Nowa sztuczna inteligencja: ogólna, dźwiękowa i istotna dla fizyki

Większość tradycyjnych systemów sztucznej inteligencji (AI) z ostatnich 50 lat jest albo bardzo ograniczona, albo opiera się na heurystyce, albo na obu. Nowe tysiąclecie przyniosło jednak znaczny postęp w dziedzinie teoretycznie optymalnych i praktycznie wykonalnych algorytmów przewidywania, wyszukiwania, wnioskowania indukcyjnego w oparciu o brzytwę Ockhama, rozwiązywania problemów, podejmowania decyzji i uczenia się przez wzmacnianie w środowiskach o bardzo ogólnym typie. Ponieważ wnioskowanie indukcyjne leży u podstaw wszystkich nauk indukcyjnych, niektóre wyniki mają znaczenie nie tylko dla sztucznej inteligencji i informatyki, ale także dla fizyki, prowokując do nietradycyjnych przewidywań opartych na tezie Zuse'a o wszechświecie wygenerowanym komputerowo.

Wprowadzenie

Co ciekawe, istnieje teoretycznie optymalny sposób przewidywania na podstawie obserwacji, zakorzeniony we wczesnych pracach Solomonoffa i Kołmogorowa. Podejście to odzwierciedla podstawowe zasady brzytwy Ockhama: lepsze są proste wyjaśnienia danych niż skomplikowane. Teoria uniwersalnego wnioskowania indukcyjnego określa ilościowo, co naprawdę oznacza prostota. Biorąc pod uwagę pewne bardzo szerokie założenia dotyczące obliczalności, zapewnia techniki tworzenia optymalnie wiarygodnych stwierdzeń na temat przyszłych zdarzeń, biorąc pod uwagę przeszłość. Gdy istnieje optymalny, formalnie możliwy do opisania sposób przewidywania przyszłości, powinniśmy być w stanie skonstruować maszynę, która w sposób ciągły oblicza i wykonuje sekwencje działań, które maksymalizują oczekiwaną lub przewidywaną nagrodę, rozwiązując w ten sposób starożytny cel badań nad sztuczną inteligencją. Jednak przez wiele dziesięcioleci badacze sztucznej inteligencji nie zwracali zbytniej uwagi na teorię wnioskowania indukcyjnego. Dlaczego nie? Jest jeszcze jeden powód, poza faktem, że większość z nich tradycyjnie ignorowała informatykę teoretyczną: teoria ta była postrzegana jako związana z nadmiernymi kosztami obliczeniowymi. W rzeczywistości jego najbardziej ogólne stwierdzenia odnoszą się do metod, które są optymalne (w pewnym sensie asymptotycznym), ale nieobliczalne. Dlatego badacze zajmujący się uczeniem maszynowym i sztuczną inteligencją często uciekają się do metod alternatywnych, którym brakuje solidnych podstaw teoretycznych, ale przynajmniej wydają się wykonalne w pewnych ograniczonych kontekstach. Na przykład od wczesnych prób zbudowania „ogólnego narzędzia do rozwiązywania problemów" włożono wiele pracy w opracowanie głównie heurystycznych algorytmów zarabiania maszyn, które rozwiązują nowe problemy w oparciu o doświadczenie z poprzednimi problemami. Wiele wskazówek dotyczących uczenia się przez fragmentację, uczenia się przez makra, uczenia się hierarchicznego, uczenia się przez analogię itp. można znaleźć w książce Mitchella i ankiecie Kaelblinga. Ostatnie lata przyniosły jednak znaczny postęp w dziedzinie obliczalnych i wykonalnych wariantów optymalnych algorytmów do przewidywania, wyszukiwania, wnioskowania indukcyjnego, rozwiązywania problemów, podejmowania decyzji i uczenia się przez wzmacnianie w bardzo ogólnych środowiskach.

Bardziej formalnie

Jaki jest optymalny sposób przewidywania przyszłości, biorąc pod uwagę przeszłość? Jaki jest najlepszy sposób działania, aby zmaksymalizować przyszłą oczekiwaną nagrodę? Jaki jest najlepszy sposób poszukiwania rozwiązania nowego problemu, optymalnie wykorzystując rozwiązania wcześniejszych problemów? biorąc pod uwagę przeszłość, Solomonoff skupia się tylko na kolejnym elemencie sekwencji. Choć rodzi to zaskakująco nietrywialne problemy związane z tłumaczeniem podejścia bitowego na alfabety inne niż binarny – udało się to osiągnąć dopiero niedawno [20] – to jednak jest wystarczające do uzyskania istotnych spostrzeżeń. Biorąc pod uwagę zaobserwowany ciąg bitów x, Solomonoff zakłada, że dane są rysowane zgodnie z miarą rekurencyjną μ; oznacza to, że istnieje

program dla uniwersalnej maszyny Turinga, który odczytuje x ∈ B∗, oblicza μ(x) i zatrzymuje się. Ocenia prawdopodobieństwo następnego bitu (zakładając, że taki będzie), korzystając z niezwykłego, dobrze zbadanego, przeliczalnego wcześniejszego M

$$M(x) = \sum_{\substack{\text{program prefix } p \text{ computes} \\ \text{output starting with } x}} 2^{-l(p)}.$$

M jest uniwersalne i dominuje nad mniej ogólnymi miarami rekurencyjnymi w następujący sposób: Dla wszystkich x ∈ B∗

$$M(x) \geq c_\mu \mu(x),$$

gdzie c_μ jest stałą zależną od μ, ale nie od x. Solomonoff zauważył, że warunkowe M-prawdopodobieństwo określonej kontynuacji, biorąc pod uwagę poprzednie obserwacje, zbiega się w kierunku nieznanego warunku warunkowego μ w miarę zbliżania się rozmiaru obserwacji do nieskończoności [63] oraz że suma odpowiednich odchyleń oczekiwanych μ po wszystkich rozmiarach obserwacji wynosi faktycznie ograniczony przez stałą. Hutter (w ramach autorskiego grantu badawczego SNF „Ujednolicenie uniwersalnej indukcji i Sekwencyjna Teoria Decyzji") wykazało niedawno, że liczba błędów przewidywań popełnianych przez uniwersalną przewidywanie Solomonoffa jest w zasadzie ograniczona przez liczbę błędów popełnianych przez jakikolwiek inny predyktor, włączając w to schemat optymalny oparty na prawdziwym μ. Ostatnie granice strat dla uniwersalnej prognozy. To bardziej ogólny, niedawny wynik. Załóżmy, że wiemy, że p należy do jakiegoś zbioru P rozkładów. Wybierz stałą wagę w_q dla każdego q w P tak, aby w_q sumowało się do 1 (dla uproszczenia niech P będzie policzalne). Następnie skonstruuj Bayesmix M(x) = Σq w_qq(x) i przewiduj, używając M zamiast optymalnego, ale nieznanego p. Jak źle jest to zrobić? Niedawna praca Huttera podaje ogólne i ostre (!) granice strat . Niech LM(n) i Lp(n) będą całkowitymi oczekiwanymi stratami jednostkowymi, odpowiednio, predyktora M i predyktora p dla pierwszych n zdarzeń. Wtedy LM(n)−Lp(n) jest co najwyżej rzędu √Lp(n). Oznacza to, że M nie jest dużo gorsze od p i ogólnie rzecz biorąc, żaden inny predyktor nie jest lepszy od tego! W szczególności, jeśli p jest deterministyczne, wówczas M-predyktor wkrótce nie będzie już popełniał błędów. Jeśli P zawiera wszystkie rozkłady obliczalne rekurencyjnie, wówczas M staje się słynnym przeliczalnym uniwersalnym wcześniejszym. Oznacza to, że po dziesięcioleciach badań, w których panuje stagnacja, obecnie mamy ostre granice strat w przypadku uniwersalnego schematu indukcyjnego Solomonoffa . Podejście Solomonoffa jest jednak nieobliczalne. Aby uzyskać podejście wykonalne, zredukuj M do tego, co otrzymasz, jeśli, powiedzmy, po prostu dodasz ważone szacunkowe prawdopodobieństwa przyszłych danych finansowych wygenerowane przez 1000 pakietów oprogramowania do przewidywania kursów akcji na giełdzie komercyjnej. Jeśli tylko jeden z rozkładów prawdopodobieństwa będzie bliski prawdziwemu (ale nie wiesz który), i tak powinieneś się wzbogacić. Należy zauważyć, że podejście to jest znacznie bardziej ogólne niż to, co zwykle stosuje się w tradycyjnej teorii statystycznego uczenia się, gdzie często przyjmuje się dość nierealistyczne założenie, że obserwacje są statystycznie niezależne

