The Intelligence Quotient of the Artificial Intelligence

Dimiter Dobrev
Institute of Mathematics and Informatics
Bulgarian Academy of Sciences

To say which programs are AI, it's enough to run an exam and recognize for AI those programs that passed the exam. The exam grade will be called IQ. We cannot say just how big the IQ has to be in order one program to be AI, but we will choose a specific value. So our definition of AI will be any program whose IQ is above this specific value. This idea has already been realized in [3], but here we will repeat this construction by bringing some improvements.

IQ-то на Изкуствения Интелект

За да кажем кои програми са AI е достатъчно да проведем един изпит и да признаем за AI тези програми, които са издържали изпита. Оценката от изпита ще наречем IQ. Не можем да кажем, колко точно е минималното IQ, за да бъде една програма AI, но ще изберем една конкретна стойност. Така нашата дефиниция за AI ще бъде всяка програма, чието IQ е над определената стойност. Тази идея вече е реализирана в [3], но тук ще повторим тази конструкция, като внесем някои подобрения.

Keywords: Artificial Intelligence, Definition of AI, IQ of AI.

Въведение:

Определяме какво е AI чрез изпит. Като резултат от изпита ще получим оценка. Тази оценка ще наречем IQ. Приемаме, че всички програми, чието IQ е над определено ниво удовлетворяват дефиницията за AI.

За да обясним идеята ще използваме аналогията с кандидатстудентските изпити. Задачите за изпита избираме случайно, но на всички студенти даваме едни и същи задачи. При това, задачите трябва да са логични, защото искаме да приемем студентите, които мислят логично, а не тези които случайно са уцелили правилния отговор. Оценката определяме на базата на това, колко от задачите кандидат-студента е решил. Не можем да кажем колко задачи трябва да бъдат решени, за да бъде приет студента, защото не знаем колко хора ще се явят на изпита и как ще се представят. Бихме могли да фиксираме една оценка (например 5.50) и да кажем, че възнамеряваме да приемем всички кандидат-студенти получили повече от 5.50. По-добре е, вместо да фиксираме минималната оценка, да я определим в последствие, когато изпита е приключил. Тогава взимаме оценката на този, който е на определена позиция (например, който е на позиция 100 по успех и така приемаме първите 100).

Тази аналогия добре описва изпита за AI, но когато провеждаме изпит между програми не можем да отделим първите 100, които са се представили най-добре,

защото в този случай кандидатите са безбройно много.Може би по-добра е аналогията, че провеждаме конкурс за директор и изпита се проточва във времето. Спираме когато някой кандидат събере достатъчно точки. Колко точки са достатъчно? Може предварително да сме избрали определено ниво, но може да променим нивото в последствие, ако предварително избраното ниво се окаже прекалено ниско или прекалено високо.

В [3] вече направихме подобна конструкция, където на различните програми се дават различни оценки ($IQ \in [0,1]$). Тук ще повторим тази конструкция, за да я подобрим. Причините, поради които статията [3] се нуждае от подобрение, са следните:

1. В [3] се разглеждат едновременно въпросите „Какво е AI?“ и „Как можем да създадем AI?“ Не е добра идея да се объркват въпросите „какво“ и „как“. Тук ще се ограничим само с въпроса „Какво е AI?“ и няма да се занимаваме с въпроса как да намерим тази програма.

2. В [3] дефинираме AI като програма, а тук ще дефинираме AI като стратегия. В [3] AI е програма и света, в който AI живее, също е програма. Така имаме две програми, които играят една срещу друга, което е малко объркващо. По-добре е да дефинираме AI като стратегия и да имаме програма играеща срещу стратегия. Разбира се, тази стратегия ще е изчислима, защото е крайна. Ще наречем AI програма, всяка програма, която реализира AI стратегия за първите хиляда игри (партии). Какво ще прави AI програмата след хиляда игри няма да е определено, но се надяваме, че тя ще продължи да се държи интелигентно и по-нататък.

3. В [3] за представянето на света се използват недетерминирани Тюринг машини. Това е едно излишно усложняване. То би имало смисъл, ако нямаше връзка между отделните игри (партии) и ако след всяка игра лентата на машината се изчиства (нулира). Тъй като лентата не се изчиства, всяка следваща игра зависи от това, което е станало в предишната игра (това което е останало върху лентата). Затова ще използваме детерминирани Тюринг машини, те са по-прости, а това че зависят от това което е останало върху лентата прави поведението им различно (недетерминирано) в различните игри.

4. На всяка стъпка имаме действие и наблюдение. В [3] действието и наблюдението са по една буква, а тук действието ще е n букви и наблюдението ще е m букви. Разбира се, ние бихме могли да кодираме няколко букви с една, но това противоречи на идеята, че трябва да избягваме излишното кодиране [6]. Света е достатъчно сложен и е трудно да го разберем. Ако добавим едно излишно кодиране, света ще стане още по-неразбираем.

5. В [3] се предполага, че всички ходове са коректни, докато тук ще добавим понятието некоректен ход. От една страна, важно е да предполагаме съществуването на некоректни ходове. От друга страна,така ще се отървем от произволното прекъсване на работата на машината на Тюринг, което правим в [3], за да избегнем зацикляне.

6. Машината на Тюринг е теоретичен можел за представяне на изчисление, който не се нуждае от ефективност. Тук ще използваме този модел за реална работа и затова ще го променим, като го направим много по-ефективен. Цената за тази по-добра ефективност ще бъде усложняването на модела.

7. В [3] използваме машината на Тюринг, за да опишем един логичен свят. Щом е изчислим, значи е логичен. Въпреки това света, който описва машината на Тюринг, не е много логичен. Всичко се записва върху една лента и програмата въобще не е структурирана. Произволно се скача от команда на команда (това програмистите го наричат „спагети код“). Работата на подобна програма е доста нелогична, затова ще я променим като добавим подпрограми и ще направим машината многолентова.

8. В [3] дефинираме IQ като едно средноаритметично, което не може да бъде изчислено точно, поради комбинаторната експлозия. Там казваме, че то може да бъде изчислено приблизително, чрез статистическа извадка. Тук ще въведем термините глобално IQ, което не може да бъде изчислено точно и локално IQ, което ще бъде лесно изчислимо и ще се изчислява чрез конкретна статистическа извадка. Локалното IQ ще клони към глобалното IQ, когато размерът на статистическата извадка клони към безкрайност. Множеството на световете, които използваме за изчисляването на глобалното IQ е крайно (огромно, но крайно). Въпреки това размерът на статистическата извадка може да клони към безкрайност, защото в тази извадка може да има повторения (макар че повторение е малко вероятно поради огромния размер на множеството, от което се избира тази извадка).

Литературен обзор

Тюринг в [7] предлага своята дефиниция за AI (теста на Тюринг). Идеята е, че ако една машина успешно може да имитира човек, то тази машина е AI. При теста на Тюринг, както и в нашата статия, имаме изпит и за да бъде призната една машина за AI, тя трябва да издържи изпита. Една разлика е, че там има изпитващ, докато при нас имаме тест с фиксирани въпроси. Тоест, теста на Тюринг е субективен, поради което той е една неформална дефиниция на AI. Все пак, основният проблем на дефиницията на Тюринг не е в нейната субективност и неформалност, а в това, че тя не дефинира интелект, а нещо повече. Тя дефинира образован интелект. За да издържи един интелект теста на Тюринг, той трябва да е образован. Дори може да предположим, че той има англосаксонско образование, защото ако не владее английски той не би се представил добре на теста.

Интелект и образован интелект са две различни неща, както различни неща са компютър без софтуер и компютър със софтуер. Ако попитате един математик, какво е компютър, той ще ви отговори: „Машина на Тюринг“. Ако зададете същия въпрос на едно дете, то то ще ви каже: „Компютърът е нещо, с което могат да се играят игри, да се гледат филми и т.н.“ Тоест, математика възприема компютъра само като хардуер, а детето възприема компютъра като неделима система от софтуер и хардуер. Когато Тюринг дава дефиниция на компютър (машината на Тюринг), той описва само хардуера, но когато дефинира интелект, той описва една неделима система от интелект и образование.

Дефиницията на AI, която ще дадем в тази статия, не включва образованието. Затова задачите от теста не предполагат никакви предварителни знания. По-точно, във всяка задача ще предполагаме, че започваме на чисто и по време на решаването на задачата се образоваме, т.е. откриваме зависимости и се учим от грешките си.

McCarthy в [8] казва, че отличителната черта на AI е: „ability to achieve goals in the world“. Ние няма да спорим с McCarthy, а само ще уточним това, което е казал. Ще уточним какво е свят и какво е произволен свят и какви са целите, които AI трябва да постигне. На тази база ще създадем IQ тест, при който програмите, които постигат повече цели, имат по-високо IQ.

McCarthy казва още, че няма „definition of intelligence that doesn't depend on relating it to human intelligence“. Действително, в тази статия ние изчисляваме стойността на IQ без това да е свързано с човешката интелигентност, но, за да кажем дали една програма е AI, трябва да кажем колко е минималното за целта IQ. За този минимум ние избрахме числото 0.7. Това число е избрано произволно и ние ще го променим, ако това ниво се окаже много по-ниско или много по-високо от човешкия интелект. Тоест, при нашата дефиниция човешкия интелект също се използва, но това е само за сравнение и целта е единствено определянето на една константа.

