Nagyméretű adathalmazok kezelése

Adatfolyamok

Mi az adatfolyam?

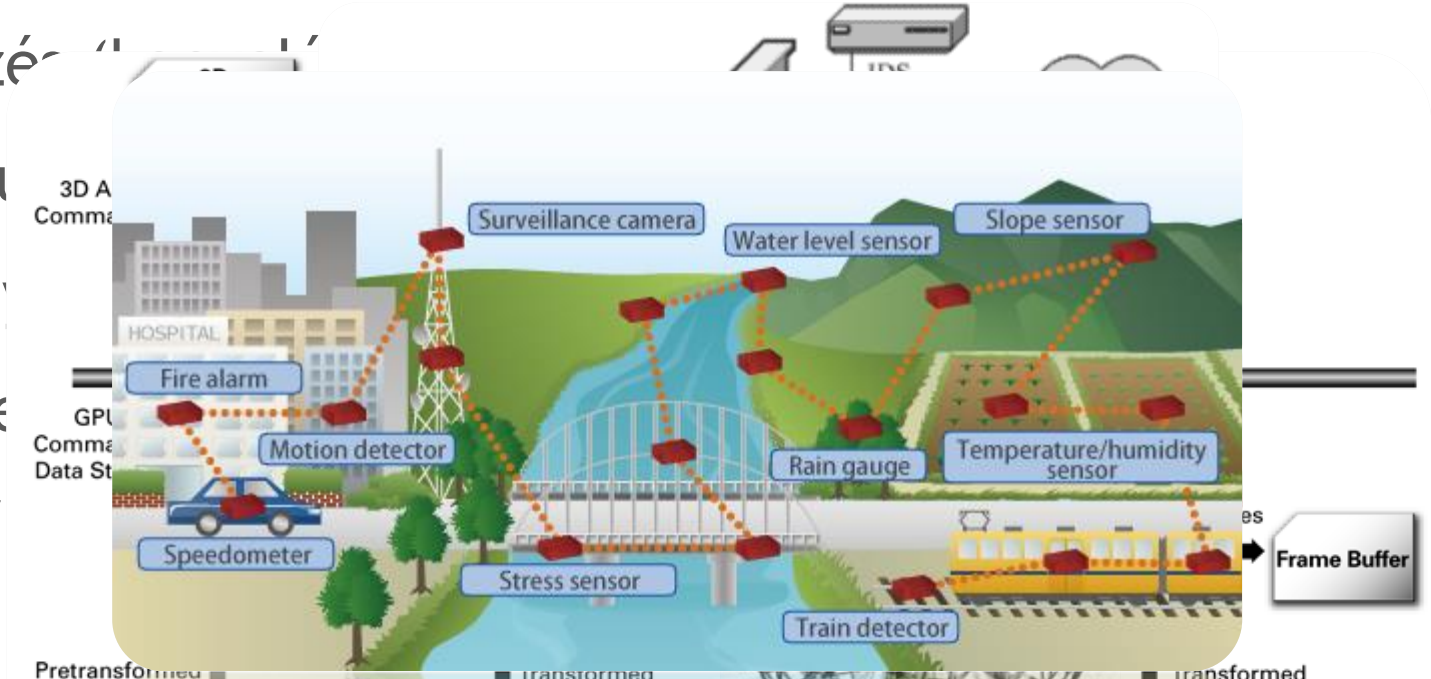
A hagyományostól eltérő adattárolási forma

- Csak egyszer olvashatjuk az adatrekordokat
- Gyorsan kell lefutnia a lekérdezéseknek, mert jön a következő adatrekord
- Historikus adatokat is fel kell használni

Jellemzők:

- Folyamatos adatáramlás
- Az adatfolyam nyíltvégű (nyílt forrás/nyelő)
- Általában automatizált forrás
- Nagyon nagy mennyiségű adat (kis részét dolgozzuk fel)
- Korreláló adatok (akkumulatív adatmodell)
- (Általában!) Online/azonnali feldolgozást igényel

- TCP/IP hálózatok
- Szenzorhálózatok
- GPU pipeline
- Filterezé
- Elő- és u
- Esemén
- Hibajele
- Report/



Az adatelemek sorrendjére nincs kihatásunk és nem ismerjük az adathalmaz méretét.

