CUFEL Arena Agent SDK

为 CUFEL-Q Arena 平台提供标准化的 Agent 开发基类。

安装

```
pip install cufel-arena-agent
```

或从源码安装：

```
cd AgentBase
pip install -e .
```

安装可选依赖：

```
# FOF Agent 需要 PostgreSQL 支持
pip install cufel-arena-agent[postgres]

# 使用 .env 文件管理配置
pip install cufel-arena-agent[dotenv]

# 安装所有可选依赖
pip install cufel-arena-agent[all]
```

快速开始

ETF Agent

ETF Agent 直接投资 ETF，需要实现 `load_current_data` 和 `get_current_holdings` 两个方法：

```
from cufel_arena_agent import ETFAgentBase
from quantchdb import ClickHouseDatabase

class MyETFAgent(ETFAgentBase):
    def __init__(self, db_config=None, **kwargs):
        super().__init__(name="MyETFAgent", db_config=db_config, **kwargs)

    def load_current_data(self, curr_date: str):
        db = ClickHouseDatabase(config=self.db_config)
        return db.fetch(f"SELECT code, close FROM etf.etf_day WHERE
date='{curr_date}'")

    def get_current_holdings(self, curr_date: str, feedback: str = None, theta:
```

```
float = None):
        df = self.load_current_data(curr_date)
        codes = df['code'].tolist()
        weight = 1.0 / len(codes)
        return {curr_date: {code: weight for code in codes}}
```

FOF Agent

FOF Agent 投资于其他 ETF Agents，可以动态筛选目标：

```
from cufel_arena_agent import FOFAgentBase

class MyFOFAgent(FOFAgentBase):
    def __init__(self, top_n=3, db_config=None, **kwargs):
        super().__init__(name="MyFOFAgent", db_config=db_config, **kwargs)
        self.top_n = top_n

    def load_current_data(self, curr_date: str):
        # 动态获取所有 ETF Agents 的收益率
        return self.data_client.get_multi_agents_returns(agent_type='ETF',
end_date=curr_date)

    def get_current_holdings(self, curr_date: str, feedback: str = None, theta:
float = None):
        returns_df = self.load_current_data(curr_date)
        # 计算夏普比率，选择 top_n 个
        sharpe = (returns_df.mean() * 252) / (returns_df.std() * 252**0.5)
        top_agents = sharpe.nlargest(self.top_n).index.tolist()
        weight = 1.0 / len(top_agents)
        return {curr_date: {agent: weight for agent in top_agents}}
```

核心概念

必须实现的方法

方法	说明
`load_current_data(curr_date)`	加载当前日期所需的数据
`get_current_holdings(curr_date, feedback=None, theta=None)`	返回当前日期的持仓

如果希望做日内交易优化，必须要定义 `get_current_holdings_intraday(curr_datetime, feedback=None, theta=None)`。`curr_datetime`需要为字符串类型，例如"2023-01-15 10:00:00"。

方法参数说明

参数	类型	说明
`curr_date`	str	当前日期，格式为 `YYYY-MM-DD`

参数	类型	说明
`feedback`	str, optional	来自上层 Agent 的反馈信息
`theta`	float, optional	风险偏好系数

持仓格式

```
{
    '2025-01-15': {
        'asset_code_1': 0.5,
        'asset_code_2': 0.3,
        'asset_code_3': 0.2
    }
}
```

约束条件：

- 所有权重必须非负
- 权重总和不超过 1（允许持有现金，需要用货币基金 ETF 的 code 来表示持有现金）

日期格式

所有日期参数必须使用 `YYYY-MM-DD` 格式。

数据库配置

ETF Agent (ClickHouse)

```
# 方式1：实例配置
agent = MyETFAgent(db_config={
    "host": "localhost",
    "port": "8123",
    "user": "default",
    "password": "xxx",
    "database": "etf"
})

# 方式2：全局配置
from cufel_arena_agent import ConfigLoader
ConfigLoader.set_clickhouse_config({...})
```

FOF Agent (PostgreSQL)

```
# 方式1：实例配置
agent = MyFOFAgent(db_config={
    "host": "localhost",
    "port": "5432",
```

```
    "user": "postgres",
    "password": "xxx",
    "database": "arena"
})

# 方式2：全局配置
ConfigLoader.set_database_config({...})
```

使用环境变量

推荐使用 `.env` 文件管理数据库配置：

```
# .env 文件
DB_HOST_Server=localhost
DB_PORT_Server=8123
DB_USER_Server=default
DB_PASSWORD_Server=xxx
DB_DATABASE_Server=etf
```

```
import os
from dotenv import load_dotenv
load_dotenv()

db_config = {
    'host': os.getenv('DB_HOST_Server'),
    'port': os.getenv('DB_PORT_Server'),
    'user': os.getenv('DB_USER_Server'),
    'password': os.getenv('DB_PASSWORD_Server'),
    'database': os.getenv('DB_DATABASE_Server')
}
```