Друг въпрос зададен от McCarthy: Can we ask a yes or no question “Is this machine intelligent?” Отговорът е „не“ и ние сме напълно съгласни, защото не се знае колко е минималното за целта IQ.

Обаче, ние няма да се съгласим с отговора на следващият въпрос на McCarthy. Той пита „Do computer programs have IQs?“ и отговаря с „не“. В тази статия, както и в [3] ние показахме, че може да се дефинира IQ за програми. В отговорът си McCarthy по-скоро е искал да каже, че IQ тестовете за хора не са подходящи за компютърни програми. Теста, който ние предлагаме не е за хора, а е за програми.

В някои статии (например в [9]) се разглежда въпроса как да се създаде програма, която може да решава IQ тестове създадени за хора. В нашата статия въпросът, който разглеждаме, е обратният. Тук ние създаваме IQ тестове предназначени за програми (за стратегии). Тестовете, които ще предложим, няма да са подходящи за хора, макар да е възможно човек след известно обучение да се научи да ги решава. Обученият човек ще си записва какво се е случило и ще анализира, за да открие зависимости. Необучения човек няма да си записва и няма да успее да забележи зависимостите, освен ако те не бъдат представени във визуален вид или по друг начин, който да е удобен за възприемане от човек.

Много трудно е да си мислим за човека като за стратегия по следните причини. Първо, човека е недетерминиран, тоест, той не осъществява една детерминирана стратегия. (Човека можем да го разглеждаме като недетерминирана стратегия или като множество от стратегии, от които случайно се избира една.) Второ, не е ясно, колко време сме дали на човека, за да направи един ход. Трето, не е ясно колко сериозно ще подходи човека към задачата (ако подходи по-сериозно, ще получи по-добър резултат). Четвърто, човека винаги е обучен и никога не започва от нулата. Дори и новороденото бебе е преминало през някакво обучение в корема на майка

си. Затова, когато един човек осъществява една стратегия, резултата до голяма степен зависи от неговото образование, обучение, опит.

В [10] Detterman заплашва, че ще създаде система от тестове (develop a unique battery of intelligence tests), които ще могат да измерят IQ-то на компютърните програми. В тази статия, както и в [3] ние не заплашваме, а направо създаваме такъв тест.

Малко объркващо е това, че Detterman казва „компютър“, когато иска да каже „системата от компютър и програма“. По-добре е тази система за по-кратко да се нарече програма, защото когато знаем коя е програмата, не е много важно, на кой компютър ще я пуснем, защото разликата между два компютъра е единствено в бързодействието им. Обратното, ако на един компютър пуснем две различни програми, то разликата в поведението ще е огромна.

Detterman възнамерява да тества компютърните програми с IQ тестове за хора. Тоест, и той, подобно на Тюринг, не прави разлика между интелект и образован интелект. Затова теста на Detterman няма да дава време за обучение, а ще предполага, че компютърната програма, която се явява на теста, е предварително обучена.

Detterman разчита на това, че компютрите са по-дори от хората в намирането на фактологична информация. Това е една способност на компютрите, която не е директно свързана с интелекта. По подобен начин компютрите са много силни в аритметичните операции, но това не ги прави интелигентни.

Има едно нещо, с което сме съгласни с Detterman. Той отбелязва, че с предварително запрограмирани рецепти (hoc algorithms) могат да се решат много задачи, но истинският интелект трябва да може сам да намери решението.

В [11] авторите си поставят амбициозната задача да създадат IQ тест, който да е подходящ и за AI, и за хора, и за програми като Siri и AlphaGo, които не са AI. Не можем да сравняваме AI с програми, които не са AI, защото първото е програма, която може да бъде обучена за произволна задача, а второто е програма написана за конкретна задача. Например, как да сравним програма играеща шах с AI? Единственото което може да прави програмата играеща шах е да играе шах, докато AI може всичко, но не веднага, а след като бъде обучен. Тоест, единствения начин да сравним тези две програми е да ги пуснем да играят шах, то там програмата играеща шах ще има предимство, защото AI ще загуби време, за да се обучи, докато неговия опонент няма да има нужда да се учи как се играе шах, защото той това го може. Ако сравним програмата играеща шах с AI, който е обучен да играе шах, то тогава първата програма пак би имала известно предимство пред AI, както специализирания хардуер (т.е. хардуер направен специално за определена задача) има предимство пред компютърна програма работеща на компютър. Тоест, идеята да сравняваме AI с програми, които не са AI, не е добра.

Авторите на [11] се обръщат към понятия като: the abilities to acquire, master, create, and feedback knowledge. Това е все едно да обясняваме термин, чието значение не знаем, с други термини, чието значение на знаем.

В [12] се прави много сериозно обсъждане на въпроса за това как се мери IQ на AI. Също така, в [12] е направен много добър обзор на различните работи посветени на тази тема. Известен недостатък на [12] е това, че там има известна противоречивост. От една страна статията се казва „IQ tests are not for machines, yet“. От друга страна в по-ранни статии на Orallo [13, 14] се дава формална дефиниция на AI на базата на IQ тест. Изглежда сякаш Orallo прави стъпка назад и се отказва от предишните си резултати. Друга противоречивост в [12] е това, че от една страна се казва: „human IQ tests are not for machines“. Това е нещо, с което ние сме напълно съгласни, както сме съгласни и с доводите, които съпътстват това твърдение. От друга страна в същата статия авторите казват, че са съгласни с Detterman, че “there is a better alternative: test computers on human intelligence tests”

В [12], както и в [11], се търси универсално IQ, което да е приложимо към всички програми и дори и към хората. Тук се разминаваме с авторите на [11, 12]. Вече обяснихме защо не е добра идея да сравняваме AI с програми, които не са AI. Също така обяснихме защо не е добра идея да използваме едни и същи тестове за AI и за хора.

Много важна е статията [13], защото това е първата статия, в която се говори за формална дефиниция на AI и това е и първата статия, в която се въвежда понятието IQ-test за AI. Действително в [13] този тест се нарича C-test, но изрично се казва, че това е IQ-test за AI. Трябва да се извиним, че сме пропуснали да цитираме [13] в [3]. Това е един пропуск, който сега коригираме.

Въпреки сериозността и задълбочеността на [13], там са допуснати някои неточности, които не можем да подминем. Това, че обръщаме внимание на някои пропуски и неточности на [13] по никакъв начин не означава, че подценяваме значимостта на тази статия. Няма съвършени статии и във всяка статия могат да се намерят неточности. Обикновено първата статия, която се появява в някоя нова област, е леко объркана и неясна. Обикновено в по-късните статии нещата се избистрят и изясняват.

Най-сериозната неточност на [13] е това, че там не се дефинира AI, а нещо друго. Това друго нещо ще го наречем „наблюдател“. Бихме могли да кажем, че наблюдател е програма, която има само вход и няма изход, но това би било твърде рестриктивно. Затова ще кажем, че наблюдател е програма, чийто изход не влияе на състоянието на света, но може да повлияе на оценката, която програмата ще получи.

Пример за наблюдател е, когато играем на борсата с виртуален портфейл. Ние не влияем на борсовите цени, защото не играем реално, а само виртуално. (Дори и да играехме реално, пак може да се предположи, че нашите действия не влияят на борсата, защото когато играем с малки суми нашето влияние е пренебрежимо малко.) Когато играем с виртуален портфейл, ние на всяка стъпка променяме портфейла си и след всяка стъпка стойността на портфейла се променя в зависимост от промяната на цените на акциите. Оценката, която ще получим, ще бъде нашата виртуална печалба или загуба.

Това, че програмата дефинирана в [13] не може да влияе на света е съществен проблем, защото както казват хората: „За да се научиш, трябва да пипнеш“. Има и други поговорки, които казват, че само с гледане не може да се научим. Лишавайки AI от възможността да експериментира, ние сериозно го ограничаваме.

В [5] се отбелязва, че има два проблема, които AI трябва да реши. Първия е да разбере света (да построи модел на света), а втория проблем е да планира действията си на базата на намерения модел. Тоест, „наблюдателят“ решава само първия проблем, без да реши втория.

В [13] дефиницията се ограничава само до „наблюдатели“, но да се даде дефиниция на наблюдател, е достатъчно значима задача, защото това е половината от това, което AI трябва да свърши. За съжаление тази задача не е решена напълно, защото наблюдателя, който се дефинира в [13] е малко по-специален. Той или разбира света изцяло или въобще не го разбира. Тоест, наблюдател, който работи на принципа „всичко или нищо“.

В [13] са дадени случайни стрингове, които могат да се продължат по един единствен начин. Тоест, предполага се, че има една единствена най-проста зависимост и тази зависимост или ще бъде открита или няма да бъде. Този подход е твърде рестриктивен, защото предполага само наблюдатели, които разбират света напълно. Това е възможно само при много прости светове. Всеки по-сложен свят не може да бъде разбран напълно и се налага той да бъде разбран частично.

