Proiect, Tema 1

Algoritm online, frecventist şi Bayesian, de selectare şi afişare a reclamelor publicitare în marketing-ul internet

Prezentarea teoretică.

Această tema de proiect are ca scop experimentarea simulării aleatoare a acţiunii de alegere eficientă a reclamelor publicitare, în *publicitatea online* `http://en.wikipedia.org/wiki/Online_advertising`.

Simularea se bazează pe o metodă de învăţare (din machine learning) ce se îmbunătăţeşte pe parcursul rulării (metoda de învăţare se numeşte *reinforcement learning*, adică învăţarea prin întărire).

Proiectul are două părţi: în prima parte algoritmul de învăţare se bazează pe modelul frecventist al teoriei probabilităţilor, iar în a doua pe modelul Bayesian.

În internet marketing providerii de conţinut şi brokerii de publicitate online (cei care facilitează contractul dintre provideri şi companiile ce doresc să-şi facă publicitatea produselor) trebuie să decidă pentru fiecare vizitator al unei pagini WEB ce anume reclamă să selecteze, dintr-o bază de k reclame, pentru a fi afişată, în scopul de a atrage click-uri (cele două părţi obţin venituri proporţionale cu numărul de click-uri; acest model de contract se numeşte *pay per click* `http://en.wikipedia.org/wiki/Pay_per_click`).

Alegerea reclamei potrivite şi afişarea ei, nu se face manual de către o persoană, ci de către un algoritm subiacent.

Algoritmul frecventist:

Presupunem că există un contract de online marketing de afişare pe un site a k reclame, $1, 2, 3, \dots k$. Iniţial potenţialul lor de a atrage click-uri este necunoscut.

• Notăm cu pr_j probabilitatea de succes a reclamei j (adică probabilitatea ca reclama să fie click-uită la o afişare). In realitate aceste probabilităţi nu sunt cunoscute, dar algoritmul "le învaţa", adică le deduce monitorizând atitudinea userilor fată de reclamele selectate şi afişate.

• X_j este variabila aleatoare Bernoulli ce indică rezultatul unei afişări a reclamei j. 1 codifică succesul (click pe reclamă), iar 0 eşecul (navigatorul WEB nu clickuieşte reclama afissată). Deci $X_j \sim Bern(pr_j)$, $j = 1, 2, \dots, k$. Cele k variabile $X_1, X_2, \dots X_k$, sunt independente.

• Alegerea câte unei reclame, afişarea ei şi contorizarea click-urilor se face succesiv pentru fiecare utilizator $1, 2, \dots N$ ($N > 500$).

Deoarece probabilităţile de succes ale reclamelor sunt necunoscute, algoritmul subiacent, de alegere, învată pe parcurs (deduce pe bază de observaţii) care reclamă are potenţial mai mare de a atrage clickuri, şi în funcţie de metoda învăţată decide care reclamă să fie afisătă următorului user.

Mai precis, necunoscând distribuţia de probabilitate a click-urilor, se contorizează click-urile după fiecare afisăre a unei reclame şi se calculează media numărului de click-uri înregistrate în afiările precedente (adică media aritmetica a nr de click-uri). De exemplu dacă reclama a fost afişată de 5 ori şi rezultatul afişărilor a atras respectiv $0, 1, 1, 0, 1$ click-uri, atunci numărul mediu de clickuri atrase este $3.0/5 = 0.6$.

Strategia de selectare pentru afisăre a unei reclame din cele k este următoarea:

• Se setează un parametru `alpha` foarte mic, de exemplu $\texttt{alpha} = \texttt{0.03}, \texttt{0.05}, \texttt{0.1}$.

Pentru fiecare vizitator al paginii algoritmul alege cu probabilitatea $\texttt{1} - \texttt{alpha}$, reclama care s-a dovedit a avea nr mediu maxim, de click-uri primite în afisările anterioare, şi cu probabilitatea α (deci o probabilitate mică) alege uniform oricare dintre cele k reclame pentru a fi afişată.

Această a doua alegere se face în scopul de a testa efectul reclamelor mai puţin click-uite asupra noilor useri, care s-ar putea să aibă alte gusturi, interese, decât precedenţii.

