1 UCB iteration

(Note: I reverse the notations $Q(s,a)$ and $\widehat{Q}(s,a)$ in Page 4 of "Dong K, Wang Y, Chen X, et al. Q-learning with ucb exploration is sample efficient for infinite-horizon mdp[J]. arXiv preprint arXiv:1901.09311, 2019.")

for t=1,2,... do:

Take action $a_t \leftarrow \arg\max_a Q(s_t, a)$;

Receive reward r_t and transit to s_{t+1};

$N(s_t, a_t) \leftarrow N(s_t, a_t) + 1$; (note: $N(s_t, a_t)$ is the state_action_counter in codes);

$k \leftarrow N(s_t, a_t)$, $b_k \leftarrow f(k)$; (note: $f(k) = \sqrt{\frac{C_1 \log(k+1) + C_0}{k}}$ is the bonus, C_1, C_0 are the tuning parameters bonus_coef_1,bonus_coef_0 in codes)

$\widehat{Q}(s_t, a_t) \leftarrow \widehat{Q}(s_t, a_t) + \alpha_k[(r_t + b_k + \max_a Q(s_t, a)) - \widehat{Q}(s_t, a_t)]$; (note: $\alpha_k = \frac{H+1}{H+k}$ is the learning_rate in codes)

$Q(s_t, a_t) \leftarrow \min(Q(s_t, a_t), \widehat{Q}(s_t, a_t))$; (note: $Q(s_t, a_t), \widehat{Q}(s_t, a_t)$ are Q_table, Q_hat_table in codes)

1.1 Summary of tuning parameter in above iteration

C_1, C_0 in bonus; H in learning rate

2 index of state space for python codes

2.1 transition prob matrix

Note that

$$P_X(\Delta) = I + \Delta Q$$

where

$$Q\_\text{i1\_i} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1\ (\lambda_{2,1}) & -1 & 0 & 0 & 0 \\ 0 & 1 & ... & ... & 0 \\ 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 1 & -1 \end{bmatrix} \text{ and } Q\_\text{i\_i1} = \begin{bmatrix} -1 & 1\ (\lambda_{1,2}) & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & ... & ... & 0 \\ 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

P_X is equal to

$$\begin{bmatrix} 1-\lambda_{1,2}\Delta & \lambda_{1,2}\Delta & 0 & 0 & 0 & \\ \lambda_{2,1}\Delta & 1-(\lambda_{2,1}+\lambda_{2,3})\Delta & \lambda_{2,3}\Delta & 0 & 0 & \\ 0 & ... & ... & ... & 0 & \\ 0 & 0 & ... & ... & ... & \\ 0 & 0 & 0 & \lambda_{|\mathcal{S}_X|-1,|\mathcal{S}_X|-2}\Delta & 1-(\lambda_{|\mathcal{S}_X|-1,|\mathcal{S}_X|-2}+\lambda_{|\mathcal{S}_X|-1,|\mathcal{S}_X|})\Delta & \lambda_{|\mathcal{S}_X} \\ 0 & 0 & 0 & 0 & \lambda_{|\mathcal{S}_X|,|\mathcal{S}_X|-1}\Delta & 1-\lambda_{|\mathcal{S}} \end{bmatrix}$$

$$\{\lambda_{i,i+1}\}_{i=1,2,...,|\mathcal{S}_X|-1} \text{ and } \{\lambda_{i+1,i}\}_{i=2,3,...,|\mathcal{S}_X|-1,|\mathcal{S}_X|}$$

2.2 index of state space

state variables:

x is an integer in $\mathcal{S}_X/(\frac{\delta}{2}) := \{1, 2, ..., (2N_P - 1)\} = \{1, 2, ...,\text{dim\_X}\} = \{0, 1, ...,\text{dim\_X}-1\} + 1$, and dim_X=$|\mathcal{S}_X|$.

so "python index" idx_x for Q_table equals x-1

y is an integer in $\mathcal{S}_Y := \{-N_Y, -(N_Y - 1), ... - 2, -1, 0, 1, 2, ..., (N_Y - 1), N_Y\} = \{0, 1, ...,\text{dim\_Y}-1\} - N_Y$

so "python index" idx_y for Q_table equals y+N_Y

Here, the set starting from 0 can always be used as python index set.

For quoted price: p_a, p_b are integers in $\mathcal{S}_X/\delta := \{0, 1, 2, ..., N_P\}$ no need to modify because it has already been index set starting from 0. However, since we add an action "doing nothing" in the state space, we make the action state be (p_a, p_b)$\in \mathcal{S}_X^+/\delta \times \mathcal{S}_X^+/\delta$, where

$$\mathcal{S}_X^+/\delta = \{0, 1, 2, ..., N_P, N_P + 1\} \text{ and } N_P + 1 \text{ means "doing nothing".}$$

So, when we specify the available action space based on the state variable (x,y), we need to do:

p_a_action_list is in $\mathcal{S}_X/\delta > x/2$, for example:

if $x = (2k)$ which means $X_i = x = 2k\frac{\delta}{2} = k\delta$, then $\{\mathcal{S}_X/\delta > x/2\} = \{\mathcal{S}_X/\delta > k\delta\} = \{(k + 1)\delta, (k + 2)\delta, ..., (2N_P - 1)\delta\}$.

if $x = (2k + 1)$ which means $X_i = x = (2k + 1)\frac{\delta}{2} = (k + \frac{1}{2})\delta$, then $\{\mathcal{S}_X/\delta > x/2\} = \left\{\mathcal{S}_X/\delta > (k + \frac{1}{2})\delta\right\} = \{(k + 1)\delta, (k + 2)\delta, ..., (2N_P - 1)\delta\}$.

2.3 iteration of the true value function satisfying the Bellman equation

(1) the expectation of the reward at step i is p_ask_fill * (-x*tick/2+p_a*tick) + p_buy_fill * (x*tick/2-p_b*tick) is

$$E_{s' \sim P(s,a)}[r(s', s, a)]$$

(2) We have that, np.dot(vec_prob_X.T ,np.dot(V_star, vec_prob_Y)) is

$$\sum_{i,j} p_i^X p_j^Y V_{i,j}^* = E_{s' \sim P(s,a)}[V^*(s')]$$

where vec_prob_X= $(p_i^X)_i$, vec_prob_Y= $(p_j^Y)_j$ captures $s' \sim P(s, a)$ because the state variable $s = (x, y)$.