Авторите на [13] полагат сериозни усилия да направят зависимостите exception-free. Много интересно е определението за exception-free зависимости, което се дава в [13]. Въпреки всичко ние не бихме тръгнали по този път, защото ако зависимостите включваха изключения, това би бил един начин да се допусне частично разбиране на света. Търсенето на една единствена exception-free зависимост, която да опише всичко е причината, поради която дефинирания в [13] наблюдател залага на принципа „всичко или нищо“.

Имаме още няколко дребни препоръки към [13].

Даваме някакво време на AI, за да намери зависимостта. Въпросът е дали ще го дадем това време на веднъж или ще го разпределим на много стъпки. По принцип AI търси зависимостите през целия си живот. Тоест, за много стъпки. В [13] зависимостта се търси само за една стъпка. В нашата статия предполагаме, че в живота стъпките са около един милион (хиляда игри по хиляда стъпки). Това означава, че трябва да дадем на програмата, която се дефинира в [13], милион пъти повече време. Нашата препоръка е зависимостите да се търсят след всяка стъпка, а не само след последната.

В [13] не ни харесва още това, че програмата, която генерира теста не работи поради комбинаторна експлозия. Това е програмата наречена „The Generator“. Тоест, в [13] не се дава тест, а само се показва, че такъв тест теоретично съществува.

Друг проблем е това, че са наложени две ограничения. Когато „наблюдателят“ прави предсказание той трябва да заложи всичко на един резултат и винаги трябва да залага една и съща сума. По-добре би било да има свободата да залага на повече от един резултат, както и да може да реши каква сума да заложи. Когато сме по-уверени залагаме повече, а когато се колебаем залагаме по-малко или пасуваме. Тези две ограничения не променят съществено нещата, защото умния ще докаже, че е умен дори и с тези ограничения, но така се замъглява картината. Ако ги нямаше тези ограничения разликата между IQ-то на умните и глупавите би била по-голяма.

Не ни харесва и това, че по-сложните задачи в [13] имат по-голяма тежест от по-простите, а би трябвало да е обратното. (Има един коефициент *e*, който се предполага че е неотрицателен, а би било по-добре да е отрицателен.) Вярно е, че когато правим контролно даваме повече точки на по-трудните задачи, но това е защото предполагаме, че студентите ще загубят повече време с по-трудната задача. Тук не е така. Тук на всяка задача даваме еднакво време. Затова, ако някоя проста задача не може да бъде решена, това е сериозен проблем и това трябва да се отрази в оценката. Освен това, понякога ще уцелваме решенията на трудните задачи случайно и затова те трябва да са по-малко и с по-малка тежест, в противен случай ще дадат незаслужено покачване на IQ-то.

Имаме понятията глобално и локално IQ (тези две понятия са дефинирани в нашата статия). Глобалното IQ е нещо точно, което не може да се сметне точно (заради комбинаторна експлозия). Локално IQ не е нещо точно, защото зависи от избора на конкретните въпроси в теста, но при конкретни въпроси, то може да се сметне лесно и точно. Това, което се дефинира в [13] е локалното IQ, а не глобалното. Все пак, да не забравяме, че локалното IQ клони към глобалното, но в [13] нищо не се казва за програмата „зубрач“, която е основния проблем на локалното IQ.

Как да коригираме дефиницията от [13], така че да получим дефиниция на наблюдател, която да не е на принципа „всичко или нищо“?

Вместо да взимаме специална k-разбираема редица, ние ще вземем съвсем произволна програма и редицата, която тази произволна програма генерира. Вместо да предсказваме продължението еднократно, ние ще го предсказваме на всяка стъпка. Няма да залагаме всичко на едно предсказание, а ще разрешим залога да бъде разпределен между няколко предсказания. Ще разрешим и залога да бъде различен. (Например, ще предположим, че сумата от залозите за последните пет стъпки е ограничен от някаква константа, но че имаме свободата да изберем кога да залагаме.) Успеха за една редица ще бъде сумата от успехите на различните стъпки. Локалното IQ ще бъде средното аритметично на успехите на редиците, които сме включили в теста.

По този начин няма да искаме от „наблюдателя“ да разбере света напълно, защото той може да хване някакви зависимости и по този начин, чрез частично разбиране на света, да получи високо IQ.

Проблем на [13] е, че програмата, която дефинира е всъщност програмата [1]. Това е една проста програмка, която предсказва продължението на редицата на базата на най-простата зависимост, която може да генерира началото й.

Аз дори твърдя нещо повече: Никоя програма удовлетворяваща дефиницията в [13] не е по-добра от [1]. Това, че никоя не е по-добра като резултат, е ясно, но аз твърдя че дори и като бързодействие никоя не е по-добра, защото дефиницията в [13] не ни дава никакви възможности за хитруване и за поетапно откриване на зависимости и единствената възможност остава глупавото изброяване на всички възможни зависимости.

Интересното е, че в по-новите статии на Orallo (например в [14]) той явно е разбрал основния пропуск на [13] и това, което дефинира вече не е „наблюдател“, а е програма, която може да влияе на света. За съжаление тази програма отново не е AI. В [14] се определя програма, която играе някаква игра, която се състои в обикаляне на един лабиринт (граф) като се гони нещо добро и се бяга от нещо лошо. За да се играе тази игра, определено има нужда от интелигентност, но програмата играеща тази игра не е AI, както и програмата играеща шах не е AI.

Пак в [14] авторите говорят за reinforcement learning. Тоест, ясно е, че те много добре знаят какъв е общия вид на AI. Защо тогава те не използват този общ вид, а се ограничават със световете на някаква определена игра? Според мен причината е, че те се опитват да избегнат световете, в които са възможни фатални грешки. За проблема с фаталните грешки се споменава още в [2], но фаталните грешки всъщност не са проблем. Ние хората също живеем в свят, в който има фатални грешки, но това не ни пречи да се изявим и да покажем кой е по-добър и кой е по-лош. Е да, при хората е проблем, защото ние живеем само един живот, но теста за IQ се състои от много задачи, всяка от които е един отделен живот. Дори и да се получат фатални грешки в няколко живота, това няма да повлияе съществено на средната оценка. Дори и при хората живота не е един. От гледна точка на отделния човек, живота действително е един, но от гледна точка на еволюцията, животите са много. Някои от нашите наследници ще загинат поради допускане на фатални грешки, но други ще оцелеят. По този начин средния успех на нашите наследници няма съществено да се повлияе от това, че някои от тях са допуснали фатални грешки.

Вярно е, че в тази наша статия, както и в [3], ние също не използваме произволен свят, а взимаме само изчислимите светове, но това ограничение не е съществено, защото всеки ограничен във времето свят е изчислим, а ние можем спокойно да приемем, че всички светове са ограничени във времето. Тоест, ограничавайки се с изчислимите светове ние въобще не се ограничаваме, а само даваме по-голямо тежест на световете, които са по-прости (по Колмогоров).

В [14] има още едно нещо, с което ние не сме съгласни. Това нещо се нарича „коефициент на обезценка“. Разбира се, това не идва на авторите на [14], а е нещо широко разпространено сред хората работещи в областта на reinforcement learning. Идеята на „коефициента на обезценка“ е че миналото е по-важно от бъдещето. Живота е потенциално безкраен, а за да оценим един безкраен живот трябва да обезценим бъдещето. Все пак, не е добра идея миналото да е по-важно от

бъдещето. Би било по-добре да е обратното, защото в миналото ние още не сме се обучили, а в бъдещето вече сме обучени. При хората ние не броим колко пъти се е напикавал човек докато е бил бебе. Вместо това ние гледаме какви постижения е постигнал човека в зрялата си възраст. Затова подхода, който ние приехме в [3] и в тази статия е да няма „коефициент на обезценка“, но живота да е ограничен. Казано с други думи, подхода тук и в [3] е коефициента на обезценка да е едно до един момент (края на живота) и да бъде нула от този момент нататък.

Постановка на задачата

Имаме устройство, което живее в някакъв свят. На всяка стъпка устройството извежда *n* букви (това е действието), след което получава *m* букви от външния свят (първата от които ще наречем оценка (reward), а останалите *m-1* букви ще наречем наблюдение). Оценката ще има 5 възможни стойности: *{nothing, victory, loss, draw, incorrect_move}*.

Думите „ход“ и „действие“ ще ги използваме като синоними. Ако гледаме на живота като на игра е по-естествено да кажем ход вместо действие. Думите „история“ и „живот“ също ще ги използваме като синоними.

Стъпка на устройството ще наречем тройка от вида <действие, оценка, наблюдение>. Живот на устройството ще наречем последователността от стъпки, която се е получила когато устройството е взаимодействало с определен свят.

Истински живот ще наричаме живота без некоректните ходове. Тоест, всички тройки <действие, оценка, наблюдение>, в които оценката е „*incorrect_move*“ трябва да се премахнат от живота, за да остане истинския живот.

Момент ще наречем поредица от стъпки такава, че последната стъпка преди поредицата и последната стъпка от поредицата да са коректни и всички стъпки между тези двете да са некоректни. Тоест, в живота стъпките може да са повече от моментите, но в истинския живот броя на стъпките е равен на броя на моментите.

