Eötvös Loránd Tudományegyetem
Természettudományi Kar

Apagyi Dávid
Matematika BSc
alkalmazott matematikus szakirány

Gépi tanulási feladatok nagy adathalmazokon

Szakdolgozat

Témavezető: Lukács András
Számítógéptudományi Tanszék

Budapest, 2023.

Köszönetnyilvánítás

Hálás vagyok Lukács András témavezetéséért, különösen a mindvégig támogató hozzáállásáért, ahogyan aktívan végigkísérte és irányította a szakdolgozati munkámat, valamint gondosan ellenőrizte a dolgozatomat, és megjegyzéseivel, tanácsaival látott el engem.

Köszönettel tartozom egyetemi oktatóim, valamint középiskolai, illetve általános iskolai tanáraim munkájáért, akik nélkül most biztosan nem tartanék itt.

Hálás vagyok családtagjaimnak és barátaimnak, akik a tanulmányaimban, illetve azon túlmenően is támogattak, valamint támogatnak engem.

Tartalomjegyzék

1. fejezet

Bevezetés

A dolgozatban arra a problémára keresünk lehetséges megoldásokat, hogy hogyan tudunk elemezni, mérni akkora adathalmazokat, amelyeket nincs esélyünk többször végigolvasni. Tipikusan úgy tekinthetünk a kérdésre, hogy folyamatosan, online érkeznek az újabb és újabb adatok, megfigyelések – gondolhatunk például egy internetes gerinchálózat egyetlen pontján keresztülmenő adatcsomagokra –, amelyek egészét eltárolni nincs lehetőségünk, azonban képesnek kell lennünk megválaszolni bizonyos előre meghatározott kérdéseket.

Ilyen feltétel mellett a kérdések megválaszolására adott algoritmusaink esetén azt is megköveteljük, hogy a tárigény ne nőjön lineárisan sem az adatfolyam hosszával. Cserébe viszont jellemzően fel kell adnunk azt az elvárásunkat, hogy egzakt végeredményt kapjunk, helyette megelégszünk valamilyen értelemben közelítő megoldásokkal. A dolgozatban alapvetően kétféle értelemben vett közelítésekkel foglalkozunk.

A dolgozat első felében egy-egy számértékű függvényt szeretnénk megbecsülni. Itt valószínűségi algoritmusokat fogunk használni: ehhez előre rögzítünk két paramétert, amelyek azt határozzák meg, hogy legfeljebb milyen relatív hibával szeretnék dolgozni, illetve hogy legfeljebb mekkora valószínűséggel érjünk el mégis az előírtnál nagyobb mértékű relatív hibát.

A dolgozat második felében olyan kérdésekkel foglalkozunk, amikor lényegében egy-egy halmazt szeretnénk meghatározni: gyakori elemeket és gyakori termékhalmazokat fogunk keresni. Ekkor már determinisztikus algoritmusokat fogunk használni. Itt a közelítés abban fog megnyilvánulni, hogy egy bővebb halmazt adunk válaszként, mint a valódi megoldás. Két nagy előnye lesz még a bemutatott módszereknek: az egyik az, hogy a válaszunk tartalmazni fog minden valódi megoldást, garantáltan nem hagyunk ki egyet sem, csak esélyes, hogy rosszakat is visszaadunk. A másik előny az, hogy a megoldásként visszaadott

rossz elemek, halmazok sem túlzottan rosszak, alulról tudjuk becsülni gyakoriságukat, ami nem lesz messze az elvárttól.

A második és a harmadik fejezetben hasonló eszközökkel közelítjük meg a különböző elemek leszámlálásának, és a frekvenciamomentumok becslésének problémáját. Előbbi esetén arra keresünk becslést, hogy hány különböző elemet láttunk eddig az adatfolyamban, míg a frekvenciamomentumok kapcsán az adatfolyamban szereplő elemek gyakoriságainak k-adik hatványainak összegét szeretnénk közelíteni.

Látni fogjuk, hogy a két probléma szorosan kapcsolódik egymáshoz, a nulladik frekvenciamomentum ($k = 0$) éppen az előbbi kérdésnek felel meg. Ezen túl a tárgyalt módszerek is párhuzamba állíthatók egymással.

A negyedik fejezetben a gyakori elemek keresésével foglalkozunk, majd az ötödik fejezetben a gyakori termékhalmazok keresése lesz a fókuszban: ehhez kapcsolódóan bevezetünk néhány gyakorlati indíttatású fogalmat, majd az előző fejezet alkalmazásaként bemutatunk egy adatfolyamokban is alkalmazható lehetséges megközelítést.

2. fejezet

Különböző elemek leszámlálása

2.1. A Flajolet–Martin-algoritmus

A következőkben arra a problémára keresünk megoldást, hogy hogyan tudjuk megbecsülni azt, hogy egy adatfolyamban érkező elemek között hány különbözőt láttunk eddig. A naiv algoritmustól, miszerint valamennyi előforduló különböző elemet eltároljuk valamilyen módon, merőben eltérő eljárást mutatunk be, amely segítségével előre rögzített paraméterek mellett az adatfolyam hosszától független, konstans méretű memóriában tehetünk valószínűségi becslést az eddig látott különböző elemek számára.

A bemutatott (idealizált) algoritmusok [6] alapötlete, hogy tételezzük fel egy $(0, 1)$ intervallumba valamilyen értelemben egyenletesen képező h hash függvény létezését, és alkalmazzuk rá ezt a függvényt minden érkező elemre. Azt várjuk, hogy ha d különböző elem volt az adatfolyamban, akkor az előbbi hash értékek egyenletesen oszlanak el a $(0, 1)$ intervallumon, és ezért a legkisebb értékre $h_{\min} \approx \frac{1}{d+1}$, így a $d \approx \frac{1}{h_{\min}} - 1$ értékkel becsülhetjük d-t, azaz a különböző elemek számát. Vegyük észre, hogy ehhez a becsléshez elegendő egyetlen értéket eltárolnunk ($h_{\min}$), és valamennyiszer egy új elemet olvasunk az adatfolyamból, akkor elég azt összehasonlítanunk az eddigi legkisebb hash értékkel, és szükség esetén frissíteni azt.

A továbbiakban ezt az intuíciót fogjuk formalizálni, és eljárást adni arra, hogyan tudunk bizonyos valószínűséggel egy előírt konfidenciaintervallumon belül maradni a becslésünkkel. A fenti tulajdonságú hash függvénnyel, valamint a $h_{\min}$ változó kellően pontos ábrázolásával járó gyakorlati nehézségeket nem tárgyaljuk. Megjegyezzük azonban, hogy ezek kiküszöbölhetők: az eredeti Flajolet–Martin-algoritmus [2] lebegőpontos értékek

helyett L-bites hash függvényeket használ, amelyek a $[0, 2^L - 1]$ intervallumba képeznek, ahol L rögzített egész szám.

Legyenek $X_1, X_2, \ldots, X_d$ független, a $(0, 1)$ intervallumon egyenletes eloszlásból származó minta – ezek felelnek meg a d darab hash értéknek –, legyen X^* a legkisebb mintaelem. Határozzuk meg X^* várható értékét és szórását.

2.1. Állítás. $\mathbb{E}(X^*) = \dfrac{1}{d+1}$.

Bizonyítás. Határozzuk meg X^* eloszlásfüggvényét:

$$\begin{aligned} F_{X^*}(t) &= \mathbb{P}(X^* < t) = \\ &= 1 - \mathbb{P}(X^* \geq t) = \\ &= 1 - \mathbb{P}(X_1, \ldots, X_d \geq t) = \\ &= \begin{cases} 0, & \text{ha } t < 0, \\ 1 - (1-t)^d & \text{ha } 0 \leq t \leq 1, \\ 1, & \text{ha } 1 < t. \end{cases} \end{aligned}$$

Ez alapján X^* sűrűségfüggvénye $f_{X^*}(t) = d(1-t)^{d-1}\mathbb{I}(0 \leq t \leq 1)$, aminek segítségével meghatározhatjuk a várható értéket:

$$\mathbb{E}(X^*) = \int_{\mathbb{R}} t \cdot f_{X^*}(t)\,\mathrm{d}t = \int_0^1 t \cdot d(1-t)^{d-1}\,\mathrm{d}t$$

Az $u = 1 - t$ helyettesítést alkalmazva:

$$\begin{aligned} &\int_1^0 (1-u) \cdot d \cdot u^{d-1}(-1)\,\mathrm{d}u = \int_0^1 d \cdot u^{d-1} - d \cdot u^d\,\mathrm{d}u = \\ &= \left[u^d - \frac{d}{d+1}u^{d+1}\right]_0^1 = \left(1 - \frac{d}{d+1}\right) - (0-0) = \frac{1}{d+1}. \end{aligned}$$

□

Tehát X^*-nek valóban annyi a várható értéke, mint amennyit a bevezető részben intuitív módon sejtettünk, ugyanakkor abból, hogy $\mathbb{E}(X^*) = \dfrac{1}{d+1}$, nem következik, hogy a $d^* = \dfrac{1}{X^*} - 1$ becslésünkre $\mathbb{E}(d^*) = d$ is teljesülne. Ennek a problémának a kiküszöbölésére adunk egy megoldást a következő szakaszban.

