# Issues with high dimensional data:（expensive）

# • In very high-dimension, almost every point lie at the edge of the space, far away from the centre

**•** Curse of Dimensionality: When dimensionality increases, data becomes increasingly sparse in space

# How deal with large dataset? distributed framework

**MapReduce: Distributed Computing for Data Mining**

Google need 4 months to read web page (single machine) **Solutions**: Cluster of commodity Linux nodes, Commodity network (ethernet) to connect them; Cluster Computing **Challenges** : Large-scale clusters for data mining problems on commodity hardware

**Map-reduce addresses many problems**

- Store data persistently on multiple nodes for persistence and availability (**for Machines fail)**

- Move computation close to data to minimize data movement **(for network bottleneck)**

- **Simple programming model** to hide the complexity of all this magic

Map-Reduce: - Redudant Storage Infrastructure – File system• Google: GFS. Hadoop: HDFS -Programming model • Map-Reduce

**Redundant Storage Infrastructure**

Problem: If nodes fail, how to store data persistently? Answer: Distributed File System:• Provides global filename space •GoogleGFS;HadoopHDFS;

Huge files (100s of GB to TB) , Data is rarely updated in place , Reads and appends are common

**Distributed File System**

Data kept in “chunks” spread across machines Each chunk **replicated** on different machines

• Seamless recovery from disk or machine failure

(Bring computation directly to the data! **Chunk servers** also serve as **compute servers**)

**Chunk servers:** File is split into contiguous chunks, Each chunk replicated, keep replicas in different racks

**Master node** Stores metadata about where files are stored. Might be replicated

**Client library** Talks to master to find chunk servers**,** Connects directly to chunk servers to access data

**Programming Model: MapReduce**

**Task: Word Count with Single Machine**

**Case 1**: File too large for memory, but all <word, count> pairs fit in memory **Case 2**:<word, count> pairs do not fit into memory Count occurrences of words: **• words doc.txt | sort | uniq -c**

**Word Count example**

**map(key, value):** // key: document name; value: text of the doc.For each word w in value: emit(w, 1)

**reduce(key, values):** // key: a word; values: an iterator over counts. result = 0; for each count v in values: result += v; emit(key, result)

**URL example:** <URL, size ...>, e.g., <www.facebook.com/xdfdfd/zzzz.html, 2000...> For each host, find the total number of bytes.

**map(key, value):** // key: document name; value: text of the document for each record r in the value:emit(hostname(r.URL), r.size)

**reduce(key, values):**//key: a host, values: an iterator counts over size. result =0; for each size v in values: result += v; emit(key, result)

**MAP-Reduce in parallel:** Use partitioning function (hash function) to send the same mapped output to the same reduce task machine

**Map-Reduce environment** takes care of:

1.**Partitioning** the input data 2.**Scheduling** the program’s execution across a set of machines 3.Performing the **group by key** step 4.Handling machine **failures** 5.Managing required inter-machine **communication**

**Coordination: Master**

**Task status**: (idle, in-progress, completed);

**Idle tasks** get scheduled as workers become available; When a map task completes, it sends the master the location and sizes of its R intermediate files, one for each reducer; Master pushes this info to reducers; Master pings workers periodically to detect failures

**Dealing with Failures**

**Map worker failure** Map tasks completed or in-progress at worker are reset to idle

Reduce workers are notified when task is rescheduled on another worker

**Reduce worker failure** Only in-progress tasks are reset to idle**,** Reduce task is restarted

**Master failure** MapReduce task is aborted and client is notified

**How many Map and Reduce jobs?**

M map tasks, R reduce tasks: Make M much larger than the number of nodes in the cluster, One DFS chunk per map is common, Improves dynamic load balancing and speeds up recovery from worker failures Usually R is smaller than M: Because output is spread across R files

**Refinements: Backup Tasks**

Problem:慢worker会延长完成时间:• Other jobs on the machine • Bad disks • Weird things

Solution: Near end of phase, spawn backup copies of tasks • 哪個快用哪個

**Refinement: Combiners (1)**

**Combiner is usually same as the reduce function**, and may also be different depending on the task

(組裡先合併了再傳給reducer,要傳的數據變少)

Combiner trick works only if the reduce function is commutative and associative

a+b = b+a; (a+b)+c=a+(b+c)

examples: Sum(Y), Average(Y), Median(N)

**Refinement: Partition Function**

Inputs to map tasks are created by contiguous splits of input file ; Reduce needs to ensure that records with the **same intermediate key** end up at the **same worker**

Default partition function: **hash(key) mod R**

Sometimes useful to 自定義the hash function:

**When to use MapRedue?**

- Problems that require sequential data access

- Large batch jobs (not interactive, real-time)

MapReduce is **inefficient** for problems where random (or irregular) access to data required:-Graphs,

-Interdependent data: Comparisons of many pairs

***Finding Similar Items: Locality-Sensitive***

**Jaccard similarity:**

𝒔𝒊𝒎(𝑪𝟏, 𝑪𝟐) = |𝑪𝟏𝑪𝟐|/|𝑪𝟏𝑪𝟐|

**Jaccard distance:**

𝒅(𝑪𝟏, 𝑪𝟐) = 𝟏 − |𝑪𝟏𝑪𝟐|/|𝑪𝟏𝑪𝟐|

**Task: Finding Similar Documents**

**Step 1: Shingling: Convert documents to sets**

Simple approaches: Document = set of words appearing in document Document = set of “important” words 🡺 **bad** because **no context**

**Shingles (good):** A k-shingle (or k-gram) for a document is a sequence of k tokens that appears in the doc: Tokens can be characters, words etc.

Example: 𝑘 = 2, document 𝑫𝟏 = 𝑎𝑏𝑐𝑎𝑏. Then the **set** of 2-shingles: 𝑺(𝑫𝟏) = {𝑎𝑏, 𝑏𝑐, 𝑐𝑎} (別重複)

𝑘 = 3, 𝐷 = 𝑎𝑏𝑐𝑏𝑐𝑎𝑐𝑏𝑑, what is the set of 𝑘-shingles? 𝑺(𝑫) = {abc, bcb, …}

**Compressing Shingles:** hashing to 4 bytes

Represent doc **by hash values** of its k-shingles

Idea: shingles diff, hash values likely diff

Example: 𝒌 = 𝟐, document 𝑫𝟏 = 𝑎𝑏𝑐𝑎𝑏

Set of 2-shingles: 𝑺(𝑫𝟏) = {𝑎𝑏, 𝑏𝑐, 𝑐𝑎}

Hash the singles: 𝒉(𝑫𝟏) = {1, 5, 7}

**Similarity Metric for Shingles:** **Jaccard similarity**

Each unique shingle is a **dimension**. Vectors are **very sparse**.

**Working assumption:** Documents have many common shingles **in common**, even if the text appears **in different order**

**Must pick k large enough,** or most documents will have most shingles: k = 5 is OK for short documents; k = 10 is better for long documents

**Encoding Sets as Bit Vectors**

Example: C1 = 10111; C2 = 10011 (both 1=>true)

Size of intersection = 3; size of union = 4,

Jaccard similarity (not distance) = 3/4

Distance: d(C1,C2) = 1 – (Jaccard similarity) = 1/4

**From sets to Boolean matrices**

Rows = elements (shingles) Columns = sets

**Column similarity** is the Jaccard similarity of the

**Finding similar columns (docs)**

**Naïve approach:** Comparing **all pairs** may take too much time:• That is why we need the third step (LSH)

**Step 2: Min Hashing**

**Hashing Columns (Signatures)**

Key idea: “hash” each column C to a small signature 𝒉(𝑪), such that: 1. ℎ(𝐶) is small enough that the signature fits in main memory 2. 𝑠𝑖𝑚(𝐶1, 𝐶2) is the same as the “similarity” of signatures ℎ(𝐶1) and ℎ(𝐶2)

Goal: Find a hash function ℎ(·) such that:

If 𝒔𝒊𝒎(𝑪𝟏, 𝑪𝟐) is high, then with high prob. 𝒉(𝑪𝟏) = 𝒉(𝑪𝟐) If 𝒔𝒊𝒎(𝑪𝟏, 𝑪𝟐) is low, then with high prob. 𝒉(𝑪𝟏) ≠ 𝒉(𝑪𝟐) sig: D1 D2 D3 D4(一个个columns)

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

自動產生的描述compressed!

**Problems of min-hashing**

-Permuting rows even once is prohibitive -**Row hashing**! Pick 𝑲 = 𝟏𝟎𝟎 hash functions 𝒌𝒊

Ordering under 𝒌𝒊 gives a random row permutation!

