Sztuczna inteligencja : Uczenie maszynowe : Probabilistyczne

(XIII / XVI)

ĆWICZENIA

1. Utwórz ukryty model Markowa, aby przedstawić sekwencję punktacji w meczu futbolu amerykańskiego, w którym przyłożenia wynoszą 6 punktów, po których można wykonać próbę 0, 1 lub 2 dodatkowych punktów. Oczywiście są dwie drużyny i każda może strzelić gola. Jeśli drużyna ma piłkę i nie zdobywa punktów, emituje 0. Załóżmy więc, że emitowany strumień wyników to 6,1,6,0,3,0,6,1,0,3,0; jak wyglądałby Twój HMM?

a. Omów ten problem i jak może wyglądać najlepszy HMM.

b. Przetestuj swój HMM na dwóch dodatkowych strumieniach wyników.

c. Śledź aktualną grę i zobacz, jak dobrze Twój system przewiduje strumień punktacji.

2. Utwórz ukryty model Markowa, aby przewidzieć sekwencję punktów w połowie zmiany w amerykańskim meczu baseballowym. Załóżmy, że sekwencja połowy zmiany jest równa 0, 0, 0, 1, 0, 1, 1, 2, 2, 0, 0, 0, 0, 2, 0, 0, 0, 0. Dla uproszczenia ogranicz punktację dla każdego drużyna do 0, 1 lub 2 runów podczas ich połowy .

a. Jak zmieniłbyś model, aby zezwolić na dowolną liczbę przebiegów na połowę zmiany?

b. Jak możesz wytrenować swój system, aby uzyskać bardziej realistyczne wartości punktacji?

3. Utwórz figurę reprezentującą hierarchiczny, ukryty model Markowa z sekcji 13.1.2. Do jakiego typu sytuacji problemowej może być odpowiedni Twój HHMM? Omów kwestię dopasowania struktur HMM do domen aplikacji.

4. Biorąc pod uwagę przykład algorytmu Viterbiego przetwarzającego probabilistyczną maszynę skończoną z rysunków 7 i 8:

a. Dlaczego nowy jest postrzegany jako lepsza interpretacja niż kolano dla obserwowanych telefonów?

b. W jaki sposób algorytm Viterbiego obsługuje stany alternatywne w probabilistycznej maszynie skończonej, na przykład wybór telefonów uw i iy w słowie nowy?

5. Biorąc pod uwagę ukryty model Markowa i algorytm Viterbiego z Części 9, przeprowadź pełne śledzenie, w tym ustawienie odpowiednich wskaźników wstecznych, które pokażą, jak obserwacja #, n, iy, t, # byłaby przetwarzana.

6. Uruchom ręcznie robota opisanego w przykładzie procesu decyzyjnego Markowa w Rozdziale 13.3.3. Użyj tego samego mechanizmu nagrody i wybierz wartości probabilistyczne dla a i b do przetwarzania decyzji.

a. Uruchom robota ponownie z różnymi wartościami a i b. Jakie zasady dają robotowi największe szanse na nagrodę?

7. Zaprogramuj robota z sekcji 13.3.3 obsługiwanego przez MDP w wybranym przez siebie języku. Eksperymentuj z różnymi wartościami a i b, które mogą zoptymalizować nagrodę. Istnieje kilka interesujących możliwych polityk: Jeśli doładowanie jest zasadą A (wysokie), czy Twój robot nauczy się, że ta zasada jest nieoptymalna? W jakich okolicznościach robot zawsze szukałby pustych puszek, tj. Zasada A (niski) = ładowanie jest nieoptymalna?

8. Jack prowadzi salon samochodowy i szuka sposobu na maksymalizację swoich zysków. Co tydzień Jack zamawia zapasy samochodów kosztem d dolarów za samochód. Te samochody są dostarczane natychmiast. Nowe samochody zostaną dodane do jego ekwipunku. Następnie w ciągu tygodnia sprzedaje losową liczbę samochodów k po cenie c każdy. Jack ponosi również koszty u każdego niesprzedanego samochodu, który musi przechowywać w ekwipunku. Sformułuj ten problem jako proces decyzyjny Markowa. Jakie są stany i działania? Jakie są nagrody? Jakie są prawdopodobieństwa przejścia?

Opisz długoterminowy zwrot.

9. Rozważ ogólną dziedzinę zadań nawigacji w świecie siatki, w której występuje stan celu, przeszkody i współczynnik dyskontowy $\gamma < 1$. Działania są stochastyczne, więc agent może wślizgnąć się do innej komórki podczas próby ruchu. Istnieje pięć możliwych działań: idź na północ, południe, wschód, zachód lub pozostań w tym samym miejscu. Rozważmy sytuację, w której podczas uderzania o ściany ponoszone są ujemne koszty. Czy potrafisz narysować przykładowe środowisko 3x3, w którym najlepszą akcją w przynajmniej jednym stanie jest pozostanie? Jeśli tak, określ działania, nagrody i prawdopodobieństwa przejścia. Jeśli nie, wyjaśnij dlaczego.

10. Kiedy wychodzimy na kolację, zawsze lubimy parkować jak najbliżej restauracji. Załóżmy, że restauracja znajduje się na bardzo długiej ulicy biegnącej ze wschodu na zachód, która umożliwia parkowanie tylko z jednej strony. Ulica podzielona jest na odcinki o długości jednego samochodu. Do restauracji zbliżamy się od wschodu, zaczynając od jednostek D. Prawdopodobieństwo, że miejsce parkingowe w odległości s od restauracji jest wolne, wynosi ps, niezależnie od wszystkich innych miejsc. Sformułuj ten problem jako proces decyzyjny Markowa. Pamiętaj, aby określić wszystkie elementy swojego MDP!

11. Jak zmieniłbyś reprezentację MDP w sekcji 13.3 na POMDP? Weźmy prosty problem z robotem i jego macierz przejść Markowa utworzoną w sekcji 13.3.3 i zmieńcie to na POMDP. Wskazówka: pomyśl o zastosowaniu macierzy prawdopodobieństwa dla stanów częściowo obserwowalnych.

12. Omówiliśmy pokrótce grę karcianą Poker w sekcji 13.3. Biorąc pod uwagę, że obecny (probabilistyczny) stan gracza to albo dobry zakład, wątpliwy zakład, albo zły zakład, opracuj POMDP, aby przedstawić tę sytuację. Możesz omówić to, korzystając z prawdopodobieństwa rozdania pewnych możliwych układów pokerowych. Oblicz koszt złożoności znalezienia optymalnej polityki dla problemu POMDP

UCZENIE MASZYNOWE: PROBABILISTYCZNE

13.0 Stochastyczne i dynamiczne modele uczenia się

Jak zauważono we wcześniejszych częściach , istnieją dwa główne powody stosowania narzędzi probabilistycznych do zrozumienia i przewidywania zjawisk na świecie: po pierwsze, zdarzenia mogą być ze sobą rzeczywiście probabilistycznie powiązane, a po drugie, deterministyczne związki przyczynowe między sytuacjami w zmieniającym się świecie mogą być tak złożone, że ich interakcje najlepiej oddają modele stochastyczne. Społeczność AI przyjęła i wdrożyła modele probabilistyczne z obu tych powodów, a te stochastyczne technologie miały bardzo ważny wpływ na projekt, moc i elastyczność algorytmów uczenia maszynowego. Reguła Bayesa, przedstawiona po raz pierwszy w sekcji 5.3, jest podstawą probabilistycznych modeli uczenia maszynowego. Podejścia bayesowskie wspierają interpretację nowych doświadczeń w oparciu o wcześniej wyuczone relacje: zrozumienie obecnych wydarzeń jest funkcją wyuczonych oczekiwań z poprzednich wydarzeń. Podobnie modele Markowa i ich warianty stanowią podstawę stochastycznego podejścia do uczenia się. W łańcuchu Markowa prawdopodobieństwo wystąpienia zdarzenia w dowolnym momencie jest funkcją prawdopodobieństwa, z jakim zdarzenia wystąpiły w poprzednich okresach. W łańcuchu Markowa pierwszego rzędu, prawdopodobieństwo wystąpienia zdarzenia o godzinie t jest jedynie funkcją prawdopodobieństwa jego poprzednika w czasie t-1. Część 13 składa się z trzech sekcji, pierwsza sekcja 13.1 kontynuuje prezentację modeli Markowa wprowadzających ukryte modele Markowa (lub HMM) oraz kilka ważnych wariantów i rozszerzeń. Sekcja 13.2 rozszerza prezentację sieci bayesowskich, skupiając się zwłaszcza na dynamicznych sieciach bayesowskich (lub DBN) i kilku rozszerzeniach. Sekcja 13.3 kontynuuje prezentację uczenia się przez wzmacnianie, z dodaniem probabilistycznych miar wzmocnienia.

13.1 Ukryte modele Markowa (HMM)

Na początku XX wieku rosyjski matematyk Andrey Markov zaproponował matematykę do modelowania probabilistycznie powiązanych dyskretnych zdarzeń schodkowych. W porównaniu z deterministyczną sekwencją zdarzeń, jak można by się spodziewać, gdy komputer wykonuje ustalony zestaw instrukcji, formalizmy Markowa wspierały ideę, że każde zdarzenie może być probabilistycznie powiązane z wydarzeniami, które je poprzedzały, jak we wzorcach fonemów lub słów w wypowiedziane zdanie. W naszych próbach zaprogramowania komputerów do uczenia się zjawisk w zmieniającym się świecie spostrzeżenia Markowa okazały się niezwykle ważne.

13.1.1 Wprowadzenie i definicja ukrytych modeli Markowa

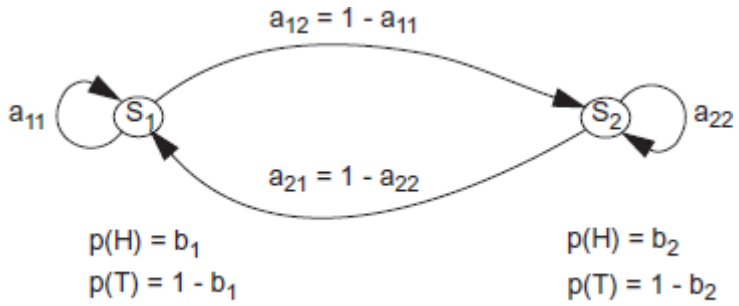
Ukryty model Markowa (lub HMM) jest uogólnieniem tradycyjnego łańcucha (lub procesu) Markowa. W łańcuchach Markowa widzianych do tej pory każdy stan odpowiadał dyskretnemu fizycznemu - i możliwemu do zaobserwowania - wydarzeniu, jak na przykład pogoda o określonej porze dnia. Ta klasa modeli jest dość ograniczona i teraz uogólniamy ją na szerszą klasę problemów. W tej sekcji rozszerzamy model Markowa na sytuacje, w których obserwacje same w sobie są probabilistycznymi funkcjami aktualnych stanów ukrytych. Zatem wynikowy model, zwany ukrytym modelem Markowa, jest podwójnie osadzonym procesem stochastycznym. HMM można opisać jako obserwowalny proces stochastyczny wspierany przez dalszy nieobserwowalny lub ukryty proces stochastyczny. Przykładowym zastosowaniem HMM może być obsługa komputerowego rozpoznawania słów poprzez interpretację hałaśliwych sygnałów akustycznych. Same wzorce telefoniczne, czyli fonemy, których mówca zamierza użyć w ramach tworzenia określonych słów w języku, składają się na ukryty poziom modelu Markowa. Obserwacje, czyli słyszalne szumne sygnały akustyczne, są stochastyczną funkcją tych zamierzonych fonemów. Fonemy, które zamierzał mówca, można zobaczyć tylko probabilistycznie przez najwyższy poziom (obserwowalny) strumień zaszumionych sygnałów akustycznych. Definiujemy HMM:

DEFINICJA

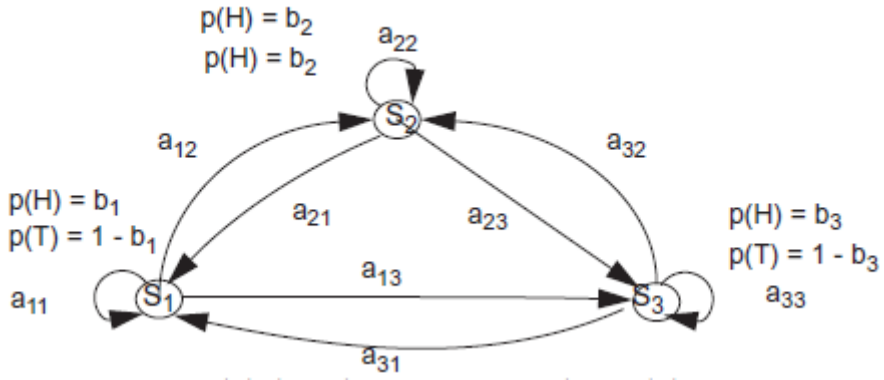
UKRYTY MODEL MARKOWA

Model graficzny nazywany jest ukrytym modelem Markowa (HMM), jeśli jest to model Markowa, którego stany nie są bezpośrednio obserwowalne, ale są ukryte przez dalszy system stochastyczny interpretujący ich wyniki. Bardziej formalnie, biorąc pod uwagę zbiór stanów $S = s_1, s_2, ..., s_n$ oraz zbiór prawdopodobieństw przejścia stanów $A = a_{11}, a_{12}, ..., a_{1n}, a_{21}, a_{22}, ..., .. ., a_{nn}$, istnieje zbiór prawdopodobieństw obserwacji $O = p_i(o_t)$, z których każde wyraża prawdopodobieństwo wygenerowania obserwacji ot (w czasie t) przez stan s_t.

