MOOC Dirbtinis intelektas

MinMax Algoritmas

Parengė: lekt. Neringa Dubauskienė

Algoritmas, kurį jums pristatysiu, naudojamas tada, kai reikia priimti sprendimą dalyvaujant priešininkui. Anksčiau minėta lošimų teorija, ekonomikos šaka, kaip tik ir nagrinėja tokias situacijas. Dirbtinis intelektas dažniausiai apsiriboja kiek supaprastintais lošimais – dviejų žaidėjų, veikiančių paeiliui, kai žinoma pilna informacija apie lošimą. Tokio lošimo pasekmė gali būti arba laimėjimas, arba pralaimėjimas arba lygiosios,. Abu žaidėjai vienu metu laimėti arba pralaimėti negali. Populiarus tokio lošimo pavyzdys yra šachmatai – vienam žaidėjui laimėjus šachmatų partiją, kitas neišvengiamai pralaimi. Kodėl šachmatai, o ne krepšinis? Nes šachmatai lengviau aprašomi, turi baigtinį skaičių galimų ėjimų, o krepšinis šiuo atveju būtu gerokai mažiau nuspėjamas, su mažiau tiksliomis taisyklėmis. Dirbtinis intelektas gali susidoroti ir su lošimais, kuriuose informacija nepilna (pavyzdžiui pokerio žaidėjai nežino, kokias kortas turi jų priešininkai) arba kur laimėjimas iš dalies priklauso nuo atsitiktinumo (stalo žaidimai su kauliukais), bet pirmiausia pabandysime sužaisti paprastesnius žaidimus.

Kokias savybes turi turėti algoritmas, kad galėtų laimėti žaidimą? Visų pirma, jis turi gebėti priimti sprendimą iš didelio kiekio galimų – jei žaidžiama šachmatais, egzistuoja iš viso 10^120 galimų ėjimų, o tai reiškia, kad joks kompiuteris negali įvertinti jų visų. Rasti sprendimą algoritmas turi greitai – priklausomai nuo to, koks žaidimas žaidžiamas, algoritmas turi ribotą laiką priimti sprendimą. Sprendimas privalo būti priimtas ir tada, kai nėra akivaizdžiai gero sprendimo arba trūksta laiko jam rasti. Visi šie apribojimai – gera treniruočių aikštelė algoritmams, kurie gali būti panaudoti ir tikroms problemoms spręsti.

Nuo ko pradedamas algoritmas? Visi žaidimai turi savo startinę poziciją, pavyzdžiui, lentoje išrikiuotos figūros šachmatams, arba nubrėžta tuščia kryžiukų nuliukų lentelė. Iš šios pozicijos judama toliau – galima pasirinkti vieną iš kelių galimų ėjimų. Ėjimus žaidėjai daro paeiliui, stengdamiesi pasirinkti kuo geresnį ėjimą. Tęsiama tol, kol pasiekiamos baigiamosios žaidimo pozicijos, tai yra, pasiekiamas laimėjimas, pralaimėjimas arba lygiosios. Kiekviename etape analizuojami visi galimi ėjimai – tai yra, sudaromas žaidimo medis. Žaidimo medžio sudėtingumas priklausys nuo to, kiek kartų žaidėjai daro ėjimus – tai yra, kiek lygių sudaro pilną žaidimų medį, ir kiek kiekviename lygyje yra galimų pasirinkimų. Net ir tokiam paprastam žaidimui kaip kryžiukai nuliukai, kuris turi devynis galimus lygius, su vis mažėjančiu galimų pasirinkimų skaičiumi kiekviename ėjime užpildant po langelį, sudarytas žaidimų medis yra per didelis tinkamai atvaizduoti vienoje skaidrėje.

Sudarius žaidimo medį, sprendimo ieškome nuo nuo medžio apačios – tai yra pasirenkime tokią galutinę būseną, kuri yra naudingiausia pirmam žaidėjui ir darykime prielaidą, kad priešininkas visada rinksis tokius ėjimus, kurie minimizuos pirmojo žaidėjo laimėjimus, tai yra, žais optimaliai. Laikydamasis šios strategijos algoritmas keliauja medžiu aukštyn iki pradinės būsenos, kuri

užtikrintų jam pergalę. Jei žaidimas nesudėtingas, ir galima išnagrinėti visą žaidimo medį per racionalų laiką – pirmasis žaidėjas visada laimės.

