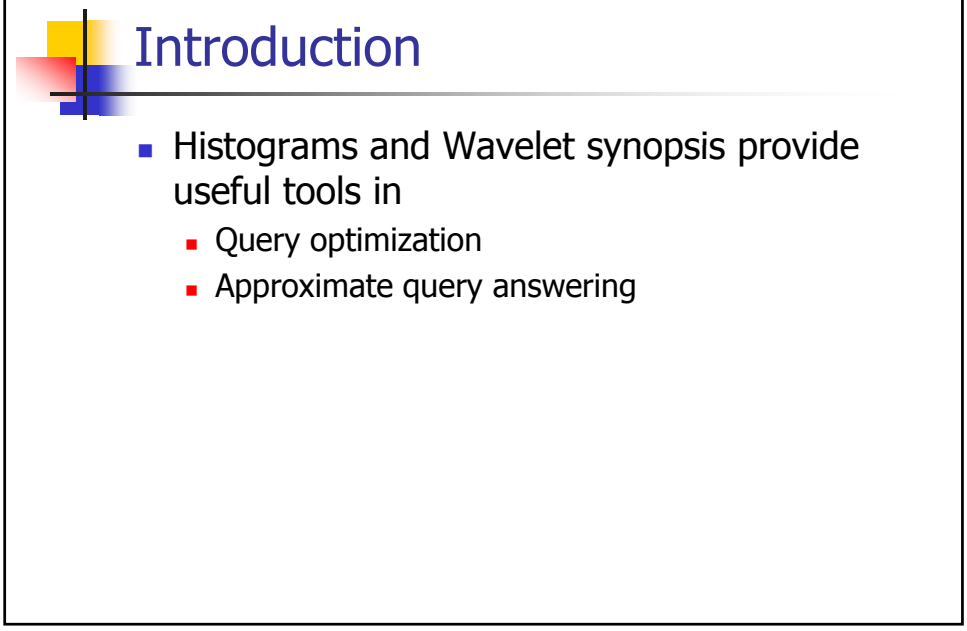
Histograms

Kyueseok Shim
Seoul National University
http://ee.snu.ac.kr/~shim

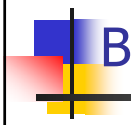


Introduction

- Histograms and Wavelet synopsis provide useful tools in
  - Query optimization
  - Approximate query answering

Overview

- Basics of Algorithms
- Histogram Construction
- Wavelet Synopses



Basics of Algorithms

[Jagadish, Koudas, Muthukrishnan, Poosala, Sevcik, Suel:VLDB'98]

[Guha, Koudas, Shim:STOC'01]

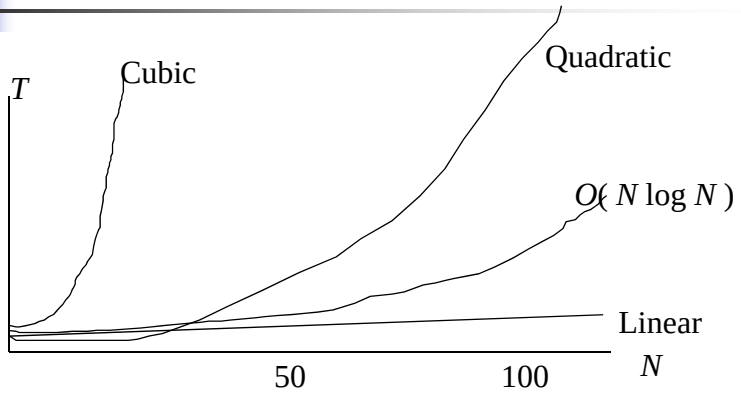
Algorithm Analysis

- Running time of an algorithm almost always depends on the amount of input: More input means more time. Thus the running time, *T*, is a function of the amount of input, *N*, or $T(N) = f(N)$.
- The exact value of the function depends on
  - the speed of the host machine;
  - the quality of the compiler and optimizer;
  - the quality of the program that implements the algorithm;
  - the basic fundamentals of the algorithm
- Typically, the last item is most important.

Worst-case Vs. Average Case

- Worst-case running time is a bound over all inputs of a certain size *N*. (Guarantee)
- Average-case running time is an average over all inputs of a certain size *N*. (Prediction): Difficult to define the distribution to compute the average cases
- Best case running time: Can be used to argue that the algorithm is really bad.

Running Time Functions



- Several common functions; for small inputs some are somewhat faster than others.

Definitions

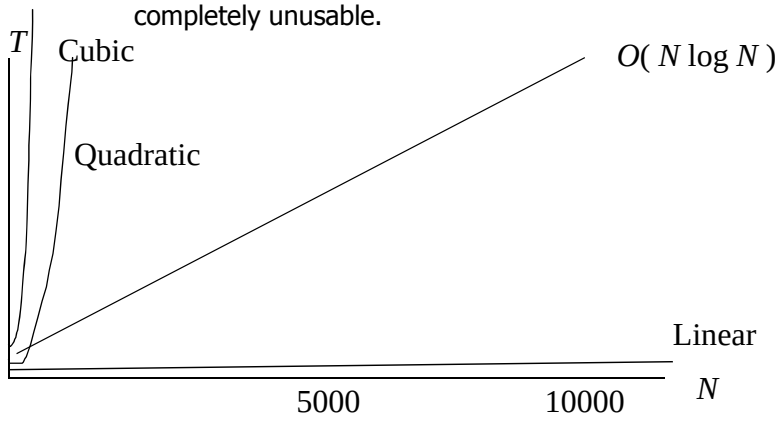
- Big-Oh for upper bound:
  - T(N) = O(f(N)) if there area positive constants c and n0 such that T(N) <= c f(N) when N >= n0.
- Big-Omega for lower bound:
  - T(N) = Omega(g(N)) if there are positive constants c and n0 such that T(N) >= c g(N) when N >= n0.
- Big-Theta:
  - T(N) = Theta(h(N)) iff T(N) = O(h(N)) and T(N) = Omega(h(N)).
- Small-Oh:
  - T(N) = o(p(N)) iff T(N) = O(p(N)) and T(N) is not Theta(p(N)).

