Biorąc pod uwagę użycie algorytmów

Dane to prełom w AI. Ostatnie postępy w sztucznej inteligencji wskazują, że w przypadku niektórych problemów wybór odpowiedniej ilości danych jest ważniejszy niż właściwy algorytm. Na przykład w 2001 r. dwóch badaczy z Microsoftu, Banko i Brilla, w swoim pamiętnym artykule „Skalowanie do bardzo, bardzo dużej korporacji w celu ujednoznacznienia języka naturalnego" wykazało, że jeśli chcesz, aby komputer stworzył model języka, nie potrzebujesz najmądrzejszego algorytmu w mieście. Po wrzuceniu ponad miliarda słów w kontekście problemu, wszystkie algorytmy zaczną działać niewiarygodnie dobrze. Ta częśćł pomaga zrozumieć związek między algorytmami a danymi użytymi do ich wykonania. bez względu na to, ile masz danych, nadal potrzebujesz algorytmu, aby był użyteczny. Ponadto musisz przeprowadzić analizę danych (szereg definiowalnych kroków), aby dane działały poprawnie z wybranymi algorytmami. weźcie skróty. Chociaż AI to inteligentna automatyzacja, czasem automatyzacja musi zająć miejsce w analizie. Maszyny, które same się uczą, są w odległej przyszłości. Nie znajdziesz maszyn, które wiedzą, co jest odpowiednie i może całkowicie ograniczyć dzisiejszą interwencję człowieka. Druga połowa pomaga zrozumieć rolę systemów eksperckich, uczenia maszynowego, głębokiego uczenia się i aplikacji takich jak AlphaGo w zbliżaniu przyszłych możliwości do rzeczywistości.

Zrozumienie roli algorytmów

Ludzie mają tendencję do rozpoznawania AI, gdy narzędzie prezentuje nowatorskie podejście i wchodzi w interakcję z użytkownikiem w sposób podobny do ludzkiego. Przykłady obejmują asystentów cyfrowych, takich jak Siri, Alexa i Cortana. Jednak niektóre inne typowe narzędzia, takie jak routery GPS i wyspecjalizowani planiści (takie jak narzędzia stosowane w celu uniknięcia kolizji samochodowych, samolotów z automatycznym pilotem i ustalania planów produkcji) nawet nie wyglądają jak sztuczna inteligencja, ponieważ są zbyt powszechne i przyjmowane za pewnik. gdy działają za kulisami.

Jest to wyraźnie efekt sztucznej inteligencji, nazwany i opisany przez Pamelę McCorduck, amerykańską pisarkę, która napisała znaczącą historię sztucznej inteligencji w 1979 r. Efekt sztucznej inteligencji stwierdza, że udane inteligentne programy komputerowe wkrótce tracą uznanie ludzi i stają się cichymi aktorami, podczas gdy uwaga przechodzi na problemy AI, które wciąż wymagają rozwiązania. Ludzie nie zdają sobie sprawy ze znaczenia klasycznych algorytmów dla sztucznej inteligencji i zaczynają fantazjować na temat sztucznej inteligencji stworzonej z ezoterycznej technologii lub utożsamiając ją z najnowszymi osiągnięciami, takimi jak uczenie maszynowe i głębokie uczenie się. Algorytm to procedura, która jest sekwencją operacji, zwykle wykonywaną przez komputer, który gwarantuje znalezienie właściwego rozwiązania problemu w skończonym czasie lub informuje, że nie ma rozwiązania. Chociaż ludzie rozwiązywali algorytmy ręcznie przez dosłownie tysiące lat, może to pochłonąć ogromne ilości czasu i wymagać wielu obliczeń numerycznych, w zależności od złożoności problemu, który chcesz rozwiązać. Algorytmy polegają na szukaniu rozwiązań, a im szybciej i łatwiej, tym lepiej. Algorytmy zostały zakodowane na stałe w inteligencji ludzi, którzy je opracowali, a każda maszyna działająca na algorytmach musi jedynie odzwierciedlać inteligencję wbudowaną w takie procedury algorytmiczne.

Zrozumienie, co oznacza algorytm

Algorytm zawsze przedstawia serię kroków, ale niekoniecznie wykonuje wszystkie te kroki, aby rozwiązać problem. Zakres algorytmów jest niezwykle duży. Operacje mogą obejmować przechowywanie danych, eksplorowanie ich oraz porządkowanie lub porządkowanie ich w strukturach danych. Możesz znaleźć algorytmy, które rozwiązują problemy w nauce, medycynie, finansach, produkcji przemysłowej i dostawach oraz komunikacji. Wszystkie algorytmy są sekwencjami operacji mających na celu znalezienie właściwego rozwiązania problemu w rozsądnym czasie (lub zgłoszenie

się, jeśli nie znaleziono rozwiązania). Algorytmy AI odróżniają się od algorytmów ogólnych, rozwiązując problemy, których rozwiązanie uważa się za typowe (lub nawet wyłącznie) produkt ludzkiego inteligentnego zachowania. Algorytmy AI mają tendencję do radzenia sobie ze złożonymi problemami, które często należą do klasy problemów NP-zupełnych (gdzie NP to niedeterministyczny czas wielomianowy), z którymi ludzie rutynowo radzą sobie, stosując połączenie racjonalnego podejścia i intuicji. Oto tylko kilka przykładów:

* Planowanie problemów i przydzielanie ograniczonych zasobów

* Wyszukiwanie tras w złożonych przestrzeniach fizycznych lub figuratywnych

* Rozpoznawanie wzorców w wizji obrazu (w przeciwieństwie do przywracania obrazu lub przetwarzania obrazu) lub percepcji dźwięku

