Set up environment

In [21]:
```
import gymnasium as gym
import numpy as np
import pandas as pd
import torch
```

数据准备

In [22]:
```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
device
```

Out[22]:
```
device(type='cuda')
```

In [23]:
```
tech_daily = pd.read_csv(r"data\科技股票.csv")
tech_daily.set_index('date', inplace=True)
tech_daily.columns=['AAPL','GOOG','MSFT']
tech_daily
```

Out[23]:

date	AAPL	GOOG	MSFT
2022-01-03	178.270326	144.088458	324.504611
2022-01-04	176.007789	143.434930	318.940290
2022-01-05	171.326001	136.717897	306.696845
2022-01-06	168.465997	136.616093	304.273361
2022-01-07	168.632504	136.073308	304.428464
...	...	...	...
2025-11-03	268.789439	284.120000	517.030000
2025-11-04	269.778480	278.060000	514.330000
2025-11-05	269.878383	284.750000	507.160000
2025-11-06	269.508741	285.340000	497.100000
2025-11-07	268.210000	279.700000	496.820000

967 rows × 3 columns

In [24]:
```
debt=pd.read_csv(r"data\无风险.csv",encoding='gbk')
debt.set_index('date', inplace=True)
debt.columns=['US_debt']
debt
```

Out[24]:

date	US_debt
2022-01-03	1.63
2022-01-04	1.66
2022-01-05	1.71
2022-01-06	1.73
2022-01-07	1.76
...	...
2025-11-03	4.13
2025-11-04	4.10
2025-11-05	4.17
2025-11-06	4.11
2025-11-07	4.11

963 rows × 1 columns

In [25]:
```
tmp=pd.read_csv(r"data\指数和贵金属.csv", encoding='gbk')
tmp.columns=['date','SP500','Gold']
tmp.set_index('date', inplace=True)
tmp
```

Out[25]:

date	SP500	Gold
2022-01-03	4796.56	1801.3
2022-01-04	4793.54	1814.9
2022-01-05	4700.58	1810.6
2022-01-06	4696.05	1790.9
2022-01-07	4677.03	1796.5
...	...	...
2025-11-03	6851.97	4013.7
2025-11-04	6771.55	3941.3
2025-11-05	6796.29	3990.4
2025-11-06	6720.32	3984.8
2025-11-07	6728.80	4007.8

1001 rows × 2 columns

In [26]:

```
df=pd.merge(tech_daily,debt,how='left',on='date')
df=pd.merge(df,tmp,how='left',on='date')
df['date']=pd.to_datetime(df.index)
df.set_index('date', inplace=True)
df[df.isnull().values == True]
```

Out[26]:

date	AAPL	GOOG	MSFT	US_debt	SP500	Gold
2022-10-10	138.103983	98.039090	223.685532	NaN	3612.39	1675.7
2022-11-11	147.475157	96.072547	241.112025	NaN	3992.93	1774.2
2023-10-09	177.082149	138.551849	324.923430	NaN	4335.66	1875.0
2024-10-14	230.005541	165.625113	416.016745	NaN	5859.85	2665.8
2024-11-11	223.220427	181.177047	414.895165	NaN	6001.35	2626.1
2025-10-13	247.420154	244.640000	514.050000	NaN	6654.72	4130.0

In [27]:
```
df.interpolate(method='time', inplace=True)
df
```

Out[27]:

date	AAPL	GOOG	MSFT	US_debt	SP500	Gold
2022-01-03	178.270326	144.088458	324.504611	1.63	4796.56	1801.3
2022-01-04	176.007789	143.434930	318.940290	1.66	4793.54	1814.9
2022-01-05	171.326001	136.717897	306.696845	1.71	4700.58	1810.6
2022-01-06	168.465997	136.616093	304.273361	1.73	4696.05	1790.9
2022-01-07	168.632504	136.073308	304.428464	1.76	4677.03	1796.5
...	...	...	...	...	...	...
2025-11-03	268.789439	284.120000	517.030000	4.13	6851.97	4013.7
2025-11-04	269.778480	278.060000	514.330000	4.10	6771.55	3941.3
2025-11-05	269.878383	284.750000	507.160000	4.17	6796.29	3990.4
2025-11-06	269.508741	285.340000	497.100000	4.11	6720.32	3984.8
2025-11-07	268.210000	279.700000	496.820000	4.11	6728.80	4007.8

