Algorithmic Trading: Reinforcement Learning

Sebastian Jaimungal
University of Toronto

many thanks to
Álvaro Cartea, Oxford

Apr, 2017
Reinforcement Learning – Intro
Reinforcement Learning – Intro

- **Reinforcement learning** is unsupervised – based only on the rewards from actions & how the system reacts
- As in continuous time stochastic control, the actions affect the reward and the system
- Can be **model-free**
- Goal is to **maximize performance criteria**

\[
H^a(s) = \mathbb{E} \left[ \sum_{k=0}^{\infty} \gamma^k R(a_k; S_k^a, S_{k+1}^a) \middle| S_t^a = s \right]
\]

- \( S_t \in S \) is the **state** of the system at time \( t \)
- \( a \in A \) an admissible set of actions which depend only on the state of the system
- The system evolves in an action dependent manner:
  \( S_{t+1}^a \sim F(S_t; a_t) \)
Reinforcement Learning – Intro

Figure: Directed graphical representation. When $Y_t = X_t$ the environment is fully observed.
Reinforcement Learning – Intro

- Reinforcement learning aims to discover the best policy by:
  - **Exploration** – trying an action, and see what the response is, update action choices (learn from the environment)
  - **Exploitation** – use what you already know to make the best action choice

- Both are important!
A Bellman principle applies and one can show that $H = H^{a^*}$ satisfies

$$H(s) = \max_{a \in A} \mathbb{E} \left[ R(a; s, S_{1}^{s,a}) + \gamma H(S_{1}^{s,a}) \right]$$

With known transition probability of states, this can be applied recursively to find $H$ and hence $a^*$

$$H(s) \leftarrow \max_{a \in A} \sum_{s'} \left[ P_{ss'}^{a} \left\{ R(a; s, s') + \gamma H(s') \right\} \right]$$

- Make an initial assumption on $H$ – e.g., zeros
- Iterate until “converged”
- Choose actions which maximize the expression
Q-learning
Q-learning

- **Q-learning** is an “off-policy” learning.

  [NB: “on-policy” means the algorithm estimates the value function of the policy which generates the data.]

- Define

\[
Q(s, a) = \mathbb{E} \left[ R(a; s, S_1^{s,a}) + \gamma H(S_1^{s,a}) \right] \\
= \mathbb{E} \left[ R(a; s, S_1^{s,a}) + \gamma \max_{a' \in A} Q(S_1^{s,a}, a') \right]
\]

because \( H(s) = \max_{a \in A} Q(s, a) \)
Q-learning

- We wish to approximate the expectation from actual observations while you learn...

- Algorithm is $\varepsilon$-greedy: at iteration $k$
  - Select a random action $a_k$ with probability $\varepsilon_k$ (Explore)
  - Otherwise select the current best policy (Exploit)

\[
a^*(s) = \arg \max_{a \in A} Q(s, a)
\]
Q-learning

1. Initialize $Q(s, a)$ (random or often just to zero)
2. Repeat (for every run)
3. initialize state $s$
4. Repeat (for each step in the run)
5. Select $\varepsilon$-greedy action $a$
6. Take action $a$, observe $s'$ & reward $R$
7. update $Q$ according to

$$Q(s, a) \leftarrow (1 - \alpha_k) Q(s, a) + \alpha_k \left[ R + \gamma \max_{a' \in A} Q(S', a') \right]$$

8. update $s \leftarrow s'$
9. goto 5 until run is done
10. goto 3 until all runs are done
Q-learning

- Require decreasing $\alpha_k \to 0 \text{ s.t.}$

$$\sum_k \alpha_k = +\infty \quad \text{and} \quad \sum_k \alpha_k^2 < +\infty$$

- Often

$$\varepsilon_k = \frac{A}{B + k} \quad \text{and} \quad \alpha_k = \frac{C}{D + k}$$
Q-learning

- The updating rule is akin to using updating to estimate the mean $\mu = \mathbb{E}[X]$ of a r.v. $X$, from its samples $x = \{x_1, \ldots, x_n\}$

$$\mu_k = \frac{1}{k} \sum_{i=1}^{k} x_i = \mu_{k-1} + \frac{1}{k}(x_k - \mu_{k-1})$$
Q-learning convergence
Q-learning convergence

- Here, we look at why Q-learning converges to the optimal solution.

- Note first that the operator $\mathcal{B}$ which acts as follows

$$ (\mathcal{B}Q)(s, a) = \mathbb{E} \left[ R(a; s, S_1^{s,a}) + \gamma \max_{a' \in \mathcal{A}} Q(S_1^{s,a}, a') \right] $$

is a contraction operator in the $L_\infty$-norm, i.e.

$$ \| \mathcal{B}Q_1 - \mathcal{B}Q_2 \|_\infty \leq \gamma \| Q_1 - Q_2 \|_\infty $$
Q-learning convergence

Proof:

\[ \| \mathcal{B} Q_1 - \mathcal{B} Q_2 \|_\infty = \gamma \max_{s, a} \mathbb{E} \left[ \max_{a' \in \mathcal{A}} Q_1(S_1^{s, a}, a') - \max_{a' \in \mathcal{A}} Q_2(S_1^{s, a}, a') \right] \]

\[ \leq \gamma \max_{s, a} \mathbb{E} \left[ \max_{a' \in \mathcal{A}} Q_1(S_1^{s, a}, a') - \max_{a' \in \mathcal{A}} Q_2(S_1^{s, a}, a') \right] \]

\[ \leq \gamma \max_{s, a} \mathbb{E} \left[ \max_{s' \in S; a' \in \mathcal{A}} \left| Q_1(s', a') - Q_2(s', a') \right| \right] \]

\[ = \gamma \| Q_1 - Q_2 \|_\infty \]

Hence, there is a chance the procedure converges... but we need more...
Q-learning convergence

To illustrate that **Q-learning converges to the optimal** we will need a general **stochastic approximation** result...

