Chúng ta muốn tác tử khám phá để tìm ra những sự tahy dổi của môi trường, nhưng khong làm giảm hiệu năng đi quá nhiều. Như trong sự xung đột giữa khám phá và khai thác trước đó, có lẽ không có giải pháp nào là hoàn hảo va thực tế, nhưng các phương pháp phỏng đoán đơn giản thường hiệu quả.

Tác tử Dyna Q+ đã đưa ra đường đi ngắn sử dụng môtj heristic như vậy. Tác tử đó giữ sự theo dõi cho mỗi cặp state-action của hành trình có bao nhiêu bước đã đi kể từ khi cặp này được thử lần cuối trong một tương tác thực sự với môi trường. Thời gian trôi qua càng nhiều, càng lớn (chúng ta có thể đoán) khả năng động của cặp này đã thay đổi và mô hình của nó không chính xác. Đêr khuyến khích hành vi kiểm tra cacs hành động chưa được thửu nghiệm, phần thưởng đặc biệt được trao cho các thử nghiệm mô phỏng liên quan đến hành dộng này. Đặc biệt, nếu phần thưởng được mô hình hóa cho một chuyển tiếp là r, và quá trình chuyển tiếp chưa được thử trong các bước thời gian, thì việc lập kế hoạch cập nhật được thực hiện như thể quá trình chuyển tiếp đó tạo ra phần thưởng r + kcawnt, với một số nhỏ k. Điều này sẽ khuyến khính tác nhân tiếp tục kiểm tra tất cả các chuyển đổi trạng thái có thể truy cập và thậm chí tìm các chuỗi hành dộng dài để thực hiện các thử nghiệm như vậy. Tất nhiến tất cả các thử nghiệm này đều có chi phó của nó, nhưng trong nhiều trường hợp, như mê cung lối tắt, loại tò mò tính toán này rất đáng để khám phá thêm.

8.4> Ưu tiên quét

Trong tác tử Dyna được thể hiện ở trong mục trước, chuyển tiếp mô phỏng được bắt đầu ở cặp state-action được chọn một cách thống nhất ngẫu nhiên từ tất cả các cặp kinh nghiệm trước đó. Nhưng sự lựa chọn thống nhất thường không phải tốt nhâts; planing có thể hiệu quả hơn nhiều nếu mô phỏng chuyển tiếp và cập nhật được tập chung vào các cặp hành động trạng thái đặc biệt. Ví dụ, xem xét điều gì xảy ra trong hành trình thứ hai của nhiệm vụ mê cung đầu tiên. Lúc bắt đầu của hành trình thứ 2, chỉ cặp state-action trực tiếp dẫn đến giá trị đích có giá trị dương, giá trị của các cặp khác vẫn bằng 0. Điều này có nghĩa là thâtj vô nghĩa khi thực hiện cập nhật cùng với hầu hết các chuyển đổi, bởi vì họ đưa tác nhân từ trạng thái có giá trị 0 sang trạng thái khác, và do đó cá bản cập nhật sẽ không có ảnh hưởng. Chỉ một cập nhật dọc theo một chuyển tiếp vào trạng thái ngay trước mục tiêu, hoặc từ nó sẽ thay đổi bất kì giá trị nào. Nếu mô phỏng chuyển tiếp được sinh ra 1 cách thống nhâts, sau đó nhiều cập nhật lãng phí sẽ được thực hiện trước khi tình cờ gặp một trong những cập nhật hữu ích. Khi kế hoạch tiến hành, các vùng của các cập nhật hữu ích tăng lên, nhưng việc lập kế hoạch vẫn hiệu quả hơn so với việc tập trung vào nơi nó sẽ làm tốt nhất. Trong các vấn đề lớn hơn nhiều đó là mục tiêu của chúng tôi, số lượng trạng thái quá lớn đến nỗi một tìm kiếm không tập trung sẽ cực kỳ thiếu hiệu quả. Ví dụ này cho thấy tìm kiếm kiếm tập trung hữu ích bằng cách backward từ trạng thái đích. Tất nhiên, chúng tôi không thực sự muốn sử dụng một số phương pháp đặc biêt cho ý tưởng “good state”. Chúng tôi muốn phương pháp đó làm việc cho hàm phần thưởng chung. Trạng thái đích sẽ là 1 trường hợp đặc biệt, thuận tiện cho việc kích thích trực giác. Nhìn chung, chúng ta muốn back không phải từ trạng thái đích nhưng từ bất kỳ trạng thái có giá trị đã thay đổi. Giả sử giá trị đó ban đầu đúng cho mô hình, như các giá trị trong ví dụ maze trước để khám phá đích. Giả sử bây giờ tác tử khám phá một thay đổi trong môi trường và thay đổi giá trị ước tính của 1 trạng thái, cả up và down. Điển hình là, điều đó sẽ ám chỉ rangừ của nhiều trạng thái khác cũng nên được thay đổi, nhưng chỉ những cập nhật one-step hữu ích là những hành động dẫn trực tiếp đến trạng thái có giá trị thay đổi. Nếu giá trị của hành động đã được cập nhật, sau đó giá trị của các trạng thái trước có có thể thay đổi lần lượt. Nếu vậy, sau đó các hành động dẫn đến chúng cần được cập nhật, và sau đó các trạng thái trước của chúng có thể được thay đổi. Theo cách này người ta có thể backwark từ các trạng thái tùy ý mà chúng đã có thay đổi về giá trị, hoặc thực hiện cập nhật hữu ích hoạc chấm dứt sự lan truyền. Ý tưởng chung đó có thể gọi là backwork focusing planing computations. Khi biên giới của các cập nhật hữu ích lan truyền backward, nó thường tăng nhanh, tạo ra nhiều cặp state-acti có thể được cập nhật một cách hữu ích.

