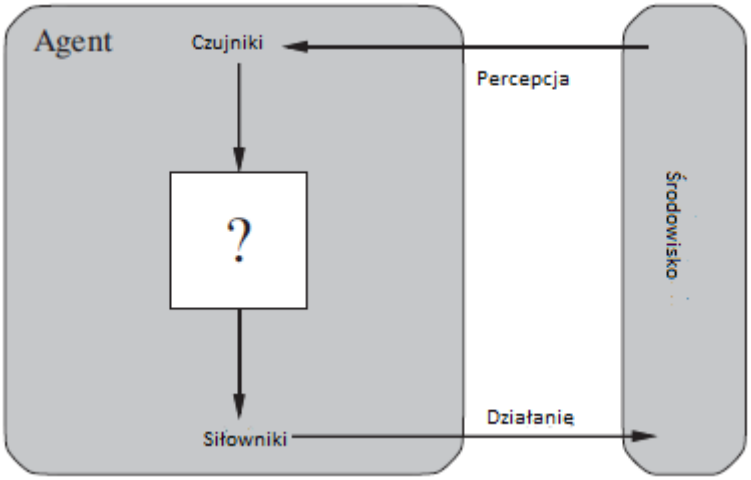
INTELIGENTNI AGENCI

Część 1 określiła pojęcie racjonalnych czynników jako kluczowe dla naszego podejścia do sztucznej inteligencji. W tej części konkretyzujemy to pojęcie. Przekonamy się, że pojęcie racjonalności można zastosować do szerokiej gamy agentów działających w dowolnym możliwym środowisku. Naszym planem jest wykorzystanie tej koncepcji do opracowania małego zestawu zasad projektowania do budowania skutecznych agentów - systemów, które można rozsądnie nazwać inteligentnymi. Zaczynamy od zbadania agentów, środowisk i powiązań między nimi. Obserwacja, że niektórzy agenci zachowują się lepiej niż inni, naturalnie prowadzi do idei agenta racjonalnego - takiego, który zachowuje się tak dobrze, jak to możliwe. To, jak agent może się zachowywać, zależy od charakteru środowiska; niektóre środowiska są trudniejsze niż inne. Podajemy przybliżoną kategoryzację środowisk i pokazujemy, w jaki sposób właściwości środowiska wpływają na projektowanie odpowiednich czynników dla tego środowiska. Opisujemy szereg podstawowych projektów agentów „szkieletowych", które omówimy w pozostałej części

AGENCI I ŚRODOWISKO

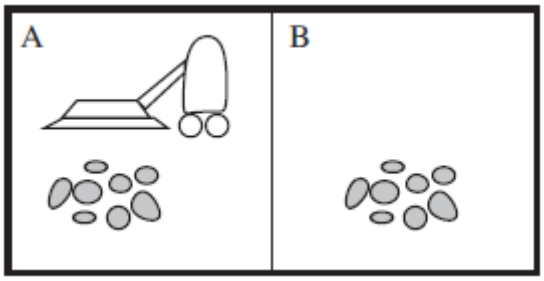
Agent to wszystko, co można postrzegać jako postrzeganie jego środowiska za pomocą czujników i działanie na to środowisko za pomocą siłowników. Ten prosty pomysł ilustruje rysunek



Ludzki agent ma oczy, uszy i inne narządy do czujników i rąk, nóg, dróg głosowych i tak dalej do siłowników. Robot może mieć kamery i dalmierze na podczerwień dla czujników i różne silniki dla siłowników. Agent oprogramowania odbiera naciśnięcia klawiszy, zawartość pliku i pakiety sieciowe jako dane sensoryczne i działa na środowisko, wyświetlając na ekranie, zapisując pliki i wysyłając pakiety sieciowe. Używamy terminu percepcja w odniesieniu do danych percepcyjnych agenta w dowolnym momencie. Sekwencja percepcji agenta to pełna historia wszystkiego, co agent kiedykolwiek widział.

Ogólnie rzecz biorąc, wybór działania agenta w danym momencie może zależeć od całej dotychczasowej sekwencji percepcji, ale nie od niczego, czego nie dostrzegł. Określając wybór działania agenta dla każdej możliwej sekwencji percepcji, powiedzieliśmy mniej więcej wszystko, co można powiedzieć o agencie. Z matematycznego punktu widzenia mówimy, że zachowanie agenta jest opisane przez funkcję agenta, która odwzorowuje dowolną określoną sekwencję percepcji na akcję. Możemy sobie wyobrazić zestawienie funkcji agenta opisującej dowolnego agenta; dla większości agentów byłby to bardzo duży stół - w rzeczywistości nieskończony, chyba że ograniczymy długość sekwencji percepcyjnych, które chcemy rozważyć. Biorąc pod uwagę agenta, z którym można eksperymentować, możemy w zasadzie zbudować tę tabelę, wypróbowując wszystkie możliwe

sekwencje percepcji i rejestrując, jakie działania agent wykonuje w odpowiedzi. Tabela jest oczywiście zewnętrzną charakterystyką agenta. Wewnętrznie funkcja agenta dla sztucznego agenta zostanie zaimplementowana przez program agenta. Ważne jest, aby te dwa pomysły były odrębne. Funkcja agenta jest abstrakcyjnym opisem matematycznym; program agenta jest konkretną implementacją działającą w ramach jakiegoś systemu fizycznego. Aby zilustrować te pomysły, używamy bardzo prostego przykładu - świata odkurzaczy pokazanego na rysunku



Ten świat jest tak prosty, że możemy opisać wszystko, co się dzieje; to także wymyślony świat, więc możemy wymyślić wiele odmian. Ten konkretny świat ma tylko dwie lokalizacje: kwadraty A i B. Odkurzacz rozpoznaje, w którym kwadracie jest i czy na kwadracie jest brud. Może wybrać ruch w lewo, ruch w prawo, ssanie ziemi lub nic nie robić. Jedną z bardzo prostych funkcji agenta jest: jeśli bieżący kwadrat jest brudny, to ssij; w przeciwnym razie przejdź na drugi kwadrat. Częściowe zestawienie tej funkcji agenta pokazano na rysunku, a program agenta, który ją implementuje, pojawi się później.

Percepcja sekwencji	**:**	**Działanie**
[A, Czyść]	:	Prawo
[A, Brud]	:	Ssanie
[B, Czyść]	:	Lewo
[B, Brud]	:	Ssanie
[A, Czyść], [A, Czyść]	:	Prawo
[A, Czyść], [A, Brud]	:	Ssanie
...		
...		
[A, Czyść], [A, Czyść], [A, Czyść]	:	Prawo
[A, Czyść], [A, Czyść], [A, Brud]	:	Ssanie
...		

Patrząc , widzimy, że różnych agentów świata próżniowego można zdefiniować po prostu wypełniając prawą kolumnę ręcznie na różne sposoby. Oczywiste pytanie brzmi zatem: jaki jest właściwy sposób na wypełnienie tabeli? Innymi słowy, co sprawia, że agent jest dobry lub zły, inteligentny lub głupi? Odpowiemy na te pytania w następnej sekcji. Przed zamknięciem tej części powinniśmy podkreślić, że pojęcie agenta ma być narzędziem do analizy systemów, a nie bezwzględną charakterystyką, która dzieli świat na agentów i nieagentów. Można postrzegać ręczny kalkulator jako agenta, który wybiera akcję wyświetlania „4”, gdy otrzyma sekwencję percepcji „2 + 2 =”, ale taka analiza raczej nie pomogłaby nam zrozumieć kalkulatora. W pewnym sensie wszystkie dziedziny inżynierii można

postrzegać jako projektowanie artefaktów, które wchodzą w interakcje ze światem; Sztuczna inteligencja działa na najbardziej interesującym końcu spektrum, gdzie artefakty mają znaczące zasoby obliczeniowe, a środowisko zadań wymaga nietrywialnych decyzji.

DOBRE ZACHOWANIE: POJĘCIE RACJONALNOŚCI

Racjonalny agent to taki, który robi właściwie, pod względem koncepcyjnym, tak ,że każdy wpis w tabeli dla funkcji agenta jest wypełniany poprawnie. Oczywiście robienie dobrych rzeczy jest lepsze niż robienie złych rzeczy, ale co to znaczy robić dobre rzeczy? Na to odwieczne pytanie odpowiadamy odwiecznie: rozważając konsekwencje zachowania agenta. Gdy agent jest pogrążony w środowisku, generuje sekwencję działań zgodnie z odbieranymi przez niego percepcjami. Ta sekwencja działań powoduje, że środowisko przechodzi przez sekwencję stanów. Jeśli sekwencja jest pożądana, to agent działał dobrze. To pojęcie celowości jest uchwycone przez miarę wydajności, która ocenia dowolną sekwencję stanów środowiska. Zauważ, że mówiliśmy o stanach środowiska, a nie o stanach agentów. Jeśli zdefiniujemy sukces w kategoriach opinii agenta na temat własnego działania, agent mógłby osiągnąć idealną racjonalność, po prostu łudząc się, że jego wydajność była idealna. Zwłaszcza ludzcy agenci są znani z „kwaśnych winogron" - wierząc, że tak naprawdę nie chcieli czegoś (np. Nagrody Nobla) .Oczywiście nie ma jednej stałej miary wydajności dla wszystkich zadań i agentów; zazwyczaj projektant opracuje taki, który będzie odpowiedni do okoliczności. To nie jest tak proste, jak się wydaje. Weźmy na przykład środek odkurzający z poprzedniej części. Możemy zaproponować pomiar wydajności przez ilość brudu oczyszczonego podczas jednej ośmiogodzinnej zmiany. W przypadku racjonalnego agenta oczywiście o to prosisz. Rozsądny agent może zmaksymalizować ten miernik wydajności, usuwając brud, a następnie zrzucając go na podłogę, a następnie ponownie czyszcząc i tak dalej. Bardziej odpowiednia miara wydajności wynagrodziłaby agenta za czystą podłogę. Na przykład jeden punkt może być przyznany za każdy czysty kwadrat na każdym etapie (być może z karą za zużycie energii elektrycznej i generowany hałasu). Zasadniczo lepiej jest projektować miary wydajności zgodnie z tym, czego naprawdę chce się w środowisku, niż z tym, jak według nas agent powinien się zachowywać. Nawet jeśli uda się uniknąć oczywistych pułapek, pozostają pewne kłopotliwe problemy do rozwiązania. Na przykład pojęcie „czystej podłogi" w poprzednim akapicie opiera się na średniej czystości w czasie. Jednak tę samą średnią czystość można osiągnąć dzięki dwóm różnym czynnikom, z których jeden cały czas ma przeciętną pracę, a drugi czyści energicznie, ale robi długie przerwy. To, co jest lepsze, może wydawać się dobrym punktem nauki o sprzątaniu, ale w rzeczywistości jest to głębokie pytanie filozoficzne o daleko idących konsekwencjach. Co jest lepsze - lekkomyślne życie wzlotów i upadków, czy też bezpieczna, ale pokorna egzystencja? Która jest lepsza - gospodarka, w której wszyscy żyją w umiarkowanym ubóstwie, lub taka, w której niektórzy żyją w obfitości, a inni są bardzo biedni? Te pytania pozostawiamy ćwiczeniu dla sumienia czytelnika.

Racjonalność

To, co jest racjonalne w danym momencie, zależy od czterech rzeczy:

* Miara wydajności, która określa kryterium sukcesu.

* Wcześniejsza wiedza agenta na temat środowiska.

* Działania, które agent może wykonać.