Bizonyos tendenciák felismerése lehetővé teszi a jövőbeli rekordok “jóslását” (CPU branch prediction)

A becslések sosem pontosak, hibák merülnek fel (egy “hibátlan” és “elég gyors” algoritmus csupán elméleti síkon létezik, összehasonlítási alapul szolgálhat)

Pár egyszerű probléma

- Számoljuk meg egy bitfolyamban a legutóbbi N bit Hamming súlyát (1-esek száma)!
- Adjuk össze a legutolsó N számot, ami a [0..R] intervallumból van!
- Számoljuk meg, hány különböző szimbólum fordul elő a legutóbbi N rekordban!

Ez mind egyszerű, ha a legutóbbi N elem befér a memóriába. De mi van, ha kevesebb, mint O(N) tárhellyel kell megoldani őket?

Közelítő algoritmusokat kell alkalmazni. Ezek elfogadhatóak, ha:

> Adott pozitív $\varepsilon < 1$ és $\delta < 1$ mellett a becsült érték $1 - \delta$ valószínűséggel legfeljebb ε hibát tartalmaz.

A szükséges tárhely és idő ezen paraméterek függvénye

A hiba lehet abszolút és relatív:

Abszolút hiba: $X - \varepsilon < E(X) < X + \varepsilon$
Relatív hiba: $(1 - \varepsilon) * X < E(X) < (1 + \varepsilon) * X$

Valszám 1.

Legyen X tetszőleges véletlen változó, és legyen c > 0 tetszőleges valós konstans. Ekkor teljesülnek az alábbi egyenlőtlenségek:

- Markov egyenlőtlenség:

$$P(|X| \geq (\delta = c * E(X))) \leq \frac{E(|X|)}{\delta} = \frac{1}{c}$$

- Csebisev egyenlőtlenség:

$$P(|X - E(X)| \geq (\varepsilon = c * \sigma(x))) \leq \frac{\sigma^2(X)}{\varepsilon^2} = \frac{1}{c^2}$$

Valszám 2.

Chernoff-korlát:

Legyenek $X_1, X_2, \ldots, X_n$ független, Bernoulli próbák. Tételezzük fel, hogy $P(X_i = 1) = p_i$. Legyen $X_s = \sum_{i=1}^{n} X_i$ valószínűségi változó, melynek várható értéke $E(X_s) = \sum_{i=1}^{n} np_i$. Ekkor $\forall \delta > 0$ -ra:

$$P(X_s > (1+\delta) * E(X_s)) \leq \frac{e^{\delta}}{(1+\delta)^{1+\delta}}^{E(X_s)}$$

Innen levezethető az abszolút hiba is:

$$\varepsilon \leq \sqrt{\frac{3\overline{E(X)}}{n} \ln(\frac{2}{\delta})}$$

Valszám 3.

Hoeffding-korlát:

Legyenek $X_1, X_2, .. X_n$ független valószínűségi változók. Tegyük fel, hogy minden x_i korlátos, azaz $P(X_i \in R = [a_i, b_i])$ = 1. Legyen $S = \frac{1}{n}\sum_{i=1}^{n} X_i$. Ekkor $\forall \varepsilon > 0$ - ra:

$$P(S - E(S) > \varepsilon) \leq e^{-\frac{2n^2\varepsilon^2}{R^2}}$$

Innen az abszolút hiba:

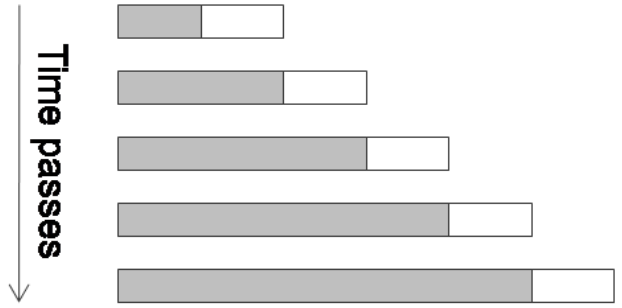
$$\varepsilon \leq \sqrt{\frac{R^2 \ln \frac{2}{\delta}}{2n}}$$

Miért jó ez?

- Felső korlátot ad a közelítő algoritmusok hibájára
- A Chernoff- és a Hoeffding-korlát exponenciálisan csökkenő korlátok
- Utóbbi kettő független a változók eloszlásától, ezért jól használhatóak a gyakorlatban.