967 rows × 6 columns

In [28]:

```python
# 保证日期有序
df = df.sort_index()

print("数据日期范围:", df.index.min(), "→", df.index.max())


# 生成滑动窗口函数
def generate_walk_forward_windows(index, train_years=2, test_months=6, step_months=6):
    """
    基于日期索引生成滑动窗口:
      - 每个窗口: 训练 train_years 年, 测试 test_months 个月
      - 每次向前滚动 step_months 个月
    返回:
      [{"train_start":..., "train_end":..., "test_start":..., "test_end":...}, ...]
```

```
    """
    windows = []

    start_date = index.min().normalize()
    last_date = index.max().normalize()

    current_train_start = start_date

    while True:
        train_start = current_train_start
        train_end = (
            train_start + pd.DateOffset(years=train_years) - pd.DateOffset(days=1)
        )
        test_start = train_end + pd.DateOffset(days=1)
        test_end = (
            test_start + pd.DateOffset(months=test_months) - pd.DateOffset(days=1)
        )

        # 测试区间超出数据范围就停止
        if test_end > last_date:
            break

        windows.append(
            {
                "train_start": train_start,
                "train_end": train_end,
                "test_start": test_start,
                "test_end": test_end,
            }
        )

        # 向前滚动 step_months 个月
        current_train_start = current_train_start + pd.DateOffset(months=step_months)

    return windows


# 生成窗口（你可以根据需要改 train_years / test_months / step_months）
windows = generate_walk_forward_windows(
    df.index, train_years=2, test_months=6, step_months=6
)
```

```
print(f"共生成 {len(windows)} 个滑动窗口：")
for i, w in enumerate(windows):
    print(
        f"Window {i}: "
        f"Train {w['train_start'].date()} ~ {w['train_end'].date()} | "
        f"Test {w['test_start'].date()} ~ {w['test_end'].date()}"
    )
```

```
数据日期范围: 2022-01-03 00:00:00 → 2025-11-07 00:00:00
共生成 3 个滑动窗口：
Window 0: Train 2022-01-03 ~ 2024-01-02 | Test 2024-01-03 ~ 2024-07-02
Window 1: Train 2022-07-03 ~ 2024-07-02 | Test 2024-07-03 ~ 2025-01-02
Window 2: Train 2023-01-03 ~ 2025-01-02 | Test 2025-01-03 ~ 2025-07-02
```

In [29]:

```
from math import inf

from networkx import sigma
from pyparsing import deque


class PortfolioOptimizationEnv(gym.Env):
    def __init__(
        self, tickers, window_size, start_date, end_date, initial_balance, seed=None
    ):
        super().__init__()
        self.tickers = tickers
        self.window_size = window_size
        self.initial_balance = initial_balance

        # 分别存储原始价格和指标
        self.raw_data, self.feature_data = self.get_data(tickers, start_date, end_date)
        self.n_features = self.feature_data.shape[1]

        self.action_space = gym.spaces.Box(low=0, high=1, shape=(len(tickers),))
        self.observation_space = gym.spaces.Box(
            low=-inf, high=inf, shape=(window_size, self.n_features)
        )

        self.return_window = deque(maxlen=window_size)
        self.last_action = np.ones(len(tickers)) / len(tickers)
```

```
        if seed is not None:
            np.random.seed(seed)
            self.action_space.seed(seed)
            torch.manual_seed(seed)
            if torch.cuda.is_available():
                torch.cuda.manual_seed_all(seed)

    def get_data(self, tickers, start_date, end_date):
        data = df.copy().dropna()
        data = data.loc[start_date:end_date, tickers]

        # 保存原始价格（用于计算投资组合收益）
        raw_data = data.copy()

        # 计算特征指标
        returns = data.pct_change()

        mom_frames = []
        for window in [5, 20]:
            mom = data / data.shift(window) - 1
            mom.columns = [f"{col}_mom_{window}" for col in data.columns]
            mom_frames.append(mom)

        vol = returns.rolling(window=20, min_periods=1).std()
        vol.columns = [f"{col}_vol_20" for col in data.columns]

        ma = data.rolling(window=20, min_periods=1).mean()
        ma_dev = data / ma - 1
        ma_dev.columns = [f"{col}_ma_dev_20" for col in data.columns]

        returns.columns = [f"{col}_ret" for col in data.columns]

        # 特征数据：returns, vol, ma_dev, momentum（不包含原始价格）
        feature_data = pd.concat([returns, vol, ma_dev] + mom_frames, axis=1)
        raw_data = raw_data.dropna()
        feature_data = feature_data.reindex(raw_data.index)
        feature_data.fillna(method="ffill", inplace=True)
        feature_data.fillna(method="bfill", inplace=True)

        return raw_data.dropna(), feature_data.dropna()
```

```python
    def reset(self, seed=None):
        self.balance = self.initial_balance
        self.current_step = self.window_size

        self.return_window.clear()
        self.last_action = np.ones(len(self.tickers)) / len(self.tickers)

        # 使用特征数据作为观察
        obs = self.feature_data.iloc[
            self.current_step - self.window_size : self.current_step
        ].values
        info = {"balance": self.balance}
        return obs, info

    def step(self, action):
        # SAC已经输出归一化的动作，因此不需要进行softmax
        action = np.asarray(action).ravel()
        action = np.clip(action, 0, 1)
        action = action / np.sum(action + 1e-8)

        prev_balance = self.balance

        # 从原始价格计算实际收益
        current_price = self.raw_data.iloc[self.current_step].values[
            : len(self.tickers)
        ]
        prev_price = self.raw_data.iloc[self.current_step - 1].values[
            : len(self.tickers)
        ]
        asset_returns = current_price / prev_price - 1

        self.return_window.append(asset_returns)

        # 基础奖励：投资组合收益
        portfolio_return = np.sum(asset_returns * action)
        self.balance = self.balance * (1 + portfolio_return)
        base_reward = np.log(self.balance / prev_balance)

        risk_penalty = 0
        if len(self.return_window) >= 5:
```

```
            R = np.vstack(self.return_window)
            cov_matrix = np.cov(R.T)
            sigma_p2 = action.T @ cov_matrix @ action
            risk_penalty = sigma_p2

        turnover = np.sum(np.abs(action - self.last_action))
        cost = turnover
        self.last_action = action

        # 总奖励
        lambda_risk = 1
        lambda_turnover = 0.001
        reward = base_reward - lambda_risk * risk_penalty - lambda_turnover * cost

        self.current_step += 1
        done = self.current_step >= len(self.raw_data) - 1

        obs_end = min(len(self.feature_data), self.current_step + self.window_size)
        obs_start = max(0, obs_end - self.window_size)
        obs = self.feature_data.iloc[obs_start:obs_end].values

        terminated = bool(done)
        truncated = False
        info = {"balance": self.balance}

        return obs, reward, terminated, truncated, info
```