* Dotychczasowa sekwencja percepcyjna agenta.

Prowadzi to do definicji racjonalnego agenta:

„Dla każdej możliwej sekwencji percepcji racjonalny agent powinien wybrać działanie, które ma zmaksymalizować miarę jego wydajności, biorąc pod uwagę dowody dostarczone przez sekwencję percepcji i jakąkolwiek wbudowaną wiedzę, jaką agent ma."

Rozważ prosty odkurzacz, który czyści kwadrat, jeśli jest brudny, i przesuwa się na inny kwadrat, jeśli nie. Czy to racjonalny agent? To zależy! Po pierwsze, musimy powiedzieć, co to jest miara wydajności, co wiadomo o środowisku oraz jakie czujniki i elementy wykonawcze ma agent. Załóżmy, że:

* Miara wydajności przyznaje jeden punkt za każdy czysty kwadrat za każdym razem, w ciągu „cyklu życia" 1000 kroków czasowych.

* „Geografia" środowiska jest znana z góry, ale rozkład zabrudzenia i początkowa lokalizacja czynnika nie są znane. Czyste kwadraty pozostają czyste, a ssanie czyści bieżący kwadrat. Akcje Lewo i Prawo przesuwają agenta w lewo i prawo, z wyjątkiem sytuacji, gdy zabrałoby to agenta poza środowisko, w którym to przypadku agent pozostaje na swoim miejscu.

* Jedynymi dostępnymi akcjami są Lewo, Prawo i Ssanie.

* Agent poprawnie rozpoznaje swoją lokalizację i to, czy ta lokalizacja zawiera brud.

Twierdzimy, że w tych okolicznościach agent jest rzeczywiście racjonalny; jego oczekiwana wydajność jest co najmniej tak wysoka, jak każdego innego agenta. Łatwo zauważyć, że ten sam czynnik byłby irracjonalny w różnych okolicznościach. Na przykład, gdy cały brud zostanie oczyszczony, środek będzie niepotrzebnie oscylował w tę i z powrotem; jeśli miara wydajności obejmuje karę jednego punktu za każdy ruch w lewo lub w prawo, agent źle sobie poradzi. Lepszy agent dla tej sprawy nie zrobiłby nic, gdy jest pewne, że wszystkie kwadraty są czyste. Jeśli czyste kwadraty mogą ponownie się zabrudzić, agent powinien od czasu do czasu je sprawdzić i w razie potrzeby ponownie wyczyścić. Jeśli geografia środowiska jest nieznana, agent będzie musiał ją zbadać, zamiast trzymać się kwadratów A i B.

Wszechwiedza, uczenie się i autonomia

Musimy uważać, aby odróżnić racjonalność od wszechwiedzy. Wszechwiedzący agent zna rzeczywisty wynik swoich działań i może działać odpowiednio; ale wszechwiedza jest w rzeczywistości niemożliwa. Rozważ następujący przykład: pewnego dnia idę wzdłuż Pól Elizejskich i widzę starego przyjaciela po drugiej stronie ulicy. W pobliżu nie ma ruchu i nie jestem inaczej zaangażowany, więc będąc racjonalnym, zaczynam przechodzić przez ulicę. Tymczasem na 15 000 km drzwi ładunkowe spadają z przelatującego samolotu , a zanim dotrę na drugą stronę ulicy, jestem spłaszczony. Czy irracjonalne było przechodzenie przez ulicę? Jest mało prawdopodobne, że mój nekrolog przebrzmiałby „Idiota próbuje przejść przez ulicę". Ten przykład pokazuje, że racjonalność to nie to samo co doskonałość. Racjonalność maksymalizuje oczekiwaną wydajność, a perfekcja maksymalizuje rzeczywistą wydajność. Wycofanie się z wymogu doskonałości to nie tylko kwestia uczciwości wobec agentów. Chodzi o to, że jeśli spodziewamy się, że agent zrobi to, co okaże się najlepszą akcją po fakcie, nie będzie możliwe zaprojektowanie agenta spełniającego tę specyfikację - chyba że poprawimy wydajność kryształowych kul lub wehikułów czasu. Nasza definicja racjonalności nie wymaga zatem wszechwiedzy, ponieważ racjonalny wybór zależy tylko od dotychczasowej sekwencji percepcji. Musimy również upewnić się, że nie pozwoliliśmy przypadkowo agentowi podjąć zdecydowanie niedoinformowanych działań. Na przykład, jeśli agent nie patrzy w obie strony przed przejściem przez ruchliwą drogę, wówczas jego sekwencja percepcji nie powie mu, że zbliża się duża ciężarówka z dużą prędkością. Czy nasza definicja racjonalności mówi, że przejście przez ulicę jest teraz w porządku? Daleko stąd! Po pierwsze, przejście przez jezdnię nie byłoby racjonalne, biorąc pod uwagę tę mało pouczającą sekwencję percepcji: ryzyko wypadku przy przejściu bez patrzenia jest zbyt duże. Po drugie,

racjonalny agent powinien wybrać akcję „szukania” przed wyjściem na ulicę, ponieważ szukanie pomaga zmaksymalizować oczekiwaną wydajność. Podejmowanie działań w celu zmodyfikowania przyszłych wyobrażeń - czasami nazywanych zbieraniem informacji - jest ważną częścią racjonalności i jest szczegółowo opisane później. Drugim przykładem gromadzenia informacji jest eksploracja, którą należy podjąć odkurzacz agent w początkowo nieznanym środowisku. Nasza definicja wymaga racjonalnego agenta nie tylko do zbierania informacji, ale także do uczenia się jak najwięcej z tego, co postrzega. Początkowa konfiguracja agenta może odzwierciedlać pewną wcześniejszą wiedzę o środowisku, ale w miarę zdobywania doświadczenia może być modyfikowana i rozszerzana. Istnieją skrajne przypadki, w których środowisko jest całkowicie znane a priori. W takich przypadkach agent nie musi postrzegać ani się uczyć; po prostu działa poprawnie. Oczywiście takie środki są delikatne. Zastanów się nad pewnym chrząszczem. Po wykopaniu gniazda i złożeniu jaja, pobiera kulkę gnoju z pobliskiej hałdy, aby zatkać wejście. Jeśli kula gnoju zostanie usunięta z uchwytu po drodze, chrząszcz kontynuuje swoje zadanie, a pantomimy zatkają gniazdo nieistniejącą kulą gnoju, nigdy nie zauważając, że jej brakuje. Ewolucja wbudowała założenie w zachowanie chrząszcza, a gdy zostaje naruszone, nieskuteczne zachowanie. Nieco bardziej inteligentna jest osa Sphex. Samica spex wykopie norę, wyjdzie i użądli gąsienicę i przeciągnie ją do nory, ponownie wejdzie do nory, aby sprawdzić, czy wszystko jest w porządku, przeciągnie gąsienicę do środka i złoży jaja. Gąsienica służy jako źródło pożywienia podczas wykluwania się jaj. Jak dotąd tak dobrze, ale jeśli entomolog przesunie gąsienicę kilka cali dalej, gdy sphex sprawdza, powróci do etapu „przeciągania” swojego planu i będzie kontynuował plan bez modyfikacji, nawet po kilkudziesięciu ruchomych interwencji gąsienicy. Sphex nie jest w stanie dowiedzieć się, że jego wrodzony plan zawodzi i dlatego go nie zmieni. W zakresie, w jakim agent opiera się na wcześniejszej wiedzy projektanta, a nie na własnych poglądach, mówimy, że agentowi brakuje autonomii. Racjonalny agent powinien być autonomiczny - powinien nauczyć się, co może zrekompensować częściową lub niepoprawną wcześniejszą wiedzę. Na przykład odkurzacz, który uczy się przewidywać, gdzie i kiedy pojawi się dodatkowy brud, będzie działał lepiej niż ten, który tego nie robi. W praktyce rzadko wymaga się pełnej autonomii od samego początku: gdy agent miał niewielkie doświadczenie lub nie miał go wcale, musiałby działać losowo, chyba że projektant udzielił pomocy. Tak więc, podobnie jak ewolucja zapewnia zwierzętom wystarczająco dużo wbudowanych odruchów, aby przetrwać wystarczająco długo, aby uczyć się same, rozsądne byłoby zapewnienie sztucznemu, inteligentnemu agentowi początkowej wiedzy i umiejętności uczenia się. Po wystarczającym doświadczeniu w środowisku zachowanie racjonalnego agenta może stać się skutecznie niezależne od jego wcześniejszej wiedzy. Dlatego włączenie uczenia się pozwala zaprojektować jednego racjonalnego agenta, który odniesie sukces w wielu różnych środowiskach. Teraz, gdy mamy definicję racjonalności, jesteśmy prawie gotowi myśleć o budowaniu racjonalnych agentów. Najpierw jednak musimy pomyśleć o środowiskach zadań, które są zasadniczo „problemami”, dla których racjonalni agenci są „rozwiązaniami”. Zaczynamy od pokazania, jak określić środowisko zadań, ilustrując ten proces wieloma przykładami. Następnie pokazujemy, że środowiska zadań mają różnorodne smaki. Smak środowiska zadań wpływa bezpośrednio na odpowiedni projekt programu agenta.

Określanie środowiska zadania

W naszej dyskusji na temat racjonalności prostego odkurzacza, musieliśmy określić miarę wydajności, środowisko oraz siłowniki i czujniki agenta. Grupujemy je wszystkie pod nagłówkiem środowiska zadań. Akronimicznie nazywamy PEAS (Performance, Environment, Actuators, Sensors). Projektując agenta, pierwszym krokiem zawsze musi być możliwie pełne określenie środowiska zadań. Świat próżni był prostym egzaminem; rozważmy bardziej złożony problem: zautomatyzowany taksówkarz. Należy zwrócić uwagę, zanim czytelnik zaalarmuje, że w pełni zautomatyzowana taksówka obecnie nieco przekracza możliwości istniejącej technologii. Pełne zadanie prowadzenia pojazdu jest wyjątkowo

otwarte. Nie ma ograniczeń co do nowych kombinacji okoliczności, które mogą się pojawić - to kolejny powód, dla którego wybraliśmy go jako przedmiot dyskusji. Poniżej mamy podsumowanie PEAS dla środowiska zadań taksówki. Omawiamy każdy element bardziej szczegółowo w poniższych akapitach.

Typ agenta: Pomiar Wydajności: Środowisko: Siłowniki: Czujniki

Taksówkarz: Bezpieczna, szybka, legalna, komfortowa podróż, maksymalizacja zysków: Drogi, inny ruch, piesi, klienci: Układ kierowniczy, przyspieszenie, hamulec, sygnał, klakson, wyświetlacz: Kamery, sonar, prędkościomierz, GPS, licznik kilometrów, akcelerometr, czujniki silnika , klawiatura

Po pierwsze, do jakiej miary wydajności zautomatyzowanego sterownika chcielibyśmy dążyć? Pożądane cechy obejmują dotarcie do właściwego miejsca docelowego; minimalizacja zużycia paliwa; minimalizacja czasu podróży lub kosztów; minimalizowanie naruszeń przepisów ruchu drogowego i zakłóceń dla innych kierowców; maksymalizacja bezpieczeństwa i komfortu pasażerów; maksymalizacja zysków. Oczywiście niektóre z tych celów są sprzeczne, więc konieczne będą kompromisy. Następnie, z jakim środowiskiem jazdy będzie miała do czynienia taksówka? Każdy taksówkarz musi radzić sobie z różnymi drogami, od wiejskich pasów i zaułków miejskich po 12-pasmowe autostrady.