Properties of Big-Oh

- If T1(N) = O(f(N)) and T2(N) = O(g(N)), then
  - T1(N) + T2(N) = max(O(f(N)), O(g(N))
    - Lower-order terms are ignored
  - T1(N)*T2(N) =O(f(N)*g(N))
- O(c f(N)) = O(f(N)) for some constant c
  - Constants are ignored!
- In reality, constants and lower-order terms may matter, especially when the input size is small.
- Can you prove the above properties with the definitions?

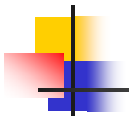
Running Time Functions

- For large inputs, some running time functions are completely unusable.



Types of Functions; Big-Oh

- Cubic: dominant term is some constant times N^3. We say $O(N^3)$.
- Quadratic: dominant term is some constant times N^2. We say $O(N^2)$.
- $O(N \log N)$: dominant term is some constant times $N \log N$.
- Linear: dominant term is some constant times N. We say $O(N)$.
- Example: $350N^2 + N + N^3$ is cubic.
- Big-Oh ignores leading constants.



Dominant Term Matters

- Suppose we estimate $350N^2 + N + N^3$ with N^3.
- For $N = 10000$:
  - Actual value is 1,003,500,010,000
  - Estimate is 1,000,000,000,000
  - Error in estimate is 0.35%, which is negligible.
- For large N, dominant term is usually indicative of algorithm's behavior.
- For small N, dominant term is not necessarily indicative of behavior, **BUT**, typically programs on small inputs run so fast we don't care anyway.

Running Time Calculation

1. Summations for Loops

```
for  i = 1 to n do {          for  i = 1 to n do {
        . . . .                       for j = 1 to n do {
        . . . .                             . . . . . .
     }                                }
```

If the loop of (a) takes $f(i)$ times, $T(n) = \sum_{i=1}^{n} f(i)$

If the loop of (b) takes $g(i, j)$ times, $T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} g(i, j)$

$\sum_{i=1}^{n} 1 = n$ **constant sum**

$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ **the linear sum**

$\sum_{i=1}^{n} c^i = \frac{c^{n+1} - 1}{c - 1}$, $c \neq 1$

Running Time Calculation

2. Sequential and If-Then-Else Blocks

```
for i = 1 to n do {
    A[i] = 0;
}
for i = 1 to n do {
    for  j = 1 to n do {
       A[i]++;
    }
}

if (cond)
   S1
else
   S2
```

$T(n) = O(n) + O(n^2) = O(n^2)$

$T(n) = max(T_{s1}(n), (T_{s2}(n))$

Actual Running Time

- For $N = 100$, actual time is 0.47 seconds on a particular computer.
- Can use this to estimate time for larger inputs:

  $T(N) = cN^3$

  $T(10N) = c(10N)^3 = 1000cN^3 = 1000\,T(N)$
- Inputs size increases by a factor of 10 means that running time increases by a factor of 1,000.
- For $N = 1000$, estimate an actual time of 470 seconds. (Actual was 449 seconds).
- For $N = 10{,}000$, estimate 449000 seconds (6 days).

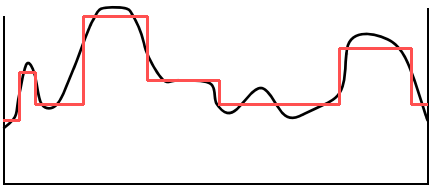
Histogram Construction Algorithms

[Jagadish, Koudas, Muthukrishnan, Poosala, Sevcik, Suel:VLDB'98]

[Guha, Koudas, Shim:STOC'01]

Histograms and Wavelets

- Assume that the input is aggregated i.e., we get <i,f(i)> and we want to approximate the fn f by B piecewise constant functions.



An Example of Histograms

Data Distribution

Location (i)	1	2	3	4	5	6	7
Value (Xi)	12	10	2	8	14	28	16

Optimal Histogram

Range	[1,4]	[5,5]	[6,6]	[7,7]
Representative	8	14	28	16

Sum of Squared Error Measure

Error for a range of [i,j]

$$ERROR[i,j] = \sum_{p=i}^{j}(x_p - \bar{x})^2 = \sum_{p=i}^{j} x_p^2 - \frac{1}{j-i+1}(\sum_{p=i}^{j} x_p)^2$$

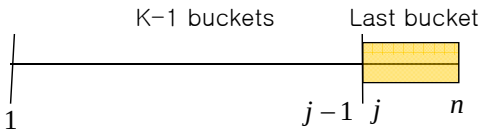
$$= (SQSUM[1,i] - SQSUM[1,i-1]) - \frac{1}{j-i+1}(SUM[1,i] - SUM[1,i-1])^2$$

$$SQSUM[1,i] = \sum_{p=1}^{i} x_p^2 \qquad SUM[1,i] = \sum_{p=1}^{i} x_p$$

Idea: V-Optimal Algorithm

- Within "step/bucket": Mean is the best.
- Assume that the last bucket is [j..n].

What can we say about the rest k-1 ?