- **One-pass implementation**

For each column 𝑪 and hash-func. 𝒌𝒊 keep a “slot” for the min-hash value Initialize all 𝒔𝒊𝒈(𝑪)[𝒊] = inf

Scan rows looking for 1s• Suppose row 𝒋 has 1 in column 𝑪• Then for each 𝒌𝒊 :

– If 𝒌𝒊(𝒋) < 𝒔𝒊𝒈(𝑪)[𝒊], then 𝒔𝒊𝒈(𝑪)[𝒊] = 𝒌𝒊(𝒋)

**Step 3: Locality-Sensitive Hashing (處理signatures)**

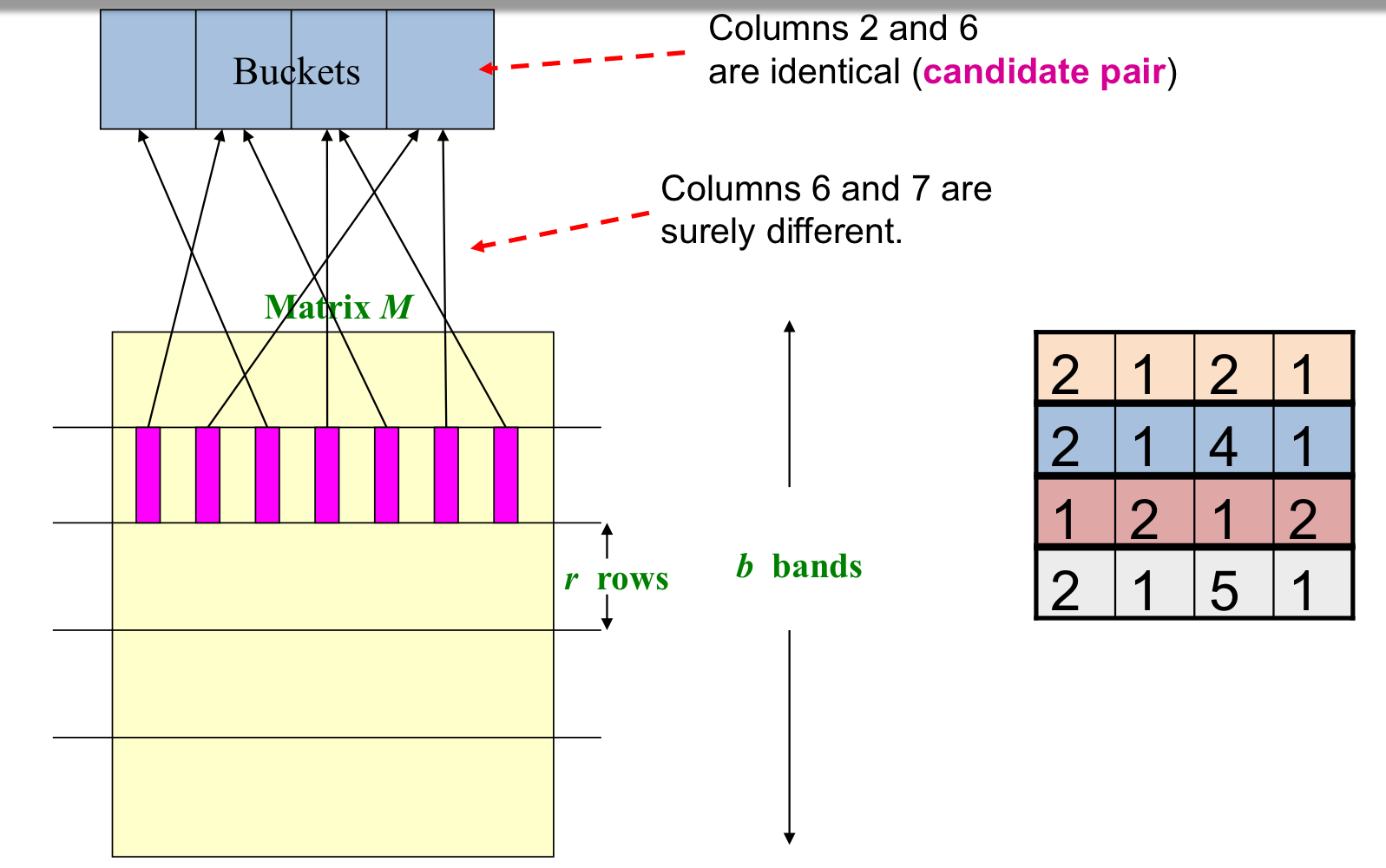
**Hash** columns of signature matrix M **several times**

Divide the signature matrix 𝑴 into 𝒃 bands of 𝒓 rows

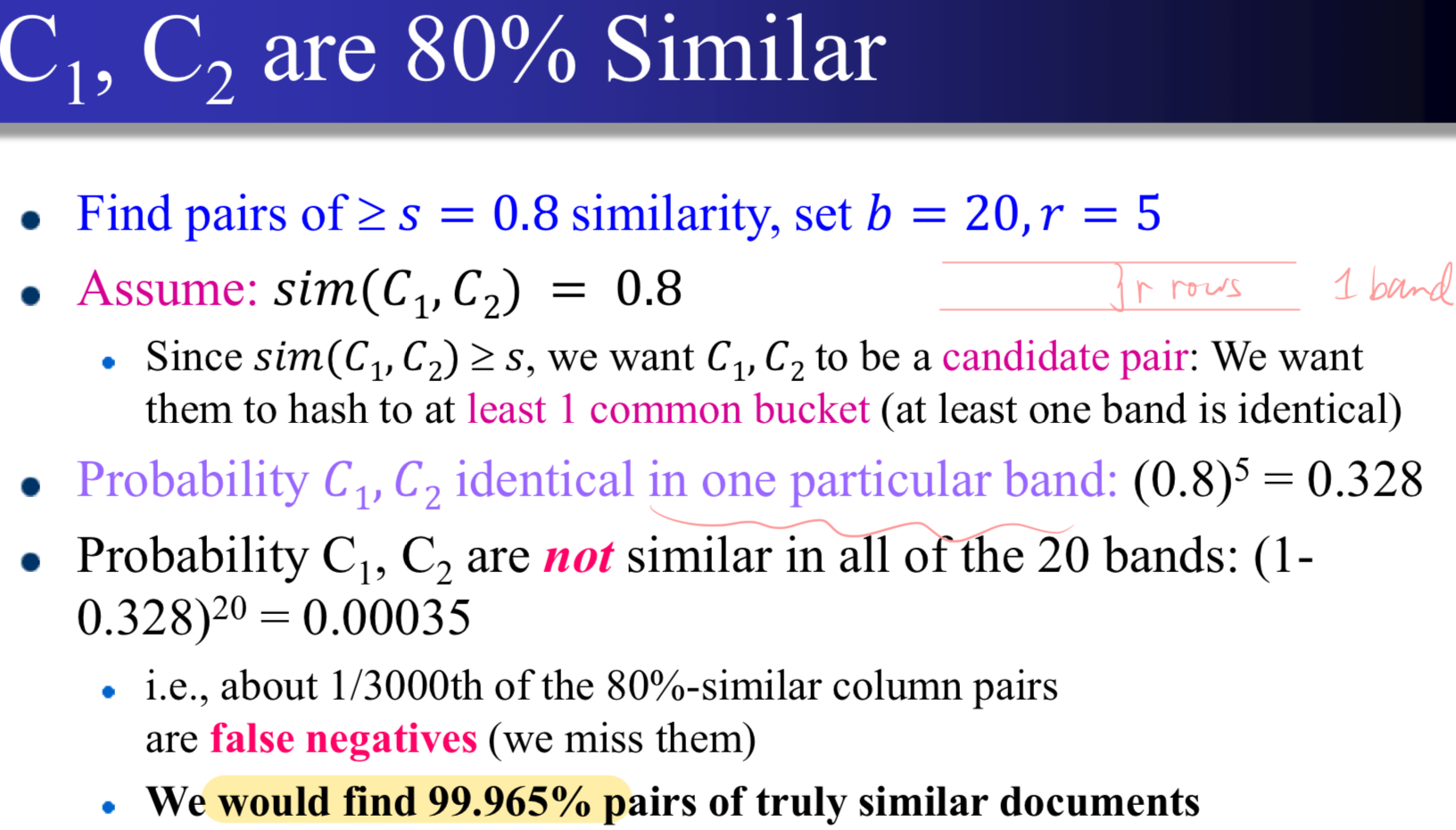
• Need to tune the parameter b and r to catch most similar pairs, but few non-similar pairs.

For each band, hash the columns with the same signature to the same slot with high probability.

• For simplicity, we assume that only identical signatures are hashed to the same slot.



C1,C2 are 80% similar



C1,C2 are 30% similar (1-(1-0.3^5)^20 (small no. of false positive)

**Example:** Given 𝑏 = 25 bands and 𝑟 = 4 rows per band, derive: probability that a record pair with similarity 0.9 be added to the candidate pair; 1-(1-0.9^4)^25

**LSH trade-off**

If we had only 15 bands of 5 rows, the number of

false positives would go down, but the number of false negatives would go up

**b bands, r rows per band**

Columns C1 and C2 have similarity 𝒕; Pick any band (𝒓 rows) Prob. that all rows in band equal = 𝒕𝒓 Prob. that some row in band unequal = 𝟏 – 𝒕^r Prob. that no band identical = (𝟏 – 𝒕^r)^𝒃 Prob. that at least 1 band identical =𝟏 − (𝟏 – 𝒕^𝒓)^𝒃

**LSH summary**

-Tune 𝑴, 𝒃, 𝒓 to get almost all pairs with similar

signatures, but eliminate most pairs that do not have similar signatures -Check in main memory that candidate pairs really do have similar signatures

-Optional: In another pass through data, check that the remaining candidate pairs really represent similar documents

**Locality-Sensitive Hashing**

**Locality-Sensitive (LS) Families**

A family 𝑯 of hash functions is said to be (𝑑1, 𝑑2, 𝑝1, 𝑝2)-sensitive (𝑑1 < 𝑑2) if for any 𝑥 and 𝑦 in 𝑆:

If 𝒅 𝒙, 𝒚 ≤ 𝒅𝟏, then the probability over all ℎ in 𝑯, that ℎ(𝒙) = ℎ(𝒚) is at least 𝑝1.

If 𝑑 𝒙, 𝒚 ≥ 𝑑2, then the probability over all ℎ in 𝑯, that ℎ(𝒙) = ℎ(𝒚) is at most 𝑝2.

**Amplifying an LSH-Family**

AND construction like “rows in a band.”

OR construction like “many bands.”

