Monte Karlo pretraga u igri Connect-four

Slobodan Gužvica

14. jun 2021.

Sažetak

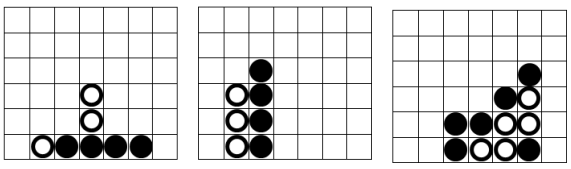
U nekim poteznim igrama gotovo je nemoguće pretražiti celo stablo, za primer možemo uzeti šah gde je donja granica kompleksnost stabla 10^{120}, što je više nego broj atoma u univerzumu, takođe postoje igre u kojima je kompleksnost jos veća. Monte Karlo pretraga je jedna od veoma bitnih tehnika za pretragu, koristi se u malopre navednim slučajevima uz kombinaciju neuronskih mreža.

Game	Board size	State space	Game tree size
Go	19 x 19	10^{172}	10^{360}
Chess	8 x 8	10^{50}	10^{123}
Checkers	8 x 8	10^{18}	10^{54}

U radu je opisana implementacija Monte Karlo algoritma na igri connect-four.

1 Uvod

Dva igrača naizmenično ubacuju tokene različtih boja na vrh table dimenzija 7×6 koji zatim propadaju do dna ili do prve zauzete ćelije, pobednik je onaj igrač koji prvi sklopi 4 u koloni, vrsti ili dijagonali.



Connect-four spada u igre sa potpunim informacijama, rešena je još 1988. godine i igrač koji igra prvi ima pobedničku strategiju. Kada bi igrali dva savršena igrača, od prvog poteza bi zavisio ishod partije. Ako se odigra u sredini prvi igrač bi uvek pobedio, ako se odigra na ivicama table(prvo i poslednje mesto i drugo i pretposlednje) drugi igrač bi pobedio, a u ostalim slučajevima igra bi se završila nerešeno.

2 Monte Karlo

Najveća mana minimax algortima je potreba za proširivanjem celog stabla. Za igre sa visokim faktorom grananja poput šaha to dovodi do ogromnih stabala i tako sigurnih neuspeha. Postoji nekoliko načina da se ovaj problem reši, a to je da se ograniči pretraživanje nekom dubinom i da se uvede funkcija koja procenjuje trenutno stanje (Iskusniji igrač šaha na osnovu date pozicije može da proceni ko pobeđuje). Tada se javlja novi problem, a to je da nam nije zagarantovan najbolji potez. Još jedan način za prevazilaženje problema sa veličinom stabla je skraćivanje stabla pomoću alfa-beta pruninga. Kombinaciju ova dva metoda koristi jedan od najpoznatijih šahovskih robota Stockfish.

U Monte Karlo algoritmu sledeći, najobećavajući potez izračunava se na drugačiji način. Simulira se partija igre mnogo puta i na osnovu rezultata simulacije pokušava da se predvidi potez koji najviše obećava. Za razliku od minimaxa Monte Karlo će uvek pretražiti stablo u dubinu do kraja dok u širinu neće duboko zalaziti. Glavno pitanje ostaje kako tokom simulacije izabrati sledeći potez. Potezi se izbiraju na osnovu funkcije koja uzima trenutno stanje igre i vraća sledeči potez. U praksi ta funkcija treba da bude dizajnirana tako da njeno računanje bude veoma brzo. Uobičajno se koristi da ta funkcija bira nasumičan potez. Za primer ćemo uzeti Alpha Zero, šahovskog robota, koji koristi unaprđenu verziju Monte Karlo algoritma gde umesto nasumičnog poteza koristi neuronske mreže da bi odredio sledeći potez.

Ove dve tehnike pretraživanja sukobile su se u eksperimentu kompanije DeepMind gde su Stockfish 8 i Alpha Zero odigrali 1000 partija šaha sa različitim zadatim otvaranjima. Alpha Zero je u velikoj većini partija pobedio, 155 pobeda 839 nerešenih partija i samo 6 poraza, što pokazuje da će uskoro tradicionalni šahovski roboti biti zamenjeni novim robotima koji koriste Monte Karlo pretragu, neuronske mreže i mašinsko učenje.