Nhưng không phải tất cả trong số đó sẽ hữu ích như nhau. giá trị của một số trạng thái có thể được thay đổi rất nhiều, trong khi những trạng thái khác có thể thay đổi rất ít. Những cập trước của những trang thái thay đổi nhiều có khả năng cũng thay đổi rất nhiều. Trong một môi trường ngẫu nhiên, các biến thể trong xác suất chuyển tiếp ước tính cungx góp phần vào các biến thể trong kích thước của các thay đổi và mức độ quan trọng mà các cặp cần được cập nhật. Đó là điều tự nhiên để ưu tiên các bản cập nhật theo một thước đo mức độ quan trọng của chúng, và thực hiện theo thứ tự ưu tiên. Đó là ý tưởng đăng sau prioritized sweeping. Một hàng đợi được duy trì cho mỗi cặp state-action có giá trị ước tính sẽ thay đổi không cần thiết nếu được cập nhật, ưu tiên bởi kích thước của sự thay đổi. Khi cặp ở đầu của hàng đợi được update, sự ảnh hưởng trên mỗi cặp trước nó được tính toán. Nếu sự ảnh hưởng tốt hơn một số ngưỡng nhỏ, sau đó cặp được chèn vào trong queue với độ ưu tiên mới (nếu có một entry trước của cặp ở trong hàng đợi, sau khi chèn kết quả chỉ những entry có độ ưu tiên cao hơn còn lại ở trong queue). Theo cách này, các ảnh hưởng của những thay đổi được lan truyền 1 cách hiệu quả cho đến khi không hoạt động. Thuật toán đầy đủ cho trường hợp môi trường xác định đươcj đưa ra ở box .

Mở rộng của priorituzed sweeping đến môi trương ngẫu nhiên khá đơn giản. Model được duy trì bằng cách giữ số lần mỗi cặp state-action có kinh nghiệm và về những trạng thái tiếp theo. Nó là tự nhiên sau khi cập nhật từng cặp không phải với 1 cập nhật mẫu, như chúng ta đã sử dụng cho đến nay, nhưng với 1 cập nhâtj được mong đợi, có tính đến tất cả các trạng thái tiếp theo có thể và xác suất xảy ra của chúng.

Prioritized sweeping chỉ là 1 cách phân phối tính toán để cải thiện hiệu quả planning, và xác suất không phải cách tốt nhất. Môtj trong những hạn chế prioritized sweeping là nó sử dụng các banr cập nhât dự kiến,

trong môi trường ngẫu nhiên có thể lãng phí rất nhiều tính toán trên các chuyển xác suất thấp. như chúng ta show ra ở mục dưới,