După ce alegeţi conform acestei proceduri reclama ce este afişată pentru fiecare user din cei N, veţi constata că algoritmul alege de fapt reclama care aduce click cu o probabilitate mai mare decât celelalte. Explicaţia teoretică pentru această alegere este următoarea:

Variabilele aleatoare Bernoulli, $X_1, X_2, \dots, X_k$, independente şi identic distribuite după legea:

$$X_j = \begin{pmatrix} 1 & 0 \\ pr_j & 1 - pr_j \end{pmatrix}, \quad j = 1, 2 \dots, k$$

ce dau rezultatul afişarii reclamei j au valoare a medie:

$$M(X_j) = 1 \cdot pr_j + 0 \cdot (1 - pr_j) = pr_j$$

Prin urmare când se afişează reclama cu cel mai mare număr mediu experimantal de clickuri se afişează de fapt reclama cu cea mai mare probabilitate de succes (de a atrage click-uri).

Date de intrare. Variabile ce trebuie definite:

• setaţi numărul k de reclame ce urmează a fi afişate pe pagina WEB. La început, în perioada de testare setaţi $k = 3$, apoi puteţi să măriţi numărul de reclame, la 5,8, etc.

• setaţi nr N, de vizitatori ai paginii pe care se vor afişa reclamele. Iniţial $N = 500, 1000$, ETC.;

• setaţi parametrul `alpha` definit mai sus;

• pentru fiecare reclamă j, din cele k, daţi într-un vector pr de k coordonate, a j-a coordonată fiind probabilitatea pr_j de a fi click-uită. De exemplu declaraţi `double * pr`, alocaţi pentru pr, k locaţii de lungime totală `k * sizeof(double)` bytes. Concret pentru $k = 3$ reclame, puteţi seta $p[0] = 0.63, p[1] = 0.47, p[2] = 0.21$ (atenţie suma probabilităţilor NU trebuie să fie 1, pentru că cele 3 probabilităţi sunt parametri pentru 3 variabile Bernoulli, independente!!!!)

• pentru fiecare reclamă, j, contorizaţi numărul de selecţii pentru afişare. Declaraţi `int *nrafis;`, alocaţi o zonă de memorie `k*sizeof(int)` pentru nr de afisări şi iniţializaţi pe 0; Astfel `nrafis[j]` indică nr de afişari ale reclamei j, $j = 0, 1, \dots k - 1$

• pentru fiecare reclamă j contorizaţi numărul mediu de clikuri primite în cele `nrafis`[`j`] afisări. De exemplu declaraţi `double *nrmclicks;` şi alocaţi memorie, corespunzător;

Numarul mediu de click-uri se poate calcula recursiv conform următoarei observaţii:

Dacă avem media aritmetică a n-1 numere, $m_{n-1}\frac{x_1 + x_2 + \cdots + x_{n-1}}{n-1}$ şi a n numere, $m_n = \frac{x_1 + x_2 + \cdots + x_{n-1} + x_n}{n}$, atunci putem exprima:

$$m_n = ((n-1)m_{n-1} + x_n)/n$$

Funcţii utile pentru simulare

Definiţi:

• o functie ce simulează o variabilă aleatoare Bernoulli de parametru p:

`int Bernoulli(double p);`

• O funcţie `int SimulUnifD (k)` ce simulează o variabilă aleatoare uniform distribuită pe mulţimea reclamelor, $\{0, 1, 2 \ldots, k-1\}$;

• o functie `indexMax(double *x, L)` ce returnează indicele j ce dă poziţia elementului maxim din vectorul `x` de lungime `L`.

• o funcţie ce selectează o reclamă pentru afişare:

`int selectAd(double alpha, int k, int imax);`

Această funcţie selectează reclamă conform strategiei:

```
u=urand();
if (u<alpha) return SimulUnifD(k);
else return  imax;// imax=codul reclamei cu nr mediu maxim
                  // de click-uri inregistrate;
```

Simularea ar trebui să parcurgă următoarele etape:

```
for user=1:N{  //pentru fiecare vizitator al paginii
1. determina  reclama cu nr mediu maxim de click-uri inregistrate;
2. selecteaza o reclama, conform strategiei enuntate si o afiseaza; fie aceasta reclama j;
3. incrementeaza contorul pentru nr de afisari ale reclamei j;
4. simuleaza variabila  Bernoulli asociata reclamei j;//rezultatul poate fi click=1 sau 0
5. actualizeaza numarul mediu de click-uri ale reclamei j;

}//end for
```

Poate vă întrebaţi cum se determină reclama cu nr mediu de click-uri înregistrate, la începutul simulării, pentru `user=1`. Initial numărul mediu de click-uri ale fiecărei reclame este 0, `nrmclicks[j]=0`, $\forall j = \overline{0, k-1}$. Deci se determina indicele celei mai mari valori. In functie de cum implementaţi determinarea indicelui celui mai mare element, va fi returnată una din reclamele $0, 1, 2, \ldots, k-1$. Prin urmare ”se descurcă”.

