A reinforcement learning approach for pricing derivatives

Thomas Grassl TGRASSL@STANFORD.EDU
Susquehanna International Group GRASSL.THOMAS@GMAIL.COM

Abstract

Analytical solutions to the derivatives pricing problem are known for only a small subset of derivatives and are usually based on strict assumptions. Practitioners will therefore frequently resort to numerical approximation techniques. In this paper, I will formulate a simple Markov decision process for which the optimal value function will, in a non-arbitrage world, be equivalent to a given derivative's fair price function. This means that derivatives pricing can be understood as a reinforcement learning problem. In order to solve this problem I will propose a simplified version of the kernel-based reinforcement learning algorithm suggested in [4] and [6].

1. Introduction

One of the biggest challenges in finance is to properly price derivatives. Analytical fair price models are only known for a small subset of derivatives and are generally based on rather strict assumptions. Practitioners thus usually turn to numerical approximation techniques in order to estimate the fair price; common approaches rely on the use of Monte Carlo simulations (see [3] or [5]) or on methods rooted in dynamic programming (see [1] or [8]). The latter cleverly relate the fair price of a derivative to the optimal value function of a Markov decision process (MDP), but require explicit knowledge of the MDP's transition probabilities in order to solve for the optimal value function. Similarly, also Monte Carlo methods need to know the probability distribution of the state space in order to generate random samples from which they can extract their fair price estimates. In reinforcement learning on the other hand, state space and transition probabilities are only used implicitly as the learning is based on trajectory samples from the MDP. We can thus learn our pricing model directly from data without having to assume that the underlying's price follows a specific price process. This means that one of the biggest pitfalls of most derivatives pricing methods can be avoided.

LAST REVISION: DECEMBER 10, 2010.

2. Theoretical framework

2.1. The derivatives pricing problem

Let U be a risky asset with a stochastic price process given by $\{U_t, t \geq 0\}$. We can construct another risky asset D such that its spot price D_t will be a deterministic function of U_t at exactly one point in the future, i.e., at some time T the prices of D and U will be related according to $D_T = g(U_T)$. In finance, such an asset D is called a derivative[1] with underlying U and maturity T. The function g is called the payoff function of the derivative D.

The derivatives pricing problem deals with the question of how the price of D depends on U_t at times $t < T$. In absence of arbitrage and under the assumption that $\{U_t\}$ is a Markov process[2], the fair price[3] D_t^F is simply the discounted, expected payoff of D given U_t, i.e., if r is the risk-free interest rate then

$$D_t^F = e^{-r(T-t)} E[g(U_T)|U_t]. \tag{1}$$

The difficult part in (1) is to determine $E[g(U_T)|U_t]$. The function g might not be analytically tractable, or even if it is, finding the expected value of $g(U_T)$ requires detailed knowledge of the structure of the probabilistic price process $\{U_t, t \geq 0\}$. Analytical solutions to this problem depend on specific assumptions on $\{U_t\}$ which may break down when applying the resulting model in practice. The classic Black-Scholes model for example assumes that the returns of U_t are sampled from a continuous, log-normal distribution, and by doing so fails to acknowledge the existence of jump discontinuities or fat tails that can often be observed in real world price data.

[1] This definition was chosen because it simplifies much of the subsequent work. However, most results should also be applicable to more complicated derivates.

[2] Note that this is the only assumption that we will impose on the stochastic process $\{U_t, t \geq 0\}$.

[3] Note the difference in notation between spot price D_t and fair price D_t^F.

2.2. Trading as a Markov decision process

Suppose now that a trader holds exactly one unit of a derivative D at time t. At each time prior to expiry T, he can either sell D at the current spot price D_t or decide to hold on to it; if he still holds D at time T, both selling and holding will have the same outcome: the derivative will be executed (or sold) for the price $D_T = g(U_T)$. A trading episode ends as soon as D is either sold or executed. We will denote the trader's possible actions as

$$a_t = \begin{cases} a_S & \text{if } D \text{ is being sold} \\ a_H & \text{otherwise} \end{cases}$$

and assume that neither action can affect the market's behaviour, i.e., D_{t+1} and U_{t+1} or, more generally, the market state m_{t+1} are independent of a_t. Only the trader's position q_{t+1} depends on his actions. Combining m_t and q_t yields the state of the world at time t, $s_t = (m_t, q_t)$. The state-action reward in this model will simply be equal to the trader's monetary compensation, i.e.

$$R(s_t, a_t) = \begin{cases} D_t & \text{if } a_t = a_S, t < T \\ D_T = g(U_T) & \text{if } t = T, q_t = 1 \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

For the sake of simplicity, we will assume that all episodes end at time T; if $q_t = 0$ for $t < T$, then the trader will simply have to choose a_H at each timestep between t and T (and thus receive zero reward).