Must also be optimal for $[1..j-1]$!

Dynamic Programming !!

The Optimal Algorithm

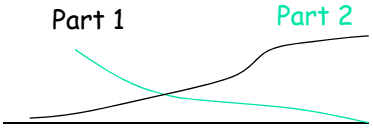
For i=1 to n do
  For K=1 to B do
    For j=1 to i-1 do (split point for the last bucket)

T[1…i, k] = Min[T[1…i, k],T[1…j,k-1]+ERROR[j+1,i]]

Give a O(Bn) space O(n^2B) time optimal algorithm.

The Approximate Algorithm Idea

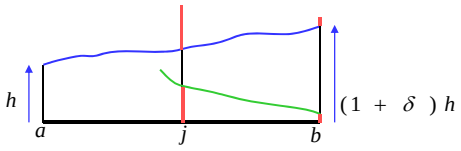
- Sum of two functions
  - Decreasing: the last bucket
  - Increasing: error from first k-1 buckets

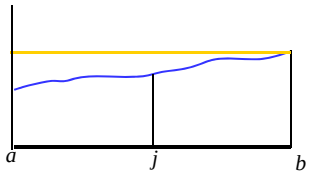


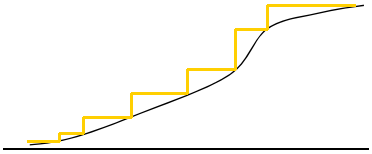
What if approximate the increasing function ?

(By a histogram !)

Approximating Increasing Fn

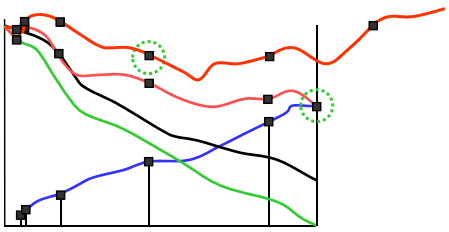


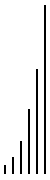




Algorithm by Picture

$k = 2$

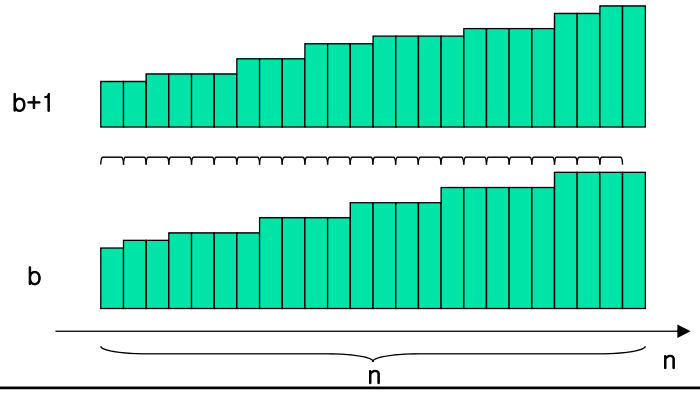




Works for piecewise small degree poly.

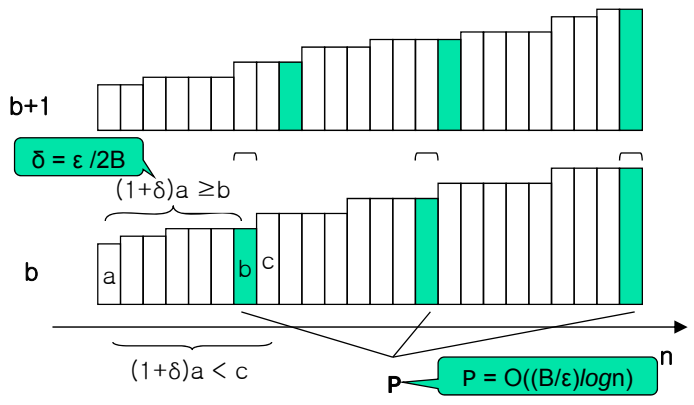
Optimal V-Optimal Construction: $O(Bn^2)$

- Jagadish, Kouda, Muthukrishnan, Poosala, Sevcik, Suel, VLDB 1998
- $OPT(i,k) = \min_{1 \le j < i}\{OPT(j,k-1)+VAR(j+1,i)\}$



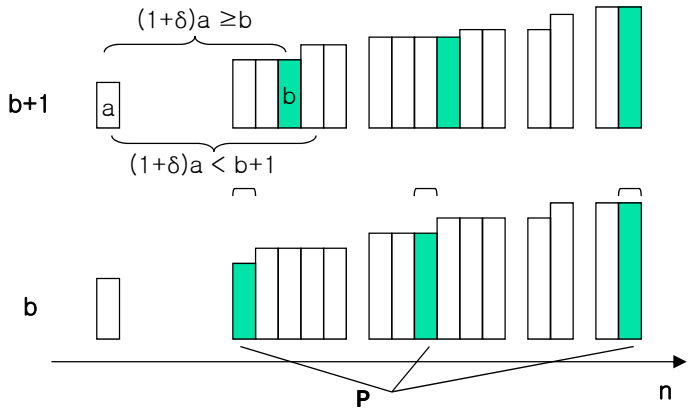
Approximate Histograms : O(BnP)

- Guha, Kouda, Shim, STOC 2001
- $SOL[p+1,b_j^p] = \min_{1 \le j < i}\{SOL[p,b_j^p]+VAR[b_j^p+1,n+1]\}$



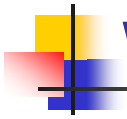
Enhanced Approximate Histogram - O(BP²*log*n)

- Guha, Kouda, ICDE 2002
- Binary Search



Can we approximate in less space

- [Guha, Koudas, Shim:STOC'01] show that indeed we can. The algorithm takes $O(B^2\varepsilon^{-1}\log n)$ space, $O(B^3 n\varepsilon^{-1}\log n)$ time with $1+\varepsilon$ times more error than optimal.
- Combined with results from [Guha, Koudas] Approximating a data stream for Querying and Estimation, ICDE, 2002, an extended version of the above shows that in space M (M>$O(B\varepsilon^{-1}\log n)1+\varepsilon$) we can get an approximation in time $O(n+\frac{n}{M}B^3\varepsilon^{-2}\log^2 n)$
- A smooth tradeoff.