**AND operation** If 𝑯 is (𝑑1, 𝑑2, 𝑝1, 𝑝2)-sensitive, then 𝑯’ is (𝑑1, 𝑑2, (𝑝1)^𝑟, (𝑝2)^𝑟)-sensitive. Proof: Use fact that all ℎ𝑖 are independent. (𝑝2)^𝑟 lowers prob for large distance(good) ; (𝑝1)^𝑟 also lowers prob for small distance (bad)

**OR Operation** If 𝑯 is (𝑑1, 𝑑2, 𝑝1, 𝑝2)-sensitive, then 𝑯’ is (𝑑1, 𝑑2,1 −(1 − 𝑝1)^𝑏 , 1 − (1 − 𝑝2)^𝑏)-sensitive. 1 −(1 − 𝑝1)^𝑏 raises prob for small distance (good); 1 −(1 – 𝑝2)^𝑏 also raise prob for large distance(bad)

**Effect of AND and OR construction**

**AND** makes all probs. shrink, but by choosing 𝑟 correctly, we can make the lower prob. approach 0 while the higher does not; **OR** makes all probs. grow, but by choosing 𝑏 correctly, we can make the higher prob. approach 1 while the lower does not

Practice: Given a (0.2,0.8,0.8,0.2)-sensitive family 𝐻, show what local sensitive hash family we drive with: 1. 10 AND then 20 OR:

(0.2, 0.8, 1-(1-0.8^10)^20, 1-(1-0.2^10)^20)

2. 5 OR then 5 AND:

(0.2, 0.8, (1-(1-0.8)^5)^5, (1-(1-0.2)^5)^5

Hint: answer in (0.2,0.8, 𝑝𝑥, 𝑝𝑦)-sensitive family

**An LSH family for cosine distance**

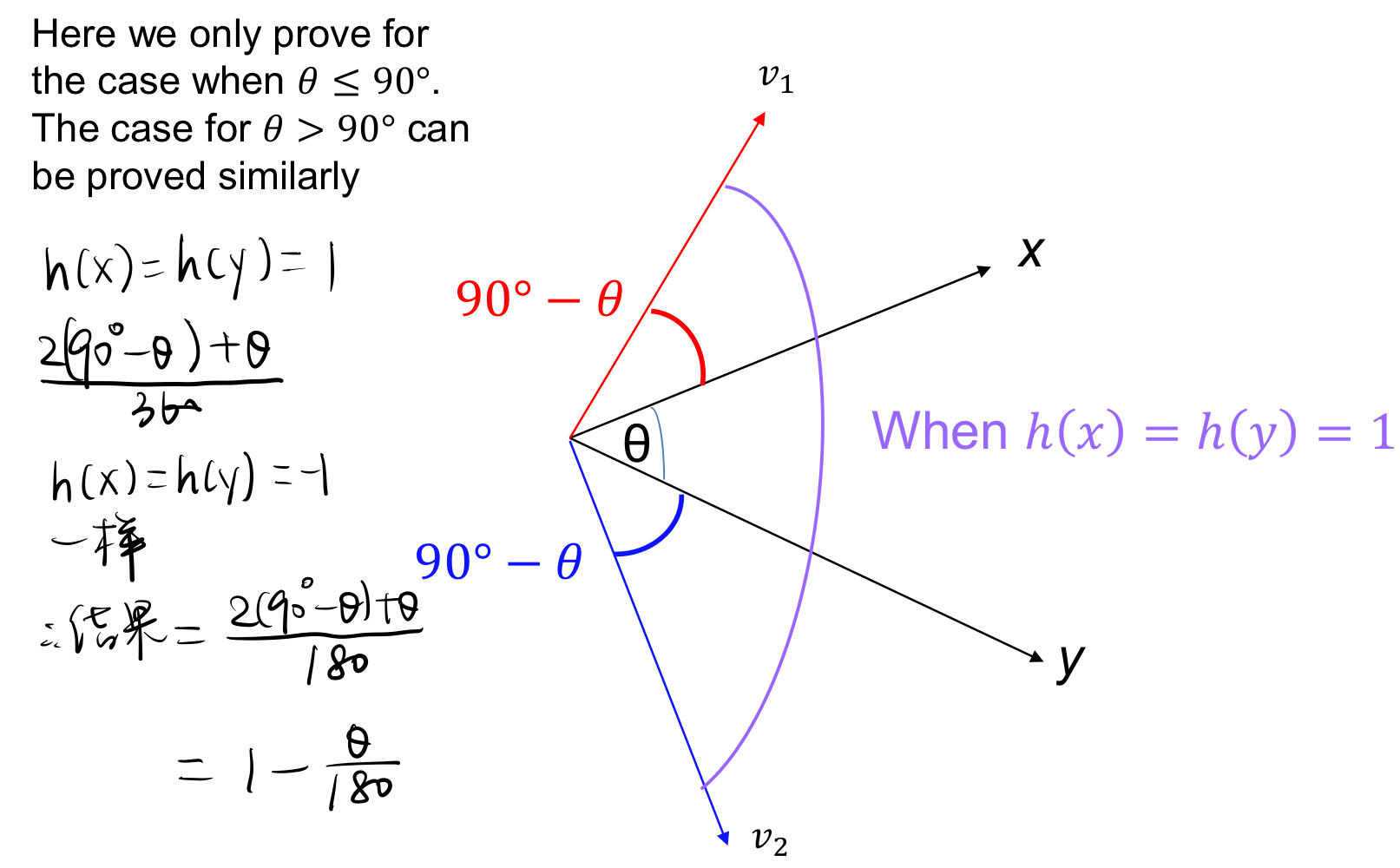
目的是把相似的向量映射到同一个BUCKET

**Random Hyperplanes**

Cosine distance of two vectors: The angles (≤ 180°)

选个向量v,相当于一个Hash function. ℎ𝑣(𝑥) = +1 if 𝑣 ⋅ 𝑥 > 0; ℎ𝑣(𝑥) = −1 if 𝑣 ⋅ 𝑥 < 0.

LSH-family 𝑯 = set of all functions derived from any vector 𝑣. Claim: 𝑃𝑟𝑜𝑏[ℎ(𝒙) = 𝒉(𝒚)] = 1 – (angle between 𝑥 and 𝑦 divided by 180). (x,y 夾角越小,哈希值一樣的概率越大) Then, we can prove that 𝑯 is a (𝑑1, 𝑑2, (1 − 𝑑1/180), (1 − 𝑑2/180))-sensitive family for cosine distances. Thus, we can apply the AND and OR construction to **amplify** the probability as we like.

****

***Streams***

**Sampling from Data Streams**

**1.Sample a fixed proportion**

**Naïve solution:**Generate a random integer in [0...9] for each query; Store the query if the integer is 0, otherwise discard; Then, return the number of unique queries on the sampled 1/10 queries

**Issues:** Assume that we have the following stream 𝑞1, 𝑞1, 𝑞2, 𝑞2, ⋯, 𝑞𝑖, 𝑞𝑖, … . , 𝑞𝑛, 𝑞𝑛, 2n queries where each query get repeated exactly twice. The number of unique queries: 0. The expected number of unique queries if we do random sample: 0.18n. So **biased.**

**Generalized (better) solution:** To get a sample of **𝑎/𝑏** fraction of the stream: Hash each tuple’s key uniformly into 𝑏 buckets. Pick the tuple if its hash value is **at most 𝑎 – 1**. **𝑥** = number of unique queries in the sample stream. The number of

estimated unique queries is: 𝑥 ⋅𝑏 / 𝑎

**Example:** Given stream 1, 1, 2, 3, 4, 5, 5, 6, 7, 8, sample 3/8 of the records

Hash function: ℎ(𝑥) = (3𝑥 + 5) 𝑚𝑜𝑑 8

• We only keep the keys whose hash values are ≤ 2 : ℎ (1) = 0-> keep, ℎ (2) = 3-> skip ℎ(3) = 6-> skip, ℎ(4) = 1-> keep, ℎ(5) = 4, skip, ℎ (6 )= 7 skip ℎ(7) = 2 keep, ℎ(8) = 5 skip. The sampled records: 1, 1, 4, 7. The estimated number of unique queries is: 2⋅8/3≈ 5.3

**2. A Random Sample of Fixed Size**

**Solution:** Reservoir Sampling

Store all the first 𝑠 elements of the stream to 𝑆

Suppose we have seen 𝑛 − 1 elements, and now the 𝑛-th element arrives (𝒏 > 𝒔)

• Randomly sample an integer 𝑥 from **[1, 𝑛]**. If 𝑥 ≤ 𝑠, keep the 𝑛-th element, else discard it (This holds with 𝑥/𝑛 probability) • If 𝑥 ≤ 𝑠, then it replaces the **𝑥-th** element in the sample 𝑆

一張含有 文字, 字型, 螢幕擷取畫面, 白色 的圖片

自動產生的描述

**Querying Over a Sliding Window**

**Binary representation: e.g.** n=12, 12%2=0, (\*\*\*0), pass 12/2=6 to the next step; 6%2=0, (\*\*00), pass 6/2=3; 3%2=1, (\*100), pass 3/2=1; 1%2=1, (1100), now 1/2=0 stop (等於0才停)

**Problem: Given a stream of 0s and 1s, How many 1s are in the last 𝑘 elements? (any 𝑘 ≤ 𝑁)**

**Obvious solution:** Store the most recent 𝑁 elements. When a new element comes in, discard the (𝑁 + 1)-th

element (can’t store all elements, so can only approx.)

**DGIM Method:** Datar, Gionis, Indyk, and Motwani.

Store only 𝑂(log^2𝑁) space per stream. (𝑂(log^2𝑁) bits). BN = window size.

