Markov Decision Problems in Finance and High Dimensionality

Frédéric Godin, Ph.D., FSA, ACIA
Concordia - Mathematics and Statistics dept.

Concordia University - Big Data Day

Winter 2018

Presention content

Content of the presentation: discuss

- **Markov Decision Problems** (MDP) and their **applications in finance**,
- challenges associated with their use e.g. **curse of dimensionality**,
- potential solution avenues.

Sequential decision problems

Many problems involve performing a **sequence of decisions**.

For instance:

- Periodic rebalancing of a financial portfolio,
- Identifying the shortest path for delivery of multiple goods,
- Optimal replenishing of a retailer's inventory.

Objective: **optimization of the sequence of decisions**.

- Decisions are optimized jointly: each decision has impact on future ones; impacts should be **anticipated**.

Sequential decision problems

Sequential decision problems are mathematical tools representing such frameworks.

Such problems involve

- Set of time points $\mathcal{T} = \{0, 1, \dots, T\}$
- Set of states S,
- Set of actions A,
- Transition probabilities P between states.

Sequential decision problems

How sequential decision problems work?

At time $t = 0, \dots T-1$,

- The system is in **state** $s_t \in S$ at the beginning of the period,
- Then an **action** $a_t \in A$ is taken,
- An agent receives a **reward** $r_t(s_t, a_t)$,
- The system randomly **transitions** to state s_{t+1} at time $t+1$ with the transition probability $P(s_{t+1}|\mathcal{F}_t)$. [1]

[1] $\mathcal{F}_t$ is the information generated by previous states $s_0, \dots, s_t$ and actions $a_0, \dots, a_t$.

Sequential decision problems

We will consider **Markov decision problems**.

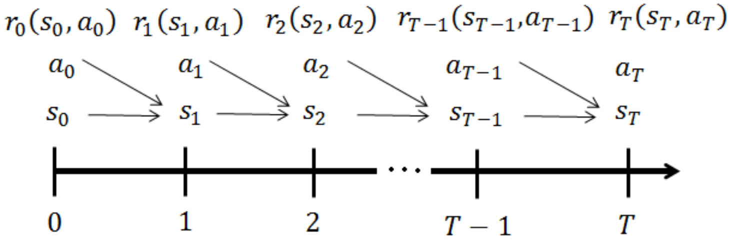
Particular case of sequential decision problems:

- transition probabilities only depend on most recent state and action.
- $P(s_{t+1}|\mathcal{F}_t) = P(s_{t+1}|s_t, a_t)$

In this case, for a given policy $\{a_t\}_{t=0}^{T}$, the state process $\{s_t\}_{t=0}^{T}$ is a **Markov** process.

Markov decision problems

Figure : Markov decision problem



Markov decision problems

Objective: **identify policy** $a = \{a_0, \ldots, a_T\}$ which optimizes the expected total reward:

$$\max_{a_0,\ldots,a_T} \mathbb{E}\left[\sum_{t=0}^{T} r_t(s_t, a_t)\right]$$

- Sometimes (e.g. in finance), rewards are discounted.
- More general risk measures ρ than $\mathbb{E}$ are sometimes consider.
  - We will not consider such extensions here.

Solving Markov decision problems

The **solution** approach for Markov decision problems through **dynamic programming** is well known.

Define the **value functions**

$$\Psi_t(s_t) = \max_{a_t,\ldots,a_T} \mathbb{E}\left[\sum_{u=t}^{T} r_u(s_u, a_u) \middle| s_t \right]$$

as the **optimal expected reward from time** t **on** if **behaving optimally** from that point.

We seek $\Psi_0(s_0)$, the maximal expected total reward at $t = 0$.