Wavelet

- A useful mathematical tool for hierarchically decomposing functions
- Represent a function in terms of
  - A coarse overall shape
  - Details that range from broad to narrow
- Haar wavelet
  - The Haar basis is the simplest wavelet basis
  - Fastest to compute and easiest to implement

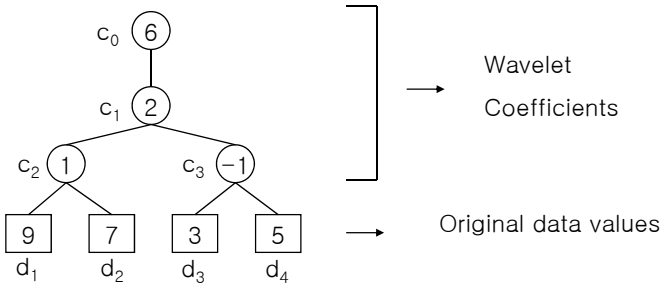
Haar wavelet

- Given a one dimensional data with a resolution 4, [9 7 3 5]
  - Recursive pairwise averaging and differencing at different resolutions

Resolution	Averages	Detail coefficients
4	[9 7 3 5]	
2	[8 4]	[1 -1]
1	[6]	[2]

  - The wavelet transform of the original data is given by [6 2 1 -1]

Error Tree



Path(u) : The set of all nodes in T that are proper ancestor of u with nonzero coefficients (definition in [GG02])

Reconstruction

- The reconstruction of any data value d_i using Error Tree

$$d_i = \sum_{c_j \in path(d_i)} \delta_{ij} \cdot c_j$$

  - Where $\delta_{ij} = +1$ if $d_i \in$ left leaves of c_j, or j=0, and
    $\delta_{ij} = -1$ otherwise

Ex) in previous error tree

$d_3 = c_0 - c_1 + c_3 = 6 - 2 + (-1) = 3$

Compression

- Wavelet compression
  - A large number of the detail coefficients turn out to be very small in magnitude
  - Removing these small coefficients introduces small errors
  - Lossy compression

Ex) from [6 2 1 -1], take two coefficients, 6, 2, that is [6 2 0 0] then

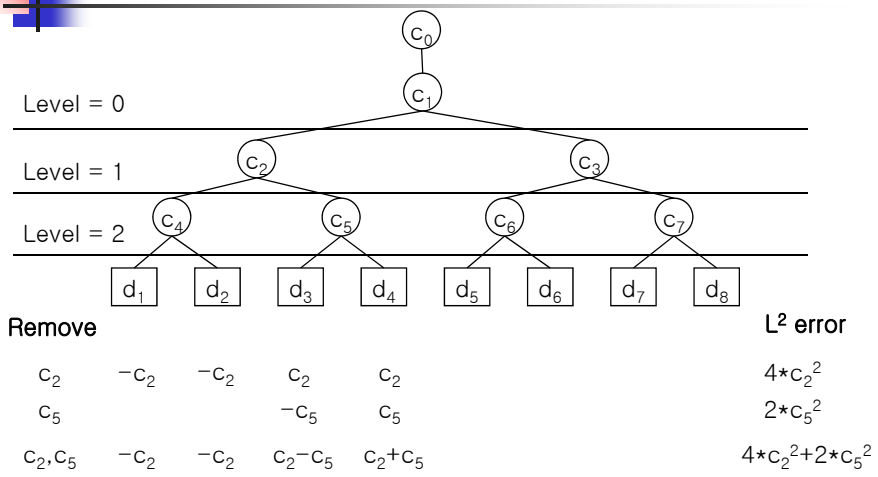
original data = [9 7 3 5]

reconstructed data = [8 8 4 4]

Normalization

- In order to equalize the importance of all wavelet coefficients
  - Normalizing the coefficients is needed
- If the coefficients have the same importance
  - We could choose the coefficients in order of absolute magnitude
  - Then we could achieve the best approximation of original data

Example error tree



Haar wavelet normalization

- Assume we use L^2 error
  - If we remove c_2, then it affects four values d_1, d_2, d_3, d_4 and results in $4*c_2^2$ L^2 error
  - If we remove c_5, then it affects two values d_5, d_6 and result in $2*c_5^2$ L^2 error
  - If we remove c_2, c_5, then it result in $4*c_2^2+ 2*c_5^2$ L^2 error
    - Removing each coefficient affects the L^2 error independently

Haar wavelet normalization(cont'd)

- If the values of c_2, c_5 are the same in absolute magnitude
  - Removing c_2 increases L^2 error more than removing c_5
- To compare the importance between c_2 and c_5 directly
  - we need to normalize the coefficients
  - If $4*c_2^2 = 2*c_5^2$, $c_2= \frac{1}{\sqrt{2}} c_5$
  - c_2 and $\frac{1}{\sqrt{2}} c_5$ have the same importance

Haar wavelet normalization(cont'd)