ex8.5: Prioritized sweeping for Rod Maneuvering

Mục tiêu trong nhiệm vụ này là điều khiển 1 rod xung quang một số chướng ngại vật được đặt ngẫu nhiên trong một không gian làm việc hình chữ nhật giới hạn đến vị trí mục tiêu với số bước ít nhất. Rod có thể dịch chuyển theo trục dài hoạc vuông góc với trục đó hoặc nó có thể được xoay theo một trong 2 hướng xung quanh tâm của nó. Khoảng cách của mỗi chuyển động là khoảng 1/20 không gian làm việc. và mức tăng của việc xoay là 20 độ. Các bản dịch được xác định và định lượng đến trong 20x20 vị trí. Hình bên phải cho thấy sự cản trở và sholution ngắn nhất từ vị trí bắt đầu đến đích, tìm bởi prioritized sweeping. Vấn đề này mang tính quyết định, nhưng có bốn hành động và 14.400 trạng thái tiềm năng( Một số trong số ngày là không thể truy cập được vì những chướng ngại vật). Vấn đề này có lẽ quá lớn để có thể giải quyết nằng các phương pháp không đượcc chuẩn hóa.

cập nhật mẫu trong nhiều trường hợp có thể tiến gần hơn đến hàm giá trị thực với ít tính toán hơn mặc dù phương sai đươc đưa ra bằng cách lấy mẫu. Các bản cập nhật mẫu có thể win vì chúng chia tính toán backing-up tổng thể thành các phần nhỏ hơn - Các công việc tương ứng với chuyển tiếp cá nhân – mà sau đó cho phép nó được tập trung hẹp hơn vào các phần sẽ có tác động lớn nất. Ý tưởng đó đã được đưa đến những gì có thể là giới hạn logic của nó trong “small backups” được giới thiệu bởi Seijen và Sutton. Đaya là những cập nhật cùng với quá trình chuyển tiếp đơn, giống như bản cập nhật mẫu, nhưng dựa trên xác suất chuyển tiếp ngoài mẫu, như một cập nhật được mong đợi. Bằng cách chọn thứ tự thực hiện các cập nhật nhỏ, có thể cải thiện đang kể hiệu quả lập kế hoach vượt quá khả năng với prioritized sweeping.

Chúng tôi đã đưa ra ở trong chương tất cả các loại lập kế hoạch không gian trạng thái có thể được xem như trình tự của việc cập nhật giá trị, chỉ khác nhau về loại cập nhật, dự kiến hoặc mẫu, lớn hay nhỏ và theo thứ tự cập nhật. Trong mục đó chúng ta đã nhấn mạnh backward focusing, nhưng đó chỉ là một chiến lược. ví dụ, một cách khác là tập trung vào các trạng thái theo mức độ dễ dàng có thể đạt được từ các trạng thái đã thăm thường xuyên theo chính sách hiện tại, nó được gọi là forward focusing.

8.5> Cập nhật dự kiến với cập nhật mẫu

Các ví dụ ở phần trước đưa ra 1 số ý tưởng của các khoảng xác suất cho phương pháp kêt hợp giữa planning và learning. Trong chuong đó, chúng tôi phân tích một số các ý tưởng thành phần liên quan, đang băt đầu với những lợi thế tương đối của các bản cập nhật dự kiến và mẫu

Phần lớn cuống sách này là về các loại cập nhật hàm giá trị khác nhau và chúng tôi xem xét rất nhiều loại tuyệt vời. Tập trung cho thời điểm này tcho on-step updates, chúng thay đổi chủ yếu dọc theo ba chiều nhị phân. Đầu tiên 2 chiều là họ cập nhật giá trị trạng thái và hành động và họ ước tính giá trị cho chính sách tối ưu. Hai chiều này tạo ra bốn lớp cập nhật để xấp xỉ bốn hàm giá trị q\*, qn, v\*, vn,

