Computing Skylines with Noisy Comparisons

PODS 2015.

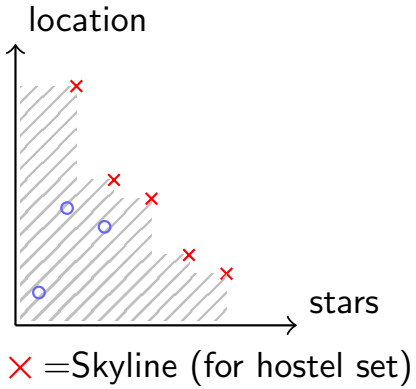
Benoît Groz and Tova Milo

June 2015

Outline

Skyline

Skyline = Pareto Optima = Maximal Vectors



Set S of d-dimensional items.

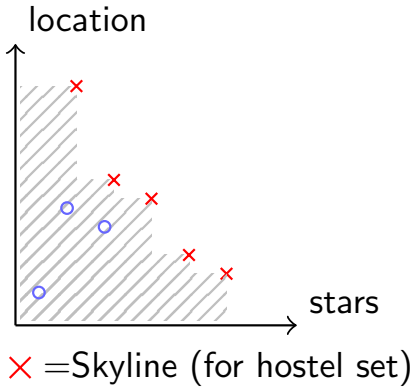
$$\mathrm{Sky}(S) = \text{items that are not dominated}$$

Motivations:

- skylines filter out interesting tuples
  alternative to top-k (multiple criteria)
  $\hookrightarrow$ popular research topic
- preferences may be noisy
- comparisons sometimes more relevant than numerical values
- in crowdsourcing scenario, cost= # comparisons

Skyline

Skyline = Pareto Optima = Maximal Vectors



Set S of d-dimensional items.

$$\mathrm{Sky}(S) = \text{items that are not dominated}$$

Motivations:

- skylines filter out interesting tuples
  alternative to top-k (multiple criteria)
  $\hookrightarrow$ popular research topic
- preferences may be noisy
- comparisons sometimes more relevant than numerical values
- in crowdsourcing scenario, cost= # comparisons

Model:

Noisy comparisons

Noisy oracle model:
Users answer with error probability $p = 1/3$ questions:
"*is o_2 better than o_1 on dimension i?*" $(o_1 <_i o_2?)$

Cost= Oracle complexity: # comparisons

(possibly, Cost $\ll$ Computational complexity)

Reduce err. pr.:
repeat comparison!

(independence)

Model:

Noisy comparisons

Noisy oracle model:
Users answer with error probability $p = 1/3$ questions:
"is o_2 better than o_1 on dimension i?" $(o_1 <_i o_2?)$

Cost= Oracle complexity: # comparisons

(possibly, Cost $\ll$ Computational complexity)

Reduce err. pr.:
repeat comparison!

(independence)

Our skyline problem:
Input: S, δ
Objective: compute $\mathrm{Sky}(S)$ with error probability $< \delta$, at minimal cost (worst case analysis).

Skyline

Simplifications:

- ground truth assumption questionable
- noise model simplistic
- does not consider HCI aspects of crowdsourcing, batch etc.

Outline

Noiseless case: skyline algorithm ($d = 2$)

Notation: $n = |S|$, $k = |\mathrm{Sky}(S)|$, d : dimension.