Super Omega i uogólnienia złożoności Kołmogorowa i prawdopodobieństwa algorytmicznego

Nasze ostatnie badania uogólniły podejście Solomonoffa do przypadku mniej restrykcyjnych, niepoliczalnych uniwersalnych priorytetów, które nadal są obliczalne w granicy. Obiekt X jest formalnie możliwy do opisania, jeśli skończona ilość informacji całkowicie opisuje X i tylko X. Mówiąc ściślej, tylko

x w sposób, który modyfikuje każdy bit wyjściowy co najwyżej skończenie wiele razy; to znaczy każdy skończony początek x ostatecznie zbiega się i przestaje się zmieniać. To konstruktywne pojęcie formalnej opisowalności jest mniej restrykcyjne niż tradycyjne pojęcie obliczalności [67], głównie dlatego, że nie nalegamy na istnienie programu zatrzymującego, który oblicza górną granicę czasu zbieżności n-tego bitu wyjściowego p. Formalna deskrypcyjność popycha zatem konstruktywizm do skrajności, ledwo unikając niekonstruktywizmu ucieleśnionego przez jeszcze mniej restrykcyjne koncepcje opisywalności (porównaj obliczalność w granicy i Δ^0_n-deskrybowalność). Tradycyjna teoria wnioskowania indukcyjnego koncentruje się na maszynach Turinga z jednokierunkową taśmą wyjściową tylko do zapisu. Prowadzi to do uniwersalnej przeliczalnej (pół) miary Solomonoffa-Levina. Wprowadziliśmy bardziej ogólne, niepoliczalne, ale wciąż dające się obliczyć miary oraz naturalną hierarchię uogólnień prawdopodobieństwa algorytmicznego i złożoności Kołmogorowa , sugerując, że „prawdziwa" zawartość informacyjna pewnego (prawdopodobnie nieskończonego) ciągu bitowego x w rzeczywistości jest rozmiar najkrótszego, nieprzerwanego programu, który zbiega się do x i tylko x, na maszynie Turinga, która może edytować swoje poprzednie dane wyjściowe. W rzeczywistości ta „prawdziwa" treść jest często mniejsza niż tradycyjna złożoność Kołmogorowa. Pokazaliśmy, że istnieją Super Omega, które można obliczyć w limicie, ale są one bardziej losowe niż „liczba mądrości" Omega Chaitina (która jest maksymalnie losowa w słabszym tradycyjnym sensie) i że każda przybliżona miara x jest mała dla dowolnego x, którego brakuje krótki opis. Pokazaliśmy również, że istnieje uniwersalna, przeliczalna miara x oparta na miarach wszystkich przeliczalnych y leksykograficznie większych od x. Jest bardziej dominujący, ale równie łatwy do obliczenia w granicach, jak Solomonoff . Oznacza to, że jeśli interesują nas uniwersalne miary obliczalne w granicach, powinniśmy preferować nową uniwersalną miarę skumulowaną przeliczalną od tradycyjnej miary przeliczalnej. Jeśli uwzględnimy w naszej mieszance Bayesa takie rozkłady, które można obliczyć z limitem, otrzymamy ponownie ostre granice strat do przewidywania na podstawie miksu. Nasze podejście podkreśla różnice między zbiorami przeliczalnymi i niepoliczalnymi. Jakie są potencjalne konsekwencje dla fizyki? Twierdzimy, że rzeczy takie jak niepoliczony czas i przestrzeń oraz niepoliczalne prawdopodobieństwa w rzeczywistości nie powinny odgrywać roli w wyjaśnianiu świata, ze względu na brak dowodów na to, że są one naprawdę konieczne. Niektórzy mogą odczuwać pokusę, aby przeciwstawić się temu tokowi rozumowania, wskazując, że od wieków fizycy obliczali za pomocą ciągów liczb rzeczywistych, z których większość była niepoliczalna. Nawet fizycy kwantowi, którzy są gotowi porzucić założenie o ciągłym wszechświecie, zwykle przyjmują za pewnik istnienie ciągłych rozkładów prawdopodobieństwa w ich dyskretnych wszechświatach, a Stephen Hawking wyraźnie powiedział: „Chociaż istnieją , po sugestiach, że czasoprzestrzeń może mieć dyskretną strukturę, nie widzę powodu, aby porzucić teorie kontinuum, które odniosły taki sukces". Należy jednak zauważyć, że w rzeczywistości wszyscy fizycy manipulowali jedynie dyskretnymi symbolami, generując w ten sposób skończone, dające się opisać dowody swoich wyników wyprowadzone z niezliczonych aksjomatów. To, że liczby rzeczywiste naprawdę istnieją w sposób wykraczający poza skończone ciągi symboli używane przez wszystkich, może być wytworem wyobraźni — porównaj konstruktywną matematykę Brouwera i twierdzenie Lowenheima-Skolema , z którego wynika, że każda teoria pierwszego rzędu z modelem nieprzeliczalnym, takim jak liczby rzeczywiste, ma również model przeliczalny. Jak to ujął Kronecker: „Bóg stworzył liczby całkowite, cała reszta jest dziełem człowieka". Kronecker przyjął ze sceptycyzmem spostrzeżenie Cantora, które celebrowano na temat liczb rzeczywistych, obiektów matematycznych, które według Kroneckera w ogóle nie istniały. Zakładając, że nasza przyszłość należy do nielicznych (przeliczalnie wielu) możliwych do opisania przyszłości, możemy zignorować niezliczoną ilość przyszłości, których nie da się opisać, w szczególności tych przypadkowych. Dodanie stosunkowo łagodnego założenia, że rozkład prawdopodobieństwa, z którego wyprowadzono nasz Wszechświat, jest przeliczalny kumulatywnie, dostarcza teoretycznego uzasadnienia przewidywania, że najbardziej prawdopodobne kontynuacje naszych wszechświatów można obliczyć za pomocą krótkich procedur

wyliczeniowych. W tym sensie brzytwa Ockhama jest po prostu naturalnym produktem ubocznym założenia o obliczalności! Ale co z falsyfikowalnością? Pseudolosowość naszego wszechświata może być w zasadzie niewykrywalna, ponieważ nie można udowodnić, że niektóre przybliżone i wyliczalne wzorce są nielosowe w rekurencyjnie ograniczonym czasie. Jednakże w następnych sekcjach zostaną wprowadzone dodatkowe prawdopodobne założenia, które faktycznie prowadzą do obliczalnych optymalnych procedur przewidywania.

Obliczalne przewidywania za pomocą priorytetu prędkości opartego na najszybszym sposobie opisywania obiektów

Niestety, podczas gdy M i bardziej ogólne priorytety z sekcji 4 są obliczalne w granicy, nie są rekurencyjne, a zatem praktycznie niewykonalne. Ta wada zainspirowała mniej ogólne, ale praktycznie bardziej wykonalne zasady minimalnej długości opisu (MDL, a także priorytety wyprowadzone z ograniczeń czasowych złożoności Kolmogorowa . Jednak żaden konkretny przypadek tych podejść nie jest powszechnie akceptowany ani nie ma ogólnej przekonującej motywacji wykraczającej poza raczej wyspecjalizowane scenariusze zastosowań. Na przykład typowe wydajne podejścia MDL wymagają specyfikacji klasy obliczalnych modeli danych, powiedzmy, pewnych typów sieci neuronowych, plus pewnej obliczalnej funkcji straty wyrażającej koszty kodowania danych w stosunku do modelu. To prowokuje liczne wybory ad hoc. Nasza niedawna praca [54] oferuje jednak alternatywę dla osławionej, ale nieobliczalnej miary prostoty algorytmicznej lub miary Solomonoffa-Levina omawianej powyżej . Wprowadziliśmy nową miarę (a priori dotyczącą obiektów obliczalnych), która nie opiera się na najkrótszym, ale na najszybszym sposobie opisywania obiektów. Załóżmy, że obserwowana sekwencja danych jest generowana przez proces obliczeniowy i że każda możliwa sekwencja obserwacji jest zatem obliczalna w granicy [50]. To założenie jest silniejsze i bardziej radykalne niż tradycyjne: Solomonoff po prostu nalega, że prawdopodobieństwo dowolnego prefiksu sekwencji jest rekurencyjnie obliczalne, ale sama (nieskończona) sekwencja może być nadal generowana probabilistycznie. Biorąc pod uwagę nasze początkowe założenie, że dane są deterministycznie generowane przez maszynę, wydaje się prawdopodobne, że maszyna cierpi na problem zasobów obliczeniowych. Ponieważ niektóre rzeczy są znacznie trudniejsze do obliczenia niż inne, punkt widzenia zorientowany na zasoby sugeruje następujący postulat.