8.6 Trajectoty Sampling

Trong mục trước, chúng ta so sánh 2 cách phân phối cập nhất. Cách tiếp cận cổ điển, từ quy hoạch động, để biểu diễn sweeps thông qua toàn bộ không gian trạng thái(hoặc state-action), cập nhật mỗi trạng thái (hoặc cặp state-action) sau mỗi lần quét. Đây là vấn đề với task lớn bởi vì nó có thể không đủ thời gian để hoàn thành ngay cả 1 lần quét. Trong nhiều task đa số phần lớn các trạng thái là không liên quan đến nhau bởi vì chúng được thăm theeo các chính sách rất kém hoặc với xác suất rất thấp. Quét toàn diện hoàn toàn dành thời gian bằng nhau cho tất cả các phần của không gian trạng thái thay vì tập trung vào nơi cần thiết. Như chúng ta thảo luận ở chương 4, quét toàn diện và đối xử bình đẳng của tất cả trạng thái rằng chúng ngụ ý không phải là thuộc tính cần thiết của lập trình động. Về nguyên tắc, cập nhật có thể được phân phối bất kỳ cách nào ta thích (để đảm bảo sự hội tụ, tất cả trạng thái hoặc cặp state-action phải được thăm trong giới số lần lặp vô hạn; mặc dù một ngoại lệ cho điều này được thảo luận trong phần 8.7 bên dưới), nhưng trong luyện tập, quét toàn diện thường được sử dụng.

Cách tiếp cận thứ 2 là lấy mẫu từ không gian trạng thái hoặc state-action theo như một số phân phối. Người ta có thể lấy mẫu đồng đều, như trong tác tử Dyna-Q, nhưng điều này sẽ bị một vấn đề tương tự như quét toàn diện. hấp dẫn hơn là phân phối các bản cập nhật theo phân phối chính sách, đó là, theo như phân phối được quan sát khi tuân theo chính sách hiện tại. Một lợi thế của phân phối này là nó dễ dàng được tạo ra, 1 cách đơn giản là tương tác với model, theo chính sách hiện tại. Trong 1 task hành trình, người ta bắt đầu ở trạng thái bắt đầu (hoặc theo phân phối starting-state) và mô phỏng cho đến trạng thái cuối cùng. Trong task tiếp theo, người ta bắt đầu ở bất kỳ đâu và chỉ cần mô phỏng. Trong cả 2 trường hợp, chuyển trạng thái mẫu và rewards đưa ra bởi mô hình, và các hành động mẫu đưa ra bởi chính sách hiện tại. Nói cách khác, một mô phỏng trajectories riêng biệt rõ ràng và thực hiện cập nhật taij trạng thái hoặc cặp state-action được bắt gặp trên đường. Chúng ta gọi cách đó là tạo ra kinh nghiệm và cập nhật trajectory sampling. Thật khó để tưởng tượng bất kỳ cách phân phối cập nhật nào theo cách phân phối chính sách khác với trajectory sampling. Nếu ai đó có một biểu diễn rõ ràng về phân phối chính sách, sau đó họ có thể quét thông qua tất cả trạng thái, trọng số cập nhật của mỗi trạng thái theo như phân phối chính sách, nhưng điều này khiến chúng ta một lần nữa với tất cả các chi phí tính toán của việc quét toàn diện. Người ta có thể lấy mẫu và caaj nhật các cặp state-action riệng biệt từ phân phối, nhưng ngay cả khi điều đó có thể hoàn thành 1 cách hiệu quả, những lợi ích này sẽ cung cấp trên quỹ đạo mô phỏng? Ngay cả khi biết phân phối chính sách ở dạng rõ ràng là không thể. Phân phối thay đổi bất cứ khi nào chính sách thay đổi, và tính toán phân phối yêu cầu tính toán tương đương với đánh giá chính sách hoàn chỉnh. Việc xem xét các khả năng như vậy làm cho việc trajectory sampling vừa hiệu quả vừa thanh lịch.

Phân phối chính sách của cập nhật có tốt không? Theo trực giác nó có vẻ như là một lựa chọn tốt, ít nhất là tốt hơn so với phân phối đồng nhất. Ví dụ, nếu bạn đang học đánh cờ, bạn học các vị trí có thể phát sinh trong các trò chơi thực sự, không phải vị trí ngẫu nhiên của quân cờ. Cái sau có thể là trạng thái hợp lệ, nhưng để có thể đạt được giá trị chính xác của chúng là 1 kỹ năng khác từ việc đáng giá các vị trí trong các trò chơi thực. Chúng ta cũng sẽ thấy trong phânf 3 rằng phân phối xác suất chính sách có những lợi thế đáng kể khi sử dụng xấp sỉ hàm. Cho dù sử dụng xấp xỉ hàm hay không, người ta có thể mong đợi tập trung vào chính sách để cải thiện đáng kể tốc độ lập kế hoạch.