- The coefficients in the same level have the same importance
- Between two coefficients which has one level difference
  - The higher level coefficients have $\frac{1}{\sqrt{2}}$ times importance of the lower level
- To normalize coefficients
  - Divide each wavelet coefficient by $\sqrt{2^l}$, where l denotes the level
  - Ex) [6 2 1 -1] $\xrightarrow{\text{Normalization}}$ [6 2 $\frac{1}{\sqrt{2}}$ $-\frac{1}{\sqrt{2}}$]

Minimize L^2 error in Haar wavelet compression

- Compressing the original N data using B(<<N) wavelet coefficients
  - Normalize the coefficients
  - Choose the B wavelet coefficients with the largest absolute value
  - This is an optimal method of minimizing L^2 error using B wavelet coefficients

Wavelet-Based Histograms for Selectivity Estimation

Yossi Matias, Jeffrey Scott Vitter,
Min Wang

Motivation

- One important task in query optimization is selectivity estimation
- We need quick approximate answers to OLAP queries
  - When the exact answers are not required
- Histograms give very good approximations with limited space usage

Overview

- Wavelet-based Histograms
  - Based on a multi-resolution wavelet decomposition
  - Approximate the frequency distribution using given limited space
  - Built on the cumulative data distribution
- Haar wavelet
  - The Haar basis is the simplest wavelet basis
  - Fastest to compute and easiest to implement

Introduction

$a \le X \le b$: Range predicates, X is a non negative attribute of the domain of a relation R and a and b are constants

The domain D of X is the set of all possible values of X

Value set $V(\subseteq D)$ is the set of values of X that are actually present in R

Let $V = \{v_i : 1 \le i \le D\}$, where $v_i < v_j$ when $i < j$

The frequency f_i of v_i is the number of tuples $t \in R$ with $t.X = v_i$

The cumulative frequency c_i of v_i is the number of tuples $t \in R$ with $t.X \le v_i$, i.e., $c_i = \sum_{j=1}^{i} f_i$

Introduction(cont'd)

$\mathrm{T} = \{(v_1, f_1), (v_2, f_2), \ldots, (v_D, f_D)\}$

: The data distribution of X in R

$\mathrm{T}^C = \{(v_1, c_1), (v_2, c_2), \ldots, (v_D, c_D)\}$

: The cumulative data distribution of X in R

The cumulative data distribution of X , denoted by T^{C^+}

: the cumulative data distribution of T^C extended over the entire domain D by assigning a zero frequency to every value in $D - V$

Histogram construction algorithm

1. Form the extended cumulative data distribution T^{C+} of the attribute X
2. Compute the wavelet decomposition of T^{C+}
   - Obtaining a set of N wavelet coefficients
3. Keep only the m most significant wavelet coefficients
   - m(<<N) is given by user
   - The choice of m coefficients depends upon the particular thresholding method

Wavelet Decomposition

- Suppose that the data distribution T of attribute X is
  - {(0,2),(2,5),(3,2)}
- Then, the cumulated values are
  - {(0,2),(1,2),(2,7),(3,9)}
- Perform a wavelet transform on the extended cumulative frequencies

Resolution	Averages	Detail coefficients
4	[2 2 7 9]	
2	[2 8]	[0 1]
1	[5]	[3]

Wavelet transform -> [5 3 0 1]

Error measures

- Use the following three error measures
  1. Absolute error : $e_i^{abs} = |S_i - S_i'|$
  2. Relative error : $e_i^{rel} = e_i^{abs}/S_i = |S_i - S_i'|/S_i$, $S_i > 0$
  3. Combined error : $e_i^{comb} = \min\{\alpha \times e_i^{abs}, \beta \times e_i^{rel}\}$, $\alpha, \beta > 0$ if $S = 0$, $e_i^{comb} = \alpha \times e_i^{abs}$
  - Where S_i is original value of query q_i, and S_i' is estimated value of query q_i

Error measures(cont'd)

- Combined error
  - For very small frequencies
    - It may be good enough if the absolute error is small
  - For large frequencies
    - The absolute error may not be as meaningful as the relative error
- p-norm average error

$$\|e\|_p = \left(\frac{1}{Q}\sum_{1 \le i \le Q} e_i^{\,p}\right)^{1/p}$$

  - Where $p > 0$, Q is the number of queries

Thresholding methods

- The first step is normalizing coefficients
  - Here, use haar basis normaization
- 2-norm average absolute error
  - There is an optimal method for choosing m best wavelet coefficients
- Other than the 2-norm
  - No efficient technique is known for choosing m best wavelet coefficients

Thresholding methods(cont'd)

1. Choose the m largest (in absolute value) wavelet coefficients
   - Optimal for 2-norm error
2. Choose the m wavelet coefficients in a greedy way
   - First, choose the m largest coefficients and then repeatedly do the following two steps m times
     (a) Choose the coeff. whose inclusion leads to the largest reduction in error
     (b) Throw away the coeff. whose deletion leads to the smallest increase in error
   - Or do the above two steps repeatedly until a cycle is reached or improvement is small

Thresholding methods(cont'd)

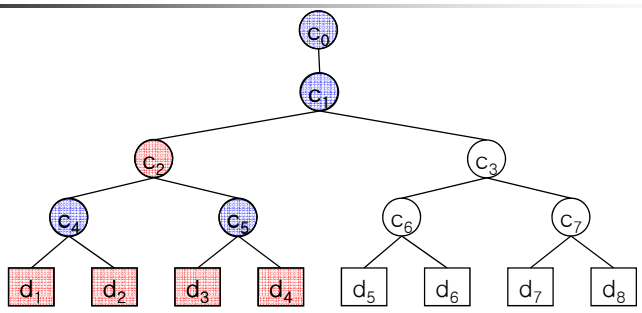
3. Start with the m/2 largest (in absolute value) wavelet coefficients
   - Choose the next m/2 coefficients greedily
4. Start with the 2m largest (in absolute value) wavelet coefficients
   - Throw away m of them greedily