Podajemy teraz dwa przykłady HMM, zaadaptowane z Rabinera. Najpierw rozważ sytuację, w której rzuca się monetą. Załóżmy, że w pokoju jest osoba, która rzuca monetami pojedynczo. Nie mamy pojęcia, co się dzieje w pokoju, w rzeczywistości może być wiele monet, losowo wybranych do przewrócenia, a każda z nich może mieć własne tendencyjne wyniki, tj. mogą nie być uczciwymi monetami. Wszystko, co obserwujemy na zewnątrz pomieszczenia, to seria wyników z odwracania, np. Obserwowalne O = H, H, T, H, T, H, ... Naszym zadaniem jest zaprojektowanie modelu monet przerzuconych wewnątrz pomieszczenia, który tworzy otrzymany obserwowalny ciąg. W tej sytuacji spróbujemy teraz zamodelować wynikowy zestaw obserwacji pochodzących z pomieszczenia. Zaczynamy od prostego modelu; załóżmy, że rzuca się tylko jedną monetą. W tym przypadku musimy tylko określić odchylenie monety, ponieważ jest ona odwracana w czasie, tworząc zestaw obserwacji głowy / ogona. To podejście daje w wyniku bezpośrednio obserwowalny (zerowego rzędu) model Markowa, który jest po prostu zbiorem prób Bernoulliego. Ten model może w rzeczywistości być zbyt prosty, aby wiarygodnie uchwycić to, co dzieje się w problemie z rzucaniem monetą. Następnie rozważymy model z dwiema monetami, z dwoma stanami ukrytymi s_1 i s_2, jak widać na rysunku 1.



Probabilistyczna macierz przejścia, A, kontroluje, w jakim stanie jest system, tj. Która moneta jest rzucana w dowolnym momencie. Każda moneta ma inne odchylenie H / T, b_1 i b_2. Załóżmy, że zaobserwowaliśmy ciąg rzutów monetą: O = H, T, T, H, T, H, H, H, T, T, H. Ten zestaw obserwacji można wyjaśnić następującą ścieżką przez diagram przejść stanów z rysunku 1: $s_2, s_1, s_1, s_2, s_2, s_2, s_1, s_2, s_2, s_1, s_2$. Trzeci model przedstawiono na Rysunku 2, gdzie trzy stany (trzy monety) są używane do przechwytywania wyniku rzutu monetą.



Ponownie, każda moneta / stan, si, ma swoje własne odchylenie, b_i, a stan systemu (która moneta jest odwrócona) jest funkcją pewnego zdarzenia probabilistycznego, takiego jak rzut kostką i macierz przejścia, A. maszyna trójstanowa może odpowiadać za wyjście H / T z sekwencją stanów: s_3, s_1, s_2, s_3, s_3, s_1, s_1, s_2, s_3, s_1, s_3. Trudną kwestią w przypadku rzutu monetą jest wybór optymalnego modelu. Najprostszy model ma jeden parametr, odchylenie jednej monety. Model z dwoma stanami ma cztery parametry, przejścia stanów i odchylenia każdej monety. Model trójstanowy z rysunku 2 ma dziewięć parametrów. Im więcej stopni swobody, tym większy model ma większe możliwości, aby uchwycić (prawdopodobnie) bardziej złożoną sytuację. Jednak trudności związane z określeniem parametrów dla większego modelu mogą zepsuć to ćwiczenie. Na przykład rzeczywiste obserwowalne mogą powstać w prostszej sytuacji, w którym to przypadku większy model byłby nieprawidłowy, niedokreślony i wymagałby znacznie więcej danych do ustalenia dowolnego stopnia pewności. Podobne problemy, takie jak nadmierne dopasowanie, również prześladują mniejsze modele. Wreszcie, w każdym modelu istnieje niejawne założenie stacjonarności, to znaczy, że prawdopodobieństwa przejścia i odchylenia systemu nie zmieniają się w czasie. Jako drugi przykład rozważmy problem N urn, gdzie każda urna zawiera zbiór M różnokolorowych kulek. Fizyczny proces uzyskiwania obserwacji polega, zgodnie z pewnym przypadkowym procesem, na wybraniu jednej z N urn. Po wybraniu urny kula jest usuwana, a jej kolor jest rejestrowany w strumieniu wyjściowym. Kula jest następnie zastępowana, a losowy proces powiązany z bieżącą urną wybiera następną (która może być taka sama), aby kontynuować proces. Ten proces generuje sekwencję obserwacji składającą się z kolorów kulek. Oczywiste jest, że najprostszym HMM odpowiadającym temu procesowi jest model, w którym każdy stan odpowiada określonej urnie, wartości macierzy przejścia dla tego stanu tworzą następny wybór stanu, a na koniec model, w którym prawdopodobieństwo koloru piłek jest określane przez liczbę i kolor piłek w każdym stanie (urna). Jednak gdyby modelarz nie miał wcześniejszych informacji, określenie wszystkich parametrów wymaganych do wybrania optymalnego HMM byłoby rzeczywiście bardzo trudne. W następnej sekcji rozważymy kilka odmian HMM.

13.1.2 Ważne warianty ukrytych modeli Markowa

W poprzedniej sekcji zauważyliśmy, że HMM są niezwykle potężnymi, ale prostymi modelami, które mogą reprezentować niepewne dziedziny. Dotychczasowa prostota obliczeniowa HMM wynika z markowej zasady systemu pierwszego rzędu: stan obecny zależy tylko od poprzedniego stanu. W tej sekcji zbadamy, jak możemy zmodyfikować podstawowe zasady HMM, aby uzyskać jeszcze bogatsze możliwości reprezentacji. Zwykłe HMM z sekcji 13.1.1 mają ograniczenia w określonych domenach, a następnie pokażemy warianty HMM, które mogą rozwiązać wiele z tych ograniczeń. Szeroko stosowane rozszerzenie HMM nazywa się autoregresywnym HMM, gdzie poprzednia obserwacja może również wpływać na wartość bieżącej obserwacji. To rozszerzenie służy do przechwytywania większej ilości informacji z przeszłości i uwzględniania ich w interpretacji stanu obecnego. W celu modelowania złożonych procesów składających się ze skojarzeń zestawów prostszych podprocesów często stosuje

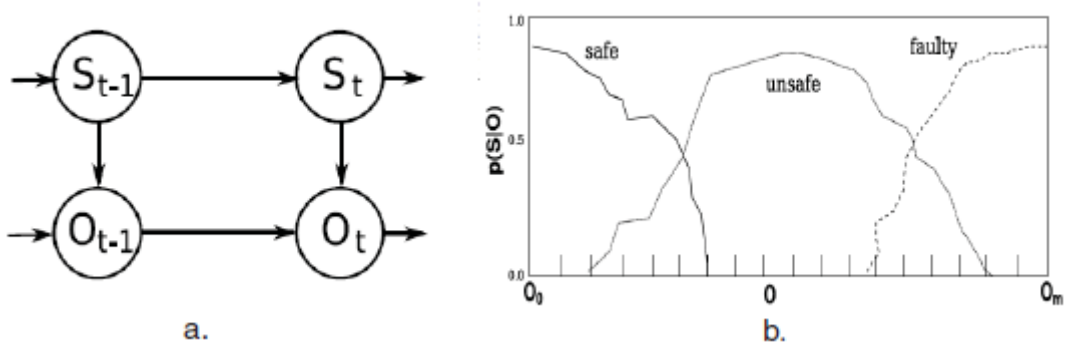
się czynnikowe HMM. Jedno rozszerzenie HMM, hierarchiczne HMM lub (HHMS), zakłada, że każdy stan systemu sam jest HMM. Następnie rozważymy bardziej szczegółowo te i inne rozszerzenia HMM.

HMM z autoregresją (AR-HMM)

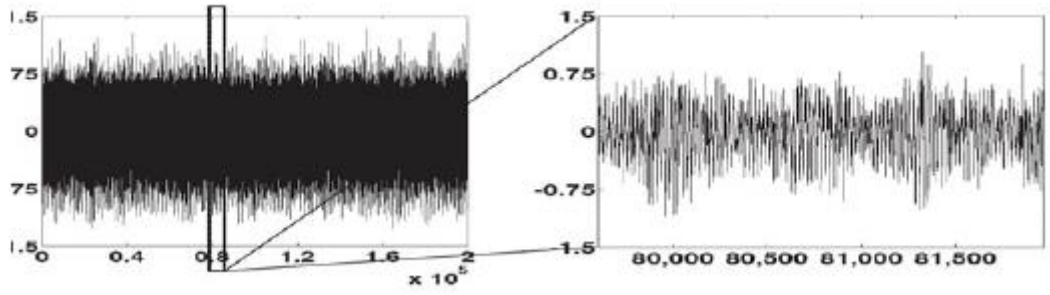
W ukrytych modelach Markowa właściwość pierwszego rzędu zakłada, że stan obecny jest zależny tylko od bezpośredniego poprzedniego stanu i że obserwowane zmienne nie są korelowane. Może to być ograniczenie, w którym obecny stan nie pozwala odpowiednio uchwycić informacji dostępnych w poprzednich stanach. Na przykład w rozumowaniu diagnostycznym może to prowadzić do gorszych uogólnień, zwłaszcza podczas modelowania danych szeregów czasowych. Faktem jest, że w złożonych środowiskach obserwowalne wartości w jednym okresie często korelują z kolejnymi obserwacjami. Należy to uwzględnić w strukturze HMM. Przezwyciężamy ten aspekt ograniczonego uogólnienia, pozwalając poprzedniej obserwacji wpływać na interpretację bieżącej obserwacji wraz ze zwykłymi wpływami HMM poprzedniego stanu. Ten model HMM nazywa się Model Markowa z ukrytą autoregresją, którego model emisji jest określony przez niezerowy model prawdopodobieństwa:

$p(O_t \mid S_t, O_{t-1})$.

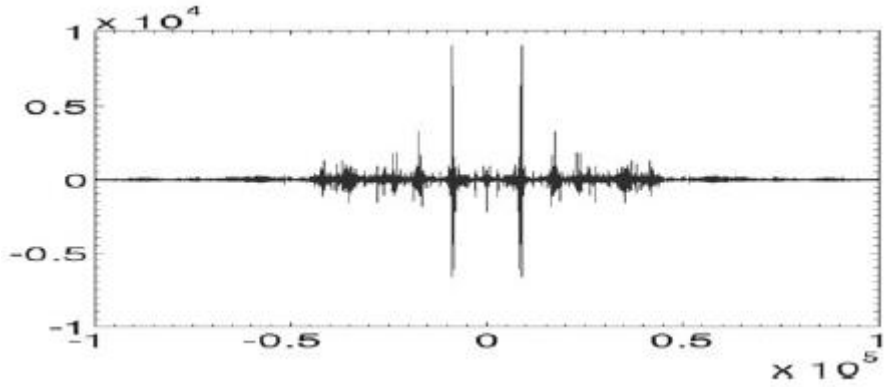
gdzie O to stan emisji lub obserwacji, a S to warstwa ukryta, jak pokazano na rysunku 3a.