Drogi zawierają inny ruch, pieszych, bezpańskie zwierzęta, roboty drogowe, samochody policyjne, kałuże i dziury. Taksówka musi także wchodzić w interakcje z potencjalnymi i rzeczywistymi pasażerami. Istnieje również kilka opcjonalnych opcji. Taksówka może potrzebować iść do pracy w Południowej Kalifornii, gdzie śnieg rzadko stanowi problem, lub na Alasce, gdzie rzadko go nie ma. Zawsze może jechać po prawej stronie lub możemy chcieć, aby był wystarczająco elastyczny, aby jechać po lewej stronie w Wielkiej Brytanii lub Japonii. Oczywiście, im bardziej ograniczone środowisko, tym łatwiejszy problem projektowy. Do siłowników zautomatyzowanej taksówki należą te, które są dostępne dla ludzkiego kierowcy: kontrola silnika za pomocą akceleratora oraz kontrola sterowania i hamowania. Ponadto będzie wymagał wyjścia na ekran wyświetlacza lub syntezatora głosu, aby rozmawiać z pasażerami, i być może jakiś sposób komunikowania się z innymi pojazdami, grzecznie lub w inny sposób. Podstawowe czujniki dla taksówki będą zawierać jedną lub więcej sterowanych kamer wideo, dzięki którym będzie widzieć drogę; może je rozszerzyć o czujniki podczerwieni lub sonary w celu wykrywania odległości od innych samochodów i przeszkód. Aby uniknąć mandatów za przekroczenie prędkości, taksówka powinna mieć prędkościomierz, a aby właściwie kontrolować pojazd, zwłaszcza na zakrętach, powinien mieć przyspieszeniomierz. Aby określić stan mechaniczny pojazdu, potrzebny będzie zwykły układ czujników silnika, paliwa i układu elektrycznego. Podobnie jak wielu ludzkich kierowców, może potrzebować globalnego systemu pozycjonowania (GPS), aby się nie zgubić. Wreszcie, pasażer będzie potrzebował klawiatury lub mikrofonu, aby poprosić o miejsce docelowe. Poniżej mamy podstawowe elementy PEAS dla szeregu dodatkowych typów agentów.

Typ agenta: Wydajność Pomiar: Środowisko: Siłowniki: Czujniki

System diagnostyki medycznej: Zdrowy pacjent, obniżone koszty: Pacjent, szpital, personel: Wyświetlanie pytań, testów, diagnoz, zabiegów, skierowań: Wprowadzanie objawów za pomocą klawiatury, wyniki, odpowiedzi pacjenta

System analizy obrazu satelitarnego: Prawidłowa kategoryzacja obrazu: Łącze w dół z satelity na orbicie: Wyświetlanie kategoryzacji sceny: Tablice kolorowych pikseli

Robot kompletujący części: Procent części we właściwych pojemnikach: Przenośnik taśmowy z częściami; pojemniki: Przegubowe ramię i ręka: Kamera, czujniki kąta przegubu

Sterownik rafinerii: czystość, wydajność, bezpieczeństwo: rafineria, operatorzy: zawory, pompy, grzejniki, wyświetlacze: czujniki temperatury, ciśnienia, chemiczne

Interaktywny korepetytor języka angielskiego: Wynik ucznia z testu: Zestaw uczniów, agencja testująca: Wyświetlanie ćwiczeń, sugestie, poprawki: Wprowadzanie z klawiatury

Niektórym czytelnikom może zaskoczyć fakt, że nasza lista typów agentów obejmuje niektóre programy działające w całkowicie sztucznym środowisku zdefiniowanym przez wprowadzanie z klawiatury i wyświetlanie znaków na ekranie. „Z pewnością", można powiedzieć, „to nie jest prawdziwe środowisko, prawda?" W rzeczywistości liczy się nie rozróżnienie między „realnym" i „sztucznym" środowiskiem, ale złożoność relacji między jego zachowaniem agenta, sekwencja percepcji generowana przez środowisko i miara wydajności. Niektóre „prawdziwe" środowiska są w rzeczywistości dość proste. Na przykład zaprojektowany robot w celu kontroli części, które pojawiają się na taśmie przenośnika, możeskorzystać z wielu uproszczonych założeń: że oświetlenie jest zawsze takie, że jedyną rzeczą na taśmie przenośnika będą części, o których wie, i że możliwe są tylko dwie akcje (zaakceptowanie lub odrzucenie). W przeciwieństwie do tego, niektórzy agenci oprogramowania (lub roboty programowe lub softboty) istnieją w bogatych, nieograniczonych domenach. Wyobraź sobie operatora strony internetowej typu softbot, który skanuje internetowe źródła wiadomości i pokazuje użytkownikom interesujące przedmioty, a jednocześnie sprzedaje powierzchnię reklamową w celu generowania przychodów. Aby dobrze się spisać, operator będzie potrzebował pewnych umiejętności przetwarzania języka naturalnego, będzie musiał dowiedzieć się, czym interesuje się każdy użytkownik i reklamodawca, i będzie musiał dynamicznie zmieniać swoje plany - na przykład, gdy połączenie z jednym źródłem wiadomości przestaje działać lub gdy pojawia się nowy. Internet to środowisko, którego złożoność dorównuje złożoności świata fizycznego, a jego mieszkańcami jest wielu sztucznych i ludzkich agentów.

Właściwości środowisk zadaniowych

Zakres środowisk zadań, które mogą pojawić się w sztucznej inteligencji, jest oczywiście ogromny. Możemy jednak zidentyfikować dość małą liczbę wymiarów, według których można podzielić środowiska zadań. Wymiary te determinują w dużym stopniu odpowiedni projekt agenta i możliwość zastosowania każdej z głównych rodzin technik implementacji agenta. Najpierw podajemy wymiary, a następnie analizujemy kilka środowisk zadań, aby zilustrować pomysły. Podane tutaj definicje są nieformalne; późniejsze rozdziały zawierają bardziej precyzyjne stwierdzenia i przykłady każdego rodzaju środowiska.

W pełni obserwowalne vs. częściowo obserwowalne: jeśli czujniki agenta dają mu dostęp do pełnego stanu środowiska w każdym momencie, to mówimy, że środowisko zadania jest w pełni obserwowalne. Środowisko zadania jest w pełni w pełni obserwowalne, jeśli czujniki wykrywają wszystkie aspekty, które są istotne dla wyboru działania; trafność z kolei zależy od miernika wyników. W pełni obserwowalne środowiska są wygodne, ponieważ agent nie potrzebuje utrzymywać żadnego stanu wewnętrznego, aby śledzić świat. Środowisko może być częściowo obserwowalne z powodu hałaśliwych i niedokładnych czujników lub z powodu braku części stanu w danych czujnika - na przykład odczynnik próżniowy z tylko lokalnym czujnikiem zabrudzenia nie może stwierdzić, czy na innych kwadratach jest brud, a zautomatyzowana taksówka nie widzi, co myślą inni kierowcy. Jeśli agent nie ma żadnych czujników, środowisko jest nieobserwowalne. Można by pomyśleć, że w takich przypadkach sytuacja agenta jest beznadziejna, cele agenta mogą być nadal osiągalne, czasami z pewnością.

Jeden agent a wielu agentów: rozróżnienie między środowiskami jedno- i wieloagentowymi może wydawać się dość proste. Na przykład agent rozwiązujący samodzielnie krzyżówkę jest wyraźnie w środowisku jednego agenta, podczas gdy agent grający w szachy jest w środowisku dwóch agentów. Są jednak pewne subtelne kwestie. Najpierw opisaliśmy, jak podmiot może być postrzegany jako agent, ale nie wyjaśniliśmy, które podmioty należy postrzegać jako agentów. Czy agent A (na przykład taksówkarz) musi traktować obiekt B (inny wehikuł) jako czynnik, czy też można go traktować jedynie jako przedmiot zachowujący się zgodnie z prawami fizyki, analogicznie do fal na plaży czy liści wiejących na wietrze? Kluczową różnicą jest to, czy zachowanie B najlepiej opisać jako maksymalizację miary wydajności, której wartość zależy od zachowania agenta A. Na przykład w szachach przeciwnik B stara się zmaksymalizować swoją miarę wydajności, co zgodnie z regułami szachów minimalizuje wydajność agenta A.

Zatem szachy są konkurencyjnym środowiskiem wielu agentów. Z drugiej strony, w środowisku taksówkarza unikanie kolizji maksymalizuje miarę wydajności wszystkich agentów, więc jest to częściowo współpracujące środowisko wieloagentowe. Jest również częściowo konkurencyjny bo np. tylko jeden samochód może zająć miejsce parkingowe. Problemy związane z projektowaniem agentów w środowiskach wieloagentowych są często zupełnie inne niż w środowiskach z jednym agentem; na przykład komunikacja często pojawia się jako racjonalne zachowanie w wielu środowiskach agentów; w niektórych konkurencyjnych środowiskach losowe zachowanie jest racjonalne, ponieważ pozwala uniknąć pułapek przewidywalności.

Deterministyczne a stochastyczne. Jeśli następny stan środowiska jest całkowicie zdeterminowany przez aktualny stan i akcję wykonywaną przez agenta, to mówimy, że środowisko jest deterministyczne; w przeciwnym razie jest stochastyczna. W zasadzie podmiot nie musi martwić się niepewnością w całkowicie obserwowalnym, deterministycznym środowisku. (W naszej definicji ignorujemy niepewność, która wynika wyłącznie z działań innych agentów w środowisku wieloagentowym; w ten sposób gra może być deterministyczna, nawet jeśli każdy agent może nie być w stanie przewidzieć działań innych). Jeśli środowisko jest częściowo obserwowalne, wówczas mogłoby się wydawać, że jest stochastyczne. Większość rzeczywistych sytuacji jest tak złożona, że niemożliwe jest śledzenie wszystkich nieobserwowanych aspektów; ze względów praktycznych należy je traktować jako stochastyczne. W tym sensie jazda taksówką jest wyraźnie stochastyczna, ponieważ nigdy nie można dokładnie przewidzieć zachowania ruchu; ponadto, opony pękają i silnik gaśnie bez ostrzeżenia. Świat próżni, tak jak go opisaliśmy, jest deterministyczny, ale wariacje mogą obejmować elementy stochastyczne, takie jak przypadkowo pojawiający się brud i zawodny mechanizm ssania. Mówimy, że środowisko jest niepewne, jeśli nie jest w pełni obserwowalne lub nie jest deterministyczne. Ostatnia uwaga: użycie przez nas słowa „stochastyczny" generalnie oznacza, że niepewność co do wyników jest określana ilościowo w kategoriach prawdopodobieństwa; środowisko niedeterministyczne to takie, w którym działania charakteryzują się możliwymi skutkami, ale nie są do nich przypisane żadne prawdopodobieństwa. Niedeterministyczne opisy środowiska są zwykle kojarzone z miernikami wydajności, które wymagają od agenta sukcesu dla wszystkich możliwych wyników jego działań.