**Gives approximate answer**, never off by more than 50%. Error factor can be reduced to any fraction > 0, with more complicated algorithm and proportionally more stored bits

Error: If we have 10 1s then 50% error means 10 +/− 5

**Bucket:** A bucket is a segment of the window ending with a 1; it is represented by a record consisting of: The timestamp of its end • Take 𝑂(log 𝑁) bits; The number of 1’s between its beginning and end.• Number of 1’s = size of the bucket. Constraint on bucket sizes: number of 1’s must be a power of 2. only 𝑂(log log 𝑁) bits are required for this count.

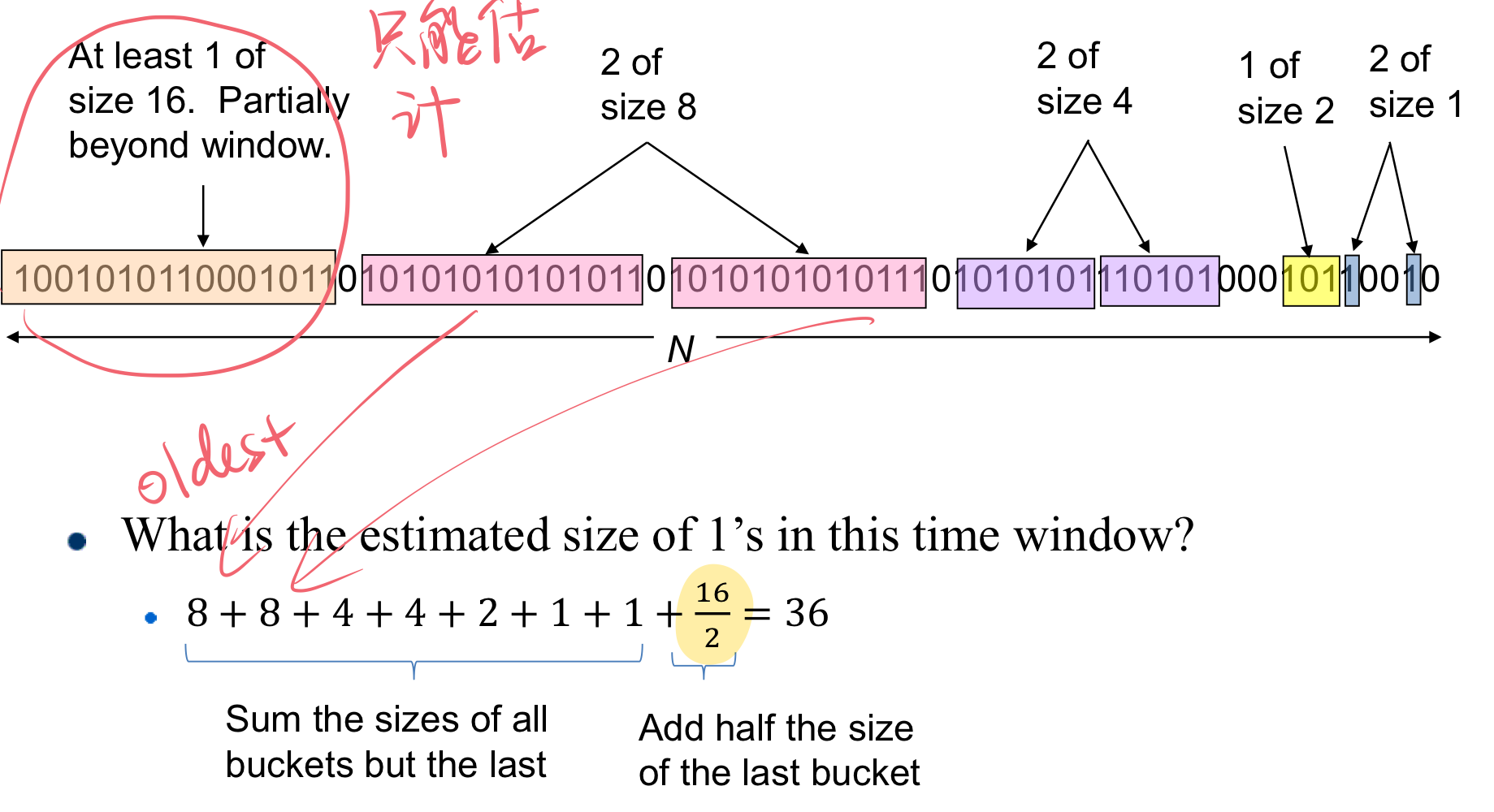
**Three properties of buckets that are maintained:**

1.Either one or two buckets with the same power-of-2 number of 1s 2.Buckets do not overlap in timestamps

3.Buckets are sorted by size

- 新data來的時候記得減掉一個舊的

**Estimate # of 1s in the most recent N elements**



**Error Bound**

Why is error at most 50%? Suppose the last bucket has size 2^𝑟. Then by assuming 2^(𝑟−1) (i.e., half) of its 1s are within the last bucket (but out of last-N bits), an error of at most 2^(𝑟−1).Since there is at least one bucket of each of the sizes less than 2^𝑟 , the true sum is at least 1 + 2 + 4 + . . + 2^(𝑟−1) = 2^𝑟− 1

Thus, the error is at most 50%

# Extenstion1. How many 1’s in the last 𝑘 elements? where 𝑘 < 𝑁?

# Answer: Find oldest bucket 𝐵 that overlaps with 𝑘. Number of 1s is the sum of sizes of more recent buckets than 𝐵 + ½ size of 𝐵. N=70

# 

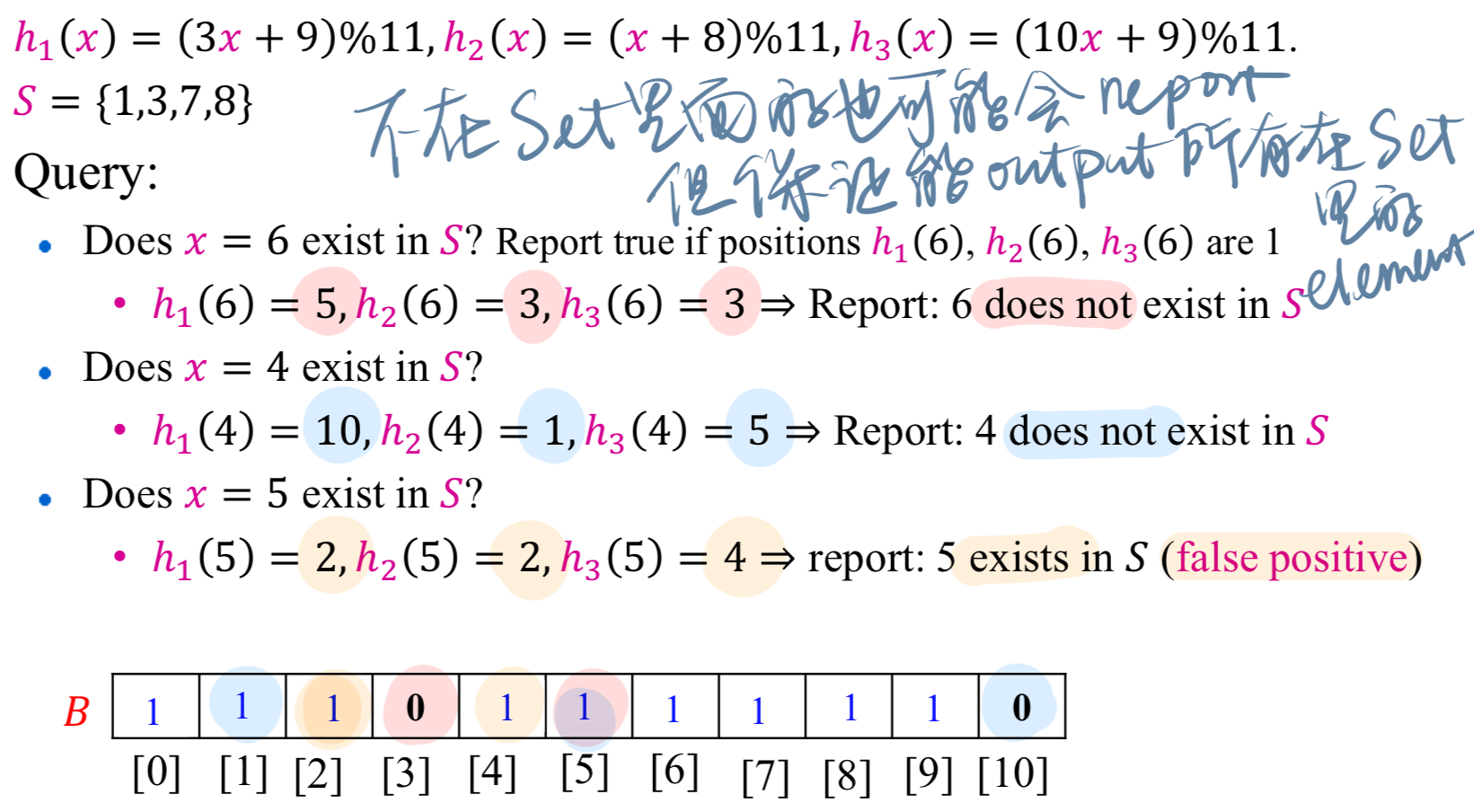
**Extenstion 2:** we want the sum of the last k elements

# 

**Filtering data stream**

# determine which tuples in the stream have key in 𝑆

**Bloom Filter (lower false positive rate, guarantee no FN and use limited memory)** : Initialization: Given a set of keys 𝑆 that we want to filter; Create a bit array 𝐵 of 𝑛 bits, initially all 0s; Choose 𝑘 independent hash function ℎ1, ℎ2,⋯, ℎ𝑘 with range [0, 𝑛):The hash function guarantees that each position in 𝐵 is hashed with equal probability, i.e., 1/𝑛; For each member of 𝑠∈𝑆, set 𝐵[ℎ𝑖(𝑠)] = 1 for each 1≤𝑖≤𝑘.



