**SO SÁNH ĐIỀU KHIỂN DỰ BÁO VÀ PHƯƠNG PHÁP**

**HỌC MÁY TĂNG CƯỜNG**

***Nguyễn Tiến Ban***

*Khoa Điện Cơ*

*Email:* *bannt@dhhp.edu.vn*

*Ngày nhận bài: 28/12/2019*

*Ngày PB đánh giá: 20/1/2020*

*Ngày duyệt đăng: 10/2/2020*

**Tóm tắt**:

*Bài báo so sánh hai phương pháp điều khiển liên quan đến điều khiển tối ưu là điều khiển dự báo (Model Predictive Control - MPC) và phương pháp học máy tăng cường (Reinforcement Learning - RL) cho hệ thống điều khiển rời rạc. Về cơ bản, MPC điều khiển dựa trên mô hình và có thể đảm bảo tính ổn định, khả thi và bền vững cho hệ đồng thời giải quyết tốt vấn đề giới hạn của trạng thái và tín hiệu điều khiển trong khi đó RL không cần biết trước mô hình và mang tính thích nghi cao. Việc kết hợp hai phương pháp để tận dụng ưu điểm của mỗi bên đang được đẩy mạnh nghiên cứu.*

**Từ khóa**: *MPC – Bộ điều khiển dự báo; RL - Học máy tăng cường; Tính ổn định; ADP - Quy hoạch động thích nghi; Điều khiển hệ có giới hạn.*

***COMPARISON OF MODEL PREDICTIVE CONTROL AND REINFORCEMENT LEARNING***

**Abstract:**

*This paper compares two trending controller design approaches, namely Model Predictive Control (MPC) and Reinforcement Learning (RL) for discrete control systems subject to inputs and states constraints. While MPC, which is a model-based method, can guarantee stability, feasibility and robustness as well as can handle inputs and states constraints well, RL does not require the model of the plant to be known in advance and can be adaptive to the changes of environment. Numerous research have been carried out Numerous studies have been conducted to combine the advantages of both approaches.*

***Keywords:*** *MPC, Nonlinear Control, LMI, Optimal control, Robust Control, Lure systems.*

|  |  |
| --- | --- |
| MPC | Model Predictive Control - Bộ điều khiển dự báo |
| RL | Reinforcement Learning - Phương pháp học máy tăng cường |
| LQR | Linear Quadratic Regulator - bộ điều khiển toàn phương tuyến tính |
| ADP | Adaptive Dynamic Programming – Quy hoạch tuyến tính thích nghi |
| RHC | Receding Horizon Control - Điều khiển cửa sổ lùi dần (Một tên khác của điều khiển dự báo) |
| PWA | Picewise-Affine - Hàm số affine từng đoạn |

1. Phần mở đầu

Điều khiển dự báo (Model Predictive Control - MPC) và phương pháp học máy tăng cường (Reinforcement Learning – RL) là hai phương pháp điều khiển trong đó bài toán điều khiển được xem xét dưới dạng một bài toán tối ưu. Vì vậy, hai phương pháp này có những điểm giống nhau. Ví dụ, cả hai phương pháp này đều liên quan đến bài toán quy hoạch động (Dynamic Programming) và lời giải của cả hai phương pháp đều là lời giải cận tối ưu (suboptimal) cho bài toán điều khiển tối ưu. Đã có nhiều nghiên cứu chỉ ra những điểm tương đồng của hai phương pháp này liên quan đến bộ điều khiển toàn phương tuyến tính (Linear Quadratic Regulator – LQR). [1][3][5]