Różnica między tradycyjnym HMM a AR-HMM polega na istnieniu powiązania (wpływu) między O_{t-1} i Ot. Motywacja dla tego wpływu jest prosta: stan obserwacji w jednym okresie będzie wskaźnikiem tego, jaki prawdopodobnie nastąpi następna obserwacja. Innymi słowy, obserwowane wartości wyjściowe analizy szeregów czasowych zmienią się zgodnie z pewnym podstawowym rozkładem. AR-HMM często prowadzi do modeli o wyższym prawdopodobieństwie zbieżności niż zwykłe HMM, szczególnie podczas modelowania niezwykle złożonych i zaszumionych danych szeregów czasowych. Przedstawiamy AR-HMM na przykładzie zaczerpniętym z pracy Chakrabarti i inych. Zadanie polega na monitorowaniu stanu pracy układu wirnika helikoptera. Rysunek 4 przedstawia próbkę danych obserwowanych z uruchomionego systemu.



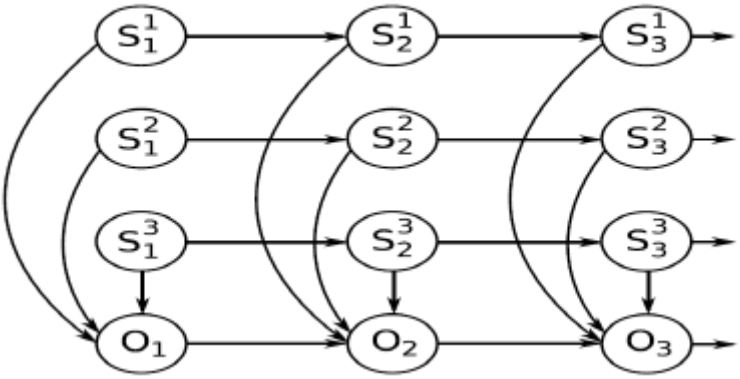
Dane pochodzą z wielu czujników, w tym informacje z pomiarów ciepła i wibracji wielu elementów. Rysunek 5 przedstawia wyniki przetwarzania danych z rysunku 4 przy użyciu szybkiej transformaty Fouriera (FFT) w celu uzyskania równoważnych danych w dziedzinie częstotliwości.



Te punkty danych są następnie wykorzystywane do szkolenia AR-HMM. Trzy stany ukrytego węzła, S_t, na rysunku 3, są bezpieczne, niebezpieczne i wadliwe. Celem tego projektu było zbadanie technik wykrywania i diagnostyki usterek w układach mechanicznych. Wykorzystując zestaw danych szeregów czasowych z czujników monitorujących procesy mechaniczne, zadaniem było zbudowanie ilościowego modelu całego procesu i przeszkolenie go na zestawach danych (usiane błędy, takie jak pęknięty wał) do późniejszego wykorzystania w diagnostyce w czasie rzeczywistym i przewidywanie przyszłych usterek. Po przeszkoleniu AR-HMM został zwolniony do pracy w czasie rzeczywistym. Chociaż raportowana dokładność wynosiła nieco ponad 75%, ważne jest, aby zobaczyć, jak można zaprojektować takie narzędzia prognostyczne.

Silnia HMMs

W naszych poprzednich dyskusjach na temat HMM i ich wariantów opisaliśmy systemy, których przestrzeń stanów była częściowo ukryta. Wskazówki dotyczące przestrzeni stanów mogliśmy uzyskać tylko poprzez serię obserwacji. Próbowaliśmy wywnioskować sekwencję ukrytych stanów ukrytych, wykorzystując informacje z obserwowanej sekwencji danych emitowanych przez system. Ale co się stanie, jeśli podstawowy proces jest bardzo złożony i nie można go skutecznie przedstawić jako prostego zestawu stanów? Co się stanie, jeśli podstawowy proces jest w rzeczywistości połączeniem kilku podprocesów? Nasz regularny HMM nie jest w stanie w wystarczającym stopniu uchwycić informacji o procesach w takich sytuacjach. Potrzebujemy bogatszego modelu, aby przedstawić bardziej złożone sytuacje. Możliwym rozwiązaniem są czynnikowe HMM. Rysunek 6 przedstawia silnię autoregresyjną



HMM. Widzimy, że podstawowy proces można podzielić na n podprocesów, gdzie każdy z tych n procesów wpływa na bieżącą obserwację, Ot. Ponadto każdy podproces jest zgodny z właściwością Markowa pierwszego rzędu, to znaczy jego bieżący stan zależy tylko od poprzedniego stanu (stanów). Ten związek z obserwowalną zmienną losową można zdefiniować za pomocą CPD:

$p(O_t|O_{t-1},S^1_t, S^2_t, ..., S^i_t)$

Hierarchiczne HMM (HHMM)

Hierarchiczne moduły HMM są używane do modelowania niezwykle złożonych systemów. W hierarchicznym ukrytym modelu Markowa (lub HHMM) każdy stan jest uważany za samodzielny model probabilistyczny. Dokładniej, każdy stan HHMM może sam być HHMM. Oznacza to, że stany HHMM emitują sekwencje obserwacji, a nie tylko pojedyncze obserwacje, jak ma to miejsce w przypadku standardowych HMM i wariantów widzianych do tej pory. Kiedy stan w HHMM zostanie osiągnięty, aktywuje swój własny model probabilistyczny, tj. Aktywuje jeden ze stanów bazowego HHMM, który z kolei może aktywować jego bazowy HHMM itd., Podstawowym przypadkiem jest tradycyjny HMM. Proces jest powtarzany do momentu aktywacji stanu zwanego stanem produkcyjnym. Tylko stany produkcyjne emitują symbole obserwacji w znaczeniu zwykłego ukrytego modelu Markowa. Gdy stan produkcji wyemituje symbol, sterowanie powraca do stanu, który aktywował stan produkcji. Stany, które nie emitują bezpośrednio symboli obserwacji, nazywane są stanami wewnętrznymi. Aktywacja stanu w HHMM w stanie wewnętrznym nazywana jest przejściem pionowym. Po zakończeniu przejścia pionowego następuje przejście poziome do stanu na tym samym poziomie. Kiedy przejście poziome prowadzi do stanu końcowego, sterowanie jest przywracane do stanu w HHMM wyżej w hierarchii, który spowodował ostatnie przejście pionowe. Należy zauważyć, że przejście wertykalne może skutkować bardziej pionowymi przejściami przed osiągnięciem sekwencji stanów produkcyjnych i ostatecznym powrotem na najwyższy poziom. W ten sposób odwiedzane stany produkcji dają początek sekwencji symboli obserwacji, które są wytwarzane przez państwo na najwyższym szczeblu.

HMM N-Gram

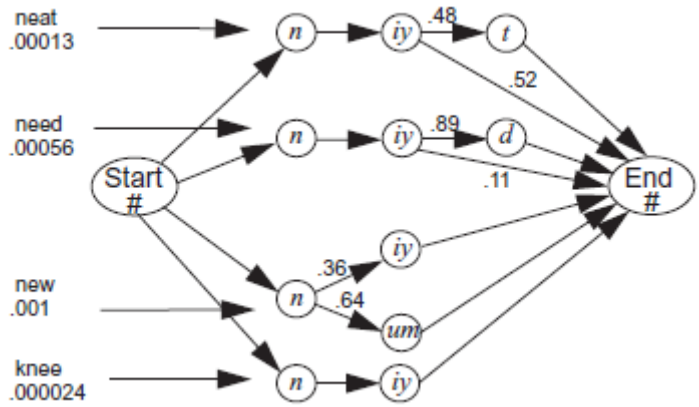
Jak wspomniano wcześniej, w tradycyjnych modelach Markowa pierwszego rzędu stan obecny, biorąc pod uwagę stan bezpośrednio poprzedni, jest niezależny od wszystkich wcześniejszych stanów. Chociaż ta reguła zmniejsza złożoność obliczeniową przy korzystaniu z procesów Markowa, może się zdarzyć, że sam poprzedni stan nie może całkowicie uchwycić ważnych informacji dostępnych w domenie problemu. Wyraźnie rozwiązujemy tę sytuację za pomocą n-gramowego ukrytego modelu Markowa (lub N-gramowego HMM). Modele te, zwłaszcza w ich postaci bi-gramowej i trigramowej, są szczególnie ważne w stochastycznych podejściach do rozumienia języka naturalnego. W bi-gramowych modelach Markowa interpretacja obserwowalnego stanu prądu zależy tylko od interpretacji stanu

poprzedniego, jako $p(O_t|O_{t-1})$; jest to odpowiednik tradycyjnego łańcucha Markowa pierwszego rzędu. Chociaż założenie Markowa pierwszego rzędu jest oczywiste, warto zastanowić się, co oznacza przetwarzanie bi-gramowe, na przykład w rozumieniu języka naturalnego. Przyjmuje hipotezę, że w przypadku rozpoznawania wyrazów lub fonemów, biorąc pod uwagę sekwencję poprzednich słów, prawdopodobieństwo wystąpienia określonego słowa jest najlepiej oszacowane, gdy używamy tylko interpretacji bezpośrednio poprzedniego słowa. Zakłada się zatem, że bigram daje lepsze wyniki niż wiedza o braku informacji o poprzednim słowie lub znajomość większej sekwencji poprzednich słów. Na przykład, biorąc pod uwagę zbiór danych językowych, p (lamb | little) jest lepszym predyktorem bieżącego obserwowanego słowa będącego jagnięciem, niż albo p (lamb | a little), p (lamb | had a little) lub p (jagnięcina | Mary miała trochę). Podobnie modele trigramowe to modele Markowa, w których interpretacja obserwowalnego stanu prądu zależy od dwóch poprzednich stanów. Model trigramowy jest reprezentowany przez $P (O_t|O_{t-1}, O_{t-2})$. Kontynuując przykład z języka naturalnego, użycie modeli tri-gramowych oznacza, że p (lamb | a little) jest lepszym predyktorem obecnego słowa będącego jagnięciem niż p (lamb | little) lub większej liczby poprzednich słów, w tym p (lamb | Mary miała trochę). Możemy rozszerzyć n-gramowy model Markowa do dowolnej długości n, która naszym zdaniem obejmuje niezmienności obecne w danych, które próbujemy modelować. N-gramowy model Markowa to łańcuch Markowa, w którym prawdopodobieństwo bieżącego stanu zależy dokładnie od n-1 poprzednich stanów. Jak zobaczymy bardziej szczegółowo w Części 15, modele n-gramowe są szczególnie przydatne do przechwytywania sekwencji głosek, sylab lub słów podczas rozumienia mowy lub innych zadań związanych z rozpoznawaniem języka. Możemy ustawić wartość n w zależności od pożądanej złożoności domeny, którą chcemy reprezentować. Jeśli n jest zbyt małe, możemy mieć słabszą siłę reprezentacji i przegapić zdolność uchwycenia mocy predykcyjnej dłuższych strun. Z drugiej strony, jeśli wartość n jest zbyt duża, wówczas koszt obliczeniowy walidacji modelu może być bardzo wysoki lub nawet niemożliwy. Głównym tego powodem jest fakt, że wcześniejsze oczekiwania dotyczące połączeń telefonicznych lub słów są pobierane z dużych baz danych (często połączonych baz danych) zgromadzonych zjawisk językowych zwanych korpusem. Po prostu możemy nie mieć wystarczających danych, aby zweryfikować n-gram z bardzo dużym n. Na przykład, możemy nie mieć żadnych informacji na temat p (baranek | Mary miała trochę), podczas gdy mamy dużo informacji o p (jagnię | mało), p (jagnię | trochę) lub p (jagnię | trochę). Zwykle w zadaniach rozpoznawania mowy lub tekstu używamy n-gramów o wartości n nie większej niż 3. Istnieje wiele innych wariantów HMM, których tutaj nie opisaliśmy, w tym HMM z pamięcią mieszaną, które używają kilku niższego rzędu Modele Markowa w celu przezwyciężenia problemów złożoności. Dalsze rozszerzenie autoregresywnych HMM, zwanych zakopanymi modelami Markowa, pozwala na nieliniowe zależności między obserwowalnymi węzłami. Moduły HMM wejściowe i wyjściowe służą do mapowania sekwencji wejść na sekwencję wyjść. W sytuacjach, w których interakcje między podprocesami złożonego procesu są bardziej skomplikowane niż tylko kompozycja, często stosuje się sprzężony HMM. Ukryty model semi-Markowa uogólnia HMM, aby uniknąć efektu rozpadu geometrycznego HMM poprzez uzależnienie prawdopodobieństwa przejść między stanami ukrytymi od ilości czasu, jaki upłynął od ostatniej zmiany stanu. Odsyłamy do sekcji 13.4 w celu uzyskania odniesień. W kolejnej części do problemu tłumaczenia ustnego zastosujemy technologię bi-gramowego HMM. Angielskie kombinacje fonemów i wykorzystanie programowania dynamicznego z algorytmem Viterbiego do implementacji wyszukiwania HMM