La sfârşitul simulării afişati nr de reclame, nr de useri, vectorul probabilităţilor de succes, `pr` si vectorul `nrmclicks`, ale cărui coordonate dă nr mediu de click-uri înregistrate pentru fiecare reclamă.

Dacă implementarea a fost făcută bine şi nr de useri este suficient de mare, de ex $N > 1000$, atunci coordonatele celor doi vectori ar trebui să fie foarte apropiate. De ce sunt apropiate? Pentru că aşa cum am ilustrat mai sus valoarea medie teroretică $M(X_j) = pr_j$, iar valoarea medie experimentală este media aritmetică a valorilor de observaţie asupra variabilei X_j, adică ceea ce noi am numit numărul mediu de click-uri obţinute în afişările reclamei j.

Afisaţi si valorile absolute ale diferentelor $\texttt{pr}[\texttt{j}] - \texttt{nrmclicks}[\texttt{j}]$, $j = \overline{0, k-1}$, pentru a ilustra succesul simulării.

Întro simulare pe care am efectuat-o pentru $k = 4$ reclame, $N = 1000$ useri, $\texttt{alpha} = \texttt{0.15}$ şi probabilităţile potenţiale pr, fiind $0.53, 0.2, 0.38, 0.13$, am obţinut numărul mediu de click-uri pe fiecare reclamă, `nrmclicks` ca fiind respectiv $0.53096179, 0.26666667,$ $0.37179487, 0.125$. Valorile absolute ale diferentîelor sunt: $0.00096179, 0.06666667,$ $0.00820513, 0.005$.

Singura deosebire dintre această simulare şi experimentul real de monitorizare a click-urilor pe reclamele afişate pe o pagină WEB este doar că noi am dat probabiliţătile de succes *pr*, pentru a putea simula acordarea sau nu a click-urilor. În realitate, aceste probabilităţi sunt necunoscute algoritmului şi deci brokerului de publicitate online, dar aşa cum observaţi din simulare ele sunt învăţate succesiv, din mers.

Algoritmul Bayesian

Abordarea Bayesiană încearcă să măsoare rata de succes a reclamelor pornind de la ideea că această rată nu e caracterizată de o valoare fixă ci de o variabilă aleatoare ce descrie valorile potenţiale ale succesului. Mai precis se consideră că parametrii necunoscuţi, pr_j, $j = \overline{0, k-1}$, ce reprezintă probabilităţile de succes ale reclamelor sunt variabile aleatoare ce au densitatea de probabilitate $f_j(x)$, nenulă pe intervalul (0,1). Şi anume cele k variabile aleatoare au distribuţia Beta, de densitate de probabilitate depinzând de doi parametri reali, pozitivi, a, b:

$$f(x; a, b) = \begin{cases} \dfrac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} x^{a-1}(1-x)^{b-1} & x \in (0,1) \\ 0 & \text{în rest,} \end{cases}$$

unde Γ (gamma) `http://en.wikipedia.org/wiki/Gamma_function` este funcţia definită prin:

$$\Gamma(x) = \int_0^\infty e^{-t} t^{x-1} dt, \quad x > 0$$

Pachetele de programe dedicate calculului numeric pun la dispoziţie o funcţie care calculează valori $\Gamma(x)$, $x > 0$, ale acestei funcţii.

În funcţie de valorile lui $a, b > 0$ graficul distribuţiei Beta poate avea diverse forme. Ilustrăm mai jos (Fig. reffig1) câteva cazuri.