Tuy nhiên, MPC và RL có nhiều điểm khác nhau, được tóm tắt trong bảng 1. ĐIều khiển dự báo MPC đã được nghiên cứu trong một thời gian dài, và trong lĩnh vực điều khiển tuyến tính, MPC đã tỏ rõ sự nổi trội trong cả lý thuyết và thực tế [2] [4]. Trong điều khiển dự báo MPC, ở mỗi bước tính, bộ điều khiển giải một bài toán tối ưu cho lời giải (u(0), u(1), … u(h)) và đưa tín hiệu u(0) đến đối tượng. Sau đó, trạng thái x(k) của hệ được cập nhât và quá trình này được lặp lại. Trong khi đó, phương pháp học máy tăng cường RL lại có xuất phát đa dạng hơn. Hiện nay, RL được nói đến phổ biển là một trong ba trụ cột của lĩnh vực học máy (machine learning), bao gồm: học có giám sát (supervised learning), học không giám sát (unspervised learing) và học tăng cường (reinforcement learning). Trong đó, học tăng cường khác với hai hình thức học kia ở chỗ tác nhân (agents) hầu như không biết gì về môi trường xung quanh và tự tương tác với môi trường xung quanh (environment) và nhận về các phần thưởng tích lũy (cumulative reward), qua đó tự điều chỉnh hành vi để tối ưu tích lũy của mình. RL gây tiếng vang lớn khi công ty DeepMind thuộc Google tích hợp thành công vào robot Alpha Go để đánh bại kỳ thủ cờ vây số 1 thế giới Lee Sedol. Điều đáng kinh ngạc nhất với công chúng là Alpha Go đã tự học chơi cờ vây qua việc tự chơi với chính nó. Thực ra, RL đã xuất hiện trong các ngành nghiên cứu khác nhiều chục năm trước đó dưới những tên gọi khác nhau. Trong điều khiển, phương pháp điều khiển gần nhất với RL là quy hoạch động thích nghi (Adaptive Dynamic Programming – ADP) và thực tế hai tên gọi này vẫn được dùng với ý bao hàm lẫn nhau trong điều khiển.

Về cơ bản, MPC xếp vào dạng điều khiển dựa vào mô hình (model-based) nên luôn cần có mô hình toán của đối tượng, Trong khi đó, RL, về cơ bản, không yêu cầu thông tin về mô hình của đối tượng điều khiển. Có hai lớp RL thường được đề cập là model-based RL (RL dựa vào mô hình) và mode-free RL (RL không sử dụng mô hình). Mặc dù gọi là model-based nhưng loại RL thứ nhật không yêu cầu mô hình của đối tượng mà xây dựng mô hình của đối tượng qua thông tin quan sát được, qua đó sẽ đưa tín hiệu điều khiển dựa trên mô hình đó. Loại thứ hai thì không xây dựng mô hình của đối tượng mà sẽ đưa ra tín hiệu điều khiển trực tiếp dựa trên thông tin quan sát. Do cách thức tiếp cận khác nhau nên model based RL cần ít thông tin hơn và có nhiều đặc điểm tương tự MPC sau khi đã xây dựng được mô hình, trong khi model-free RL cần một lượng lớn thông tin để học [6].

Lý thuyết xây dựng xung quanh MPC về tính ổn định (stability) của hệ thống cũng như tính khả thi (feasibility ) khi áp dụng đã khá đầy đủ. Chúng ta cần giải thích về thuật ngữ tính khả thi (feasibility) trong bộ điều khiển MPC. Do cách thiết lập, MPC sẽ giải bài toán tối ưu trong mỗi bước tính, đưa ra tín hiệu điều khiển, cập nhật lại các biến trạng thái của hệ sau tín hiệu điều khiển và giải lại bài toán tối ưu. Bài toán tối ưu ở đây nói chung là bài toán tối ưu có ràng buộc do tín hiệu điều khiển và các biến trạng thái luôn bị giới hạn bởi các yếu tố kỹ thuật. Bài toán đó là tuyến tính hay phi tuyến thì phụ thuộc vào đối tượng điều khiển là tuyến tính hay phi tuyến. Vì vậy, một điều rất quan trọng trong quá trình sử dụng bộ điều khiển MPC là phải chứng minh được bài toán tối ưu đó luôn có lời giải trong suốt quá trình hoat động. Nói cách khác, nếu bài toàn tối ưu đó có lời giải ở bước t thì bài toán đó phải được đảm bảo rằng cũng có lời giải ở bước t+1, đó chính là tính khả thi của bộ điều khiển MPC. Đối với hệ tuyến tính thì lý thuyết về MPC đã được chứng minh khá đầy đủ [2] [4]. Hơn nữa, vì các ràng buộc về tín hiệu điều khiển và trạng thái đã được đưa trực tiếp vào cách thức thiết lập bài toán nên một cách tự nhiên, MPC giải quyết tốt các bài toán điều khiển có ràng buộc và đó luôn là lợi thế của MPC so với các phương pháp điều khiển khác. Ngược lại, các lý thuyết về tính ổn định và tính khả thi của RL còn khá sơ khởi. Do cách thiết lập bài toán mà khi muốn xây dựng được các điều kiện ổn định, chúng ta cần giả thiết rằng chúng ta đã có một phần thông tin về hệ thống. Kèm theo đó, các điệu kiện ràng buộc về trang thái của hệ cũng không được đảm bảo [1] [6].