Solving Markov decision problems

Value functions can be **calculated recursively**:

$$
\begin{aligned}
\Psi_t(s_t) &= \max_{a_t,\ldots,a_T} \mathbb{E}\left[\sum_{u=t}^{T} r_u(s_u, a_u) \middle| s_t \right] \\
&= \max_{a_t} \max_{a_{t+1},\ldots,a_T} r_t(s_t, a_t) + \mathbb{E}\left[\mathbb{E}\left[\sum_{u=t+1}^{T} r_u(s_u, a_u) \middle| s_{t+1} \right] \middle| s_t \right] \\
&= \max_{a_t} r_t(s_t, a_t) + \mathbb{E}\left[\max_{a_{t+1},\ldots,a_T} \mathbb{E}\left[\sum_{u=t+1}^{T} r_u(s_u, a_u) \middle| s_{t+1} \right] \middle| s_t \right] \\
&= \max_{a_t} r_t(s_t, a_t) + \mathbb{E}\left[\Psi_{t+1}(s_{t+1}) \middle| s_t \right]
\end{aligned}
$$

Recall distribution of s_{t+1} given s_t depends on a_t.

Solving Markov decision problems

Recursive solution scheme is called the **Bellman Equation**:

$$\Psi_t(s_t) = \max_{a_t} \left(r_t(s_t, a_t) + \mathbb{E}\left[\Psi_t(s_{t+1}) \middle| s_t \right] \right).$$

Furthermore, define

$$a_t^* := \arg\max_{a_t} \left(r_t(s_t, a_t) + \mathbb{E}\left[\Psi_t(s_{t+1}) \middle| s_t \right] \right).$$

Then, $(a_0^*, \ldots, a_T^*)$ solves the original problem i.e.

$$(a_0^*, \ldots, a_T^*) = \arg\max_{a_0, \ldots, a_T} \mathbb{E}\left[\sum_{t=0}^{T} r_t(s_t, a_t) \right].$$

Principle of optimality

The approach consists in **decomposing a large optimization problem into smaller subproblems** more easily solvable.

- This is a core idea of dynamic programing.
- The fact that **joining solutions** of all subproblems yield a **solution to the original problem** is referred to as the **principle of optimality**.

Applications to finance

Some examples of Markov Decision Problems applications to finance:

- Investment portfolio optimization,
- Hedging,
- Optimal liquidation of a portfolio.

Investment portfolio optimization

Investment portfolio optimization problem:
Consider a market with J assets.

- $S_t^{(j)}$ denotes the time-t price of asset j.
- $R_t^{(j)} := \left(\frac{S_t^{(j)}}{S_{t-1}^{(j)}}\right) - 1$ is its time t return.

Defining $\mathbf{R}_t = \left(R_t^{(1)}, \ldots, R_t^{(J)}\right)$, we first assume $\mathbf{R}_1, \ldots, \mathbf{R}_T$ are i.i.d.

- Asset prices S is a Markov process.

Investment portfolio optimization

Define $w_{t+1}^{(j)}$ as the **percentage of portfolio invested in asset** j during $[t, t+1)$.

- The w's are called **weights**.
- Weights are decided by the portfolio manager (decision variable).

Denote by V_t the time-t **portfolio value** evolving as

$$V_{t+1} = V_t \left(1 + \sum_{j=1}^{J} w_{t+1}^{(j)} R_{t+1}^{(j)} \right).$$

- Portfolio value V_t acts as the **state** of our system.
- Transition probabilities characterized by distribution of $\mathbf{R}_{t+1}$.

Investment portfolio optimization

Reward characterized by a **utility function** U:

- Measures **satisfaction** associated with a level of wealth.

Optimization problem becomes

$$\max_{\mathbf{w}_1,\ldots,\mathbf{w}_T} \mathbb{E}\left[U(V_T)\right]$$

i.e. **maximizing portfolio allocation at each time step** to maximize expected utility.

Investment portfolio optimization

The state variable in this problem is the portfolio value: $s_t = V_t$.

A generalization involves including **consumption** from portfolio.

- Each step, allocation and consumption are optimized.
- Rewards include utility for consumed and terminal wealth.

Hedging optimization

A slightly different but related problem is **hedging**.

- A financial institution has a **random liability** $L(S_T)$ it has to pay at time T.

It attempts **offsetting the liability payoff** with the portfolio:

$$\min_{\mathbf{w}_1,\ldots,\mathbf{w}_T} \mathbb{E}\left[g\left(L(S_T) - V_T\right)\right]$$

where g penalizes hedging shortfalls.

- State variables are $s_t = (S_t, V_t)$ i.e. the stock prices and portfolio value.

Optimal fund liquidation

A third financial problem being an MDP is **optimal fund liquidation**.