Postulat 1 Skumulowana miara prawdopodobieństwa a priori wszystkich x nieobliczalnych w czasie t dowolną metodą jest co najwyżej odwrotnie proporcjonalna do t. Ten postulat prowadzi do Speed Prior S(x), prawdopodobieństwa, że wyjście następującego algorytmu probabilistycznego zaczyna się od x :

Inicjalizacja: Ustaw t := 1. Niech głowica skanująca wejściowa uniwersalnej TM wskazuje na pierwszą komórkę początkowo pustej taśmy wejściowej. Powtarzaj w nieskończoność: Dopóki liczba wykonanych dotychczas instrukcji przekracza t: rzuć nieobciążoną monetą; jeśli orzeł jest w górze, ustaw t := 2t; w przeciwnym razie wyjdź. Jeśli głowica skanująca wejściowa wskazuje na komórkę, która już zawiera bit, wykonaj odpowiednią instrukcję (rosnącego programu samoograniczającego, np. [30, 31]). W przeciwnym razie rzuć monetą ponownie, ustaw bit komórki na 1, jeśli orzeł jest w górze (w przeciwnym wypadku 0), i ustaw t := t/2.

Algorytm GUESS jest bardzo podobny do algorytmu wyszukiwania probabilistycznego używanego w poprzednich pracach nad indukcyjnym wnioskowaniem stosowanym. W przypadku kilku problemów zabawowych uogólniał się on niezwykle dobrze w sposób nieporównywalny z tradycyjnymi algorytmami uczenia się sieci neuronowych. Z S pojawia się obliczalna metoda AS do przewidywania optymalnie z dokładnością [54]. Rozważmy skończony, ale nieznany program p obliczający $y \in B^{\infty}$. Co

się stanie, jeśli postulat 1 jest spełniony, ale p nie jest optymalnie wydajne i/lub obliczone na komputerze, który różni się od naszej maszyny odniesienia? Wtedy skutecznie nie pobieramy próbek początków y_k z S, ale z alternatywnej półmiary S' . Czy nadal możemy dobrze przewidywać? Tak, ponieważ priorytet prędkości S dominuje nad S' . Ta dominacja to wszystko, czego potrzebujemy, aby zastosować ostatnie granice strat. Strata, którą spodziewamy się otrzymać, przewidując zgodnie z AS zamiast używać prawdziwego, ale nieznanego S , nie przekracza optymalnej straty o wiele .

Szybkie prognozy oparte na priorytecie dla naszego wszechświata

„Na początku był kod".

Pierwsze zdanie Biblii Wielkiego Programisty. Fizycy, ekonomiści i inni naukowcy indukcyjni formułują prognozy na podstawie obserwacji. Zaskakująco jednak niewielu fizyków jest świadomych teorii optymalnego wnioskowania indukcyjnego . W rzeczywistości, gdy mowa o samej naturze ich indukcyjnego biznesu, wielu fizyków cytuje dość niejasne koncepcje, takie jak falsyfikowalność Poppera , zamiast odwoływać się do ilościowych wyników. Wszystkie powszechnie akceptowane teorie fizyczne są jednak akceptowane nie dlatego, że są falsyfikowalne — nie są — lub dlatego, że pasują do danych — wiele alternatywnych teorii również pasuje do danych — ale dlatego, że są proste w pewnym sensie. Na przykład teoria grawitacji jest indukowana z lokalnie obserwowalnych przykładów szkoleniowych, takich jak spadające jabłka i ruchy odległych źródeł światła, prawdopodobnie gwiazd. Teoria przewiduje, że jabłka na odległych planetach w innych galaktykach również spadną. Obecnie nikt nie jest w stanie tego zweryfikować ani sfalsyfikować. Ale wszyscy w to wierzą, ponieważ ten krok uogólnienia sprawia, że teoria jest prostsza niż alternatywne teorie z odrębnymi prawami dla jabłek na innych planetach. To samo dotyczy teorii superstrun lub teorii wielu światów Everetta , które obecnie również nie są ani weryfikowalne, ani falsyfikowalne, ale oferują stosunkowo proste wyjaśnienia licznych obserwacji. W szczególności większość postulowanych przez Everetta wielu światów pozostanie nieobserwowalna na zawsze, ale założenie ich istnienia upraszcza teorię, czyniąc ją tym samym piękniejszą i bardziej akceptowalną. W sekcjach 3 i 4 przyjęliśmy założenie, że prawdopodobieństwa następnych zdarzeń, biorąc pod uwagę poprzednie zdarzenia, są (granicznie) obliczalne. Tutaj przyjmujemy silniejsze założenie, przyjmując tezę Zuse'a, a mianowicie, że sam wszechświat jest faktycznie obliczany deterministycznie, np. na automacie komórkowym (CA) . Fizyka kwantowa, obliczenia kwantowe, zasada nieoznaczoności Heisenberga i nierówność Bella nie implikują żadnych fizycznych dowodów przeciwko tej możliwości. Ale jaki jest dokładny algorytm naszego wszechświata?

Systematycznie twórz i wykonuj wszystkie programy dla komputera uniwersalnego, takiego jak maszyna Turinga lub CA; pierwszy program jest uruchamiany średnio co drugi krok dla jednej instrukcji, następny średnio co sekundę dla pozostałych kroków itd.

Ta metoda w pewnym sensie implementuje najprostszą teorię wszystkiego: wszystkie obliczalne wszechświaty, w tym nasz i nas samych jako obserwatorów, są obliczane przez bardzo krótki program, który generuje i wykonuje wszystkie możliwe programy. W sposób zagnieżdżony niektóre z tych programów będą wykonywać procesy, które ponownie obliczają wszystkie możliwe wszechświaty itd. . Oczywiście obserwatorzy we wszechświatach „wyższego poziomu" mogą być zupełnie nieświadomi obserwatorów lub wszechświatów obliczonych przez zagnieżdżone procesy i odwrotnie. Na przykład wydaje się, że trudno jest śledzić i interpretować obliczenia wykonywane przez filiżankę herbaty. Prosta metoda powyżej jest bardziej wydajna, niż może się wydawać na pierwszy rzut oka. Odrobina namysłu pokazuje, że ma ona nawet optymalny rząd złożoności. Na przykład wyprowadza historię naszego wszechświata tak szybko, jak najszybszy program tej historii, z wyjątkiem (być może ogromnego) stałego współczynnika spowolnienia, który nie zależy od rozmiaru wyjścia. Niemniej

jednak niektóre wszechświaty są zasadniczo trudniejsze do obliczenia niż inne. Odzwierciedla to omawiany powyżej Speed Prior S (sekcja 5). Załóżmy więc, że historia naszego wszechświata została pobrana z S lub mniej dominującego wcześniejszego okresu odzwierciedlającego suboptymalne obliczenia historii. Teraz możemy natychmiast przewidzieć:

1. Nasz wszechświat nie będzie wiele razy starszy niż jest teraz - zasadniczo prawdopodobieństwo, że będzie trwał 2n razy dłużej niż do tej pory, wynosi co najwyżej 2−n.