Timing windows

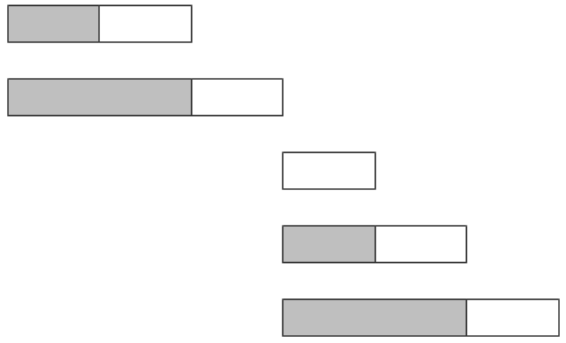
- Ha minden bejövő adatcsomagot ellátunk egy időbélyeggel (elosztott rendszerben ez sem triviális), beszélhetünk a csomagok “frissességéről”. Gyakori feladat az N legfrissebb tuple valami jellemzőjét mérni.
- Pl. Landmark window. A mérés eleje óta beérkezett csomagokat aggregálja minden lekérdezésnél. Nagyon tárhelyigényes, ezért nem túl praktikus.



Landmark window

Jumping window

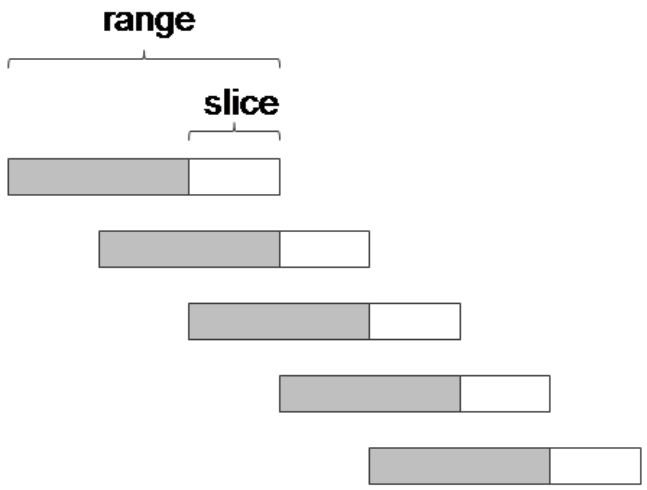
- Ennél valamivel jobb a jumping window. Bizonyos “checkpoint”-oknál reseteli az összes tárolt értéket, egyébként minden lekérdezés a legutolsó checkpoint/landmark óta bejött csomagokat aggregálja. (Pl napi jelentések)



Jumping window

Sliding window

- A tolóablakos megoldás is gyakori. Egy fix méretű ablakban tároljuk a legutolsó N csomagot és ezeket aggregáljuk minden lekérdezésnél.



Sliding window

Tilted window

- Mi van ha a korábbi adatokra is szükség van az aggregációhoz? Az előző példák a landmark kivételével mind “felejtő” ablakok (a landmark pedig túl sok memóriát fogyaszt)
- Minden korábban látott adatot el kéne tárolni, de a régi adatok nem olyan fontosak, mint a frissek. Ezért a régi adatokat tömörítve, de megőrizzük az ablakban.
- Natural/Logarithmic tilted window

Példa

Mintavételezés

- Célja, hogy lelassítsuk a beáramló adatot
- Az adatoknak csak egy részét dolgozzuk fel
- Éppen ezért fontos, hogy releváncs mintákat vegyünk

- Pl természetes illesztés eredményének méretbeli csökkentése.
- Mintát veszünk külön-külön a két adafolyamból és azokat illesztjük.
- Ha reprezentatív volt a két minta, az eredmény is az lesz.

Frekvencia momentumok

Adott nemnegatív k-ra, a k-dik frekvencia momentum:

$F^k = \sum_{i=1}^{v} m_i^k$, ahol m_i az i-dik fajta elem gyakorisága és v a különböző elemek száma.

- F^0: Hány különböző elem van?
- F^1: Hány elem van?
- F^2: Self-join size. Az egyes “vödrök” frekvenciáinak négyzetösszege.

A második momentum logaritmikus időben közelíthető. Lekérdezés-optimalizáló motorok használják főleg.

Hash sketch

- Képezzük le az $x \in [0, \dots, N-1]$ értékeket egyenletesen a $[0, \dots, 2^L - 1]$ -ba, ha $L = O(\log N)$ egy h(x) hash-függvénnyel.
- Jelölje lsb(x) az x bináris alakjában előforduló utolsó 1-es helyiértékét (utolsó~jobb szélső).
- Tároljunk egy L hosszú bitsorozatot, mely kezdetben csupa 0, majd egy x input hatására az lsb(h(x)) helyiértéken 1-esbe billentjük.