- Method 2 does best overall in terms of accuracy

Thresholding methods(cont'd)

- Time complexity of naïve algorithm
  - Reconstructing of wavelet coefficients takes O(N) time
  - Each iteration of the greedy method requires $O(N^2)$ time
  - Thus the total time is $O(mN^2)$
- Using error tree, we can reduce the time to O(N(logN)logm) time
  - Store the error change introduced by adding or deleting wavelet coefficients
  - In each iteration, we just need to update value of the error change
    - Which is an ancestor or descendant of added or deleted nodes

Thresholding methods(cont'd)



Assume that we use method 2

If c2 is deleted in greedy choice

The error change value of c0, c1, c2, c4, c5 have to be updated

Thresholding methods(cont'd)

- Suppose that
  - At the ith step of the greedy choice of method 2, the node n_i is deleted
  - The subtree rooted at n_i has k' leaves
- Then update cost is k' x logN
- The worst case is m deleted or added coeff. are in the top logm levels
- So there are m terms of k' x logN
  - The sum of the terms in the same level becomes N x logN
  - Thus overall time complexity is O(N x logN x logm)

Probabilistic Wavelet Synopses

Minos Garofalakis, Phillip B. Gibbons

Motivation

Take 8 coefficients from 16 values
using L^2 error minimize method

original values	127	71	87	31	59	3	43	99
	100	42	0	58	30	88	72	130
wavelet answers	65	65	65	65	65	65	65	65
	100	42	0	58	30	88	72	130

- Similar data values have widely different approximations
  - 30 and 31 have approximations 30 and 65
- Widely different values, 3 and 127, have the same approximate answers 65

Motivation

- Major shortcoming of deterministic wavelet
  - The quality of the answers varies widely
  - No informative guarantees on the accuracy of a particular answer

Overview

- Propose probabilistic wavelet synopses
  - Based on probabilistic thresholding scheme
  - Assign each coefficient a probability of being included
    - based on its importance to the reconstruction of individual data values
  - Flip coins to select the synopsis
- Minimize
  - The expected mean-squared error
  - Upper bound on the maximum error in the reconstruction of the data

General Approach

- For each non zero wavelet coefficient c_i
  - Set random variable C_i such that

$$C_i = \begin{cases} \lambda_i & with\ probability\ \frac{C_i}{\lambda_i} \\ 0 & with\ probability\ 1-\frac{C_i}{\lambda_i} \end{cases}$$

    - Where λ_i is selected rounding value, $0 < \frac{c_i}{\lambda_i} \le 1$
    - Let $y_i = \frac{C_i}{\lambda_i}$ be the probability of rounding up
- The main idea of this approach
  - Select proper rounding value λ_i which minimize a desired error metric

General Approach(cont'd)

$$E[C_i] = \lambda_i \cdot \frac{c_i}{\lambda_i} + 0 \cdot (1 - \frac{c_i}{\lambda_i}) = c_i$$

$$Var\ (C_i) = E[C_i^{\ 2}] - (E[C_i])^2 = \lambda_i^{\ 2} \cdot \frac{c_i}{\lambda_i} - c_i^{\ 2} = (\lambda_i - c_i) \cdot c_i$$

d_j is original data, d_j' is estimate for d_j and d_j' is random variable

$$E[d_j'] = E(\sum_{c_i \in path\ (d_j)} \delta_{ij} \cdot C_i) = \sum_{c_i \in path\ (d_j)} \delta_{ij} \cdot E[C_i] = d_j$$

Each coefficient is rounded independently

$$Var\ (d_j') = E(\sum_{c_i \in path\ (d_j)} \delta_{ij} \cdot C_i) = \sum_{c_i \in path\ (d_j)} (\delta_{ij})^2 \cdot Var\ (C_i) = \sum_{c_i \in path\ (d_j)} (\lambda_i - c_i) \cdot c_i$$

General Approach(cont'd)

$$E[|WS_A|] = \sum_{i|c_i \neq 0} \frac{c_i}{\lambda_i} = \sum_{i|c_i \neq 0} y_i$$

$|WS_A|$ means the number of retained non-zero coefficients

Given desired number of retained coefficients B,

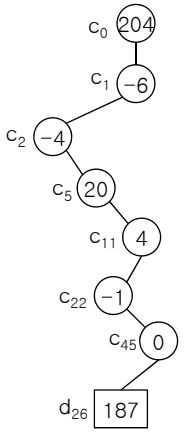
the choice of λ_i's needs to ensure that

$$E[|WS_A|] \leq B$$

$E[|WS_A|]$ means that expected number of retained coefficients

General Approach(cont'd)

i	0	1	2	5	11	22	45
c_i	204	−6	−4	20	4	−1	0
y_i	1	2/3	1/2	1	1/2	1/6	–
λ_i	204	−9	−8	20	8	−6	⊥
Coins	S	S	F	S	S	F	–
WS	204	-9	–	20	8	–	–
Coin flips	0.7	0.4	0.8	0.5	0.1	0.3	–



$d_{26}' = 204 - 9 - 20 + 8 = 183$

(the actual value for $d_{26} = 187$)