In [30]:

```
tickers = df.columns.tolist()
window_size = 30
start_date = '2022-01-01'
end_date = '2025-09-01'
initial_balance = 10000
seed = 8

# Initialize the environment
env = PortfolioOptimizationEnv(
    tickers,
    window_size,
    start_date,
    end_date,
```

```
    initial_balance,
    seed)

# Get the initial state
state = env.reset(seed=seed)
# Sample and execute a random action
action = env.action_space.sample()
next_state, reward, terminated, truncated, info = env.step(action)
done = bool(terminated or truncated)
# print(f"State: {state}")
print(f"Action: {action}")
# print(f"Next state: {next_state}")
print(f"Reward: {reward}")
print(f"Balance: {info['balance']}")
print(f"Done: {done}")
```

```
Action: [0.32697228 0.98727685 0.31871083 0.78854895 0.86989653 0.39108482]
Reward: 0.015598913072904633
Balance: 10161.65097308565
Done: False
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:64: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="ffill", inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:65: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="bfill", inplace=True)
```

Training DRL agent

SAC

In [31]:

```
# ===== Walk-Forward 滑动窗口训练 + 测试 SAC =====
from stable_baselines3 import SAC
from stable_baselines3.common.vec_env import DummyVecEnv, VecNormalize
from stable_baselines3.common.monitor import Monitor
import numpy as np
import pandas as pd
```

```python
import matplotlib.pyplot as plt
import os

# 日志目录
log_dir = "./sb3_logs_wf"
os.makedirs(log_dir, exist_ok=True)


# 为某个时间段构建一个 env 的工厂函数
def make_env_for_period(start_date_str, end_date_str, monitor_file=None):
    """
    start_date_str, end_date_str: 'YYYY-MM-DD'
    monitor_file: 如不为 None，则用 Monitor 记录训练信息
    """

    def _init():
        env_ = PortfolioOptimizationEnv(
            tickers=tickers,
            window_size=window_size,
            start_date=start_date_str,
            end_date=end_date_str,
            initial_balance=initial_balance,
            seed=seed,
        )
        if monitor_file is not None:
            env_ = Monitor(env_, filename=monitor_file)
        return env_

    return _init


# 保存每个窗口的测试结果
wf_results = []

# 训练总步数（你可以调大，比如 50_000 / 100_000）
TOTAL_TIMESTEPS = 20_000

for i, w in enumerate(windows):
    print("=" * 80)
    print(f"Window {i}")
    print(f"Train: {w['train_start'].date()} ~ {w['train_end'].date()}")
```

```
    print(f"Test : {w['test_start'].date()} ~ {w['test_end'].date()}")

    train_start_str = w["train_start"].strftime("%Y-%m-%d")
    train_end_str = w["train_end"].strftime("%Y-%m-%d")
    test_start_str = w["test_start"].strftime("%Y-%m-%d")
    test_end_str = w["test_end"].strftime("%Y-%m-%d")

    # -------- 1) 构建训练环境 + 归一化 --------
    monitor_path = os.path.join(log_dir, f"monitor_train_window_{i}.csv")

    vec_train_env = DummyVecEnv(
        [make_env_for_period(train_start_str, train_end_str, monitor_file=monitor_path)]
    )
    vec_train_env = VecNormalize(
        vec_train_env, norm_obs=True, norm_reward=False, clip_obs=10.0
    )

    # -------- 2) 训练 SAC --------
    model = SAC(
        "MlpPolicy",
        vec_train_env,
        verbose=1,
        device=device,  # 你在前面 Cell 3 已定义 device
        # 下面这些超参数可以之后再细调
        learning_rate=3e-4,
        batch_size=256,
        buffer_size=200_000,
        tau=0.005,
        ent_coef="auto",
        train_freq=(1, "step"),
        gradient_steps=1,
    )

    model.learn(total_timesteps=TOTAL_TIMESTEPS)

    # 保存模型和归一化器（方便以后单独加载某个窗口）
    model_path = os.path.join(log_dir, f"sac_window_{i}.zip")
    vecnorm_path = os.path.join(log_dir, f"sac_vecnorm_window_{i}.pkl")
    model.save(model_path)
    vec_train_env.save(vecnorm_path)
    print(f"Saved model to {model_path}")
```

```
print(f"Saved VecNormalize to {vecnorm_path}")

# -------- 3) 构建测试环境（使用同一归一化统计） --------
test_env_raw = DummyVecEnv(
    [make_env_for_period(test_start_str, test_end_str, monitor_file=None)]
)
test_env = VecNormalize.load(vecnorm_path, test_env_raw)
test_env.training = False
test_env.norm_reward = False  # 测试时不要再归一化 reward

# -------- 4) 在测试集上回测该窗口策略 --------
obs = test_env.reset()
dones = [False]
balances = []
rewards = []
weights_seq = []

while not dones[0]:
    action, _ = model.predict(obs, deterministic=True)
    obs, reward, dones, infos = test_env.step(action)
    rewards.append(float(reward[0]))
    balances.append(float(infos[0].get("balance", np.nan)))
    weights_seq.append(infos[0].get("weights", None))

# 尝试从测试环境中拿到日期索引
try:
    base_env = test_env.envs[0].env
    while hasattr(base_env, "env"):
        base_env = base_env.env
    price_index = base_env.raw_data.index  # 你环境里存的是 raw_data
    start_idx = window_size
    end_idx = start_idx + len(balances)
    dates = price_index[start_idx:end_idx]
except Exception:
    dates = pd.RangeIndex(start=1, stop=len(balances) + 1)

balances_arr = np.asarray(balances, dtype=float)
rets = (