# Optimal value of k (# of hash function) = n/m\*ln(2); n=# of bits in bit array B, m= # of elements in S

# False positive rate = (1-e^(-km/n))^k

**Counting distinct elements**

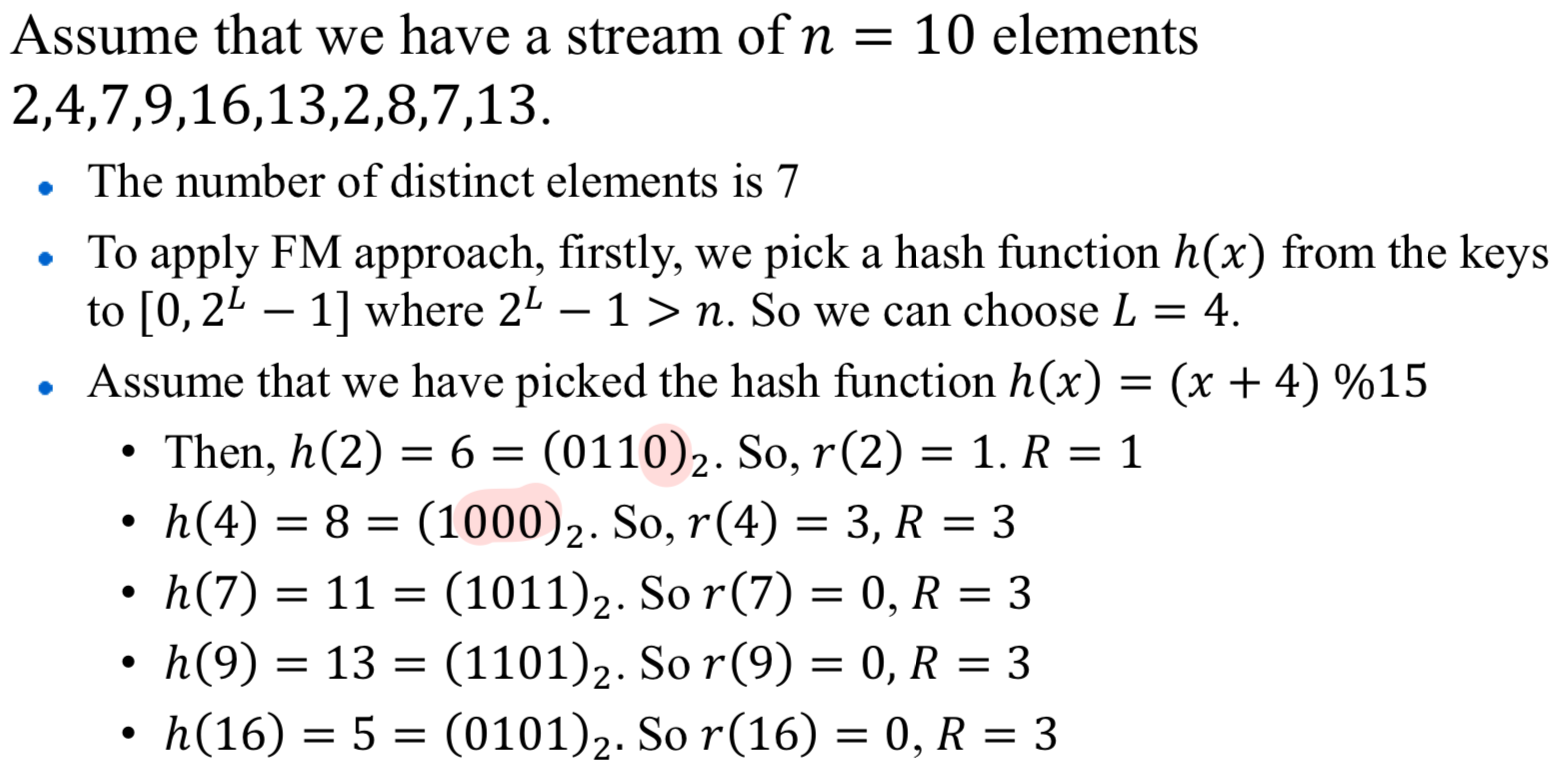
# Obvious approach: Maintain the set of elements seen so far (hash table) but no space to maintain the set

# of elements seen so far. We need to estimate in an unbiased way; accept small error, but limit the prob that the error is large

**Flajolet-Martin (FM) Approach:** Pick a hash function ℎ that maps each of the 𝑛 elements to integers in the range

[0, 2^𝐿− 1] with equal probability (2^𝐿− 1 > 𝑛).

For each stream element 𝑎, let 𝑟(𝑎) be the number of **trailing 0’s** in the binary representation of ℎ(𝑎), denoted as the tail length. 𝑟(𝑎) = number of zeros before we see the first 1 counting from the right for the binary representation of ℎ(𝑎) . Let 𝑅 be the maximum 𝑟(𝑎) seen so far, i.e., 𝑅 = maxa𝑟(𝑎), over all the items a seen so far. Estimated number of distinct elements = 2𝑅

=2^3

**Improving accuracy**

-use **many hash function**s hi and getting an estimation Ri for each hash function hi

-How to combine Ri? Average?: what if one very large value 2^Ri? Median? All estimates are a power of 2

**Solution:** Partition the estimation 2^𝑅𝑖 s into small groups;

Take the median of groups; Then take the average of the estimations in each group

More Queries : 1. 高阶矩估计：使用哈希函数将输入元素映射为整数值。 初始化足够大的位图（BITMAP）。对每个元素，计算其哈希值，找到其二进制表示中最低有效位（LSB）的位置。根据计算出的位位置更新位图。利用位图信息估计高阶矩。例如，通过不同位图的组合可以估算不同的矩。

2. 寻找频繁项：使用多个哈希函数将每个元素映射到多个（buckets）。为每个桶初始化计数器。对每个输入元素，使用哈希函数更新对应桶的计数器。检查计数器值，找到超过阈值的频繁项。

***Time Series***

**Defining the similarity between two time series**

Sol: Euclidean distance, Dynamic Time Warping (DTW)

**Preprocessing: why?** Euclidean distance is sensitive to some distortions in the data.(distortions not meaningful)

1.Offset translation(平移):減掉各自的mean; 2.Amplitude scaling(amplitude大小不一樣): ➗sd; 3. remove linear trend: find the best fitting line and subtract that line from time series; 4.Noise : take the average values with its neighbour

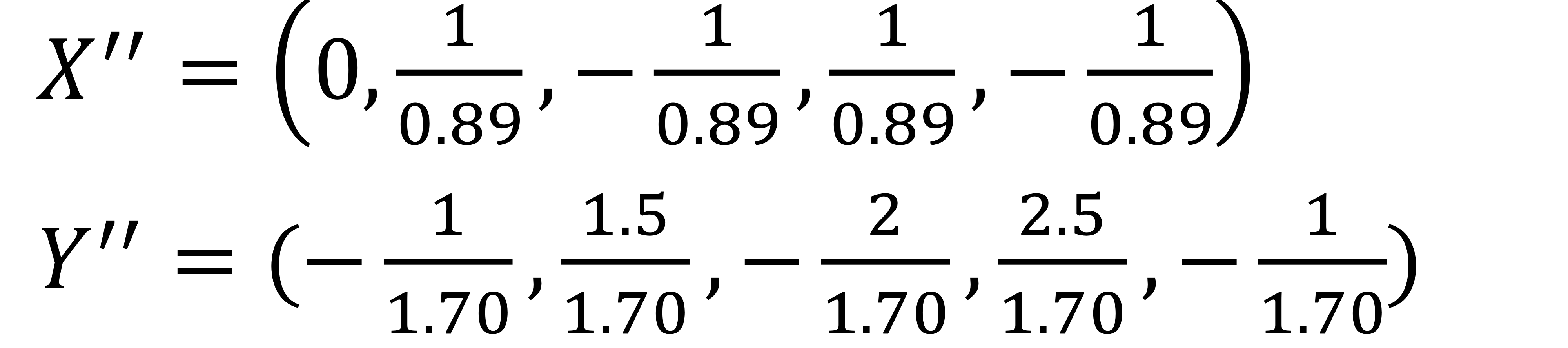
**Example:** 𝑋 = (0,2,1,4,3),𝑌 = 5,9,7,13,11, Assume that the best fitting straight line for 𝑋, 𝑌 are 𝑓 𝑋(𝑡) = 𝑡 − 0.5,

𝑓 𝑌(𝑡) = 1.5𝑡 + 4.Then, **after removing linear trend**, the new time series are:𝑋′= (0 − 𝑓𝑋(1) , 2 – 𝑓𝑋(2) , 1 − 𝑓𝑋(3) , 4 – 𝑓𝑋(4) , 3 – 𝑓𝑋(5)) = (−0.5, 0.5,−1.5, 0.5,−1.5)