- Így kb. a (különböző) értékek fele a L. bitre képződik le (utolsó bit paritásától függ)
- Az értékek negyede a L-1-ik bitre, nyolcada az L-2-ikre és így tovább
- Ha R jelöli a bitsorozatunkban az aktuálisan utolsó 1-es helyiértékét, akkor $E(R) = \log \varphi d$, ahol d a különböző értékek száma a folyamban és $\varphi = 0.7735$.
- A fenti képletből megbecsülhető a folyamban előforduló (avagy a t. pillanatig látott) különböző szimbolúmok száma.

$$d = \frac{2^R}{\varphi}$$

Exponenciális hisztogram

- A tilted window-hoz hasonlóan itt is exponenciálisan csökkenő granularitással tároljuk az elemeket, vödrökben.
- Segítségével megszámlálhatjuk a bináris folyamba érkező egyeseket.
- Tárolunk 2 változót: TOTAL és LAST
  - TOTAL: az eddigi egyesek száma
  - LAST: Az utolsó tárolt vödör mérete
- Az 1-esek száma: TOTAL – LAST/2

- Ha a bejövő bit 0, nem foglalkozunk vele.
- Ha 1-es, beletesszük egy újonnan létrehozott 1 méretű vödörbe (melyet a megfelelő időbélyeggel látunk el). Ezzel együtt inkrementáljuk TOTAL-t.
- Adott ε parameter. Ha létezik $|\frac{1}{2\varepsilon}| + 2$ egyforma méretű vödör, egyesítsük a legutolsó kettőt (az új időbélyeg a frissebb érték lesz)!
- Ha LAST által referált vödröt egyesítettük, frissítjük LAST értékét (duplájára nő).

- Így minden lekérdezésnél kidobáljuk a memóriából a túl régi vödröket (az időbélyeg és a query window alapján).
- Egész pontosan meghatározunk egy N ablakméretet, azaz a legutolsó N elemben keressük az egyesek számát
- A maradékban TOTAL db 1-es van, de az utolsó vödör aggregált információt tartalmaz, átlagosan a fele esne bele ténylegesen az ablak által meghatározott időintervallumba.
- Ezért az 1-esek száma: TOTAL – LAST/2

Példa

Haar wavelet

- Az eredeti jelet rekonstruáljuk primitív alapjelek súlyozott kompozíciójaként.
- Ha a bemenet egy n hosszú számsorozat, alkossunk belőlük párokat (egymás utániak egy párba)
- Egy vektorban első n/2 komponensében tároljuk el minden számpárra: $\frac{a+b}{\sqrt{2}}$.
- Ugyanezen vektor további n/2 komponensében tároljuk el minden számpárra: $\frac{a-b}{\sqrt{2}}$.
- Ismételjük ezt addig, amíg csak egy “összeg komponens” marad.

Példa 1

Our input array will be these eight elements:

```
[  1.000   2.000   3.000   1.000   2.000   3.000   4.000   0.000  ]
```

After calculating the sums and differences:

```
[  2.121   2.828   3.536   2.828  -0.707   1.414  -0.707   2.828  ]
```

Recursing on the first half:

```
[  3.500   4.500  -0.500   0.500  -0.707   1.414  -0.707   2.828  ]
```

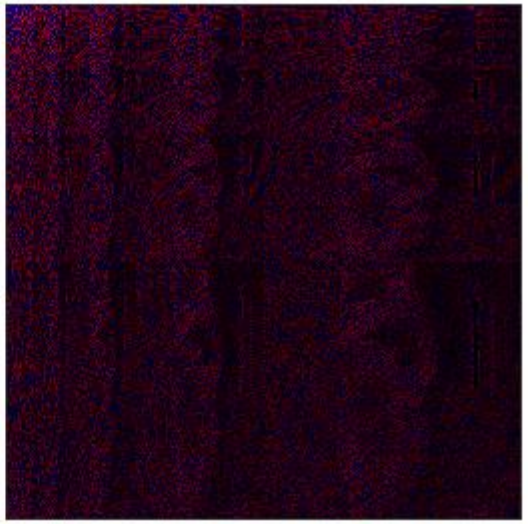
Recursing again gives us the output array:

```
[  5.657  -0.707  -0.500   0.500  -0.707   1.414  -0.707   2.828  ]
```