    balances_arr[1:] / balances_arr[:-1] - 1.0
    if len(balances_arr) > 1
    else np.array([])
```

```
    )

    # 简单算一下每个窗口的指标（年化收益、波动、Sharpe）
    def infer_ppy(idx):
        if isinstance(idx, pd.DatetimeIndex) and len(idx) >= 2:
            deltas = np.diff(idx.view("int64")) / 1e9 / 86400.0
            median_days = np.median(deltas) if len(deltas) else 1.0
            if median_days <= 0:
                return 252.0
            return 365.25 / median_days
        return 252.0

    PPY = infer_ppy(dates)

    if len(rets) > 1:
        total_growth = balances_arr[-1] / balances_arr[0]
        ann_return_geom = total_growth ** (PPY / len(rets)) - 1.0
        ann_vol = np.std(rets) * np.sqrt(PPY)
        sharpe = ann_return_geom / ann_vol if ann_vol > 1e-12 else np.nan
    else:
        total_growth = np.nan
        ann_return_geom = np.nan
        ann_vol = np.nan
        sharpe = np.nan

    wf_results.append(
        {
            "window_id": i,
            "train_start": w["train_start"],
            "train_end": w["train_end"],
            "test_start": w["test_start"],
            "test_end": w["test_end"],
            "test_steps": len(rets),
            "start_wealth": balances_arr[0] if len(balances_arr) > 0 else np.nan,
            "end_wealth": balances_arr[-1] if len(balances_arr) > 0 else np.nan,
            "total_growth": total_growth,
            "ann_return": ann_return_geom,
            "ann_vol": ann_vol,
            "sharpe": sharpe,
        }
    )
```

```
    print(f"Window {i} 测试结束: ")
    print(f"  终值资产: {balances_arr[-1]:.2f}  (初始 {balances_arr[0]:.2f})")
    print(f"  总增长倍数: {total_growth:.4f}")
    print(f"  年化收益:  {ann_return_geom:.2%}")
    print(f"  年化波动:  {ann_vol:.2%}")
    print(f"  Sharpe:    {sharpe:.3f}")

# -------- 5) 汇总所有窗口结果 --------
wf_results_df = pd.DataFrame(wf_results)
wf_results_df
```

```
================================================================================
Window 0
Train: 2022-01-03 ~ 2024-01-02
Test : 2024-01-03 ~ 2024-07-02
Using cuda device
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:64: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="ffill", inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:65: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="bfill", inplace=True)
```

```
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | -0.0944  |
| time/              |          |
|    episodes        | 4        |
|    fps             | 42       |
|    time_elapsed    | 43       |
|    total_timesteps | 1884     |
| train/             |          |
|    actor_loss      | -26.7    |
|    critic_loss     | 1.44     |
|    ent_coef        | 0.586    |
|    ent_coef_loss   | -5.38    |
|    learning_rate   | 0.0003   |
|    n_updates       | 1783     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | -0.0647  |
| time/              |          |
|    episodes        | 8        |
|    fps             | 42       |
|    time_elapsed    | 87       |
|    total_timesteps | 3768     |
| train/             |          |
|    actor_loss      | -38.3    |
|    critic_loss     | 0.37     |
|    ent_coef        | 0.333    |
|    ent_coef_loss   | -11      |
|    learning_rate   | 0.0003   |
|    n_updates       | 3667     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | -0.0934  |
| time/              |          |
|    episodes        | 12       |
|    fps             | 43       |
```

```
|    time_elapsed    | 130      |
|    total_timesteps | 5652     |
| train/             |          |
|    actor_loss      | -41.8    |
|    critic_loss     | 0.048    |
|    ent_coef        | 0.19     |
|    ent_coef_loss   | -16.6    |
|    learning_rate   | 0.0003   |
|    n_updates       | 5551     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | -0.0891  |
| time/              |          |
|    episodes        | 16       |
|    fps             | 43       |
|    time_elapsed    | 174      |
|    total_timesteps | 7536     |
| train/             |          |
|    actor_loss      | -41.8    |
|    critic_loss     | 0.147    |
|    ent_coef        | 0.108    |
|    ent_coef_loss   | -22      |
|    learning_rate   | 0.0003   |
|    n_updates       | 7435     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | -0.0965  |
| time/              |          |
|    episodes        | 20       |
|    fps             | 43       |
|    time_elapsed    | 218      |
|    total_timesteps | 9420     |
| train/             |          |
|    actor_loss      | -38.7    |
|    critic_loss     | 1.01     |
|    ent_coef        | 0.0614   |
|    ent_coef_loss   | -27.6    |
```

```
|    learning_rate   | 0.0003   |
|    n_updates       | 9319     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | -0.101   |
| time/              |          |
|    episodes        | 24       |
|    fps             | 43       |
|    time_elapsed    | 260      |
|    total_timesteps | 11304    |