MinL2 Algorithm

- To minimize the expected mean-squared error
- Select λ_i to minimize E[L^2]

$$E[L^2] = E[\sum_j (d_j' - d_j)^2] = \sum_{i|c_i \neq 0} \frac{(\lambda_i - c_i) \cdot c_i}{2^{level(c_i)}}$$

- Expected L^2 error minimization
  - Minimize

$$\sum_{i|c_i \neq 0} \frac{(\lambda_i - c_i) \cdot c_i}{2^{level(c_i)}} \quad , 0 < \frac{c_i}{\lambda_i} \leq 1, \quad E[|WS_A|] \leq B$$

- The result of MinL2 is not optimal for L^2 error

MinRelVar Algorithm

- Minimize individual answer errors
  - Minimizing relative error is more important to approximate query answering
  - MinRelVar algorithm is minimizing the maximum reconstruction error
- Goal
  - Produce estimates d_i' for each data value d_i such that for a given sanity bound S > 0, the ratio

$$\frac{|d_i' - d_i|}{\max\{|d_i|, S\}}$$

  is small with high probability

MinRelVar Algorithm(cont'd)

- Normalized standard error(NSE)

$$NSE\ (d_i') = \frac{\sqrt{Var\ (d_i')}}{\max\{\ |d_i|, S\}}$$

  - The value of NSE means that expected relative error at d_i
- Maximum normalized standard error minimization

$$Minimize \max_{path(d_k) \in PATHS} \frac{\sqrt{\sum_{i \in path(d_k)} (\lambda_i - c_i) \cdot c_i}}{\max\{|d_k|, s\}}, \quad 0 < \frac{c_i}{\lambda_i} \le 1, \quad E[|WS_A|] \le B$$

  - PATHS={path(d_i) : i=1, …, N

MinRelVar Algorithm(cont'd)

- Formulating a dynamic programming recurrence
  - Let T_j be the sub-tree rooted at the node c_j
  - M[j, B] denotes the optimal (i.e., minimum) value of the maximum NSE(d_k^2)
  - d_k is the leaf node data of T_j
  - Assume a space budget of B

MinRelVar Algorithm(cont'd)

$$M[j,B] = \min_{\substack{y_i \in (0,1],\, i \in T_j | c_i \neq 0: \\ \sum_{i \in T_j | c_i \neq 0} y_i \leq B}} \left\{ \max_{path(d_k) \in PATHS_j} \left\{ \sum_{i \in path(d_k)} \frac{VAR(i, y_i)}{\max\{d_k^{\,2}, s_2\}} \right\} \right\}$$

$$VAR(i, y_i) = (\lambda_i - c_i) \cdot c_i = \frac{1 - y_i}{y_i} \cdot c_i^{\,2}$$

$PATHS_j$ denotes the set of all root-to-leaf paths in T_j

MinRelVar Algorithm(cont'd)

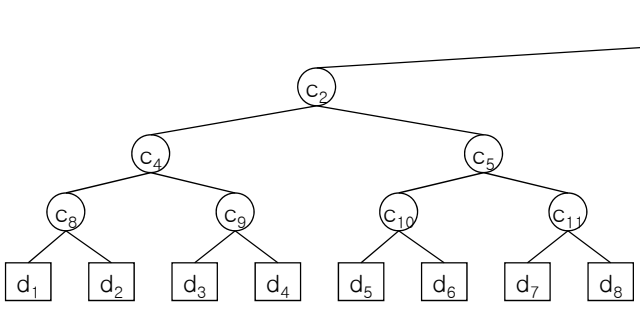
$$M[j,B] = \begin{cases} \min\limits_{\substack{y_j \in (0, \min\{1,B\}]; \\ b_L \in [0, B - y_j]}} \left\{ \max \left\{ \begin{array}{l} \dfrac{VAR(j, y_j)}{NORM(2j)} + M[2j, b_L], \\ \dfrac{VAR(j, y_j)}{NORM(2j+1)} + M[2j+1, B - y_j - b_L] \end{array} \right\} \right\} & \text{if } j < N, c_j \neq 0, \text{and } B > 0 \\ \min_{b_L \in [0,B]} \{\max\{M[2j, b_L], M[2j+1, B - b_L]\}\} & \text{if } j < N \text{ and } c_j = 0 \\ 0 & \text{if } j \geq N \\ \infty & \text{otherwise} \end{cases}$$

where $NORM(i) = \max\{\min_{k \in T_i}\{d_k^{\,2}\}, S^2\}$

MinRelVar Algorithm(cont'd)

- Give constraint to the value $y_i \in (0,1]$
  - If there is no constraint, y_i will have infinite choice which is undesirable
  - Quantize the y_i value to the number of q values
  - q is an input parameter
  - So we take y_i from $\{\frac{1}{q}, \frac{2}{q}, \ldots, 1\}$
  - As a result budget $b \in [0,B]$ also has quantized value $\{0, \frac{1}{q}, \ldots, B\}$

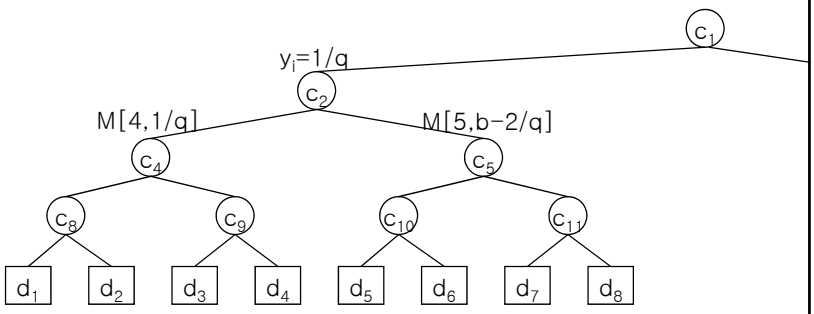
MinRelVar Algorithm(cont'd)