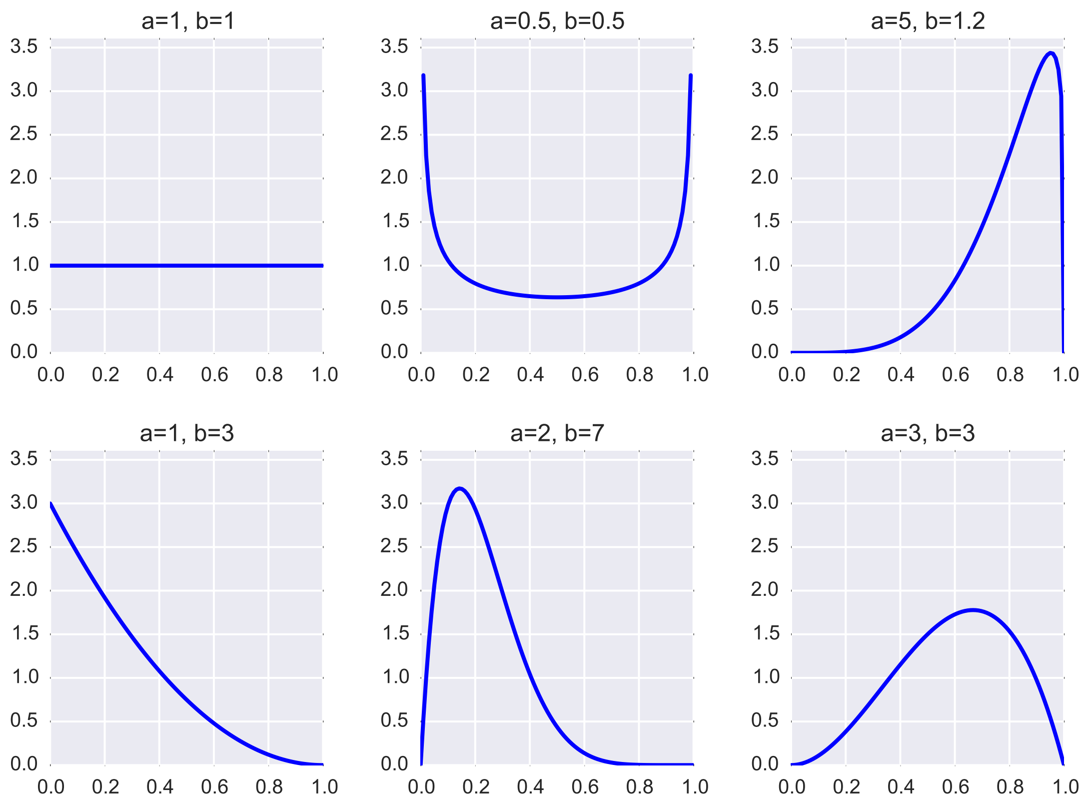

Fig. 1: Densităţi de probabilitate Beta(a,b), pentru diverse valori ale lui a şi b.

Distribuţia Beta(a,b) este o distribţie "celebră" în machine learning pentru că se poate controla prin cei doi parametri forma graficului densităţii şi deci se pot modela o serie de variabile ce iau valori într-un interval mărginit.

Pentru $a = 1, b = 1$ avem:

$$f(x; 1, 1) = x^0(1 - x)^0 = 1, \quad \forall\, x \in [0, 1)$$

Prin urmare Beta(a=1,b=1) coincide cu distribuţia uniformă pe $[0, 1)$, Unif[0,1).

Valoarea medie a unei variabile aleatoare $X \sim Beta(a, b)$ este $\dfrac{a}{a+b}$. Cu cât a şi b sunt mai mari, cu atât mai concentrată este distribuţia de probabilitate în jurul mediei (vezi Fig. 2), adică probabilitatea ca variabila aleatoare Beta să ia valori într-un interval suficient de mic, ce include media este foarte mare:

Precizăm că distribuţia Bernoulli de parametru pr se poate exprima astfel

$$P(X = x) = pr^x(1 - pr)^{1-x}$$

Într-adevăr, dacă $x = 1$=succes avem $P(X = 1) = pr$, iar dacă dacă $x = 0$=eşec, atunci $P(X = 0) = pr^0(1 - pr) = 1 - pr$.