Epizodyczne a sekwencyjne: w środowisku zadań epizodycznych doświadczenie agenta jest podzielone na niepodzielne epizody. W każdym odcinku agent otrzymuje spostrzeżenie, a następnie wykonuje jedną akcję. Co najważniejsze, kolejny odcinek nie zależy od działań podjętych w poprzednich odcinkach. Wiele zadań klasyfikacyjnych ma charakter epizodyczny. Na przykład agent, który musi wykryć wadliwe części na linii montażowej, opiera każdą decyzję na bieżącej części, niezależnie od wcześniejszych decyzji; ponadto obecna decyzja nie ma wpływu na to, czy następna część jest wadliwa. Z drugiej strony w środowiskach sekwencyjnych bieżąca decyzja może wpłynąć na wszystkie przyszłe

decyzje. Gra w szachy i jazda taksówką jest sekwencyjna: w obu przypadkach krótkoterminowe działania mogą mieć długofalowe konsekwencje. Środowiska epizodyczne są znacznie prostsze niż środowiska sekwencyjne, ponieważ agent nie musi myśleć z wyprzedzeniem.

Statyczne a dynamiczne: jeśli środowisko może się zmienić, gdy agent rozważa, wówczas mówimy, że jest ono dynamiczne dla tego agenta; w przeciwnym razie jest statyczny. Ze środowiskami statycznymi łatwo sobie poradzić, ponieważ agent nie musi patrzeć na świat podczas podejmowania decyzji ani martwić się upływem czasu. Z drugiej strony dynamiczne środowiska nieustannie pytają agenta, co chce zrobić; jeśli jeszcze nie zdecydował, liczy się to jako decyzja o braku działań. Jeśli samo środowisko nie zmienia się wraz z upływem czasu, ale zmienia się ocena wydajności agenta, wówczas mówimy, że środowisko jest półdynamiczne. Jazda taksówką jest wyraźnie dynamiczna: inne samochody i sama taksówka poruszają się, podczas gdy algorytm jazdy waha się, co robić dalej. Gra w szachy z zegarem jest półdynamiczna. Krzyżówki są statyczne.

Dyskretne a ciągłe: Rozróżnienie dyskretne / ciągłe dotyczy stanu środowiska, sposobu traktowania czasu oraz percepcji i działań agenta. Na przykład środowisko szachowe ma skończoną liczbę odrębnych stanów (z wyłączeniem zegara). Szachy mają również dyskretny zestaw spostrzeżeń i działań. Jazda taksówką jest problemem ciągłym i ciągłym: prędkość i lokalizacja taksówki oraz innych pojazdów zmieniają się w szereg ciągłych wartości i robią to płynnie w czasie. Czynności podczas kołowania są również ciągłe (kąty skrętu itp.). Sygnał wejściowy z kamer cyfrowych jest, ściśle mówiąc, dyskretny, ale zazwyczaj jest traktowany jako reprezentujący stale zmieniającą się intensywność i lokalizację.

Znane a nieznane: Ściśle mówiąc, rozróżnienie to nie odnosi się do samego środowiska, ale do stanu wiedzy agenta (lub projektanta) na temat „praw fizyki" środowiska. W znanym środowisku wyniki (lub prawdopodobieństwa wyniku, jeśli środowisko jest stochastyczne) dla wszystkich działań są podane. Oczywiście, jeśli środowisko jest nieznane, agent będzie musiał nauczyć się, jak działa, aby podejmować dobre decyzje. Należy zauważyć, że rozróżnienie między znanymi i nieznanymi środowiskami nie jest tym samym, co różnica między w pełni i częściowo obserwowalnymi środowiskami. Jest całkiem możliwe, że znane środowisko będzie częściowo obserwowalne - na przykład w grach karcianych w pasjansie znam zasady, ale nadal nie widzę kart, które nie zostały jeszcze odwrócone. I odwrotnie, nieznane środowisko może być w pełni widoczne - w nowej grze wideo ekran może pokazywać cały stan gry, ale nadal nie wiem, co robią przyciski, dopóki ich nie wypróbuję. Jak można się spodziewać, najtrudniejszy przypadek jest częściowo obserwowalny, wieloagentowy, stochastyczny, sekwencyjny, dynamiczny, ciągły i nieznany. Jazda taksówką jest trudna pod każdym względem, z wyjątkiem tego, że w większości znane jest środowisko kierowcy. Prowadzenie wypożyczonego samochodu w nowym kraju o nieznanym położeniu geograficznym i przepisach drogowych jest o wiele bardziej ekscytujące. Poniżej przedstawiamy właściwości wielu znanych środowisk.

Środowisko zadania: Obserwowalne: Czynniki: Deterministyczne: Epizodyczne: Statyczne: Dyskretne

Krzyżówka: Całkowicie: Pojedyncza: Deterministyczna: Sekwencyjna: Statyczna Dyskretna

Szachy z zegarem: Całkowicie: Multi: Deterministyczne: Sekwencyjne: Pół: Dyskretne

Poker: Częściowo: Multi: Stochastyczny: Sekwencyjny: Statyczny: Dyskretny

Backgammon: W pełni: Multi: Stochastyczny: Sekwencyjny: Statyczny: Dyskretny

Jazda taksówką: Częściowo: Multi: Stochastyczna: Sekwencyjna: Dynamiczna: Ciągła

Diagnoza lekarska: Częściowo: Pojedyncza: Stochastyczna: Sekwencyjna: Dynamiczna: Ciągła

Analiza obrazu: Całkowicie: Pojedynczy: Deterministyczny: Epizodyczny: Pół: Ciągła

Robot do pobierania części: Częściowo: Pojedynczy: Stochastyczny: Epizodyczny: Dynamiczny: Ciągły

Sterownik rafinerii: Częściowo: Pojedynczy: Stochastyczny: Sekwencyjny: Dynamiczny: Ciągła

Interaktywny korepetytor języka angielskiego: Częściowo: Multi: Stochastyczny: Sekwencyjny: Dynamiczny: Dyskretny

Zwróć uwagę, że odpowiedzi nie zawsze są wycinane i suszone. Na przykład opisujemy robota pobierającego części jako epizodyczny, ponieważ zwykle rozpatruje każdą część osobno. Ale jeśli pewnego dnia pojawi się duża partia wadliwych części, robot powinien wyciągnąć wnioski z kilku obserwacji, że rozkład defektów uległ zmianie i powinien zmodyfikować swoje zachowanie dla kolejnych części.

Nie uwzględniliśmy kolumny „znane / nieznane", ponieważ, jak wyjaśniono wcześniej, nie jest to wyłącznie właściwość środowiska. W niektórych środowiskach, takich jak szachy i poker, dość łatwo jest przekazać agentowi pełną wiedzę o zasadach, niemniej jednak interesujące jest rozważenie, w jaki sposób agent mógłby nauczyć się grać w te gry bez takiej wiedzy. Kilka odpowiedzi w tabeli zależy od sposobu zdefiniowania środowiska zadań. Wymieniliśmy zadanie diagnozy medycznej jako pojedynczy czynnik, ponieważ proces chorobowy u pacjenta nie jest modelowany w sposób opłacalny jako czynnik; ale system diagnostyki medycznej może również mieć do czynienia z opornymi pacjentami i sceptycznym personelem, więc środowisko może mieć aspekt wieloagentowy. Co więcej, diagnoza medyczna ma charakter epizodyczny, jeśli postrzega się zadanie jako wybór diagnozy na podstawie listy objawów; problem jest sekwencyjny, jeśli zadanie może obejmować zaproponowanie serii testów, ocenę postępów w trakcie leczenia i tak dalej. Ponadto wiele środowisk jest epizodycznych na wyższych poziomach niż indywidualne działania agenta. Na przykład turniej szachowy składa się z sekwencji gier; każda gra jest odcinkiem, ponieważ (ogólnie rzecz biorąc) ruchy z poprzedniej gry nie wpływają na wkład ruchów w jednej grze w ogólną wydajność agenta. Z drugiej strony, podejmowanie decyzji w ramach jednej gry jest z pewnością sekwencyjne. Repozytorium kodu związane z tą książką (aima.cs.berkeley.edu) zawiera implementacje wielu środowisk, wraz z symulatorem środowiska ogólnego przeznaczenia który umieszcza jednego lub więcej agentów w symulowanym środowisku, obserwuje ich zachowanie w czasie i ocenia ich zgodnie z podaną miarą wydajności. Takie eksperymenty są często przeprowadzane nie dla jednego środowiska, ale dla wielu środowisk zaczerpniętych z klasy środowiska. Na przykład, aby ocenić taksówkarza w symulowanym ruchu drogowym, chcielibyśmy przeprowadzić wiele symulacji z różnymi warunkami ruchu, oświetleniem i pogodą. Gdybyśmy zaprojektowali agenta dla pojedynczego scenariusza, moglibyśmy być w stanie wykorzystać określone właściwości konkretnego przypadku, ale moglibyśmy nie zidentyfikować dobrego projektu do jazdy w ogóle. Z tego powodu repozytorium kodu zawiera również generator środowiska dla każdej klasy środowiska, który wybiera określone środowiska (z pewnym prawdopodobieństwem), w których ma być uruchomiony agent. Na przykład generator środowiska próżniowego inicjuje losowo wzór zabrudzenia i lokalizację środka. Następnie interesuje nas średnia wydajność agenta w klasie środowiska. Racjonalny agent dla danej klasy środowiska maksymalizuje tę średnią wydajność.

STRUKTURA AGENTA

Jak dotąd mówiliśmy o agentach, opisując zachowanie - działanie, które jest wykonywane po dowolnej sekwencji spostrzeżeń. Teraz musimy ugryźć to i porozmawiać o tym, jak działa wnętrze. Zadaniem sztucznej inteligencji jest zaprojektowanie programu agenta, który implementuje funkcję agenta -

odwzorowanie percepcji na działania. Zakładamy, że ten program będzie działał na jakimś urządzeniu komputerowym z fizycznymi czujnikami i siłownikami - nazywamy to architekturą:

agent = architektura + program.

Oczywiście wybrany przez nas program musi być odpowiedni dla danej architektury. Jeśli program ma rekomendować takie akcje jak Walk, to architektura powinna mieć nogi. Architektura może być zwykłym komputerem PC lub automatycznym samochodem z kilkoma komputerami pokładowymi, kamerami i innymi czujnikami. Ogólnie rzecz biorąc, architektura udostępnia programowi spostrzeżenia z czujników, uruchamia program i przekazuje wybrane działania programu do elementów wykonawczych w miarę ich generowania. Większość tego tekstu dotyczy projektowania programów agentowych

Programy agentów

Wszystkie programy agentów, które projektujemy, mają ten sam szkielet: przyjmują bieżącą percepcję jako dane wejściowe z czujników i zwracają działanie do siłowników. Zwróć uwagę na różnicę między programem agenta, który przyjmuje bieżącą percepcję jako dane wejściowe, a funkcją agenta, która zajmuje całą historię percepcji. Program agenta przyjmuje tylko bieżące postrzeganie jako dane wejściowe, ponieważ nic więcej nie jest dostępne ze środowiska; jeśli działania agenta muszą zależeć od całej sekwencji percepcji, agent będzie musiał zapamiętać percepcje. Programy agenta opisujemy w prostym języku pseudokodowym. (Repozytorium kodu online zawiera implementacje w prawdziwych językach programowania). Na przykład poniżej przedstawia dość trywialny program agent, który śledzi sekwencję percepcji, a następnie używa go do indeksowania w tabeli działań, aby zdecydować, co zrobić.