Ще предполагаме, че устройството и света се държат детерминистично. Тест, ще предполагаме, че ако знаем кое е устройството и кой е света, то знаем и коя е историята.

Поведението на устройството можем да го представим като стратегия, тоест като функция, която за всяко начало на живота дава следващия ход на устройството. Аналогично можем да представим поведението на света като стратегия, която на всяко начало на живота и всяко действие на устройството дава оценката и наблюдението, които устройството ще получи на следващата стъпка. Трябва да отбележим, че стратегията на света не зависи от некоректните ходове. Тоест, стратегията на света можем да си я мислим като функция на истинския живот. Обратно, стратегията на устройството ще зависи от некоректните ходове (тези ходове ще представляват допълнителна информация, която устройството ще използва).

Можем да си мислим че устройството и света са две стратегии играещи една срещу друга, но това не е точно, защото устройството има цел, докато света няма цел. Тоест, света не играе срещу устройството. Предполагаме, че света просто съществува и че той не се интересува от това дали на устройството му е добре или зле.

Представянето на устройството и на света като стратегии не е много добра идея, защото стратегията помни всичко (т.е. тя зависи от живота до момента). Нормално е да предполагаме, че устройството може и да не помни всичко. Аналогично нещо може да се каже и за света. Може да имаме свят от чието вътрешно състояние ние да можем да възстановим цялото минало (тоест живота до момента). Възможно е обаче света да не помни всичко и да имаме две различни истории, които да водят до едно и също вътрешно състояние на света. Затова ние ще представим света и устройството като функции.

Нека имаме две множества Q и S. Това ще са множествата на вътрешните състояния на устройството и на света. Тези множества ще са крайни или най-много изброими. Нека q_0 и s_0 са началните състояния на устройството и на света. Ще предположим, че тези начални състояния са фиксирани, защото живота ще зависи от това от кои начални състояния сме тръгнали, а ние искаме живота да зависи само от устройството и от света.

Устройството и света ще бъдат функциите:

$$Device: Q \times Rewards \times Observations \times 2^{Actions} \to Actions \times Q$$
$$World: S \times Actions \to Rewards \times Observations \times S$$

Функцията *Device* за всяко вътрешно състояние на устройството, оценка, наблюдение и множество от доказано некоректни в този момент ходове ще върне действие и ново вътрешно състояние на устройството. Ще предполагаме, че *Device* никога не връща действие, което е доказано некоректен в този момент ход.

Вътрешното състояние на устройството ще отразява това, което то е запомнило. Какво може да е запомнило? То може да помни всичко, което се е случило до момента, плюс последното си действие. Затова написахме $Actions \times Q$, а не $Q \times Actions$. Искахме да подчертаем, че новото вътрешно състояние на устройството може да помни последното действие.

Функцията *World* за всяко вътрешно състояние на света и действие ще върне оценка, наблюдение и ново вътрешно състояние на света. Естествено е да предполагаме, че има моменти (вътрешни състояния на света), в които определено действие е невъзможно или некоректно. Тоест, естествено е да предполагаме, че функцията *World* е частична. Ние ще додефинираме функцията и за тези моменти като по този начин ще я продължим до тотална. В тези моменти *Reward* ще бъде равно на *incorrect_move*, стойността на наблюдението ще е без значение, а новото вътрешно състояние ще бъде същото като старото, макар че и то е без значение.

Вътрешното състояние на света ще отразява това, което света е запомнил. Какво може той да е запомнил? Той може да помни всичко, което се е случило до

момента, плюс последните оценка и наблюдение. Затова написахме $Rewards \times Observations \times S$, а не $S \times Rewards \times Observations$. Искахме да подчертаем, че новото вътрешно състояние може да помни последните оценка и наблюдение.

В [2] и в [3] дефинирахме новите <Reward, Observation> като функция на новото вътрешно състояние. Тоест там предполагахме, че те задължително се помнят, докато сега това изискване отпада. Да вземем като пример свят, в които играем шах. Партията завършва и новото вътрешно състояние на света е шахматна дъска с началната позиция. В този случай не е нужно да помним кой е победил в последната партия. Подобно изискване само би ни затруднило.

Живота на устройството ще изглежда така:
$<a_1, r_1, o_1>, <a_2, r_2, o_2>, \ldots, <a_{t-1}, r_{t-1}, o_{t-1}>$

Да видим как функциите *Device* и *World* ни определят живота.

$<a_{i+1}, q_{i+1}> = Device(q_{i-j}, r_{i-j}, o_{i-j}, incorrect\_actions_i)$
$<r_{i+1}, o_{i+1}, s_{i+1}> = World(s_{i-j}, a_{i+1})$

Тук $i-j$ е последната коректна стъпка преди $i+1$. Множеството $incorrect\_actions_i$ съдържа доказано некоректните действия в този момент. Това множество има j елемента. Тоест $incorrect\_actions_i = \{a_{i-j+1}, \ldots, a_i\}$. Множеството $incorrect\_actions_0$ ще бъде празното множество, защото в първия момент още преди първата стъпка няма да има доказано некоректни действия.

За да определим живота ще трябва да фиксираме първата оценка и първото наблюдение (r_0, o_0). Не бихме искали живота да зависи от това кои ще са първата оценка и първото наблюдение и затова решаваме всичките букви на тези два вектора да имат стойността *nothing*. Това е едно логично решение, защото естествено е в първия момент ние да не получавам никаква оценка и да не виждаме нищо съществено (т.е. в първия момент мие виждаме нулевата стъпка).

След като сме фиксирали ($q_0, s_0, r_0, o_0, incorrect\_actions_0$) можем да построим живота до стъпка t и този живот ще зависи само от функциите *Device* и *World*.
$<a_1, r_1, o_1>, <a_2, r_2, o_2>, \ldots, <a_{t-1}, r_{t-1}, o_{t-1}>$

Заедно с живота, ние ще построим и редиците с вътрешните състояния на устройството и на света, както и редицата $incorrect\_actions_i$ от доказано некоректните ходове в съответния момент. Да отбележим, че некоректните действия в даден момент може да са много, но доказано некоректните са само тези които вече сме пробвали и вече сме видели, че действително са некоректни в този момент.

Ще предполагаме, че функцията *Device* връща винаги ход, който не е доказано некоректен. В противен случай ще се получи зацикляне. Какво ще правим, ако всички ходове са доказано некоректни? (Тоест, ако $incorrect\_actions_i$ съвпада с цялото множество *Actions*.) В този случай ще предполагаме, че функцията *Device* не е дефинирана. Това е случая когато влизаме в тупик и няма никакъв възможен следващ ход.

Забележка: Дефинициите на *Device* като функция и като стратегия са еквивалентни с тази разлика, че ако разглеждаме *Device* като стратегия, то може да има значение в даден момент в какъв ред сме пробвали некоректните ходове, а при дефиницията като функция това е без значение. Тази разлика може да се отстрани по два начина. Първият е при дефиницията на функция вместо множеството на доказано некоректните ходове да вземем списъка на тези ходове. Вторият начин, е да се ограничим със стратегии, при които да няма значение реда, в който сме пробвали некоректните ходове. В тази статия няма да има значение какво би се случило, ако стратегията пробва некоректните ходове в друг ред, защото предполагаме че стратегията е детерминирана и реда, в който тя ще пробва некоректните ходове е фиксиран.

Забележка: Тук предполагаме, че когато се опитаме да играем некоректен ход, нищо не се случва, а само получаваме информация, че хода е бил некоректен. Можем да разрешим на устройството да пробва дали хода е коректен или некоректен без задължително да го играе (както сме направили в някои предишни наши статии). В този случай ще трябва да променим постановката на задачата и да добавим още една оценка *correct_move*. Функцията *Device* ще трябва да има още един аргумент съдържаш доказано коректните ходове. Когато пробваме ход, който е доказано коректен ще приемем, че играем този ход. Когато ходът не е в множеството на доказано коректните, тогава ще приемем че само го пробваме. Функцията *World* ще има още един булев параметър, който ще казва дали хода се играе действително или само се пробва. Когато хода само се пробва ще върне една от двете оценки *correct_move* или *incorrect_move*. Тогава хода няма да се играе, а само ще се добави към следващото множество на доказано коректните или на доказано некоректните ходове.

Игра (пария) ще наричаме част от живота, която се намира между две последователни заключителни оценки. Заключителни оценки ще наричаме стойностите *{victory, loss, draw}*. Ще предполагаме, че всяка игра е не по-дълга от хиляда хода (т.е. хиляда момента, броя на стъпките може да е и по-голям заради некоректните ходове). Ще предполагаме, че в живота имаме не повече от хиляда игри.

Ще дадем дефиниция на това коя стратегия е AI и нашата дефиниция ще зависи от редица параметри. Повечето параметри ще ги фиксираме да бъдат числото хиляда, защото това е едно хубаво кръгло число. Друго подобно кръгло число е милион. Ако заменим числото хиляда с милион, то ще получим друга дефиниция на AI, която няма да се отличава съществено от дадената.

