Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Celtrin programeski izziv

Multi-armed bandit problem

Avtor: Žiga Resnik

Ljubljana, 23. 11. 2014

Navodila za zagon

Povezava do repozitorija: https://github.com/polysalama/celtra
Gitbash ukaz: git clone https://github.com/polysalama/celtra

a) Kot izvršljiva Windows 7/8/8.1 datoteka:

Povezava do izvršljive datoteke: https://github.com/polysalama/celtra/raw/master/bandit.7z

Izvršljiva datoteka deluje samo na **64 bitni verziji** operacijskega sistema!

Iz repozitorija povlecite bandit.7z arhiv in ga razširite. Odprite ukazno vrstico in se prestavite v imenik kamor ste razširili arhiv. Program zaženete z ukazom:

>>>bandit.exe http://<ime_domene>/<št._primera>

Program bo po končanem izvajanju izpisal število uspešnih potegov.

b) Kot Python skripta:

Povezava do skripte: https://github.com/polysalama/celtra/raw/master/source/bandit.py

Pred uporabo je potrebno namestiti:
- Python 3.4 https://www.python.org/download/releases/3.4.0/
- NumPy 1.9.1 http://www.numpy.org/
- SciPy 0.14 http://www.scipy.org/

Najprej namestite Python. V PATH dodajte pot do Python34 in Python/Scripts direktorija. Za namestitev NumPy-a in SciPy-a v ukazni vrstici zaženite:

>>>pip install numpy
>>>pip install scipy

Za Windows platformo lahko SciPy in NumPy dobite tudi v obliki primerni za klasično nameščanje:

https://dl.dropboxusercontent.com/u/32392228/numpy1.9.1_scipy0.14_win_amd64_py3.4.7z

https://dl.dropboxusercontent.com/u/32392228/numpy1.9.1_scipy0.14_win32_py3.4.7z

Python skripta se nahaja v source mapi repozitorija. Za zagon programa v ukazno vrstico vnesite:

>>>python bandit.py http://<ime_domene>/<št._primera>

Program bo po končanem izvajanju izpisal število uspešnih potegov.

c) Kot Python skripta z Anacondo

Anaconda je verzija Pythona, ki vsebuje potrebne dodatke za Python. Iz http://continuum.io/downloads povlecite 3.4 verzijo za Windows, Linux ali Mac OS. Po potrebi dodajte v PATH pot do Anaconda3/ in Anaconda3/Scripts direktorija. Program poženite z:

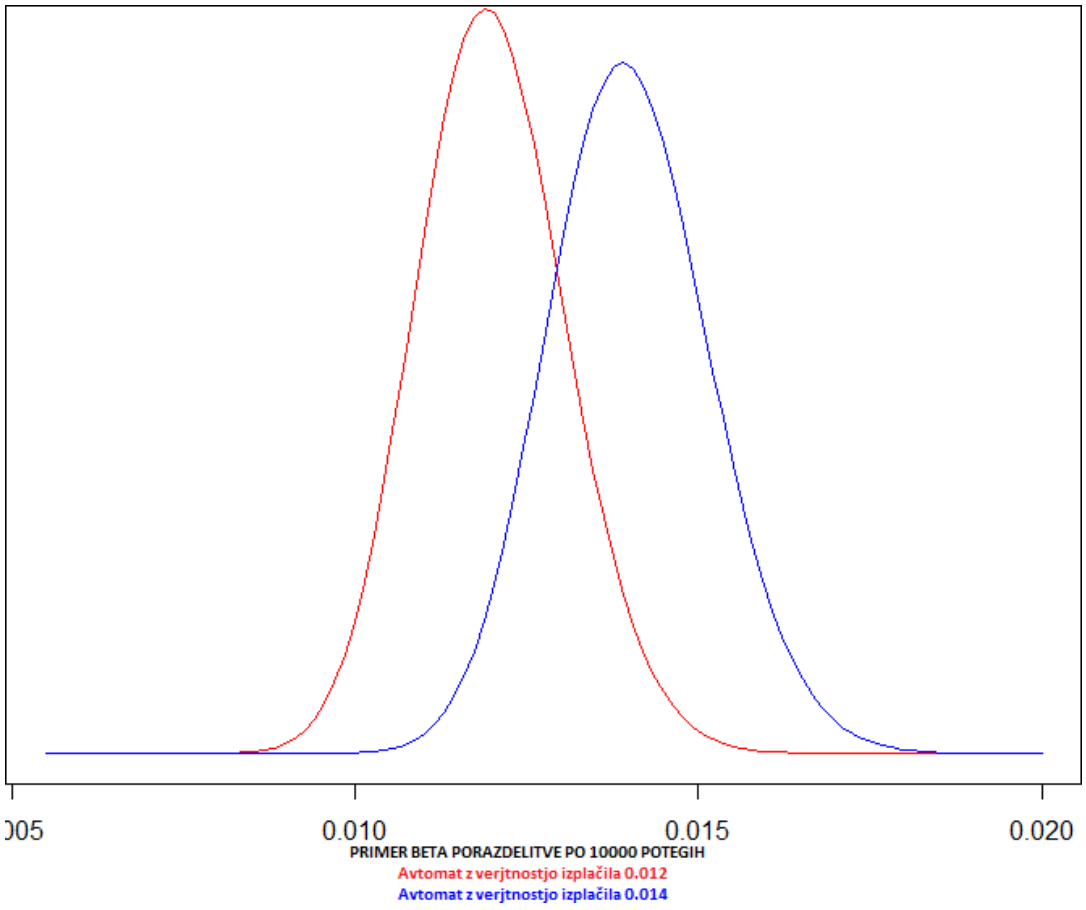
>>> python bandit.py http://<ime_domene>/<št._primera>