13.1.3 Używanie HMM i Viterbi do dekodowania ciągów fonemów

Nasza ostatnia sekcja o ukrytych modelach Markowa pokazuje ważne zastosowanie technologii HMM: identyfikowanie wzorców w mówionym języku naturalnym. Zakładamy obecność dużego korpusu językowego, który wspiera wcześniejsze oczekiwania dotyczące kombinacji telefonów. Na koniec używamy algorytmów programowania dynamicznego, do implementacji wnioskowania HMM. Kiedy

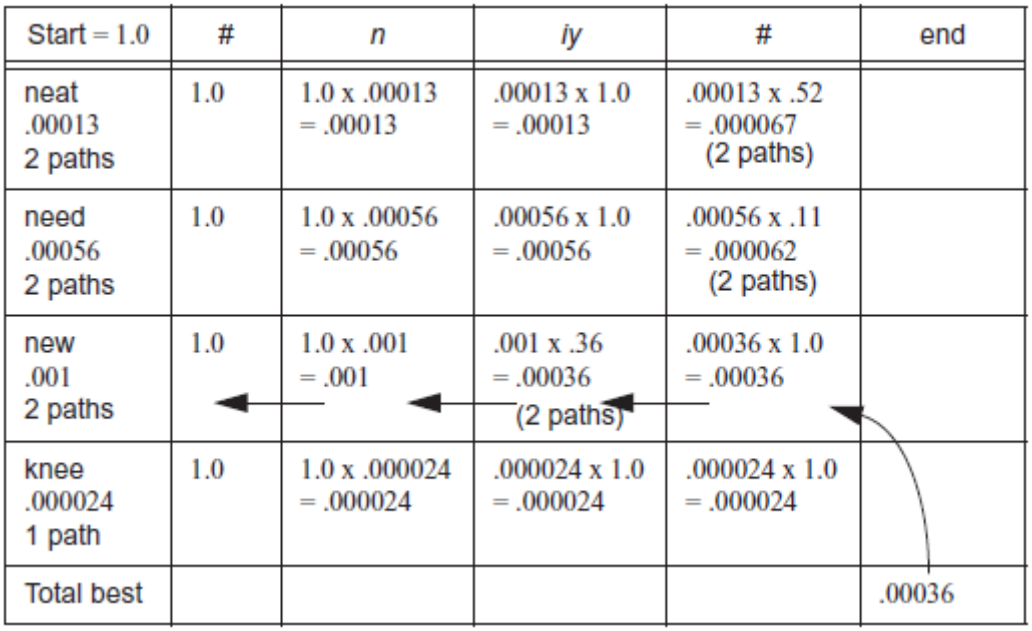
programowanie dynamiczne jest używane do znalezienia maksymalnego prawdopodobieństwa wystąpienia ciągu a posteriori, często nazywa się to algorytmem Viterbiego. Poniższy przykład obliczeniowej analizy wzorców mowy ludzkiej i wykorzystanie algorytmu Viterbiego do interpretacji ciągów fonemów jest zaadaptowane z Jurafsky i Martin. Dane wejściowe do tej probabilistycznej maszyny to ciąg telefonów lub podstawowe dźwięki mowy. Wynikają one z rozkładu sygnału akustycznego wytwarzanego podczas używania języka mówionego. W automatycznym rozumieniu mowy niezwykłe jest to, że sygnały akustyczne byłyby jednoznacznie rejestrowane jako ciąg fonemów. Sygnały mowy są raczej interpretowane jako określone telefony probabilistycznie. Zakładamy jednoznaczną interpretację sygnałów, aby uprościć naszą prezentację algorytmu Viterbiego przetwarzania HMM. Rysunek 7 przedstawia wycinek bazy danych słów, powiązanych pod względem bliskości zbiorów fonemów tworzących ich składowe akustyczne.



Chociaż ten zestaw słów, schludne, nowe, potrzeba i kolano, to tylko niewielka część pełnego słownictwa angielskiego, można sobie wyobrazić, że bardzo duża liczba tych powiązanych klastrów mogłaby wspierać system rozumienia mowy. Rysunek 13.7 jest przykładem probabilistycznej maszyny skończonej, przedstawionej po raz pierwszy w sekcji 5.3. Zakładamy, że nasz korpus językowy jest zbiorem podobnych probabilistycznych maszyn skończonych, które mogą dawać miary prawdopodobieństwa różnych możliwych kombinacji telefonów, i jak ostatecznie te telefony można postrzegać jako części słów.

Celem analizy jest określenie, które słowo angielskie z naszej bazy danych najlepiej reprezentuje wejście sygnału akustycznego. Wymaga to użycia technologii HMM, ponieważ przedstawienie możliwych słów samo w sobie jest stochastyczne. W niedeterministycznej maszynie skończonej z rysunku 7, ciąg znaków daje nam zbiór obserwacje do zinterpretowania. Załóżmy, że ciąg obserwacji składa się z telefonów #, n, iy, #; gdzie # oznacza przerwę między dźwiękami. Używamy algorytmu Viterbiego, aby zobaczyć, która ścieżka przez probabilistyczną maszynę skończoną najlepiej oddaje te obserwacje. W trybie do przodu Viterbi iteracyjnie znajduje następny najlepszy stan i jego wartość, a następnie ustawia na niego wskaźnik. W trybie wstecznym śledzimy te wskaźniki, aby uzyskać najlepszą ścieżkę. Zatem wyjście Viterbiego jest jedną z optymalnych ścieżek stanów na wykresie związanym z prawdopodobieństwem tej ścieżki. Używając Viterbi, każdy stan wyszukiwania jest powiązany z wartością. Wartość bycia w stanie si w czasie t to viterbi [s_i, t]. Wartość związana z następnym stanem s_j w automacie stanu w czasie t + 1 to viterbi [s_j, t + 1]. Wartość następnego stanu jest obliczana jako wynik wyniku obecnego stanu, viterbi [s_i, t], razy prawdopodobieństwo przejścia ze stanu obecnego do następnego, ścieżka [s_i, s_j], razy prawdopodobieństwo obserwacji s_j dane si, p (s_j | s_i). Prawdopodobieństwo przejścia, ścieżka [si, sj], jest pobierane z niedeterministycznej maszyny skończonej, a p (s_j | s_i) jest pobierane ze znanych prawdopodobieństw obserwacji par telefonicznych

występujących w języku angielskim. Następnie przedstawiamy pseudokod algorytmu Viterbiego. Zwróć uwagę na podobieństwo do programowania dynamicznego. Tablica do przechowywania i koordynowania wartości musi obsługiwać iterację w R wierszach - równą liczbie telefonów w probabilistycznym automacie skończonym (PFSM) plus dwa, aby obsłużyć każdy stan oraz stany początkowe i końcowe. Musi również iterować po kolumnach C - równej liczbie obserwacji plus dwa, aby obsłużyć użycie pustego numeru telefonu #. Kolumny wskazują również sekwencję czasową obserwacji, a każdy stan musi być powiązany z odpowiednim obserwowanym telefonem, jak pokazano na rysunku 8.



function Viterbi(Observations of length T, Probabilistic FSM)

begin

number := number of states in FSM

create probability matrix viterbi[R = N + 2, C = T + 2];

viterbi[0, 0] := 1.0;

for each time step (observation) t from 0 to T do

for each state si from i = 0 to number do

for each transition from si to sj in the Probabilistic FSM do

begin

new-count := viterbi[si, t] x path[si, sj] x p(sj | si);

if ((viterbi[sj, t + 1] = 0) or (new-count > viterbi[sj, t + 1]))

then

begin

viterbi[si, t + 1] := new-count

append back-pointer [sj , t + 1] to back-pointer list

end;

end;

return viterbi[R, C];

return back-pointer list

end.

Rysunek 8, zaadaptowany na podstawie Jurafsky i Martina, przedstawia ślad algorytmu Viterbiego przetwarzającego składnik probabilistycznej maszyny skończonej z rysunku 7 oraz sekwencję telefoniczną #, n, iy, # (obserwacje o długości T = 4) . Linki śledzenia wstecznego wskazują optymalną ścieżkę przez probabilistyczną skończoną maszynę stanów. Ta ścieżka wskazuje, że najbardziej prawdopodobną interpretacją ciągu obserwowanych telefonów jest słowo nowy. W następnej sekcji przedstawiamy inny model graficzny, uogólnienie modeli Markowa, zwany dynamiczną siecią bayesowską (lub DBN).

13.2 Dynamiczne sieci bayesowskie i uczenie się

Często napotykamy problemy w modelowaniu, w których podstawowy proces, który ma być scharakteryzowany, najlepiej opisać jako sekwencję lub progresję stanów. Dane zebrane z procesów sekwencyjnych są zwykle indeksowane za pomocą parametru, takiego jak znacznik czasu pozyskania danych lub pozycja w ciągu tekstowym. Takie zależności sekwencyjne mogą być zarówno deterministyczne, jak i stochastyczne. Dynamiczne procesy, które są z natury stochastycznymi modelami popytu, które są w stanie modelować ich niedeterminizm. Chociaż większość deterministycznych sekwencji można dobrze scharakteryzować za pomocą maszyn skończonych, niektóre ważne sekwencje mogą być dowolnie długie. Dlatego modelowanie każdego wystąpienia tych sekwencji jako zbiorów poszczególnych stanów staje się problematyczne, ponieważ liczba przejść między stanami rośnie wykładniczo w stosunku do liczby stanów w danym momencie. Dlatego też probabilistyczne modelowanie takich sekwencji pozwala nam na utrzymanie modeli wykonalnych, jednocześnie rejestrując informacje o sekwencjach istotne dla danego problemu. W sekcji 13.2 opisujemy dwa różne podejścia do uczenia się, uczenia się struktury i uczenia się parametrów w kontekście modeli graficznych. Pokrótce opisujemy techniki uczenia się struktur, w tym wykorzystanie próbkowania metodą Monte Carlo (lub MCMC) w łańcuchu Markowa.

Opisujemy również popularny meta-algorytm zwany Maksymalizacją oczekiwań (lub EM) do uczenia parametrów. Pokazujemy, jak rekurencyjny algorytm programowania dynamicznego zwany algorytmem Bauma-Welcha dostosowuje meta-algorytm EM do użytku zarówno z modułami DBM, jak i HMM. W sekcji 13.3 wyjaśniamy, w jaki sposób modele Markowa mogą być wykorzystywane do wspomagania decyzji poprzez wprowadzenie procesu decyzyjnego Markowa (lub MDP) i częściowo obserwowanego MDP (lub POMDP). Następnie bierzemy naukę ze wzmocnieniem i uzupełniamy ją o probabilistyczne modele stanu i sprzężenia zwrotnego. Wreszcie, kończymy Część 13, wykorzystując uczenie ze wzmocnieniem przy użyciu MDP do zaprojektowania systemu nawigacji robota.

