term memory method based on attention mechanism and rolling update for short-term load forecasting. *International Journal of Electrical Power & Energy Systems*, 109:470–479, 2019.

[Yang 等, 2017] Bing Yang, Zi-Jia Gong, and Wenqi Yang. Stock market index prediction using deep neural network ensemble. In *2017 36th Chinese Control Conference (CCC)*, pages 3882–3887. IEEE, 2017.

[Yang 等, 2019] Xiao Yang, Weiqing Liu, Lewen Wang, Cheng Qu, and Jiang Bian. A divide-and-conquer framework for attention-based combination of multiple investment strategies. In *2019 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pages 1–5. IEEE, 2019.

[Zhao 等, 2017] Yang Zhao, Jianping Li, and Lean Yu. A deep learning ensemble approach for crude oil price forecasting. *Energy Economics*, 66:9–16, 2017.