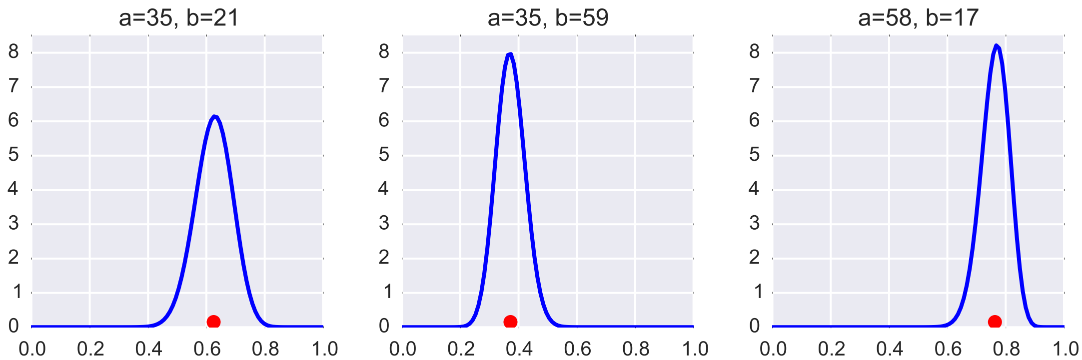

Fig. 2: Densităţi de probabilitate Beta(a,b), cu a, b, suficient de mari, ilustrând concentrarea în jurul medie (punctul roşu.

În machine learning un parametru necunoscut (în cazul nostru parametrul distribuţiei Bernoulli se notează cu θ şi se consideră că acesta este o observaţie asupra unei variabile aleatoare având densitatea probabilitate apriori $p(\theta)$, ce este o distribuţie Beta.

Algoritmul Bayesian învată observând succesiv reacţia userilor în fată reclamelor afişate care sunt probabilităţile de succes, ascunse $pr[i], i = \overline{0, k-1}$, ale celor k reclame.

Şi anume consideră că la începutul experimentului de afişare a reclamelor, în condiţiile în care nu se cunoaşte încă reacţia userilor la afişarea acestora, probabilităţile reale de succes a reclamelor pot fi orice numere din $[0, 1)$, prin urmare pot fi considerate ca observaţii asupra distribuţiei uniforme pe $[0, 1)$, adică asupra distribuţiei Beta (a=1, b=1).

Pe măsură ce se analizează noi date din afişari şi click-uri, această distribuţie apriori $p(\theta)$ a parametrului θ se actualizează succesiv conform formulei lui Bayes pentru densităţi de probabilitate) şi anume (formula de mai jos o putem demonstra doar peste 2 săptămâni!!!):

$$p(\theta|x) = \frac{p(\theta)p(x|theta)}{\int_{-\infty}^{\infty} p(\theta')p(x|\theta' d\theta'}$$

unde $p(\theta)$ este distribuţia iniţială de probabilitatea a parametrului θ, adică densitatea lui Beta(a,b), iar $p(x|\theta) = \theta^x(1-\theta)^{1-x}$ este distribuţia Bernoulli de parametru θ. $p(x|\theta)$ se interpretează ca o distribuţie condiţionată, adică $p(x|\theta)$=probabilitatea de a obţine rezultatul x=1 (succes=click) sau x=0 (eşec= nu obţii click) când probabilitatea de succes este θ

Se demonstrează că: dacă distribuţia apriori $p(\theta)$ este Beta(a,b), iar distribuţia $p(x|\theta)$, este Bernoulli, atunci distribuţia posterioară $p(\theta|x)$ este Beta(a+x, b+1-x), unde (x=1 sau 0) după cum s-a înregistrat succes sau eşec.

Prin urmare după fiecare afişare şi înregistrarea rezultatului x al afisării (click, adica x=1 sau fără click, adică x=0) se actualizează distribuţia parametrului θ. Această observaţie stă la baza Algoritmului Bayesian de simulare a procesului de selectare şi afişare a reclamelor.

Remarcăm că distribuţiile Beta ce sunt implicate în această problemă au parametrii a şi b numere întregi pozitive.