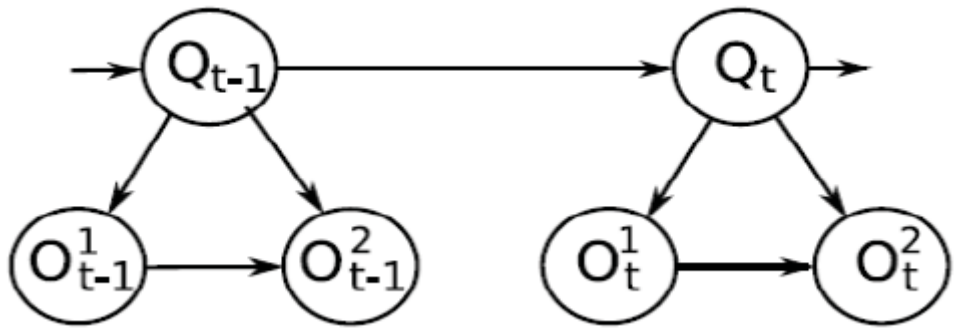
13.2.1 Dynamiczne sieci bayesowskie

Przedstawiliśmy sieć przekonań bayesowskich, bardzo popularny i z powodzeniem stosowany formalizm do modelowania probabilistycznego. Jednak w przypadku modelowania procesów dynamicznych lub sekwencyjnych BBN są dość nieelastyczne. Nie mają możliwości modelowania danych szeregów czasowych, w tym procesów przyczynowych lub sekwencyjnych korelacji danych,

które często występują w złożonych zadaniach, w tym w zrozumieniu języka naturalnego i diagnostyce w czasie rzeczywistym. Problem polega na tym, że zakłada się, że wszystkie warunkowe zależności BBN są aktywne w tym samym momencie. Krótko mówiąc, dynamiczna, czasami nazywana czasową, sieć bayesowska to sekwencja identycznych sieci bayesowskich, których węzły są połączone w (ukierunkowanym) wymiarze czasu. dynamiczne sieci bayesowskie to sekwencja ukierunkowanych modeli graficznych procesów stochastycznych. Są to uogólnienia popularnych narzędzi stochastycznych, takich jak ukryte modele Markowa (HMM) i liniowe systemy dynamiczne (LDS), które reprezentują stany ukryte (i obserwowane) jako zmienne stanu, które mogą mieć złożone współzależności w różnych okresach czasu. Struktura graficzna umożliwia określenie zależności warunkowych, a tym samym oferuje kompaktową parametryzację modelu. Poszczególne BN, które tworzą DBN, nie są dynamiczne w tym sensie, że same nie zmieniają się w czasie, przy założeniu stacjonarności, ale nadal wspierają modelowanie procesów dynamicznych. Zatem DBN działają przy założeniu, że ich parametry definiujące są niezmienne w czasie. Jednak, jak zobaczymy, ukryte węzły mogą być dodawane do reprezentowania bieżących warunków pracy, tworząc w ten sposób mieszanki modeli w celu uchwycenia okresowych niezmienności w wymiarze czasowym. W każdym przedziale czasowym (wycinku) DBN jest podobny do statycznego BBN, który ma zestaw Q_t, z N_h zmiennych losowych reprezentujących stany ukryte i zbiór zmiennych losowych O_t z N_o reprezentujących stany, które są obserwowalne. Każda zmienna losowa może być dyskretna lub ciągła. Musimy zdefiniować model reprezentujący rozkłady prawdopodobieństwa warunkowego między stanami w pojedynczym wycinku czasu i modelem przejścia, który reprezentuje relacje między BBN dla kolejnych wycinków czasu. Dlatego też, jeśli $Z_t = \{Q_t \cup O_t\}$ jest sumą obu zbiorów zmiennych, model przejścia i obserwacji można zdefiniować jako iloczyn warunkowych rozkładów prawdopodobieństwa w dwóch okresach czasu BBN bez cykli, nazwijmy to B_t. Notacja (1: N) oznacza dla wszystkich zmiennych od 1 do N:

$$p(Z_t(1:N) \mid Z_{t-1}(1:N)) = \prod_{i=1}^{N} p(Z_t(i) \mid Pa(Z_t(i)))$$

gdzie $Z_t(i)$ jest i-tym węzłem w wycinku czasu t, $Pa(Z_t(i))$ są rodzicami $Z_t(i)$ i $N = N_h + N_o$. Należy zauważyć, że tutaj, dla uproszczenia, ograniczyliśmy definicję do pierwszego -porządek procesów Markowa, w których na zmienne w czasie t wpływa tylko ich poprzednik w czasie t - 1. Możemy przedstawić bezwarunkowy rozkład stanu początkowego $p(Z_1(1: N))$, jako bezwarunkową sieć Bayesa B_1. B_1 i B_t, dla wszystkich t, definiują DBN. Na rysunku 9 pokazujemy dwa wycinki czasu prostego DBN. Jak powinno być oczywiste, dynamiczna sieć bayesowska jest uogólnieniem ukrytych modeli Markowa przedstawionych w rozdziale 13.1. Na rysunku 9 sieci bayesowskie są połączone między czasami t i t + 1.



Co najmniej jeden węzeł z dwóch sieci musi być połączony w czasie.

13.2.2 Nauka sieci bayesowskich

Uczenie się ma kluczowe znaczenie dla inteligentnego działania. Inteligentni agenci często rejestrują zdarzenia zaobserwowane w przeszłości w oczekiwaniu, że te informacje mogą w przyszłości posłużyć do jakiejś użyteczności. W ciągu swojego życia inteligentny agent napotyka wiele zdarzeń, być może znacznie więcej, niż ma zasoby do przechowywania. Dlatego konieczne staje się kodowanie informacji o zdarzeniach za pomocą zwięzłej abstrakcji, a nie pełnej dokumentacji każdej obserwacji. Ponadto zagęszczenie informacji pomaga w szybszym wyszukiwaniu. Dlatego uczenie się można postrzegać jako połączenie kompresji danych, indeksowania i przechowywania w celu wydajnego odzyskiwania. Każda nowa obserwacja agenta może nieznacznie różnić się od poprzednich obserwacji, ale nadal może należeć do ogólnej klasy wydarzeń wcześniejszych. W takim przypadku pożądane jest nauczenie się tego zdarzenia jako wystąpienia poprzedniej klasy, ignorując mniej ważne szczegóły konkretnego zdarzenia. Nowe zdarzenie może również bardzo różnić się od wszystkich wcześniejszych, a wtedy agent będzie musiał nauczyć się go jako nowej klasy lub określonej anomalii bezklasowej. Agent musi również utworzyć wystarczającą liczbę klas, które w pełni charakteryzują również przestrzeń parametrów jak być w stanie skutecznie obsługiwać nowe anomalne zdarzenia. Zatem dobre uczenie się to równowaga zarówno ogólności, jak i specyfiki. W kontekście probabilistycznych modeli graficznych napotykamy dwa ogólne problemy związane z uczeniem się, które pasują do naszej poprzedniej dyskusji: uczenie się struktury, w którym uczone są całe nowe relacje (modele graficzne), oraz uczenie parametrów, w przypadku którego komponenty istniejącego modelu są rekalibrowane.

Uczenie się i wyszukiwanie struktur

Uczenie się struktur dotyczy ogólnego problemu określania istnienia zależności statystycznych między zmiennymi. Na przykład odpowiedzi na poniższe pytania pomogłyby w określeniu struktury probabilistycznego modelu graficznego pogody. Czy ciemne chmury są związane z szansą na deszcz? Czy istnieje korelacja między położeniem Saturna a liczbą huraganów w danym dniu? Czy globalne ocieplenie jest związane z obecnością tajfuny? Czy pogoda w Arktyce jest związana ze zmieniającymi się granicami Sahary? Jak widać, w modelu probabilistycznym pozytywna odpowiedź na takie pytania uzasadniałaby warunkową zależność między danymi zmiennymi, a odpowiedź negatywna pozwoliłaby nam traktować te zmienne jako niezależne. Projektując probabilistyczny model graficzny, każda zmienna może być zależna od każdej innej zmiennej, przy jednoczesnym zachowaniu ograniczenia, że wynikowa sieć jest skierowanym grafem acyklicznym. W najgorszym przypadku otrzymujemy wtedy w pełni podłączony DAG jako model. Jest to jednak bardzo rzadkie w większości praktycznych zastosowań dowolnej wielkości, ponieważ tworzenie tabel prawdopodobieństwa warunkowego staje się trudne do wykonania. Ważnym spostrzeżeniem wspierającym projektowanie modeli graficznych jest to, że bardzo często możemy stwierdzić, że kilka zmiennych wykazuje niezależność względem siebie w kontekście modelowanego problemu. Ponieważ złożoność wnioskowania w sieciach przekonań bayesowskich zależy od liczby zależności między zmiennymi w modelu, chcielibyśmy mieć jak najmniej zależności, jednocześnie zachowując ważność modelu. Innymi słowy, chcemy usunąć jak najwięcej (niepotrzebnych) zależności z w pełni połączonego grafu. Uczenie się struktur można więc traktować jako problem poszukiwania optymalnej struktury w przestrzeni wszystkich możliwych struktur dla danego zbioru zmiennych reprezentujących jakąś dziedzinę aplikacji. Jak zauważyli Chickering i inni optymalnym rozwiązaniem tego wyszukiwania w istniejących zbiorach danych jest NP-trudne. Optymalna struktura dobrze opisuje dane, pozostając tak prostymi, jak to tylko możliwe. Jednak ta wyuczona struktura musi opisywać zarówno wcześniej zaobserwowane dane, jak i pasować do nowych, prawdopodobnie istotnych informacji. Tak więc, kiedy tworzymy (uczymy się) złożoną strukturę, która dokładnie charakteryzuje zgromadzone dane, możemy otrzymać strukturę, która jest zbyt specyficzna, w wyniku czego nie da się dobrze uogólnić na nowych danych. W literaturze

dotyczącej optymalizacji sytuacja ta jest czasami nazywana nadmiernym dopasowaniem. Chociaż przeszukiwanie przestrzeni wykładniczej możliwych ukierunkowanych struktur acyklicznych grafów metodami brutalnej siły jest raczej nierealistyczne dla praktycznych zastosowań z wieloma zmiennymi, istnieją pewne zasady, które pomagają rozwiązać ten problem. Pierwsza praktyczna zasada to brzytwa Ockhama, czyli zasada oszczędności. Spośród dwóch konkurencyjnych struktur, które działają porównywalnie, należy preferować prostszą konstrukcję nad bardziej złożoną. Brzytwa Ockhama podpowiadałaby, że w niektórych przypadkach dobrym pomysłem może być poszukiwanie możliwych struktur przechodzących od prostych do złożonych, kończąc poszukiwania, gdy znajdziemy strukturę, która jest wystarczająco prosta, ale udaje się być odpowiednim modelem do zadania na dłoni. Użycie tej heurystyki prostoty narzuca Bayesian przed przeszukiwaniem struktury. Jednak często trudno jest zdefiniować a priori miarę złożoności konstrukcji. Wybór rozsądnej miary złożoności, która narzuca całkowity porządek na możliwe konstrukcje, nie jest oczywisty. Mogą istnieć klasy struktur, które są równie złożone, w takim przypadku moglibyśmy narzucić częściowy porządek w przestrzeni struktury. Jednak nawet wśród struktur tej samej klasy wyczerpujące przeszukiwanie wszystkich możliwości jest często niepraktyczne. Podobnie jak w przypadku wielu trudnych problemów związanych ze sztuczną inteligencją, w ogólnym przypadku indukowanie struktur jest po prostu niemożliwe w rozsądnie złożonych sytuacjach. W rezultacie stosujemy heurystykę w próbaie ułatwienia przeszukiwania struktury. Mając działający, choć niezadowalający model sytuacji, jedną z metod heurystycznych jest przeprowadzenie chciwego wyszukiwania lokalnego w pobliżu uszkodzonych komponentów modelu. Biorąc pod uwagę, że celem jest znalezienie tylko lokalnej optymalizacji w strukturze graficznej, może to stanowić wystarczająco dobre rozwiązanie problemu modelowania. Inną heurystyką byłoby poleganie na ekspertach zajmujących się problematyką w celu uzyskania porady, która pomoże przemyśleć strukturę modelu. Znowu ekspert może skupić się na punktach załamania obecnego modelu. Bardziej ogólne podejście do problemu indukcji modeli polega na próbkowaniu przestrzeni możliwych modeli. Przedstawiamy następną popularną technikę pobierania próbek tego typu, łańcuch Markowa Monte Carlo.

Łańcuch Markowa Monte Carlo (MCMC)

Łańcuch Markowa Monte Carlo (lub MCMC) to klasa algorytmów do próbkowania z rozkładów prawdopodobieństwa. MCMC opiera się na konstrukcji łańcucha Markowa, który ma pożądany rozkład jako rozkład równowagi. Stan łańcucha po dużej liczbie kroków, czasami nazywane wypalaniem w czasie, są następnie używane jako próbka dla pożądanych rozkładów możliwej struktury DAG dla domeny aplikacji. Oczywiście jakość próbki łańcucha Markowa poprawia się wraz z liczbą pobranych próbek. W kontekście przeszukiwania struktur zakładamy, że możliwe struktury dla danego zbioru zmiennych tworzą przestrzeń stanów łańcucha Markowa. Macierz przejść dla tych stanów jest wypełniona jakąś techniką opartą na wadze. Wagi mogą zaczynać się jako jednolite, gdzie losowy spacer Monte Carlo między tymi strukturami ważyłby równie prawdopodobne wszystkie modyfikacje konstrukcyjne. Z biegiem czasu, oczywiście, większe wagi są używane do przejść między blisko spokrewnionymi strukturami. Jeśli struktura A różni się od struktury B tylko jednym połączeniem, to spacer Monte Carlo w tej przestrzeni jest poszukiwaniem w heurystycznie zorganizowanych dzielnicach tych struktur. Każda struktura, gdy próbkowana, jest powiązana z wynikiem, zwykle maksymalnym wynikiem opartym na a posteriori (lub MAP), p (struktura | dane). Technika próbkowania Monte Carlo przemierza przestrzeń struktury, wybierając z większym prawdopodobieństwem przejścia poprawiające prawdopodobieństwo danych, a nie przejścia, które pogarszają wynik. Po wielu iteracjach podejście MCMC zbiega się w kierunku lepszych struktur, poprawiając późniejszy rozkład p (struktura | dane) w porównaniu z punktem początkowym. Znowu pojawia się problem odkrycia lokalnych maksimów w próbkowanej przestrzeni; problem ten można rozwiązać za pomocą technik, takich jak losowy restart lub symulowane wyżarzanie.