Параметри

Брой на буквите на действието	n
Брой на буквите на наблюдението	m
Брой на възможните символи за всяка от буквите на действието	$k_1, \ldots, k_n$, $k_i \geq 2$.
Брой на възможните символи за всяка	$k_{n+1}, \ldots, k_{n+m}$,

от буквите на оценката и наблюдението	$k_i \geq 2$, $k_{n+1}=5$.
Брой символи на лентите	$MaxSymbols = 10 + \max_{i \in [1,\ n+m]} k_i$
Брой на глобалните ленти	7 (от 3 до 9)
Брой на вътрешните състояния	1000
Брой тестови светове	1000
Максимален брой игри в един живот	1000
Максимален брой ходове в една игра	1000
Максимален брой стъпки за един ход	1000
Вероятност, с помощта на която се генерира машината	$\frac{1}{10}$ = 10%
Минималното IQ, за да може да бъде призната стратегията за AI	0.7 = 70%

Първите четири реда на таблицата ни дават параметрите, които описват входа и изхода на AI. Тези параметри ни казват какъв е формата на търсения AI. Затова тези параметри не можем да променяме произволно. Следващите осем параметъра влияят на избора на световете избрани за изпита и затова влияят на IQ-то, което ще получим и от там влияят на дефиницията на AI. Последния параметър също влияе на дефиницията. Тоест, променяйки последните девет параметъра ние бихме променили дефиницията на AI.

Тук не се включват някои параметри, от които зависи дефиницията на AI. Например не се казва точно кой е генератора на псевдо-случайни числа, който използваме, за да изберем световете от изпита. Разбира се, зависимостта на дефиницията от този параметър е незначителна.

Възможните символи за *i*-тата буква на действието ще са от 0 до k_i*-1*. Аналогично възможните букви за *i*-тата буква на наблюдението ще са от 0 до k_{n+1+i}*-1*. Символът 0 ще го наречем nothing. Първата буква на наблюдението ще бъде оценката. Когато става дума за оценката, символите 1, 2, 3 ще ги наречем victory, loss, draw и те ще са заключителните оценки. Оценката 4 (incorrect move) няма да се извежда от машината на Тюринг в резултат на извикване на командата q_1. Тази оценка ще се извежда само когато машината на Тюринг зацикли (тоест направи повече от 1000 стъпки без да достигне до заключителното състояние) или когато изгърми (например да извика командата **return** при празен стек).

Символите на лентата ще бъдат колкото е нужно, за да се кодира с тях действието и наблюдението. Тоест, максималното k_i за i от 1 до $n+m$. Към това ще добавим още 10 служебни символа, първият от които ще бъде празния символ λ.

Какъв е изпита (теста)

За изпита ще изберем 1000 свята. Във всеки от тези 1000 свята кандидата ще изживее по един живот, състоящ се от не повече от 1000 игри. Накрая ще изчислим броя на победите, загубите и ремитата. Като резултат ще получим IQ, което е равно на средното аритметично, където победата е едно, загубата е нула, а ремито е 1/2.

Световете ще ги изберем случайно, но искаме избраните светове да са фиксирани и затова ще ги изберем псевдо-случайно като преди да започнем избора ще инициализираме генератора на псевдо-случайни числа с числото 1. По този начин ще провеждаме изпита винаги с едни и същи светове.

Много от случайно генерираните светове ще са такива, в които задължително се печели или задължително се губи. Включването на подобни светове в изпита е безсмислено и затова ние ще изхвърлим тези светове от изпита. Така ще останат 1000 смислени свята.

Какво е свят

Във вестника можете да намерите задачи от вида „Кое е следващото число в редицата?“. Ако всички възможни редици са равновероятни, то следващото число може да бъде което си поиска. Когато търсим следващото число в редицата ние предполагаме, че по-простите редици са по-вероятни от по-сложните. Тоест, ние използваме принципа наречен „Бръснача на Окам“.

Аналогично е и положението със световете. Ако разглеждаме света като стратегия и ако всички стратегии са равновероятни, то няма как да предскажем бъдещето и няма на базата на какво да изберем един ход пред друг ход. За световете ние също ще използваме „Бръснача на Окам“ и ще предположим, че по-простите светове са по-вероятни. Кога един свят е по-прост от друг? Ще използваме сложността по Колмогоров. Тоест, ако света е стратегия, то по-простата стратегия е тази, която се генерира от машина на Тюринг с по-малко състояния.

Ние сме се ограничили само до крайните стратегии. Следователно всички стратегии са изчислими. Затова ние ще приемем, че света е някаква машина на Тюринг, която изчислява някаква стратегия.

Кои ще са възможните светове

Ще се ограничим до машините на Тюринг с 1000 състояния. Това множество включва и машините с по-малко състояния, защото всяка машина може да бъде допълнена с недостижими състояния. Машините на Тюринг, които използват съществено повече от 1000 състояния няма да са част от това множество. Тези светове ще ги приемем за твърде сложни и затова те няма да участват в дефиницията.

Получаваме едно огромно множество от стратегии (светове). Тук по-простите стратегии ще имат по-голяма тежест (те ще са по-вероятни), защото ще се генерират от повече машини на Тюринг.

Допълнително, на всяка машина на Тюринг ще дадем някаква тежест. Тоест, ще предпочитаме някои машини на Тюринг пред други. Например, ако машината

използва повече състоянията с по-малък номер, ще я предпочетем пред тази, която използва повече от тези, които са с по-голям номер.

Причините, поради които даваме различна тежест на различните машини на Тюринг, са две. Първата е, че по този начин даваме по-голяма тежест на по-простите машини (например, тези, при които достижимите състояния са по-малко, са по-прости). Втората причина е, че искаме по случаен начин да генерираме работеща машина, което е много трудно. Тези машини, които имат по-голям шанс да са работещи са с по-голяма тежест, следователно избираме ги с по-голяма вероятност и по този начин увеличаваме шанса си да уцелим работеща машина.

Тежестта на машината е равна на вероятността, с която бихме я избрали. Ние няма да изчисляваме тази вероятност. Това би било едно доста трудно изчисление. Просто избираме машината на Тюринг случайно и чрез този избор вкарваме вероятността като параметър във формулата. Тоест, при изчисляването на локалното IQ, тази вероятност няма да се изчислява. Ако изчисляваме глобалното IQ, то тогава би трябвало да вземем всички машини на Тюринг, за всяка да сметнем успеха на устройството при тази машина и вероятността тази машина да бъде избрана. Трябва да умножим тези две числа и да сумираме по всички машини. Подобно изчисление е невъзможно поради комбинаторната експлозия. Не е проблем това, че усложняваме това изчисление, като добавяме изчисляване на вероятности. По този начин ние нищо не променяме. Глобалното IQ остава на теория изчислимо, а на практика пак е не изчислимо.

Как да сметнем IQ-то на конкретна програма

Бихме могли да кажем, че IQ-то ще е равно на средното аритметично на успеха във всички светове. (Тук трябва да отчетем, че световете не са равновероятни и трябва да умножим успеха по вероятността (теглото) на съответния свят.)

Това IQ ще го наречем глобално. Дефиницията на глобалното IQ е много хубава. Единствената забележка е, че това IQ не може да бъде изчислено. По-точно, то може да бъде изчислено на теория, но на практика не може, поради огромния брой светове които допуснахме за възможни.

Все пак, глобалното IQ можем да го изчислим приблизително по методите на статистиката. Ще изберем случайно 1000 свята и ще сметнем средното аритметично за тези светове. Полученият резултат би бил близък до глобалното IQ.

Тук проблемът е, че при различен избор на тестовите светове ще получим различно приближение на глобалното IQ. Ние искаме да имаме програма, която за всеки кандидат да дава неговото IQ и това да е една определена стойност, а не някакво приближение на нещо друго. Затова ние ще фиксираме произволно избраните 1000 свята и ще кажем, че локалното IQ е средния успех върху тези 1000 свята. (Тук различните светове няма да имат различно тегло, защото това е отчетено при избора на тестовите светове. По-тежките се избират с по-голяма вероятност.)

Идеята да фиксираме произволно избраните светове е същата като да дадем на всички кандидат-студенти едни и същи задачи.

Локалното IQ е една лесно изчислима функция и то добре описва идеята ни за това какво е IQ. Има само един проблем. Има една програма, която ще наречем програмата зубрач. Тази програма е подготвена специално за тестовите 1000 свята и нейното локално IQ е много високо, но глобалното ѝ IQ е ниско. Как ще решим този проблем? Когато търсим AI ще използваме локалното IQ. Когато намерим програма, която има много високо локално IQ, за която подозираме, че е програмата зубрач, ще ѝ дадем допълнителни задачи. Тоест, ще изчислим второто локално IQ. Това означава, че ще вземем следващите 1000 произволни свята и върху тях ще сметнем друго средно аритметично. Може да продължим и с третото и с четвъртото локално IQ.

Как машина на Тюринг прави ход

Светът го представяме като машина на Тюринг. Тоест, тя трябва да приема действията като вход и да извежда наблюденията като изход.

Първите $m+1$ състояния на машината ще бъдат специални. Състоянието q_{m+1} ще бъде началното и заключителното състояние на машината. Състоянията от q_1 до q_m ще бъдат състоянията, при които се извеждат буквите на наблюдението.