Using $R(s_t, a_t)$ as defined above, the sum of expected future rewards of a given strategy (or policy) π discounted to time t can be written as

$$V^\pi(s_t) = E\left[\sum_{\tau=t}^{T} e^{-r(\tau-t)} R(s_\tau, \pi(s_\tau)) \;\middle|\; s_t\right].$$

From this is it obvious that the value function V^π of our trading strategy obeys the Bellman equation

$$V^\pi(s_t) = R(s_t, \pi(s_t)) + e^{-r} E\left[V^\pi(s_{t+1}) | s_t\right]. \quad (3)$$

Note now that because of the special structure of the rewards outlined in (2), the state-action value function Q satisfies

$$Q^\pi(s_t, a_t) = \begin{cases} e^{-r} E\left[V^\pi(s_{t+1}) | s_t\right] & \text{if } a_t = a_H \\ D_t & \text{if } a_t = a_S \end{cases} \quad (4)$$

2.3. Equivalence of optimal value and fair price function

For the policy π_H where the trader will hold D until expiry, the corresponding value function V^{π_H} can easily be computed:

$$V^{\pi_H}(s_t) = e^{-r(T-t)} E\left[g(U_T) | U_t\right].$$

Using (1) we can see that this is nothing else than the fair price of D:

$$V^{\pi_H}(s_t) = D_t^F. \quad (5)$$

Suppose now that π_H is not optimal, i.e., that there is a state s_τ where it is better to sell than to hold:

$$Q^{\pi_H}(s_\tau, a_S) > Q^{\pi_H}(s_\tau, a_H).$$

Using equation (4) this implies that

$$D_\tau > e^{-r} E\left[V^{\pi_H}(s_{\tau+1}) | s_\tau\right].$$

The assumption that $\{U_t\}$ is Markov implies that

$$e^{-r} E\left[V^{\pi_H}(s_{\tau+1}) | s_\tau\right] = e^{-r(T-\tau)} E\left[g(U_T) | U_\tau\right] = D_\tau^F$$

and thus that $D_\tau > D_\tau^F$.

Since in a no-arbitrage world $D_t = D_t^F$ for all t, it follows that no such state s_τ can exist, and therefore that π_H is an optimal strategy. A direct implication of this is that the optimal value function V^* satisfies

$$V^*(s_t) = D_t^F \quad (6)$$

for all states s_t. This shows that in the absence of arbitrage, learning the optimal value function V^* is equivalent to learning the fair value function D_t^F. In other words, by solving the above MDP for V^*, we will be able to solve the derivatives pricing problem.

Note that the above derivation required that $D_t = D_t^F$. The result however remains valid even if the market tends to underestimate the fair price (i.e. $D_t \leq D_t^F$) since π_H would still be an optimal policy. If we allow D_t to be greater than D_t^F, then also the resulting value function would be greater than D_t^F. It would thus provide a measure for the expected extent of mispricings that a trader can exploit in this flawed market.

2.4. Generalizing the fair price model

In the above section I have shown that for a specific derivative D its fair price can be determined by learning the optimal value function of a simple MDP. While theoretically useful, this approach would likely be unpractical in real-world trading as traders would be required to maintain a separate model for each derivative they are interested in. Pricing models with analytical solutions, e.g., the classic Black-Scholes model, do not suffer from this shortcoming and can easily be applied to a whole class of derivatives. Such a generalization is achieved by finding a suitable parameterization of the considered derivative; the classic Black-Scholes pricing approach for example parameterizes a European option as a tuple consisting of expiration time T, strike

price K and the volatility of returns σ of the underlying asset. The trading MDP can be generalized with a similar trick: we simply absorb[4] the characteristics of a derivative D into the state s_t by using a suitable parameterization ψ_D and define an enhanced state

$$s_t^D = (s_t, \psi_D)\,.$$

As long as the mapping $D \to \psi_D$ is bijective, it is impossible to move from a state s_t^D to $s_{t+1}^{\tilde{D}}$ where $D \neq \tilde{D}$. This means that we will learn the optimal value function $V^*(s_t^D)$ using the same trajectories as before and thus that in a non-arbitrage world $V^*(s_t^D) = D_t^F$. If on the other hand $D \to \psi_D$ is not a bijective mapping then either $V^*(s_t^D) = V^*(s_t^{\tilde{D}})$ is the desired outcome for $s_t^D = s_t^{\tilde{D}}$ or ψ needs to be improved.