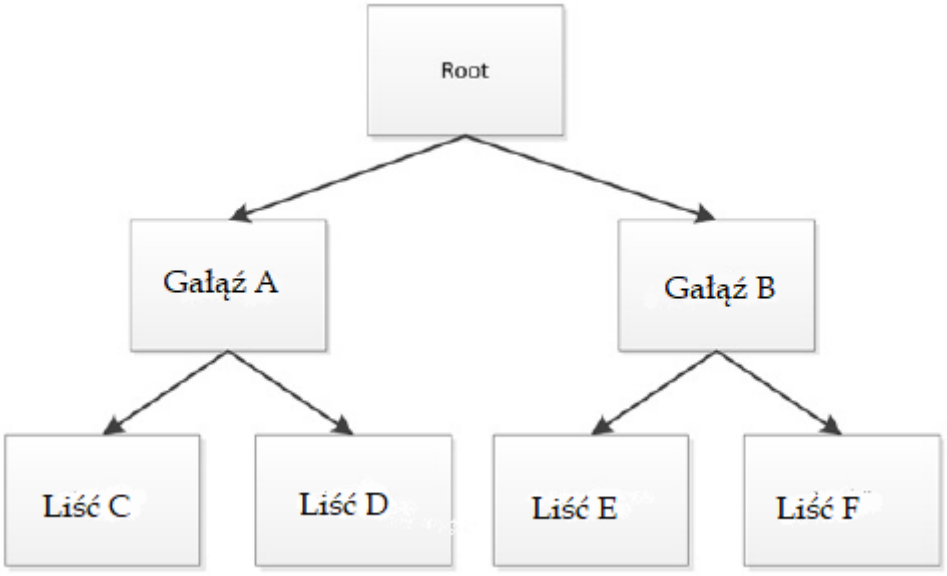
* Język przetwarzania (zarówno rozumienie tekstu, jak i tłumaczenie językowe)

* Granie (i wygrywanie) konkurencyjnych gier

Problemy z NP-complete odróżniają się od innych problemów algorytmicznych, ponieważ znalezienie dla nich rozwiązania w rozsądnych ramach czasowych nie jest jeszcze możliwe. NP-complete nie jest rodzajem problemu, który rozwiązujesz, wypróbowując wszystkie możliwe kombinacje lub możliwości. Nawet jeśli posiadasz komputery o większej mocy niż te, które istnieją obecnie, poszukiwanie rozwiązania potrwa prawie wiecznie. W podobny sposób w AI ten rodzaj problemu nazywa się AI-complete.

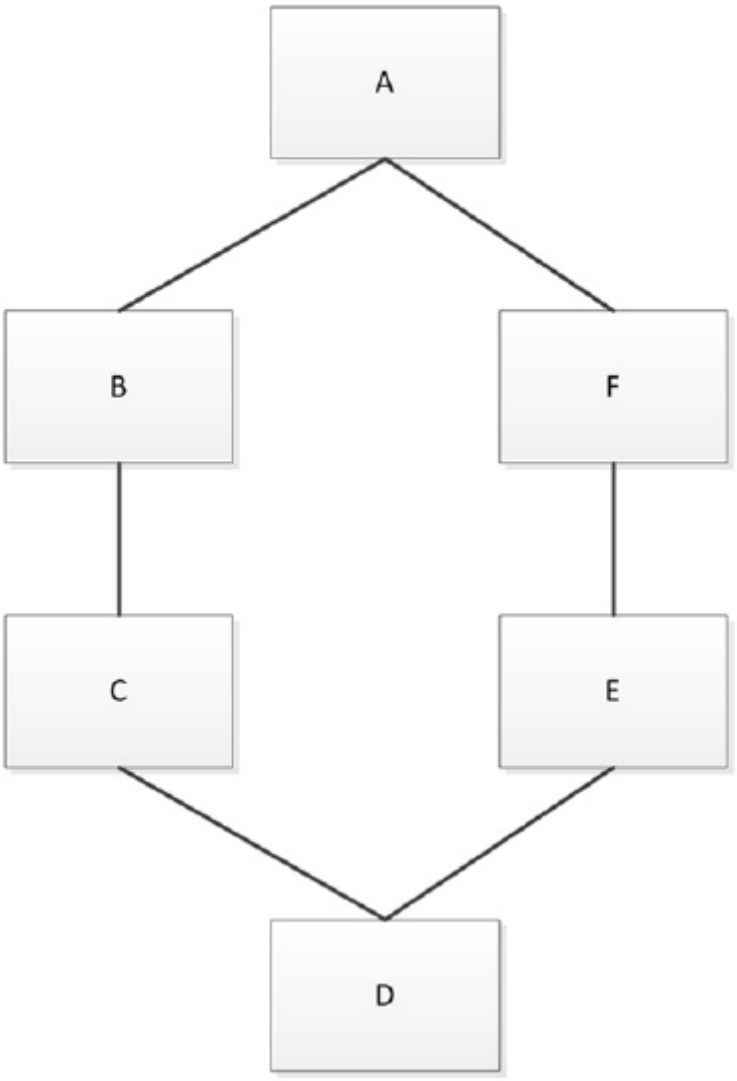
Począwszy od planowania i rozgałęziania

Planowanie pomaga określić sekwencję działań, które należy wykonać, aby osiągnąć określony cel. Jest to klasyczny problem sztucznej inteligencji i można znaleźć przykłady planowania produkcji przemysłowej, alokacji zasobów i przenoszenia robota w pomieszczeniu. Począwszy od obecnego stanu, AI najpierw określa wszystkie możliwe działania z tego stanu. Technicznie rozszerza obecny stan na szereg przyszłych stanów. Następnie rozszerza wszystkie przyszłe stany na ich własne przyszłe stany i tak dalej. Kiedy nie można już rozszerzyć stanów, a sztuczna inteligencja zatrzymuje ekspansję, sztuczna inteligencja utworzyła przestrzeń stanów, która składa się z wszystkiego, co może się zdarzyć w przyszłości. AI może wykorzystać przestrzeń stanu nie tylko jako możliwą prognozę (w rzeczywistości przewiduje wszystko, chociaż niektóre przyszłe stany są bardziej prawdopodobne niż inne) ale także dlatego, że sztuczna inteligencja może wykorzystać tę przestrzeń stanu do zbadania decyzji, które może podjąć, aby osiągnąć swój cel w najlepszy sposób. Jest to znane jako wyszukiwanie w przestrzeni stanu. Praca z przestrzenią stanów wymaga użycia zarówno określonych struktur danych, jak i algorytmów. Podstawowymi często stosowanymi strukturami danych są drzewa i wykresy. Preferowane algorytmy wykorzystywane do efektywnego eksplorowania wykresów obejmują wyszukiwanie od pierwszego do końca lub wyszukiwanie od pierwszego do głębokiego. Budowanie drzewa działa podobnie jak budowanie drzewa w świecie fizycznym. Każdy element dodawany do drzewa jest węzłem. Węzły łączą się ze sobą za pomocą łączy. Połączenie węzłów i łączy tworzy strukturę, która wygląda jak drzewo