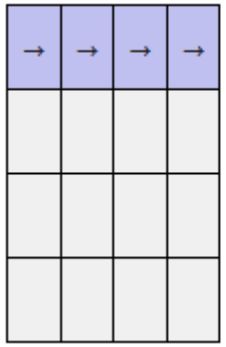
Képekre

- Először a sorokra
  transzormációt,

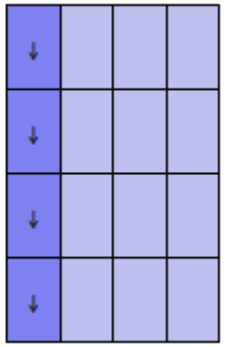
1. Original image



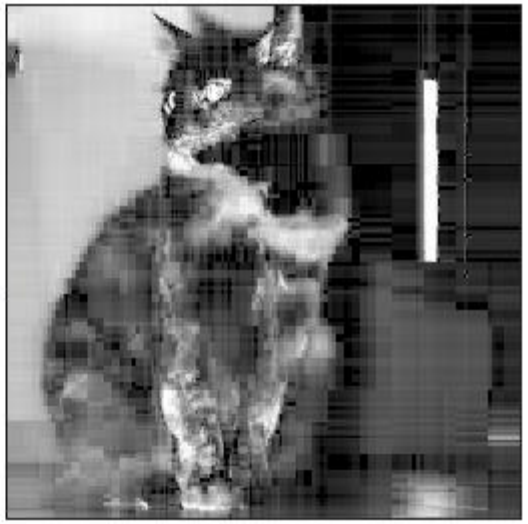
2. Haar coefficients



Majd



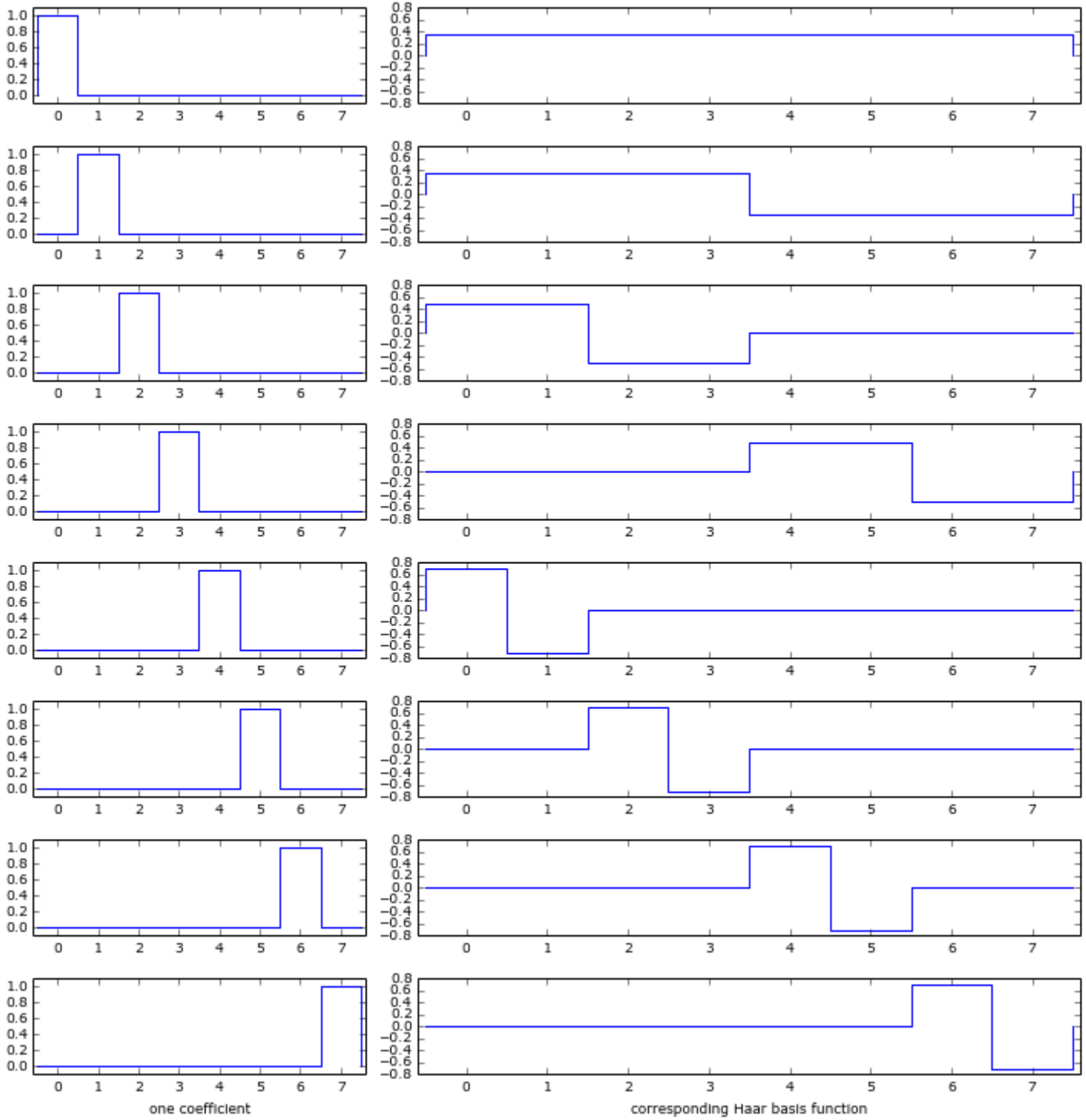
3. Fewer Haar coefficients



4. Reconstructed image

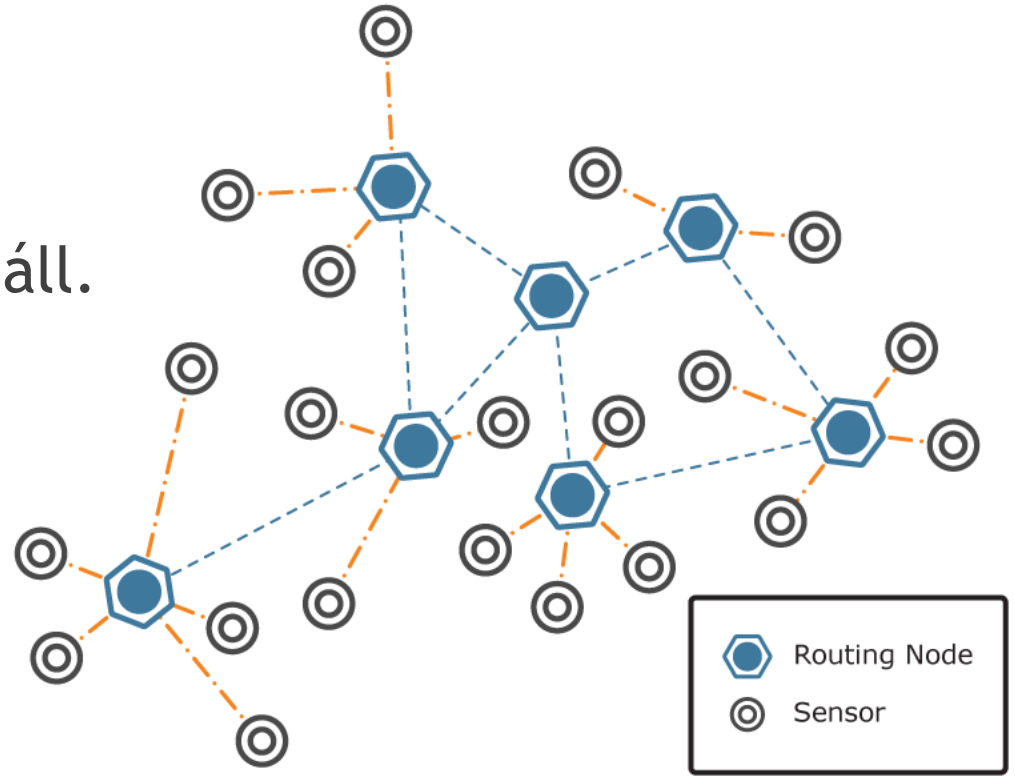
Példa

$$W = \begin{bmatrix} 1/\sqrt{8} & 1/\sqrt{8} & 1/2 & 0 & 1/\sqrt{2} & 0 & 0 & 0 \\ 1/\sqrt{8} & 1/\sqrt{8} & 1/2 & 0 & -1/\sqrt{2} & 0 & 0 & 0 \\ 1/\sqrt{8} & 1/\sqrt{8} & -1/2 & 0 & 0 & 1/\sqrt{2} & 0 & 0 \\ 1/\sqrt{8} & 1/\sqrt{8} & -1/2 & 0 & 0 & -1/\sqrt{2} & 0 & 0 \\ 1/\sqrt{8} & -1/\sqrt{8} & 0 & 1/2 & 0 & 0 & 1/\sqrt{2} & 0 \\ 1/\sqrt{8} & -1/\sqrt{8} & 0 & 1/2 & 0 & 0 & -1/\sqrt{2} & 0 \\ 1/\sqrt{8} & -1/\sqrt{8} & 0 & -1/2 & 0 & 0 & 0 & 1/\sqrt{2} \\ 1/\sqrt{8} & -1/\sqrt{8} & 0 & -1/2 & 0 & 0 & 0 & -1/\sqrt{2} \end{bmatrix}$$



Szenszorhálózatok

A hálózat *szenszorokból* és *útválasztó csomópnotokból* áll.