Vấn đề điều khiển bền vững cũng được nghiên cứu cho MPC. Bản thân MPC đã có tính bền vững của riêng nó. Đặt dưới góc độ hệ thống điều khiển có sai lệch do nhiều tác động, có hai cách tiếp cận là điều khiển bền vững và điều khiển MPC ngẫu nhiên. Nhiều cách tiếp cận đã được đưa ra như Tube-MPC, Multi-mode MPC, Multi-Stage MPC [2] [4]. Trong khi đó, tính bền vững của RL vẫn còn là một dấu hỏi.

Trong bài báo này, cả hai phương pháp được so sánh và chỉ ra những điểm tương đồng và khác biệt dưới góc nhìn như hai cách tiếp cận khác nhau của điều khiển tối ưu, qua đó cố gắng đưa hai phương pháp lại gần nhau hơn để tận dụng được ưu điểm của cả hai cách tiếp cận.

Bảng 1: So sánh tính chất cơ bản của MPC và RL [1]

|  |  |  |
| --- | --- | --- |
| **Tính chất** | **Điều khiển dự báo MPC** | **Học máy tăng cường** |
| *Mô hình toán* | Phải có | Không yêu cầu |
| *Tính lồi* | Thường phải có | Không yêu cầu |
| *Tính thích nghi* | không có sẵn | có sẵn |
| *Tính phức tạp khi hoạt động online* | cao  | Thấp |
| *Tính phức tạp khi hoạt động offline* | Thấp | Cao |
| *Tính ổn định* | có lý thuyết xây dựng | chưa có lý thuyết |
| *Tính khả thi khi giải bài toán tối ưu* | có lý thuyết  | chưa có lý thuyết |
| *Tính bền vững* | có lý thuyết | chưa có lý thuyết |
| *Đảm bảo các giới hạn* | tín hiệu vào và trạng thái | tín hiệu vào |

**2. Điều khiển tối ưu**

**2.1. Đối tượng điều khiển và hàm mục tiêu**

Xét hệ rời rạc mô tả bởi phương trình

$x\left(t+1\right)=Ax\left(t\right)+Bu\left(t\right)$ (1)

trong đó x, u lần lượt là vector biến trạng thái và tín hiệu điều khiển. A, B là ma trận trạng thái và ma trận tín hiệu vào, có chiều n x n và n x m. Hệ (A, B) giả thiết là có thể ổn định được. Giả thiết các biến trạng thái x(t), là giá trị trạng thái x tại thời điểm t, có thể đo đạc được. Biến trạng thái thỏa mãn điều kiện

 $x\left(t\right)\in X⊂R^{n},u\left(t\right)\in U⊂R^{m}$. (2)

trong đó hai miền X và U là hai khối đa diện có chứa gốc tọa độ.

Xét phiếm hàm mục tiêu

 $V\_{N}\left(x\left(t\right)\right)=r\_{T}\left(x\_{t+N},u\_{t+N}\right)+\sum\_{k=0}^{N-1}γ^{k}r\left(x\_{t+k},u\_{t+k}\right)$, (3)

trong đó hàm tổn thất giai đoạn là

 $r\left(x\_{t+k},u\_{t+k}\right)=x\_{t+k}^{T}Qx\_{t+k}+u\_{t+k}^{T}Ru\_{t+k}$, (4)

và hàm tổn thất cuối là

$r\_{T}\left(x\_{t+N},u\_{t+N}\right)=x\_{t+N}^{T}Px\_{t+N}$ (5)

trong đó N là cửa số dự báo, Q và R là ma trận trọng số với số chiều phù hợp. P là ma trận trọng số cuối. Các ma trận Q, R, P là đối xứng và xác định dương. $γ$là hệ số giảm, nằm trong khoảng [0,1]. Chú ý rằng với điều khiển MPC, $γ $thường được chọn là 1. Ma trận P được xác định bằng cách giải phương trình đại số Riccati sau [2] [5]:

$\left(A+BK\right)^{T}P\left(A+BK\right)-P+Q+K^{T}RK=0$, (6)

trong đó

$K=-\left(R+B^{T}PB\right)^{-1}B^{T}PA$ (7)

**2.2. Bài toán điều khiển tối ưu với cửa sổ hữu hạn (finite horizon)**

Xét bài toán tối ưu sau đây

$V\_{N}\left(x\left(t\right)\right)=min\_{U\left(t\right)}r\_{T}\left(x\_{t+N},u\_{t+N}\right)+\sum\_{k=0}^{N-1}γ^{k}r\left(x\_{t+k},u\_{t+k}\right)$ (8a)

với

$x\_{t+k+1}=x\_{t+k}+u\_{t+k},k=1,2,...,N$ (8b)

$x\_{t+k}\in X,k=1,2,...,N$ (8c)

$u\_{t+k}\in U,k=1,2,...,N-1$ (8d)

$x\_{t+N}\in X\_{N}$ (8e)

$x\_{t}=x\left(t\right)$ (8g)

$U\left(t\right)=\left(u\_{t}^{T}....u\_{t+N-1}^{T}\right)^{T}\in R^{Nm}$ (8h)

Trong đó XN là miền giới hạn cuối của trạng thái cuối cùng xt+N. Miền giới hạn cuối và hàm tổn thất cuối đóng vai trò trong đảm bảo tính ổn định của hệ thống điều khiển.

**2.3. Bài toán điều khiển tối ưu với cửa sổ vô hạn (infinite horizon)**

Xét bài toán tối ưu sau đây

$V\_{\infty }\left(x\left(t\right)\right)=min\_{U\left(t\right)}\sum\_{k=0}^{\infty }γ^{k}r\left(x\_{t+k},u\_{t+k}\right)$ (9a)

với

$x\_{t+k+1}=x\_{t+k}+u\_{t+k},k=1,2,...$ (9b)

$x\_{t+k}\in X,k=1,2,...$ (9c)

$u\_{t+k}\in U,k=1,2,...$ (9d)

$x\_{t}=x\left(t\right)$ (9e)

$U\left(t\right)=\left(u\_{t}^{T}....\right)^{T}$ (9g)

Điểm khác biệt giữa bài toán điều khiển tối ưu với của sổ hữu hạn và vô hạn là trong bài toán điều khiển tối ưu vô han không có hàm tổn thất cuối và miền giới hạn cuối. Chú ý rằng bài toán này tương tự như bài toán điều khiển tối ưu LQR, tuy nhiên LQR không có ràng buộc về điều kiện tín hiệu điều khiển và trạng thái, vì thế bài toán điều khiển LQR có thể tìm được lời giải dưới dạng giải tích là một tín hiệu điều khiển phản hồi tuyến tính, còn bài toán (9) thì phức tạp hơn do có bao gồm các giới hạn (9c,d).

**3. Điều khiển dự báo MPC**

**3.1. Lời giải bài toán tối ưu cho trường hợp hệ tuyến tính**

Trong điều khiển dự báo, hệ số giảm $γ $được chọn bằng 1. MPC giải bài toán tối ưu (8) hoặc (9) trong từng bước tính t, sau đó đưa ra tín hiệu điều khiển ut+1 tới đối tượng điều khiển, sau khi có đáp ứng của đối tượng, đo đạc lại giá trị trạng thái xt+1 rồi các bước lặp lại như trên cho đến khi hệ được đưa tới trạng thái mong muốn. Chính vì cách thức hiện như vậy nên MPC còn thường được gọi là phương pháp điều khiển cửa sổ lùi dần (Receding Horizon Control – RHC).

Bài toán điều khiển (8) thực chất có thể đưa về giải một bài toán tối ưu dạng toàn phương [2] [4] [5]. Thực vậy, bài toán (8) tương đương với bài toán

$V\_{N}\left(x\left(t\right)\right)=x\left(t\right)^{T}Yx\left(t\right)+min\_{U}U^{T}HU+x\left(t\right)^{T}FU$ (10a)

với

$GU\leq W+Ex\left(t\right)$, (10b)

trong đó các ma trận Y, H, F ,G ,W và E có thể tìm được bằng cách sử dụng mô hình toán của hệ (1), viết dưới dạng sau:

$x\_{t+k}=A^{k}x\left(t\right)+\sum\_{}^{}A^{k}Bu\_{t+k-1-i}$. (11)

Nghiệm của (10) được chứng minh là hàm số affine từng đoạn (Piecewise Affine – PWA), nói cách khác u\*(t) có dạng như sau:

$u\left(t\right)=F\_{r}x\left(t\right)+g\_{r}$, (12)

trong đó Fr và gr có giá trị khác nhau trên các miền khác nhau.