2.2. Átlagot használó módosítás

Annak ellenére, hogy nem tudjuk a várható értékét a becslésünknek, ha az előző algoritmust s-szer párhuzamosan futtatjuk, akkor az így kapott értékek átlagának relatív hibáját megbecsülhetjük. A továbbiakban nevezzük az eredeti algoritmust `FM`-nak, a most vázoltat pedig `FM+`-nak. Szükségünk van a Csebisev-egyenlőtlenség valószínűségi változók átlagára vonatkozó következményére:

2.2. Lemma. *Legyenek $X_1, \ldots, X_s$ független, azonos eloszlású valószínűségi változók, $\varepsilon > 0$, és legyen $\overline{X} = \dfrac{X_1 + \ldots + X_s}{s}$. Ekkor:*

$$\mathbb{P}(|\overline{X} - \mathbb{E}(X_1)| \geq \varepsilon) \leq \frac{D^2(X_1)}{s \cdot \varepsilon^2}.$$

Bizonyítás. A bizonyítás során felhasználjuk, hogy független, azonos eloszlású valószínűségi változók várható értéke és szórásnégyzete is összeadódik. Alakítsuk át a bal oldali kifejezést.

$$\mathbb{P}(|\overline{X} - \mathbb{E}(X_1)| \geq \varepsilon) = \mathbb{P}(|s \cdot \overline{X} - s \cdot \mathbb{E}(X_1)| \geq s \cdot \varepsilon) =$$
$$= \mathbb{P}\left(\left|\sum_{i=1}^{s} X_i - \mathbb{E}\left(\sum_{i=1}^{s} X_i\right)\right| \geq s \cdot \varepsilon\right).$$

Ezen a ponton alkalmazzuk az összegre a $\mathbb{P}(|X - \mathbb{E}(X)| \geq \varepsilon) \leq \dfrac{D^2(X)}{\varepsilon^2}$ klasszikus Csebisev-egyenlőtlenséget. Ez alapján:

$$\mathbb{P}(|\overline{X} - \mathbb{E}(X_1)| \geq \varepsilon) \leq \frac{D^2\left(\sum_{i=1}^{s} X_i\right)}{(s \cdot \varepsilon)^2} = \frac{s \cdot D^2(X_1)}{(s \cdot \varepsilon)^2} = \frac{D^2(X_1)}{s \cdot \varepsilon^2}. \qquad \square$$

Az előző lemma használatához először meg kell határoznunk a szórásnégyzetet.

2.3. Állítás. $\mathbb{E}\left((X^*)^2\right) = \dfrac{2}{d+1} - \dfrac{2}{d+2}$.

Bizonyítás. A 2.1 állítás bizonyításához hasonló módon:

$$\begin{aligned}\mathbb{E}\left((X^*)^2\right) &= \int_{\mathbb{R}} t^2 \cdot f_{X^*}(t)\,\mathrm{d}t = \int_0^1 t^2 \cdot d(1-t)^{d-1}\,\mathrm{d}t = \\ &= \int_1^0 (1-u)^2 \cdot d \cdot u^{d-1}(-1)\,\mathrm{d}u = \int_0^1 d \cdot u^{d-1} - 2d \cdot u^d + d \cdot u^{d+1}\,\mathrm{d}u = \\ &= \left[u^d - \frac{2d}{d+1}u^{d+1} + \frac{d}{d+2}u^{d+2}\right]_0^1 = \left(1 - \frac{2d}{d+1} + \frac{d}{d+2}\right) - (0-0+0) = \\ &= 1 - \left(2 - \frac{2}{d+1}\right) + \left(1 - \frac{2}{d+2}\right) = \frac{2}{d+1} - \frac{2}{d+2}.\end{aligned}$$

□

A 2.1 és 2.3 állítások alapján egyszerűen meghatározhatjuk X^* szórásnégyzetét. A későbbiek szempontjából rögtön felül is becsüljük.

2.4. Állítás. $D^2(X^*) \leq \dfrac{1}{(d+1)^2}$.

Bizonyítás. A szórásnégyzet definíciója alapján:

$$\begin{aligned}D^2(X^*) &= \mathbb{E}\left((X^*)^2\right) - \mathbb{E}(X^*)^2 = \frac{2}{d+1} - \frac{2}{d+2} - \frac{1}{(d+1)^2} = \\ &= \frac{2}{(d+1)(d+2)} - \frac{1}{(d+1)^2} \leq \frac{2}{(d+1)^2} - \frac{1}{(d+1)^2} = \frac{1}{(d+1)^2},\end{aligned} \tag{2.1}$$

és éppen ezt kellett bizonyítani. □

Futtassuk az `FM` algoritmusunkat s darab szálon párhuzamosan, és tekintsünk minden szálról a legkisebb hash értéket: jelölje ezek átlagát $\overline{X}$. Ekkor az előzőek alapján:

$$\begin{aligned}&\mathbb{P}\left(\left|\overline{X} - \mathbb{E}\left(X^*\right)\right| \geq \varepsilon \cdot \mathbb{E}\left(X^*\right)\right) = \\ &= \mathbb{P}\left(\left|\overline{X} - \frac{1}{d+1}\right| \geq \frac{\varepsilon}{d+1}\right) \overset{(2.2)}{\leq} \frac{D^2(X^*)}{s \cdot \dfrac{\varepsilon^2}{(d+1)^2}} \overset{(2.1)}{\leq} \frac{1}{s \cdot \varepsilon^2}.\end{aligned} \tag{2.2}$$

Utóbbi egyenlőtlenségünk segítségével már meg tudjuk becsülni a hibánkat, ha a $d \approx \overline{d} = \dfrac{1}{\overline{X}} - 1$ közelítést alkalmazzuk.

2.5. Állítás. *Az `FM+` algoritmus ε-relatív közelítő legfeljebb $\delta = \dfrac{16}{s \cdot \varepsilon^2}$ valószínűségű hibával, azaz*

$$\mathbb{P}(|\overline{d} - d| > \varepsilon \cdot d) < \delta,$$

vagy másképpen

$$\mathbb{P}((1-\varepsilon)d \leq \overline{d} \leq (1+\varepsilon)d) \geq 1 - \frac{16}{s \cdot \varepsilon^2}.$$

Bizonyítás. Vizsgáljuk meg, hogy mit jelent a 2.2 egyenlőtlenségben szereplő esemény a $\overline{d} = \dfrac{1}{\overline{X}} - 1$ közelítésünkre vonatkozóan:

$$\left|\overline{X} - \frac{1}{d+1}\right| \geq \frac{\varepsilon}{d+1} \iff \frac{1-\varepsilon}{d+1} \leq \overline{X} \leq \frac{1+\varepsilon}{d+1} \iff \frac{d+1}{1+\varepsilon} - 1 \leq \underbrace{\frac{1}{\overline{X}} - 1}_{\overline{d}} \leq \frac{d+1}{1-\varepsilon} - 1.$$

A továbbiakban az utóbbi egyenlőtlenség bal és jobb oldalát fogjuk rendre alulról és felülről becsülni: az a célunk, hogy $(1 \pm c \cdot \varepsilon)d$ alakú korlátok közé szorítsuk $\overline{d}$-t úgy, hogy az részhalmazként tartalmazza az előző intervallumot.

Tekintsük először a bal oldalt. Mivel $\dfrac{1}{1+\varepsilon} \geq 1 - \varepsilon$, hiszen $(1+\varepsilon)(1-\varepsilon) = 1 - \varepsilon^2 \leq 1$, ezért:

$$\frac{d+1}{1+\varepsilon} - 1 \geq (1-\varepsilon)(d+1) - 1 = d - \varepsilon \cdot d - \varepsilon \geq d - \varepsilon \cdot d - \varepsilon \cdot d = (1-2\varepsilon)d.$$

Hasonlóan tekintsük a jobb oldalt, és tegyük fel, hogy $0 < \varepsilon < \dfrac{1}{2}$: ekkor $\dfrac{1}{1-\varepsilon} \leq 1 + 2\varepsilon$, mivel $(1-\varepsilon)(1+2\varepsilon) = 1 + \varepsilon(1-2\varepsilon) \geq 1$. Tehát:

$$\frac{d+1}{1-\varepsilon} - 1 \leq (1+2\varepsilon)(d+1) - 1 = d + 2\varepsilon \cdot d + 2\varepsilon \leq d + 2\varepsilon \cdot d + 2\varepsilon \cdot d = (1+4\varepsilon)d.$$

A bizonyítást azzal kezdtük, hogy a korábbi, $\overline{X}$-re vonatkozó

$$\Big((1-\varepsilon)\mathbb{E}(X^*), (1+\varepsilon)\mathbb{E}(X^*)\Big)$$

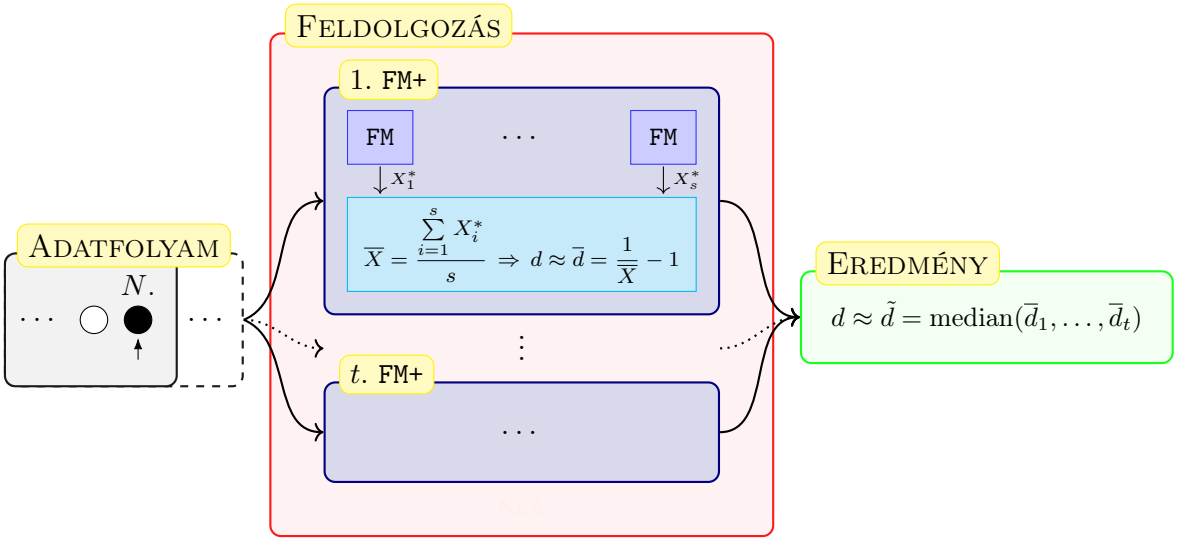
konfidenciaintervallumunk segítségével $\overline{d}$-re vonatkozóan fogalmaztunk meg egy ekvivalens intervallumbecslést, amiről tudtuk, hogy legalább $1 - \delta = 1 - \dfrac{1}{s \cdot \varepsilon^2}$ valószínűséggel teljesül. Ezután egy ennél bővebb halmazt tekintettünk, ezért $\overline{d}$ a $((1-2\varepsilon)d, (1+4\varepsilon)d)$ halmazba is legalább ekkora valószínűséggel esik. Hasonlóan a még bővebb $((1-4\varepsilon)d, (1+4\varepsilon)d)$ halmazt tekintve, valamint átparaméterezve adódik, hogy:

$$\mathbb{P}((1-\varepsilon)d \leq \overline{d} \leq (1+\varepsilon)d) \geq 1 - \frac{1}{s \cdot \left(\frac{\varepsilon}{4}\right)^2} = 1 - \frac{16}{s \cdot \varepsilon^2}. \qquad \square$$

