Celtrin programerski izziv 2014:
Jackpot - koliko denarja priigra tvoj algoritem?

Delovno poročilo

Gašper Žgajnar (63090178)

24. november 2014

1 Uvod

Za Celtrin programerski izziv 2014 sem izbral nalogo z naslovom *Jackpot - koliko denarja lahko priigra tvoj algoritem?* Cilj danega problema je bila implementacija algoritma, ki izbira med n igralnimi avtomati, ki vsak z določeno verjetnostjo vrača 1 ali 0, tako da po m poskusih skupaj dobi čim večjo vsoto pozitivnih odzivov igralnih avtomatov (tj. odzivov z vrednostjo 1).

2 Potek reševanja

Problema sem se lotil postopoma. Najprej sem izdelal ogrodje programa - funkcije, ki kličejo strežnik s podanimi parametri. Na strežnik se na začetku poda zahteva za število avtomatov in število potegov, ki so na voljo za želen primer igre.

2.1 Naključni generator

Nato sem izdelal najpreprostejši algoritem - generator naključnega avtomata; za vsak poteg generira naključno številko avtomata. S tem sem dobil tudi osnovno število pozitivnih odzivov, ki naj bi jih moj naslednji, inteligentnejši algoritem presegel.

2.2 Algoritem 1

Pri naslednjem postopku sem izbiro avtomatov razdelil na dva dela. Na začetku se algoritem poskuša za vsak avtomat naučiti verjetnost s katero generira vrednosti 1 in 0. To naredim tako, da za prvih 10% primerov izmenično kličem vsak avtomat. Po vseh začetnih klicih, izračunam verjetnosti in izberem avtomat z najvišjo ter ga nato v preostalih 90% klicev uporabim.
Ta postopek se izkaže za boljšega od naključnega generatorja v primerih, ko se verjetnosti avtomatov ne spreminjajo. V nasprotnem primeru pa sta postopka enakovredna.

2.3 Algoritem 2

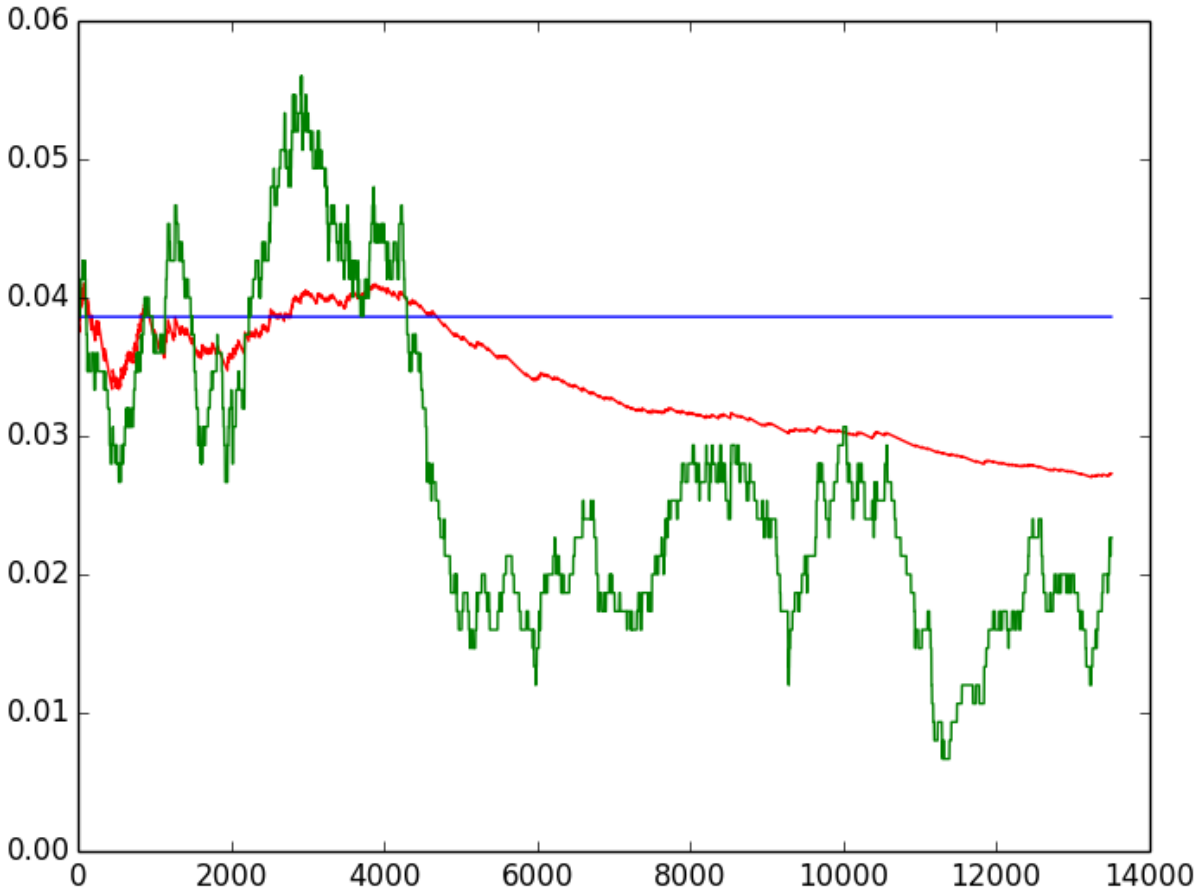
Da bi odpravil težavo spreminjanja verjetnosti avtomatov, sem algoritem 1 popravil tako, da se računanje verjetnosti avtomatov večkrat ponovi. Določil sem dva parametra - število izvajanj za določitev verjetnosti avtomatov in število izvajanj, ko se te verjetnosti uporabijo. Algoritem tako v iteracijah izvaja najprej računanje verjetnosti in nato uporabo avtomata z najvišjo verjetnostjo vse dokler ima še na voljo potege. Algoritem sem testiral pri dveh parametrih - 10% za določanje verjetnosti in 20% za uporabo teh ter v drugem primeru vrednosti 10% in 30%. Rezultati so bili boljši v drugem primeru; pri obeh primerih pa algoritem vrača boljše rezultate od algoritma 1.