| train/             |          |
|    actor_loss      | -36.3    |
|    critic_loss     | 0.975    |
|    ent_coef        | 0.035    |
|    ent_coef_loss   | -32.7    |
|    learning_rate   | 0.0003   |
|    n_updates       | 11203    |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | -0.102   |
| time/              |          |
|    episodes        | 28       |
|    fps             | 43       |
|    time_elapsed    | 304      |
|    total_timesteps | 13188    |
| train/             |          |
|    actor_loss      | -33.5    |
|    critic_loss     | 0.00989  |
|    ent_coef        | 0.0199   |
|    ent_coef_loss   | -38.6    |
|    learning_rate   | 0.0003   |
|    n_updates       | 13087    |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | -0.0938  |
```

```
| time/              |          |
|    episodes        | 32       |
|    fps             | 43       |
|    time_elapsed    | 348      |
|    total_timesteps | 15072    |
| train/             |          |
|    actor_loss      | -30.3    |
|    critic_loss     | 0.162    |
|    ent_coef        | 0.0113   |
|    ent_coef_loss   | -43.4    |
|    learning_rate   | 0.0003   |
|    n_updates       | 14971    |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | -0.0814  |
| time/              |          |
|    episodes        | 36       |
|    fps             | 43       |
|    time_elapsed    | 392      |
|    total_timesteps | 16956    |
| train/             |          |
|    actor_loss      | -27.6    |
|    critic_loss     | 0.00887  |
|    ent_coef        | 0.00648  |
|    ent_coef_loss   | -47.5    |
|    learning_rate   | 0.0003   |
|    n_updates       | 16855    |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | -0.0624  |
| time/              |          |
|    episodes        | 40       |
|    fps             | 43       |
|    time_elapsed    | 436      |
|    total_timesteps | 18840    |
| train/             |          |
|    actor_loss      | -24.3    |
```

```
|    critic_loss     | 0.00392  |
|    ent_coef        | 0.00371  |
|    ent_coef_loss   | -51.2    |
|    learning_rate   | 0.0003   |
|    n_updates       | 18739    |
---------------------------------
Saved model to ./sb3_logs_wf\sac_window_0.zip
Saved VecNormalize to ./sb3_logs_wf\sac_vecnorm_window_0.pkl
Window 0 测试结束:
  终值资产: 12143.97  (初始 9971.90)
  总增长倍数: 1.2178
  年化收益:  70.57%
  年化波动:  11.21%
  Sharpe:    6.293
================================================================================
Window 1
Train: 2022-07-03 ~ 2024-07-02
Test : 2024-07-03 ~ 2025-01-02
Using cuda device
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:64: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="ffill", inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:65: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="bfill", inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:64: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="ffill", inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:65: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="bfill", inplace=True)
```

```
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | 0.0683   |
| time/              |          |
|    episodes        | 4        |
|    fps             | 46       |
|    time_elapsed    | 40       |
|    total_timesteps | 1884     |
| train/             |          |
|    actor_loss      | -26.1    |
|    critic_loss     | 1.37     |
|    ent_coef        | 0.586    |
|    ent_coef_loss   | -5.38    |
|    learning_rate   | 0.0003   |
|    n_updates       | 1783     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | 0.0846   |
| time/              |          |
|    episodes        | 8        |
|    fps             | 45       |
|    time_elapsed    | 83       |
|    total_timesteps | 3768     |
| train/             |          |
|    actor_loss      | -37.7    |
|    critic_loss     | 0.35     |
|    ent_coef        | 0.333    |
|    ent_coef_loss   | -11      |
|    learning_rate   | 0.0003   |
|    n_updates       | 3667     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | 0.0698   |
| time/              |          |
|    episodes        | 12       |
|    fps             | 44       |
```

```
|    time_elapsed    | 126      |
|    total_timesteps | 5652     |
| train/             |          |
|    actor_loss      | -41.4    |
|    critic_loss     | 0.058    |
|    ent_coef        | 0.19     |
|    ent_coef_loss   | -16.6    |
|    learning_rate   | 0.0003   |
|    n_updates       | 5551     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | 0.0768   |
| time/              |          |
|    episodes        | 16       |
|    fps             | 44       |
|    time_elapsed    | 168      |
|    total_timesteps | 7536     |
| train/             |          |
|    actor_loss      | -41.4    |
|    critic_loss     | 0.172    |
|    ent_coef        | 0.108    |
|    ent_coef_loss   | -22      |