Megjegyzés. Az algoritmus futtatásához már csak azt kell megmondanunk, hogy a paraméterek függvényében, hogyan válasszuk meg az s értéket, de a fentiek alapján ez nyilvánvaló: adott ε és δ esetén az eredeti algoritmust $s = \dfrac{16}{\delta \cdot \varepsilon^2}$-szer futtatva megfelelő közelítést kapunk.

2.3. Újabb módosítás medián használatával

Úgy szeretnénk változtatni az algoritmusunkon, hogy a ε relatív közelítés mellett a hiba δ valószínűsége is egy előre rögzített paraméter legyen. Ehhez futtasuk az előző, egyenként s darab szálon futó algoritmust, t-szer párhuzamosan (egymástól függetlenül), és legyen a végső becslésünk a t darab becslés mediánja. A teljes, összesen $t \times s$ részegységből álló algoritmust (`FM++`) tehát a 2.1 ábrán látható módon foglalhatjuk össze.



2.1. ábra. Az `FM++` algoritmus sematikus felépítése

A továbbiakban arra keresünk választ, hogy milyen közelítést eredményez ez az algoritmus, és milyen s és t választással tudunk a paraméterként megadott ε és δ által meghatározott módon közelíteni.

2.6. Tétel. *Megfelelő s és t választásával az `FM++` algoritmus által meghatározott $\tilde{d}$ közelítésünkre teljesül, hogy:*

$$\mathbb{P}(|\tilde{d} - d| > \varepsilon \cdot d) \leq \delta.$$

A tétel igazolásához szükségünk van egy lemmára [5], amelyet nem bizonyítunk:

2.7. Lemma (Hoeffding-egyenlőtlenség)**.** *Legyen $X \sim \text{Bin}(n, p)$ binomiális eloszlású valószínűségi változó. Ekkor*

$$\mathbb{P}\left(X \geq \mathbb{E}(X) + t\right) \leq \exp\left(\frac{-t^2}{2n}\right)$$

teljesül bármely $t \geq 0$ esetén.

Bizonyítás (2.6 tétel). Ahhoz, hogy a $\tilde{d}$ medián az $[(1-\varepsilon)d, (1+\varepsilon)d]$ intervallumon kívülre essen, a t darab `FM+` algoritmus $\overline{d}_1, \ldots, \overline{d}_t$ közelítései közül több, mint $\frac{t}{2}$-nek $(1-\varepsilon)d$-nál kisebbnek; vagy több, mint $\frac{t}{2}$-nek $(1+\varepsilon)d$-nál nagyobbnak kell lennie: tehát összességében több, mint $\frac{t}{2}$ közelítés hibás. Jelöle Y ezeket, azaz a $\overline{d}_1, \ldots, \overline{d}_t$ közelítések közül a hibásak számát. Ekkor

$$\mathbb{P}(|\tilde{d} - d| > \varepsilon \cdot d) \leq \mathbb{P}\left(Y > \frac{t}{2}\right). \tag{2.3}$$

Tegyük fel, hogy adottak az ε és δ^+ paraméterű `FM+` algoritmusaink. (A δ^+ paraméter megválasztására később térünk ki.) Ekkor Y várható értékét a következő módon becsülhetjük:

$$\mathbb{E}(Y) = \mathbb{E}\left(\sum_{i=1}^{t} \mathbb{I}(i\text{-edik közelítés hibás})\right) = \sum_{i=1}^{t} \mathbb{P}(i\text{-edik közelítés hibás}) \leq \delta^+ \cdot t.$$

Az a célunk, hogy a Hoeffding-egyenlőtlenség segítségével megbecsüljük a `FM++` hibájának (2.3) valószínűségét (a könnyebbség kedvéért a szigorú egyenlőtlenség helyett megengedjük az egyenlőséget is):

$$\begin{aligned}\mathbb{P}\left(Y \geq \frac{t}{2}\right) &= \mathbb{P}\left(Y \geq \delta^+ \cdot t + \left(\frac{1}{2} - \delta^+\right) t\right) \leq \\ &\leq \mathbb{P}\left(Y \geq \mathbb{E}(Y) + \left(\frac{1}{2} - \delta^+\right) t\right).\end{aligned}$$

Ugyan nem ismerjük pontosan Y eloszlását (hiszen csak annyit tudunk, hogy az egyes komponensek egyenként *legfeljebb* δ^+ valószínűséggel hibáznak), a vizsgált esemény valószínűségét felülről becsülhetjük úgy, ha Y-ra egy $Y \sim \mathrm{Bin}(n, \delta^+)$ binomiális eloszlású valószínűségi változóként tekintünk. A jelölésben nem teszünk különbséget. Tegyük fel, hogy $\frac{1}{2} - \delta^+ > 0$, azaz $\delta^+ < \frac{1}{2}$. Ekkor a lemma alapján:

$$\begin{aligned}\mathbb{P}\left(Y \geq \frac{t}{2}\right) &\leq \exp\left[\frac{-\left(\frac{1}{2} - \delta^+\right)^2 t^2}{2t}\right] = \\ &= \exp\left[-\frac{1}{2}\left(\frac{1}{2} - \delta^+\right)^2 t\right] \leq \\ &\leq \delta,\end{aligned}$$

ahol utóbbi egyenlőtlenség pontosan akkor teljesül, ha

$$t \geq \frac{2}{\left(\frac{1}{2} - \delta^+\right)^2} \log\left(\frac{1}{\delta}\right).$$

Tehát adott ε és δ paraméterek esetén például a $\delta^+ = \frac{1}{3} < \frac{1}{2}$ választással $s = \frac{16}{\delta^+ \cdot \varepsilon^2} = \frac{48}{\varepsilon^2}$ és $t = \frac{2}{\left(\frac{1}{2} - \delta^+\right)^2} \log\left(\frac{1}{\delta}\right) = 72 \log\left(\frac{1}{\delta}\right)$ esetén az `FM++` algoritmus megfelelő közelítést ad. □

3. fejezet

Frekvenciamomentumok becslése

3.1. Problémafelvetés

Ebben a fejezetben az előzőleg tárgyalt kérdést, a különböző elemek leszámlását szeretnénk általánosítani. Tekintsünk az adatfolyamra úgy, mint egy $S = (x_1, \ldots, x_m)$ rendezett m-esre, amelynek minden eleme egy rögzített $A = \{a_1, \ldots, a_n\}$ n-elemű halmazból kerül ki. Praktikusan lehet $A = \{1, \ldots, n\}$, de az elemek konkrét értéke nem játszik szerepet a becslésben. Jelölje m_i minden $i = 1, \ldots, n$-re az a_i elem előfordulásainak számát az S adatfolyamban, azaz

$$m_i = |\{j \colon x_j = a_i\}|.$$

Ekkor minden nemnegatív egész k értékre definiáljuk a következő mennyiséget.

3.1. Definíció (k-adik frekvenciamomentum)**.**

$$F_k = \sum_{i=1}^{n} m_i^k.$$

A $k = 0$ esetben azokat az i indexet, amelyekre $a_i \notin S$, nem értjük bele a szummába, így elkerülve a „0^0” miatt felmerülő problémákat. Szokás még a $k = \infty$ esetben is definiálni, ekkor a legnagyobb gyakoriságot értjük alatta:

$$F_\infty = \max_{1 \le i \le n} m_i.$$

Vegyük észre, hogy a fenti jelöléssel F_0 éppen a különböző elemek számát jelöli, amelynek becslésére az előző fejezetben adtunk közelítő algoritmust, a Flajolet–Martin-algoritmust.

Az első frekvenciamomentum (F_1) nem más, mint az adatfolyam hossza. Ennek ugyanolyan értelemben vett közelítésére, mint az előzőekben, ad választ Morris algoritmusa [6]. A továbbiakban az adatfolyam m hosszát ismertnek tekintjük, de sampling-módszerekkel módosítható a bemutatott algoritmus úgy, hogy ez ne jelentsen problémát: ezt a fejezet végén részletesebben tárgyaljuk.