Drzewa mają jeden węzeł główny, podobnie jak drzewo fizyczne. Węzeł główny jest punktem początkowym wykonywanego przetwarzania. Z korzeniem są połączone gałęzie lub liście. Węzeł liścia jest punktem końcowym dla drzewa. Węzły gałęzi obsługują inne gałęzie lub liście. Typ drzewa pokazany na rysunku jest drzewem binarnym, ponieważ każdy węzeł ma co najwyżej dwa połączenia (ale drzewa reprezentujące przestrzenie stanu mogą mieć wiele gałęzi).

Patrząc na drzewo, gałąź B jest dzieckiem węzła głównego. Jest tak ponieważ węzeł główny pojawia się pierwszy na liście. Zarówno liść E, jak i liść F są dziećmi gałęzi B, przez co gałąź B jest rodzicem liścia E i liścia F. Relacje między węzłami są ważne, ponieważ dyskusje na temat drzew często uwzględniają relacje dziecko / rodzic między węzłami. Bez tych warunków dyskusje na temat drzew mogłyby stać się dość mylące. Wykres jest rodzajem rozszerzenia drzewa. Podobnie jak w przypadku drzew, masz węzły, które łączą się ze sobą, aby tworzyć relacje. Jednak w przeciwieństwie do drzew binarnych węzeł wykresu może mieć więcej niż jedno lub dwa połączenia. W rzeczywistości węzły grafów często mają wiele połączeń, a co najważniejsze, węzły mogą łączyć się w dowolnym kierunku, nie tylko od rodzica do dziecka. Aby jednak uprościć sprawę, rozważ wykres pokazany na rysunku



Wykresy to struktury przedstawiające wiele węzłów (lub wierzchołków) połączonych pewną liczbą krawędzi lub łuków (w zależności od reprezentacji). Kiedy myślisz o wykresie, pomyśl o strukturze takiej jak mapa, w której każda lokalizacja na mapie jest węzłem, a ulice są krawędziami. Ta prezentacja różni się od drzewa, gdzie każda ścieżka kończy się w węźle liścia. Wykresy są szczególnie przydatne przy ustalaniu stanów reprezentujących rodzaj przestrzeni fizycznej. Na przykład GPS wykorzystuje wykres do reprezentowania miejsc i ulic. Wykresy dodają również kilka nowych zwrotów akcji, których mogłeś nie wziąć pod uwagę. Na przykład wykres może obejmować pojęcie kierunkowości. W przeciwieństwie do drzewa, które ma relacje rodzic / dziecko, węzeł grafu może łączyć się z dowolnym innym węzłem z określonym kierunkiem. Pomyśl o ulicach w mieście. Większość ulic jest dwukierunkowa, ale niektóre są ulicami jednokierunkowymi, które umożliwiają ruch tylko w jednym kierunku. Prezentacja połączenia wykresu może nie odzwierciedlać rzeczywistości wykresu. Wykres może oznaczać wagę dla konkretnego połączenia. Ciężar może określać odległość między dwoma punktami, określać czas potrzebny do pokonania trasy lub zapewniać inne rodzaje informacji. Drzewo to nic innego jak wykres, na którym dowolne dwa wierzchołki są połączone dokładnie jedną ścieżką, co nie pozwala na cykle (aby móc wrócić do rodzica od dowolnego dziecka). Wiele algorytmów graficznych dotyczy tylko drzew. Przemierzanie wykresu oznacza wyszukiwanie (odwiedzanie) każdego wierzchołka (węzła) w określonej kolejności. Proces odwiedzania wierzchołka może obejmować zarówno jego czytanie, jak i aktualizację. Podczas przeglądania wykresu odkrywasz nieodwiedzone wierzchołki. Wierzchołek

zostaje odkryty (ponieważ właśnie go odwiedziłeś) lub przetworzony (ponieważ algorytm wypróbował wszystkie odchodzące od niego krawędzie) po wyszukiwaniu. Kolejność wyszukiwania określa rodzaj przeprowadzonego wyszukiwania: niedoinformowane (ślepe wyszukiwanie) i poinformowane (heurystyczne). W niedoinformowanej strategii sztuczna inteligencja bada przestrzeń stanu bez dodatkowych informacji, z wyjątkiem wykreślonej przez nią struktury wykresu podczas jej przemierzania. W poniższych sekcjach omówiono dwa popularne algorytmy przeszukiwania w ciemno: wyszukiwanie na szerokość i wyszukiwanie na głębokość. Pełne wyszukiwanie (BFS) rozpoczyna się w katalogu głównym wykresu i bada każdy węzeł, który jest dołączony do katalogu głównego. Następnie przeszukuje następny poziom, badając kolejno każdy poziom, aż dojdzie do końca. W rezultacie na przykładowym wykresie wyszukiwanie bada od A do B i C, zanim przejdzie do eksploracji D. BFS bada wykres w sposób systematyczny, badając wierzchołki wokół wierzchołka początkowego w sposób kołowy. Zaczyna się od odwiedzenia wszystkich wierzchołków o krok od wierzchołka początkowego; następnie przesuwa się o dwa kroki, potem trzy kroki i tak dalej. Wyszukiwanie w pierwszej kolejności (DFS) rozpoczyna się od katalogu głównego wykresu, a następnie bada każdy węzeł od tego katalogu głównego w dół pojedynczą ścieżką do końca. Następnie cofa się i zaczyna eksplorować ścieżki, które nie zostały wybrane w bieżącej ścieżce wyszukiwania, dopóki nie dotrze ponownie do katalogu głównego. W tym momencie, jeśli dostępne są inne ścieżki do pobrania z katalogu głównego, algorytm wybiera jedną i ponownie rozpoczyna to samo wyszukiwanie. Chodzi o to, aby zbadać każdą ścieżkę całkowicie przed eksploracją innej ścieżki.