Assume that we already store optimal M for T_4 and T_5 for all possible budget number

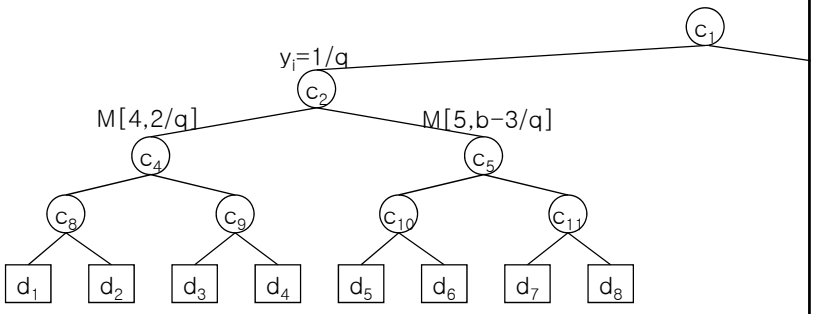
Then using table M, find M[2,b]

MinRelVar Algorithm(cont'd)



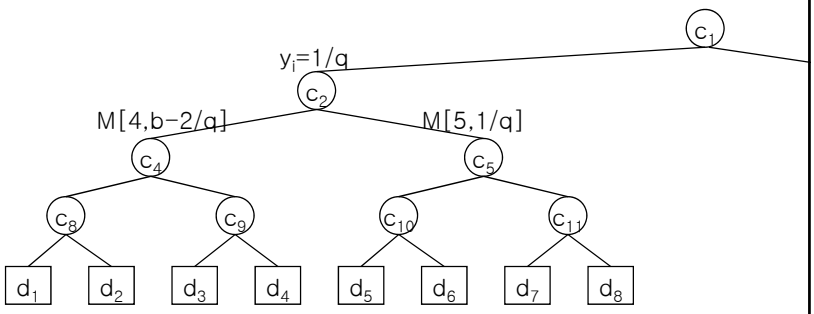
$$M[2,b] = \max\{\frac{VAR(2,y_2)}{NORM(4)} + M[4,\frac{1}{q}],\quad \frac{VAR(2,y_2)}{NORM(5)} + M[5,b-\frac{1}{q}-\frac{1}{q}]\}$$

MinRelVar Algorithm(cont'd)



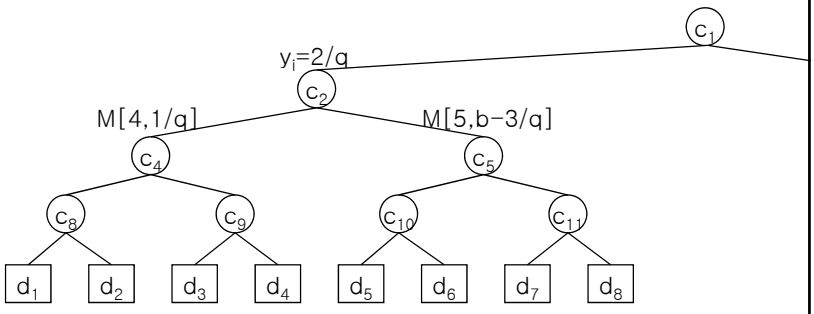
$$M[2,b] = \max\{\frac{VAR(2,y_2)}{NORM(4)} + M[4,\frac{2}{q}],\quad \frac{VAR(2,y_2)}{NORM(5)} + M[5,b-\frac{1}{q}-\frac{2}{q}]\}$$

MinRelVar Algorithm(cont'd)

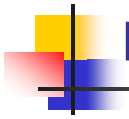


$$M[2,b] = \max\{\frac{VAR(2,y_2)}{NORM(4)} + M[4,b-\frac{2}{q}],\quad \frac{VAR(2,y_2)}{NORM(5)} + M[5,\frac{1}{q}]\}$$

MinRelVar Algorithm(cont'd)

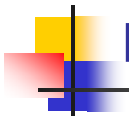


$$M[2,b] = \max\{\frac{VAR(2,y_2)}{NORM(4)} + M[4,\frac{1}{q}],\quad \frac{VAR(2,y_2)}{NORM(5)} + M[5,b-\frac{3}{q}]\}$$



MinRelVar Algorithm(cont'd)

- If complete assigning y_i from 1/q to b
  - Then minimum value M[2,b] is stored in table
- Compute 1 table entry takes $O(q^2B)$
  - Using property that M[j,b] is a decreasing function of the budget B,
    - We can use more efficient $O(q\log(qB_j))$
- Needed table entry size is O(NqB)
  - Thus overall running time complexity is $O(Nq^2B\log(qB))$



References

- H. V. Jagadish, Nick Koudas, S. Muthukrishnan, Viswanath Poosala, Kenneth C. Sevcik, Torsten Suel: Optimal Histograms with Quality Guarantees. VLDB 1998: 275-286
- Yossi Matias, Jeffrey Scott Vitter, Min Wang: Wavelet-Based Histograms for Selectivity Estimation. SIGMOD Conference 1998: 448-459
- Sudipto Guha, Nick Koudas, Kyuseok Shim: Data-streams and histograms. STOC 2001: 471-475
- Minos N. Garofalakis, Phillip B. Gibbons: Wavelet synopses with error guarantees. SIGMOD Conference 2002: 476-487