Do tính chất như vậy, các miền mà Fr và gr có giá trị khác nhau có thể được tính toán offline. Đây chính là ý tưởng của phương pháp MPC hiệu quả để giải tỏa khối lượng tính toán cho bộ điều khiển. Thay vì mỗi bước phải giải lại bài toán tối ưu, các miền này được tính toán trước cùng các hệ số Fr và gr được lưu trữ trong bộ nhớ của bộ điều khiển. Khi trạng thái hệ đi vào miền nào thì chỉ cần dùng hệ số đã có để tính ra tín hiệu điều khiển. Tín hiệu điều khiển lúc này chỉ là một tín hiệu affine.

Trong trường hợp cửa sổ là vô hạn, định lý sau đây được sử dụng

***Định lý 1:*** Tồn tại một giá trị cửa sổ N (x(t)) mà với mọi N > N(x(t)) thì bài toán (8) và (9) là tương đương.

Định lý chỉ ra rằng với giá trị N đủ lớn thì (12) cũng là nghiệm của bài toán (9). Phương pháp tính N cũng đã được nghiên cứu. Chi tiết xem trong [1] và các tài liệu tham khảo được liệt kê.

**3.2. Tính ổn định và khả thi**

Nghiệm của bài toán (9) đã được chứng minh là ổn định và khả thi do tính chất của bộ điều khiển MPC với cửa sổ vô hạn. Đình lý 1 nói trên chỉ ra rằng bài toán (8) cũng có thể đảm bảo được sự ổn định khi sử dụng cửa sổ N đủ lớn. Một biện pháp khác để đảm bảo tính ổn định của bài toán (8) là sử dụng hàm tổn thất cuối và giới hạn miền trạng thái cuối. Hàm tổn thất cuối đóng vai trò như một giá trị bù cho phiếm hàm mục tiêu của (8) so với (9). Vì trong (8) cửa sổ chỉ đến N nên phần tổng chi phí từ N+1 đến vô cùng bị hụt, hàm tổn thất cuối này chính là để bù cho giá trị đó. Miền giới hạn cuối ép rằng giá trị cuối cùng của trạng thái trong mỗi bước tính phải thuộc tập XN. Tập này thường được chọn là tập điều khiển bất biến (Positive Control Invariance Set). Ý nghĩa của tập này là từ mọi trạng thái xuất phát trong tập này luôn tồn tại một tín hiệu điều khiển thỏa mãn các giới hạn đưa hệ trở về gốc tọa độ. Qua việc chọn tập giới hạn cuối như vậy tính ổn định và tính khả thi của hệ (8) được đảm bảo.

**4. Phương pháp học máy tăng cường**

Phương pháp học máy tăng cường cũng giải bài toán (9) và đưa ra tín hiệu điều khiển u(t) = h(x(t)), đo đạc trạng thái mới x(t+1), đánh giá hàm chi phí r(x(t),u(t)) và cải thiện tín hiêu điều khiển h dựa trên đánh giá đó. Điều khác biệt là mô hình của hệ (9b) không hề được biết.

**4.1. Phương pháp quy hoạch động (Dynamic Programming) [5], [6]**

Bằng cách tách riêng số hạng thứ nhất khỏi tổng, phương trình (9) có thể viết dưới dạng như sau

$V\_{\infty }\left(x\left(t\right)\right)=r\left(x\left(t\right),u\left(t\right)\right)+\sum\_{k=0}^{\infty }γ^{k}r\left(x\_{t+k+1},u\_{t+k+1}\right)$ (13)

Nếu thay u(t) bằng tín hiệu điều khiển u(t) = h(t) và thực chất số hạng tổng chính là $V\_{\infty }\left(x\left(t+1\right)\right)$, ta có thể viết (13) dưới dạng