При първия ход на машината всички ленти ще са празни (т.е. ще са покрити със символа λ). Първата текуща лента ще бъде с номер 3 (номера 0, 1 и 2 се използват служебно).

Ход машината ще прави като започне от началното състояние q_{m+1} и завърши в същото състояние (то е и заключително). В началото на всеки ход върху текущата лента под главата на машината ще се запише n буквена дума, която ще е действието. (Това, което е било на първите n букви на лентата преди да се запише тази дума ще се изтрие.)

При всяка стъпка ще се следи за извикването на състоянията от q_1 до q_m. При извикването на тези състояния ще се извеждат буквите на наблюдението. Ако състоянието q_i се извика в рамките на един ход няколко пъти, то ще се гледа само първото му извикване. Ако не се извика нито веднъж, то i-тата буква на наблюдението ще бъде символа nothing. Ако се извика поне веднъж, то i-тата буква на наблюдението ще бъде стойността на „паметта на главата“ в момента след първото извикване на q_i. Ако i-тата буква на наблюдението е по-голяма или равна от съответното k_{n+i}, то тогава машината ще изгърми и ще се случи същото, което се случва при зацикляне.

Некоректни ходове

Когато машината на Тюринг не успее да направи ход, защото зацикля или изгърмява по някаква причина, тогава ще считаме, че действието, което е въведено в началото на хода е некоректно и невъзможно. В този случай ще се върнем назад и ще пробваме да въведем друго действие. По-точно, няма да се връщаме ние, а ще върнем назад света (машината на Тюринг). Ще дадем на устройството оценка *incorrect _move* и ще вземем следващия ход, който то ще ни даде. Ще продължим с този нов ход, все едно че устройството е изиграло него вместо некоректния ход. (Ако устройството повтори същия некоректен ход, ще го дисквалифицираме, защото то няма право да пробва два пъти един и същи ход в един и същи момент.)

Връщането назад на машината на Тюринг е съвсем естествена операция. Трябва само да сме запомнили конфигурацията на машината преди началото на хода. Ако възстановим тази конфигурация , можем да въведем друго действие и да продължим все едно, че това е първото действие, което сме пробвали.

В [3] се прилага друг подход. Там се предполага, че всички ходове са коректни и проблемът със зацикляне се решава като се прекъсва изпълнението на програмата, присъжда се служебно равенство и програмата се рестартира от началното състояние. При това рестартиране лентата е останала в състоянието, в което е била в момента на прекъсването. Това е една много лоша практика. Вие знаете, че когато изключвате компютъра си, трябва да дадете командата „Shut Down“. Другият вариант е да издърпате щепсела от контакта, но тогава хард диска ще остане в състоянието, в което е бил когато сте дръпнали щепсела. Подобно отношение би накарало вашия компютър да се държи странно и нелогично. Същото може да се каже за машина на Тюринг, която се прекъсва в произволен момент и след това се рестартира от началното състояние. Ние искаме света да е колкото се може по-логичен и затова ще се погрижим да няма подобни прекъсвания.

При положение, че допускаме някой от ходовете да са некоректни, трябва да кажем какво правим когато всички ходове са некоректни. Да предположим, че имаме стотина възможни действия. Пробваме ги всичките и те всичките се оказват некоректни. Тогава ще считаме че сме попаднали в тупик и че няма път наникъде.

Ако живота е последователността от 1000 игри, то живота свършва естествено след 1000 игри или с внезапна смърт (попадане в тупик). Как да оценим един живот, ако той е завършил преждевременно? Можем да сметнем само изиграните до момента игри. Тогава нашата AI стратегия ще предпочете да се самоубие (да влезе в тупик), когато разбере че в текущия живот нещата са се объркали и оттук нататък се очакват само загуби.

Ние не искаме нашата дефиниция да ни даде AI стратегия със суицидни наклонности и затова ще изберем друго решение. Когато стратегията попадне в тупик, ще смятаме че всички останали игри до 1000 са загуби. По този начин, ще сме сигурни, че стратегията няма да влезе доброволно в тупик, а ще се бори до последно.

Ще разбере ли стратегията, че влиза в тупик? Подобно нещо няма как да се научи по метода на проба и грешка, защото в тупик се влиза само веднъж. Въпреки това,

ако стратегията е много умна, то тя може да предскаже някои от влизанията в тупик. Например, ако броя на възможните ходове намалява, то това не е добре, защото може накрая да не остане нито един възможен ход. Друг пример е човека. Да вземем един конкретен човек, който никога не е умирал. Той няма личен опит, но въпреки това той може да предвиди някои от ситуациите, които биха довели до смъртта му.

Наказваме мотаенето

Казахме, че една игра продължава не повече от 1000 хода. Какво ще направим, ако играта продължи повече от 1000 хода? Тогава ще отсъдим служебно реми. Забележете, че тук не се намесваме в работата на машината на Тюринг и тя продължава да играе същата партия. Намесата е само към стратегията, защото тя ще получи оценка реми, въпреки че света (машината) е дал оценка nothing. Това, че не прекъсваме работата на машината гарантира, че тя ще запази логичното си поведение.

Ако изминат още 1000 хода без заключителна оценка ще присъдим служебна загуба. Тоест, ще накажем стратегията за мотаенето. Искаме да имаме AI стратегия, която се стреми бързо да завърши партията и да започне нова.

Наказвайки преждевременната смърт и мотаенето ние променяме IQ-то на случайната стратегия. Ако играем случайно, то очакваното IQ е 1/2. За да бъде повече, трябва нарочно да се стремим да печелим. За да бъде по-малко, трябва нарочно да се стремим да губим. Обявявайки всички игри след внезапната смърт за загуби, ние намаляваме IQ-то на всички стратегии. Аналогично, добавянето на служебни загуби също ще даде такова намаление. С това решение IQ-то на случайната стратегия няма да е 1/2, а ще е по-малко.

Колко точно е IQ-то на случайната стратегия ще можем да изчислим когато напишем програмата изчисляваща локалното IQ. Разбира се, случайната стратегия не е детерминирана и затова ще трябва да я изпитаме няколко пъти и да вземем средното. Тоест, IQ-то на случайната стратегия ще е приблизително. Точно локално и глобално IQ ще имат само детерминираните стратегии.

По-логична и по-ефективна машина на Тюринг

Както казахме, ще променим дефиницията на машината на Тюринг, за да я направим по-логична и по-ефективна. За целта ще направим машината на Тюринг многолентова и ще ѝ дадем възможност да вика подпрограми.

Защо тази машина ще ни даде по-логичен свят?

Първото е това, че машината на Тюринг ще е многолентова. Състоянието на света е естественото да се представи като декартово произведение на много параметри,

които слабо си взаимодействат. Затова многолентовата машина на Тюринг дава по-логичен модел на света от еднолентовата.

Второто е това, че в тази машина ще има подпрограми, които се извикват от много места. Това ще е по-логично отколкото всеки път да се вика различна подпрограма. Когато нямаме стек се налага да помним къде да се върнем след изпълнението на подпрограмата. За да стане това, трябва всяка подпрограма да се извиква само от едно място (иначе няма да знае къде да се върне).

Когато викаме подпрограма ще ѝ дадем една чиста лента, на която тя да записва междините си резултати. Ако не ѝ дадем такава чиста лента, то тя ще трябва да използва някоя от общите ленти и това ще направи работата ѝ доста по-нелогична, защото ще се получат странни взаимодействия между различните извиквания на една подпрограма.

Защо тази машина ще прави по-малко стъпки и ще е с по-малко вътрешни състояния?

По-добра ефективност, ще означава по-малко стъпки и най-вече по-малко вътрешни състояния. Важно е стъпките да не са прекалено много, защото ако машината направи повече от 1000 стъпки за един ход ние приемаме, че е зациклила и я спираме. Важно е и вътрешните състояния да не са прекалено много, защото ние се ограничихме до машините с не повече от 1000 състояния. Затова нашата машина е добре да използва по-малко състояния.

Това, че машината е многолентова ще намали броя на стъпките, защото когато лентата е една ще се наложи на главата да се движи много, за да записва междините резултати. По-лесно ще бъде тези резултати да бъдат записани на друга лента.

По-съществено ще е това, че ще намалим броя на вътрешните състояния на машината. Класическата машина на Тюринг използва огромен брой вътрешни състояния (тя помни всичко, което трябва да се помни, във вътрешното си състояние). Например, когато се вика подпрограма трябва да се запомни от къде тази подпрограма е извикана и къде трябва да се върнем. Когато искаме да преместим символ от едно място на друго трябва да помним кои символ сме взели.

По тази причина усложняваме машината на Тюринг и тя освен вътрешното си състояние ще помни още коя е текущата лента, показалеца на стека (за подпрограми) и един символ „паметта на главата“.

Машина на Тюринг със стек

Как ще изглежда програмата на тази машина? Тя ще бъде една таблица с размери 1000 на MaxSymbols. Тук 1000 са възможните команди, а MaxSymbols са възможните символи. Във всяко от полетата на тази таблица ще има пет команди.

