Compound Reinforcement Learning:
Theory and An Application to Finance

Tohgoroh Matsui[1], Takashi Goto[2], Kiyoshi Izumi[3,4], and Yu Chen[3]

[1]Chubu University,
1200 Matsumoto-cho, Kasugai, 487-8501 Aichi, Japan
TohgorohMatsui@tohgoroh.jp, http://tohgoroh.jp

[2]Bank of Tokyo-Mitsubishi UFJ, Ltd.
2-7-1 Marunouchi, Chiyoda, 100-8388 Tokyo, JAPAN
takashi_6_gotou@mufg.jp

[3]The University of Tokyo
7-3-1 Hongo, Bunkyo, 113-8656 Tokyo, JAPAN
{izumi@sys.t.,chen@k}.u-tokyo.ac.jp

【强化学习应用 70】Finance 1

张楚珩
清华大学 交叉信息院博士在读

继续强化学习在金融上的应用调研。

原文传送门

Matsui, Tohgoroh, et al. "Compound reinforcement learning: Theory and an application to finance." European Workshop on Reinforcement Learning. Springer, Berlin, Heidelberg, 2011.

Moody, John, and Matthew Saffell. "Learning to trade via direct reinforcement." IEEE transactions on neural Networks 12.4 (2001): 875-889.

Deng, Yue, et al. "Deep direct reinforcement learning for financial signal representation and trading." IEEE transactions on neural networks and learning systems 28.3 (2016): 653-664.

特色

下午调研了三篇 RL 在 trading 上应用的文章。其中第一篇观察到金融里面总收益其实是每个时间段收益的乘积，而不是加和，基于该观察提出了相应的奖励函数以及相应的算法 compound Q-learning。该做法确实有一定的创新，但是个人直观感受上，实际中该项改进可能效果不会很明显。我调研的主要动机就是感觉到，在金融领域智能体的行动对于市场状态没有明显的影响，这样不同时刻上的行动决策可能相互没有太强的联系，在此情况下强化学习可能用处不大。第二篇和第三篇文章都考虑了手续费这一因素，使得强化学习的使用有一定的合理性，不过个人感觉仅仅由于手续费这一因素就放弃更为稳定的有监督学习方法而使用强化学习方法，还是不够有说服力。因此，就泛泛读了这几篇，稍微总结一下。

现阶段调 表述把多少比例的钱投入到该股票上，即只有一个投资标的；之后会调研 portfolio management，即有多个投资标的。

过程

1. Compound RL（2011）

第一篇文章观察到金融里面总收益是每个时间段收益的连乘，而在普通强化学习里面通常认为总的

收益是各步产生奖励的连加。当然这其中为了避免发散，还是会加上 discount rate。这篇文章就研究如果收益是连乘关系时相应的强化学习奖励应该如何表述，相应的算法应该怎么变化。

容易想到，连乘在对数空间上就是连加，因此总收益的对数可以写作

$$\begin{aligned}\log \rho_t &= \log \prod_{k=0}^{\infty} \left(1 + R_{t+k+1} f\right)^{\gamma^k} \\ &= \sum_{k=0}^{\infty} \log \left(1 + R_{t+k+1} f\right)^{\gamma^k} \\ &= \sum_{k=0}^{\infty} \gamma^k \log \left(1 + R_{t+k+1} f\right). \end{aligned} \tag{8}$$

其中 f 表述把多少比例的钱投入到该股票上， R_t 表示 t 时间段上的收益， γ 是discount rate，金融上可以看做是对于风险的调整（尽早拿到收益所暴露的风险更少）。

相应地，得到 compound Q-learning。

Algorithm 1 Compound Q-Learning

Input: discount rate γ, step size α, bet fraction f
Initialize $Q(s, a)$ arbitrarily, for all s, a
loop {for each episode}
  Initialize s
  repeat {for each step of episode}
    Choose a from s using policy derived from Q (e.g., ϵ-greedy)
    Take action a, observe return R, next state s'
    $Q(s,a) \leftarrow Q(s,a) + \alpha \left[\log(1 + Rf) + \gamma \max_{a'} Q(s', a') - Q(s,a)\right]$
    $s \leftarrow s'$
  until s is terminal
end loop

2. Direct Reinforcement （2001）

这一篇里面讲的 direct reinforcement 其实就是 policy-based RL，相应地文章对比了 value-based RL，后者更像是先预测再决策，而前者更接近直接不预测得到最优策略。同样，在本文中，使用 RL 的原因仍然是我们需要尽量减少手续费。

本文使用的状态主要是价格时间序列，策略是线性回归策略

$$F_t = F(\theta_t; F_{t-1}, I_t) \quad \text{with}$$
$$I_t = \{z_t, z_{t-1}, z_{t-2}, \ldots; y_t, y_{t-1}, y_{t-2}, \ldots\} \qquad (1)$$

其中 z 表示价格序列， y 表示外部变量（具体是什么没找到）。策略直接输出行动，行动为离散的两种情况，分别是 long position 和 short position。

$$F_t = \text{sign}(uF_{t-1} + v_0 r_t + v_1 r_{t-1} + \ldots + v_m r_{t-m} + w) \quad (2)$$