Quantile digest

- Hisztogram-szerű, vödrökben tárolja az adatok gyakoriságát (frekvenicáját)
- Nem egyenlő nagyságú intervallumot fednek le a vödrök, de közel egyenlő sok elem van bennük
- Célja a kvantilisek meghatározása.

Kvantilis: Adott egy *q* 0 és 1 közti szám. A kvantilis a *q*n*-edik szám az adatok nagyság szerinti növekvő sorában.

Quantile-digest

- Építsünk egy bináris partíciót reprezentáló fagráfot a(z) (tolóablakban lévő aktuális) adatokból
- Ennek egy részhalmaza lesz a q-digest (kezdetben csak a levelek, majd ezt tömörítjük).
- Minden q-digestbeli (v) vödörre (ahol k a tömörítési tényező):

$$count(v) \leq \left\lfloor {}^{n}/_{k} \right\rfloor$$

$$count(v) + count(v_p) + count(v_s) > \left\lfloor {}^{n}/_{k} \right\rfloor$$

- Járjuk be a fát alulról felfelé és ha nem teljesíti a második feltételt a 2 gyereket olvasszuk bele a szülőbe

Elosztott q-digest

- A routing tree levelei építenek egy q-digestet ezzel összegezve a náluk lévő adatokat. Csak ezt küldik tovább.
- A feljebb lévő node-ok a beérkező q-digestek unióját képzik (a megfelelő vödrök számlálóit összeadják).
- Mivel az aggregált összegzés már lehet, hogy sérti a q-digest tulajdonságait, alulról felfelé újraellenőrizzük.
- Az így kapott aggregált q-digestet küldi tovább a node a szülőnek, mely egész a routing root-ig elmegy.

Kvantilis lekérdezések

- Post-order bejárjuk a fát és render összegezzük a vödrök tartalmát (legyen ez *c*).
- Amint a *c* nagyobb (vagy egyenlő) lenne *q*n*-nél/nel, a legutoljára összegzett vödör felső korlátját választjuk a *q*-adik kvantilisnek.
- A túlcsorduló vödörben is lehetnek még olyan elemek, amik az “igazi” kvantilisnél előrébb vannak, ezért van hibája az algoritmusnak (ami felülbecsülhető).

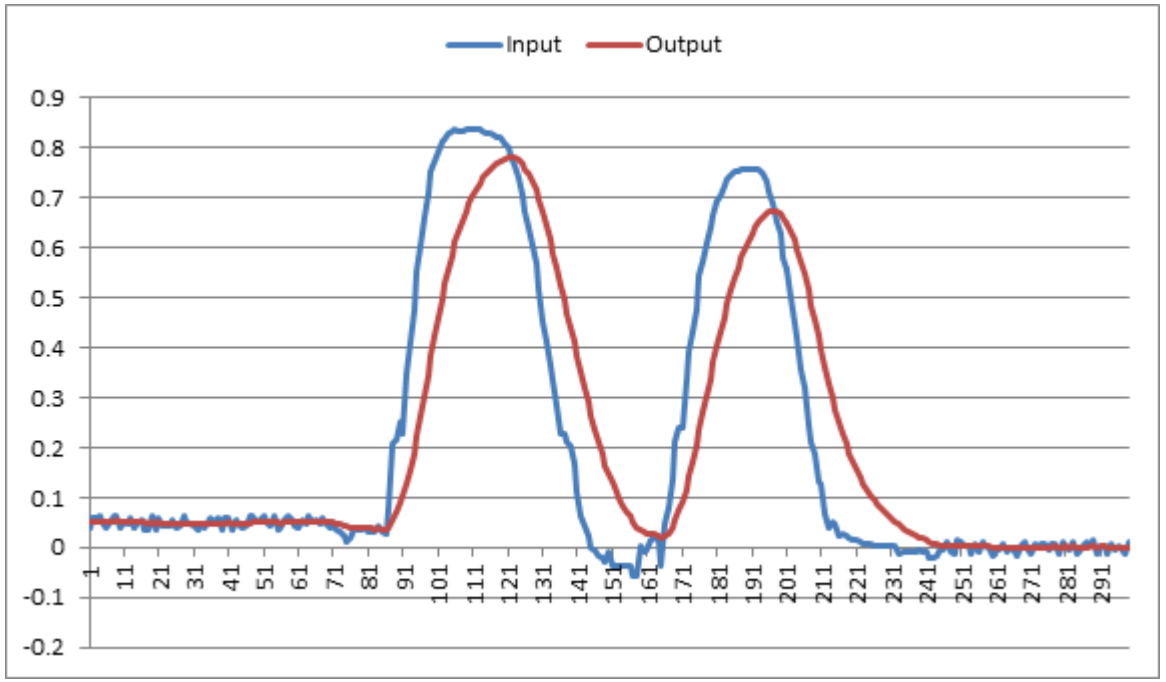
$$|r - qn| < \varepsilon n \; (ha \; k = \frac{1}{\varepsilon} \log n)$$