2. Każda pozorna losowość w dowolnej obserwacji fizycznej musi być spowodowana jakimś nieznanym, ale szybkim pseudolosowym generatorem PRG , który powinniśmy spróbować odkryć.

2a. Ponowne zbadanie wzorców rozpadu beta może ujawnić, że bardzo prosty, szybki, ale być może nie do końca trywialny PRG jest odpowiedzialny za pozornie losowe rozpady neutronów na protony, elektrony i antyneutrina.

2b. Zawsze, gdy istnieje kilka możliwych kontynuacji naszego wszechświata odpowiadających różnym kolapsom funkcji falowej Schrödingera — porównaj powszechnie akceptowaną teorię wielu światów Everetta - powinniśmy mieć większe prawdopodobieństwo, że skończymy w jednym obliczalnym przez krótki i szybki algorytm. Ponowne zbadanie danych z podzielonego eksperymentu obejmujących stany splątane, takie jak obserwacje spinów początkowo bliskich, ale wkrótce odległych cząstek ze skorelowanymi spinami, może ujawnić nieoczekiwaną, nieoczywistą, nielokalną regularność algorytmiczną z powodu szybkiego PRG.

3. Obliczenia kwantowe na dużą skalę nie będą działać dobrze, ponieważ wymagałyby zbyt wielu wykładniczo rosnących zasobów obliczeniowych w zakłócających „równoległych wszechświatach".

4. Żaden algorytm probabilistyczny zależny od prawdziwie losowych danych wejściowych ze środowiska nie będzie dobrze skalowalny w praktyce. Przewidywanie 2 jest weryfikowalne, ale niekoniecznie falsyfikowalne w ustalonym przedziale czasowym podanym z góry. Mimo to, być może głównym powodem obecnego braku dowodów empirycznych w tym duchu jest to, że niewielu ich szukało. W ostatnich dekadach kilku znanych fizyków zaczęło pisać o tematach informatyki, czasami sugerując, że fizyka świata rzeczywistego może umożliwiać obliczanie rzeczy, które tradycyjnie nie są obliczalne. Niezrażeni tym trendem, informatycy argumentowali na rzecz przeciwnego założenia: ponieważ nie ma dowodów na to, że potrzebujemy czegoś więcej niż tradycyjnej obliczalności, aby wyjaśnić świat, powinniśmy spróbować obejść się bez tego założenia.

Optymalni racjonalni decydenci

Do tej pory mówiliśmy o biernym przewidywaniu, biorąc pod uwagę obserwacje. Należy jednak zauważyć, że agenci wchodzący w interakcje ze środowiskiem mogą również używać przewidywań przyszłości do obliczania sekwencji działań, które maksymalizują oczekiwaną przyszłą nagrodę. Najnowszy model AIXI Huttera robi dokładnie to, łącząc oparty na M uniwersalny schemat przewidywania Solomonoffa z obliczeniem expectimax. W cyklu t działanie y_t skutkuje percepcją xt i nagrodą r_t, gdzie wszystkie wielkości mogą zależeć od całej historii. Percepcja x t i nagroda r_t są próbkowane z (reaktywnego) rozkładu prawdopodobieństwa środowiskowego μ. Sekwencyjna teoria decyzji pokazuje, jak zmaksymalizować całkowitą oczekiwaną nagrodę, zwaną wartością, jeśli μ jest znane. Uczenie się przez wzmacnianie jest stosowane, jeśli μ jest nieznane. AIXI definiuje rozkład mieszaniny ξ jako ważoną sumę rozkładów $\nu \in M$, gdzie M jest dowolną klasą rozkładów obejmującą prawdziwe środowisko μ. Można wykazać, że warunkoweM prawdopodobieństwo danych wejściowych ze środowiska dla agenta AIXI, biorąc pod uwagę wcześniejsze dane wejściowe i działania agenta, zbiega się ze wzrastającą długością interakcji względem prawdziwego, nieznanego

prawdopodobieństwa, o ile to ostatnie jest rekurencyjnie obliczalne, analogicznie do przypadku biernej predykcji. Ostatnie prace wykazały również optymalność AIXI w następującym sensie. Optymalna polityka Bayesa p^{ξ} oparta na mieszance ξ jest samooptymalizująca w tym sensie, że średnia wartość zbiega się asymptotycznie dla wszystkich $\mu \in M$ do optymalnej wartości osiągniętej przez (niewykonalną) optymalną politykę Bayesa p_{μ}, która zna μ z góry. Wystarczający jest również warunek konieczny, aby M dopuszczała polityki samooptymalizujące. Nie poczyniono żadnych innych założeń strukturalnych w odniesieniu do M. Co więcej, p^{ξ} jest Pareto-optymalne w tym sensie, że nie ma innej polityki dającej wyższą lub równą wartość we wszystkich środowiskach $\nu \in M$ i ściśle wyższą wartość w co najmniej jednym . Możemy zmodyfikować model AIXI tak, aby jego przewidywania opierały się na -przybliżonej prędkości wcześniejszej S zamiast na nieobliczalnym M. W ten sposób otrzymujemy tzw. model AIS. Stosując podejście Huttera możemy teraz pokazać, że warunkowe prawdopodobieństwo S danych wejściowych ze środowiska dla agenta AIS, biorąc pod uwagę wcześniejsze dane wejściowe i działania, zbiega się do prawdziwego, ale nieznanego prawdopodobieństwa, o ile to ostatnie jest zdominowane przez S, takie jak S powyżej.

Optymalne uniwersalne algorytmy wyszukiwania

W pewnym sensie wyszukiwanie jest mniej ogólne niż uczenie się przez wzmacnianie, ponieważ niekoniecznie obejmuje przewidywania niewidzianych danych. Mimo to wyszukiwanie jest centralnym aspektem informatyki (a każdy uczący się przez wzmacnianie potrzebuje wyszukiwarki jako podmodułu). Zaskakująco jednak wiele książek o algorytmach wyszukiwania nie wspomina nawet o następującym, bardzo prostym asymptotycznie optymalnym, „uniwersalnym” algorytmie dla szerokiej klasy problemów wyszukiwania.

Zdefiniuj rozkład prawdopodobieństwa P na skończonym lub nieskończonym zestawie programów dla danego komputera. P reprezentuje początkowe odchylenie wyszukiwarki (np. P może być oparte na długości programu lub na diagramie składni probabilistycznej).

Metoda Lsearch: Ustaw bieżący limit czasu T=1. Podczas gdy problem nie został rozwiązany, wykonaj:

Przetestuj wszystkie programy q tak, aby t(q), maksymalny czas spędzony na tworzeniu, uruchamianiu i testowaniu q, spełniał $t(q) < P(q)\,T$.

Ustaw T := 2T.

Lsearch (dla Levin Search) może być algorytmem, do którego Levin odnosił się w swojej dwustronicowej pracy , w której stwierdza, że istnieje asymptotycznie optymalna uniwersalna metoda wyszukiwania dla problemów z łatwo weryfikowalnymi rozwiązaniami, to znaczy rozwiązaniami, których ważność można szybko przetestować. Biorąc pod uwagę pewną klasę problemów, jeśli pewien nieznany optymalny program p wymaga f(k) kroków do rozwiązania instancji problemu o rozmiarze k, to Lsearch będzie potrzebował co najwyżej $O(f(k)/P(p)) = O(f(k))$ kroków — stały czynnik $1/P(p)$ może być ogromny, ale nie zależy od k. Hutter opracował bardziej złożony asymptotycznie optymalny algorytm wyszukiwania dla wszystkich dobrze zdefiniowanych problemów, nie tylko tych z łatwo weryfikowalnymi rozwiązaniami. Hsearch sprytnie przydziela część całkowitego czasu wyszukiwania na przeszukiwanie przestrzeni dowodów w celu znalezienia programów kandydackich o udowodnionych poprawnych wynikach z udowodnionych górnych granic czasu wykonania i w dowolnym momencie koncentruje zasoby na tych programach z obecnie najlepszymi udowodnionymi granicami czasu. Nieoczekiwanie Hsearch udaje się zredukować nieznany stały współczynnik spowolnienia Lsearch do