单步收益会考虑手续费，表示为

$$R_t = \mu\{F_{t-1} r_t - \delta|F_t - F_{t-1}|\}. \qquad (6)$$

优化的目标为估计出来的 Sharpe ratio 和 Sterling ratio，但是需要把这些全局的量化目标分散到每一步作为奖励，文章对此作了一些计算，得到了相应分散到每一步的奖励。

对于 Sharpe ratio，

$$D_t \equiv \frac{dS_t}{d\eta} = \frac{B_{t-1}\Delta A_t - \frac{1}{2}A_{t-1}\Delta B_t}{(B_{t-1} - A_{t-1}^2)^{3/2}}. \qquad (14)$$

对于 Sterling ratio，

$$\begin{aligned} D_t &\equiv \frac{d\text{DDR}_t}{d\eta} \\ &= \frac{R_t - \frac{1}{2}A_{t-1}}{\text{DD}_{t-1}}, \qquad R_t > 0 \qquad (23) \\ &= \frac{\text{DD}_{t-1}^2 \cdot (R_t - \frac{1}{2}A_{t-1}) - \frac{1}{2}A_{t-1}R_t^2}{\text{DD}_{t-1}^3}, \qquad R_t \le 0. \end{aligned}$$

(24)

其中 A,B 分别表示滚动估计的关于收益的一阶矩和二阶矩。

$$A_t = A_{t-1} + \eta \Delta A_t = A_{t-1} + \eta(R_t - A_{t-1})$$
$$B_t = B_{t-1} + \eta \Delta B_t = B_{t-1} + \eta(R_t^2 - B_{t-1}). \quad (15)$$

$$\mathrm{DD}_T = \left(\frac{1}{T} \sum_{t=1}^{T} \min\{R_t, 0\}^2 \right)^{1/2}. \quad (19)$$

接下来推导出相应的策略梯度就可以更新策略里面的参数了。

3. Deep direct RL（2016）

这一篇文章问了说明用 RL 的必要性，也是加上了手续费这一点。其特色是增加了 representation learning 这一个环节。对本文最为概括的是如下优化问题的表述。

$$\begin{aligned} &\max_{\{\Theta, g_d(\cdot), v(\cdot)\}} U_T(R_1..R_T) \\ &\text{s.t. } R_t = \delta_{t-1} z_t - c|\delta_t - \delta_{t-1}| \\ &\quad \delta_t = \tanh(\langle \mathbf{w}, \mathbf{F}_t \rangle + b + u\delta_{t-1}) \\ &\quad \mathbf{F}_t = g_d(v(\mathbf{f}_t)) \end{aligned} \quad (8)$$

其中 R_t 表示每个时间片上的收益，该收益不仅考虑到了价格的变动 z_t，也考虑到了相邻两个时间片上仓位的变化产生的手续费 $c|\delta_t - \delta_{t-1}|$，其中 c 为手续费率，δ 表示仓位。U 表示最后的评价标准，比如夏普比。每个时间片上的仓位是通过倒数第二行的式子产生的，其中 $\Theta = \{\mathbf{w}, b, u\}$ 表示策略的参数。状态的表示由两层结构产生，一层为模糊表示 $v(\cdot)$，另一层为神经网络产生的表示 $g_d(\cdot)$。示意图如下所示。

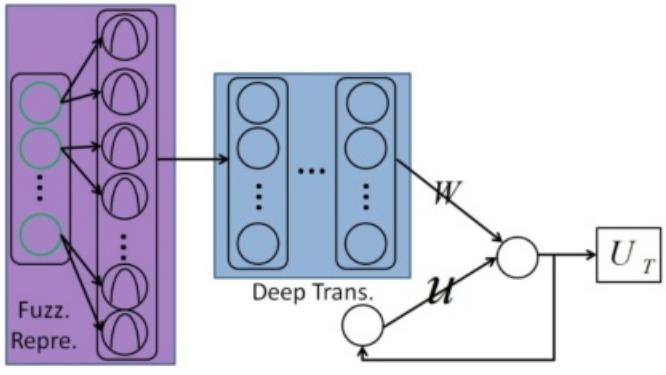

Fig. 2. Overview of fuzzy DRNNs for robust feature learning and self-taught trading.

其中模糊表示 $v_{(\cdot)}$，是先把数据进行聚类（可能是人工分类），分成三类（图中紫色区域中显示的是分为两类的情形），分别表示上涨趋势、下跌趋势、没有大趋势，然后计算出来这些趋势段中各个特征分量的均值和方差。每个分量都产生相应的三个分量，分别代表该特征在这三类中的概率。具体形式为

$$o_i^{(l)} = v_i\left(a_i^{(l)}\right) = e^{-\left(a_i^{(l)} - m_i\right)^2 / \sigma_i^2} \quad \forall i. \tag{7}$$

其中 m_i, σ_i 为对应的均值和方差。

另外本文对于参数初始化也做了特别的处理，比如蓝色区域的参数初始化是使用 autoencoder 预训练得到的参数。

发布于 2019-06-11

强化学习 (Reinforcement Learning)　金融　量化交易

文章被以下专栏收录