Esettanulmány

- Kinect zajszűrés (ízületek pozíciója, ami a mélységtérképből származtatott adat)

Esettanulmány

- A kis zajokat elsimítja (+)
- Késleltetve reagál a hirtelen változásokra (-)
- A maximumok és minimumok “csökkennek” (-)



ARMA filterek

- Auto Regressive Moving Average

Az n-edik output a legutolsó N darab input és a legutolsó M output súlyozott átlaga:

$$\hat{X}_n = \sum_{i=0}^{N} a_i X_{n-i} + \sum_{i=0}^{M} b_i \hat{X}_{n-i}$$

Moving Average (MA)

Hasonló, csak az outputok súlya mindenhol 0 (nemregresszív).

$$\hat{X}_n = \sum_{i=0}^{N} a_i X_{n-i}$$

Centrális mozgó átlag:

$$\hat{X}_n = \sum_{i=-M}^{N} a_i X_{n-i}$$

Utóbbinak szüksége van M darab “jövőbeli” mintára (inputra) is, ezért főként offline alkalmazásai vannak.

Számtani közép

$$\hat{X}_n = \frac{1}{N+1}\sum_{i=0}^{N} X_{n-i}$$

Ez csak kevésbé mozgó ízületekre jó (konstans mintára kiválóan “jósol”, hiszen konstans minta átlaga önmaga).

A mozgásokat nagy delay-jel követi (az állandó sebességűt is).

Double Moving Averaging Filter

$$MA_n^{(1)} = \frac{1}{N+1}\sum_{i=0}^{N} X_{n-i}$$

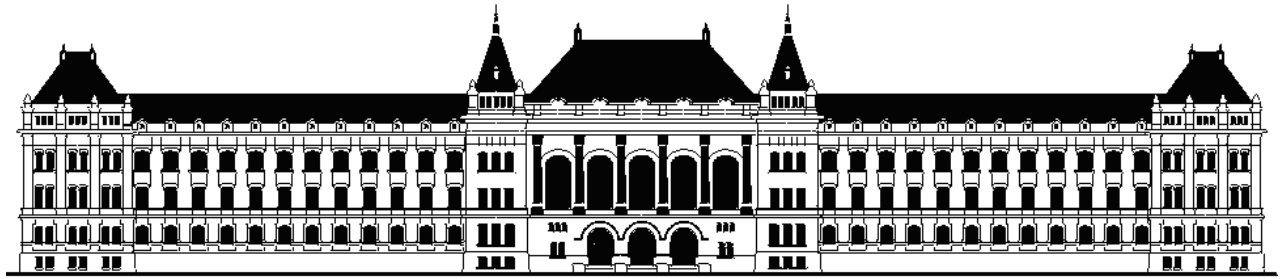
$$MA_n^{(2)} = \frac{1}{N+1}\sum_{i=0}^{N} MA_n^{(1)}$$

Állandó sebességű mozgások lekövetésére alkalmas (tőzsdében is alkalmazzák, ahol kb lineáris az árfolyam változása).

Lineáris inputra $MA^{(1)}$ kb egy fele akkora meredekségű egyenes mentén követi, $MA^{(2)}$ pedig negyedakkora. Így:

$$\hat{X}_n = 2MA^{(1)} \text{-} MA^{(2)}$$

Köszönöm a figyelmet!



M Ű E G Y E T E M 1 7 8 2