2.4 Algoritem 3

Problem algoritma 2 je v statični določitvi ponovnega računanja verjetnosti avtomatov. To sem želel popraviti v algoritmu 3. Ideja je bila, da na začetku izračunam avtomat z najvišjo verjetnostjo (tako kot pri algoritmu 1). Nato ta avtomat uporabljam in ob vsakem potegu, iz shranjene zgodovine rezultatov potegov za ta avtomat, izberem n zadnjih in izračunam *trenutno* verjetnost, ki jo nato primerjam s tisto izračunano na žačetku. Če je razlika med njima dovolj velika (oz. prevelika) se vse verjetnosti pri vseh avtomatih resetirajo in pričnejo ponovno računati. Tako naj bi se detektirala sprememba v verjetnostih avtomatov (oz. sprememba verjetnosti trenutnega avtomata, saj imamo realne podatke le zanj). V teoriji naj bi ta rešitev delovala, v praksi pa je rezultat drugačen. Algoritem se izkaže za slabšega kot algoritem 2. Problem je v premajhnem številu potegov, saj je varianca prevelika in je težko določiti mejo pri kateri se spremeni verjetnost avtomata. Če sem število zadnjih n potegov povečav, je algoritem deloval bolje, vendar pa sem nato moral povečati tudi število vseh potegov. Spremembo verjetnosti skozi čas prikazuje slika **??**.

3 Program

Vso programsko kodo sem napisal v jeziku Python. Program se nahaja na repozitoriju na povezavi `https://bitbucket.org/gzgajnar/celtrachallenge2014.git`



Slika 1: Modra črta prikazuje izračunano verjetnost avtomata na začetku igre po n naključnih izbirah. Rdeča črta prikazuje spreminjane verjetnosti avtomata skozi čas - verjetnost se izračuna na podlagi celotne zgodovine klicev. Zelena črta prikazuje spreminjanje verjetnosti avtomata skozi čas - verjetnost se izračuna na podlagi zadnjih 1500 klicev. Sprememba verjetnosti avtomata se zgodi po 5000 klicih. Graf prikazuje idealen primer (oz. najbolj idealen primer, ki sem ga lahko dobil). Na začetku ima avtomat verjetnost 0.04 po 5000 klicih pa se spremeni na 0.02.