- A fund manager must liquidate the assets from the fund.
- He is subject to **market impact**; selling too many assets draws price down and creates losses.
- Waiting for too long before selling generates **market risk**.

Can be formulated as MDP:

- Every step, need to choose the optimal amount of assets to sell.

High dimensionality in financial problems

Previous versions of problems presented have low dimensionality:

- $s_t = V_t$ for investment portfolio optimization,
- $s_t = (V_t, S_t)$ for the hedging problem.

However, to **increase the model realism**, several other state variables could be included.

For instance, i.i.d. returns assumption does not hold. One could include

- stochastic volatilities (one per asset),
- stochastic correlations,
- market regimes (Hidden Markov Models),
- autocorrelation (lagged return).

High dimensionality in financial problems

Other features could be embedded in the state space:

- stochastic interest rates (e.g. factor models),
- stochastic exchange rates,
- transaction fees (need to include previous portfolio positions),
- stochastic mortality (actuarial liabilities),
- etc.

Including all these features would make s_t a high-dimensional state variable.

Lookup table solution

Bellman Equation requires solving

$$\Psi_t(s_t) = \max_{a_t} r_t(s_t, a_t) + \mathbb{E}\left[\Psi_{t+1}(s_{t+1})\middle| s_t\right].$$

for all values of s_t and all t.

s_t often takes continuous values;

- Can **discretize** possible values for s_t and use **interpolation** to approximate Ψ_t between grid nodes.
- Requires solving the problem (i.e. calculating $\Psi_t(s_t)$) for all s_t.
- Referred to as **lookup table approach**.

Approximate dynamic programming

The **lookup table** approach is **infeasible** when s_t is **in high dimension**.

- If N nodes in the grid for each dimension,
- s_t in D dimension,
- $\Rightarrow N^D$ optimization problems to solve.
- **Curse of dimensionality**

A possibility is to use **approximate dynamic programming** methods.

- Calculate $\Psi_t(s_t)$ for a few values of s_t (e.g. randomly selected),
- Use **high-dimensional generalization** approaches (e.g. neural networks) to obtain estimates of $\Psi_t(s_t)$ for other values of s_t.

Approximate dynamic programming

Issues related to backward induction solution with neural network representation:

- Ψ_t and Ψ_{t+1} are likely to be **very similar**.
  - Making several times very similar calculations.
- **Some values** of the state space are very **unlikely to be reached**.
  - Wasting time on values unlikely to be used.

Reinforcement learning

Reinforcement learning methods could be considered to handle these issues.

We can include t in the state space i.e. state space is (t, s_t).

- A single value function Ψ.

1. Start with an initial estimate of Ψ using simplistic assumptions.
2. Continuously **simulate** the Markov decision process, and **iteratively refine your estimate** of Ψ.
    - See for instance temporal difference (TD) methods.
3. While the Ψ estimate is refined, the optimal policy $\{a_t\}_{t=0}^{T}$ also is.

Reinforcement learning

Having a single value function **avoids repeating similar calculations** for Ψ_{t+1} and Ψ_t.

Simulating the Markov dynamics leads to infrequent **visits of unlikely states**.

- Few effort is applied in identifying optimal policies for these states.

Conclusion

Several finance problems can be expressed as **Markov decision problems**.

To increase model realism, such problems involve **high dimensional state spaces**.

For such high dimensional problems, **typical lookup table solution** of Bellman Equation does not work.

Need to resort to **approximate dynamic programming**, e.g.

- Neural network representation of value function,
- Forward dynamic programming/reinforcement learning.

References

- Bertsekas, D. P., Tsitsiklis, J. (1992). Neuro-Dynamic Programming. *Athena Scientific*.
- François, P., Gauthier, G., Godin, F. (2014). Optimal hedging when the underlying asset follows a regime-switching Markov process. *European Journal of Operational Research*, **237**(1), 312-322.
- Godin, F. (2016). Minimizing CVaR in global dynamic hedging with transaction costs. *Quantitative Finance*, **16**(3), 461-475.
- Powell, W. B. (2007). Approximate Dynamic Programming: Solving the curses of dimensionality (Vol. 703). *John Wiley & Sons*.