3.2. Alon–Matias–Szegedy-algoritmus

A következőkben Noga Alon, Yossi Matias és Mario Szegedy algoritmusát [1] mutatjuk be, amely általános k esetén ad közelítést F_k-ra a következő tárigény mellett:

$$O\left(\frac{1}{\varepsilon^2}\cdot\log\left(\frac{1}{\delta}\right)\cdot k\cdot n^{1-\frac{1}{k}}\cdot(\log n+\log m)\right),$$

ahol ε és δ ugyanazzal a jelentéssel bír, mint az előző fejezetben, azaz az $\tilde{F}_k$ közelítésünkre teljesül, hogy:

$$\mathbb{P}(|\tilde{F}_k - F_k| > \varepsilon\cdot F_k) \leq \delta.$$

Az algoritmus hasonló elemekből épül fel, mint a Flajolet–Martin-algoritmus. Az a célunk, hogy konstruáljunk egy X valószínűségi változót, amelynek várható értékére $\mathbb{E}(X) = F_k$, valamint a szórását könnyen felülről tudjuk becsülni. Ezután s-szer párhuzamosan futtatva átlagoljuk az egyes futtatások közelítéseit, és az előző komponenst t-szer párhuzamosan futtatva a közelítések mediánját tekintjük a végső becslésünknek.

Az alapvető becslést a következőképpen határozzuk meg: legyen p egy véletlenszerűen (egyenletes eloszlás szerint) kiválasztott index S adatfolyam indexei ($\{1,\ldots,m\}$) közül – itt használjuk ki először, hogy az adatfolyam m hosszát ismertnek tekintjük. Vezessük be a következő függvényt, amely megadja, hogy egy adott i index esetén hány elem van ettől az indextől kezdve S-ben, amely megegyezik x_p-vel:

$$r(i) = |\{j\colon j \geq p,\, x_j = x_p\}|.$$

Világos, hogy $r(i) \geq 1$, hiszen i mindig eleme annak a halmaznak, amelynek a számosságát tekintjük $r(i)$ definíciójában. Legyen $r = r(p)$, és legyen a becsléshez használt valószínűségi változónk a következő:

$$X = m\left(r^k - (r-1)^k\right),$$

amely meghatározásának tárigénye az eddigiek alapján $O(\log n + \log m)$ bit.

Határozzuk meg az X valószínűségi változó várható értékét, majd adjunk felső becslést a második momentumra, amit a szórásnégyzet felső becsléseként fogunk hasznosítani.

3.2. Állítás. $\mathbb{E}(X) = F_k$.

Bizonyítás. Definiáljuk a következő eseményeket minden $i = 1, \ldots, n$-re és $j = 1, \ldots, m_i$-re:

$$A_{i,j} = \{x_p = x_i \quad \text{és} \quad r(i) = j\},$$

ami éppen annak felel meg, hogy egy i és j által meghatározott elemet választunk S-ből, aminek a valószínűsége $\mathbb{P}(A_{i,j}) = \frac{1}{m}$, mivel p-t egyenletesen eloszlás szerint választottuk. Világos, hogy ha $A_{i,j}$ bekövetkezik, akkor $r = j$, ezért a kérdéses X valószínűségi változót a következőképpen írhatjuk:

$$X = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \mathbb{I}(A_{i,j}) \cdot m \left(j^k - (j-1)^k\right).$$

Ez alapján már könnyen meghatározhatjuk X várható értékét. Először használjuk fel, hogy a várható érték lineáris, majd pedig azt, hogy egy esemény indikátorának várható értéke egyenlő az esemény valószínűségével.

$$\begin{aligned}
\mathbb{E}(X) &= \mathbb{E}\left(\sum_{i=1}^{n} \sum_{j=1}^{m_i} \mathbb{I}(A_{i,j}) \cdot m \left(j^k - (j-1)^k\right)\right) = \\
&= \sum_{i=1}^{n} \sum_{j=1}^{m_i} \mathbb{E}\left(\mathbb{I}(A_{i,j})\right) \cdot m \left(j^k - (j-1)^k\right) = \\
&= \sum_{i=1}^{n} \sum_{j=1}^{m_i} \mathbb{P}\left(A_{i,j}\right) \cdot m \left(j^k - (j-1)^k\right) = \\
&= \sum_{i=1}^{n} \sum_{j=1}^{m_i} \frac{1}{m} \cdot m \left(j^k - (j-1)^k\right) = \qquad (3.1)
\end{aligned}$$

a belső szummát teleszkopikusan össszegezve pedig adódik, hogy:

$$\begin{aligned}
&= \sum_{i=1}^{n} \Big[\left(1^k - (1-1)^k\right) + \ldots \left(m_i^k - (m_i - 1)^k\right)\Big] = \\
&= \sum_{i=1}^{n} m_i^k = F_k,
\end{aligned}$$

és éppen ezt akartuk igazolni. □

Az előző fejezetben a kritikus mértékű relatív hiba valószínűségének becslésekor azt használtuk ki – lásd (2.2) –, hogy a szórásnégyzetet a várható értékkel *összemérhető* módon becsültük meg (pontosabban a négyzetének valami konstansszorosával), így egy attól független felső becslést kaptunk a Csebisev-egyenlőtlenség alkalmazásával a hiba valószínűségére. Most is ezt fogjuk csinálni: a célunk egy $D^2(X) \leq c \cdot F_k^2$ alakú becslés keresése.

3.3. Állítás. $\mathbb{E}(X^2) \leq k \cdot m \cdot F_{2k-1} = k \cdot F_1 \cdot F_{2k-1}$.

Bizonyítás. Az előző bizonyításhoz hasonló módon adódik, hogy:

$$X^2 = \sum_{i=1}^{n} \sum_{j=1}^{m_i} \mathbb{I}(A_{i,j}) \cdot \left[m\left(j^k - (j-1)^k\right)\right]^2.$$

Az $\mathbb{E}(X^2)$ második momentum meghatározásához végezzük el először ugyanazokat a lépéseket, mint a (3.1) egyenletig tettük a korábbiakban:

$$\begin{aligned}\mathbb{E}(X^2) &= \sum_{i=1}^{n} \sum_{j=1}^{m_i} \frac{1}{m} \cdot \left[m\left(j^k - (j-1)^k\right)\right]^2 = \\ &= m \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left[\left(j^k - (j-1)^k\right)\right]^2.\end{aligned}$$

Becsüljük felül a szummában lévő tagokat még a négyzetre emelésük előtt:

$$\begin{aligned}j^k - (j-1)^k = (j-(j-1)) \cdot \Big(j^{k-1} + j^{k-2}(j-1) + \dots \\ \dots + j(j-1)^{k-2} + (j-1)^{k-1}\Big) \leq k \cdot j^{k-1}.\end{aligned}$$

Bontsuk fel két tényezőre a négyzeteket, és becsüljük csak az egyiket felül az előbbi egyen-

lőtlenséggel, majd pedig ismét összegezzünk teleszkopikusan:

$$\begin{aligned}
\mathbb{E}(X^2) &= m \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left[\left(j^k - (j-1)^k \right) \right]^2 \\
&= m \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left(j^k - (j-1)^k \right) \cdot \left(j^k - (j-1)^k \right) \leq \\
&\leq m \sum_{i=1}^{n} \sum_{j=1}^{m_i} k \cdot j^{k-1} \cdot \left(j^k - (j-1)^k \right) = \\
&= k \cdot m \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left(j^{2k-1} - j^{k-1} \cdot (j-1)^k \right) \leq \\
&\leq k \cdot m \sum_{i=1}^{n} \sum_{j=1}^{m_i} \left(j^{2k-1} - (j-1)^{2k-1} \right) = \\
&= k \cdot m \sum_{i=1}^{n} m_i^{2k-1} = k \cdot m \cdot F_{2k-1} = k \cdot F_1 \cdot F_{2k-1}.
\end{aligned}$$

□

Végezzünk további becsléseket az előzőleg kapott eredményre vonatkozóan.

3.4. Állítás. $F_1 \cdot F_{2k-1} \leq n^{1-\frac{1}{k}} \cdot F_k^2$.

Bizonyítás. Jelöljük M-mel az m_i értékek közül a legnagyobbat:

$$M = \max_{1 \leq i \leq n} m_i.$$

Ekkor triviálisan teljesül, hogy $M^\alpha \leq F_\alpha$, mivel a jobb oldal csupa nemnegatív tagból áll, és tartalmazza M^α-t. Ezt felhasználva becsüljük a bizonyítandó egyenlőtlenség bal oldalát:

$$\begin{aligned}
F_1 \cdot F_{2k-1} &= F_1 \cdot \left(\sum_{i=1}^{n} m_i^{2k-1} \right) = \\
&= F_1 \cdot \left(\sum_{i=1}^{n} \underbrace{m_i^{k-1}}_{\leq M^{k-1}} \cdot m_i^k \right) = \\
&= F_1 \cdot M^{k-1} \underbrace{\sum_{i=1}^{n} m_i^k}_{F_k} \leq
\end{aligned}$$

használjuk fel, hogy $M^k \leq F_k$, és hogy $f(x) = x^{\frac{k-1}{k}}$ monoton növő a pozitív félegyenesen – ezen a ponton feltesszük még, hogy $k \geq 1$ –, ezért $M^{k-1} \leq F_k^{1-\frac{1}{k}}$:

$$\leq F_1 \cdot F_k^{1-\frac{1}{k}} \cdot F_k = F_1 \cdot F_k^{2-\frac{1}{k}}.$$

Becsüljük felül F_1-et a hatványközepek közötti egyenlőtlenség segítségével ($1 \leq k$):

$$\frac{F_1}{n} = \frac{\sum_{i=1}^{n} m_i}{n} \leq \left(\frac{\sum_{i=1}^{n} m_i^k}{n} \right)^{\frac{1}{k}} = \left(\frac{F_k}{n} \right)^{\frac{1}{k}},$$

ahonnan

$$F_1 \leq n^{1-\frac{1}{k}} \cdot F_k^{\frac{1}{k}}.$$

Ezt visszahelyettesítve az előző számításba, adódik, hogy:

$$F_1 \cdot F_{2k-1} \leq n^{1-\frac{1}{k}} \cdot F_k^{\frac{1}{k}} \cdot F_k^{2-\frac{1}{k}} = n^{1-\frac{1}{k}} \cdot F_k^2,$$