Granie w gry przeciwne

Interesującą rzeczą w wyszukiwaniu w przestrzeni stanów jest to, że reprezentuje zarówno bieżącą funkcjonalność AI, jak i przyszłe możliwości. Dotyczy to gier przeciwnych (gier, w których jeden wygrywa, a inni przegrywają) lub jakiejkolwiek podobnej sytuacji, w której gracze dążą do celu sprzecznego z celami innych. Prosta gra, taka jak kółko i krzyżyk, stanowi doskonały przykład gry w kosmiczne poszukiwania, w którą mogłeś już grać w AI. W filmie Gry wojenne z 1983 r. Superkomputer WOPR (War Operation Plan Response) gra przeciwko sobie z niesamowitą szybkością, ale nie może wygrać, ponieważ gra jest naprawdę prosta i jeśli użyjesz przeszukiwania przestrzeni stanu, nigdy nie przegrasz. Masz dziewięć komórek do wypełnienia X i O dla każdego gracza. Pierwszy, który umieści trzy znaki w rzędzie (poziomym, pionowym lub ukośnym) wygrywa. Podczas budowania drzewa w przestrzeni stanów dla drzewa, każdy poziom drzewa reprezentuje turę gry. Węzły końcowe reprezentują ostateczny stan planszy i określają zwycięstwo, remis lub porażkę dla AI. Każdy węzeł końcowy ma wyższy wynik za wygraną, niższy za losowanie, a nawet niższy lub ujemny za przegraną. AI propaguje wyniki do górnych węzłów i gałęzi za pomocą sumowania, aż do osiągnięcia węzła początkowego. Węzeł początkowy reprezentuje rzeczywistą sytuację. Użycie prostej strategii pozwala ci przejść przez drzewo: kiedy nadchodzi kolej AI i musisz propagować wartości wielu węzłów, sumujesz maksymalną wartość (prawdopodobnie dlatego, że AI musi uzyskać maksymalny wynik z gry); kiedy nadchodzi kolej przeciwnika, sumujesz minimalną wartość. W końcu otrzymujesz drzewo, którego gałęzie są klasyfikowane na podstawie wyników. Kiedy nadchodzi kolej AI, wybiera swój ruch w oparciu o gałąź, której wartość jest najwyższa, ponieważ implikuje to powiększanie węzłów z najwyższą możliwością wygranej.

Takie podejście nazywa się przybliżeniem min-maks. Ronald Rivest, z laboratorium informatycznego w MIT, wprowadził go w 1987 roku. Od tego czasu ten algorytm i jego warianty napędzają wiele konkurencyjnych gier, a także najnowsze osiągnięcia w grze, takie jak AlphaGo z Google DeepMind, która wykorzystuje podejście, które przypomina przybliżenie minimum-maksimum (które znajduje się również w filmie Gry wojenne z 1983 roku). Czasami słyszysz o przycinaniu alfa-beta w połączeniu z przybliżeniem min-maks. Przycinanie alfa-beta to inteligentny sposób propagowania wartości w górę

hierarchii drzew w złożonych przestrzeniach stanu ograniczających obliczenia. Nie wszystkie gry zawierają zwarte drzewa w przestrzeni stanów; gdy twoje gałęzie są w liczbie milionów, musisz je przyciąć i skrócić swoje obliczenia.

Korzystanie z wyszukiwania lokalnego i heurystyki

Wiele dzieje się za podejściem do wyszukiwania w przestrzeni stanu. W końcu żadna maszyna, bez względu na to, jak potężna, nie jest w stanie wymienić wszystkich możliwości wynikających z sytuacji. W tej sekcji omówiono gry, ponieważ są przewidywalne i mają ustalone reguły, podczas gdy wiele rzeczywistych sytuacji jest nieprzewidywalnych i pozbawionych jasnych zasad, co czyni gry optymistycznymi i sprzyjającymi warunkami. Checkers, stosunkowo prosta gra w porównaniu do szachów lub Go, ma 500 miliardów (500 000 000 000 000 000 000 000) możliwych pozycji na planszy, co według obliczeń matematyków z Uniwersytetu Hawajskiego równa się wszystkim ziarnom piasku na Ziemi. To prawda, że w miarę postępu gry w warcaby możliwe jest zmniejszenie liczby ruchów. Jednak liczba potencjalnie oceniana przy każdym ruchu jest zbyt wysoka. Wykorzystanie potężnych komputerów zajęło 18 lat do obliczenia wszystkich 500 miliardów możliwych ruchów. Wyobraź sobie, ile czasu może zająć komputer konsumenta, aby wypracować jeszcze mniejszy podzbiór ruchów. Aby było możliwe do zarządzania, powinien to być bardzo mały podzbiór wszystkich potencjalnych ruchów. Optymalizacja za pomocą lokalnego wyszukiwania i heurystyki pomaga ograniczać początkową liczbę możliwych ocen (jak w przycinaniu alfa, gdzie niektóre obliczenia są pomijane, ponieważ nie dodają niczego do sukcesu wyszukiwania). Wyszukiwanie lokalne to ogólne podejście do rozwiązywania problemów, które obejmuje szeroki zakres algorytmów, które pomagają uniknąć złożoności wykładniczej wielu problemów NP. Lokalne wyszukiwanie rozpoczyna się od bieżącej sytuacji lub niedoskonałego rozwiązania problemu i odsuwa się od niej krok po kroku. Lokalne wyszukiwanie określa wykonalność pobliskich rozwiązań, potencjalnie prowadząc do idealnego rozwiązania, opartego na losowym wyborze lub bystrej heurystyce (co oznacza, że nie stosuje się żadnej dokładnej metody).