Note that for a bijective parameterization on a discrete state space, this trick only provides a unified notation for accessing the fair price model of distinct derivatives; learning the fair price still takes place separately for each of the considered derivatives. This will however change when we attempt to compute approximate value functions over continuous state spaces.

3. Implementation considerations

3.1. Exploitation vs. Exploration

A problem inherent to reinforcement learning is the exploration/exploitation dilemma where a reinforcement agent faces a trade-off between maximizing the short-term reward by exploiting his current knowledge of the MDP or maximizing the long-term reward by exploring unknown regions of the state-action space (see [7]). The special nature of the trading MDP eliminates this dilemma: the state s_t is described as a tuple (m_t, q_t) where the trader's position q_t is directly and deterministically affected by his actions. A transition from m_t to m_{t+1} will thus provide information regarding both possible action choices a_S and a_H. This property of the trading MDP simplifies the collection of data as only the transitions of the market states m_t need to be observed.

3.2. Kernel-Based Reinforcement Learning

Trading is essentially a finite MDP: a trader only needs to act at discrete times (e.g., when new information becomes available), prices and sizes can only change in discrete increments and the action space in the above MDP consists of only two choices, a_H and a_S. However, the dimensions of this MDP can quickly become so large that the problem is computationally unmanageable. To avoid this, we will assume that the trading MDP has a continuous state space.

Most canonical reinforcement learning algorithms and their corresponding convergence guarantuees deal with the problem of approximating optimal policies for finite state spaces and can usually not easily be generalized to the continuous case. Instead, reinforcement learning in continuous state space often attempts to approximate the optimal value function directly from a given sample of trajectories from the MDP.

One approach that is equipped with a guarantueed convergence to the optimal value function V^* is Kernel-Based Reinforcement Learning (see [6]). A meaningful modification to it is provided in [4]: it is shown that finding the approximate value function of an exact, continuous MDP can be understood to be equivalent to finding the exact value function of an approximate, finite MDP while still maintaining the same convergence guarantuees. Applying this approach to the trading MDP essentially means that we will interpret samples of a high-dimensional discrete MDP as samples from a continuous MDP which we will then solve by approximating it with a simpler discrete MDP.

3.3. Constructing an approximate MDP

Suppose that we have sampled n transitions $(s^{(i)}, \hat{s}^{(i)}, a^{(i)})$ (where we transitioned from state $s^{(i)}$ to $\hat{s}^{(i)}$ as a result of action $a^{(i)}$) from an MDP $M = (S, A, T, R)$ with continuous state space S, discrete action space A, transition probabilities T and reward function R. Jong and Stone (see [4]) use these samples to approximate M with a finite MDP $\tilde{M} = (D, A, \tilde{T}, \tilde{R})$ where $D = \{\hat{s}^{(i)}\}$ is the set of successor states. For some kernel function ϕ with suitably chosen bandwidth b and premetric d, $\tilde{T}$ and $\tilde{R}$ are defined as

$$\tilde{T}(s, a, \hat{s}^{(i)}) = \begin{cases} \frac{1}{Z^{s,a}} \phi\left(\frac{d(s, s^{(i)})}{b}\right) & \text{if } a^{(i)} = a \\ 0 & \text{otherwise} \end{cases}$$

$$\tilde{R}(s, a) = \frac{1}{Z^{s,a}} \sum_{i | a^{(i)} = a} \phi\left(\frac{d(s, s^{(i)})}{b}\right) R(s^{(i)}, a^{(i)})$$

$$Z^{s,a} = \sum_{i | a^{(i)} = a} \phi\left(\frac{d(s, s^{(i)})}{b}\right).$$

The authors argue that the exact solution of $\tilde{M}$, $\tilde{V}^*$, converges to the exact solution of M, V^*, as the number of samples increases.

In our trading MDP, the market transitions independently of the chosen action a; this yields the following

[4] For a practical example of how to absorb the derivative's parameters into the state space see section (4.2).