function TABLE-DRIVEN-AGENT(percept) returns an action

persistent: percepts, a sequence, initially empty

table, a table of actions, indexed by percept sequences, initially fully specified

append percept to the end of percepts

action ←LOOKUP(percepts, table)

return action

Tabela - której przykład podano wcześniej dla świata próżni - wyraźnie przedstawia funkcję agenta, którą uosabia program agenta. Aby w ten sposób zbudować racjonalnego agenta, jako projektanci musimy skonstruować tabelę zawierającą odpowiednie działanie dla każdej możliwej sekwencji percepcji. Warto zastanowić się, dlaczego tabelaryczne podejście do konstrukcji agentów jest skazane na niepowodzenie. Niech P będzie zbiorem możliwych percepcji i niech T będzie czasem życia agenta (całkowita liczba percepcji, które otrzyma). Tabela przeglądowa będzie zawierać

$$\sum_{t=1}^{T} |\mathcal{P}|^t$$

wpisy. Weźmy pod uwagę automatyczną taksówkę: sygnał wizualny z pojedynczej kamery dociera z prędkością około 27 megabajtów na sekundę (30 klatek na sekundę, 640 × 480 pikseli z 24 bitami informacji o kolorze). Daje to tabelę przeglądową z ponad 10^{250 000 000 000} wpisów dla domeny na godzinę jazdy. Nawet tabela wyszukiwania szachów - maleńki, dobrze wychowany

fragment świata rzeczywistego - zawierałaby co najmniej 10¹⁵⁰ wpisów. Zniechęcający rozmiar tych tabel (liczba atomów w obserwowalnym wszechświecie jest mniejsza niż 1080) oznacza, że (a) żaden czynnik fizyczny w tym wszechświat będzie miał miejsce na przechowywanie tabeli, (b) projektant nie miałby czasu na utworzenie tabeli, (c) żaden agent nigdy nie nauczyłby się wszystkich prawidłowych wpisów w tabeli z własnego doświadczenia, oraz (d) nawet jeśli środowisko jest projektant jest na tyle prosty, aby uzyskać możliwy rozmiar tabeli brak wskazówek dotyczących wypełniania wpisów w tabeli. Mimo wszystko robi to, co chcemy: realizuje żądaną funkcję agenta. Kluczowym wyzwaniem dla sztucznej inteligencji jest nauczenie się, jak pisać programy, które w miarę możliwości generują racjonalne zachowanie z małego programu, a nie z ogromnej tabeli. Mamy wiele przykładów pokazujących, że można to z powodzeniem zrobić w innych obszarach: na przykład ogromne tabele pierwiastków kwadratowych używane przez inżynierów i dzieci w wieku szkolnym przed latami 70. XX wieku zostały obecnie zastąpione pięcioliniowym programem metody Newtona działającym na kalkulatorach elektronicznych . Pytanie brzmi, czy sztuczna inteligencja może zrobić dla ogólnego inteligentnego zachowania to samo, co Newton zrobił dla pierwiastków kwadratowych? Wierzymy, że odpowiedź brzmi tak. W pozostałej części tej sekcji zarysujemy cztery podstawowe rodzaje programów agentowych, które ucieleśniają zasady leżące u podstaw prawie wszystkich inteligentnych systemów:

* Proste środki odruchowe;

* środki odruchowe oparte na modelach;

* agenci nastawieni na cele; i

* Agenci narzędziowi.

Każdy rodzaj programu agenta łączy poszczególne komponenty w określony sposób do generowania akcji. Sekcja późniejsza wyjaśnia ogólnie, jak przekształcić wszystkich tych agentów w agentów uczących się, które mogą poprawić wydajność ich komponentów, aby generować lepsze działania. Wreszcie, sekcja kolejna opisuje różne sposoby, w jakie same komponenty mogą być reprezentowane w agencie. Ta różnorodność zapewnia główną zasadę organizacyjną dla dziedziny

Agenci odruchowi

Najprostszym rodzajem agenta jest agent odruchowy. Agenci wybierają działania na podstawie aktualnego postrzegania, ignorując pozostałą część historii percepcji. Na przykład odczynnik próżniowy jest prostym środkiem odruchowym, ponieważ jego decyzja opiera się tylko na aktualnej lokalizacji i na tym, czy jest to zabrudzenie. Program agenta dla tego agenta został przedstawiony wcześniej. Zauważ, że program odczynników próżniowych jest rzeczywiście bardzo mały w porównaniu z odpowiednią tabelą. Najbardziej oczywista redukcja wynika z zignorowania historii percepcji, która ogranicza liczbę możliwości z 4^T do zaledwie 4. Dalsza, niewielka redukcja wynika z faktu, że gdy obecny plac jest brudny, akcja nie zależy od lokalizacji. Proste zachowania odruchowe występują nawet w bardziej złożonych środowiskach. Wyobraź sobie siebie jako kierowcę zautomatyzowanej taksówki. Jeśli samochód z przodu hamuje i zapalają się światła stopu, należy to zauważyć i zainicjować hamowanie. Innymi słowy, pewne przetwarzanie odbywa się na wejściu wizualnym w celu ustalenia stanu, który nazywamy „Samochód z przodu hamuje”. Następnie wyzwala to nawiązane połączenie w programie agenta z akcją „zainicjuj hamowanie”. Takie połączenie nazywamy regułą warunek – akcja, zapisaną jako jeśli samochód jadący z przodu hamuje, zainicjuj hamowanie. Ludzie mają również wiele takich połączeń, z których część to reakcje wyuczone (jak w przypadku jazdy), a niektóre z nich to wrodzone odruchy (takie jak mruganie, gdy coś zbliża się do oka). Pokazujemy kilka różnych sposobów

uczenia się i realizacji takich połączeń. Program poniższy jest specyficzny dla jednego określonego środowiska próżniowego.

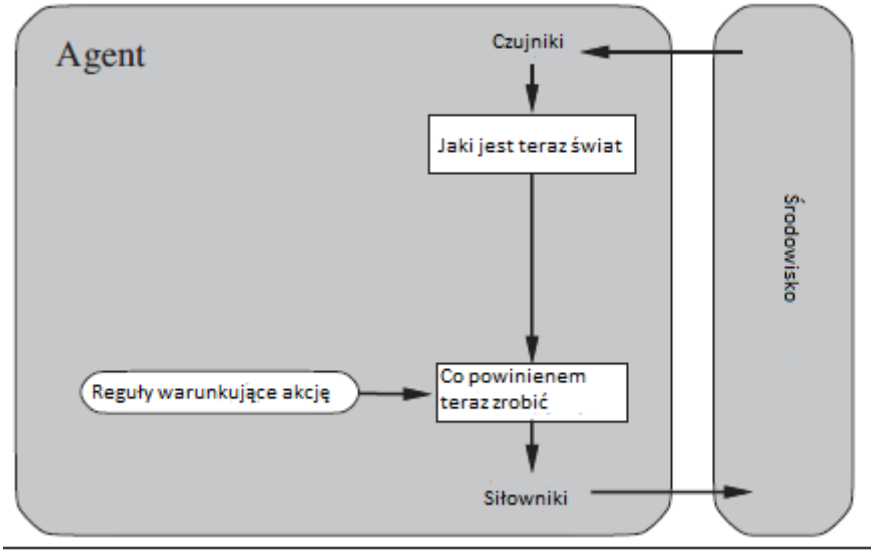
funkcja REFLEX-VACUUM-AGENT ([lokalizacja, stan]) zwraca akcję

jeśli status = Brudny, a następnie zwróć Ssij

w przeciwnym razie, jeśli location = A, to zwróć Right

w przeciwnym razie, jeśli location = B, to wróć Left

Bardziej ogólne i elastyczne podejście polega najpierw na zbudowaniu interpretera ogólnego przeznaczenia dla reguł warunkowych i akcji, a następnie na utworzeniu zestawów reguł dla określonych środowisk zadań. Rysunek przedstawia strukturę tego ogólnego programu w formie schematu, pokazując, w jaki sposób reguły warunek-akcja pozwalają agentowi na nawiązanie połączenia od percepcji do działania.



(Nie martw się, jeśli wydaje się to trywialne; wkrótce stanie się bardziej interesujące). Używamy prostokątów do oznaczania bieżącego stanu wewnętrznego procesu decyzyjnego agenta i owali do reprezentowania informacji tła używanych w tym procesie. Program agenta, który również jest bardzo prosty, pokazano tu:

funkcja SIMPLE-REFLEX-AGENT (percept) zwraca akcję

trwałe: reguły, zestaw reguł warunkowo-akcji

stan ← INTERPRET-INPUT (percept)

reguła ← RULE-MATCH (stan, reguły)

akcja ← reguła.AKCJA

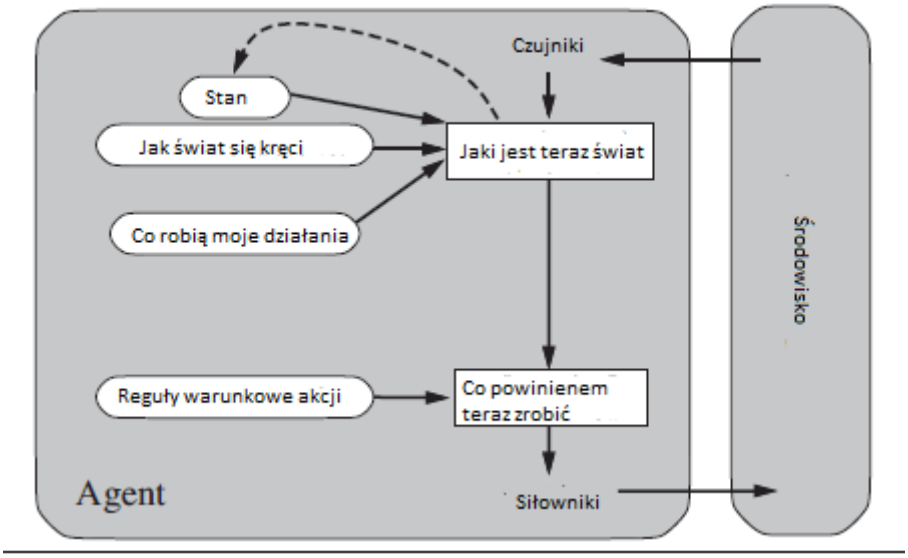
akcja powrotna

Funkcja generuje abstrakcyjny opis bieżącego stanu z percepcji, a funkcja zwraca pierwszą regułę ze zbioru reguł, która pasuje do podanego opisu stanu. Zwróć uwagę, że opis w kategoriach „reguły” i „dopasowanie” jest czysto koncepcyjny; rzeczywiste implementacje mogą być tak proste, jak zbiór bramek logicznych implementujących obwód boolowski. Proste środki odruchowe mają godną podziwu właściwość bycia prostymi, ale okazują się mieć ograniczoną inteligencję. Agent z tym

programem będzie działał tylko wtedy, gdy właściwą decyzję można podjąć na podstawie tylko aktualnego spostrzeżenia, czyli tylko wtedy, gdy środowisko jest w pełni obserwowalne. Nawet odrobina nieobserwowalności może spowodować poważne problemy. Na przykład podana wcześniej reguła hamowania zakłada, że stan hamującego samochodu z przodu można określić na podstawie aktualnego postrzegania - pojedynczej klatki wideo. Działa to, jeśli samochód z przodu ma centralnie zamontowane światło hamowania. Niestety starsze modele mają różne konfiguracje tylnych świateł, świateł hamowania i kierunkowskazów i nie zawsze można stwierdzić na jednym zdjęciu, czy samochód hamuje. Prosty odruchowy środek jadący za takim samochodem albo hamowałby nieprzerwanie i niepotrzebnie, albo, co gorsza, w ogóle nie hamował. Podobny problem widzimy w świecie próżni. Załóżmy, że prosty odruchowy odczynnik próżniowy jest pozbawiony czujnika położenia i ma tylko czujnik zabrudzenia. Taki agent ma tylko dwa możliwe percepcje: [brudny] i [czysty]. Może ssać w odpowiedzi na [Brudne]; co powinien zrobić w odpowiedzi na [Czyste]? Ruch w lewo kończy się niepowodzeniem (na zawsze), jeśli zdarzy się, że zaczyna się w kwadracie A, a ruch w prawo kończy się niepowodzeniem (na zawsze), jeśli zdarzy się, że zaczyna się w kwadracie B. Nieskończone pętle są często nieuniknione dla prostych agentów odruchowych działających w częściowo obserwowalnych środowiskach. jest to możliwe, jeśli agent może losować swoje działania. Na przykład, jeśli odkurzacz zauważy [Wyczyść], może rzucić monetą, aby wybrać między Lewym a Prawym. Łatwo jest pokazać, że agent dotrze na drugi plac średnio w dwóch krokach. Następnie, jeśli ten kwadrat jest brudny, agent wyczyści go i zadanie zostanie zakończone. W związku z tym zrandomizowany prosty środek odruchowy może przewyższać deterministyczny prosty środek odruchowy. Wspomnieliśmy wcześniej, że randomizowane zachowanie odpowiedniego rodzaju może być racjonalne w niektórych środowiskach wieloagentowych. W środowiskach z jednym agentem randomizacja zwykle nie jest racjonalna. Jest to przydatna sztuczka, która pomaga prostemu czynnikowi odruchowemu w niektórych sytuacjach, ale w większości przypadków możemy zrobić znacznie lepiej z bardziej wyrafinowanymi czynnikami deterministycznymi