Uczenie się parametrów i maksymalizacja oczekiwań (EM)

Uczenie parametryczne można skontrastować z uczeniem się struktur w przestrzeni modeli graficznych. Biorąc pod uwagę istniejący model dziedziny problemowej, uczenie parametrów próbuje wywnioskować optymalny wspólny rozkład dla zbioru zmiennych przy danym zbiorze obserwacji. Uczenie się parametrów jest często używane jako podprogram w ramach ogólnego wyszukiwania struktury. Gdy istnieje kompletny zestaw danych, uczenie się parametrów dla dystrybucji ogranicza się do zliczania. W wielu interesujących zastosowaniach modeli graficznych może jednak wystąpić problem z niepełnym zestawem danych dla zmiennej. Kolejnym - i pokrewnym - problemem jest istnienie w modelu ewentualnych ukrytych lub utajonych zmiennych. W takich przypadkach algorytm maksymalizacji oczekiwań (lub EM) służy jako potężne narzędzie do wnioskowania o prawdopodobnych szacunkach dla wspólnych rozkładów. Algorytm EM został po raz pierwszy wyjaśniony i nadano mu nazwę w klasycznej pracy A. Dempstera, N. Lairda i D. Rubina. W artykule uogólniono metodologię i rozwinięto stojącą za nią teorię. EM okazała się jedną z najpopularniejszych metod uczenia się w modelowaniu statystycznym i probabilistycznym. Algorytm EM jest używany w statystykach do znajdowania oszacowań parametrów maksymalnego prawdopodobieństwa w modelach probabilistycznych, gdzie model zależy od możliwych nieobserwowanych zmiennych latentnych. EM to iteracyjny algorytm, który naprzemiennie wykonuje krok oczekiwania (E), który oblicza oczekiwanie prawdopodobieństwa zmiennej poprzez uwzględnienie zmiennych latentnych tak, jakby były obserwowane, oraz krok maksymalizacji (M), który oblicza oszacowania maksymalnego prawdopodobieństwa parametrów z oczekiwanego prawdopodobieństwa znalezionego w kroku E. Parametry znalezione na kroku M są następnie używane do rozpoczęcia kolejnego kroku E, a proces jest powtarzany do osiągnięcia zbieżności, to znaczy do momentu aż pozostaje bardzo mała oscylacja między wartościami w krokach E i M. W sekcji 13.2.3 przedstawiamy przykład algorytmu maksymalizacji oczekiwań.

Maksymalizacja oczekiwań dla HMM (Baum-Welch)

Ważnym narzędziem do uczenia parametrów jest Baum-Welch, wariant algorytmu EM. Oblicza oszacowania największej wiarygodności i oszacowania trybu późniejszego dla parametrów (prawdopodobieństwa przejścia i emisji) HMM lub innego czasowego modelu graficznego, gdy podane są tylko zestawy danych treningowych emisji (obserwowalnych). Algorytm Baum-Welch ma dwa kroki:

1. Oblicz prawdopodobieństwa w przód i wstecz dla każdego stanu czasowego;

2. Na podstawie 1 wyznaczyć częstość wartości par przejścia-emisja i podzielić przez prawdopodobieństwo całej sekwencji.

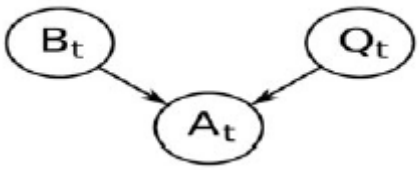
Krok 1 Baum-Welch wykorzystuje algorytm do przodu-do tyłu. Jest to algorytm obsługujący programowanie dynamiczne.

Krok 2 algorytmu Bauma-Welcha służy do obliczenia prawdopodobieństwa wystąpienia określonej sekwencji wyjściowej, przy danych parametrach modelu, w kontekście ukrytych modeli Markowa lub innych modeli czasowych. Krok 2 sprowadza się do obliczenia spodziewanej liczby konkretów para przejście-emisja. Za każdym razem, gdy zostanie znalezione przejście, wartość stosunku przejścia do prawdopodobieństwa całej sekwencji wzrasta i ta wartość może być następnie zmieniona na nową wartość przejścia. Brute force procedura rozwiązywania problemu uczenia parametrów HMM polega na wygenerowaniu wszystkich możliwych sekwencji obserwowanych zdarzeń i stanów ukrytych wraz z ich prawdopodobieństwami przy użyciu dwóch macierzy przejść. Łączne prawdopodobieństwo dwóch sekwencji, biorąc pod uwagę model oblicza się poprzez pomnożenie odpowiednich prawdopodobieństw w macierzach. Procedura ta ma złożoność czasową O ($2TN^T$), gdzie T jest długością

sekwencji obserwowanych zdarzeń, a N jest całkowitą liczbą symboli w alfabecie stanu. Tym razem koszt złożoności jest nie do pokonania w przypadku realistycznych problemów, dlatego wybrana metoda wdrażania Baum-Welch wykorzystuje programowanie dynamiczne. Następnie przedstawiamy prosty przykład uczenia parametrów przy użyciu propagacji zapętlonych przekonań z algorytmem maksymalizacji oczekiwań .

13.2.3 Maksymalizacja oczekiwań: przykład

Zilustrujemy maksymalizację oczekiwań (EM) na przykładzie i używamy algorytmu propagacji zapętlonych przekonań Pearl'a , aby pokazać kroki wnioskowania. Ten przykład został zaimplementowany w programie, a reprezentacja pola losowego Markowa jest zaczerpnięta z logiki pętli ,stochastycznego języka modelowania pierwszego rzędu. Chociaż dostępne są prostsze metody rozwiązywania ograniczeń przedstawionego przez nas przykładu alarmu włamaniowego, naszym celem jest zademonstrowanie ważnego schematu wnioskowania dla BBN, propagacji zapętlonych przekonań i algorytmu EM w zrozumiałym otoczeniu. Rozważmy sieć przekonań bayesowskich przedstawioną na rysunku 13.10, BBN z 3 węzłami, alarmem A, włamaniem B i trzęsieniem ziemi Q.



Model ten opisuje system antywłamaniowy At, aktywny w czasie obserwacji t, zależny przyczynowo od możliwości włamania B_t lub trzęsienie ziemi Qt. Model ten, z pewnymi założonymi miarami prawdopodobieństwa, można opisać w logice pętli przez program:

W | B_t, Q_t = [[[.999, .001], [. 7, .3]], [[. 8, .2], [. 1, .9]]]

Uprościliśmy rzeczywistą składnię logiki pętli, aby prezentacja była wyraźniejsza. W tym programie zakładamy, że wszystkie zmienne są logiczne. Zakładamy również, że BBN z rysunku 10 zawiera określony rozkład prawdopodobieństwa warunkowego (CPD) powyżej. Na przykład tabela podaje, że prawdopodobieństwo nie zadziałania alarmu, biorąc pod uwagę włamanie i trzęsienie ziemi, wynosi 0,001; prawdopodobieństwo uruchomienia alarmu, biorąc pod uwagę ani włamanie, ani trzęsienie ziemi, wynosi 0,1. Zauważ również, że ten program określa ogólną klasę BBN, a raczej BBN dla każdego czasu t. Jest to nieco podobne do podejścia reprezentacyjnego przyjętego przez Kersting i DeRaedt (2000). Jednak logika pętli rozszerza podejście przyjęte przez Kerstinga i DeRaedta na kilka sposobów. Na przykład logika loopy przekształca zdania logiki probabilistycznej, takie jak to powyżej, w pole losowe Markowa, aby zastosować algorytm wnioskowania, propagację zapętlonych przekonań (Pearl 1988). Pole losowe Markowa lub sieć Markowa to probabilistyczny model graficzny, którego struktura jest grafem nieukierunkowanym (w porównaniu z ukierunkowanymi modelami graficznymi sieci przekonań bayesowskich). Węzły grafu struktury pola losowego Markowa odpowiadają zmiennym losowym, a powiązania między węzłami odpowiadają zależnościom między połączonymi zmiennymi losowymi. W logice loopy pole losowe Markowa jest używane jako pośrednia i równoważna reprezentacja oryginalnego modelu graficznego (tak jak drzewo kliki zostało użyte w rozdziale 9.3), pole losowe Markowa wspiera wnioskowanie probabilistyczne, w naszym przypadku propagacja przekonań o pętli . Ilościowe składowe pola losowego Markowa są określone przez zbiór funkcji potencjalnych, których dziedziną jest klika struktury grafowej pola losowego Markowa, i które definiują zgodność między zbiorem wszystkich możliwych wspólnych przypisań do zmiennej z kliki i zbioru

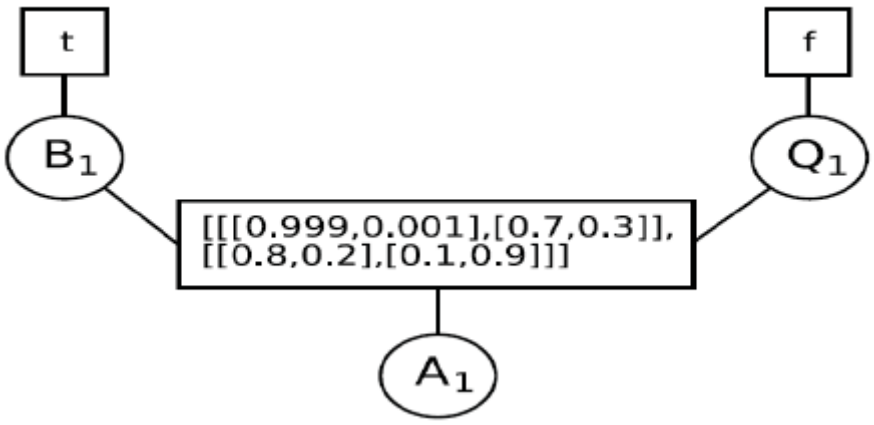
nieujemnych liczb rzeczywistych. W systemie logiki pętli używana jest specjalna wersja pola losowego Markowa. Do każdej zależności między zmiennymi losowymi przypisana jest funkcja potencjalna. W konsekwencji, losowe pole Markowa może być reprezentowane przez dwudzielny graf z dwoma typami węzłów, zmiennymi węzłami i węzłami klastra. Węzły zmienne odpowiadają zmiennym losowym, podczas gdy węzły klastrów odpowiadają potencjalnym funkcjom zmiennych losowych określonych przez sąsiadów węzła klastra. Rysunek 11 ilustruje dwudzielne pole losowe Markowa, które obejmuje potencjalne funkcje BBN z rysunku 10. W logice loopy możemy określić fakty dotyczące modelu i zadawać pytania; na przykład możemy stwierdzić:

B_1 = t.

Q_1 = f.

A_1 =?

Interpretacja tych trzech stwierdzeń jest taka, że wiemy, że w czasie 1 doszło do włamania, nie było trzęsienia ziemi i chcielibyśmy poznać prawdopodobieństwo włączenia alarmu. Używając tych trzech instrukcji wraz z $A_t \mid B_t, Q_t$ = [[[.999, .001], [.7, .3]], [[.8, .2], [.1, .9]]], pętlowy system logiczny konstruuje losowe pole z faktami i potencjalnymi funkcjami przedstawionymi na Rysunku 11.