wartości 1 + ε , gdzie ε jest dowolną dodatnią stałą. Niestety jednak wyszukiwanie w przestrzeni dowodów wprowadza nieznany problem addytywny, klasę specyficznego stałego spowolnienia, która znowu może być ogromna. Podczas gdy stałe addytywne są ogólnie preferowane od stałych mnożnikowych, oba typy mogą sprawić, że uniwersalne metody wyszukiwania staną się praktycznie niewykonalne. Hsearch i Lsearch są nieinkrementalne w tym sensie, że nie próbują minimalizować swoich stałych, wykorzystując doświadczenie zebrane w poprzednich wyszukiwaniach. Nasza metoda Adaptive Lsearch lub Als próbuje to przezwyciężyć - porównaj powiązane pomysły Solomonoffa. Zasadniczo działa ona w następujący sposób: kiedykolwiek wyszukiwanie znajdzie program q, który oblicza rozwiązanie bieżącego problemu, prawdopodobieństwo q P(q) jest znacząco zwiększane przy użyciu „wskaźnika uczenia się", podczas gdy prawdopodobieństwa alternatywnych programów odpowiednio maleją. Następne Lsearches dla nowych problemów używają następnie skorygowanego P itd. Nieuniwersalna odmiana tego podejścia była w stanie rozwiązać zadania uczenia się przez wzmacnianie (RL) w częściowo obserwowalnych środowiskach nierozwiązywalnych przez tradycyjne algorytmy RL . Każde Lsearch wywołane przez Als jest optymalne w odniesieniu do ostatniej korekty P. Z drugiej strony, modyfikacje samego P niekoniecznie są optymalne. Ostatnie prace omówione w następnej sekcji przezwyciężają tę wadę w sposób zasadniczy.

Optymalny uporządkowany rozwiązywacz problemów (OOPS)

Nasz ostatni oops to prosty, ogólny, teoretycznie poprawny, w pewnym sensie optymalny czasowo sposób poszukiwania uniwersalnego zachowania lub programu, który rozwiązuje każdy problem w sekwencji problemów obliczeniowych, nieustannie organizując, zarządzając i ponownie wykorzystując wcześniej zdobytą wiedzę. Na przykład n-ty problem może polegać na obliczeniu n-tego zdarzenia z poprzednich zdarzeń (przewidywanie) lub znalezieniu szybszej drogi przez labirynt niż ta znaleziona podczas poszukiwania rozwiązania n − 1-tego problemu (optymalizacja). Najpierw wprowadźmy ważną koncepcję optymalności stronniczej, która jest pragmatyczną definicją optymalności czasowej, w przeciwieństwie do asymptotycznej optymalności zarówno Lsearch, jak i Hsearch, które można postrzegać jako ćwiczenia akademickie pokazujące, że notacja O() może czasami być praktycznie nieistotna pomimo jej szerokiego zastosowania w teoretycznej informatyce. W przeciwieństwie do asymptotycznej optymalności, optymalność stronniczości nie ignoruje ogromnych stałych spowolnień: Definicja 1 (Poszukiwacze optymalni pod względem stronniczości). Podana jest klasa problemu R, przestrzeń poszukiwań C kandydatów na rozwiązania (gdzie każdy problem r ∈ R powinien mieć rozwiązanie w C), zależne od zadania stronniczość w postaci rozkładów prawdopodobieństwa warunkowego P(q | r) na kandydatach q ∈ C oraz wstępnie zdefiniowana procedura, która tworzy i testuje dowolne dane q na dowolnym r ∈ R w czasie t(q, r) (zwykle nieznanym z góry). Poszukiwacz jest n-optymalny pod względem stronniczości (n ≥ 1), jeśli dla dowolnego maksymalnego całkowitego czasu wyszukiwania Tmax > 0 gwarantuje rozwiązanie dowolnego problemu r ∈ R, jeśli ma rozwiązanie p ∈ C spełniające t(p, r) ≤ P(p | r) Tmax/n. Jest to optymalne pod względem stronniczości, jeśli n = 1. Ta definicja ma intuicyjny sens: najbardziej prawdopodobni kandydaci powinni uzyskać lwią część całkowitego czasu wyszukiwania, w sposób, który dokładnie odzwierciedla początkowe stronniczość. Teraz jesteśmy gotowi przedstawić ogólny przegląd podstawowych składników oops:

Podstawy: Zaczynamy od początkowego zestawu zdefiniowanych przez użytkownika prymitywnych zachowań. Prymitywy mogą być instrukcjami podobnymi do asemblera lub czasochłonnym oprogramowaniem, takim jak na przykład dowodzące twierdzeń, operatorami macierzy dla równoległych architektur podobnych do sieci neuronowych lub generatorami trajektorii dla symulacji robotów lub procedurami aktualizacji stanu dla systemów wieloagentowych itp. Każdy prymityw jest reprezentowany przez token. Istotne jest, aby te prymitywy, których czasy wykonania nie są znane z góry, mogły zostać przerwane w dowolnym momencie. Kody prefiksów specyficzne dla zadania:

Złożone zachowania są reprezentowane przez sekwencje tokenów lub programy. Aby rozwiązać dane zadanie reprezentowane przez specyficzne dla zadania dane wejściowe programu, oops próbuje sekwencyjnie ułożyć odpowiednie złożone zachowanie z prymitywnych, zawsze przestrzegając reguł danego, zdefiniowanego przez użytkownika, początkowego języka programowania. Programy są rozwijane przyrostowo, token po tokenie; ich początki lub prefiksy są natychmiast wykonywane podczas tworzenia; może to modyfikować pewien specyficzny dla zadania stan wewnętrzny lub pamięć i może przenieść kontrolę z powrotem do wcześniej wybranych tokenów (np. pętli). Aby dodać nowy token do pewnego prefiksu programu, musimy najpierw poczekać, aż wykonanie prefiksu do tej pory wyraźnie zażąda takiego przedłużenia, ustawiając odpowiedni sygnał w stanie wewnętrznym. Prefiksy, które przestają żądać dalszych tokenów, nazywane są programami samoograniczającymi się lub po prostu programami (programy są swoimi własnymi prefiksami). Binarne programy samoograniczające się były badane w kontekście maszyn Turinga oraz teorii złożoności Kołmogorowa i prawdopodobieństwa algorytmicznego. Oops używa jednak bardziej praktycznego, niekoniecznie binarnego frameworka. Powyższa procedura konstrukcji programu generuje kody prefiksów specyficzne dla zadania w przestrzeni programu: w przypadku dowolnego zadania programy, które zatrzymują się, ponieważ znalazły rozwiązanie lub napotkały jakiś błąd, nie mogą żądać więcej tokenów. Biorąc pod uwagę bieżące dane wejściowe specyficzne dla zadania, żaden program nie może być prefiksem innego. Jednak w przypadku innego zadania ten sam program może nadal żądać dodatkowych tokenów. Jest to ważne dla naszego nowatorskiego podejścia — stopniowo rosnące programy samoograniczające są zbędne dla asymptotycznych właściwości optymalności Lsearch i Hsearch, ale niezbędne dla oops.

Dostęp do poprzednich rozwiązań: Niech p^n oznacza znaleziony prefiks rozwiązujący pierwsze n zadań. Poszukiwanie p^{n+1} może w znacznym stopniu skorzystać z informacji przekazywanych przez (lub wiedzy ucieleśnionej przez) p^1, p^2, . . . , p^n, które są przechowywane lub zamrożone w specjalnej niemodyfikowalnej pamięci współdzielonej przez wszystkie zadania, tak że są dostępne dla p^{n+1} (to kolejna różnica w stosunku do nieinkrementalnych Lsearch i Hsearch). Na przykład, p^{n+1} może wykonać sekwencję tokenów, która wywołuje p^{n-3} jako podprogram lub kopiuje p^{n-17} do pewnej wewnętrznej modyfikowalnej pamięci specyficznej dla zadania, a następnie modyfikuje kopię nieco, a następnie stosuje nieznacznie edytowaną kopię do bieżącego zadania. W rzeczywistości, ponieważ liczba zamrożonych programów może wzrosnąć do dużej wartości, większość wiedzy ucieleśnionej przez p^j może dotyczyć sposobu dostępu, edycji i używania starszego p^i ($i < j$).