Първата команда е „Пишем върху лентата“

MaxSymbols+2 възможни стойности:
(без промяна, старата стойност на паметта на главата, конкретен символ)

Втората команда е „Променяме паметта на главата“
MaxSymbols+2 възможни стойности:
(без промяна, старата стойност на символа от лентата, конкретен символ)

Третата команда е „Движение на главата“
Три възможни стойности:
(ляво, дясно, на място)

Четвъртата команда е „Подпрограма“
Има две полета:
„Състояние“ в интервала [0, 1000] (тук 0 означава командата NULL).
„Нова текуща лента“ в интервала [0, 9] (тук 0 означава текущата лента, 1 текущата лента на бащината подпрограма, 2 временната лента, която е създадена специално за това извикване на тази подпрограма, от 3 до 9 са глобалните ленти).

Петата команда е „Следващо състояние“
Стойността е в интервала [0, 1000] (тук 0 означава командата return).

Когато се извиква една подпрограма, какво записваме в стека? Записваме три неща: Къде трябва да се върнем след return (това е петата команда), коя е старата текуща лента (за да я възстановим при return) и коя е временната лента създадена специално за това извикване на тази подпрограма. Новата лента също трябва някъде да се запише. Нека това да не е в стека, а някъде другаде. При изпълнение на return съответната временна лента се унищожава.

Как попълваме таблицата

За да създадем произволна машина на Тюринг, ще трябва да запълним таблицата с размери 1000 на MaxSymbols със случайни команди. За целта първо ще кажем как избираме една случайна команда. Трябва да генерираме 6 числа (четвъртата команда има две полета). Биха могли тези числа да са равно вероятни, но ние предпочитаме, по-малките да са по-вероятни от по големите. Защо е това наше предпочитание? Защото, ако състоянията са равновероятни, то програмата ще се пръсне по много различни състояния, а ние искаме някои състояния да се използват по-често от други. Аналогично с лентите, искаме някои ленти да се използват по-често от други. Това важи и за служебните символи (за неслужебните не важи).

Как да изберем число от 0 до k с намаляваща вероятност. Например нека хвърлим монета и ако се падне ези избираме с вероятност 1/2 числото 0, в противен случай отново хвърляме монета и ако се падне ези избираме с вероятност 1/2 числото 1 и т.н. Когато стигнем до k, ако не се е паднало ези започваме отново от 0.

Вероятността от 1/2 ни дава прекалено стръмно намаляване на вероятността на следващото число. Затова ние ще използваме вероятността 1/10. Така с тази

вероятност от 1/10 ще генерираме всичките тези 6 числа. Всъщност, това което използваме е геометричното разпределение.

Забележка: Само за номера на подпрограмата ще подходим различно. Там 0 ще го вземем с вероятност 9/10 вместо с вероятност 1/10. (Нататък ще продължим пак с 1/10.) Това е така защото не искаме да се викат прекалено често подпрограми и да се пълни излишно стека. Така вероятността на командата return ще е равна на вероятността да се извика подпрограма.

Казахме как се генерира една команда (едно квадратче в таблицата). Да кажем как ще се генерира един стълб състоящ се от MaxSymbols квадратчета. Ще разглеждаме командата на тази машина като switch, който има MaxSymbols случая (case:). Когато програмираме и използваме switch ние не описваме всичките случай, а само няколко. Останалите случай ги описваме с default (тоест, останалите случай са еднакви). По-логична би била една програма, ако при голяма част от случаите командата е една и съща. Затова ние първо ще изберем случайно колко ще са различните команди в тази колона. Ще изберем по описания по-горе начин с намаляваща вероятност. След като сме избрали колко от позициите ще са различни ще изберем случайно кои ще са различните позиции (пак по-малките номера ще са по-вероятни). Накрая ще запълним позициите, които трябва да са различни различно, а останалите позиции ще запълним с една и съща команда.

Вече имаме алгоритъм за попълването на една колона и можем да попълним 1000 колони. Така ще получим първата случайна машина на Тюринг. Тази процедура е твърде тежка и затова втората машина ще я получим от първата като променим първите $m+1$ състояния (които са специални) и още 10 случайни състояния. Тази промяна е достатъчна, защото огромната част от състоянията не се използват и променяйки специалните състояния ние ще започнем да използваме други състояния. Оттам новата машина на Тюринг ще е много различна в състоянията, които използва, макар че в недостижимите състояния двете машини да са почти еднакви.

Изхвърляне на шлаката

Вече имаме процедура, с която бързо и лесно можем да генерираме 1000 тестови свята. Проблемът е, че повечето от тези светове не са интересни. От 1000 свята интересните може да се окажат само два-три. Затова ние ще искаме да изхвърлим тези светове, които не са интересни и да проведем тест с 1000 интересни свята.

Пример за свят, който не е интересен е, ако още на първия ход се влиза в тупик.

Затова, за да докажем, че един свят е интересен ще пуснем случайната стратегия да изживее един живот в него. Ще искаме през този живот стратегията да не влиза в тупик. Ще искаме да имаме поне една победа и поне една загуба. Ще искаме служебните ремита и служебните загуби да не са повече от 10. Ако това е изпълнено при този случаен живот ще приемем, че света е интересен.

Как ще направим теста от 1000 интересни свята. Първо ще инициализираме генератора на псевдо-случайни числа с числото 0. После ще генерираме случайно нулевият свят. После ще инициализираме генератора с 1 и ще получим първия свят от нулевия чрез малка промяна. Ако първия свят е интересен ще инициализираме генератора с 2 и ще създадем втория свят. Ако първия свят не е интересен ще инициализираме с 2, 3, 4, 5 и т.н. докато не получим от нулевия свят интересен първи. По този начин ще създадем 1000 интересни свята. Няма да ги помним всичките, а ще помним само масив от 1000 числа. Това ще са стойностите, с които трябва да инициализираме генератора, за да получим от предишния интересен свят нов интересен свят.

Забележка: Трябва да отбележим, че различните машини на Тюринг ние ги избираме с различна вероятност. Тази различна вероятност е различната тежест на различните машини. Това уточнение ни трябва, ако искаме да дадем точна дефиниция на глобалното IQ. Дефиницията е:

$$\text{Global IQ}(Strategy) = \sum_{TM}^{\text{TM} \in Interesting} \text{P}(\text{TM} \mid Interesting).\text{Success}(Strategy, TM)$$

Тук P(TM | Interesting) е условната вероятност машината TM да бъде избрана при условие, че света на TM е интересен. Success(Strategy, TM) е средното аритметично получено след като стратегията Strategy е изживяла един живот в света определен от машината TM. Сумата е по всички машини с 1000 състояния, чиито светове са интересни.

Това глобално IQ не може да бъде изчислено, но това е теоретичната стойност, която се опитваме да доближим с локалното IQ.

Съответно локалното IQ ще бъде равно на:

$$\text{Local IQ}(Strategy) = \sum_{i=1}^{1000} \text{Success}(Strategy, TM_i)$$

Тук TM_i е i-тата от предварително избраните тестови светове (машини на Тюринг).

Окончателна дефиниция

Дефиниция: Ще кажем, че една стратегия е AI, ако нейното локално IQ е повече от 0.7.

Тук избрахме същата стойност, която избрахме и в [3]. Тази стойност и тук и в [3] я избираме съвсем произволно. Това е все едно предварително да кажем, че ще назначим за директор на нашата фирма всеки, който реши 70% от задачите в теста. Тази летва може да се окаже твърде ниска или твърде висока и в последствие може да се наложи тази стойност да бъде променена.

Дефиниция: Ще кажем, че една програма е AI, ако стратегията, която тя играе в първите 1000 игри е AI стратегия.

Аналогия с дефиницията на гросмайстор

За да обясним още веднъж дефиницията на AI ще повторим същата конструкция, но този път ще дефинираме какво е програма играеща шах. Това вече го направихме в [4], но тъй като настоящата статия повтаря и подобрява конструкцията описана в [3], тук ще повторим и конструкцията от [4], като отразим съответните изменения.

Програма играеща шах ще наричаме такава програма, която играе като гросмайстор. За да бъде един човек гросмайстор трябва неговият ЕЛО коефициент (Elo rating system) да бъде най-малко 2500 точки. Лошото е, че ЕЛО коефициента се изчислява на базата на играта на човека с други хора. За да получим обективна оценка на това, колко добър е играча, ще заменим другите играчи с крайно множество от детерминирани компютърни програми. Ще изиграем по една партия с всеки от тези играчи и резултата ще бъде средното аритметично от резултатите на отделните партии. Ако някой играч увисне (зацикли) и не успее да довърши играта ще присъдим служебна победа в полза на неговия опонент. Аналогично, ако някой изиграе некоректен ход. Тоест програмата, която кандидатства за гросмайстор, трябва да играе само коректни ходове и да не зацикля, защото, ако го направи, ще я накажем със служебна загуба. Аналогично, това важи и за компютърните програми от крайното множеството, които сме избрали, за да оценим чрез тях нашата програма.