ArenaDataClient API 参考

`ArenaDataClient` 是 FOF Agent 获取 ETF Agents 数据的核心客户端。FOF Agent 可通过 `self.data_client` 访问。

初始化

```
from cufel_arena_agent import ArenaDataClient

client = ArenaDataClient(db_config={
    "host": "localhost",
    "port": "5432",
    "user": "postgres",
    "password": "xxx",
```

```
    "database": "arena"
})
```

Agent 信息查询

`get_all_agents(agent_type=None)`

获取所有 Agent 的基础信息。

```
# 获取所有 ETF Agents
df = client.get_all_agents(agent_type='ETF')
# 返回 DataFrame: agent_id, agent_name, type, agent_dirname, other_info,
created_at, update_at
```

`get_agent_names(agent_type=None)`

获取 Agent 名称列表。

```
names = client.get_agent_names(agent_type='ETF')
# 返回: ['Agent1', 'Agent2', ...]
```

`get_agent_id(agent_name)`

根据名称获取 Agent ID。

```
agent_id = client.get_agent_id('MyAgent')
# 返回: 123 或 None
```

`get_agent_info(agent_name)`

获取单个 Agent 的完整信息。

```
info = client.get_agent_info('MyAgent')
# 返回: {'agent_id': 123, 'agent_name': 'MyAgent', 'type': 'ETF', ...}
```

净值数据查询

`get_agent_nav(agent_name, start_date=None, end_date=None)`

获取单个 Agent 的净值序列。

```
nav = client.get_agent_nav('MyAgent', start_date='2024-01-01', end_date='2024-12-31')
# 返回: pd.Series，index 为日期
```

get_multi_agents_nav(agent_names=None, agent_type=None, start_date=None, end_date=None, fillna_method=None)

获取多个 Agent 的净值数据（宽表格式）。

```
# 获取指定 Agents
nav_df = client.get_multi_agents_nav(
    agent_names=['Agent1', 'Agent2'],
    start_date='2024-01-01',
    end_date='2024-12-31',
    fillna_method='ffill'  # 向前填充缺失值
)

# 获取所有 ETF Agents
nav_df = client.get_multi_agents_nav(agent_type='ETF')
# 返回: pd.DataFrame，columns 为 agent_name，index 为日期
```

get_nav_date_range(agent_name=None)

获取净值数据的日期范围。

```
date_range = client.get_nav_date_range()
# 返回: {'min_date': '2024-01-01', 'max_date': '2024-12-31'}
```

收益率数据查询

get_agent_returns(agent_name, start_date=None, end_date=None)

获取单个 Agent 的日收益率序列。

```
returns = client.get_agent_returns('MyAgent', end_date='2024-12-31')
# 返回: pd.Series，index 为日期
```

get_multi_agents_returns(agent_names=None, agent_type=None, start_date=None, end_date=None, fillna_value=0.0)

获取多个 Agent 的日收益率数据（宽表格式）。

```
# 获取所有 ETF Agents 的收益率
returns_df = client.get_multi_agents_returns(
    agent_type='ETF',
    start_date='2024-01-01',
    end_date='2024-12-31',
    fillna_value=0.0
)
# 返回: pd.DataFrame，columns 为 agent_name，index 为日期
```

持仓数据查询

get_agent_daily_positions(agent_name, start_date=None, end_date=None)

获取单个 Agent 的每日持仓权重。

```
positions = client.get_agent_daily_positions('MyAgent', start_date='2024-12-01')
# 返回: {'2024-12-01': {'510300': 0.5, '510500': 0.5}, ...}
```

get_multi_agents_daily_positions(agent_names=None, agent_type='ETF', start_date=None, end_date=None)

获取多个 Agent 的每日持仓权重。

```
positions = client.get_multi_agents_daily_positions(agent_type='ETF')
# 返回: {'Agent1': {'2024-12-01': {...}, ...}, 'Agent2': {...}}
```

get_agents_positions_for_date(agent_names=None, agent_type='ETF', target_date=None)

获取指定日期所有 Agent 的持仓权重。

```
# 获取最新持仓
positions = client.get_agents_positions_for_date(agent_type='ETF')
# 返回: {'Agent1': {'510300': 0.5, ...}, 'Agent2': {...}}

# 获取指定日期持仓
positions = client.get_agents_positions_for_date(target_date='2024-12-01')
```

回测指标查询

get_agent_metrics(agent_name, start_date=None, end_date=None, latest_only=False)

获取单个 Agent 的回测指标。

```
# 获取所有历史指标
metrics_df = client.get_agent_metrics('MyAgent')

# 只获取最新指标
metrics_df = client.get_agent_metrics('MyAgent', latest_only=True)
# 返回: pd.DataFrame，包含 date 和 metrics 中的各字段
```

get_latest_metrics(agent_names=None, agent_type=None)