simplified version of the above expressions:

$$\tilde{T}(s,a,\hat{s}^{(i)}) = \tilde{T}(m,\hat{m}^{(i)}) = \frac{1}{Z^m}\phi\left(\frac{d(m,m^{(i)})}{b}\right)$$

$$\tilde{R}(s,a) = \frac{1}{Z^m}\sum_i \phi\left(\frac{d(m,m^{(i)})}{b}\right) R(s^{(i)},a)$$

$$Z^m = \sum_i \phi\left(\frac{d(m,m^{(i)})}{b}\right).$$

For the state-action value function $\tilde{Q}$, it follows that

$$\tilde{Q}(s,a) = \tilde{R}(s,a) + e^{-r}\sum_i \tilde{T}(m,\hat{m}^{(i)})\tilde{V}(\hat{s}^{(i)})$$

$$= \begin{cases} \tilde{R}(s,a) & \text{if } a = a_S \\ e^{-r}\sum_i \tilde{T}(m,\hat{m}^{(i)})\tilde{V}(\hat{s}^{(i)}) & \text{otherwise} \end{cases}$$

where $\tilde{V}(s) = \max_a \tilde{Q}(s,a)$. Thus, estimating V^* boils down to two steps: generating $\tilde{T}$ by observing enough market transitions, and then solving the MDP $\tilde{M}$ (for example by using value iteration). A significant computational obstacle is that the size of $\tilde{T}$ is quadratic in the number of observed samples. In [4], the authors suggest that very small entries of $\tilde{T}$ should be set to zero. In the below example this resulted in a very sparse transition matrix[5] without noticeably affecting the approximation quality of the model.

4. Example: Pricing a European call

4.1. The classic Black-Scholes approach

Suppose that the underlying's spot price U_t follows a log-normal random walk with drift μ and volatility σ

$$dU_t = \mu U_t\,dt + \sigma U_t\,dW_t$$

where W_t is a Wiener process. Using a non-arbitrage argument it can be shown that μ has to equal the risk-less interest rate r. In such a setting, the Black-Scholes model (see [2]) defines the fair price of a European call option C_t with strike price K and expiration T as

$$C_t = U_t N(d_1) - Ke^{-r(T-t)}N(d_2) \tag{7}$$

where N is the standard normal cumulative distribution function and

$$d_{1|2} = \frac{\ln(U_t/K) + (r \pm \sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}.$$

4.2. Parameterizing the state space

Equation (7) can be normalized by recognizing that C_t is the product of K and the price of a call $\tilde{C}_t$ with strike price 1 and underlying prices given by $\frac{U_t}{K}$:

$$C_t = K\tilde{C}_t.$$

Instead of learning C_t directly, one can thus learn $\tilde{C}_t$ and recover C_t as required. Using strike-relative prices $\frac{U_t}{K}$ yields a more compact price space and makes it feasible to efficiently learn our model using data from a variety of derivatives with potentially vastly different price levels.

Note now that σW_t models the randomness in the Black-Scholes world. Since W_t is a Wiener process, the future price risk $\sigma(W_T - W_t)$ is distributed according to $\mathcal{N}(0, \sigma\sqrt{T-t})$. This suggests that in order to capture the random extent of future price movements, it is not necessary to explicitly use t, T and σ as separate state features; instead we can use a single feature equal to $\sigma\sqrt{T-t}$.

These observations indicate that a compact but reasonably complete definition of a market state is

$$m_t = \left(\frac{U_t}{K}, \sigma\sqrt{T-t}\right).$$

This definition should enable us to easily combine derivatives with different underlyings, strike prices, expiration times or volatilities into the same model. U_t, K and $T-t$ are all directly observable while the volatility σ needs to be estimated from historical data.[6]

4.3. Results

I initially implemented the above version of Kernel-Based Reinforcement Learning using a Euclidian distance metric d and a Gaussian kernel ϕ. Intuitively, a small bandwidth b should result in a very bumpy value function approximation $\tilde{V}^*$ as each sampled transition $(m^{(i)}, \hat{m}^{(i)})$ can only noticeably affect predictions in its close proximity. As b increases, the value function should take on an increasingly smooth shape. Initial experiments confirmed this inuition, but also unveiled that, for large values of b, $\tilde{V}^*$ was consistently overestimating the fair price of out-of-the money options: for large b, the perceptive regions of ϕ around sampled transitions will frequently overlap, which means that high prices from rather far away can be propagated through to regions of the state space where the considered option is nearly worthless. This problem is intrinsic to the chosen algorithm.

We can ameliorate this problem by recognizing that

[5] A pruning threshold of 0.001 reduced the fraction of nonzero entries from 67% to 1%.

[6] Note that σ is needed for differentiating between different underlyings; if we would focus on a single underlying we could disregard σ as the model should pick up this information directly from the data.

the Euclidian distance measure is unbiased with regards to the single components of a state: e.g., points given by $(x_1 + c, x_2)$ and $(x_1, x_2 + c)$ are equally far way from (x_1, x_2). Such symmetry is not really desired when pricing options: it seems to be a better idea to compare options with similar strike-relative prices but different risk than to compare options with similar risk but different strike-relative prices. Furthermore, notice that the future risk $\sigma\sqrt{T-t}$ is decreasing with inreasing t. Since the true value function reflects the expected future reward, its approximation should ideally only depend on states that truly lie in the future, i.e., on states where the future risk is smaller than in the current state. If we would strictly enforce this forward-looking perspective, estimates close to expiry would suffer from a lack of data; this suggests that we need a compromise that is looking forward more than it is looking backwards. These ideas can be incorporated into the model by modifying the distance function d such that its contours are egg-shaped curves tilted towards smaller values of $\sigma\sqrt{T-t}$.