és éppen ezt kellett bizonyítani. □

Az előző, technikaibb jellegű számításokat követően folytassuk az algoritmus tárgyalását a szakasz elején eltervezett módon: futassuk az eddigi algoritmust s-szer függetlenül, párhuzamosan, és jelöljük $\overline{X}$-szel a meghatározott X becslések átlagát. Ennek várható értéke természetesen – a várható érték linearitása alapján – ugyanannyi, mint egyetlen becslésnek, azaz $\mathbb{E}(\overline{X}) = F_k$. Vizsgáljuk most egyetlen becslés szórását:

$$D^2(X) \leq \mathbb{E}(X^2) \overset{(3.3)}{\leq} k \cdot F_1 \cdot F_{2k-1} \overset{(3.4)}{\leq} k \cdot n^{1-\frac{1}{k}} \cdot F_k^2. \tag{3.2}$$

Alkalmazzuk az átlag relatív hibájára vonatkozó a Csebisev-egyenlőtlenséget (a 2.2 állítást a (2.2) egyenlőtlenséghez hasonló módon):

$$\begin{aligned}
\mathbb{P}\left(\text{kritikus relatív hiba}\right) &= \mathbb{P}\left(\left|\overline{X} - \mathbb{E}\left(X\right)\right| \geq \varepsilon \cdot \mathbb{E}\left(X\right)\right) = \\
&= \mathbb{P}\left(\left|\overline{X} - F_k\right| \geq \varepsilon \cdot F_k\right) \overset{(2.2)}{\leq} \\
&\overset{(2.2)}{\leq} \frac{D^2(X)}{s \cdot \varepsilon^2 \cdot F_k^2} \overset{(3.2)}{\leq} \\
&\overset{(3.2)}{\leq} \frac{k \cdot n^{1-\frac{1}{k}} \cdot F_k^2}{s \cdot \varepsilon^2 \cdot F_k^2} = \\
&= \frac{k \cdot n^{1-\frac{1}{k}}}{s \cdot \varepsilon^2} \leq \delta.
\end{aligned}$$

Tehát nagyságrendileg $s = O\left(\varepsilon^{-2} \cdot \delta^{-1} \cdot k \cdot n^{1-\frac{1}{k}}\right)$ párhuzamos futtatás szükséges a kívánt (ε, δ)-közelítés eléréséhez.

Alkalmazzuk az előző fejezetben tárgyalt „mediántrükköt". Futassuk az előző, s darab futtatásból álló algoritmust t-szer egymástól függetlenül, párhuzamosan (tehát összesen $t \times s$ egységgel), és jelölje az így kapott közelítést $\tilde{X}$.

3.5. Tétel. *Megfelelő s és t választásával az algoritmus által meghatározott $\tilde{X}$ közelítésünkre teljesül, hogy:*

$$\mathbb{P}\left(\left|\tilde{X} - F_k\right| \geq \varepsilon \cdot F_k\right) \leq \delta.$$

Bizonyítás. A bizonyítás azonos a 2.6 tétel bizonyításával. □

Az s és t paraméterek megválasztásához is hasonlóan kell eljárnunk: a belső, átlagot használó algoritmust (ε, δ^+)-közelítőnek állítjuk be. Így adódik, hogy:

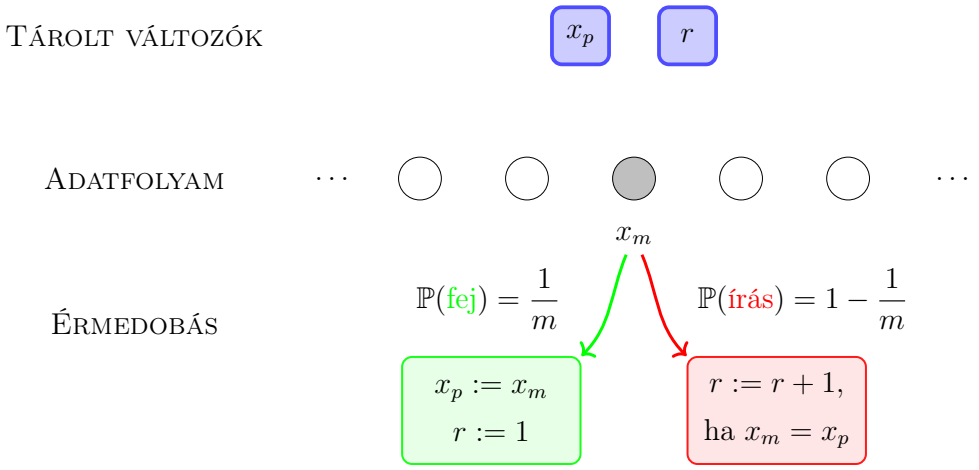
$$s = \frac{1}{\varepsilon^2} \cdot \frac{1}{\delta^+} \cdot k \cdot n^{1-\frac{1}{k}}$$
$$t = \frac{2}{\left(\frac{1}{2} - \delta^+\right)^2} \log \frac{1}{\delta}$$

Megjegyzés. Az összes egység $s \cdot t$ számának minimalizálásához a δ^+ értéket kell jól beállítanunk, a $\frac{2}{\left(\frac{1}{2} - \delta^+\right)^2 \cdot \delta^+}$ kifejezetést minimalizálva. Elemi eszközökkel a nevező a következő módon becsülhető: $\frac{1}{2} \cdot \left[(2\delta^+) \cdot \left(\frac{1}{2} - \delta^+\right) \cdot \left(\frac{1}{2} - \delta^+\right)\right] \leq \frac{1}{2} \cdot \left(\frac{1}{3}\right)^3$, és a maximum ekkor $2\delta^+ = \frac{1}{2} - \delta^+$ teljesülése esetén, azaz a $\delta^+ = \frac{1}{6}$ pontban vétetik fel; ekkor a tört értéke 108. A korábban tárgyalt Flajolet–Martin-algoritmusnak is ez az optimális paraméterválasztása.

3.3. Implementáció

Még nem ejtettünk szót arról, hogy hogyan lehet ténylegesen megvalósítani a tárgyalt algoritmust egy valódi adatfolyam esetén, amikor nincs lehetőségünk „visszanézni" a korábbi elemekre, és véletlenszerűen választani közülük egyet. Egyúttal arra is megoldást adunk, hogy az adatfolyam hosszának előzetes ismerete hogyan küszöbölhető ki. A bemutatott eljárás sampling-módszeren alapszik.

Ennek az alapgondolata az, hogy minden pillanatban nyilvántartjuk a szükséges változókat úgy, hogy azok tényleg olyanok legyenek, mintha csak most választottuk volna ki az x_p elemet egyenletes eloszlás szerint az eddigi adatfolyamból. Ezt úgy érjük, hogy valamennyiszer egy új elemet olvasunk, egy $\frac{1}{m}$-valószínűségű „érmedobással" döntünk arról, hogy ez az új elem legyen-e a kiválasztott, vagy maradjon a korábbi. A döntésünk alapján a 3.1 ábrán látható módon frissítjük az egyes eltárolt értékeket, ha szükséges. Az adatfolyam m hosszát az érmedobás kimenetelétől függetlenül folyamatosan nyilvántartjuk.



3.1. ábra. Az Alon–Matias–Szegedy-algoritmus implementációjának alapgondolata

Az érmedobás által azt érjük el, hogy ha eddig minden elem ugyanakkora valószínűséggel volt kiválasztva, akkor ez továbbra is teljesülni fog; az algoritmus működése az adatfolyam kezdetekor pedig nyilvánvaló: az első elem beolvasása után $x_p := x_1$ és $r := 1$, majd innentől az előzőek szerint haladhatunk tovább.

A végső algoritmusunk

$$s \cdot t = O\left(\frac{1}{\varepsilon^2} \cdot \log \frac{1}{\delta} \cdot k \cdot n^{1-\frac{1}{k}}\right)$$

egységgel, egységenként – a kiszámítást nem tekintve, csak az x_p és r tárolásához szükséges – legfeljebb $O(\log n + \log m)$ tárral, ezekből adódik a szakasz elején célul kitűzött tárigény.

3.4. A bemutatott módszerek áttekintése

Vonjuk le az előbbi két fejezetben bemutatott módszerek tanulságait. Mindkét esetben definiáltunk egy valószínűségi változót, amelynek várható értéke jó volt nekünk (előbbi esetén csak közel volt hozzá, utóbbinál meg is egyezett a becsülni kívánt értékkel). Ennek a változónak a szórását felülről becsültük a várható érték konstansszorosával, hogy aztán az átlagként definiált közelítésünk relatív hibáját meg tudjuk becsülni, illetve ennek alapján az elérendő maximális hibavalószínűséghez szükséges futtatások számát meghatározni. Ezt követően vetettük be a „mediántrükköt”, amelynek köszönhetően a tárigényben megjelenő $\frac{1}{\delta}$ tényezőt mindkét esetben $\log \frac{1}{\delta}$-ra tudtuk cserélni. A következő fejezetben egy merőben eltérő megközelítést igénylő problémacsaládra térünk át.

4. fejezet

Gyakori elemek keresése

4.1. Problémafelvetés

A felhasználás céljától függően többféle módon értelmezhetjük a gyakori elemeket. Például egy szerver hálózati adatforgalmának vizsgálata során értelmes kérdés lehet egy adott időablakon belül a sok kapcsolatot létesíteni akaró kliensek azonosítása egy lehetséges DoS-támadás kivédése érdekében. Mi most az egész adatfolyamra nézve gyakran előforduló elemeket szeretnénk megtalálni, azonban bármikor lekérdezhető módon.