Błąd: Początkowe błędy wyszukiwania są ucieleśnione przez początkowe, zdefiniowane przez użytkownika, zależne od zadania rozkłady prawdopodobieństwa w skończonej lub nieskończonej przestrzeni wyszukiwania możliwych prefiksów programu. W najprostszym przypadku zaczynamy od maksymalnego rozkładu entropii na tokenach i definiujemy prawdopodobieństwa prefiksów jako iloczyny prawdopodobieństw ich tokenów. Jednak prawdopodobieństwa kontynuacji prefiksów mogą również zależeć od poprzednich tokenów w sposób zależny od kontekstu. Samodzielnie obliczone prawdopodobieństwa sufiksów: W rzeczywistości zezwalamy, aby każdy wykonany prefiks przypisywał zależny od zadania, samodzielnie obliczony rozkład prawdopodobieństwa do własnych możliwych kontynuacji. Ten rozkład jest kodowany i manipulowany w pamięci wewnętrznej specyficznej dla zadania. Tak więc, w przeciwieństwie do Als , nie używamy wstępnie skonfigurowanego schematu uczenia się do aktualizacji rozkładu prawdopodobieństwa. Zamiast tego pozostawiamy takie aktualizacje prefiksom, których wykonanie online modyfikuje prawdopodobieństwa ich sufiksów. Na przykład poprzez wywołanie wcześniej zamrożonego kodu, który redefiniuje rozkład prawdopodobieństwa przyszłych kontynuacji prefiksu, obecnie testowany prefiks może całkowicie zmienić najbardziej prawdopodobne ścieżki w przestrzeni wyszukiwania własnych kontynuacji, w oparciu o doświadczenie ignorowane przez nieinkrementalne Lsearch i Hsearch. Może to wprowadzić

znaczącą wiedzę specyficzną dla klasy problemu, pochodzącą z rozwiązań wcześniejszych zadań. Dwa wyszukiwania: Zasadniczo oops zapewnia równe zasoby dla dwóch wyszukiwań prawie optymalnych, które są uruchamiane równolegle, dopóki pn+1 nie zostanie odkryte i zapisane w pamięci niemodyfikowalnej. Pierwsze jest wyczerpujące; systematycznie testuje wszystkie możliwe prefiksy we wszystkich zadaniach do n+1. Alternatywne prefiksy są testowane równolegle we wszystkich bieżących zadaniach, podczas gdy nadal rosną; gdy zadanie zostanie rozwiązane, usuwamy je z bieżącego zestawu; prefiksy, które nie powiodą się w jednym zadaniu, są odrzucane. Drugie wyszukiwanie jest znacznie bardziej ukierunkowane; wyszukuje tylko prefiksy zaczynające się od p^n i testuje je tylko w zadaniu n+1, co jest bezpieczne, ponieważ wiemy już, że takie prefiksy rozwiązują wszystkie zadania do n.

Optymalne cofanie się: Hsearch i Lsearch zakładają potencjalnie nieskończoną pamięć. Dlatego mogą w dużej mierze ignorować kwestie zarządzania pamięcią masową. Jednak w każdym praktycznym systemie musimy efektywnie ponownie wykorzystać ograniczoną pamięć masową. Dlatego w obu wyszukiwaniach oops alternatywne kontynuacje prefiksów są oceniane przez nową, praktyczną, zorientowaną na tokeny procedurę cofania się, która może obsługiwać kilka zadań równolegle, biorąc pod uwagę pewne odchylenie kodu w postaci wcześniej znalezionego kodu. Procedura zawsze zapewnia niemal optymalną stronniczość: żadne zachowanie kandydata nie otrzymuje więcej czasu, niż na to zasługuje, biorąc pod uwagę odchylenie probabilistyczne. Zasadniczo przeprowadzamy przeszukiwanie w głąb w przestrzeni programu, gdzie gałęzie drzewa wyszukiwania są prefiksami programu, a cofanie się (częściowe resetowanie częściowo rozwiązanych zestawów zadań i modyfikacje stanów wewnętrznych i prawdopodobieństw kontynuacji) jest wyzwalane, gdy suma czasów wykonania bieżącego prefiksu dla wszystkich bieżących zadań przekroczy prawdopodobieństwo prefiksu pomnożone przez całkowity czas wyszukiwania do tej pory. W przypadku nieznanych, nieskończonych sekwencji zadań zazwyczaj nigdy nie możemy wiedzieć, czy znaleźliśmy już optymalnego rozwiązywacza dla wszystkich zadań w sekwencji. Ale gdy nieświadomie go znajdziemy, co najwyżej połowa całkowitego przyszłego czasu wykonania zostanie zmarnowana na poszukiwanie alternatyw. Biorąc pod uwagę początkowe odchylenie i późniejsze przesunięcia odchylenia z powodu p^1, p^2, . . . , żaden inny poszukiwacz optymalny pod względem odchylenia nie może oczekiwać rozwiązania n + 1-tego zestawu zadań znacznie szybciej niż ups. Produktem ubocznym tej własności optymalności jest to, że daje nam ona naturalny i precyzyjny pomiar stronniczości i przesunięć stronniczości, koncepcyjnie powiązany z rozmiarem skoku koncepcyjnego Solomonoffa. Ponieważ nie ma zasadniczej różnicy między programami rozwiązywania problemów specyficznych dla domeny a programami, które manipulują rozkładami prawdopodobieństwa, a tym samym zasadniczo przepisują samą procedurę wyszukiwania, łączymy zarówno uczenie się, jak i metauczenie w tym samym optymalnym czasowo systemie. Przykład języka początkowego. W przypadku przykładowej aplikacji napisaliśmy interpreter dla uniwersalnego języka programowania opartego na stosie, zainspirowanego przez Fortha , z początkowymi prymitywami do definiowania i wywoływania funkcji rekurencyjnych, pętli iteracyjnych, operacji arytmetycznych i zachowań specyficznych dla domeny. Optymalne metawyszukiwanie w celu uzyskania lepszych algorytmów wyszukiwania jest możliwe dzięki uwzględnieniu instrukcji przesuwających stronniczość, które mogą modyfikować prawdopodobieństwa warunkowe przyszłych opcji wyszukiwania w aktualnie uruchomionych prefiksach programu. Eksperymenty. Używając języka podobnego do assemblera, o którym mowa powyżej, najpierw uczymy oops czegoś o rekurencji, trenując ją do konstruowania próbek prostego języka bezkontekstowego $\{1^k2^k\}$ (k 1, po których następuje k 2), dla k do 30 (w rzeczywistości system odkrywa uniwersalny rozwiązywacz dla wszystkich k). Zajmuje to około 0,3 dnia na standardowym komputerze osobistym (PC). Następnie, w ciągu kilku dodatkowych dni, oops demonstruje przyrostowy transfer wiedzy: wykorzystuje aspekty swojego wcześniej odkrytego uniwersalnego

rozwiązywacza 1^k2^k, przepisując swoją procedurę wyszukiwania tak, aby łatwiej odkrywała uniwersalnego rozwiązywacza dla wszystkich problemów k dysków Wież Hanoi — w eksperymentach rozwiązuje wszystkie wystąpienia do k = 30 (rozmiar rozwiązania 2^k-1), ale działałoby również dla k > 30. Poprzednie, mniej ogólne uczące się przez wzmacnianie i nieuczące się planujące AI mają tendencję do zawodzenia w przypadku znacznie mniejszych wystąpień. Przyszłe badania mogą skupić się na opracowaniu szczególnie zwartych, szczególnie rozsądnych zestawów początkowych kodów o szczególnie szerokim zastosowaniu praktycznym. Może się okazać, że najbardziej użyteczne języki początkowe nie są tradycyjnymi językami programowania podobnymi do języka Forth, ale opierają się na garstce prymitywnych instrukcji dla masowo równoległych automatów komórkowych lub na kilku nieliniowych operacjach na macierzowych strukturach danych, takich jak te używane w badaniach nad rekurencyjnymi sieciami neuronowymi . Na przykład moglibyśmy użyć zasad oops do stworzenia nieopartej na gradientach, optymalnej pod względem nearbias wariantu udanego rekurencyjnego metalearnera sieciowego Hochreitera. Powinno być również interesujące zbadanie probabilistycznych wariantów oops opartych na Speed Prior i opracowanie zastosowań metod podobnych do oops jako składników uniwersalnych uczących się przez wzmacnianie. W trwającej pracy stosujemy oops do problemu optymalnego planowania trajektorii dla robotyki w realistycznej symulacji fizycznej. Wiąże się to z interesującym kompromisem między stosunkowo szybkimi prymitywami tworzenia programów lub „prymitywami myślenia" a czasochłonnymi „prymitywami działania", takimi jak rozciąganie ramienia do momentu wejścia czujnika dotykowego.

