DQN paper summary

**Chapter 1 - Overview**

**Stochastic Optimal Control problems**

* Model: Propose a stochastic model (based on empirical observations – e.g OU or stochastic volatility process)
* Calibrate: Use historical data to estimate model parameters (e.g volatility, mean reversion etc)
* Formulate: Propose a performance criterion which they aim to maximize
* Solve: analytically using methods in stochastic optimal control (e.g Bellman’s equation)
* E.g Almgren-Chriss (most popular)

**RL**

Use case

* Can solve more complex price models that don’t have analytical solutions
* OR non-parametric models (model free)

Overview

* Reinforcement learning [12] attempts to learn optimal policies for sequential decision problems by optimizing a cumulative future reward function with few modeling assumptions (such as Markovian structure of the state space).
  + Q-learning (pros and cons -> e.g limits the ability to generalize i.e interpolate between actions and sates (discrete), computationl power etc)
  + Deep Q learning (pros -> solves the above cons)

**Chapter 2 – Background**

* LOB defining
* Optimal execution defining
* Reinforcement Learning formulation
* Deep Q-Learning formulation

**Chapter 3 – Optimal execution in a RL setting**

Setting

* MO only
* Benchmark: TWAP
* Time horizons: Time periods (T; execution decisions taken) and trades at each tick
* State space: Inventory, Time, (Optional: Price, Returns, Volatiltiy)
* Action: discretised amounted of shares to execute (range(0, Inventory))
* Rewards: depend on returns and amount of shares executed
  + /Users/andreastheodoulou/Desktop/Screen Shot 2019-08-25 at 5.20.48 PM.png(at each time period – some over time this quantity for total rewards)
  + the more the shares executed the less the rewards because the higher your price impact
  + Penalize price impact by a (a = 0.01)

**Chapter 4 – Network Architecture**