4.1. Definíció. Legyen $s \in (0,1)$ egy rögzített szám, amit küszöbnek nevezünk. Egy adatfolyamban lévő e elemet nevezzünk *gyakori elemnek*, ha az eddig beolvasott rekordok legalább s-ed részével azonos; azaz ha az eddigi előfordulásainak c_e számára teljesül, hogy $c_e \geq sN$.

A következőkben két közelítő algoritmust mutatunk be, amelyeknél a fent definiált s küszöbön kívül a bemenet részét képezi egy $\varepsilon \in (0, s)$ szám is – tipikusan $\varepsilon \ll s$ –, amelyet *hibaparaméternek* nevezünk. A kimenet egy $(e, \tilde{c})$-párokból álló halmaz, amelyek egy adatfolyamban előforduló e elemet, és annak az eddigi előfordulásainak becsült $\tilde{c}$ számát jelölik. Ezekkel szemben a következő elvárásokat támasztjuk:

1. A kimenet részét képezi minden gyakori elem, azaz nincsenek *hamis negatív* eredmények.

2. Nem része a kimenetnek olyan e elem, melyre

$$c_e < (s - \varepsilon)N,$$

azaz a tévesen visszaadott elemek esetében sem tévedünk az ε hibaparaméter szerint megengedettnél többet.

3. Adott e elem becsült és valódi előfordulásainak számára teljesül, hogy

$$0 \leq c_e - \tilde{c} \leq \varepsilon N,$$

azaz a becsült értékek nem többek, és legfeljebb εN-nel kevesebbek a valódi értéknél.

4.2. A Lossy Counting algoritmus

Az elsőként bemutatott algoritmus Gurmeet Singh Manku és Rajeev Motwani nevéhez fűződik [4].

A fő adatszerkeszetünk egy $\mathcal{D}$ jelű halmaz, amely $(e, \tilde{c}, \Delta)$ alakú elemekből áll: e és $\tilde{c}$ azonos jelentéssel bír, mint korábban, míg Δ-val a $\tilde{c}$-re vonatkozó közelítésünkben fellépő maximális hibát fogjuk felülről becsülni. Inicializáljuk $\mathcal{D}$-t üres halmazként.

Osszuk az érkező adatfolyamot $w = \left\lceil \frac{1}{\varepsilon} \right\rceil$ szélességű blokkokra, és számozzuk is be őket 1-től kezdődően az aktuális $b_{\text{current}} = \left\lceil \frac{N}{w} \right\rceil$-ig. Amikor egy e elemet olvasunk az adatfolyamból, ellenőrizzük, hogy szerepel-e már $\mathcal{D}$-ben: ha nem szerepel, vegyük fel egy új $(e, 1, b_{\text{current}} - 1)$ rekordként; ha szerepel, akkor növeljük meg a hozzá tartozó $\tilde{c}$ értéket eggyel.

Minden blokk végén, azaz amikor $w|N$, végzünk egy szűrést $\mathcal{D}$ elemein: egy $(e, \tilde{c}, \Delta)$ elemet pontosan akkor törlünk, ha

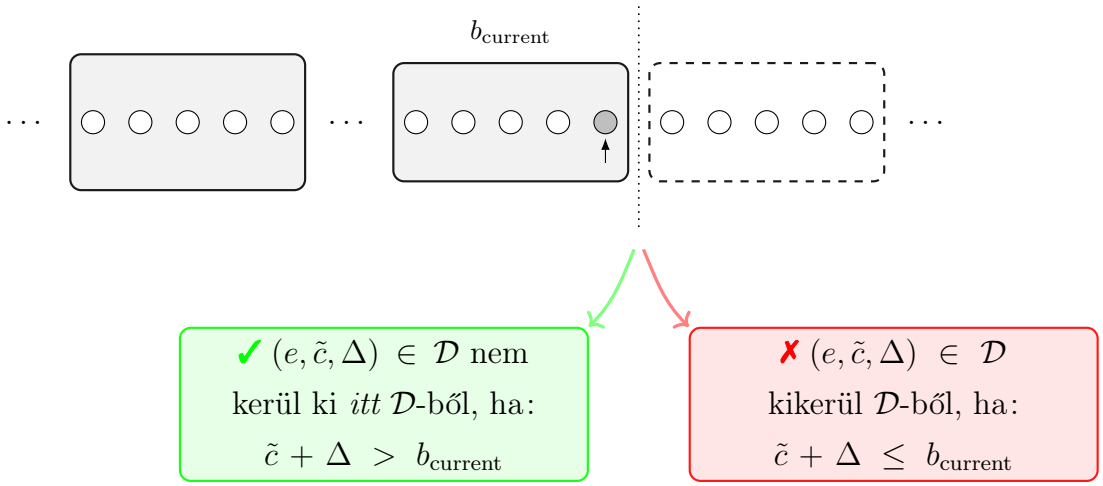
$$\tilde{c} + \Delta \leq b_{\text{current}}. \tag{4.1}$$

Erre az egyenlőtlenségre a továbbiakban *törlési feltételként* fogunk hivatkozni.

Bármikor, amikor le szeretnénk kérdezni a közelítő megoldásunkat, a következő halmazt adjuk vissza:

$$\{(e, \tilde{c}) : (e, \tilde{c}, \Delta) \in \mathcal{D} \quad \text{és} \quad \tilde{c} \geq (s - \varepsilon)N\}.$$

Intuíció szintjén azért törölhetünk bizonyos elemeket, mert például ha egy elemet ugyanannak a blokknak a végén törlünk, mint amelyikben bekerült, akkor az csak úgy lehetséges a törlési feltétel alapján, ha az pontosan egyszer szerepelt ebben a blokkban:



4.1. ábra. Az algoritmus minden blokk határán végez egy szűrést $\mathcal{D}$ elemein

egyetlen ilyen elemtől pedig egy $w = \left\lceil \frac{1}{\varepsilon} \right\rceil$ széles blokkból gond nélkül megválhatunk úgy, hogy az ε paraméter által meghatározott relatív hibahatáron belül maradjunk.

Világos, hogy egy $(e, \tilde{c}, \Delta) \in \mathcal{D}$ rekordban $\tilde{c}$ megegyezik a bekerülése óta történt előfordulásainak számával. Korábban azt mondtuk, hogy a Δ értékekkel a közelítésünkben fellépő maximális hibát szeretnénk felülről becsülni. Miért becsülhetjük felül a hibánkat az újonnan bekerülő elemek esetében $(b_{\text{current}} - 1)$-gyel?

Hasonlóan az előző érveléshez, egy éppen törlésre kerülő rekord eleme legfeljebb annyiszor fordulhatott elő az adatfolyamban a bekerülése óta, mint amennyi blokk óta az $\mathcal{D}$-ben volt. Emiatt egy éppen bekerülő elem a korábbi $b_{\text{current}} - 1$ blokkban legfeljebb ugyanennyiszer szerepelhetett úgy, hogy legkésőbb a $(b_{\text{current}} - 1)$-edik blokk végén kikerüljön. Tehát teljesül az, amit az adatszerkezetünktől vártunk a Δ értékek vonatkozásában:

4.2. Állítás. *Az algoritmus futása során minden $(e, \tilde{c}, \Delta) \in \mathcal{D}$ rekordra és az e elem valódi előfordulásainak c_e számára igaz, hogy:*

$$0 \leq c_e - \tilde{c} \leq \Delta.$$

A továbbiakban megmutatjuk, hogy a kimenet megfelel az elvárásainknak, azaz teljesíti a korábban támasztott három feltételünket.

4.3. Állítás. *Egy $(e, \tilde{c}, \Delta)$ elem törlésekor $c_e \leq b_{current}$.*

Bizonyítás. A blokkok számára vonatkozó teljes indukcióval igazoljuk az állítást. Korábban meggondoltuk, hogy a $b_{\text{current}} = 1$ esetben a (4.1) törlési feltétel alapján, ha törlünk

egy $(e, \tilde{c}, \Delta)$ rekordot, akkor $c_e = \tilde{c} = 1$, tehát ekkor teljesül az állítás. A továbbiakban tegyük fel, hogy $b_{\text{current}} > 1$, és hogy az ennél kisebb azonosítójú blokkokra már igazoltuk az állítást, és vizsgáljunk egy törlésre kerülő $(e, \tilde{c}, \Delta)$ rekordot. Mivel a bekerülése óta a Δ értéket nem változtattuk, ezért a $(\Delta + 1)$-edik blokkban került be $\mathcal{D}$-be. Az indukciós feltevés alapján tudjuk, hogy az azt megelező Δ blokkban, ha szerepelt egyáltalán, akkor is legfeljebb Δ alkalommal. Mivel a $\mathcal{D}$-be való bekerülése óta pontos a közelítésünk, ezért a következőt írhatjuk:

$$c_e \leq \tilde{c} + \Delta \leq b_{\text{current}},$$

ahol az utolsó egyenlőtlenség a törlési feltétel alapján teljesül, így igazoltuk az állítást. □

4.4. Következmény. *Ha egy e elem nem szerepel $\mathcal{D}$-ben, akkor $c_e \leq \varepsilon N$.*

Bizonyítás. Vegyük észre, hogy elegendő megmutatnunk, hogy egy éppen törlésre kerülő elem esetén igaz a fenti egyenlőtlenség. Felhasználva az előző állítást, valamint hogy törléskor $w|N$, a következőket írhatjuk:

$$c_e \overset{(4.3)}{\leq} b_{\text{current}} = \left\lceil \frac{N}{w} \right\rceil = \frac{N}{w} = \frac{N}{\left\lceil \frac{1}{\varepsilon} \right\rceil} \leq \varepsilon N, \tag{4.2}$$

ezzel igazoltuk az állítást. □

4.5. Lemma. *Bármely $(e, \tilde{c}, \Delta) \in \mathcal{D}$ rekordra és az e elem valódi előfordulásainak c_e számára teljesül, hogy*

$$\tilde{c} \leq c_e \leq \tilde{c} + \varepsilon N,$$

azaz teljesül a kimenettel szemben támasztott harmadik elvárásunk is.