Uczenie się przez wzmacnianie oparte na OOPS

W dowolnym momencie uczący się przez wzmacnianie będzie próbował znaleźć politykę (strategię przyszłego podejmowania decyzji), która maksymalizuje jego oczekiwaną przyszłą nagrodę. W wielu tradycyjnych zastosowaniach uczenia się przez wzmacnianie (RL) polityka, która działa najlepiej w danym zestawie prób szkoleniowych, będzie również optymalna w przyszłych próbach testowych . Czasami jednak tak nie będzie. Aby zobaczyć różnicę między wyszukiwaniem (temat poprzednich sekcji) a uczeniem się przez wzmacnianie (RL), rozważmy agenta i dwa pudełka. W n-tej próbie agent może otworzyć i zebrać zawartość dokładnie jednego pudełka. Lewe pudełko będzie zawierało 100n franków szwajcarskich, prawe pudełko 2^n franków szwajcarskich, ale agent nie wie o tym z góry. Podczas pierwszych 9 prób optymalną polityką jest „otwórz lewe pudełko". To jest to, co dobry poszukiwacz powinien znaleźć, biorąc pod uwagę wyniki pierwszych 9 prób. Ale ta polityka będzie suboptymalna w próbie 10. Dobry uczący się przez wzmacnianie powinien jednak wyodrębnić podstawową regularność w procesie generowania nagród i przewidzieć przyszłe zadania i nagrody, wybierając właściwe pole w próbie 10, bez jego wcześniejszego zobaczenia. Pierwszym ogólnym, asymptotycznie optymalnym uczącym się przez wzmacnianie jest najnowszy model AIXI (sekcja 7). Jest on ważny dla bardzo szerokiej klasy środowisk, których reakcje na sekwencje działań (sygnały sterujące) są próbkowane z dowolnych obliczalnych rozkładów prawdopodobieństwa. Oznacza to, że AIXI jest o wiele bardziej ogólny niż tradycyjne podejścia RL. Jednak podczas gdy AIXI wyjaśnia teoretyczne ograniczenia RL, nie jest praktycznie wykonalny, tak jak Hsearch nie jest. Z pragmatycznego punktu widzenia jesteśmy naprawdę zainteresowani uczącym się przez wzmacnianie, który optymalnie wykorzystuje dane, ograniczone zasoby obliczeniowe. Poniżej przedstawiono jeden sposób wykorzystania metod oops-like bias-optimal jako składników ogólnych, ale wykonalnych uczących się przez wzmacnianie. Potrzebujemy dwóch modułów oops. Pierwszy nazywany jest predyktorem lub modelem świata. Drugi to wyszukiwarka akcji wykorzystująca model świata. Życie całego systemu powinno składać się z sekwencji cykli 1, 2, ... W każdym cyklu, ograniczona ilość czasu obliczeniowego będzie dostępna dla każdego modułu. Dla uproszczenia zakładamy, że podczas każdego cyklu system może podjąć dokładnie jedną akcję. Uogólnienia na akcje pochłaniające kilka cykli są jednak proste. W dowolnym danym cyklu, system wykonuje następującą procedurę:

1. W ustalonym z góry przedziale czasowym predyktor jest najpierw trenowany w sposób stronniczo optymalny, aby znaleźć lepszy model świata, czyli program, który przewiduje dane wejściowe ze środowiska (w tym nagrody, jeśli takie istnieją), biorąc pod uwagę historię poprzednich obserwacji i działań. Tak więc n-te zadanie (n = 1, 2, . . .) pierwszego modułu oops polega na znalezieniu (jeśli to możliwe) lepszego predyktora niż najlepszy znaleziony do tej pory.

2. Po zakończeniu bieżącego cyklu czasu na udoskonalenie predyktora bieżący model świata (program predykcyjny) znaleziony przez pierwszy moduł oops zostanie użyty przez drugi moduł, również w sposób stronniczo optymalny, aby wyszukać przyszłą sekwencję działań, która maksymalizuje przewidywaną skumulowaną nagrodę (do pewnego limitu czasu). Oznacza to, że n-te zadanie (n = 1, 2, . . .) drugiego modułu oops będzie polegało na znalezieniu programu sterującego, który oblicza sekwencję działań sterujących, które mają zostać wprowadzone do programu reprezentującego bieżący model świata (którego przewidywania wejściowe są sukcesywnie przekazywane z powrotem do samego siebie w oczywisty sposób), tak aby ta sekwencja sterująca prowadziła do wyższej przewidywanej nagrody niż ta wygenerowana przez najlepszy program sterujący znaleziony do tej pory.

3. Po zakończeniu czasu bieżącego cyklu na wyszukiwanie programu sterującego wykonamy bieżącą akcję najlepszego programu sterującego znalezionego w kroku 2. Teraz jesteśmy gotowi na kolejny cykl.

Podejście to przypomina wcześniejsze, heurystyczne, nieoptymalne pod względem stronniczości podejście RL oparte na dwóch adaptacyjnych rekurencyjnych sieciach neuronowych, z których jedna reprezentuje model świata, a druga kontroler, który wykorzystuje model świata do wyodrębnienia polityki maksymalizacji oczekiwanej nagrody [46]. Metoda ta została zainspirowana wcześniejszymi kombinacjami nierekurencyjnych, reaktywnych modeli świata i kontrolerów. W dowolnym momencie, do którego horyzontu czasowego predyktor powinien próbować przewidywać? W przypadku AIXI właściwym sposobem traktowania horyzontu czasowego nie jest dyskontowanie go wykładniczo, jak robi się to w większości tradycyjnych prac nad uczeniem się przez wzmacnianie, ale pozwolenie, aby horyzont przyszłości rósł proporcjonalnie do dotychczasowego życia uczącego się . Pozostaje pytanie, czy ta wiedza przeniesie się do oops-rl. Jednak pomimo właściwości optymalności stronniczości oops dla pewnych uporządkowanych sekwencji zadań, oops-rl nie jest koniecznie najlepszym sposobem spędzania ograniczonego czasu w ogólnych sytuacjach uczenia się przez wzmacnianie. Z drugiej strony możliwe jest użycie oops jako podmodułu do wyszukiwania dowodów w najnowszej, optymalnej, uniwersalnej maszynie Gödla uczącej się przez wzmacnianie [56], omówionej w następnej sekcji.