Кое ще е крайното множество от детерминирани компютърни програми, които ще използваме за да проведем изпита за гросмайстор? Бихме могли да вземем всички програми не по-дълги от определена дължина. Повечето от тези програми ще играят случайно, често ще зациклят и ще играят много некоректни ходове. Разумно би било да ги отсеем и да оставим само тези програми, които са интересни (програмите, които играят твърде безумно са баласт, който само утежнява теста.) Интересни ще са тези програми, които не зациклят, не играят некоректни ходове и който, освен това, играят сравнително добре.

Когато търсихме интересни светове, за да направим теста за AI, тогава използвахме метода на пълното изброяване (Brute-force search). Тук няма как да използваме този метод, защото много малко вероятно е случайно да попаднем на програма, която не зацикля и играе само коректни ходове. Затова вместо множеството на всички програми не по-дълги от определена дължина, ние ще вземем едно определено множество от програми такива, че всичките да са интересни (да не зациклят, да играят само коректни ходове и да играят сравнително добре)

Ще вземем една конкретна програма, която смята пет хода напред и разделя позициите на три вида: печеливши (такива, при които се печели до пет хода), губещи (такива, при които се губи до пет хода) и неопределени (останалите позиции). Кой ход ще изиграе тази програма? Тя ще избере случайно една от печелившите позиции. Ако няма такава ще избере случайно една от

неопределените позиции. Ако и такава няма, то тогава ще избере случайно една от губещите позиции.

Това, което описахме е една недетерминирана програма. Ако вземем детерминираните стратегии, които тази програма може да реализира, те са огромен брой (все пак крайно много, защото дължината на играта е ограничена). Всяка от тези стратегии се изчислява от безбройно много програми, но ние ще приемем, че за всяка стратегия сме избрали по една програма, която я изчислява.

Като резултат получихме едно огромно множество от програми и можем да кажем, че ЕЛО коефициента ще го сметнем на базата на играта с всичките тези програми. За съжаление тези програми са твърде много и ние не можем да изчислим този коефициент поради комбинаторната експлозия. Вместо това, ние ще изберем 1000 от тези програми и ще сметнем коефициента след изиграването на 1000 игри (по една игра с всяка от тях). Ще изберем тези програми случайно, но ще ги изберем еднократно и всеки път ще определяме ЕЛО коефициента на базата на едни и същи тестови програми.

Как от недетерминираната програма ще направим детерминирана? Много просто, вместо да избираме случаен ход, ние ще избираме псевдо-случаен. Преди да започнем играта ще инициализираме генератора на случайни числа с числото едно. Така получихме една детерминирана програма. Трябват ни 1000 такива програми. Ще инициализираме с числата от 1 до 1000 и така ще получим 1000 различни детерминирани програми (между тях може да има и еднакви, особено ако генератора на псевдо-случайни числа не е много добър). Това ще са програмите, с които ще смятаме ЕЛО коефициента.

Ще наречем една програма гросмайстор, ако получения по този начин коефициент е повече от 90%. Тази стойност я избрахме произволно. Може да се окаже, че стойността трябва да е по-голяма. Може да се наложи дори да променим тестовите светове и вместо 5 хода напред да ги накараме да изчисляват например 10 хода.

Отново ще имаме проблема със зубрачите. Може да се назубри как да се победят някой от тези 1000 програми. Става дума за детерминирани програми, а една детерминирана програма, ако я победим веднъж, можем да я побеждаваме колкото пъти си искаме като повтаряме същата партия.

Получаваме доста добра аналогия между дефиницията на програмата гросмайстор и дефиницията на AI. В единия случай говорим за ЕЛО коефициент, а в другия случай говорим за IQ. В първия случай ще играем шах срещу хиляда опонента, а във втория случай ще живеем хиляда живота в хиляда свята. Разликата е, че в първия случай срещу всеки опонент ние ще изиграем по една партия, а във втория във всеки живот ще направим по хиляда партии. Това е така, защото в първия случай правилата на играта са определени и се предполага, че програмата гросмайстор знае правилата и знае да играе (т.е. не се учи докато играе). Във втория случай AI не знае правилата на живота и има нужда от хиляда партии, за да разбере тези правила и да се научи да живее успешно.

Заключение

Тук описахме един изпит (тест), с който можем да изчислим IQ-то на произволна програма. По-точно писахме програмата, която ще проведе този изпит и ще ни каже какво е IQ-то на кандидата. Тази програма, беше описана толкова детайлно, че спокойно можем да поверим написването ѝ на някой студент, като му я дадем като курсова работа.

С помощта на тази програма бихме могли да проведем изпита за AI в рамките само на няколко минути. Толкова време ще е нужно на изпитващата програма да провери резултата от теста. Към това време трябва да добавим времето, което ще дадем на кандидата за мислене. Ако разглеждаме AI като стратегия, то тогава не си задаваме въпросът за това колко време ще мисли кандидата. Ако разглеждаме AI като програма, която изчислява AI стратегия, тогава трябва да кажем колко време даваме на тази програма, за пресмятането на една стъпка.

Нека си зададем въпроса, каква би била ползата от подобен тест. Ако някой ни даде конкретна програма, ни бихме могли да я тестваме и да кажем колко е IQ-то на тази програма. Но ние нямаме програми, които да са кандидати за AI. Тоест, ние нямаме кого да тестваме.

Едно възможно приложение на описания в тази статия тест е да го използваме за намирането на AI. Бихме могли да търсим по метода на пълното изчерпване. Разбира се, по този начин бихме могли да търсим, но не и да намерим. Поради комбинаторната експлозия, с този метод няма да стигнем далеч. Има и по интелигентен начин за търсене. Можем да направим един генетичен алгоритъм. В някой голям компютър ще създадем популация от програми кандидати за AI. За всеки от тези кандидати ще изчислим неговото IQ. Ще съчетаваме кандидатите с високо IQ, за да получим потомство с още по-високо IQ. Тези кандидати, чието IQ е много ниско ще ги убиваме, за да освободим място за по-перспективните. По този начин, по пътя на естествения подбор, ще получим програми с много високо IQ.

Генетичния алгоритъм е една възможност за намирането на AI, но по този начин ние ще получим програма, за която не знаем как работи. Ако искаме да контролираме една програма, е по-добре сами да сме си я написали вместо да сме я генерирали автоматично. Затова аз съм привърженик на директния подход при създаването на AI. Тоест, аз съм привърженик на това, че сами трябва да напишем тази програма.

Acknowledgements

Искам да благодаря на моите колеги Ivan Koychev, Anton Zinoviev и Andrey Sariev за полезните дискусии на тема „Дефиниция на AI“. Особено искам да благодаря на Ivan Koychev за това, че той съществено ми помогна при изготвянето на литературния обзор.

References

[1] Dimiter Dobrev. First and oldest application. *1993.* http://dobrev.com/AI/first.html

[2] Dimiter Dobrev. AI - What is this. *PC Magazine - Bulgaria, November'2000, pp.12-13* (in Bulgarian, also in English at http://dobrev.com/AI/definition.html).

[3] Dimiter Dobrev. Formal Definition of Artificial Intelligence. *International Journal "Information Theories & Applications", vol.12, Number 3, 2005, pp.277-285.* http://www.dobrev.com/AI/

[4] Dimiter Dobrev. Parallel between definition of chess playing program and definition of AI. International Journal "Information Technologies & Knowledge ", vol.1, Number 2, 2007, pp.196-199.

[5] Dimiter Dobrev. Two fundamental problems connected with AI. *Proceedings of Knowledge - Dialogue - Solution 2007, June 18 - 25, Varna, Bulgaria, Volume 2, p.667.*

[6] Dimiter Dobrev. Giving the AI definition a form suitable for engineers. *April, 2013.* http://www.dobrev.com/AI/

[7] Alan Turing. Computing machinery and intelligence. *Mind, 1950.*

[8] John McCarthy. What is Artificial Intelligence? November, 2007. (www-formal.stanford.edu/jmc/whatisai/)

[9] Huazheng Wang, Fei Tian, Bin Gao, Jiang Bian, Tie-Yan Liu. Solving Verbal Comprehension Questions in IQ Test by Knowledge-Powered Word Embedding. *arXiv: 1505.07909, May, 2015.*

[10] Detterman, Douglas K. A Challenge to Watson. *Intelligence, v39 n2-3 p77-78 Mar-Apr 2011.*

[11] Feng Liu, Yong Shi, Ying Liu. Intelligence Quotient and Intelligence Grade of Artificial Intelligence. *arXiv: 1709.10242, September, 2017.*

[12] Dowe, D.L., & Hernández-Orallo, J. IQ tests are not for machines, yet. *Intelligence (2012), doi:10.1016/j.intell.2011.12.001*

[13] Hernández-Orallo, J., & Minaya-Collado, N. (1998). A formal definition of intelligence based on an intensional variant of Kolmogorov complexity. Proc. intl symposium of engineering of intelligent systems (EIS'98), February 1998, La Laguna, Spain (pp. 146–163). : ICSC Press.

[14] Insa-Cabrera, J., Dowe, D. L., & Hernandez-Orallo, J. (2011). Evaluating a reinforcement learning algorithm with a general intelligence test. *In J. M. J. A. Lozano, & J. A. Gamez (Eds.), Current topics in artificial intelligence. CAEPIA 2011. : Springer (LNAI Series 7023).*