Datele pentru Algoritmul Bayesian

- `k` reclame, `N` useri, vectorul `pr` al probabilităţilor de succes ale celor k reclame, necunoscute algoritmului (broker-ului) ;
- vectorul `nrafis` de k elemente întregi, `nrafis`[i]=nr de afişari ale reclamei $i = 0, 1, 2, \dots, k-1$;
- vectorul `nrclicks` de k elemente întregi; `nrclicks[i]`= nr de click-uri pe reclama i, înregistrate în cursul afişarii ei pe pagina WEB; ATENŢIE aici NU este numarul mediu de click-uri ca la Algoritmul frecventist, ci nr efectiv de click-uri înregistrate.
- vectorul `theta` de k elemente reale; θ[`i`] este observaţie asupra distribuţiei Beta asociate reclamei $i = 0, 1, \dots, k-1$. O observaţie asupra acestei distribuţii este valoarea simulată de algoritm pentru probabilitatea de succes a reclamei i;

O funcţie în plus ce trebuie definită pentru a implementa Algoritmul Bayesian este funcţia ce simulează distribuţia Beta de parametri întregi `SimulBeta(a,b)` şi funcţia `SimulExp(param)` ce simulează distribuţia Exponenţială de parametru `param`.

Algoritmul de simulare a distribuţiei Beta(a, b), $a, b \in \mathbb{N}^*$ invocă şi generatorul distribuţiei exponenţiale, de parametru 1, Exp(1):

```
function SimulBeta(a,b)
    n=a+b-1;
    G=0;
    for i =0:n // i ia valorile 0, 1, 2, ..., n
        x[i]=SimulExp(1);
        G=G+x[i];
    end for
    B=0;
    for i =0:a-1// i ia valorile 0,1,..., a-1
        B=B+x[i]/G
    end for
    return B;
end function
```

Pseudocodul pentru Algoritmul Bayesian este următorul:

```
for user=1:N{
   for i=0:k-1
      theta[i]=SimulBeta(a=1+nrclicks[i], b=1+nrafis[i]-nrclicks[i]);
   // afiseaza reclama cu probabilitatea maxim‘a  de succes:
   j=indicele elementului maxim din vectorul theta;
```

```
incrementeza contorul pt nr de afisari a reclamei j;
x=Bernoulli(pr[j]);//x=1=click sau x=0=ne-clickuit
daca reclama j a fost clickuita, incrementeaza contorul pt numarul de click-uri primite
}\\end for
for i=0:k-1
   afiseaza pe ecarn si scrie intr-un fiser ultimii parametri a=, b= pt Beta;
   afiseaza  media fiecarei distributii Beta m[i]=a[i]/(a[i]+b[i]);
   afiseaza probabilitatea  pr[i];
   afiseaza |pr[i]-m[i]|;
```

Atenţie!!! iniţial $\texttt{nrclicks}[\texttt{i}] = 0$, $\texttt{nrafis}[\texttt{i}] = 0$.

Detaliere:

• Când începe simularea se generează k probabilităţi de succes $\texttt{theta}[0], \ldots, \texttt{theta}[\texttt{k}-1]$ din distribuţia uniforma pe [0,1], adică din Beta(a=1, b=1); Adică algoritmul (brokerul) nu poseda nicio informaţie despre potentialul de succes al reclamelor, deci probabilităţile lor de succes estimat pot fi orice numere subunitare).

• Afişează reclama j, căreia algoritmul i-a atribuit cea mai mare probabilitate de succes; şi incrementeaza $\texttt{nrafis}[\texttt{j}]$ cu 1

• Atitudinea userului 1 fată de această reclamă depinde de probabilitatea reală de succes, nu de cea simulată. Pentru a vedea atitudinea lui se generează o observaţie asupra variabilei Bernoulli având parametrul de succes, real, $\texttt{pr}[\texttt{j}]$. Cu alte cuvinte acţionează potenţialul real de atractivitate al reclamei;

• Dacă reclama este clickuită se incrementează contorul $\texttt{nrcliks}[\texttt{j}]$ cu 1;

• pentru userul al doilea, etc, se parcurge aceeasi procedură, doar că pentru fiecare user se decide ce reclamă să i se afişeze, generând k valori de observaţie asupra a k distribuţii Beta actualizate faţa de cele simulate pentru userului anterior şi anume asupra distribuţiilor

$$Beta(a = 1 + nrclicks[i], b = 1 + nrafis[i] - nrclicks[i]), \quad i = \overline{0, k-1}$$

şi reţinând indicele j al valorii maxime din cele k generate.

• etc.

La sfârşit salvaţi într-un fisier `DateBeta.txt`, pe câte o linie ultimii parametri $a = 1 + nrclicks[i], b = 1 + nrafis[i] - nrclicks[i]$, $i = \overline{0, k-1}$ (deci fisierul va conţine k linii de câte 2 elemente.