Cilj in pristop k reševanju problema

Cilj iziva je bil maksimizirati dobiček pri igranju igralnih avtomatov, ki imajo različne možnosti izplačila. Gre za tako imenovani multi-armed bandit problem.
Odločil sem se za statistično metodo reševanja, ki v primerjavi z požrešnimi algoritmi daje boljše rezultate. Algoritem temelji na dejstvu, da lahko za vsak avtomat zgradimo apriorno verjetnostno distribucijo iz podatkov, ki jih pridobimo pri igranju avtomata.
Za apriorno porazdelitev je najbolj primerna beta porazdelitev. Obliko beta distribucije določata parametra α (=uspešni potegi) in β (=neuspešni potegi). Na začetku sta parametra enak 1, zato ima distribucija uniformno obliko in je enakovredna enakomerni porazdelitvi. Več kot odigramo iger s posameznim avtomatom bolj se beta distribucija približuje pravi verjetnosti izplačila avtomata.
Pred vsakim potegom zgradimo za vsak avtomat njegovo distribucijo in nato iz nje vzorčimo. Nato izberemo avtomat, ki ima najvišjo vrednost vzorca in ga igramo. Ob zmagi povišamo njegov števec zmag. Na kratko:

1.) Za vsak avtomat zgradi beta(α, β)
2.) Vzorči iz beta porazdelitve posameznega avtomata
3.) Igraj avtomat z najvišjo vrednostjo vzorca
4.) Avtomatu po potrebi povečaj števec zmag
5.) Vrni se na prvi korak

Algoritem deluje dovolj dobro tudi za primere, kjer se verjetnost izplačila spreminja s časom, saj algoritem nikoli ne neha preizkušati ostalih avtomatov. Če avtomat začne izgubljati se beta porazdelitev ustrezno posodobi. Stopnjo raziskovanja se sicer lahko poveča z omejevanjem α in β, s čimer se posledično poveča standardni odklon beta porazdelitve. S tem bi se pri vzorčenju povečal razpon možnih vzorcev, tako bi se povečala možnost, da pri vzorčenju zmaga kakšen drug avtomat. Pri temu se seveda lahko nekoliko pokvari uspešnost za primere kjer se verjetnosti ne spreminjajo. Za omejevanja odklona se sam nisem odločil, ker pri testiranju z omejenim in ne omejenim odklonov ni bilo dovolj velike razlike.



Izbira programskega jezika

Za Python sem se odločil zaradi prenosljivosti med operacijskimi sistemi in razširitve SciPy, ki je vsebovala potrebne elemente za mojo rešitev.

Čas reševanja problema

Večino časa sem porabil za prebiranje raziskav in spletnih strani na temo reševanja multi-armed bandit problema. Sam program sem zaradi njegove kratkosti napisal hitro. Nato sem še testiral ali je smiselno omejevati odklon beta porazdelitve, kar je zaradi števila potegov in avtomatov trajalo približno dva dni.