Maszyna Gödla

Maszyna Gödla , również w tym tomie, wyraźnie zajmuje się „Wielkim Problemem Sztucznej Inteligencji” , optymalnie radząc sobie z ograniczonymi zasobami w ogólnych ustawieniach uczenia się przez wzmacnianie oraz z potencjalnie ogromnymi (ale stałymi) spowolnieniami ukrytymi przez AIXI(t, l) w nieco mylącej notacji O(). Została zaprojektowana do rozwiązywania dowolnych problemów obliczeniowych wykraczających poza te, które można rozwiązać przez zwykłe błędy, takich jak maksymalizacja oczekiwanej przyszłej nagrody robota w potencjalnie stochastycznym i reaktywnym środowisku (należy zauważyć, że całkowita użyteczność pewnego zachowania robota może być trudna do zweryfikowania - jego ocena może pochłonąć cały okres życia robota). Jak to działa? Podczas wykonywania pewnej dowolnej początkowej strategii rozwiązywania problemu maszyna Gödla jednocześnie uruchamia wyszukiwarkę dowodów, która systematycznie i wielokrotnie testuje techniki dowodowe. Techniki dowodowe to programy, które mogą odczytać dowolną część stanu maszyny Gödel i zapisać w zarezerwowanej części, która może zostać zresetowana przy każdym nowym teście

techniki dowodowej. W przykładowej maszynie Gödel ta zapisywalna pamięć obejmuje zmienne proof i switchprog, gdzie switchprog przechowuje potencjalnie nieograniczony program, którego wykonanie mogłoby całkowicie przepisać dowolną część bieżącego oprogramowania maszyny Gödel. Zwykle bieżący switchprog nie jest wykonywany. Jednak techniki dowodowe mogą wywołać specjalną podprocedurę check(), która sprawdza, czy proof obecnie przechowuje dowód pokazujący, że użyteczność zatrzymania systematycznego przeszukiwacza dowodów i przeniesienia kontroli do bieżącego switchprog w określonym punkcie w niedalekiej przyszłości przewyższa użyteczność kontynuowania wyszukiwania do momentu znalezienia alternatywnego switchprog. Takie dowody można wyprowadzić ze schematu aksjomatu poszukiwacza dowodów, który formalnie opisuje funkcję użyteczności, która ma zostać zmaksymalizowana (zwykle oczekiwaną przyszłą nagrodę w oczekiwanym pozostałym czasie życia maszyny Gödla), koszty obliczeniowe instrukcji sprzętowych (z których składają się wszystkie programy) oraz wpływ instrukcji sprzętowych na stan maszyny Gödla. Schemat aksjomatu formalizuje również znane własności probabilistyczne potencjalnie reaktywnego środowiska, a także początkowy stan maszyny Gödla i oprogramowanie, które obejmuje sam schemat aksjomatu (tutaj nie ma argumentu kołowego). Zatem techniki dowodowe mogą wnioskować o oczekiwanych kosztach i wynikach wszystkich programów, w tym poszukiwacza dowodów. Gdy check() zidentyfikuje udowodniony dobry switchprog, ten ostatni jest wykonywany (tutaj należy zachować pewną ostrożność, ponieważ sama weryfikacja dowodu i przeniesienie kontroli do switchprog również pochłaniają część typowo ograniczonego czasu życia). Odkryty switchprog reprezentuje globalnie optymalną auto-zmianę w następującym sensie: prawdopodobnie żaden ze wszystkich alternatywnych switchprogów i dowodów (które można znaleźć w przyszłości, kontynuując wyszukiwanie dowodów) nie jest wart czekania. Istnieje wiele sposobów inicjalizacji wyszukiwarki dowodów. Chociaż identyczne techniki dowodzenia mogą dawać różne dowody w zależności od czasu ich wywołania (z powodu ciągle zmieniającego się stanu maszyny Gödla), istnieje stronniczo optymalna i asymptotycznie optymalna inicjalizacja wyszukiwarki dowodów oparta na wariancie oops . Wykorzystuje ona fakt, że weryfikacja dowodów jest prostym i szybkim zadaniem, w którym odpowiednia jest konkretna koncepcja optymalności oops. Sama maszyna Gödla może jednak mieć dowolne, zazwyczaj inne i silniejsze poczucie optymalności ucieleśnione przez daną funkcję użyteczności.

Wnioski

Ostatnie postępy teoretyczne i praktyczne obecnie napędzają renesans w dziedzinie uniwersalnych uczniów i optymalnego wyszukiwania . Pojawia się nowy rodzaj sztucznej inteligencji. Czy naprawdę zasługuje na miano „nowej", biorąc pod uwagę, że jej korzenie sięgają lat 30. XX wieku, kiedy Gödel opublikował fundamentalny wynik teoretycznej informatyki [16], a Zuse zaczął budować pierwszy komputer ogólnego przeznaczenia (ukończony w 1941 r.) oraz lat 60-tych XX wieku, kiedy Solomonoff i Kolmogorov opublikowali swoje pierwsze istotne wyniki? Odpowiedź twierdząca wydaje się uzasadniona, ponieważ to ostatnie wyniki dotyczące praktycznie wykonalnych obliczalnych wariantów starych nieobliczalnych metod obecnie ożywiają długo uśpioną dziedzinę. „Nowa" sztuczna inteligencja jest nowa w tym sensie, że porzuca głównie heurystyczne lub nieogólne podejścia ostatnich dekad, oferując metody, które są zarówno ogólne, jak i teoretycznie poprawne, a także udowodnione jako optymalne w sensie, który ma sens w prawdziwym świecie. Jesteśmy skłonni twierdzić, że przyszłość będzie należeć do uniwersalnych lub prawie uniwersalnych uczących się, którzy są bardziej ogólni niż tradycyjni uczący się wzmacniający/decydenci, którzy opierają się na silnych założeniach Markowa, lub niż uczący się bazujący na tradycyjnej statystycznej teorii uczenia się, która często wymaga nierealistycznych założeń i.i.d. lub Gaussa. Ze względu na ciągły postęp w sprzęcie nadszedł czas na optymalne wyszukiwanie w przestrzeni algorytmów, w przeciwieństwie do ograniczonej przestrzeni reaktywnych odwzorowań ucieleśnianych przez tradycyjne metody, takie jak sztuczne sieci neuronowe z wyprzedzeniem. Wydaje się, że można bezpiecznie założyć, że nie tylko informatycy, ale także fizycy

i inni naukowcy indukcyjni zaczną zwracać większą uwagę na pola uniwersalnej indukcji i optymalnego wyszukiwania, ponieważ ich podstawowe koncepcje są nieodparcie potężne, ogólne i proste. Jak długo potrwa, zanim te idee rozwiną swój pełny wpływ? Bardzo naiwne i spekulatywne przypuszczenie napędzane myśleniem życzeniowym może opierać się na zidentyfikowaniu „największych momentów w historii informatyki" i ekstrapolacji z tego. Które to są te „największe momenty"? Oczywistymi kandydatami są:

1. 1623: pierwszy kalkulator mechaniczny Schickarda rozpoczyna erę obliczeniową (następnie maszyny Pascala, 1640 i Leibniza, 1670).

2. Około dwa wieki później: koncepcja programowalnego komputera (Babbage, Wielka Brytania, 1834-1840).

3. Wiek później: fundamentalna praca teoretyczna nad uniwersalnymi językami programowania opartymi na liczbach całkowitych oraz granicami dowodu i obliczeń (Gödel, Austria, 1931, przeformułowane przez Turinga, Wielka Brytania, 1936); pierwszy działający programowalny komputer (Zuse, Berlin, 1941). (Następne 50 lat przyniosło wiele postępów teoretycznych, a także coraz szybsze przełączniki — przekaźniki zastąpiono lampami, pojedynczymi tranzystorami, licznymi tranzystorami wytrawionymi na chipach — ale można by twierdzić, że był to raczej przewidywalny, stopniowy postęp bez radykalnych wydarzeń wstrząsających.)

4. Pół wieku później: World Wide Web (Berners-Lee z Wielkiej Brytanii, Szwajcaria, 1990).

Ta lista wydaje się sugerować, że każdy wielki przełom ma tendencję do nadchodzenia mniej więcej dwa razy szybciej niż poprzedni. Ekstrapolując trend, optymiści powinni oczekiwać, że następna radykalna zmiana ujawni się ćwierć wieku po ostatniej, to jest do 015, co przypadkowo pokrywa się z datą, kiedy najszybsze komputery dorównają mózgom pod względem surowej mocy obliczeniowej, zgodnie z częstymi szacunkami opartymi na prawie Moore'a. Autor jest przekonany, że nadchodzący przewrót w 2015 roku (jeśli w ogóle nastąpi) będzie obejmował uniwersalne algorytmy uczenia się i maszynowe, optymalne, przyrostowe wyszukiwanie w przestrzeni algorytmów — prawdopodobnie kładąc podwaliny pod pozostałą serię coraz szybszych dodatkowych rewolucji, które osiągną punkt kulminacyjny w „punkcie Omega" spodziewanym około 2040 roku.