Heurystyka to wyrafinowane domysły na temat rozwiązania, takie jak praktyczna zasada, która wskazuje kierunek pożądanego rezultatu, ale nie może dokładnie powiedzieć, jak go osiągnąć. To tak, jakby zagubić się w nieznanym mieście i zachęcać ludzi do wskazania określonego sposobu dotarcia do hotelu (ale bez dokładnych instrukcji) lub odległości od niego.

Lokalne algorytmy wyszukiwania są iteracyjnie ulepszane od stanu początkowego, przechodząc krok po kroku przez sąsiednie rozwiązania w przestrzeni stanu, dopóki nie będą w stanie dalej ulepszać rozwiązania. Ponieważ lokalne algorytmy wyszukiwania są tak proste i intuicyjne, zaprojektowanie lokalnego wyszukiwania problemu algorytmicznego nie jest trudne; zwiększenie skuteczności jest zwykle trudniejsze. Kluczem jest zdefiniowanie prawidłowej procedury:

1. Zacznij od istniejącej sytuacji (może to być obecna sytuacja lub przypadkowe lub znane rozwiązanie).

2. Wyszukaj zestaw możliwych nowych rozwiązań w sąsiedztwie obecnego rozwiązania, który stanowi listę kandydatów.

3. Określ, które rozwiązanie zastosować zamiast obecnego rozwiązania na podstawie wyników heurystyki, która przyjmuje listę kandydatów jako dane wejściowe.

4. Kontynuuj wykonywanie kroków 2 i 3, dopóki nie zobaczysz dalszej poprawy rozwiązania, co oznacza, że masz najlepsze dostępne rozwiązanie.

Mimo że łatwe do zaprojektowania, lokalne rozwiązania wyszukiwania mogą nie znaleźć rozwiązania w rozsądnym czasie (możesz zatrzymać proces i skorzystać z obecnego rozwiązania) lub stworzyć

rozwiązanie o minimalnej jakości. Nie masz gwarancji, że wyszukiwanie lokalne doprowadzi do rozwiązania problemu, ale Twoje szanse wzrosną od punktu początkowego, gdy zapewnisz wystarczająco dużo czasu na uruchomienie obliczeń. Zatrzymuje się dopiero wtedy, gdy nie może znaleźć dalszego sposobu na ulepszenie rozwiązania. Sekret polega na określeniu właściwej dzielnicy do eksploracji. Jeśli odkryjesz wszystko, wrócisz do wyczerpującego wyszukiwania, które oznacza eksplozję możliwości odkrywania i testowania. Opierając się na limitach heurystycznych, gdy patrzysz na podstawie praktycznej zasady. Czasami heurystyka jest przypadkowością, a takie rozwiązanie, mimo że nie jest inteligentnym podejściem, może działać dobrze. Na przykład niewiele osób wie, że Roomba, autonomiczny odkurzacz automatyczny stworzony przez trzech absolwentów MIT, początkowo nie planował swojej ścieżki czyszczenia, ale po prostu wędrował losowo. Mimo to został uznany przez właścicieli za inteligentne urządzenie i wykonał doskonałą robotę czyszczącą. (Inteligencja ma na myśli wykorzystanie losowości do rozwiązania problemu, który w innym przypadku byłby zbyt skomplikowany.) Losowy wybór nie jest jedyną dostępną heurystyką. Lokalne wyszukiwanie może polegać na bardziej uzasadnionych rozwiązaniach eksploracyjnych z wykorzystaniem dobrze opracowanej heurystyki, aby uzyskać wskazówki, jak w przypadku optymalizacji wspinaczki lub skręcania, i uniknąć pułapki przyjmowania miernych rozwiązań, jak w symulowanym wyżarzaniu i wyszukiwaniu tabu. Optymalizacja wspinaczki, twiddle, symulowane wyżarzanie i wyszukiwanie tabu to algorytmy wyszukiwania, które skutecznie wykorzystują heurystykę do uzyskania wskazówek. Wspinaczka górska czerpie inspirację z siły grawitacji. Opiera się na spostrzeżeniu, że gdy piłka toczy się w dół doliny, pokonuje strome zejście. Kiedy wspina się na wzgórze, piłka ma tendencję do podążania najbardziej bezpośrednim kierunkiem w górę, aby dotrzeć na szczyt, który ma największe nachylenie. Problem sztucznej inteligencji jest zatem postrzegany jako zejście do doliny lub wspinanie się na szczyt góry, a heurystyka jest dowolną regułą, która wskazuje na najlepsze podejście z góry lub z góry wśród możliwych stanów przestrzeni państwowej. Jest to skuteczny algorytm, chociaż czasami uderza w sytuacje zwane płaskowyżami (doliny pośrednie) i szczytami (lokalne maksimum punktów). Twiddle lub algorytmy zejścia ze współrzędnymi są podobne do algorytmów wspinania się na wzgórze. Heurystyka Twiddle'a polega na eksplorowaniu wszystkich możliwych kierunków, ale koncentrowaniu poszukiwań w kierunku najlepiej działającej okolicy. Czyniąc to, kalibruje swój krok, zwalniając, ponieważ trudno jest znaleźć lepsze rozwiązania, aż do momentu zatrzymania. Termin symulowane wyżarzanie bierze swoją nazwę od techniki metalurgicznej, która ogrzewa metal, a następnie powoli chłodzi go, aby zmiękczyć metal do obróbki na zimno i usuwania wad krystalicznych. Lokalne wyszukiwanie replikuje tę technikę, traktując wyszukiwanie rozwiązania jako strukturę atomową, która zmienia się w celu poprawy jego wykonalności. Temperatura zmienia grę w procesie optymalizacji. Tak jak wysokie temperatury sprawiają, że struktura materiału rozluźnia się (ciała stałe topią się, a ciecze odparowują w wysokich temperaturach), tak wysokie temperatury w lokalnym algorytmie wyszukiwania indukują rozluźnienie funkcji celu, pozwalając mu preferować gorsze rozwiązania od lepszych. Symulowane wyżarzanie modyfikuje procedurę wspinaczki, utrzymanie funkcji celu dla oceny rozwiązania sąsiada, ale pozwalając mu określić wybór rozwiązania wyszukiwania w inny sposób. Wyszukiwanie tabu wykorzystuje zapamiętywanie, aby zapamiętać, które części okolicy należy zbadać. Gdy wydaje się, że znalazło rozwiązanie, stara się wrócić do innych możliwych ścieżek, których nie próbował, aby znaleźć najlepsze rozwiązanie. Używanie miar kierunku (w górę, w dół), temperatury (kontrolowana losowość) lub po prostu ograniczanie lub wyszukiwanie części wyszukiwania to wszystkie sposoby na skuteczne uniknięcie próbowania wszystkiego i skoncentrowanie się na dobrym rozwiązaniu. Rozważmy na przykład chodzącego robota. Prowadzenie robota w nieznanym środowisku oznacza unikanie przeszkód w celu osiągnięcia określonego celu. Jest to zarówno podstawowe, jak i trudne zadanie w zakresie sztucznej inteligencji. Roboty mogą polegać na laserowym dalmierzu (LIDAR) lub sonar (który obejmuje urządzenia, które wykorzystują dźwięk, aby zobaczyć otoczenie) do nawigacji