3 Monte Karlo i Connect-four

U ovom odeljku pokazaćemo implementaciju Monte Karlo algoritma u igri Connect-four u programskom jeziku Java. Tokom simulacije na slučajan način biraćemo sledeći odigran potez.

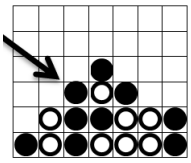
Za svaki od sedam poteza ako je potez moguć simuliraj 10 000 partija, ako je u simulaciji došlo do pobede monteCarlo[i] se povećava za jedan, tj. smanjuje ili ostaje isti ako je krajnji ishod poraz ili nerešena partija. Na kraju se čuvaju najveća i najmanja vrednost.

```
public int monteCarlo() {
        for(int i=0;i<7;i++) {
                if(monteCarlo[i]==10000 || monteCarlo[i]==-10000 )
                        continue;
                for(int j=0;j<10000;j++) {
                        monteCarlo[i]+=monteCarlo(i);
                }
        }
        int largest = monteCarlo[0], indexMax = 0;
        int smallest =monteCarlo[0], indexMin = 0;
        for (int i = 1; i < monteCarlo.length; i++) {
                if (monteCarlo[i] > largest ) {
                        largest = monteCarlo[i];
                        indexMax = i;
                        }
                if(monteCarlo[i]<smallest) {
                        smallest=monteCarlo[i];
                        indexMin=i;
                }
        }
}
```

Ostalo je još da vidimo kako se partija simulira, dat je najbitniji deo koda, a metode koje se koriste su lake za implementaciju.

```
private int monteCarlo(int row) {
        play(row);
        Random r = new Random();
        while(!isOver()) {
                play(r.nextInt(7));
        }
        return winner  // -1 0 or 1
}
```

U ovako implemntiranom algoritmu može doći do loših rezultata, to se najbolje vidi na sledećem primeru.



Robot će odigrati ponekad ovakav potez kojim preti da spoji četiri ista, ali isti taj potez vodi u poraz. To je zbog toga što se u simulacijama igraju nasumični potezi, pa ovakav potez može da ima dosta dobru evaluaciju iako je zapravo jako loš. Da bi izbegli ovakve situacije za svaki mogući potez pustićemo minimax algoritam na maloj dubini da proverimo da li odigran potez vodi do zasigurnog poraza, ako vodi onda ga uopšte nećemo simulirati. Pre simulacije u niz upišemo vrednosti 10 000 ili -10 000 ako potez vodi do zasigurnog poraza.

```
public int monteCarlo() {
        for(int i=0;i<7;i++) {
                monteCarlo[i]=-minMax(depth,board,turn*-1);
        }
}
```

4 Zaključak

Ovakva implemntacija robota koji igra Connect-four nije savršena, robot se može pobediti, ali će robot poteze odigrati sa velkom brzinom. Ove osnovne

koncepte prikazane u ovom radu koriste svi napredniji roboti poput Alpha Go-a i Alpha Zero-a uz još neke dodatke. Dalje unapređenje ovog robota može da se postigne korišćenjem neuronskih mreža gde bi očekivani rezultat bio da robot igra savršeno.

Literatura

[1] Bradberry, Jeff (2015-09-07). "Introduction to Monte Carlo Tree Search".

[2] Vincent, James (December 6, 2017). ĐeepMind's AI became a superhuman chess player in a few hours, just for fun". The Verge. Retrieved December 6, 2017.

[3] Silver, David; Hubert, Thomas; Schrittwieser, Julian; Antonoglou, Ioannis; Lai, Matthew; Guez, Arthur; Lanctot, Marc; Sifre, Laurent; Kumaran, Dharshan; Graepel, Thore; Lillicrap, Timothy; Simonyan, Karen; Hassabis, Demis (December 5, 2017). "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm". arXiv:1712.01815 [cs.AI].