Bizonyítás. A bal oldali egyenlőtlenség triviálisan teljesül, a továbbiakban csak a jobb oldalival foglalkozunk. Ha $\Delta = 0$, akkor a közelítésünk pontos, tehát $\tilde{c} = c_e$.

Ha $\Delta > 0$, akkor a rekord a $(\Delta+1)$-edik blokkban került be, de lehet, hogy már előtte is szerepelt az e elem az adatfolyamban, csak a hozzá tartozó rekord törölve lett: 4.3 alapján amikor ez megtörtént, $c_e \leq \Delta$ volt, hiszen ez csak az első Δ blokk valamelyikében történhetett meg. Ez alapján

$$c_e \leq \tilde{c} + \Delta.$$

Az utóbbi, Δ tagot triviálisan felülről becsülhetjük $(b_{\text{current}} - 1)$-gyel. Ha a 4.4 állítás bizonyításában a (4.2) egyenlőtlenségben b_{current} becslésekor, nem használjuk ki, hogy

$w|N$, akkor azt kapjuk, hogy $b_{\text{current}} \leq \lceil \varepsilon N \rceil$, ahonnan a $b_{\text{current}} - 1 \leq \varepsilon N$ becslés adódik. Az előző egyenlőtlenségeinket összefűzve:

$$c_e \leq \tilde{c} + \Delta \leq \tilde{c} + (b_{\text{current}} - 1) \leq \tilde{c} + \varepsilon N. \qquad \square$$

A fentiekben bizonyítottuk az algoritmus jóságát, azaz hogy minden korábban támasztott elvárásunknak megfelel. A következő tételben a tárigényre, a használt $\mathcal{D}$ adatszerkezet méretére vonatkozóan teszünk megállapítást.

4.6. Tétel. *Az algoritmus futása során minden pillanatban teljesül, hogy*

$$|\mathcal{D}| \leq w \left(1 + \log \lceil \varepsilon N \rceil\right).$$

Bizonyítás. Minden $i = 1, \ldots, b_{\text{current}}$ egészre jelölje d_i azon $\mathcal{D}$-beli $(e, \tilde{c}, \Delta)$ rekordok számák, amelyekre $\Delta = b_{\text{current}} - i$, azaz amelyek a $(b_{\text{current}} - i + 1)$-edik blokkban kerültek be $\mathcal{D}$-be, azaz a végéről számolva az i-edikben.

Vegyük észre, hogy minden ilyen rekordbeli e elem legalább i-szer kellett, hogy szerepeljen az utolsó i blokkban, hiszen $i \geq 2$ esetén a bentmaradáshoz a

$$\tilde{c} + \Delta = \tilde{c} + (b_{\text{current}} - i) > b_{\text{current}} - 1$$

feltételnek kellett teljesülnie – a (4.1) törlési feltétel alapján –, ahonnan $\tilde{c} \geq i$ adódik. Az $i = 1$ eset triviális, hiszen ahhoz, hogy egy elem az utolsó blokkban bekerüljön $\mathcal{D}$-be, szerepelnie is kellett legalább egyszer.

Vizsgáljuk az utolsó j blokkban ($j = 1, \ldots, b_{\text{current}}$) érkezett legfeljebb $j \cdot w$ darab elemet. Ezek számát a fentiek alapján alulról becsülhetjük az előzőleg bevezetett d_i értékek segítségével, így a következő egyenlőtlenség adódik:

$$\sum_{i=1}^{j} i \cdot d_i \leq j \cdot w. \tag{4.3}$$

A következőkben ennek az egyenlőtlenségnek a felhasználásával igazoljuk, hogy bármely $j = 1, \ldots, b_{\text{current}}$ esetén a következő is teljesül:

$$\sum_{i=1}^{j} d_i \leq \sum_{i=1}^{j} \frac{w}{i} \tag{4.4}$$

Ha ezt beláttuk, akkor a $j = b_{\text{current}}$ választással azonnal adódik a bizonyítandó állítás, hiszen:

$$\begin{aligned}
|\mathcal{D}| = \sum_{i=1}^{b_{\text{current}}} d_i \overset{(4.4)}{\leq} \sum_{i=1}^{b_{\text{current}}} \frac{w}{i} \leq \\
\leq w \cdot \left(1 + \int_1^{b_{\text{current}}} \frac{1}{x}\,\mathrm{d}x\right) = \\
= w\,(1 + \log b_{\text{current}}) \leq \\
\leq w\,(1 + \log \lceil \varepsilon N \rceil)\,.
\end{aligned}$$

A (4.4) egyenlőtlenséget j szerinti teljes indukcióval igazoljuk. A $j = 1$ esetben triviálisan igaz az állítás. Tegyük fel, hogy $j = 1$-től valamely $j = (p-1)$-ig már igazoltuk az állítást, megmutatjuk, hogy ebből következik, hogy $j = p$-re is igaz. Írjuk fel a (4.3) egyenlőtlenséget $j = p$-re, a (4.4) egyenlőtlenséget pedig $j = 1, \ldots, p-1$-re, és adjuk össze ezeket. Így a következő egyenlőtlenség adódik:

$$\underbrace{\sum_{i=1}^{p} i \cdot d_i}_{(4.3)\ j=p} + \underbrace{\sum_{j=1}^{p-1}\sum_{i=1}^{j} d_i}_{(4.4)\ j=1,\ldots,p-1} \leq \underbrace{p \cdot w}_{(4.3)\ j=p} + \underbrace{\sum_{j=1}^{p-1}\sum_{i=1}^{j} \frac{w}{i}}_{(4.4)\ j=1,\ldots,p-1}$$

Hozzuk mindkét oldalt egyszerűbb alakra:

$$\sum_{i=1}^{p} i \cdot d_i + \sum_{i=1}^{p-1} (p-i) d_i \leq p \cdot w + \sum_{i=1}^{p-1} (p-i) \frac{w}{i}.$$

További átalakításokat követően:

$$\begin{aligned}
\sum_{i=1}^{p} p \cdot d_i \leq p \cdot w + \sum_{i=1}^{p-1} \left(\frac{p \cdot w}{i} - w\right) = \\
= p \cdot w + \sum_{i=1}^{p-1} \frac{p \cdot w}{i} - (p-1)w = \sum_{i=1}^{p-1} \frac{p \cdot w}{i} + w = \sum_{i=1}^{p} \frac{p \cdot w}{i}.
\end{aligned}$$

Az egyenlőtlenség mindkét oldalát p-vel osztva adódik a bizonyítandó állítás $j = p$-re, amiből, mint azt fentebb láttuk, következik a tétel állítása. □

5. fejezet

Gyakori termékhalmazok keresése

5.1. Problémafelvetés

A következőkben az előző fejezetben bemutatott algoritmus módosításának lehetséges gyakorlati felhasználását mutatjuk be. Legyenek a vizsgált adatfolyam rekordjai egy előre adott $\mathcal{I}$ halmaz részhalmazai.

Jelölje tetszőleges $I \subset \mathcal{I}$ halmaz esetén support(I) azt a számosságot, hogy az adatfolyamban hány elem tartalmazza I-t részhalmazként. A feladatunk olyan halmazok keresése, amelyek az összes eddigi N darab halmaz közül legalább sN-ben szerepelnek részhalmazként, tehát az előző jelölés használatával support(I) $\geq sN$. A szokásos elnevezést követve az s értéket minimális supportnak nevezzük, de egy (0,1) intervallumbeli értékként tekintünk rá.

A rekordokra, mint halmazokra kosarakként is hivatkozhatunk, míg az elemeire termékekként. Ez az elnevezés a probléma egyik lehetséges gyakorlati megközelítéséből ered. Eszerint a kosarak megfeleltethetőek egy áruház látogatói által megvásárolt termékeknek, és az áruház szeretné feltárni a vásárlási adatokból, hogy mik azok a termékek, amelyeket gyakran vásárolnak együtt. Ez az információ már önmagában is értékes lehet az áruháznak az árazási stratégiája szempontjából, azonban a következőkben bemutatott asszociációs szabályok segítségével még kifinomultabb módszerekre nyílik lehetősége.

5.1. Definíció. Egy $I \Rightarrow J$ kifejezést asszociációs szabálynak nevezünk, ahol $I, J \subset \mathcal{I}$ két diszjunkt termékhalmaz. Az asszociációs szabály confidence értéke:

$$\text{confidence}(I \Rightarrow J) = \frac{\text{support}(I \cup J)}{\text{support}(I)}$$

Előbbi érték annak a tapasztalati feltételes „valószínűségnek” felel meg, hogy egy adott I-t részhalmazként tartalmazó kosár tartalmazza J-t is. Ezzel a fogalommal az is egy érdekes feladat, hogy megkeressük egy adott küszöbérték feletti confidence értékkel rendelkező asszociációs szabályokat. Gyakran elegendő csak az $I \Rightarrow \{i\}$ alakúakat azonosítani, ahol $I \subset \mathcal{I}$ és $i \in \mathcal{I}$. Ezáltal lehetősége nyílik az áruháznak olyan jellegű leárazásokra, amikor I elemei közül többnek csökkenti az árát, míg i-nek megemeli, ezáltal – legalábbis ebből a szempontból – növelve a bevételeit (vagy éppen csak csökkentve a bevételkiesését).

A továbbiakban csak az eredeti kérdésfelvetéssel, a gyakori termékhalmazok közelítő értelemben vett keresésével foglalkozunk.