|    learning_rate   | 0.0003   |
|    n_updates       | 7435     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | 0.066    |
| time/              |          |
|    episodes        | 20       |
|    fps             | 44       |
|    time_elapsed    | 211      |
|    total_timesteps | 9420     |
| train/             |          |
|    actor_loss      | -38.5    |
|    critic_loss     | 1.08     |
|    ent_coef        | 0.0614   |
|    ent_coef_loss   | -27.5    |
```

```
|    learning_rate   | 0.0003   |
|    n_updates       | 9319     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | 0.0553   |
| time/              |          |
|    episodes        | 24       |
|    fps             | 44       |
|    time_elapsed    | 253      |
|    total_timesteps | 11304    |
| train/             |          |
|    actor_loss      | -36      |
|    critic_loss     | 1.05     |
|    ent_coef        | 0.035    |
|    ent_coef_loss   | -32.8    |
|    learning_rate   | 0.0003   |
|    n_updates       | 11203    |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | 0.0531   |
| time/              |          |
|    episodes        | 28       |
|    fps             | 44       |
|    time_elapsed    | 298      |
|    total_timesteps | 13188    |
| train/             |          |
|    actor_loss      | -33.4    |
|    critic_loss     | 0.0109   |
|    ent_coef        | 0.0199   |
|    ent_coef_loss   | -38.8    |
|    learning_rate   | 0.0003   |
|    n_updates       | 13087    |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | 0.0569   |
```

```
| time/              |          |
|    episodes        | 32       |
|    fps             | 43       |
|    time_elapsed    | 343      |
|    total_timesteps | 15072    |
| train/             |          |
|    actor_loss      | -30.1    |
|    critic_loss     | 0.188    |
|    ent_coef        | 0.0113   |
|    ent_coef_loss   | -43.5    |
|    learning_rate   | 0.0003   |
|    n_updates       | 14971    |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | 0.0667   |
| time/              |          |
|    episodes        | 36       |
|    fps             | 43       |
|    time_elapsed    | 385      |
|    total_timesteps | 16956    |
| train/             |          |
|    actor_loss      | -27.5    |
|    critic_loss     | 0.0116   |
|    ent_coef        | 0.00647  |
|    ent_coef_loss   | -48.2    |
|    learning_rate   | 0.0003   |
|    n_updates       | 16855    |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 471      |
|    ep_rew_mean     | 0.0788   |
| time/              |          |
|    episodes        | 40       |
|    fps             | 43       |
|    time_elapsed    | 429      |
|    total_timesteps | 18840    |
| train/             |          |
|    actor_loss      | -24.3    |
```

```
|    critic_loss     | 0.00214  |
|    ent_coef        | 0.00368  |
|    ent_coef_loss   | -52.8    |
|    learning_rate   | 0.0003   |
|    n_updates       | 18739    |
---------------------------------
Saved model to ./sb3_logs_wf\sac_window_1.zip
Saved VecNormalize to ./sb3_logs_wf\sac_vecnorm_window_1.pkl
Window 1 测试结束:
  终值资产: 11412.39  (初始 10124.11)
  总增长倍数: 1.1272
  年化收益:  37.40%
  年化波动:  12.15%
  Sharpe:    3.077
================================================================================
Window 2
Train: 2023-01-03 ~ 2025-01-02
Test : 2025-01-03 ~ 2025-07-02
Using cuda device
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:64: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="ffill", inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:65: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="bfill", inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:64: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="ffill", inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:65: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="bfill", inplace=True)
```

```
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 472      |
|    ep_rew_mean     | 0.12     |
| time/              |          |
|    episodes        | 4        |
|    fps             | 45       |
|    time_elapsed    | 41       |
|    total_timesteps | 1888     |
| train/             |          |
|    actor_loss      | -26.7    |
|    critic_loss     | 0.199    |
|    ent_coef        | 0.585    |
|    ent_coef_loss   | -5.42    |
|    learning_rate   | 0.0003   |
|    n_updates       | 1787     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 472      |
|    ep_rew_mean     | 0.139    |
| time/              |          |
|    episodes        | 8        |
|    fps             | 44       |
|    time_elapsed    | 84       |
|    total_timesteps | 3776     |
| train/             |          |
|    actor_loss      | -37.6    |
|    critic_loss     | 0.166    |
|    ent_coef        | 0.332    |
|    ent_coef_loss   | -11.1    |
|    learning_rate   | 0.0003   |
|    n_updates       | 3675     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 472      |