获取多个 Agent 的最新回测指标。

```
metrics_df = client.get_latest_metrics(agent_type='ETF')
# 返回: pd.DataFrame，每行一个 Agent，包含 agent_name, date, sharpe_ratio, cum_ret
等
```

get_multi_agents_metrics(agent_names=None, agent_type=None, metric_name='sharpe_ratio', start_date=None, end_date=None)

获取多个 Agent 的指定指标时间序列。

```
# 获取夏普比率时间序列
sharpe_df = client.get_multi_agents_metrics(
    agent_type='ETF',
    metric_name='sharpe_ratio'
)

# 获取累计收益率时间序列
cumret_df = client.get_multi_agents_metrics(metric_name='cum_ret')

# 可用指标: sharpe_ratio, cum_ret, max_drawdown, turnover 等
# 返回: pd.DataFrame，columns 为 agent_name，index 为日期
```

综合查询方法

get_agents_summary(agent_type=None)

获取所有 Agent 的汇总信息（包含最新指标）。

```
summary_df = client.get_agents_summary(agent_type='ETF')
# 返回: pd.DataFrame，包含基础信息和最新指标
```

`get_data_for_fof(agent_names=None, agent_type='ETF', start_date=None, end_date=None)`

一次性获取 FOF Agent 构建所需的完整数据。

```
data = client.get_data_for_fof(agent_type='ETF', start_date='2024-01-01')
# 返回字典:
# {
#     'agents_info': pd.DataFrame,    # Agent 基础信息
#     'nav': pd.DataFrame,            # 净值宽表
#     'returns': pd.DataFrame,        # 收益率宽表
#     'sharpe': pd.DataFrame,         # 夏普比率时间序列
#     'cum_ret': pd.DataFrame,        # 累计收益率时间序列
#     'max_drawdown': pd.DataFrame,   # 最大回撤时间序列
#     'latest_metrics': pd.DataFrame  # 最新指标
# }
```

FOFAgentBase 便捷方法

FOFAgentBase 封装了 ArenaDataClient 的常用方法：

```
class MyFOFAgent(FOFAgentBase):
    def load_current_data(self, curr_date):
        # 获取净值
        nav_df = self.get_etf_agents_nav(agent_names=['A', 'B'],
end_date=curr_date)

        # 获取收益率
        returns_df = self.get_etf_agents_returns(end_date=curr_date)

        # 获取持仓
        positions = self.get_etf_agents_positions(target_date=curr_date)

        # 获取所有 ETF Agents 信息
        agents_info = self.get_all_etf_agents_info()

        return nav_df
```

API 速查表

ETFAgentBase

属性/方法	说明
`name`	Agent 名称
`agent_type`	Agent 类型 ('ETF')

属性/方法	说明
`db_config`	ClickHouse 数据库配置
`load_current_data(curr_date)`	加载数据（抽象方法）
`get_current_holdings(curr_date, feedback=None, theta=None)`	获取持仓（抽象方法）
`get_daily_holdings(start_date, end_date, theta=None)`	获取日期范围内的持仓

FOFAgentBase

属性/方法	说明
`name`	Agent 名称
`agent_type`	Agent 类型 ('FOF')
`target_agents`	目标 ETF Agents 列表（可选）
`data_client`	ArenaDataClient 实例
`load_current_data(curr_date)`	加载数据（抽象方法）
`get_current_holdings(curr_date, feedback=None, theta=None)`	获取持仓（抽象方法）
`get_daily_holdings(start_date, end_date, theta=None)`	获取日期范围内的持仓
`get_etf_agents_nav(...)`	获取 ETF Agents 净值
`get_etf_agents_returns(...)`	获取 ETF Agents 收益率
`get_etf_agents_positions(...)`	获取 ETF Agents 持仓
`get_all_etf_agents_info()`	获取所有 ETF Agents 信息

ConfigLoader

方法	说明
`set_clickhouse_config(config)`	设置 ClickHouse 配置
`get_clickhouse_config()`	获取 ClickHouse 配置
`set_database_config(config)`	设置 PostgreSQL 配置
`get_database_config()`	获取 PostgreSQL 配置
`reset()`	重置所有配置

注意事项

1. **方法签名**：必须严格按照基类定义的参数名称实现方法，包括 `curr_date`、`feedback`、`theta`
2. **日期格式**：所有日期参数会自动验证，必须为 `YYYY-MM-DD` 格式
3. **权重约束**：权重必须非负，总和不超过 1
4. **数据库配置**：使用环境变量或 `.env` 文件，不要硬编码密码

5. **theta 参数**：风险偏好系数，可用于调整策略的风险敞口

依赖

核心依赖：

- Python >= 3.8
- pandas >= 1.3.0
- numpy >= 1.20.0
- quantchdb >= 0.1.0

可选依赖：

- psycopg2-binary >= 2.9.0 (FOF Agent PostgreSQL 支持)
- python-dotenv >= 0.19.0 (环境变量管理)

License

MIT License