Agent odruchowy oparty na modelu

Najskuteczniejszym sposobem radzenia sobie z częściową obserwowalnością jest śledzenie przez agenta części świata, której teraz nie widzi. Oznacza to, że agent powinien utrzymywać pewien rodzaj stanu wewnętrznego, który zależy od historii percepcji, a tym samym odzwierciedla przynajmniej część nieobserwowanych aspektów obecnego stanu. W przypadku problemu z hamowaniem stan wewnętrzny nie jest zbyt rozległy - tylko poprzednia klatka z kamery, dzięki czemu agent może wykryć, kiedy jednocześnie zapalają się lub gasną dwa czerwone światła na krawędzi pojazdu. W przypadku innych zadań związanych z prowadzeniem pojazdu, takich jak zmiana pasa, agent musi śledzić, gdzie znajdują się inne samochody, jeśli nie może zobaczyć ich wszystkich naraz. Aby jakakolwiek jazda była w ogóle możliwa, agent musi śledzić, gdzie znajdują się jego klucze. Aktualizowanie tych wewnętrznych informacji o stanie w miarę upływu czasu wymaga zakodowania w programie agenta dwóch rodzajów wiedzy. Po pierwsze, potrzebujemy pewnych informacji o tym, jak świat ewoluuje niezależnie od agenta - na przykład, że wyprzedzający samochód będzie generalnie bliżej niż przed chwilą. Po drugie, potrzebujemy pewnych informacji o tym, jak własne działania agenta wpływają na świat - na przykład, że gdy agent obraca kierownicę w prawo, samochód skręca w prawo lub że po pięciu minutach jazdy autostradą w kierunku północnym zwykle około pięciu kilometrów na północ od miejsca, w którym znajdowało się pięć minut temu. Ta wiedza o tym, „jak działa świat” - czy to zaimplementowana w prostych obwodach boolowskich, czy w pełnych teoriach naukowych - nazywana jest modelem świata. Agent korzystający z takiego modelu jest nazywany agentem opartym na modelu. Rysunek 2.11 przedstawia strukturę agenta odruchowego opartego na modelu ze stanem wewnętrznym, pokazując, w jaki sposób bieżąca percepcja jest łączona ze starym stanem wewnętrznym w celu wygenerowania zaktualizowanego opisu bieżącego stanu, w oparciu o model agenta dotyczący działania świata.



Program agenta pokazano tu

funkcja MODEL-BASED-REFLEX-AGENT (percept) zwraca akcję

trwały: stan, aktualna koncepcja stanu świata przez podmiot

model, opis tego, jak następny stan zależy od bieżącego stanu i akcji

reguły, zbiór reguł warunek-akcja

akcja, ostatnia akcja, początkowo brak

stan ←UPDATE-STATE (stan, akcja, percept, model)

reguła ← RULE-MATCH (stan, reguły)

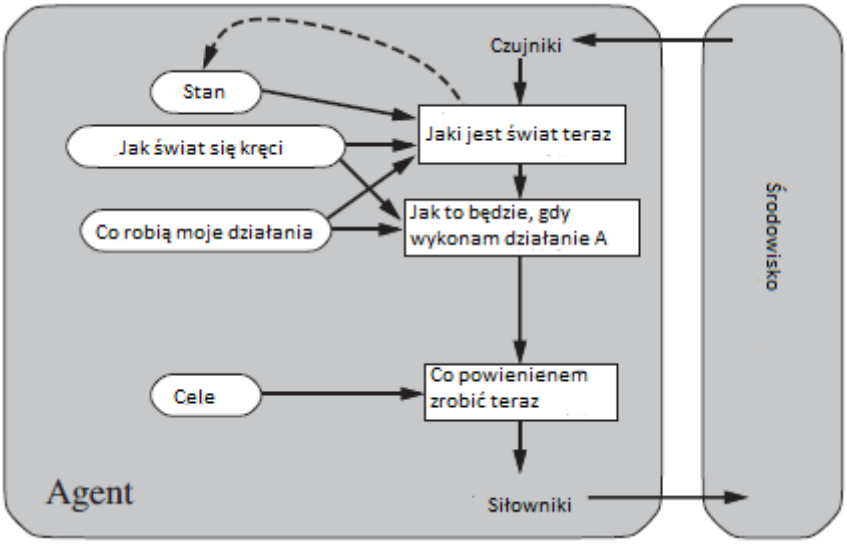
akcja ← reguła.AKCJA

akcja powrotna

Ciekawą częścią jest funkcja, która odpowiada za tworzenie nowego opisu stanu wewnętrznego. Szczegóły dotyczące modeli i stany są reprezentowane bardzo różnie w zależności od typu środowiska i konkretnej technologii użytej w projekcie agenta. Niezależnie od rodzaju użytej reprezentacji, agent rzadko jest w stanie dokładnie określić aktualny stan częściowo obserwowalnego środowiska. Zamiast tego, ramka z napisem „Jaki jest teraz świat” przedstawia „najlepsze przypuszczenie” agenta (lub czasami najlepsze domysły). Na przykład zautomatyzowana taksówka może nie być w stanie rozejrzeć się wokół dużej ciężarówki, która zatrzymała się przed nią i może tylko zgadywać, co może być przyczyną zatoru. Tym samym niepewność co do stanu obecnego może być nieunikniona, ale agent musi jeszcze podjąć decyzję. Być może mniej oczywistym punktem dotyczącym wewnętrznego „stanu” utrzymywanego przez podmiot oparty na modelach jest to, że nie musi on opisywać „jaki jest teraz świat” w sensie dosłownym. Na przykład taksówka może wracać do domu i może mieć regułę nakazującą jej zatankować gaz w drodze do domu, chyba że ma co najmniej pół zbiornika. Chociaż „jazda do domu” może wydawać się pewnym aspektem stanu świata, fakt przeznaczenia taksówki jest w rzeczywistości aspektem stanu wewnętrznego agenta. Jeśli uznasz to za zagadkowe, weź pod uwagę, że taksówka może znajdować się dokładnie w tym samym miejscu w tym samym czasie, ale zamierzając dotrzeć do innego miejsca docelowego.

Agenci nastawieni na cele

Znajomość aktualnego stanu środowiska nie zawsze wystarczy, aby zdecydować, co robić. Na przykład na skrzyżowaniu dróg taksówka może skręcić w lewo, w prawo lub jechać prosto. Prawidłowa decyzja zależy od tego, dokąd próbuje się dostać taksówka. Innymi słowy, oprócz aktualnego opisu stanu, agent potrzebuje pewnego rodzaju informacji o celu, które opisują pożądane sytuacje - na przykład przebywanie w miejscu docelowym pasażera. Program agenta może połączyć to z modelem (te same informacje, które zostały użyte w agencie odruchowym opartym na modelu), aby wybrać działania, które osiągną cel. Rysunek przedstawia strukturę agenta opartego na celach.

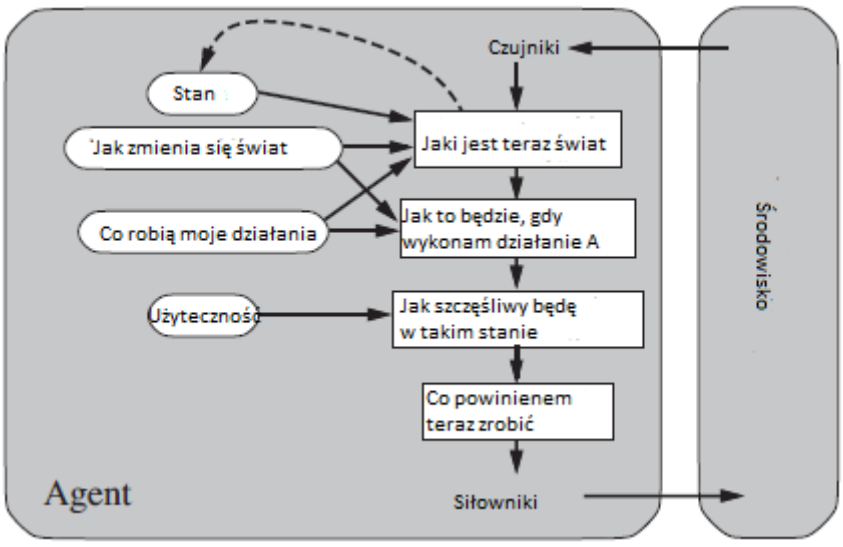


Czasami wybór działań na podstawie celu jest prosty - na przykład, gdy satysfakcja z celu wynika bezpośrednio z pojedynczego działania. Czasami będzie to trudniejsze - na przykład, gdy agent musi rozważyć długie sekwencje zwrotów akcji, aby znaleźć sposób na osiągnięcie celu. Wyszukiwanie (rozdziały od 3 do 5) i planowanie (rozdziały 10 i 11) to poddziedziny AI poświęcone znajdowaniu sekwencji działań, które osiągają cele agenta. Zauważ, że tego rodzaju podejmowanie decyzji zasadniczo różni się od warunku - reguł działania opisanych wcześniej, ponieważ obejmuje rozważenie przyszłości - zarówno „Co się stanie, jeśli zrobię to a to?” i „Czy to mnie uszczęśliwi?” W projektach agentów refleksyjnych ta informacja nie jest jawnie reprezentowana, ponieważ wbudowane reguły odwzorowują bezpośrednio z percepcji na działania. Środek odblaskowy hamuje, gdy widzi światła hamowania. Agent oparty na celach, zasadniczo może to spowodować, że jeśli samochód z przodu ma włączone światła hamowania, zwolni. Biorąc pod uwagę sposób, w jaki zwykle ewoluuje świat, jedyną czynnością, która pozwoli nie uderzyć w inne samochody, jest hamowanie. Chociaż agent oparty na celach wydaje się mniej wydajny, jest bardziej elastyczny, ponieważ wiedza, która wspiera jego decyzje, jest reprezentowana w sposób jawny i może być modyfikowana. Jeśli zacznie padać, agent może zaktualizować swoją wiedzę o skuteczności jego hamulców; spowoduje to automatyczną zmianę wszystkich odpowiednich zachowań w celu dostosowania do nowych warunków. Z drugiej strony, w przypadku agenta odruchowego musielibyśmy przepisać wiele reguł warunek-akcja. Zachowanie agenta opartego na celu można łatwo zmienić, aby przejść do innego miejsca docelowego, po prostu określając to miejsce docelowe jako cel. Reguły agenta refleksyjnego dotyczące tego, kiedy skręcić, a kiedy jechać prosto, będą działać tylko dla jednego miejsca docelowego; wszystkie muszą zostać wymienione, aby znaleźć się w nowym miejscu.