$V\_{\infty }\left(x\left(t\right)\right)=r\left(x\left(t\right),h\left(x\left(t\right)\right)\right)+γ^{k}V\_{\infty }\left(x\left(t+1\right)\right)$ (14)

Phương trình (14) chính là phương trình Bellman. Lời giải của phương trình Bellman chính là giá trị tối ưu của $V\_{\infty }\left(x\left(t\right)\right)$ và hàm số h(x(t)) tương ứng để đạt được giá trị tối ưu đó

$V\_{\infty }^{}\left(x\left(t\right)\right)=min\_{h}r\left(x\left(t\right),h\left(x\left(t\right)\right)\right)+γ^{k}V\_{\infty }\left(x\left(t+1\right)\right)$ (15)

Nguyên tắc tối ưu của Bellman phát biểu rằng “Một quỹ đạo tối ưu, cho dù các quyết định phía trước như thế nào, thì phần cuối của nó phải là một quỹ đạo tối ưu xuất phát từ vị trí hiện tại”. Áp dụng nguyên tắc này cho phương trình (20), ta có

$V\_{\infty }^{}\left(x\left(t\right)\right)=min\_{h}r\left(x\left(t\right),h\left(x\left(t\right)\right)\right)+γ^{k}V\_{\infty }^{}\left(x\left(t+1\right)\right)$ (16)

Phương trình (16) được gọi là phương trình Hamilton-Jacobi-Bellman (hoặc phương trình tối ưu Bellman). Tín hiệu điều khiển tối ưu chính là

$h^{}\left(x\left(t\right)\right)=argmin\_{h}r\left(x\left(t\right),h\left(x\left(t\right)\right)\right)+γ^{k}V\_{\infty }\left(x\left(t+1\right)\right)$ (17)

Phương trình tối ưu Bellman (16) có thể giải bằng các phương pháp: lặp giá trị (value iteration), lặp tín hiệu điều khiển (policy iteration) tương ứng vói hai phương pháp lớn để của RL. Một cách tiếp cận khác là sử dụng cấu trúc hành động - đánh giá (actor - critic structure). Bài báo này sẽ đề cập sâu về cấu trúc hành động - đánh giá vì nó minh họa rõ được phương pháp học máy tăng cường, đồng thời có thể kết hơp với MPC.

**4.2. Cấu trúc Hành động - đánh giá (Actor - Critic structure)**

Ý tưởng của cấu trúc hành động - đánh giá

được minh họa trên hình 1 - [3] [5]. Khâu hành động đưa ra tín hiệu điều khiển tới đối tượng điều khiển. Khâu đánh giá nhận giá tri trạng thái đo đạc được, đánh giá lại hàm mục tiêu $V\_{\infty }\left(x\left(t\right)\right)$, dựa trên giá trị này để cập nhật tín hiệu điều khiển u = h(t). Quá trình này lặp đi lặp lại cho đến khi $V\_{\infty }$ và h hội tụ. Như vậy, bản chất là khâu đánh giá cố gắng xấp xỉ giá trị $V\_{\infty }\left(x\left(t\right)\right)$ trong khi khâu hành động cố gắng xấp xỉ hàm h\*(t). Việc xấp xỉ này có thể sử dụng nhiều phương pháp khác nhau, trong đó một phương án thường sử dụng là dùng mạng neural để học hai hàm số này. Phần tiếp theo sẽ nói về việc dùng mạng neural để xấp xỉ hai hàm số nói trên.

Hình 1: Cấu trúc Hành động - Đánh giá (actor - critic)

**4.2.1. Mạng neural cho khâu đánh giá**

Từ phương trình (14), ta có thể viết lại dưới dạng hàm sai lệch như sau

$e\_{c}\left(t\right)=r\left(x\left(t\right),h\left(x\left(t\right)\right)\right)+γ^{k}V\_{\infty }\left(x\left(t+1\right)\right)-V\_{\infty }\left(x\left(t\right)\right)$. (18)

Đây là sai lệch giữa giá trị thật từ đáp ứng hệ thống và giá trị ước lượng của khâu đánh giá. Hiển nhiên ta mong muốn sai lệch này càng nhỏ dần và tiệm cận về 0. Vì vậy, hàm mất mát (loss function) của mạng neural khâu đánh giá sẽ là

$E\_{c}\left(t\right)=0.5e\_{c}^{2}\left(t\right)$. (19)