This change greatly improved the quality of the model. Figures (1) and (2) show the fair price approximation (using $b = 0.13$) and its associated error as compared to the Black-Scholes price for a European call with strike price 10, annualized volatility of 40% and an annual riskless interest rate of 2%. Learning was based on data from 1000 randomly constructed European calls (with different underlyings, strike prices, expirations etc.) and roughly 11000 randomly generated transitions; the returns of the respective underlyings were sampled from a log-normal distribution and the spot prices of the derivatives were assumed to be equal to the exact Black-Scholes price. The resulting root-mean squared error of this approximation corresponds to a $-value of less than 0.02 which is a promising first result. The model's behaviour was consistent across successive runs with different initializations of the random number generator.

5. Conclusion

I showed how the derivatives pricing problem can be written as an MDP to which reinforcement learning techniques can readily be applied. The proposed algorithm, a version of Kernel-Based Reinforcement Learning, delivered enouraging results on a simple test problem, but extensive testing is needed in order to evaluate the quality of this approach in more realistic scenarios. Applying the model to American or Asian options and avoiding the quadratic storage requirements for $\tilde{T}$ by making better use of its sparseness could be promising directions for future research.

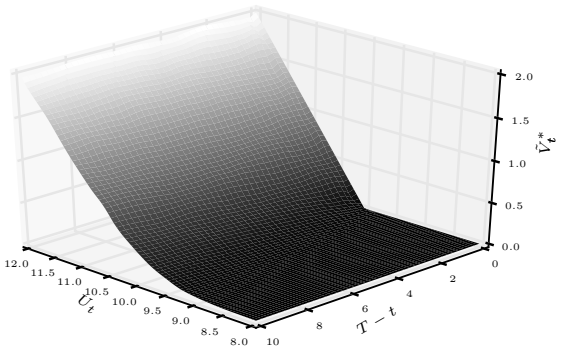

Figure 1. Approximated value function $\tilde{V}^*$ for a European call with strike $K = 10$

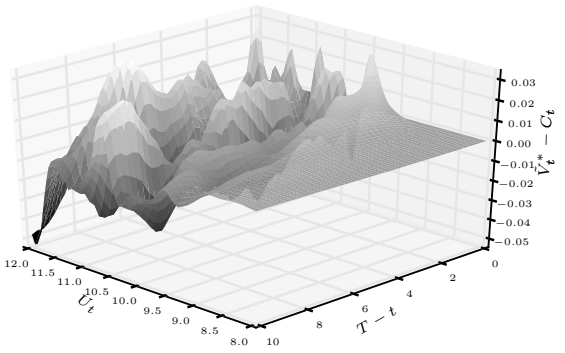

Figure 2. Approximation error $\tilde{V}^*_t - C_t$

References

[1] Hatem Ben-Ameur, Michèle Breton, Lotfi Karoui, and Pierre L'Ecuyer. A dynamic programming approach for pricing options embedded in bonds. *Journal of Economic Dynamics and Control*, 31(7):2212 – 2233, 2007.

[2] Fischer Black and Myron Scholes. The Pricing of Options and Corporate Liabilities. *The Journal of Political Economy*, 81(3):637–654, 1973.

[3] Mark Broadie and Paul Glasserman. Pricing American-style securities using simulation. *Journal of Economic Dynamics and Control*, 21(8-9):1323 – 1352, 1997. Computational financial modelling.

[4] Nicholas Jong and Peter Stone. Kernel-based models for reinforcement learning in continuous state spaces. In *ICML workshop on Kernel Machines and Reinforcement Learning*, June 2006.

[5] Francis A. Longstaff and Eduardo S. Schwartz. Valuing American options by simulation: a simple least-squares approach. *Review of Financial Studies*, 14(1):113–147, 2001.

[6] Dirk Ormoneit and Śaunak Sen. Kernel-based reinforcement learning. *Machine Learning*, 49:161–178, November 2002.

[7] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning)*. The MIT Press, March 1998.

[8] John N. Tsitsiklis and Benjamin Van Roy. Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks*, 12:694–703, 2000.