Agenci oparci na użyteczności

Same cele nie wystarczą do wygenerowania wysokiej jakości zachowań w większości środowisk. Na przykład wiele sekwencji akcji doprowadzi taksówkę do celu (a tym samym osiągnie cel), ale niektóre

są szybsze, bezpieczniejsze, bardziej niezawodne lub tańsze niż inne. Cele po prostu zapewniają surowe binarne rozróżnienie między stanami „szczęśliwymi" i „nieszczęśliwymi". Bardziej ogólna miara wydajności powinna pozwolić na porównanie różnych stanów świata w zależności od tego, jak bardzo uszczęśliwiliby agenta. Ponieważ „szczęśliwy" nie brzmi zbyt naukowo, ekonomiści i informatycy używają zamiast tego terminu użyteczność. Widzieliśmy już, że miara wydajności przypisuje punkty do dowolnej sekwencji stanów środowiska, dzięki czemu można łatwo odróżnić mniej i bardziej pożądane sposoby dotarcia do celu taksówki. Funkcja użyteczności agenta jest zasadniczo internalizacją miary wydajności. Jeśli wewnętrzna funkcja użyteczności i zewnętrzna miara wydajności są zgodne, wówczas agent, który wybiera działania mające na celu maksymalizację swojej użyteczności, będzie racjonalny zgodnie z zewnętrzną miarą wydajności. Podkreślmy jeszcze raz, że nie jest to jedyny sposób, aby być racjonalnym - widzieliśmy już program racjonalnego agenta dla świata próżni , który nie ma pojęcia, jaka jest jego funkcja użyteczności - ale, podobnie jak agenci celowi, agent oparty na użyteczności ma wiele zalet pod względem elastyczności i uczenia się. Co więcej, w dwóch rodzajach przypadków cele są nieodpowiednie, ale agent oparty na użyteczności może nadal podejmować racjonalne decyzje. Po pierwsze, gdy istnieją sprzeczne cele, z których tylko niektóre można osiągnąć (na przykład prędkość i bezpieczeństwo), funkcja użyteczności określa odpowiedni kompromis. Po drugie, gdy istnieje kilka celów, do których podmiot może dążyć, a żadnego z nich nie można osiągnąć z pewnością, użyteczność zapewnia sposób, w jaki prawdopodobieństwo sukcesu można wyważyć względem wagi celów. Częściowa obserwowalność i stochastyczność są wszechobecne w świecie rzeczywistym, a zatem podejmowanie decyzji odbywa się w warunkach niepewności. Mówiąc technicznie rzecz biorąc, racjonalny agent oparty na użyteczności wybiera działanie, które maksymalizuje oczekiwaną użyteczność wyników działania - to znaczy użyteczność, którą agent spodziewa się wyprowadzić, biorąc pod uwagę prawdopodobieństwa i użyteczność każdego wyniku. Pokaż, że każdy racjonalny agent musi zachowywać się tak, jakby posiadał funkcję użyteczności, której oczekiwaną wartość próbuje zmaksymalizować. Agent posiadający wyraźną funkcję użyteczności może podejmować racjonalne decyzje z algorytmem ogólnego przeznaczenia, który nie zależy od maksymalizacji określonej funkcji użyteczności. W ten sposób „globalna" definicja racjonalności - oznaczająca jako racjonalne te funkcje agentów, które mają najwyższą wydajność - zostaje przekształcona w „lokalne" ograniczenie dla projektów racjonalnych agentów, które można wyrazić w prostym programie. Struktura agentów opartych na narzędziach została przedstawiona na rysunku.

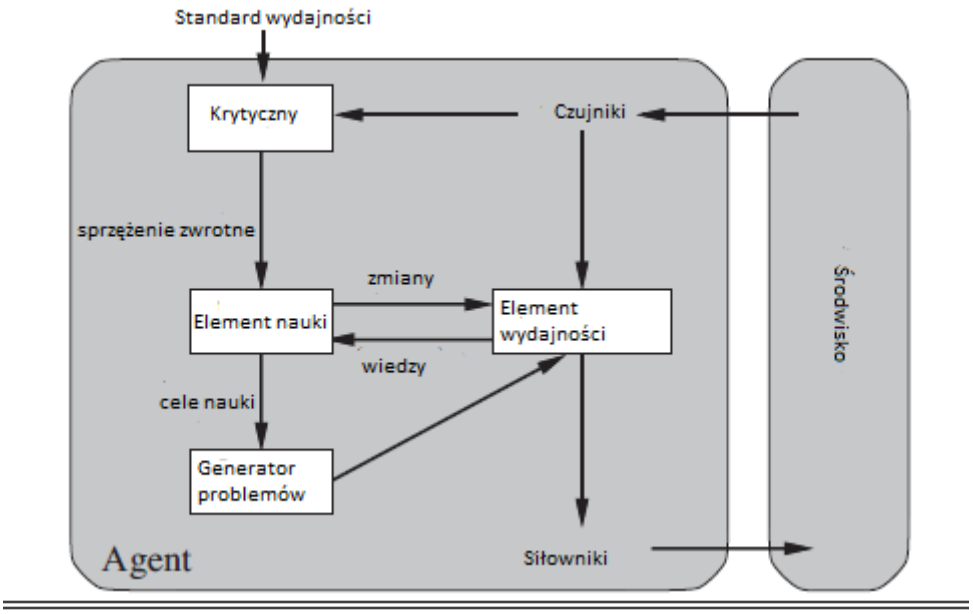


Programy agentowe oparte na użyteczności pojawiają się w części, w której zaprojektujemy agentów decyzyjnych, którzy muszą radzić sobie z niepewnością nieodłączną w środowiskach stochastycznych

lub częściowo obserwowalnych. W tym momencie czytelnik może się zastanawiać: „Czy to takie proste? Po prostu tworzymy agentów, które maksymalizują oczekiwaną użyteczność i gotowe? ” To prawda, że tacy agenci byliby inteligentni, ale nie jest to proste. Agenci oparci o użyteczność muszą modelować i śledzić swoje otoczenie, zadania, które wymagały wielu badań dotyczących percepcji, reprezentacji, rozumowania i uczenia się. Wyniki tych badań wypełnią nam wiele , wiele miejsca. Wybór sposobu działania maksymalizującego użyteczność jest również trudnym zadaniem, wymagającym pomysłowych algorytmów, które wypełniają kilka kolejnych rozdziałów. Nawet w przypadku tych algorytmów doskonała racjonalność jest zwykle nieosiągalna w praktyce ze względu na złożoność obliczeniową

Agenci uczący się

Opisaliśmy programy agentowe z różnymi metodami wyboru akcji. Jak dotąd nie wyjaśniliśmy, jak powstają programy agentowe. W swoim słynnym wczesnym artykule Turing (1950) rozważa pomysł ręcznego programowania swoich inteligentnych maszyn. Szacuje, ile pracy może to zająć, i konkluduje: „Wydaje się, że pożądana jest szybsza metoda”. Metoda, którą proponuje, to budowanie uczących się maszyn, a następnie ich nauczanie. W wielu obszarach sztucznej inteligencji jest to obecnie preferowana metoda tworzenia najnowocześniejszych systemów. Uczenie się ma jeszcze jedną zaletę, jak zauważyliśmy wcześniej: pozwala agentowi działać w początkowo nieznanych środowiskach i stać się bardziej kompetentnym, niż może na to pozwolić jego początkowa wiedza. W tej sekcji pokrótce przedstawimy główne idee agentów uczenia się. Generalnie komentujemy możliwości i metody uczenia się poszczególnych rodzajów agentów. Części kolejne bardziej szczegółowo omawiają same algorytmy uczenia się. Agenta uczącego się można podzielić na cztery koncepcyjne komponenty, jak pokazano na rysunku



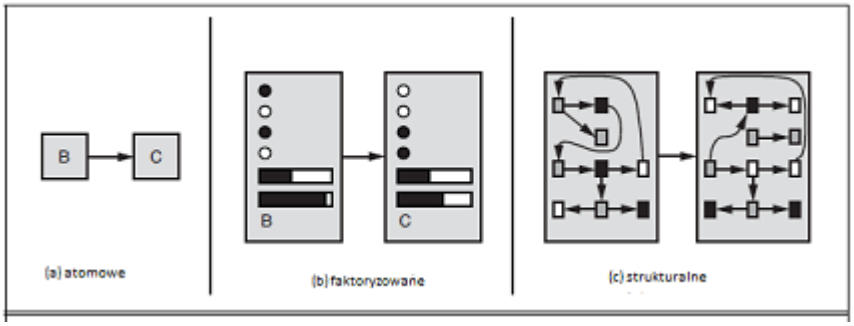
Najważniejsze rozróżnienie dotyczy elementu uczenia się, który jest odpowiedzialny za wprowadzanie ulepszeń, oraz elementu wydajności, który jest odpowiedzialny za wybór działań zewnętrznych. Element wykonawczy jest tym, co wcześniej uważaliśmy za całego agenta: przyjmuje percepcje i decyduje o działaniach. Element uczenia się wykorzystuje informacje zwrotne od krytyka na temat tego, jak sobie radzi agent i określa, w jaki sposób element wydajności powinien zostać zmodyfikowany, aby działał lepiej w przyszłości. Projekt elementu uczenia się zależy w dużym stopniu od projektu elementu wykonawczego. Kiedy próbujesz zaprojektować agenta, który uczy się pewnych umiejętności, pierwsze pytanie nie brzmi: „Jak mam go nauczyć, żeby się tego nauczył?” ale „Jakiego