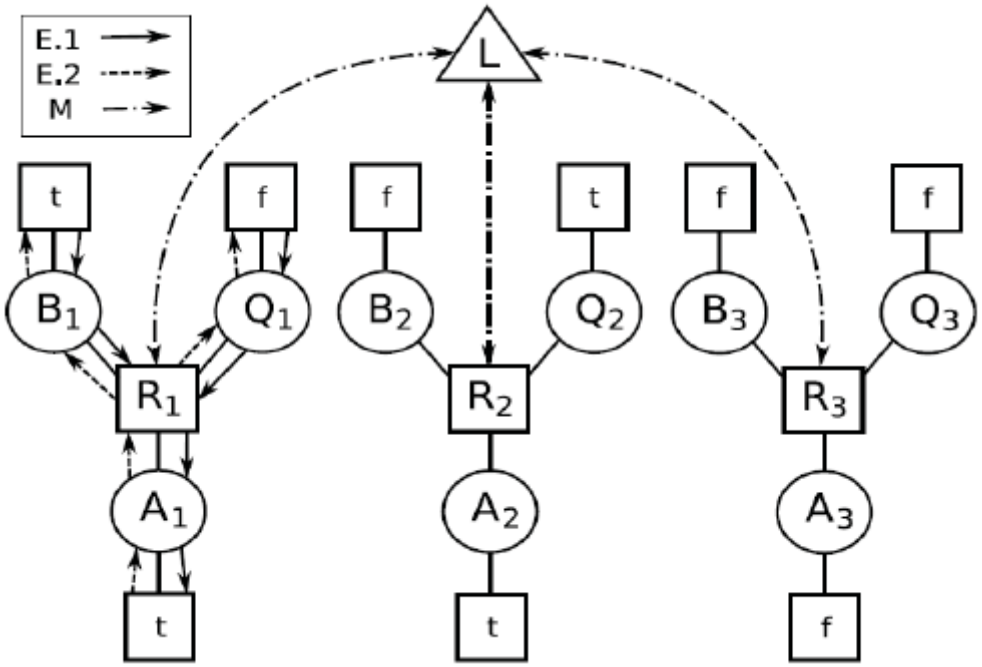
Zauważ, że na rysunku 11 fakty z programu są włączone do pola losowego Markowa jako węzły klastra-liścia, a tabela warunkowego rozkładu prawdopodobieństwa (CPD) jest zakodowana w węźle klastra łączącym trzy zmienne węzły. Aby wywnioskować odpowiedź na to zapytanie, system stosuje algorytm propagacji zapętlonych przekonań do pola losowego Markowa. Ponieważ określiliśmy proste deterministyczne fakty, a oryginalny model graficzny nie ma pętli, algorytm wnioskowania jest dokładny (Pearl 1988) i zbiega się w jednej iteracji, zwracając rozkład dla alarmu, [0.8,0.2]. Ponadto logika pętli wspomaga uczenie się, umożliwiając użytkownikom przypisywanie możliwych do nauczenia dystrybucji do wybranych reguł programu logicznego w pętli. Parametry te można oszacować za pomocą algorytmu przekazywania wiadomości wyprowadzonego z maksymalizacji oczekiwań. Aby to zademonstrować, zakładamy następnie, że nie znamy warunkowego rozkładu prawdopodobieństwa BBN z rysunku 10. W logice pętli, aby uzyskać wartości dla tego rozkładu, użytkownik ustawia CPD jako rozkład, którego można się nauczyć w dowolnym czasie t. Aby to oznaczyć, używamy w programie zmiennej L:

$W \mid B_t, Q_t = L$

Możemy teraz przedstawić zebrany zestaw obserwacji interpreterowi logiki pętli w zbiorze okresów czasu, używając reprezentacji faktów poniżej. Dla uproszczenia zbieramy tylko trzy punkty danych w każdym z trzech przedziałów czasowych:

$Q_1 = f$
$B_1 = t$
$A_1 = t$
$Q_2 = t$
$B_2 = f$
$A_2 = t$
$Q_3 = f$
$B_3 = f$
$A_3 = f$

Logika pętli wykorzystuje kombinację schematu wnioskowania propagacji przekonań pętli w połączeniu z maksymalizacją oczekiwań w celu oszacowania parametrów modelu, których można się nauczyć. Aby to osiągnąć, interpreter konstruuje, jak poprzednio, pole losowe Markowa dla modelu. Dla alarmu BBN pokazanego na rysunku 13.10, odpowiadające mu pole losowe Markowa, z możliwym do nauczenia parametrem L, powiązanym z każdym węzłem klastra, którego reguła jest ujednolicona z regułą możliwą do nauczenia przedstawioną wcześniej (A_t | B_t, Q_t = L), przedstawiono na rysunku 12.



Następnie wypełniamy znane rozkłady BBN i dokonujemy losowej inicjalizacji w węzłach klastra R_1, R_2 i R_3, a także w węźle możliwym do nauczenia L. Na koniec system ten jest przedstawiany z danymi reprezentującymi trzy okresy, w których znane fakty są włączane do węzłów klastra typu liść, jak poprzednio. Rysunek 12 przedstawia wynik. Algorytm maksymalizacji oczekiwań jest procesem iteracyjnym realizowanym poprzez przekazywanie wiadomości. Po każdej iteracji bieżące przybliżenie tabeli prawdopodobieństwa warunkowego jest wiadomością możliwego do nauczenia się węzła do każdego z jego połączonych węzłów klastra. Kiedy węzeł, którego można się nauczyć, jest aktualizowany, każdy sąsiedni węzeł klastra wysyła wiadomość do L (strzałki-kropki oznaczone M na

rysunku 12). Ta wiadomość jest wynikiem wszystkich wiadomości przychodzących do tego węzła klastra z sąsiednich węzłów zmiennych. Te (nie znormalizowane) tabele są oszacowaniem wspólnego prawdopodobieństwa w każdym węźle klastra. Jest to rozkład na wszystkie stany zmiennych warunkowych i warunkujących. Węzeł, którego można się nauczyć, pobiera sumę wszystkich tych komunikatów klastra i konwertuje je na znormalizowaną tabelę prawdopodobieństwa warunkowego. Dokonując wnioskowania (propagacja zapętlonych przekonań) na klastrze i węzłach zmiennych, obliczamy komunikat dla możliwych do nauczenia się węzłów. Odzwierciedlając dwudzielną strukturę podstawowego pola losowego Markowa, propagacja zapętlonych przekonań odbywa się z dwoma etapami przekazywania wiadomości: wysyłaniem wiadomości ze wszystkich węzłów klastra do ich sąsiednich węzłów zmiennych, pełnymi strzałkami oznaczonymi E.1 na rysunku 13.12, a następnie z powrotem , przerywane strzałki oznaczone E.2 na rysunku 13.12. Zastosowanie tego algorytmu propagacji przekonań do momentu, gdy zbieżność da przybliżenie oczekiwanych wartości. Jest to równoważne z krokiem oczekiwania w algorytmie EM. Uśrednianie, które ma miejsce we wszystkich możliwych do nauczenia węzłach klastra, odpowiada krokowi maksymalizacji zwracającemu oszacowanie maksymalnego prawdopodobieństwa parametrów w możliwym do nauczenia się węźle. Iteracja, która obsługuje konwergencję w węzłach zmiennych i klastrach, po której następuje aktualizacja węzłów, których można się nauczyć, a następnie kontynuowanie iteracji jest równoważne z pełnym algorytmem EM.

Podsumowując algorytm, jak pokazano na rysunku 12:

W oczekiwaniu lub kroku E:

1. Wszystkie węzły klastra wysyłają swoje komunikaty do węzłów zmiennych.

2. Wszystkie węzły zmienne wysyłają swoje komunikaty z powrotem do węzłów klastra, aktualizując informacje o węzłach klastra.

W kroku maksymalizacji lub M:

1. Węzły klastra wysyłają komunikaty zaktualizowane w kroku E. do możliwych do nauczenia się węzłów, gdzie są uśrednione.

W opisanym algorytmie węzły mogą być zmieniane w dowolnej kolejności, a aktualizacje klastra i węzłów zmiennych mogą pokrywać się z aktualizacjami węzłów uczących się. Ten iteracyjny proces aktualizacji reprezentuje rodzinę algorytmów w stylu EM, z których niektóre mogą być bardziej wydajne niż standardowe EM dla niektórych domen. Rozszerzeniem algorytmicznym, które również obsługuje ta struktura, jest uogólniony algorytm propagacji przekonań zaproponowany przez Yedidia i innych.

13.3 Stochastyczne rozszerzenia uczenia się ze wzmocnieniem

W sekcji z Części 10 po raz pierwszy wprowadziliśmy uczenie się ze wzmocnieniem, gdzie wraz ze wzmocnieniem nagrody agent uczy się podejmować różne zestawy działań w środowisku. Celem agenta jest maksymalizacja długoterminowej nagrody. Ogólnie rzecz biorąc, agent chce nauczyć się polityki lub mapowania między stanami nagrody na świecie a jego własnymi działaniami na świecie. Aby zilustrować koncepcje uczenia się ze wzmocnieniem, rozważmy przykład robota recyklingowego. Rozważmy robota mobilnego, którego zadaniem jest zbieranie pustych puszek po napojach z biur. Robot posiada akumulator. Wyposażony jest w czujniki do wyszukiwania puszek oraz w ramię do ich zbierania. Zakładamy, że robot posiada system sterowania do interpretacji informacji sensorycznych, nawigacji i poruszania ramieniem w celu zbierania puszek. Robot wykorzystuje uczenie się przez

wzmacnianie oparte na aktualnym poziomie naładowania swojej baterii, aby wyszukać puszki. Agent musi podjąć jedną z trzech decyzji:

1. Robot może aktywnie szukać puszki po napojach w określonym przedziale czasu;

2. Robot może się zatrzymać i czekać, aż ktoś przyniesie mu puszkę; lub

3. Robot może wrócić do bazy domowej, aby naładować swoje baterie.

Robot podejmuje decyzje w określonym przedziale czasowym, po znalezieniu pustej puszki lub w przypadku wystąpienia innego zdarzenia. W konsekwencji robot wykonuje trzy akcje, a jego stan jest określany przez poziom naładowania akumulatora. Nagroda jest ustawiona na zero przez większość czasu staje się dodatnia po zebraniu puszki, a nagroda jest ujemna, jeśli bateria wyczerpie się. Należy zauważyć, że w tym przykładzie agent uczący się oparty na wzmocnieniach jest częścią robota, która monitoruje zarówno stan fizyczny robota, jak i sytuację (stan) jego otoczenia zewnętrznego: agent robot monitoruje zarówno stan robota a także otoczenie zewnętrzne. Istnieje poważne ograniczenie tradycyjnej struktury uczenia się przez wzmacnianie, którą właśnie opisaliśmy: ma ona charakter deterministyczny. Aby przezwyciężyć to ograniczenie i móc wykorzystać uczenie się przez wzmacnianie w szerszym zakresie złożonych zadań, rozszerzamy ramy deterministyczne o składnik stochastyczny. W kolejnych rozdziałach rozważymy dwa przykłady stochastycznego uczenia się ze wzmocnieniem: proces decyzyjny Markowa i częściowo obserwowalny proces decyzyjny Markowa.

13.3.1 Proces decyzyjny Markowa (lub MDP)

Przykłady uczenia się przez wzmacnianie przedstawione do tej pory, zarówno w sekcji 10.7, jak i we wprowadzeniu do sekcji 13.3, miały charakter deterministyczny. W efekcie, gdy agent wykonuje akcję z określonego stanu w czasie t, zawsze skutkuje to deterministycznie określony nowy stan w czasie t +1: system jest całkowicie deterministyczny. Jednak w wielu rzeczywistych zastosowaniach uczenia się przez wzmacnianie może tak nie być: agent nie ma pełnej wiedzy ani kontroli nad następnym stanem świata. W szczególności, podjęcie działania ze stanu st może prowadzić do więcej niż jednego możliwego stanu wynikowego w czasie t + 1. Na przykład w grze w szachy, kiedy wykonujemy określony ruch, często nie wiemy, jaki ruch wykona nasz przeciwnik. Nie mamy pełnej wiedzy, w jakim stanie będzie świat w wyniku podjęcia naszego ruchu. W rzeczywistości ta niepewność dotyczy wielu gier wieloosobowych. Drugi przykład to loteria lub inne gry losowe. Wybieramy ruch niepewny co do prawdopodobnej nagrody. Trzeci przykład dotyczy podejmowania decyzji przez jednego agenta (a nie rywalizacji z jednym lub większą liczbą innych agentów), kiedy świat jest tak złożony, że deterministyczna odpowiedź na wybór stanu nie jest obliczalna. Ponownie, w tej sytuacji odpowiedź oparta na prawdopodobieństwach może być najlepszą, jaką możemy uzasadnić. We wszystkich tych sytuacjach potrzebujemy mocniejszych ram, które pozwolą rozwiązać problem uczenia się przez wzmacnianie. Jedną z takich ram jest proces decyzyjny Markowa (lub MDP). MDP są oparte na własności Markowa pierwszego rzędu, która stwierdza, że przejście do następnego stanu jest reprezentowane przez rozkład prawdopodobieństwa, który zależy tylko od aktualnego stanu i możliwych działań. Jest niezależny od wszystkich stanów poprzedzających stan obecny. MDP to stochastyczne procesy dyskretne składające się z zestawu stanów, zestawu działań, które mogą być wykonane z każdego stanu oraz funkcji nagrody związanej z każdym stanem. W rezultacie MDP zapewniają bardzo potężne ramy, które można wykorzystać do modelowania szerokiej klasy sytuacji uczenia się przez wzmacnianie. Następnie definiujemy MDP:

DEFINICJA

PROCES DECYZJI MARKOV, czyli MDP

Proces decyzyjny Markowa to krotka <S, A, P, R>, gdzie:

S jest zbiorem stanów i

A to zbiór działań.