**Example:** 𝑋′= (−0.5, 0.5,−1.5, 0.5,−1.5)mean=−0.5. std=0.89

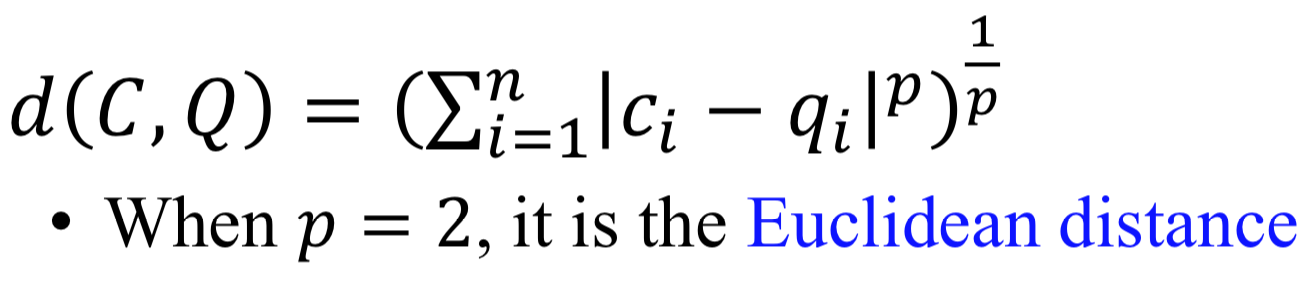
𝑌′= (−0.5,2,−1.5, 3,−0.5) mean=0.5, std=1.70

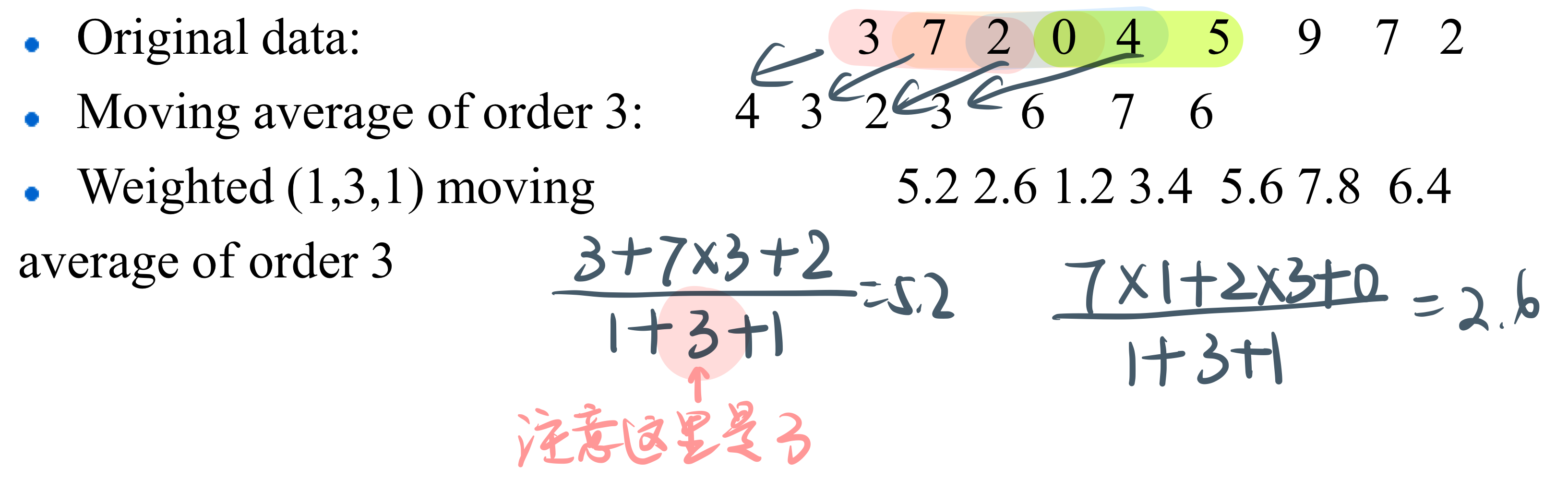
After **offset translation and amplitude scaling**:

 一張含有 字型, 行, 白色, 圖表 的圖片

自動產生的描述

Finally, we can calculate the distance between 𝑋′′ and 𝑌′′



**Example (smoothening):** ****

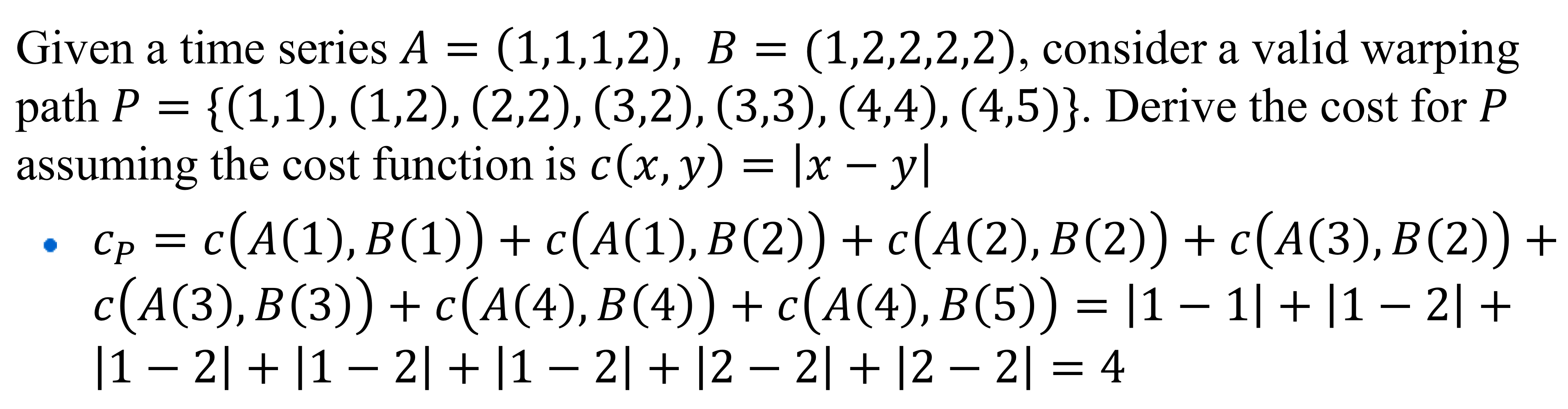
Btw, **anomoly detection** don’t do smoothening

**Dynamic Time Warpping**

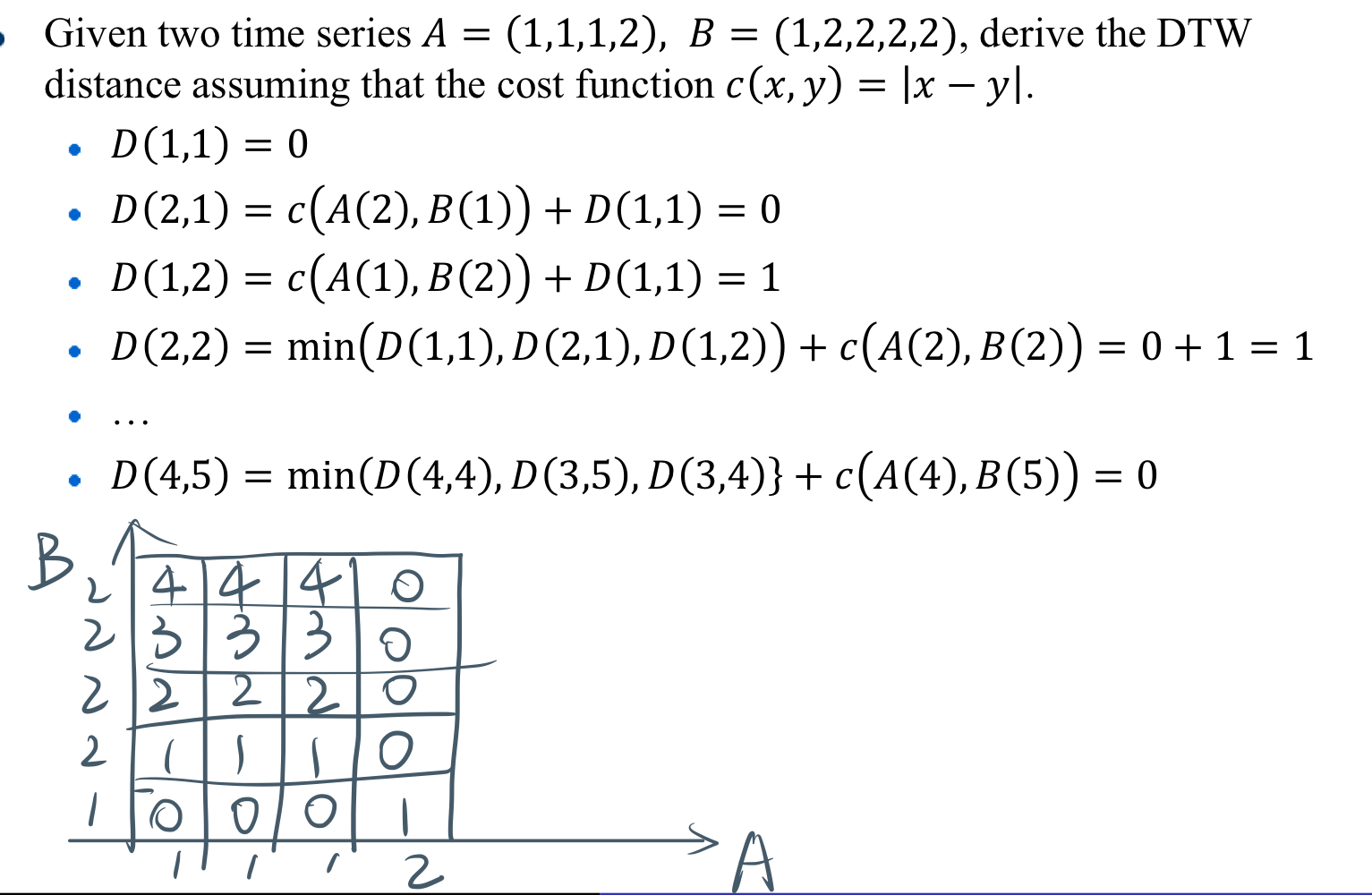
**Warpping function:** To find the best alignment between A and B one needs to find an (N,M)-warping path through the gridP = p1, … , ps , … , pk ps = (is , js ) which minimizes the total distance between them. P is called a warping function.