rodzaju elementu wydajności będzie potrzebował mój agent, gdy nauczy się, jak to zrobić?" Mając projekt agenta, można skonstruować mechanizmy uczenia się, aby ulepszyć każdą część agenta. Krytyk mówi elementowi uczenia się, jak dobrze agent radzi sobie w odniesieniu do ustalonego standardu wydajności. Krytyk jest konieczny, ponieważ percepcje same w sobie nie wskazują na sukces agenta. Na przykład program szachowy może otrzymać spostrzeżenie wskazujące, że zamatował swojego przeciwnika, ale potrzebuje standardu wykonania, aby wiedzieć, że to dobra rzecz; sam percept tego nie mówi. Ważne jest ustalenie standardu wydajności. Koncepcyjnie należy myśleć o nim jako całkowicie poza agentem, ponieważ agent nie może modyfikować go, aby pasował do jego własnego zachowania. Ostatnim komponentem problemów agenta z uczeniem się jest generator problemów. Jest odpowiedzialny za sugerowanie działań, które doprowadzą do nowych i pouczających doświadczeń. Chodzi o to, że gdyby element wykonawczy miał swój sposób, robiłby to, co najlepsze, biorąc pod uwagę to, co wie. Ale jeśli agent chce trochę zbadać i wykonać być może nieoptymalne działania w krótkim okresie, może odkryć znacznie lepsze działania na dłuższą metę. Zadaniem generatora problemów jest sugerowanie tych działań eksploracyjnych. To właśnie robią naukowcy, przeprowadzając eksperymenty. Galileusz nie sądził, że zrzucanie kamieni ze szczytu wieży w Pizie jest samo w sobie cenne. Nie próbował rozbijać skał ani modyfikować mózgów nieszczęsnych przechodniów. Jego celem było zmodyfikowanie własnego mózgu poprzez zidentyfikowanie lepszej teorii ruchu obiektów. Aby ogólny projekt był bardziej konkretny, wróćmy do przykładu zautomatyzowanej taksówki. Element wydajności składa się z dowolnego zbioru wiedzy i procedur, którymi dysponuje taksówka, aby wybrać sposób prowadzenia pojazdu. Taksówka wyjeżdża na drogę i jedzie, korzystając z tego elementu wydajności. Krytyk obserwuje świat i przekazuje informacje elementowi uczenia się. Na przykład, gdy taksówka skręca szybko w lewo przez trzy pasy ruchu, krytyk zauważa szokujący język używany przez innych kierowców. Na podstawie tego doświadczenia element uczenia się jest w stanie sformułować regułę mówiącą, że było to złe działanie, a element wydajności jest modyfikowany przez instalację nowej reguły. Generator problemów może zidentyfikować pewne obszary zachowania wymagające poprawy i zasugerować eksperymenty, takie jak wypróbowanie hamulców na różnych nawierzchniach dróg w różnych warunkach. Element uczenia się może wprowadzać zmiany w dowolnym elemencie „wiedzy" przedstawionym na diagramach agenta. Najprostsze przypadki obejmują uczenie się bezpośrednio z sekwencji spostrzeżeń. Obserwacja par kolejnych stanów środowiska może pozwolić agentowi dowiedzieć się, „jak rozwija się świat", a obserwacja wyników jego działań może pozwolić agentowi dowiedzieć się, „co robią moje działania". Na przykład, jeśli taksówka wywiera określone ciśnienie hamowania podczas jazdy po mokrej drodze, wkrótce dowie się, jakie rzeczywiście osiągnięto opóźnienie. Oczywiście te dwa zadania uczenia się są trudniejsze, jeśli środowisko jest widoczne tylko częściowo. Formy uczenia się, o których mowa w poprzednim akapicie, nie wymagają dostępu do zewnętrznego standardu wykonania - w pewnym sensie standard jest uniwersalny, polegający na prognozowaniu zgodnym z eksperymentem. Sytuacja jest nieco bardziej złożona w przypadku agenta opartego na narzędziach, który chce poznać informacje o narzędziach. Na przykład załóżmy, że agent taksówkarz nie otrzymuje żadnych napiwków od pasażerów, którzy byli mocno wstrząśnięci podczas podróży. Zewnętrzna norma wydajności musi informować agenta, że utrata napiwków ma negatywny wpływ na ogólne wyniki; wtedy agent mógłby się dowiedzieć, że gwałtowne manewry nie przyczyniają się do jego własnej użyteczności. W pewnym sensie standard wydajności wyróżnia część przychodzącej percepcji jako nagrodę (lub karę), która zapewnia bezpośrednią informację zwrotną na temat jakości zachowania agenta. W ten sposób można zrozumieć ustalone na stałe standardy działania, takie jak ból i głód u zwierząt.. Podsumowując, agenci mają wiele różnych komponentów, które mogą być reprezentowane na wiele sposobów w programie agenta, więc wydaje się, że istnieje duża różnorodność metod uczenia się. Istnieje jednak jeden jednoczący temat. Uczenie się inteligentnych agentów można podsumować jako proces modyfikacji

każdego komponentu agenta w celu zbliżenia komponentów do dostępnych informacji zwrotnych, poprawiając w ten sposób ogólną wydajność agenta.

Jak działają składniki programów agentowych

Opisaliśmy programy agentowe (w kategoriach bardzo wysokiego poziomu) jako składające się z różnych komponentów, których funkcją jest odpowiadanie na pytania typu: „Jaki jest teraz świat?" „Co mam teraz zrobić?" „Co robią moje działania?" Kolejne pytanie dla studenta sztucznej inteligencji brzmi: „Jak u licha działają te komponenty?" Prawidłowa odpowiedź na to pytanie zajmuje około tysiąca stron, ale tutaj chcemy zwrócić uwagę czytelnika na kilka podstawowych różnic między różnymi sposobami, w jakie komponenty mogą przedstawiać środowisko, w którym przebywa agent. Z grubsza mówiąc, możemy umieścić reprezentacje wzdłuż osi rosnącej złożoności i mocy ekspresji - atomowej, podzielonej na czynniki i uporządkowanej. Aby zilustrować te pomysły, warto rozważyć konkretny komponent agenta, na przykład ten, który dotyczy „Co robią moje działania". Ten komponent opisuje zmiany, które mogą wystąpić w środowisku w wyniku podjęcia działania, a rysunek przedstawia schematyczny obraz tego, jak te przejścia mogą być reprezentowane.



W reprezentacji atomowej każdy stan świata jest niepodzielny - nie ma wewnętrznej struktury. Rozważmy problem znalezienia trasy dojazdu z jednego końca kraju na drugi przez jakąś sekwencję miast. Dla rozwiązania tego problemu może wystarczyć sprowadzenie stanu świata do nazwy miasta, w którym się znajdujemy - do jednego atomu wiedzy; „czarna skrzynka", której jedyną dostrzegalną właściwością jest identyczność lub różność od innej czarnej skrzynki. Algorytmy leżące u podstaw wyszukiwania i grania w gry, ukryte modele Markova i procesy decyzyjne Markowa działają z reprezentacjami atomowymi - lub co najmniej traktują reprezentacje tak, jakby były atomowe. Rozważmy teraz bardziej dokładny opis tego samego problemu, w którym musimy zająć się czymś więcej niż tylko lokalizacją atomów w jednym lub innym mieście; być może będziemy musieli zwrócić uwagę na to, ile gazu jest w zbiorniku, nasze aktualne współrzędne GPS, czy działa lampka ostrzegawcza oleju, ile mamy reszty na przejazdach płatnych, jaka stacja jest w radiu i tak dalej reprezentacja z faktorami dzieli każdy stan na ustalony zestaw zmiennych lub atrybutów, z których każdy może mieć wartość. Podczas gdy dwa różne stany atomowe nie mają ze sobą nic wspólnego - to po prostu różne czarne skrzynki - dwa różne stany faktoryzowane mogą mieć wspólne atrybuty (np. przebywanie w określonej lokalizacji GPS), a inne nie (takie jak dużo gazu lub brak gazu); dzięki temu dużo łatwiej jest wymyślić, jak zmienić jeden stan w inny. W przypadku reprezentacji z faktorami możemy również przedstawić niepewność - na przykład ignorancję dotyczącą ilości gazu w zbiorniku można przedstawić, pozostawiając ten atrybut pusty. Wiele ważnych obszarów sztucznej inteligencji opiera się na reprezentacjach faktoryzowanych, w tym algorytmy spełniania ograniczeń, logika zdań, planowanie , sieci bayesowskie oraz algorytmy uczenia maszynowego . Z wielu powodów musimy rozumieć, że świat zawiera rzeczy, które są ze sobą powiązane, a nie tylko zmienne z wartościami. Na przykład możemy zauważyć, że duża ciężarówka przed nami wjeżdża tyłem na podjazd farmy mlecznej,

ale krowa się poluzowała i blokuje drogę ciężarówki. Jest mało prawdopodobne, aby reprezentacja faktoryzowana była wstępnie wyposażona w atrybut

TruckAheadBackingIntoDairyFarmDrivewayBlockedByLooseCow z wartością true lub false. Zamiast tego potrzebowalibyśmy ustrukturyzowanej reprezentacji, w której obiekty takie jak krowy i ciężarówki oraz ich różne i zróżnicowane relacje można wyraźnie opisać. Ustrukturyzowane reprezentacje leżą u podstaw relacyjnych baz danych i logiki pierwszego rzędu, modeli prawdopodobieństwa pierwszego rzędu , uczenia się opartego na wiedzy i wielu innych rozumienie języka naturalnego . W rzeczywistości prawie wszystko, co ludzie wyrażają w języku naturalnym, dotyczy przedmiotów i ich relacji. Jak wspomnieliśmy wcześniej, oś, wzdłuż której leżą reprezentacje atomowe, faktoryzowane i strukturalne, jest osią rosnącej ekspresji. Z grubsza rzecz biorąc, bardziej ekspresyjna reprezentacja może uchwycić, przynajmniej tak zwięźle, wszystko, co może uchwycić mniej ekspresyjna, a także trochę więcej. Często bardziej wyrazisty język jest znacznie bardziej zwięzły; na przykład reguły gry w szachy mogą być zapisane na jednej lub dwóch stronach w języku strukturalnej reprezentacji, takim jak logika pierwszego rzędu, ale wymagają tysięcy stron, gdy są zapisane w języku reprezentacji faktorowej, takim jak logika zdań. Z drugiej strony, rozumowanie i uczenie się stają się bardziej złożone, gdy wzrasta siła ekspresji reprezentacji. Aby czerpać korzyści z ekspresyjnych reprezentacji, jednocześnie unikając ich wad, inteligentne systemy dla świata rzeczywistego mogą wymagać jednoczesnego działania we wszystkich punktach wzdłuż osi.

PODSUMOWANIE

Ta część była czymś w rodzaju burzliwej wycieczki po sztucznej inteligencji, którą pomyśleliśmy jako nauka o projektowaniu agentów. Najważniejsze punkty do zapamiętania są następujące:

* Agent to coś, co postrzega i działa w środowisku. Funkcja agenta dla agenta określa działanie podejmowane przez agenta w odpowiedzi na dowolną sekwencję percepcji.

* Miara wydajności ocenia zachowanie agenta w środowisku. Racjonalny agent działa tak, aby zmaksymalizować oczekiwaną wartość miary wydajności, biorąc pod uwagę sekwencję percepcji, którą widział do tej pory.

* Specyfikacja środowiska zadań obejmuje miarę wydajności, środowisko zewnętrzne, siłowniki i czujniki. Podczas projektowania agenta pierwszym krokiem zawsze musi być jak najpełniejsze określenie środowiska zadań.

* Środowiska zadań różnią się w kilku istotnych wymiarach. Mogą być w pełni lub częściowo obserwowalne, jednoczynnikowe lub wieloagentowe, deterministyczne lub stochastyczne, epizodyczne lub sekwencyjne, statyczne lub dynamiczne, dyskretne lub ciągłe oraz znane lub nieznane.

* Program agenta implementuje funkcję agenta. Istnieje wiele podstawowych projektów agentów i programów odzwierciedlających rodzaj informacji, które są jawne i wykorzystywane w procesie decyzyjnym. Projekty różnią się wydajnością, zwartością i elastycznością. Odpowiedni projekt programu agenta zależy od charakteru środowiska.

* Proste środki odruchowe reagują bezpośrednio na spostrzeżenia, podczas gdy środki odruchowe oparte na modelach utrzymują stan wewnętrzny, aby śledzić aspekty świata, które nie są widoczne w obecnym postrzeganiu. Agenci nastawieni na cele działają, aby osiągnąć swoje cele, a agenci działający w oparciu o użyteczność starają się zmaksymalizować własne oczekiwane „szczęście”.

* Wszyscy agenci mogą poprawić swoje wyniki poprzez naukę.