$p_a(s_t, s_{t+1}) = p(s_{t+1} \mid st, a_t = a)$ jest prawdopodobieństwem, że jeśli agent wykona akcję $a \in A$ ze stanu s_t w czasie t, skutkuje to stanem s_{t+1} w czasie t +1. Ponieważ prawdopodobieństwo $p_a \in P$ jest zdefiniowane w całej przestrzeni stanów działań, często jest przedstawiane za pomocą macierzy przejść.

R(s) to nagroda otrzymana przez agenta w stanie s.

Powinniśmy zauważyć, że jedyną różnicą między schematem MDP dla uczenia się ze wzmocnieniem a prezentacją uczenia się ze wzmocnieniem z sekcji z Części 10 jest to, że funkcja przejścia jest teraz zastąpiona funkcją rozkładu prawdopodobieństwa. Jest to ważna modyfikacja naszej teorii, która pozwala nam uchwycić niepewność w świecie, gdy świat jest albo nieobliczalnie złożony, albo w rzeczywistości jest stochastyczny. W takich przypadkach reakcja nagrody na działanie agenta jest najlepiej przedstawiana jako rozkład prawdopodobieństwa. Zauważyliśmy, że MDP można wykorzystać do modelowania gry takiej jak szachy, w której następny stan świata może znajdować się poza deterministyczną kontrolą agenta. Kiedy wykonujemy ruch w szachach, wynikowy stan gry zależy od ruchu naszego przeciwnika. Dzięki temu jesteśmy świadomi obecnego stanu, w jakim się znajdujemy, jesteśmy świadomi, jakie działanie (ruch) podejmujemy, ale wypadkowy stan nie jest dla nas natychmiast dostępny. Mamy tylko rozkład prawdopodobieństwa na zestaw możliwych stanów, w których prawdopodobnie się znajdziemy, w zależności od ruchu naszego przeciwnika. Rozważmy teraz grę w pokera. Tutaj nie jesteśmy nawet pewni naszego obecnego stanu! Znamy karty, które posiadamy, ale nie mamy doskonałej wiedzy o kartach naszych przeciwników. Możemy się tylko domyślać, być może dzięki wiedzy z licytacji, jakie karty ma nasz przeciwnik. Świat jest jeszcze bardziej niepewny niż świat MDP. Nie tylko nie wiemy, w jakim stanie się znajdujemy, ale musimy również wykonać akcję w oparciu o nasze przypuszczenia co do rzeczywistego stanu świata. W rezultacie nasze przypuszczenia co do wywołanego działaniem nowego stanu świata będą jeszcze bardziej niepewne. Dlatego potrzebujemy bogatszych ram niż MDP, aby modelować tę podwójnie niepewną sytuację. Jedną z ram realizacji tego zadania jest częściowo obserwowalny proces decyzyjny Markowa (lub POMDP). Nazywa się to POMDP, ponieważ możemy tylko częściowo obserwować stan, w którym się znajdujemy, oraz reakcję przeciwnika. Zamiast doskonałej wiedzy o naszym obecnym stanie, mamy tylko rozkład prawdopodobieństwa na możliwy zbiór stanów, w których moglibyśmy się ewentualnie znaleźć. Definiujemy POMDP:

DEFINICJA

CZĘŚCIOWO OBSERWOWALNY PROCES DECYZJI MARKOV lub POMDP

Częściowo obserwowalny proces decyzyjny Markowa to krotka <S, A, O, P, R>, gdzie:

S jest zbiorem stanów i

A to zbiór działań.

O to zbiór obserwacji oznaczających, co agent może zobaczyć w swoim świecie. Ponieważ agent nie może bezpośrednio obserwować swojego aktualnego stanu, obserwacje są prawdopodobnie powiązane z podstawowym faktycznym stanem świata. $p_a(s_t, o, s_{t+1}) = p(s_{t+1}, o_t = o \mid s_t, a_t = a)$ to prawdopodobieństwo, że gdy agent wykona akcję a ze stanu s_t w czasie t, skutkuje to obserwacją o co

prowadzi do stanu podstawowego s_{t+1} w czasie t + 1. R (s_t, a, s_{t+1}) to nagroda otrzymana przez agenta, gdy wykonuje akcję a w stanie s_t i przechodzi do stanu s_{t+1}.

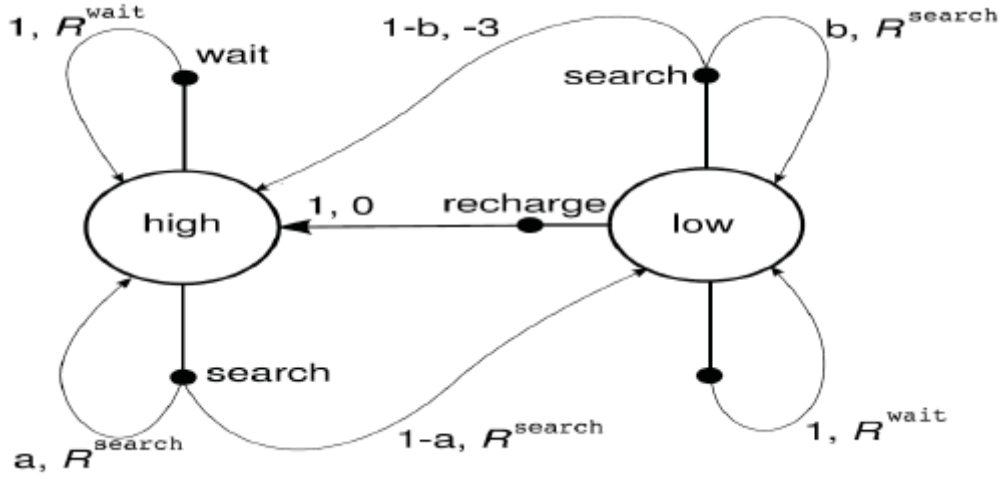
Podsumowując, MDP można uznać za szczególny przypadek POMDP, gdy obserwacja o daje nam dokładne wskazanie stanu aktualnego. W zrozumieniu tego rozróżnienia pomocna może być analogia: MDP jest powiązany z łańcuchem Markowa, tak jak POMDP jest powiązany z ukrytym modelem Markowa. Istnieje kilka algorytmów tworzenia MDP i POMDP. Złożoność tych algorytmów jest oczywiście większa niż w przypadku czysto deterministycznych przypadków przedstawionych wcześniej. W rzeczywistości algorytmy rozwiązywania POMDP są obliczeniowo niewykonalne. Oznacza to, że znalezienie optymalnego rozwiązania problemu przy użyciu POMDP w czasie wielomianowym może być niemożliwe, a rozwiązanie może być rzeczywiście nieobliczalne. W rezultacie rozwiązujemy te problemy za pomocą algorytmów, które przybliżają optymalne rozwiązanie.

13.3.3 Przykładowa implementacja procesu decyzyjnego Markowa

Aby zilustrować prosty MDP, ponownie zajmiemy się przykładem robota recyklingowego przedstawionego na początku sekcji 13.3. Przypomnij sobie, że za każdym razem, gdy ten agent uczący się wzmacniania napotyka zdarzenie zewnętrzne, podejmuje decyzję o aktywnym poszukiwaniu puszki po napojach do recyklingu, nazywamy to wyszukiwaniem, czekamy, aż ktoś przyniesie puszkę, nazywamy to czekaniem lub powrotem do bazę domową, aby naładować baterię, naładuj. Te decyzje są wykonywane przez robota wyłącznie na podstawie stanu energii poziom naładowania baterii. Dla uproszczenia rozróżnia się dwa poziomy baterii, niski i wysoki: przestrzeń stanów S = {niski, wysoki}. Jeśli aktualny stan akumulatora jest wysoki, akumulator jest ładowany, w przeciwnym razie jest niski. Dlatego zestawy działań agenta to A (niski) = {szukaj, czekaj, doładuj} i A (wysoki) = {szukaj, czekaj}. Jeśli A (wysoka) obejmuje ładowanie, spodziewalibyśmy się, że agent nauczy się, że polityka wykorzystująca to działanie byłaby nieoptymalna! Stan baterii jest określany niezależnie od podjętych działań. Jeśli jest wysoka, spodziewamy się, że robot będzie w stanie ukończyć okres aktywnego poszukiwania bez ryzyka wyczerpania baterii. Stąd, jeśli stan jest wysoki, to agent pozostaje w tym stanie z prawdopodobieństwem a i zmienia swój stan na niski z prawdopodobieństwem 1-a. Jeśli agent zdecyduje się przeprowadzić aktywne wyszukiwanie będąc w stanie niskim, poziom energii baterii pozostanie to samo z prawdopodobieństwem b, a bateria wyczerpie się z prawdopodobieństwem 1–b. Następnie tworzymy system nagród: gdy bateria jest wyczerpana, S = niski, robot jest uratowany, jego bateria jest ładowana do wysokiego poziomu i otrzymuje nagrodę w wysokości -3. Za każdym razem, gdy robot znajdzie puszkę, otrzymuje nagrodę w wysokości 1. Oczekiwaną ilość puszek oznaczamy robot zbierze (oczekiwaną nagrodę, którą agent otrzyma) podczas aktywnego wyszukiwania i podczas oczekiwania przez R_search i R_wait, i przyjmie, że R_search> R_wait. Zakładamy również, że robot nie może odebrać żadnych puszek po napojach wracając do naładowania oraz że nie może zebrać żadnych puszek podczas kroku, na którym bateria jest słaba. Opisany właśnie system może być reprezentowany przez skończone MDP, którego prawdopodobieństwa przejścia (pa (s_t, s_{t+1})) i oczekiwane nagrody są podane w tabeli 1.

s_t	s_{t+1}	a_t	$p_a(s_t,s_{t+1})$	$R_a(s_t,s_{t+1})$
high	high	search	a	R_search
high	low	search	1 - a	R_search
low	high	search	1 - b	-3
low	low	search	b	R_search
high	high	wait	1	R_wait
high	low	wait	0	R_wait
low	high	wait	0	R_wait
low	low	wait	1	R_wait
low	high	recharge	1	0
low	low	recharge	0	0

Aby przedstawić dynamikę tego skończonego MDP, możemy użyć wykresu, takiego jak Rysunek 13, wykres przejścia MDP dla robota recyklingowego.



Graf przejścia ma dwa typy węzłów: węzły stanu i węzły działań. Każdy stan agenta jest reprezentowany przez węzeł stanu przedstawiony jako duży okrąg z nazwą stanu, s, wewnątrz. Każda para stan-akcja jest reprezentowana przez węzeł akcji połączony z węzłem stanu oznaczonym jako małe czarne kółko. Jeśli agent uruchomi się w stanie s i podejmie akcję a, porusza się wzdłuż linii od węzła stanu s do węzła akcji (s_t, a). W konsekwencji środowisko reaguje przejściem do węzła następnego stanu przez jedną ze strzałek wychodzących z węzła akcji (st, a), gdzie każda strzałka odpowiada trójce (s_t, a, s_{t+1}), gdzie s_{t+1} to następny stan. Strzałka jest oznaczona prawdopodobieństwem przejścia, pa (s_t, s_{t+1}), a oczekiwana nagroda za przejście, $R_a(s_t, s_{t+1})$, jest reprezentowana przez strzałkę. Zwróć uwagę, że prawdopodobieństwa przejścia związane ze strzałkami opuszczającymi węzeł akcji zawsze sumują się do 1.