Trọng số wc của mạng neural cho khâu đánh giá được cập nhật theo công thức sau

$w\_{c}\left(t+1\right)=w\_{c}\left(t\right)+δw\_{c}\left(t\right)$, $δw\_{c}\left(t\right)=-l\_{c}\frac{∂E\_{c}\left(t\right)}{∂w\_{c}\left(t\right)}$, $\frac{∂E\_{c}\left(t\right)}{∂w\_{c}\left(t\right)}=\frac{∂E\_{c}\left(t\right)}{∂V\_{\infty }\left(t\right)}\frac{∂V\_{\infty }\left(t\right)}{∂w\_{c}\left(t\right)}$ , (20)

với $l\_{c}>0$ là bước tính của phương pháp gradient.

**4.2.2. Mạng neural cho khâu hành động**

Hàm mất mát (loss function) của mạng neural khâu hành động có thể chọn là

$E\_{a}\left(t\right)=0.5e\_{a}^{2}\left(t\right)$, với $e\_{a}\left(t\right)=V\_{\infty }\left(x\left(t\right)\right)$ (21)

Trọng số wa của mạng neural cho khâu hành động được cập nhật theo công thức sau

$w\_{a}\left(t+1\right)=w\_{a}\left(t\right)+δw\_{a}\left(t\right)$, $δw\_{a}\left(t\right)=-l\_{a}\frac{∂E\_{a}\left(t\right)}{∂w\_{a}\left(t\right)}$, $\frac{∂E\_{a}\left(t\right)}{∂w\_{a}\left(t\right)}=\frac{∂E\_{a}\left(t\right)}{∂V\left(x\left(t\right)\right)}\frac{∂V\left(x\left(t\right)\right)}{∂w\_{a}\left(t\right)}$, (22)

với $l\_{a}>0$ là bước tính của phương pháp gradient.

**5. Kết luận và định hướng nghiên cứu kết hợp MPC và RL**

Bài báo đã nêu những điểm cơ bản về hai phương pháp điều khiển tối ưu nổi trội hiện nay. Hai phương pháp này về bản chất là hai hình thức xấp xỉ khác nhau để giải phương trình Bellman. Trong trường hợp MPC, do mô hình toán của hệ đã biết nên thay vì tìm giá trị tối ưu của $V\_{\infty }\left(x\left(t\right)\right)$ với cửa sổ N vô hạn, MPC xấp xỉ bằng cửa sổ hữu hạn và bù lại bằng hàm mất mát cuối. Với RL, do mô hình chưa biết nên phải sử dụng phương pháp lặp để hội tụ về giá trị tối ưu của $V\_{\infty }\left(x\left(t\right)\right)$. Việc xấp xỉ hàm $V\_{\infty }\left(x\left(t\right)\right)$ có thể thực hiện thông qua sử dụng mạng neural.

Hai cách tiếp cận này có tiềm năng bổ sung cho nhau. Việc xấp xỉ $V\_{\infty }\left(x\left(t\right)\right)$ trong MPC có thể được cải thiện nếu sử dụng phương pháp lặp trong RL. Ngược lại, MPC có thể là một công cụ để giúp RL đảm bảo được tính bền vững, tính ổn đinh đông thời thỏa mãn các giới hạn của trạng thái và tín hiệu điều khiển.

**Tài liệu tham khảo**

1. Daniel Gorges (2017): *Relation between Model Predictive Control and Reinforcement Learning*, IFAC.
2. Basil Kouvaritakis, Mark Cannon (2016): *Model Predictive Control*, Nhà xuất bản Springer.
3. Dimitri P. Bertsekas (2019): *Reinforcement Learning and Optimal Control,* Nhà xuất bản Athena Scientific.
4. Rolf Findeisen, Frank Allgöwer, Lorenz T. Biegler (2007): *Assessment and Future Directions of Nonlinear Model Predictive Control (Lecture Notes in Control and Information Sciences),* Nhà xuất bản Springer.
5. Frank L. Lewis, Draguna Vrabie, Vassilis L. Syrmos (2012):  *Reinforcement Learning and Optimal Adaptive Control,* Nhà xuất bản John Wiley & Sons.
6. Richard S. Sutton, Andrew G. Barto (2018): *Reinforcement Learning: An Introduction (Adaptive Computation and Machine Learning),* Nhà xuất bản MIT Press.