w otoczeniu. Jednak bez względu na poziom zaawansowania sprzętowego roboty nadal potrzebują odpowiednich algorytmów

* Znajdź najkrótszą drogę do celu (lub przynajmniej rozsądnie krótką)

* Unikaj przeszkód na drodze

* Wykonuj niestandardowe zachowania, takie jak minimalizowanie skrętu lub hamowania

Algorytm znajdowania ścieżki pomaga robotowi uruchomić się w jednym miejscu i osiągnąć cel, wykorzystując najkrótszą ścieżkę między nimi, przewidując i omijając przeszkody po drodze. (Reakcja po uderzeniu w ścianę nie jest wystarczająca.) Wyszukiwanie ścieżek jest również przydatne, gdy przenosisz dowolne inne urządzenie do celu w kosmosie, nawet wirtualnego, np. W grze wideo lub na stronach internetowych. Podczas korzystania z wyszukiwania ścieżki z robotem robot postrzega ruch jako przepływ przestrzeni stanów do granic czujników. Jeśli cel nie znajduje się w zasięgu, robot nie będzie wiedział, dokąd iść. Heurystyka może skierować go we właściwym kierunku (na przykład może wiedzieć, że cel jest w kierunku północnym) i pomóc mu w odpowiednim czasie unikać przeszkód bez konieczności określania wszystkich możliwych sposobów.

Odkrywanie uczenia się maszyny

Wszystkie dotychczasowe przykłady algorytmów są powiązane ze sztuczną inteligencją, ponieważ są inteligentnymi rozwiązaniami, które rozwiązują powtarzalne i dobrze określone, ale złożone problemy wymagające inteligencji. Wymagają architekta, który bada problem i wybiera odpowiedni algorytm do jego rozwiązania. Zmiany problemów, mutacje lub nietypowe wyświetlanie charakterystyczne mogą stać się prawdziwym problemem dla pomyślnego wykonania algorytmu. Wynika to z faktu, że poznanie problemu i jego rozwiązanie następuje raz na zawsze w momencie pojawienia się algorytmu w oprogramowaniu. Na przykład możesz bezpiecznie zaprogramować AI do rozwiązywania Sudoku. Możesz nawet zapewnić elastyczność, która pozwala algorytmowi na przyjęcie większej liczby reguł lub większych kart później. Peter Norvig, dyrektor ds. badań w Google, napisał niezwykle interesujący esej na ten temat, który pokazuje, jak mądrze korzystać z wyszukiwania w pierwszej kolejności, ograniczając liczbę obliczeń (w przeciwnym razie obliczenia mogą trwać wiecznie), stosując ograniczenia i badając mniejsze najpierw oddziały mogą umożliwić rozwiązania Sudoku. Niestety, nie wszystkie problemy mogą polegać na rozwiązaniu podobnym do Sudoku. Problemy Reallife nigdy nie są osadzone w prostych światach doskonałych informacji i dobrze określonych działań. Rozważ problem ze znalezieniem oszusta oszukującego roszczenia ubezpieczeniowe lub problem diagnozowania choroby medycznej:

* Duży zestaw zasad i możliwości: liczba możliwych oszustw jest niewiarygodnie wysoka; wiele chorób ma podobne objawy.

* Brakujące informacje: Oszuści mogą ukrywać informacje; lekarze często polegają na niepełnych informacjach (mogą brakować badań).

* Zasady problemów nie są niezmienne: Oszuści odkrywają nowe sposoby organizowania oszustw lub oszustw; nowe choroby powstają lub zostają odkryte.

Aby rozwiązać takie problemy, nie możesz zastosować z góry określonego podejścia, ale potrzebujesz elastycznego podejścia i musisz zgromadzić przydatną wiedzę, aby stawić czoła każdemu nowemu wyzwaniu. Innymi słowy, kontynuujesz naukę, tak jak ludzie muszą robić przez całe życie, aby radzić sobie w zmieniającym się i trudnym środowisku.