Dacă totul este OK, media fiecărei distribuţii Beta (a,b) de parametri a, b (salvati in fisier), adica: $\dfrac{a}{a+b} = \dfrac{1 + nrcliks[i]}{1 + nrclicks[i] + 1 + nrafis[i] - nrclicks[i]} \approx \dfrac{nrclicks[i]}{nrafis[i]}$ aproximează foarte bine probabilitatea experimentală $\dfrac{nrclicks[i]}{nrafis[i]}$ de a clickui reclama i. Aceasta ar trebui să fie aproximativ egală cu probabilitatea teoretică $\texttt{pr}[\texttt{i}]$, $i = \overline{0, k-1}$.

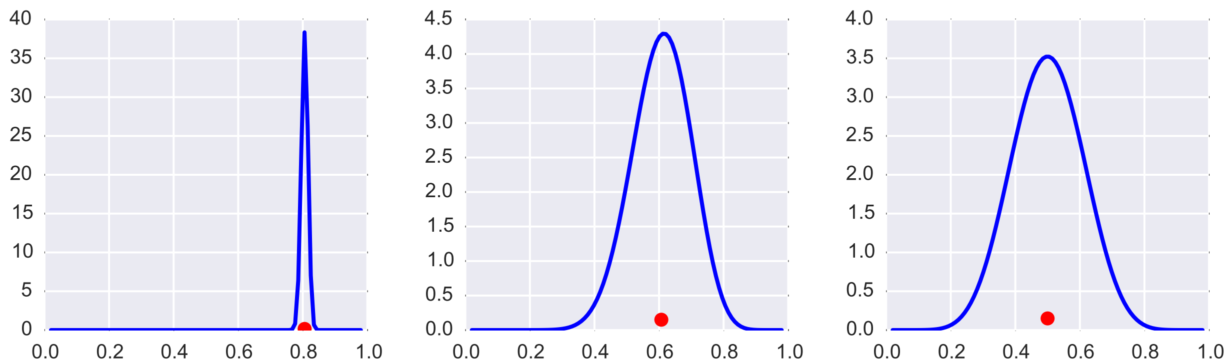

Fig. 3: Rezultatul simulării. Punctele roşii indică poziţia valorii medii.

Experimentând pentru 3 reclame având probabilităţile reale de succes pr ca fiind respectiv: 0.8, 0.63, 0.4 şi numărul de useri $N = 1500$ am obţinut la sfărşit următoarele distribuţii Beta pentru probabiliăţile de succes învăţate în cursul rulării:

Numărul de afişări ale fiecărei reclame din cele 3: $1452, 28, 20$ şi nr de clik-uri: $1170, 17, 10$. Astfel probabilităţile frecventiste de succes sunt $nrcliks[i]/nrafis[i]$, i=0,1,2: $0.80578512, 0.60714286, 0.5$, adică suficient de apropiate de probabilităţile ascunse pr.

Într-un IPython Notebook `Concluzii-Alg-Bayesian.ipynb` citiţi fişierul si vizualizaţi k densităţi de probabilitate Beta(a,b), ce ilustrează distribuţiile de probabilitate ale parametrilor de succes ale celor k reclame, învăţate pe parcursul rulării.

Dacă totul este OK, media fiecărei distribuţii Beta (a,b) de parametri a, b (salvati in fisier), adica: $\dfrac{a}{a+b} = \dfrac{1+nrcliks[i]}{1+nrclicks[i]+1+nrafis[i]-nrclicks[i]} \approx \dfrac{nrclicks[i]}{nrafis[i]}$ aproximează foarte bine probabilitatea experimentală $\dfrac{nrclicks[i]}{nrafis[i]}$ de a clickui reclama i. Aceasta ar trebui să fie aproximativ egală cu probabilitatea teoretică `pr[i]`, $i = \overline{0, k-1}$.

DETALII IMPORTANTE:

• Codul `C/C++` pentru această tema de proiect va conţine un `switch` cu două cazuri: cazul frecventist şi cazul Bayesian.

• Pentru ilustrare in Ipython Notebook luaţi pentru cazul Bayesian doar 3 reclame ca să poată fi rulat notebook-ul inclus. În rest puteţi experimenta cu câte vreţi.