Tập trung vào phân phối chính sách có thể có lợi vì nó khiến các phần không gian rộng lớn, không thú vị được bỏ qua hoặc có thể gây bất lợi vì nó khiến các phần cũ của không gian được cập nhật lặp đi lặp lại. Chúng tôi đac tiến hành 1 thí nghiệm nhỏ để đánh giá hiệu quả theo kinh nghiệm. Để tách biệt hiệu quả của phân phối cập nhật, chúng tôi đã sử dụng hoàn toàn các cập nhật tabular dự kiến one-step, như được định nghĩa bởi 8.1. Trong trường hợp đồng nhất, chúng tôi đã đi qua tất cả các cặp state-action, cập nhật từng cặp tại chỗ, và trong trường hợp on-policy, chúng tôi đã mo phỏng hành trình, tất cả bắt đầu trong cùng 1 trạng thái, cập nhật mỗi cặp state-action xảy ra theo chính sách e-greedy.(e = 0,1). Các task là task hành trình, được tạo ra ngẫu nhiên như sau. Từ mỗi trạng thái trong tập S, 2 action là có thể, một trong số đó dẫn đến một trong các trạng thái tiếp theo, tất cả đều có khả năng như nhau, với sự lựa chọn ngẫu nhiên các trạng thái b cho mỗi cặp state-action. Yếu tố phân nhánh b là giống nhau cho tất cả cặp state-action. Thêm vào đó, trên tất cả các chuyển tiếp chúng có một xác xuất 0.1 của chuyển tiếp để đến trạng thái đichs, kết thúc hành trình. Phần thưởng mong đợi trên mỗi chuyển tiếp được chọn từ phân phối gaus với mean = 0 và phương sai = 1. Tại bất kỳ thời điểm nào ở trong quy trình lập kế hoạch, người ta có thể dừng và tính toán Vn(So), giá trị đúng của trạng thái bắt đầu dưới chính sách tham lam, n, giữ hàm giá trị hành đọng Q, như một dấu hiệu cho thấy tác nhân sẽ làm tốt như thế nào trong một hành trình mới mà nó đã hành động một cách tham lam(tất cả trong khi giả sử mô hình là chính xác).

Phần trên của hình bên phải hiển thị kết quả trung bình hơn 200 task mẫu với 1000 trạng thái vầ các yếu tố phân nhánh của 1, 3, và 10. Chất lượng của chính sách được tìm thấy được vẽ như một hàm của số lượng dự kiến mong đợi đã hoàn thành. Trong tất cả trường hợp, lấy mẫu theo như phân phối on-policy dẫn đến việc lập kế hoạch ban đầu nhanh hơn và lập kế hoạch chậm lại trong thời gian dài. Hiệu quả mạnh hơn, và thời gian của kế hoạch nhanh hơn là dài hơn, tại các yếu tố phân nhánh nhỏ. Trong các thí nghiệm khác nhau, chúng tôi htaays rằng những ảnh hưởng này cxg trở nên mạnh mẽ hơn khi số lượng trạng thái tăng lên. Ví dụ, phần dưới của hình biểu thị kết quả cho một yếu tố phân nhánh của 1 cho 1 task với 10000 trạng thái. Trong trường hợp này, lợi thế của việc tập trung vào chình sách là lơn và lâu dài.

Tất cả những kêts quả này có ý nghĩa. Trong ngắn hạn, lấy mẫu theo phân phối on-policy sẽ giúp bằng cách tập trung vào các trạng thái gần con chau của trạng thái bắt đầu. Nếu có nhiều trạng thái cà một yếu tố phân nhánh nỏ, hiệu quả đó sẽ lớn và lâu dài. Về lâu dài, tập trung vào phân phối on-policy có thể tổn thương, bởi vì các trạng thái thường xảy ra đều có giá trị chính xác của chúng. Lấy mẫu chúng là vô ích, trong khi lấy mẫu các trạng thái khác thực sự có thể thực hiện một số công việc hữu ích. Đây có lẽ là lý do tại sao cách tiếp cận toàn diện, không tập trung tốt hơn về lâu dài, ít hơn cho vấn đề nhỏ. Những kết quả này không được kết luận vì chúng chỉ dành cho các vấn đề được tạo theo một cachs ngẫu nhiên, cụ thể, nhưng chúng cho thấy việc lấy mẫu theo phân phối on-policy có thể là một lợi thế lớn cho các vấn đề lớn, đặc biệt là các vấn đề trong đó một tập con của không gian state-action dược thăm dựa trên phân phối on-policy