Wykorzystanie systemów eksperckich

Systemy eksperckie były pierwszą próbą ucieczki z królestwa algorytmów na stałe i stworzenia bardziej elastycznych i inteligentnych sposobów rozwiązywania rzeczywistych problemów. Idea leżąca u podstaw systemów eksperckich była prosta i dobrze nadawała się w czasach, gdy przechowywanie i przetwarzanie dużej ilości danych w pamięci komputera było nadal kosztowne. Dzisiaj może to zabrzmieć dziwnie, ale w latach 70. XX wieku naukowcy zajmujący się sztuczną inteligencją, tacy jak Ross Quillian, musieli zademonstrować, jak budować modele języków roboczych oparte na słownictwie składającym się tylko z 20 słów, ponieważ pamięć komputerowa tamtych czasów mogła pomieścić tylko tyle. Dostępnych było niewiele opcji, jeśli komputer nie mógł pomieścić wszystkich danych, a rozwiązaniem było radzenie sobie z kluczowymi informacjami o problemach i uzyskiwanie ich od ludzi, którzy znali je najlepiej. Systemy eksperckie były ekspertami nie dlatego, że oparli swoją wiedzę na własnym procesie uczenia się, ale raczej dlatego, że zgromadzili ją od ludzkich ekspertów, którzy dostarczyli wstępnie ustalony system kluczowych informacji zaczerpniętych ze studiowania książek, uczenia się od innych ekspertów lub odkrywania ich przez nich samych. Był to w zasadzie sprytny sposób na przekazanie wiedzy na maszynę. Przykładem jednego z pierwszych tego rodzaju systemów jest MYCIN, system do diagnozowania chorób krzepnięcia krwi lub infekcji wywołanych przez bakterie, takich jak bakteriemia (bakterie infekują krew) i zapalenie opon mózgowych (zapalenie błon chroniących mózg i rdzeń kręgowy sznur). MYCIN zalecił odpowiednią dawkę antybiotyków, stosując ponad 500 zasad i polegał, w razie potrzeby, na lekarzach korzystających z systemu. Kiedy nie było wystarczających informacji, na przykład brakujących badań laboratoryjnych, MYCIN rozpoczął następnie konsultacyjny dialog, zadając odpowiednie pytania, aby uzyskać pewną diagnozę i terapię.

MYCIN, napisany w LisP jako rozprawa doktorska Edwarda Shortliffe'a na Uniwersytecie Stanforda, zajął ponad pięć lat i osiągnął lepsze wyniki niż jakikolwiek młodszy lekarz, osiągając wysoką dokładność diagnozy doświadczonego lekarza. Pochodzi z tego samego laboratorium, które opracowało DENDRAL, pierwszy system ekspercki, jaki kiedykolwiek stworzono, kilka lat wcześniej. DENDRAL, specjalizująca się w chemii organicznej, jest trudną aplikacją, w której algorytmy brutalnej siły okazały się niewykonalne w obliczu heurystyk opartych na ludziach, które opierają się na doświadczeniu w terenie. Jeśli chodzi o sukces MYCIN, pojawiły się pewne problemy. Po pierwsze, warunki odpowiedzialności były niejasne. (Gdyby system miał postawić błędną diagnozę, kto wziął na siebie odpowiedzialność?) Po drugie, MYCIN miał problem z użytecznością, ponieważ lekarz musiał połączyć się z MYCIN za pomocą zdalnego terminala do komputera mainframe w Stanford, co jest dość trudne i powolne na raz kiedy Internet był jeszcze w powijakach. MYCIN wciąż udowodnił swoją skuteczność i użyteczność we wspieraniu ludzkich decyzji, i utorował drogę dla wielu innych systemów eksperckich, które rozprzestrzeniły się w latach 70. i 80. XX wieku. Ogólnie rzecz biorąc, ówczesne systemy eksperckie składały się z dwóch różnych komponentów: bazy wiedzy i silnika wnioskowania. Baza wiedzy zachowuje wiedzę jako zbiór reguł w postaci instrukcji „jeśli-to” (jeśli dotyczy jednego lub wielu warunków, a następnie obejmuje stwierdzenia końcowe). Te instrukcje występowały w formie symbolicznej, rozróżniającej instancje (pojedyncze zdarzenia lub fakty), klasy i podklasy, którymi można manipulować za pomocą logiki logicznej lub wyrafinowanej logiki pierwszego rzędu, która obejmuje więcej możliwych operacji. Logika pierwszego rzędu to zestaw operacji, który wykracza poza zwykłe łączenie twierdzeń PRAWDA i FAŁSZ. Na przykład wprowadza takie pojęcia, jak WSZYSTKO lub ISTNIEJE, umożliwiając radzenie sobie z oświadczeniami, które mogą być prawdziwe, ale których nie można udowodnić na podstawie dowodów, którymi dysponujesz w tym momencie. Mechanizm wnioskowania to zestaw instrukcji, które mówią systemowi, jak manipulować warunkami w oparciu o logiczny zestaw operatorów logicznych, takich jak AND, OR, NOT. Korzystając z tego zestawu logicznego, PRAWDA (reguła jest uruchamiana lub, technicznie rzecz biorąc, „odpalana”) lub FAŁSZ (reguła nie ma zastosowania), warunki symboliczne mogą łączyć się w złożone rozumowanie. Ponieważ system powstał w oparciu o serię if-ów (warunki) i następnie (wnioski), a także został zagnieżdżony i

ustrukturyzowany w warstwach, uzyskanie wstępnych informacji pomogło wykluczyć niektóre wnioski, a jednocześnie pomogło systemowi w interakcji z użytkownikiem na temat informacji, które może prowadzić do odpowiedzi. W przypadku silnika wnioskowania typowe operacje systemów ekspertowych były następujące:

* Łączenie w przód: dostępne dowody uruchomiły szereg zasad i wykluczyły inne na każdym etapie. System początkowo koncentrował się na regułach, które mogły doprowadzić do ostatecznego zakończenia strzelania. Takie podejście jest wyraźnie sterowane danymi.

* Łączenie wstecz: System ocenia każdy możliwy wniosek i próbuje udowodnić każdy z nich na podstawie dostępnych dowodów. To podejście ukierunkowane na cel pomaga określić, które pytania należy postawić, i wyklucza całe zestawy celów. MYCIN zastosował łączenie wsteczne; przejście od hipotezy do przeszłości do dowodów jest powszechną strategią w diagnostyce medycznej.

* Rozwiązywanie konfliktów: jeśli system dojdzie do więcej niż jednego wniosku w tym samym czasie, system faworyzuje wniosek, który ma pewne cechy (pod względem wpływu, ryzyka lub innych czynników). Czasami system konsultuje się z użytkownikiem, a rozdzielczość jest realizowana na podstawie ocen użytkowników. Na przykład MYCIN zastosował czynnik pewności, który oszacował prawdopodobieństwo dokładności diagnozy.