|    ep_rew_mean     | 0.13     |
| time/              |          |
|    episodes        | 12       |
|    fps             | 44       |
```

```
|    time_elapsed    | 127      |
|    total_timesteps | 5664     |
| train/             |          |
|    actor_loss      | -41.2    |
|    critic_loss     | 5.02     |
|    ent_coef        | 0.189    |
|    ent_coef_loss   | -16.5    |
|    learning_rate   | 0.0003   |
|    n_updates       | 5563     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 472      |
|    ep_rew_mean     | 0.133    |
| time/              |          |
|    episodes        | 16       |
|    fps             | 44       |
|    time_elapsed    | 170      |
|    total_timesteps | 7552     |
| train/             |          |
|    actor_loss      | -42.1    |
|    critic_loss     | 0.0771   |
|    ent_coef        | 0.107    |
|    ent_coef_loss   | -22.3    |
|    learning_rate   | 0.0003   |
|    n_updates       | 7451     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 472      |
|    ep_rew_mean     | 0.129    |
| time/              |          |
|    episodes        | 20       |
|    fps             | 44       |
|    time_elapsed    | 213      |
|    total_timesteps | 9440     |
| train/             |          |
|    actor_loss      | -39      |
|    critic_loss     | 0.0419   |
|    ent_coef        | 0.061    |
|    ent_coef_loss   | -27.9    |
```

```
|    learning_rate   | 0.0003   |
|    n_updates       | 9339     |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 472      |
|    ep_rew_mean     | 0.135    |
| time/              |          |
|    episodes        | 24       |
|    fps             | 44       |
|    time_elapsed    | 257      |
|    total_timesteps | 11328    |
| train/             |          |
|    actor_loss      | -36.2    |
|    critic_loss     | 0.68     |
|    ent_coef        | 0.0347   |
|    ent_coef_loss   | -32.6    |
|    learning_rate   | 0.0003   |
|    n_updates       | 11227    |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 472      |
|    ep_rew_mean     | 0.13     |
| time/              |          |
|    episodes        | 28       |
|    fps             | 44       |
|    time_elapsed    | 300      |
|    total_timesteps | 13216    |
| train/             |          |
|    actor_loss      | -33.3    |
|    critic_loss     | 0.0109   |
|    ent_coef        | 0.0197   |
|    ent_coef_loss   | -38.6    |
|    learning_rate   | 0.0003   |
|    n_updates       | 13115    |
---------------------------------
---------------------------------
| rollout/           |          |
|    ep_len_mean     | 472      |
|    ep_rew_mean     | 0.128    |
```

```
| time/              |          |
|    episodes        | 32       |
|    fps             | 43       |
|    time_elapsed    | 343      |
|    total_timesteps | 15104    |
| train/             |          |
|    actor_loss      | -30.4    |
|    critic_loss     | 0.158    |
|    ent_coef        | 0.0112   |
|    ent_coef_loss   | -44      |
|    learning_rate   | 0.0003   |
|    n_updates       | 15003    |
----------------------------------
----------------------------------
| rollout/           |          |
|    ep_len_mean     | 472      |
|    ep_rew_mean     | 0.128    |
| time/              |          |
|    episodes        | 36       |
|    fps             | 43       |
|    time_elapsed    | 387      |
|    total_timesteps | 16992    |
| train/             |          |
|    actor_loss      | -28      |
|    critic_loss     | 0.0797   |
|    ent_coef        | 0.00636  |
|    ent_coef_loss   | -50.1    |
|    learning_rate   | 0.0003   |
|    n_updates       | 16891    |
----------------------------------
----------------------------------
| rollout/           |          |
|    ep_len_mean     | 472      |
|    ep_rew_mean     | 0.136    |
| time/              |          |
|    episodes        | 40       |
|    fps             | 43       |
|    time_elapsed    | 433      |
|    total_timesteps | 18880    |
| train/             |          |
|    actor_loss      | -24.7    |
```

```
|    critic_loss     | 0.00266  |
|    ent_coef        | 0.00362  |
|    ent_coef_loss   | -54.9    |
|    learning_rate   | 0.0003   |
|    n_updates       | 18779    |
---------------------------------
Saved model to ./sb3_logs_wf\sac_window_2.zip
Saved VecNormalize to ./sb3_logs_wf\sac_vecnorm_window_2.pkl
Window 2 测试结束:
  终值资产: 10603.74  (初始 10025.39)
  总增长倍数: 1.0577
  年化收益:  16.80%
  年化波动:  24.45%
  Sharpe:    0.687
```

```
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:64: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="ffill", inplace=True)
C:\Users\HP\AppData\Local\Temp\ipykernel_9704\1757598842.py:65: FutureWarning: DataFrame.fillna with 'method' is deprecated and
will raise in a future version. Use obj.ffill() or obj.bfill() instead.
  feature_data.fillna(method="bfill", inplace=True)
```