**Theorem**

An iterative process

\[
\zeta_{k+1}(x) = (1 - \alpha_k(x)) \zeta_k(x) + \beta_k(x) F_k(x)
\]

converges to zero a.s. under the following assumptions

1. \( \sum_k \alpha = \infty, \sum_k \alpha_k^2 < \infty, \sum_k \beta_k^2 < \infty \) and \( E[\beta_k(x) \mid \mathcal{F}_k] \leq E[\alpha_k(x) \mid \mathcal{F}_k] \) *uniformly* a.s.

2. \( \|E[F_k(x) \mid \mathcal{F}_k, \beta_k]\|_W \leq \delta \|\zeta_k\|_W \), for some \( \delta \in (0, 1) \)

3. \( \nabla[F_k(x) \mid \mathcal{F}_k, \beta_k] \leq C(1 + \|\zeta_k\|_W)^2 \)

*Here, \( \| \cdot \|_W \) denotes a weighted norm.*
Next, set
\[ \Delta_k(s, a) = Q_k(s, a) - Q^*(s, a) \]
where \( Q_k \) is the \( k \)-th iteration, i.e., \( Q_k = \bar{B}^k Q_0 \) and
\[ \bar{B} Q_k = (1 - \alpha_k) Q_k(s, a) + \alpha_k \left[ R_k + \gamma \max_{a' \in A} Q_k(S', a') \right] \]
Q-learning convergence

For Q-learning, by definition, we then have

\[ \Delta_k(s_k, a_k) = (1 - \alpha_k) Q_k(s_k, a_k) + \alpha_k \left[ R_k + \gamma \max_{a' \in A} Q_k(s_{k+1}^{a_k, s_k}, a') - Q^*(s_k, a_k) \right] \]

\[ \psi_k(s, a) \]
Q-learning convergence

Writing

\[ \Psi_k(s, a) := R_k^{a,s} + \gamma \max_{a' \in A} Q_k(s_{k+1}^{a,s}, a') - Q^*(s, a) \]

then,

\[ \mathbb{E}[\Psi_k(s, a) \mid \mathcal{F}_k] = (\mathcal{B}Q_k)(s, a) - Q^*(s, a) \]

and since \( Q^* \) is a fixed point of \( \mathcal{B} \),

\[ \mathbb{E}[\Psi_k(s, a) \mid \mathcal{F}_k] = (\mathcal{B}Q_k - \mathcal{B}Q^*)(s, a) \]

so that

\[ \left\| \mathbb{E}[\Psi_k(s, a) \mid \mathcal{F}_k] \right\|_{\infty} \leq \gamma \left\| Q_k - Q^* \right\|_{\infty} \]

so part 2) of the general SA result holds
Q-learning convergence

Next, we need the variance to be bounded...

\[ \mathbb{V}[\psi_k(s, a) \mid \mathcal{F}_k] \]

\[ = \mathbb{V} \left[ R_k^{a,s} + \gamma \max_{a' \in A} Q_k(s_{k+1}^{a,s}, a') \middle| \mathcal{F}_k \right] \]

\[ = \mathbb{V} \left[ R_k^{a,s} \mid \mathcal{F}_k \right] + 2 C \left[ R_k^{a,s} \max_{a' \in A} Q_k(s_{k+1}^{a,s}, a') \mid \mathcal{F}_k \right] \]

\[ + \mathbb{V} \left[ \max_{a' \in A} Q_k(s_{k+1}^{a,s}, a') \mid \mathcal{F}_k \right] \]

\[ \leq C(1 + \|\zeta_k\|_W^2) \]

under the assumption of bounded rewards, the variance constraint holds
Dyna-Q Learning
Dyna-Q Learning

- Idea is to **combine experience and model**
  - Update $Q$ from experience ($\epsilon$-greedy)
  - Learn a model from experience
  - Simulate from model, and update $Q$
Dyna-Q Learning

1. Initialize $Q(s, a)$ (random or often just to zero)
2. initialize state $s$
3. Select $\varepsilon$-greedy action $a$ from $Q$
4. Take action $a$, observe $s'$ & reward $R$
5. update $Q$ according to
   \[ Q(s, a) \leftarrow (1 - \alpha_k) Q(s, a) + \alpha_k \left[ R + \gamma \max_{a' \in A} Q(S', a') \right] \]
6. Update Model: $(s, a) \mapsto (s', r)$. Repeat $n$ times
   - randomly select $\tilde{s}$ from previously visited
   - randomly select $\tilde{a}$ from previous actions at state $\tilde{s}$
   - Use model to get $(\tilde{s}, \tilde{a}) \mapsto (\tilde{s}', \tilde{r})$
   - Update $Q$ according to
     \[ Q(\tilde{s}, \tilde{a}) \leftarrow (1 - \alpha_k) Q(\tilde{s}, \tilde{a}) + \alpha_k \left[ \tilde{r} + \gamma \max_{a' \in A} Q(\tilde{s}', a') \right] \]
7. update $s \leftarrow s'$
8. goto 3
Dyna-Q Learning

A mean-reverting asset
Dyna-Q Learning
An execution strategy
Dyna-Q Learning

An execution strategy