**Constraints: 1. Boundary condition:** the wrapping path starts at the bottom left and ends at the top right (Guarantees that the alignment does not consider partially one of the sequences.) 2. **Monotonicity:** The warping path does not go back in “time” index. (Guarantees that features are not repeated in the alignment.) 3. **Continuity:** The warping path does not jump in “time” index. In the path, it either move one step right, up, or up-right (Guarantees that the alignment does not omit important features.)

**Optimal Warping Path:** 𝑃∗ is the path that has the minimum cost. **DTW distance** between two time series 𝑋 and 𝑌 is the **cost** of the optimal warping path.



**How to find the optimal path? Dynamic Programming**

****

**Time complexity: 𝑂(𝑛 ⋅ 𝑚)**

**How to speedup?**

**Approach 1:** Restrict more on the warping path. For example, for P = p1, … , ps , … , pk , ps = (is , js )

• 𝑖𝑠− 𝑗𝑠 ≤ 𝛿. Then, we can bound the time complexity to 𝑂(𝛿 ⋅ 𝑛) or 𝑂(𝛿𝑚)

**Approach 2:** Approximate the time series with some compressed representation, and do DTW on the compressed representation

**Dynamic Time Warping is very very slow to calculate!**

**Much slower than Euclidean distances**

**Usually used for short time series**

**DWT with Harr Wavelet (会有平地) 一張含有 文字, 字型, 行, 白色 的圖片

自動產生的描述**

**一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述**

The wavelet coefficient is (6,2,1, −1). With the wavelet coefficient, we can recover the original input data.

If we take only first two coefficients (6 2) and transform back, we get: (8 8 4 4). An estimation of the original time series. When take the first two, the remaining two are assumed to be **zero**

**Advantage of DFT:** Good ability to compress most natural signals, say with periodic patterns.

**Advantage of DWT:** DWT generally takes 𝑂(𝑛) running cost while DFT takes 𝑂 (𝑛 ⋅ log 𝑛)**.** Wavelet transforms give good results on sparse or skewed data and on data with ordered attributes**.** There is only one DFT, yet there are several families of DWTs.

***Content-based Recommendations***

Main idea: Recommend items to customer 𝒙 similar to previous items rated highly by 𝒙

**一張含有 文字, 螢幕擷取畫面, 行, 數字 的圖片

自動產生的描述**

**Step 1: calculate profile**

**Simple average**

**一張含有 文字, 字型, 白色, 行 的圖片

自動產生的描述**

**OR Normalized Weight**

Suppose star rating by user 𝑋 for M1,M4,and M6 are 5, 1, 3

Average rating of user 𝑋: 3



Weight of A2: 0 Weight of A3: (5−3 + 1−3)/2 = 0

Weight of A4: (1−3)/1 = −2 Weight of A5: (3−3)/1 = 0

Final user profile vector: (1,0,0, −2,0)

Step2: Making Predictions

一張含有 字型, 文字, 行, 圖形 的圖片

自動產生的描述

一張含有 文字, 字型, 螢幕擷取畫面, 白色 的圖片

自動產生的描述

Step3: Recommend movies with high prediction scores

**Pros: Content-based Approach**

+: No need for data on other users(No cold-start or sparsity problems) +: Able to recommend to users with unique tastes +: Able to recommend new & unpopular items (No first-rater problem) +: Able to provide explanations (Can provide explanations of recommended items by listing content- features that caused an item to be recommended)

**Cons:** –: Finding the appropriate features is hardE.g., images, movies, music–: Recommendations for new users (How to build a user profile?) –: Overspecialization(Never recommends items outside user’s content profile; People might have multiple interests; Unable to exploit quality judgments of other users

**Collaborative Filtering: Harnessing quality judgments of other users** (People often get the best recommendations from someone with similar tastes to themselves)

***User-User***

**Finding Similar Users (given 2 users)**

Jaccard similarity x (ignore the ratings); cosine x (Treats missing ratings as “negative”) Solution: subtract the (row) mean, (Pearson’s correlation (**Centered Cosine similarity**) )

First, subtract the (row) mean for non-empty entries

Then, calculate their cosine similarity (Captures intuition better• missing ratings treated as average, • handle “tough” raters and “easy raters” 好处)

Eg. Use top-2 rating

一張含有 文字, 字型, 螢幕擷取畫面, 收據 的圖片

自動產生的描述

**Item-Item**

Let k=2, estimate the rating of movie 1 by user 5

注意要用centered cosine similarity

一張含有 文字, 螢幕擷取畫面, 正方形, 數字 的圖片

自動產生的描述

In theory, Item-Item and User-User are dual approaches

In practice, it has been observed that item-item often **works better** than user-user

Why? Items are simpler than users; Items belong to a small set of “genre”, users may have multiple tastes Item similarity is more meaningful than user similarity

**Pros/Cons of CF**

+ Works for any kind of item :No feature selection needed (Biggest advantage)

- Cold Start: Need enough users in the system to find a match

- Sparsity: The user/ratings matrix is sparse; Hard to find users that have rated the same items

- First rater: Cannot recommend an item that has not been previously rated; New items, Esoteric items (小众)

- Popularity bias: Cannot recommend items to someone with unique taste Tends to recommend popular items

**Evaluation**

Compare predictions with known ratings: Root-mean-square error一張含有 字型, 文字, 行, 螢幕擷取畫面 的圖片

自動產生的描述一張含有 文字, 字型, 行, 白色 的圖片

自動產生的描述

局限性：平等地对待所有评分误差。在高评分预测上表现良好但在低评分预测上表现较差的方法可能会因为低评分预测的误差而受到惩罚。

**Complexity:** Expensive step is finding 𝒌 most similar customers/items: 𝑶(|𝑼|) where 𝑈 is the utility matrix

Too expensive to do at runtime: Could pre-compute

Naïve pre-computation takes time 𝑶(𝒏 · |𝑼|) where 𝑛 is the number of users /items

->Near-neighbor search in high dimensions (LSH)

Other solutions: Clustering, Dimensionality reduction

**Global baseline +CF**

**一張含有 字型, 文字, 行, 數字 的圖片

自動產生的描述** **一張含有 文字, 字型, 白色, 印刷術 的圖片

自動產生的描述** To further improve the accuracy, the solution has the following issues:

1) Similarity measures are “arbitrary” 2) Pairwise similarities neglect interdependencies among users 3) Taking a weighted average can be restricting Solution: Instead of sij use wij that we estimate directly from data

Example: Overall average rating 𝜇 = 3.17 Rating bias of user 5: 𝑏(𝑥=5) = 0.33 Rating bias of movies 1,3,6: 𝑏(𝑖=1) = 0.43, 𝑏(𝑖=3) = −0.17, 𝑏(𝑖=6) = −0.57 (same table as bef) ; b5,1 = 3.93 ; 𝑏5,3 = 3.33 ; 𝑏5,6 = 2.93

Improvement (by optimization)

**Use a weighted sum rather than weighted avg:**

**一張含有 字型, 筆跡, 文字, 行 的圖片

自動產生的描述Find wij that minimize SSE(差的平方的和) on the the training data**

**Objective function**

**一張含有 字型, 文字, 筆跡, 白色 的圖片

自動產生的描述**

**一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述**

***Dimension Reduction***

Assumption: Data on a 𝐷-dimensional space lies on or near a low 𝑑-dimensional subspace (𝑑 ≪ 𝐷): Axes of this subspace are effective representations of the data

**Why Dimension Reduction?**-Discover hidden correlations

-Curse of dimensionality: Distance becomes less meaningful. The running time of the algorithm depends exponentially on dimension -Remove redundant and noisy features: Not all words are useful -Interpretation and visualization -Easier storage and processing of the data

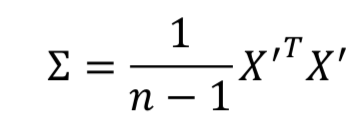
**Covariance Matrix**

For a set of 𝑛 **𝑑**-dimensional points 𝑿 = {𝒙1 , 𝒙2 ··, 𝒙n }, the covariance matrix A is a **𝑑 × 𝑑** matrix with 𝑨𝑖,𝑗 representing the covariance between the i -th and j -th coordinate sequences of all 𝑛 points.(一行和另一行算)一張含有 字型, 文字, 白色, 行 的圖片

自動產生的描述一張含有 字型, 筆跡, 文字, 書法 的圖片

自動產生的描述

**Principle Component Analysis (PCA)** 通过找到数据中的主要方向（主成分），将数据从高维空间映射到低维空间，同时尽可能保留数据的主要信息（方差）。Step 1: 全部减mean

Step 2: Compute the covariance matrix 𝑨 for the points after adjustment.  一張含有 字型, 文字, 白色, 圖形 的圖片

自動產生的描述 Step 3: Find the eigenvectors and eigenvalues of 𝑨 and sort the eigenvectors in descending order of their eigenvalues.

Step 4: Keep the first 𝑘 eigenvectors 𝒗1, 𝒗2, ⋯ , 𝒗𝑘, whose corresponding eigenvalues are the **top-𝑘** highest.