5.2. Algoritmikus nehézségek és egy lehetséges megközelítés

A gyakori termékhalmazok keresése kapcsán az egyik legnagyobb nehézséget a tárigény jelenti, hiszen $\mathcal{I}$ összes részhalmazának előfordulásainak a számát nyilvántartani meglehetősen költséges volna. Véges adathalmazok esetén az apriori algoritmus számított úttörőnek, ennek az alapgondolata az, hogy a már megtalált $(k-1)$-elemű gyakori termékhalmazok alapján keresi meg a lehetséges k-eleműeket – azonban minden ilyen iterációhoz szükséges végigolvasnia az algoritmusnak az összes rekordot. Azóta sikerült már olyan algoritmusokat is találni, amelyek hatékonyabbak ebből a szempontból, például pontosan kettő végigolvasással érik el ugyanezt az eredményt, ilyen például a TOIVONEN-algoritmus.

Mi azonban, mint a korábbiakban is, olyan algoritmust szeretnénk, aminek nem szükséges több végigolvasás, és bármikor le tudjuk kérdezni a jelenlegi eredményt: ehhez ismételten feladjuk az egzakt eredmény iránti elvárásainkat. Módosítsuk az előző fejezetben tárgyalt algoritmust, és igazítsuk ehhez a modellünket. Az input részét képezi ugyanúgy az s minimális support és az ε hibaparaméter.

A rendelkezésre álló memóriát fogjuk fel egy bufferként, amibe megpróbálunk a lehető legtöbb – előző fejezet szerinti – blokkot betölteni. A blokkonkénti szűrések helyett végezzünk most csak akkor szűrést, amikor betelik a buffer: jelölje ekkor a bufferben lévő blokkok számát β. Ez az érték változhat az algoritmus futása során, vagy a termékhalmazok méretéből adódóan, vagy pedig az algoritmus többi részének memóriaigénye miatt. Amikor betelt a buffer, készítsük el a memóriában lévő halmazok alapján a *potenciálisan gyakori* termékhalmazokat, és az előzőekhez hasonlóan töröljük azokat, amelyeket a

hibaparaméterből adódóan elhagyhatunk:

– Minden $(I, \tilde{c}, \Delta) \in \mathcal{D}$ rekordban növeljük a $\tilde{c}$ közelítést az I halmaz a bufferben való előfordulásainak számával. Ha ezután $\tilde{c} + \Delta \leq b_{\text{current}}$, akkor töröljük a rekordot $\mathcal{D}$-ből.

– Ha egy I halmaz előfordulásainak a száma a bufferben $\tilde{c}$, még nincs neki megfelelő rekord $\mathcal{D}$-ben, és $\tilde{c} \geq \beta$ teljesül, akkor vegyük fel egy $(I, \tilde{c}, b_{\text{current}} - \beta)$ rekordként $\mathcal{D}$-be.

Az eredeti algoritmushoz hasonlóan itt is minden gyakori halmaz bekerül $\mathcal{D}$-be, és minden $(I, \tilde{c}, \Delta) \in \mathcal{D}$ rekordra teljesül, hogy

$$\tilde{c} \leq \text{support}(I) \leq \tilde{c} + \Delta.$$

Bármikor, amikor le szeretnénk kérdezni a közelítő megoldásunkat, a következő halmazt adjuk vissza:

$$\{(I, \tilde{c}) : (e, \tilde{c}, \Delta) \in \mathcal{D} \quad \text{és} \quad \tilde{c} \geq (s - \varepsilon)N\}.$$

A bufferben lévő halmazok generálása továbbra is egy nehéz, sok technikai jellegű optimalizálást is igénylő probléma, ezt teljes mélységében nem tárgyaljuk. Az eredeti cikk ehhez először növekvő sorrendbe rendezi az egyes termékhalmazokat, és folytonosan elhelyezi őket a memóriában. Ehhez feltesszük, hogy $\mathcal{I}$ egy adott, természetes számokból álló halmaz.

A $\mathcal{D}$ halmazt egy trie (vagy másnéven prefix fa) adatszerkezetként valósítjuk meg, amelyben az egyes csúcsok a lehetséges részhalmazoknak felelnek meg. A mélyebb szinteken lévő halmazok a nagyobb számosságoknak felelnek meg, és minden gyermek az ősének (élnek megfelelő elemmel való) bővítése. Tehát például a gyökérben az üres halmaz szerepel, míg annak gyermekei az egyelemű halmazok, azoknak pedig a megfelelően bővített kételemű halmazok, és így tovább.

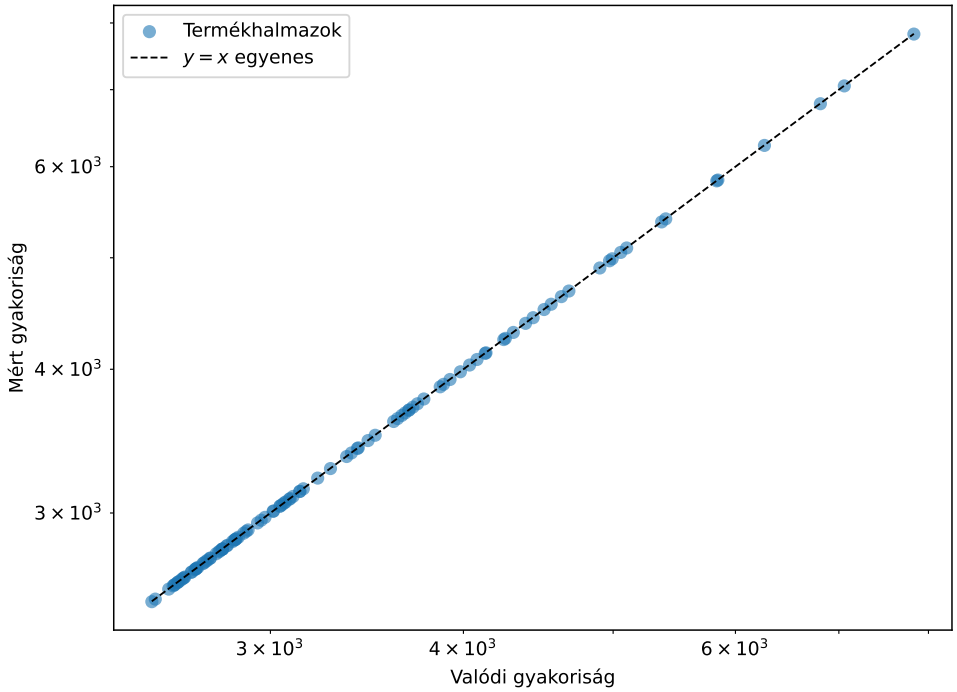
Ezt követően pointereket állítunk a memóriában folytonosan elhelyezett termékhalmazok elejére, és ezeket megfelelően léptetgetve számoljuk meg lexikografikus sorrendben az egyes halmazok gyakoriságát a bufferben. Ebben ugyanaz az elv szab határt a generált halmazok számának, mint ami az apriori algoritmus egyik alapgondolata is: ha egy halmaz gyakori, akkori annak minden részhalmaza is az; és ebből adódóan, ha egy halmaz nem gyakori, akkor semmilyen olyan halmaz sem az, amelyik tartalmazza őt részhalmazként.

Az előbbiekben vázolt algoritmus segítségével tehát mindössze egyetlen végigolvasással közelítő megoldást kaphatunk a gyakori termékhalmazokra.

5.3. Implementáció

A fenti algoritmust kisebb módosításokkal valósítottam meg. Az implementációnak nem volt célja, hogy sebességét tekintve versenytársa legyen a már létező (többszöri végigolvasást igénylő), kiforrott szoftvereknek; illetve nem volt szempont az sem, hogy a memóriaigényt az eredeti cikk szerinti módokon a végletekig redukáljam, így a Python nyelv használata mellett döntöttem.

Az algoritmus futását $s = 0{,}025$ és $\varepsilon = s/10$ paraméterek mellett egy 100 000 termékhalmazból álló, szintetikusan generált adathalmazon teszteltem. A megtalált, valódi gyakori termékhalmazok valódi és mért gyakorisága az 5.1. ábrán látható.



5.1. ábra. Logaritmikus skálán a gyakori termékhalmazok valódi és mért gyakorisága

Szembetűnő, hogy a két érték gyakorlatilag megegyezik, ami jól reprodukálja az eredeti cikk szerinti tapasztalatokat. Ennek az az oka, hogy a gyakori termékhalmazok tipikusan már az első bufferben is kellően gyakoriak ahhoz, hogy bekerüljenek a $\mathcal{D}$ adatszerkezetbe; és miután bekerültek, elég valószínűtlen, hogy törlődjenek onnan.

6. fejezet

Összegzés

Dolgozatomban a nagy adathalmazok kezelésével járó különböző nehézségeken keresztül jártam körül néhány adatfolyamokkal kapcsolatos becslési problémát, illetve közelítő eljárást. A feltett kérdések könnyen értelmezhetőek voltak, de a válasz hatékony becsléséhez összetettebb eszközön keresztül jutottunk el.

A dolgozat zárásaként a gyakori termékhalmazok keresésére vonatkozó, egyszeri végigolvasást igénylő algoritmus gyakorlati alkalmazhatóságát mutattam be azt illusztrálva, hogy egy tesztadathalmazon mennyire szolgáltat ténylegesen jó eredményt.

Irodalomjegyzék

[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996.

[2] Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

[3] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive data sets*. Cambridge University Press, 2020.

[4] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. 2002.

[5] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge University Press, 1995.

[6] Nicola Prezza. Algorithms for massive data – lecture notes. 2023.

NYILATKOZAT

Név: Apagyi Dávid

ELTE Természettudományi Kar, szak: Matematika BSc

NEPTUN azonosító: XMV30S

Szakdolgozat címe:
Gépi tanulási feladatok nagy adathalmazokon

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2023. június 07.

Apagyi Dávid

a hallgató aláírása