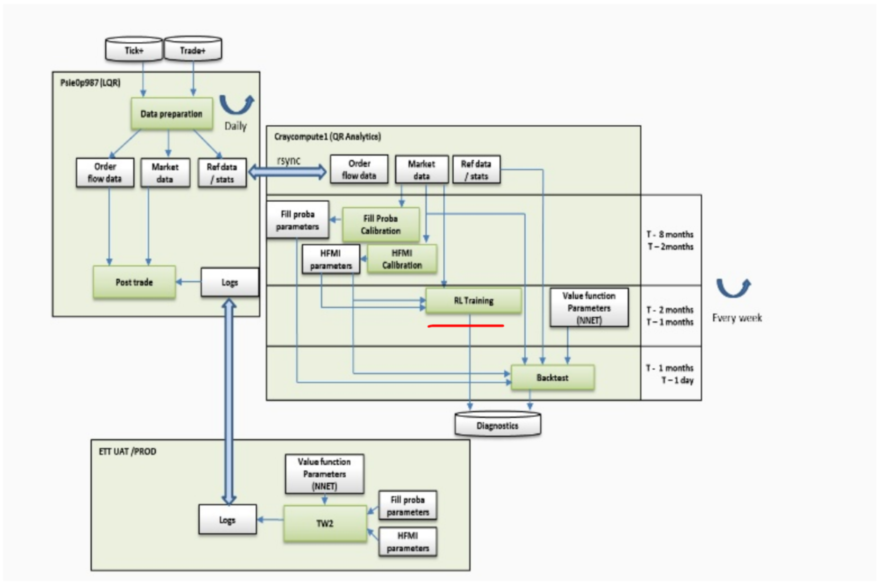
Aug 1, 2017 · 3 min read

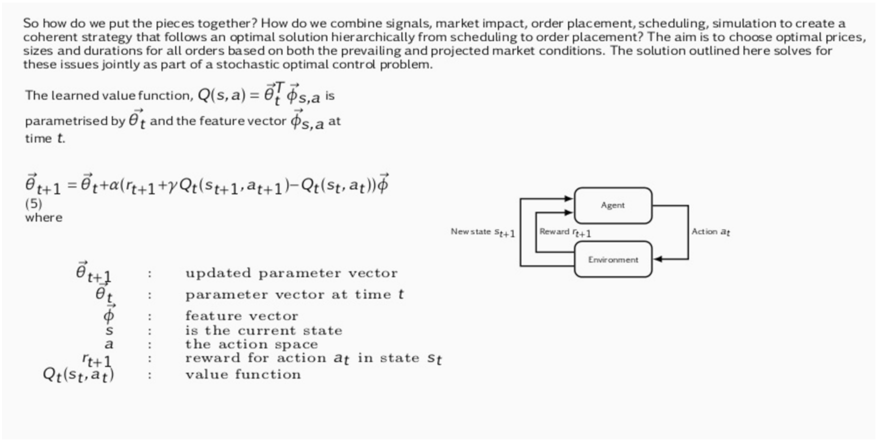
Deep Reinforcement Learning Based Trading Application at JP Morgan Chase

FT released a story today about the new application that will optimize JP Morgan Chase trade execution (Business Insider article on the same topic for readers that do not have FT subscription). The intent is to reduce market impact and provide best trade execution results for large orders.

It is a complex application with many moving parts:



Its core is an RL algorithm that learns to perform the best **action** (choose optimal price, duration and order size) based on market conditions. It is not clear if it is Sarsa (On-Policy TD Control) or Q-learning (Off-Policy Temporal Difference Control Algorithm) as both algorithms are present in JP Morgan slides:



So how do we put the pieces together? How do we combine signals, market impact, order placement, scheduling, simulation to create a coherent strategy that follows an optimal solution hierarchically from scheduling to order placement? The aim is to choose optimal prices, sizes and durations for all orders based on both the prevailing and projected market conditions. The solution outlined here solves for these issues jointly as part of a stochastic optimal control problem.

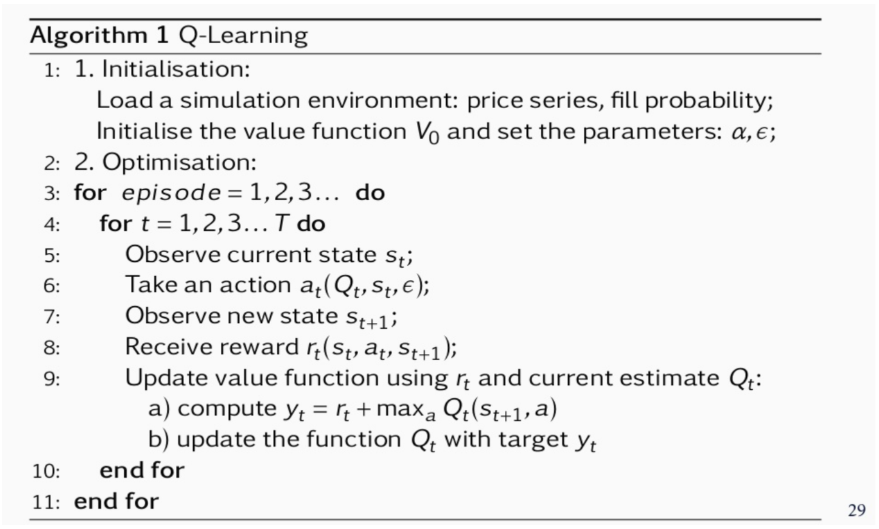
The learned value function, $Q(s,a) = \vec{\theta}_t^T \vec{\phi}_{s,a}$ is parametrised by $\vec{\theta}_t$ and the feature vector $\vec{\phi}_{s,a}$ at time t.