Jedną wielką zaletą takich systemów było reprezentowanie wiedzy w formie czytelnej dla człowieka, dzięki czemu decyzja była przejrzysta dla zrozumienia i modyfikacji. Jeśli system dojdzie do wniosku, zwraca reguły zastosowane do tego wniosku. Użytkownik może systematycznie sprawdzać działanie systemu i uzgadniać go lub sprawdzać pod kątem oznak błędu wejściowego. Co więcej, systemy eksperckie były łatwe do programowania przy użyciu języków takich jak LisP lub ALGOL. Użytkownicy z czasem ulepszali systemy eksperckie, dodając nowe reguły lub aktualizując istniejące reguły. Można nawet zmusić ich do pracy w niepewnych warunkach poprzez zastosowanie logiki rozmytej (rodzaj logiki wielowartościowej, w której wartość może zawierać dowolne wartości od 0, absolutnie fałszywe, do 1 lub absolutnie prawdziwe). Logika rozmyta unika gwałtownych kroków wyzwalania reguły opartej na progu. Na przykład, jeśli reguła jest ustawiona na wyzwalanie, gdy w pomieszczeniu jest gorąco, reguła nie jest uruchamiana przy dokładnej temperaturze, ale raczej, gdy temperatura osiąga ten próg. Systemy eksperckie były świadkami zmierzchu pod koniec lat osiemdziesiątych, a ich rozwój zatrzymał się, głównie z następujących powodów:

* Logika i symbolika takich systemów okazały się ograniczone w wyrażaniu reguł leżących u podstaw decyzji, co doprowadziło do stworzenia systemów niestandardowych, to znaczy, polegania na sztywnych regułach przy użyciu klasycznych algorytmów.

* W przypadku wielu trudnych problemów systemy eksperckie stały się tak złożone i skomplikowane, że straciły atrakcyjność pod względem wykonalności i kosztów ekonomicznych.

* Ponieważ dane stają się coraz bardziej rozproszone i dostępne, nie ma sensu starać się o dokładny wywiad, gromadzenie i destylację rzadkiej wiedzy eksperckiej, gdy taka sama (lub nawet lepsza) wiedza może zostać przeniesiona z danych.

Systemy eksperckie nadal istnieją. Można je znaleźć w punktacji kredytowej, wykrywaniu oszustw i innych dziedzinach, w których konieczne jest nie tylko udzielenie odpowiedzi, ale także jasne i przejrzyste określenie zasad i decyzji w sposób, który użytkownik systemu uzna za akceptowalny (tak jak zrobiłby to ekspert w tej dziedzinie).

Przedstawiamy uczenie maszynowe

Rozwiązania zdolne do uczenia się bezpośrednio z danych bez uprzedniej predyspozycji do renderowania ich jako symboli pojawiły się kilkadziesiąt lat przed systemami ekspertowymi. Niektóre miały charakter statystyczny; inni naśladowali naturę na różne sposoby; a jeszcze inni próbowali wygenerować autonomiczną logikę symboliczną w formie reguł na podstawie surowych informacji. Wszystkie te rozwiązania pochodzą z różnych szkół i pojawiły się pod różnymi nazwami, które dziś obejmują uczenie maszynowe. Uczenie maszynowe jest częścią świata algorytmów, chociaż w przeciwieństwie do wielu omówionych dotąd algorytmów, nie jest ono przeznaczone jako seria wstępnie zdefiniowanych kroków, które mogą rozwiązać problem. Z reguły uczenie maszynowe rozwiązuje problemy, których ludzie nie potrafią szczegółowo opisać krokami, ale które naturalnie rozwiązują. Przykładem takiego problemu jest rozpoznawanie twarzy na obrazach lub niektórych słowach w dyskusji mówionej. Uczenie maszynowe jest wspomniane w prawie każdej części tego tekstu, ale trzy części poświęcone są ujawnieniu, jak działają główne algorytmy uczenia maszynowego, szczególnie głębokie uczenie się, czyli technologia napędzająca nową falę aplikacji AI, która dociera do nagłówków wiadomości prawie codziennie.

Dotykając nowych wysokości

Rola uczenia maszynowego w nowej fali algorytmów AI polega częściowo na zastąpieniu, a częściowo uzupełnieniu, istniejących algorytmów poprzez udostępnienie czynności wymagających inteligencji z ludzkiego punktu widzenia, który nie jest łatwy do sformalizowania jako dokładna sekwencja kroków . Wyraźnym przykładem tej roli jest mistrzostwo pokazane przez eksperta Go, który na pierwszy rzut oka rozumie zagrożenia i możliwości konfiguracji tablicy oraz chwyta intuicję właściwych ruchów. Go to niezwykle złożona gra dla AI. Szachy mają średnio 35 możliwych ruchów do oceny na planszy, a gra zwykle obejmuje ponad 80 ruchów, podczas gdy gra Go ma około 140 ruchów do oceny, a gra zwykle obejmuje ponad 240 ruchów. Brak mocy obliczeniowej obecnie istnieje na świecie, aby stworzyć pełną przestrzeń stanu dla gry Go. Zespół Google DeepMind w Londynie opracował AlphaGo, program, który pokonał wielu czołowych graczy Go. Program nie opiera się na podejściu algorytmicznym opartym na przeszukiwaniu ogromnej przestrzeni stanów, ale zamiast tego wykorzystuje:

* Metodę inteligentnego wyszukiwania opartą na losowych testach możliwego ruchu. Sztuczna inteligencja stosuje przeszukiwanie w pierwszej kolejności wiele razy, aby ustalić, czy pierwszy znaleziony wynik jest dodatni czy ujemny (niekompletna i częściowa przestrzeń stanu).

* Algorytm głębokiego uczenia przetwarza obraz planszy (na pierwszy rzut oka) i wyprowadza zarówno najlepszy możliwy ruch w tej sytuacji (algorytm nazywa się siecią polis), jak i oszacowanie prawdopodobieństwa, że AI wygra grę, używając ten ruch (algorytm nazywa się siecią wartości).

* Możliwość uczenia się, oglądając poprzednie gry ekspertów Go i grając przeciwko sobie, podobnie jak WOPR w filmie WarGames z 1983 roku. Najnowsza wersja programu, o nazwie AlphaGo Zero, może nauczyć się wszystkiego sama, bez żadnych ludzkich przykładów. Ta zdolność uczenia się nazywa się uczeniem przez wzmacnianie.