Out[31]:

w_id	train_start	train_end	test_start	test_end	test_steps	start_wealth	end_wealth	total_growth	ann_return	ann_vol	sha
0	2022-01-03	2024-01-02	2024-01-03	2024-07-02	93	9971.897342	12143.969822	1.217819	0.705697	0.112133	6.293
1	2022-07-03	2024-07-02	2024-07-03	2025-01-02	95	10124.113547	11412.386682	1.127248	0.374006	0.121533	3.077
2	2023-01-03	2025-01-02	2025-01-03	2025-07-02	91	10025.393147	10603.735845	1.057688	0.168023	0.244493	0.687

修改部分

添加了科技股票、国债、标普500和黄金等数据作为环境输入特征

收益严重偏大，进行修改：

=== Portfolio Performance (deterministic rollout) === Periods per year (inferred): 252.00 Start wealth: 10297.47 End wealth: 23731074.08 Total growth: 2304.553868 CAGR: 804.485651% Ann. Return (geom from equity): 804.485651% Ann. Vol: 22.710210% Sharpe (rf=0): 9.8502 Max Drawdown: -5.59% Calmar: 143.9561

1、对vec_env进行了归一化处理

=== Portfolio Performance (deterministic rollout) === Periods per year (inferred): 252.00 Start wealth: 10180.25 End wealth: 1013360.39 Total growth: 99.541782 CAGR: 270.067928% Ann. Return (geom from equity): 270.067928% Ann. Vol: 22.074986% Sharpe (rf=0): 6.0514 Max Drawdown: -8.50% Calmar: 31.7818

2、调整奖励函数

用协方差矩阵估计组合波动 添加每次交易成本千一

=== Portfolio Performance (deterministic rollout) === Periods per year (inferred): 252.00 Start wealth: 10055.45 End wealth: 54941.09 Total growth: 5.463811 CAGR: 62.092061% Ann. Return (geom from equity): 62.092061% Ann. Vol: 17.712724% Sharpe (rf=0): 2.8174 Max Drawdown: -11.18% Calmar: 5.5563

3、更换为sac算法

=== Portfolio Performance (deterministic rollout) === Periods per year (inferred): 252.00 Start wealth: 10145.52 End wealth: 22875.43 Total growth: 2.254731 CAGR: 26.016893% Ann. Return (geom from equity): 26.016893% Ann. Vol: 14.523819% Sharpe (rf=0): 1.6653 Max Drawdown: -12.48% Calmar: 2.0840

删除了step函数中对动作的softmax处理

=== Portfolio Performance (deterministic rollout) === Periods per year (inferred): 252.00 Start wealth: 10096.53 End wealth: 36203.07 Total growth: 3.585696 CAGR: 43.791829% Ann. Return (geom from equity): 43.791829% Ann. Vol: 12.005115% Sharpe (rf=0): 3.0874 Max Drawdown: -8.52% Calmar: 5.1418

优化奖励函数权重

benchmark:

lambda_risk = 1 lambda_turnover = 0.001 reward = base_reward - lambda_risk * risk_penalty - lambda_turnover * cost

=== Portfolio Performance (deterministic rollout) === Periods per year (inferred): 252.00 Start wealth: 10164.10 End wealth: 39643.65 Total growth: 3.900359 CAGR: 47.273482% Ann. Return (geom from equity): 47.273482% Ann. Vol: 17.140595% Sharpe (rf=0): 2.3455 Max Drawdown: -13.67% Calmar: 3.4585