$$\vec{\theta}_{t+1} = \vec{\theta}_t + \alpha(r_{t+1} + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t))\vec{\phi} \quad (5)$$

where

$\vec{\theta}_{t+1}$	:	updated parameter vector
$\vec{\theta}_t$	:	parameter vector at time t
$\vec{\phi}$	:	feature vector
s	:	is the current state
a	:	the action space
r_{t+1}	:	reward for action a_t in state s_t
$Q_t(s_t, a_t)$	:	value function

Sarsa

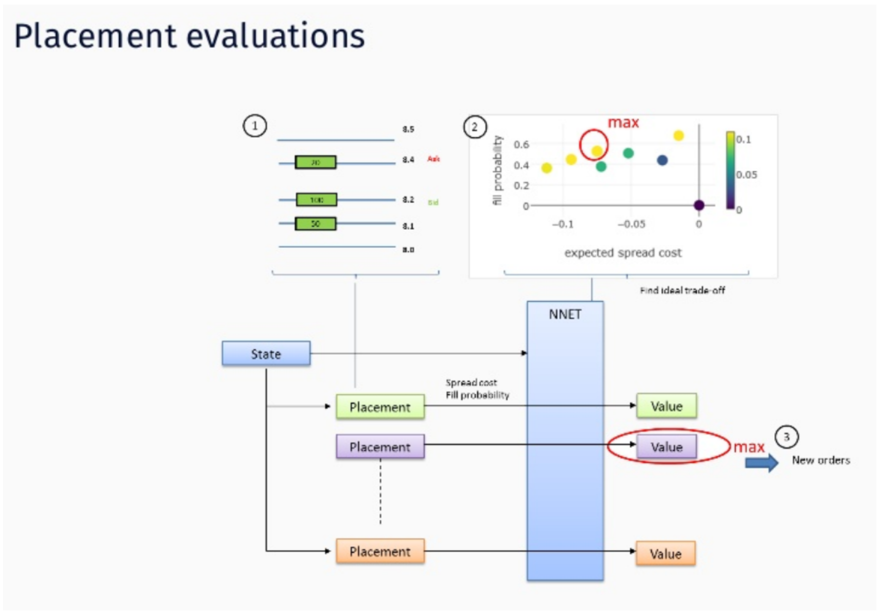


```
Algorithm 1 Q-Learning
1:  1. Initialisation:
       Load a simulation environment: price series, fill probability;
       Initialise the value function $V_0$ and set the parameters: $\alpha, \epsilon$;
2:  2. Optimisation:
3:  for $episode = 1, 2, 3 \ldots$ do
4:      for $t = 1, 2, 3 \ldots T$ do
5:          Observe current state $s_t$;
6:          Take an action $a_t(Q_t, s_t, \epsilon)$;
7:          Observe new state $s_{t+1}$;
8:          Receive reward $r_t(s_t, a_t, s_{t+1})$;
9:          Update value function using $r_t$ and current estimate $Q_t$:
               a) compute $y_t = r_t + \max_a Q_t(s_{t+1}, a)$
               b) update the function $Q_t$ with target $y_t$
10:     end for
11: end for
```

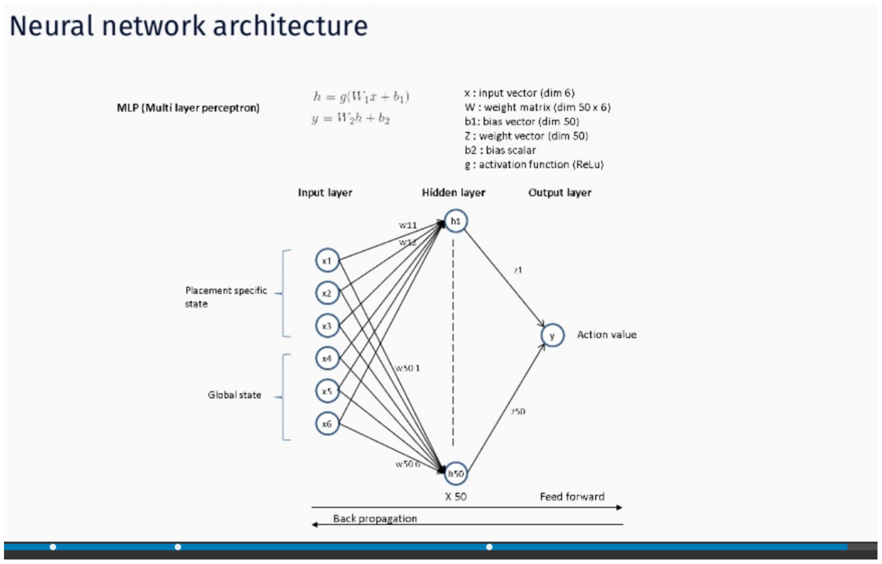
29

Q-learning

State consists of price series, expected spread cost, fill probability, size placed, as well as elapsed time, %progress, etc. **Rewards** are immediate rewards (price spread) and terminal (end of episode) rewards like completion, order duration and market penalties (obviously those are negative rewards that punish the agent along these dimensions).



Actions are memorized as weights of a Deep Neural Network—function approximation via NN is used since state, action space is too big to be handled in tabular form. We assume stochastic gradient descent is used for both feed forward and backprop operation operation (hence Deep designation):



JP Morgan is convinced this is the very first real time trading AI/ML application on Wall Street. We are assuming this is not true i.e. there are surely other players operating in this space as RL implementation to order execution is known for quite a while now (Kearns and Nevmyvaka 2006).

The latest LOXM developments will be presented at QuantMinds Conference in Lisbon (May of 2018).

Instinet is also using Q-learning, probably for the same purpose (market impact reduction).