REINFORCEMENT LEARNING

# MATLAB

## Adaptive cruise control (ACC)

* Định nghĩa: là một hệ thống kiểm soát hành trình thông minh được sử dụng trong ô tô để tự động điều chỉnh tốc độ của xe dựa trên khoảng cách với phương tiện phía trước.
* Một hệ thống cơ bản có hai chế độ điều khiển:
* nếu như khoảng cách với xe phía trước lớn hơn D\_safe – điều khiển để đạt vận tốc mong muốn
* nếu như khoảng cách với xe phía trước nhỏ hơn D\_safe – điều khiển để giữ khoảng cách an toàn

## Ví dụ: Train RL Agent for Adaptive Cruise Control with Constraint Enforcement

* Ứng dụng giải quyết vấn đề ràng buộc:

A math symbols and symbols

AI-generated content may be incorrect.

* Phase 1:
* Thu thập dữ liệu: tạo ra 1 agent để sinh ra các action (gia tốc đặt) từ khoảng [-10;6] ngẫu nghiên và quan sát các giá trị trả về
* Vẫn theo 2 chế độ điều khiển: kiểm tra khoảng cách an toàn

Nếu an toàn -> vận tốc mong muốn (v\_set)  
Vi phạm an toàn -> vận tốc = min(v\_set,v\_lead)

* Sau khi xác định vận tốc ref

Tính hàm reward:   
P =

Trong đó – là gia tốc thực đo được, – là hằng số phạt, nếu sai số vận tốc () >= 0.25 thì bằng 1, ngược lại bằng 0

* Mong muốn minimize ⬄ tức ổn định vận tốc đúng với vận tốc mong muốn, cũng là không cần thay đổi gia tốc.
* Điều kiện kết thúc 1 tập học (kết thúc trò chơi):  
  Khi hai xe va chạm

Khi vận tốc xe ego < 0

* Truyền lại các giá trị đo về khoảng cách, vận tốc xe trước, vận tốc xe ego và gia tốc xe ego => đây là 4 giá trị quan sát.
* Sau đó gọi hàm .m để lưu giữ liệu gồm 9 cột: 4 giá trị quan sát hiện tại, action đưa ra, 4 giá trị quan sát tiếp theo
* Mục tiêu: Tìm mô hình tuyến tính để dự đoán khoảng cách tương đối và vận tốc ego từ trạng thái và đầu vào.
* Phase 2:
* Sau khi xuất ra được mô hình tuyến tính
* Sử dụng được block constraint enforcement
* Train agent
* Sau khi train xong test với

Kết quả ví dụ:  
A screenshot of a computer

AI-generated content may be incorrect.

## Ví dụ: Train Reinforcement Learning Agent with Constraint Enforcement

* Mục tiêu bài toán: đưa quả bóng màu xanh lá cây đến gần nhất có thể với vị trí quả bóng đỏ bị thay đổi liên tục.
* Ràng buộc cần giải quyết:

Vị trí quả bóng xanh lá => dùng constraint enforcement

Vận tốc quả bóng xanh [-1;1] => điều kiện đặt trước cho agent

* !!! Hơn nữa, giá trị quan sát vị trí quả bóng đỏ cho agent sẽ bị ảnh hưởng bởi nhiễu.
* Hàm reward:

Trong đó, là vị trí quả bóng đỏ, nếu reward < 0 – sẽ tính reward = 0

* Kết thúc 1 lần học (episode): vị trí quả bóng xanh vượt khỏi ngưỡng, tức x < 0 hoặc x > 1
* Phase 1:  
  Tương tự như ví dụ ACC, phase 1 sẽ là khiến agent đưa ra các action ngẫu nghiên [1;1]
* 3 giá trị quan sát là vị trí quả bóng xanh, tín hiệu điều khiển đưa ra, vị trí quả bóng đỏ

!!! Khó hiểu: mục tiêu của phase 1 là để đưa về dạng chuẩn fx +gx\*u = c để dùng block constraint enforcement nhưng ở đây đầu ra lại là sai lệch vị trí 2 quả bóng chia cho u (đúng ra phải là vị trí tiếp theo của quả bóng xanh. Hơn nữa, trông simulation với RL, network học contrainst không được sử dụng (?)

**Chú ý: trong matlab, các tín hiệu đưa ra ( tín hiệu điều khiển) đều được chặn từ trước, bài toán tối ưu tập trung vào các trạng thái (giá trị trả ra từ env).**

Create RL Agent

## DDPGAgent:

### Ví dụ DDPGAgentBall và DDPGAgentACC:

* Đều là Agent DDPG với actor-critic kiến trúc