Step 5: For each point 𝒙, it is then converted to a 𝑘-dimensional point using these 𝑘 eigenvectors: (𝒗1 ⋅ 𝒙, 𝒗2 ⋅ 𝒙, ⋯ , 𝒗𝑘 ⋅ 𝒙) 原理：Projecting the 𝑛 points according to the direction of the 𝑖-th eigenvector, it keeps the 𝑖-th largest variance. Projection variance越大more variation information among different data points is kept, and better

SVD – Dimensionality Reduction

Goal: Minimize the sum of reconstruction errors or maximize ||𝑨 ⋅ 𝒗||(2 2)

How to choose 𝒗𝟏? Minimize reconstruction error

Q: How exactly is dim. reduction done? A: Set smallest singular values to zero SVD（奇异值分解）通过矩阵分解，提取数据中最重要的特征，同时去除冗余信息

**Best Low Rank approximation**

𝑨𝒌 is the best rank-k approximation of 𝐴 by keeping the first k columns of 𝑼, 𝚺, 𝐕 Frobenius Norm一張含有 字型, 圖形, 行, 文字 的圖片

自動產生的描述

What do we mean by “best”?Let 𝑩 be any rank-𝑘 𝑚 × 𝑛 matrix (with the same dimension as 𝑨).𝑨𝑘 is a solution to the minimization problem ||𝑨−𝑨𝑘|| 𝐹≤ ||𝑨−𝑩||𝐹一張含有 字型, 文字, 行, 白色 的圖片

自動產生的描述



一張含有 文字, 字型, 白色 的圖片

自動產生的描述

一張含有 文字, 字型, 白色, 行 的圖片

自動產生的描述

Complexity: 𝑶(𝒏𝒎𝟐) or 𝑶(𝒏𝟐𝒎) (whichever is less)

But:Less work, if we just want singular values Or if we want first 𝑘 singular vectors Or if the matrix is sparse

**How to query?** Q: Find users that like ‘Matrix’

A: Map query into a ‘concept space’ – how?

**𝒒𝒄𝒐𝒏𝒄𝒆𝒑𝒕 = 𝒒 𝑽**

一張含有 文字, 字型, 螢幕擷取畫面, 圖表 的圖片

自動產生的描述

+ Optimal low-rank approximation in terms of Frobenius norm

- Interpretability problem:A singular vector specifies a linear combination of all input columns or rows.

What are the meanings? Not easy to identify the meaning of each “concept” every time.

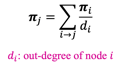
- Lack of sparsity: Singular vectors are dense!

***Graph Data Analysis***

**PageRank**

Idea: Page is more important if it has more links. Links from important pages count more

**The Flow model**

**** 一張含有 文字, 字型, 筆跡, 數字 的圖片

自動產生的描述 di: number of out-degree

No unique solution-> Additional constraint forces uniqueness: 𝝅𝒚 + 𝝅𝒂 + 𝝅𝒎 = 𝟏 (bad for large size)

**Matrix Formulation**

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述一張含有 圖表, 螢幕擷取畫面, 行 的圖片

自動產生的描述一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述一張含有 字型, 數字, Rectangle, 文字 的圖片

自動產生的描述 So the rank vector 𝝅 is an eigenvector of the stochastic web matrix 𝑴

**Power iteration**

Suppose there are 𝑁 web pages

Initialize: 𝝅(𝟎) = [1/𝑁, ... . , 1/𝑁]𝑇 Iterate:𝝅(𝑡+1) = 𝑴 ∙𝝅𝑡 Stop when|𝝅(𝑡+1)–𝝅𝑡 | < epsilon

**Why works?** Sequence𝑴⋅𝝅(𝟎),𝑴𝟐⋅𝝅(𝟎),...𝑴𝒌⋅𝝅(𝟎)

approaches the dominant eigenvector of 𝑴

**Random Walk interpretation (解释)**

Imagine a random web surfer: At any time 𝒕, surfer is on some page 𝒊. At time 𝒕 + 𝟏, the surfer follows an out-link from 𝒊 uniformly at random. Ends up on some page 𝒋 linked from 𝒊 Process repeats indefinitely

**Where is the surfer at time t+1?** Follows a link uniformly at random 𝒑( 𝒕 + 𝟏) = 𝑴 ⋅ 𝒑(𝒕)

Suppose the random walk reaches a state 𝒑 𝒕 + 𝟏 = 𝑴 ⋅ 𝒑(𝒕) = 𝒑(𝒕).then 𝒑(𝒕) is stationary distribution of a random walk

 Our original rank vector 𝝅 satisfies 𝝅 = 𝑴 ⋅ 𝝅

So, 𝝅 is a stationary distribution for the random walk

**Existence and Uniqueness**

For graphs that satisfy the following two conditions:

(i) The graph is strongly connected; this is it is

possible to go from any node to any other node,

(ii) There are no dead end; nodes that have no out-

going edges,

the stationary distribution is unique and eventually will be reached no matter what the initial probability distribution at time t = 0在满足特定条件的图中，平稳分布是唯一的，并且无论初始概率分布如何，最终都会达到这个平稳分布。

**PageRank: The Google Formulation**

原版pagerank的问题：2 problems:

(1) Some pages are dead ends (have no out-links)

Random walk has “nowhere” to go to; Such pages cause importance to “leak out” 重要性流入了死胡同，无法再流向其他地方 (0,0,0) (2) Spider traps: (all out-links are within the group) Random walk gets “stuck” in a trap; And eventually spider traps absorb all importance 封闭的子集内进行跳转，无法跳转到图中的其他部分(0,0,1)

**Solution: Teleports!**

中心思想：At each time step, the random surfer has two options.With prob. 𝛽, follow an out-link at random. With prob. 1 − 𝛽, jump to some random page. Common values for 𝛽 are in the range 0.8 to 0.9. Once chosen, it is always **fixed** in all random walks.

**Why are dead-ends and spider traps a problem and why do teleports solve the problem?** Spider-traps are not a problem, but with traps PageRank scores are not what we wantSolution: Never get stuck in a spider trap by teleporting out of it in a finite number of steps

Dead-ends are a problem. The matrix is not column stochastic so our initial assumptions are not

Met. Solution: Make matrix column stochastic by always teleporting when there is nowhere else to go

一張含有 字型, 行, 白色, 圖表 的圖片

自動產生的描述 This formulation assumes that 𝑴 has no dead ends. Can preprocess 𝑴 to remove all dead ends or explicitly follow random teleport links with probability 1.0 from dead-ends.

一張含有 文字, 字型, 白色 的圖片

自動產生的描述

一張含有 文字, 圖表, 行, 字型 的圖片

自動產生的描述

**Problems of PageRanks**: Measures generic popularity of a page.Every one will get the same ranking results of web pages for the same keyword • The higher chance we visit a node by a random walk, the larger the PageRank score it is and the more important the node it is.

Will ignore/miss topic-specific queries结果不一定相关以及不一定符合用户的兴趣

Solution: Personalized PageRank, also known as, Topic-Specific PageRank or Topic-Sensitive PageRank(next)

**Personalized PageRank**

If each element has equal weight. Then for each column 𝑗 of the teleport matrix𝑇s:𝑇i,j =1/|𝑆| for𝑖∈𝑆.

Thus, each column sums still to 1.

• Google matrix 𝑨=𝛽𝑴+ (1−𝛽) 𝑻s • 𝝅s = 𝑨𝝅s

Example: S={y,a}

一張含有 文字, 螢幕擷取畫面, 行, 圖表 的圖片

自動產生的描述

**Social Network Analysis**

A graph exhibits locality if eg. if 𝑦 is friends with 𝑥 and 𝑧, then there is a good chance 𝑥 and 𝑧 are friends.

Community: set of nodes with an unusually high density of edges.; Small world: social network has very small diameters在一个网络中，直径是指任意两个节点之间的最短路径中，最长的那条路径的长度。

**Girvan-Newman’s Algorithm for Community Detection**

1. Calculate the betweenness for all edges in the network.

2. Remove the edge with the highest betweenness.

3. Recalculate betweenness scores for all edges affected by the removal. 4. Repeat from step 2 until no edges remain.

**Calculate betweenness**

Step 1: Do a breadth-first search (BFS) from each node as the source in the graph.Derive a DAG 𝐺 for each source𝑠.

注意:we keep an edge (𝑢, 𝑤) during the traversal if the distance 𝑑(𝑠,𝑢) + 1 = 𝑑(𝑠,𝑤) 最短路径都保留

Step 2: Label nodes top-down to count the number of shortest paths from the source to that node.=in-neighbor 的最短路径之和。Count 𝑐(𝑠) of the source: 1.

最短路径数量的意义: 计算图中边或节点对最短路径的贡献Derive the fraction of shortest paths

Step 3: 从下到上开始算betweenness.一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述

一張含有 字型, 文字, 圖形, 白色 的圖片

自動產生的描述 weights of its out-going edges

**一張含有 圖表, 行, 圓形 的圖片

自動產生的描述**

Repeat steps 2-3 for the BFS traversal for **each node** as the source. Sum the scores for each edge and take half of the sum as the final betweenness score. Half to avoid double-counting each path.

把最大betweenness的边删了，然后重新算betweenness.随便删一个边如果有很多一样大的。

**How to Choose the # of Communities?**

Simple solution: Just stop when we have 𝑘 communities.

In real cases, we may not know what is a good option of k

Solution: Modularity 𝑄 -A measure of how well a network is partitioned into communities

一張含有 文字, 字型, 螢幕擷取畫面, 行 的圖片

自動產生的描述一張含有 文字, 螢幕擷取畫面, 字型, 行 的圖片

自動產生的描述

**Modularity: Number of clusters**

Modularity values take range [−1,1]

It is positive if the number of edges within groups exceeds the expected number.Q in [0.3,0.7] means **good** community structure. Modularity is useful for selecting the number of clusters

Girvan-Newman Algorithm adaptively removes the edges that may **go across** different communities (which has a high betweenness score)把可能跨群体的删了

The remaining nodes within a community tends to be closely connected

**How to stop?** Check modularity and stops when modularity no longer increases

Time complexity is high: 𝑂(𝑚 ⋅ 𝑛(𝑛 + 𝑚)) in the worst case