Tam, kad būtų galima įvertinti, kuris ėjimas naudingesnis kuriam žaidėjui, reikalinga MiniMax vertė – tai yra, kiekvienam lygmeniui priskiriama skaitinė vertė, pagal tai, kiek ta pozicija yra naudinga pirmajam žaidėjui. Šiuo atveju, žaidimas nagrinėjamas iš pirmojo žaidėjo pozicijos ir ieškoma strategijos, kuri padėtų jam laimėti. Tai atsispindi ir algoritmo MiniMax pavadinime: minimizuojami galimi priešininko laimėjimai ir maksimizuojamas galimas laimėjimas. MiniMax vertė priklauso nuo to, kokį žaidimą žaidžiame ir kaip jame skaičiuojami galimi taškai. Pažiūrėkime į teorinį žaidimo medį skaidrėje – žaidimas turi tik du lygius, ir jame jau surašyti pirmam žaidėjui galimi laimėti taškai. Daugiausia, ką galime laimėti, yra 15, tai yra didžiausias laimėjimas. Bet pasiekti šio laimėjimo mums neleis priešininkas – jo ėjimas yra paskutinis, taigi jei pasirinksime ėjimą B, priešininkas neabejotinai rinksis B2 ir mes galiausiai gausime tik 1 tašką. Ar galime pasirinkti tokį ėjimą, kad neabejotinai gautume daugiau taškų? Tą mums leidžia ėjimas A – jei priešininkas žais optimaliai, jis pasirinks A3, ir žaidimas baigsis mums turint 4 taškus. Jei norime laimėti prieš racionalų priešininką, mums reikia rasti ėjimą kuriuo gautume didžiausią mažiausią vertę – MiniMax. O kas jei priešininkas nėra racionalus? Tokiu atveju laimėsime dar daugiau!

Šiuo paprastu atveju, išnagrinėjome nuo apačios visą žaidimo medį ir kildami aukštyn, priskyrėme MinimMax vertę kiekvienam lygiui. Deja, realiems žaidimams to neužtenka.

Algoritmui paspartinti realiems žaidimams taikoma vadinamo Alfa-Beta atkirtimo strategija. Strategija iš tiesų yra labai paprasta – jei radome sprendimą, kuris neabejotinai tinka, kitų šakų galime nebenagrinėti. Pažiūrėkime į jau išnagrinėtą žaidimo medį – išnagrinėjus A, matome, kad galime tikėtis daugiausia 4 taškų. Nagrinėjant B, pirmiausia randame 2 – tai mažesnė vertė už 4, todėl nepriklausomai nuo to, kokios yra kitų ėjimų vertės, B mes nesirinksime – nėra prasmės nagrinėti likusių ėjimų. C ėjimo atveju aptinkame 6, todėl tenka patikrinti likusius ėjimus. B ėjimas žaidime niekada nebus pasirinktas, todėl ir nėra jokios prasmės jo pilnai nagrinėti, taip sutaupant laiko ir skaičiavimo išteklių. Taip atkirsti galime bet kokio gylio medžius, kai kuriais atvejais ženkliai sutrumpinant sprendimo paieškos laiką. Alfa Beta pavadinimą ši strategija gavo dėl to, kad nagrinėjant žaidimo medį visada saugomos dvi reikšmės – Alfa, tai yra geriausias garantuotas laimėjimas pirmam žaidėjui, ir Beta – geriausias garantuotas laimėjimas antrajam žaidėjui. Nagrinėjant žaidimų medį šios vertės nuolat atnaujinamos, ir atmetamos tos šakos, kuriose vertės yra mažesnės. Deja, šios strategijos sėkmė priklauso ir nuo to, kokia tvarka nagrinėjami ėjimai – jei pirmiausia nagrinėjami „blogi“ ėjimai, Alfa beta vertės bus nuolat atnaujinamos ir teks pilnai išnagrinėti visas šakas. Žinant kokia tikimybė, kad ėjimas bus geresnis ar blogesnis, galima pradėti nuo tikėtinai gerų ėjimų.

O kas, jei negalime išnagrinėti viso žaidimo medžio per duotą laiką? Tada teks nagrinėti žaidimo medį iki tokio gylio, kiek laiko tam turime. Pasieksime ne galutines pozicijas, bet numatysime keletą ėjimų į priekį – šachmatų didmeistriai paprastai tai ir daro. Nutraukus žaidimo nagrinėjimą anksčiau, neturime tikslios MinMax vertės, todėl reikia kito būdo ėjimams įvertinti. Tam pasitelkiama įvertinimo funkcija. Ši funkcija apskaičiuoja apytikslę MinMax vertę, remdamasi informacija apie iki šiol išnagrinėtus ėjimus. Ši įvertinimo funkcija yra gana komplikuota –

neužtenka įvertinti tik to, kokią vertę turi žaidimo figūra ar kauliukas, reikia atsižvelgti ir į jų poziciją vienas kito atžvilgiu. Ši vertė bet kokiu atveju bus netiksli, bet dažnai tokio įvertinimo pakanka, kad sprendimas būtų priimtas.

Kitas svarbus klausimas, kurį reikia atsakyti kuriant algoritmą – kada nutraukti paiešką? Žinoma, išnagrinėto žaidimų medžio lygis priklausys nuo turimo laiko, ir galima tiesiog nutraukti paiešką tada, kai baigiasi laikas, bet kas jeigu vienu lygiu giliau pamatysime, kad situacija žaidime apsiverčia aukštyn kojom? Tiesa pasakius, tai neblogas kriterijus paieškai užbaigti – kai pasiekiama situacija, kuri ženkliai keičia žaidimo eigą, paieška nutraukiama. Tai negarantuoja laimėjimo, bet ženkliai padidina galimybes.

Tai ne vienintelės strategijos patobulinti MinMax algoritmui, bet tai jau nebe šių mokymų sritis. MinMax algoritmas – nesunkiai suprantamas ir plačiai naudojamas dirbtinio intelekto algoritmas. Tikiuosi, kad jums bus lengviau suprasti, kokius apribojimus tokio tipo algoritmai turi ir kuo remdamiesi jie priima sprendimus.