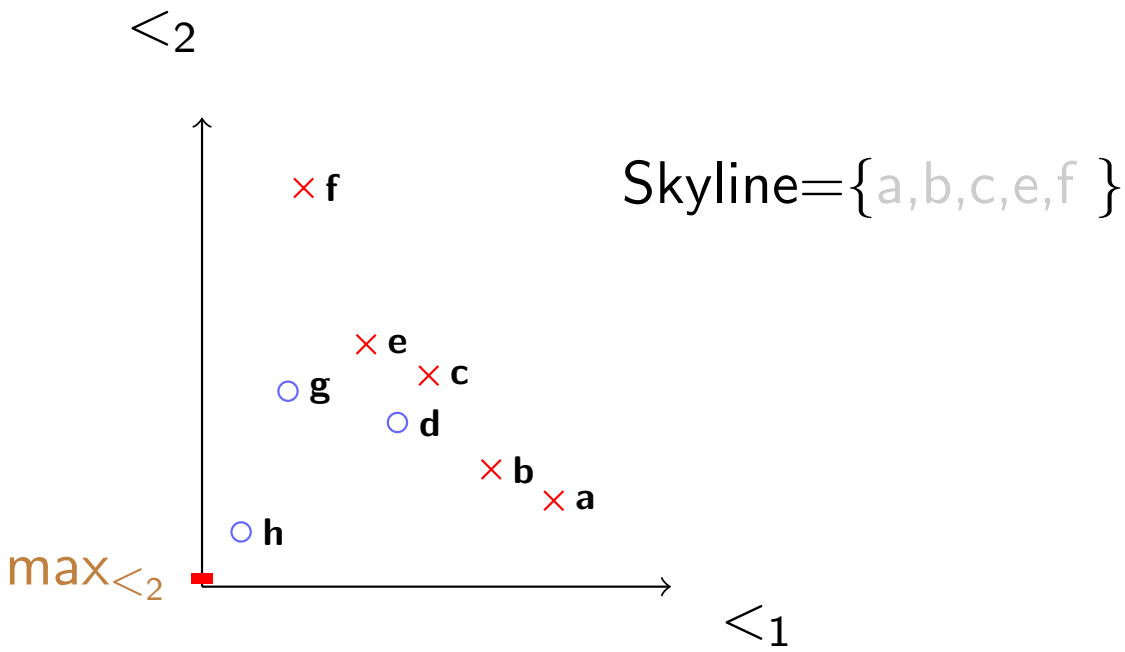
Res $\leftarrow \emptyset$

For $i = 1, \ldots, \infty$:

$\quad$ Res $\leftarrow$ Res $\cup \max_{<_1}\{x \in S \setminus \text{Res} \mid x \not\prec \text{Res}\}$

Cost: $O(nk)$

for $d = 2$, $x \not\prec$ Res checked in $O(1)$ if we maintain $\max_{<_2}$ over Res



Noiseless case: skyline algorithm ($d = 2$)

Notation: $n = |S|$, $k = |\mathrm{Sky}(S)|$, d : dimension.

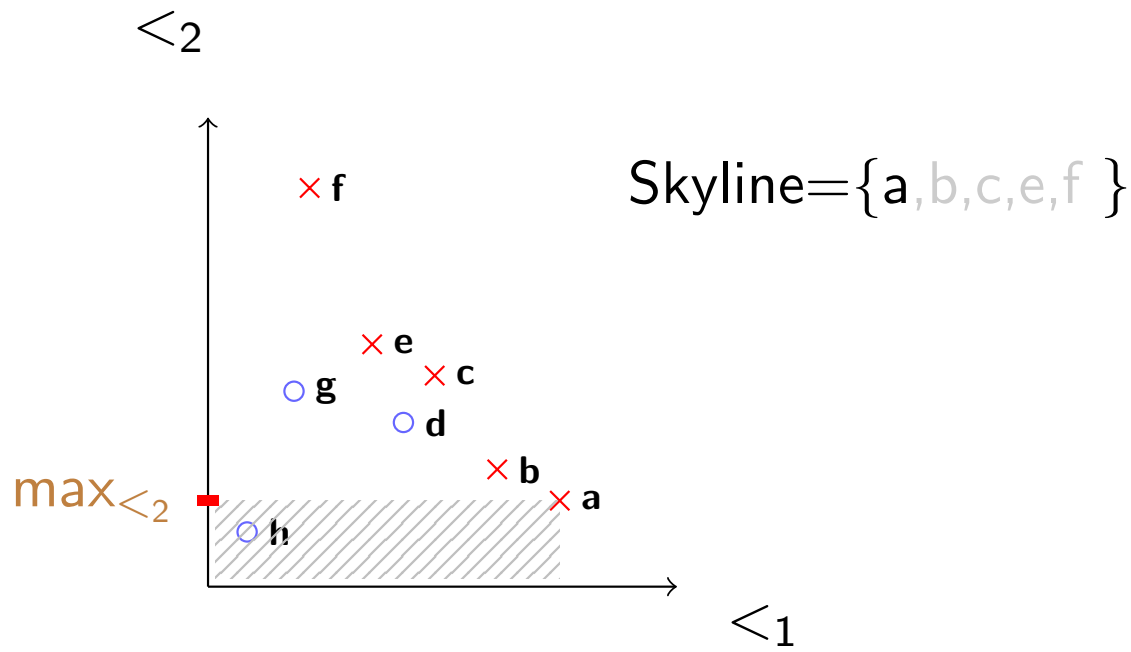
Res $\leftarrow \emptyset$

For $i = 1, \ldots, \infty$:

  Res $\leftarrow$ Res $\cup \max_{<_1}\{x \in S \setminus \text{Res} \mid x \not\prec \text{Res}\}$

Cost: $O(nk)$

for $d = 2$, $x \not\prec$ Res checked in $O(1)$ if we maintain $\max_{<_2}$ over Res



Noiseless case: skyline algorithm ($d = 2$)

Notation: $n = |S|$, $k = |\mathrm{Sky}(S)|$, d : dimension.

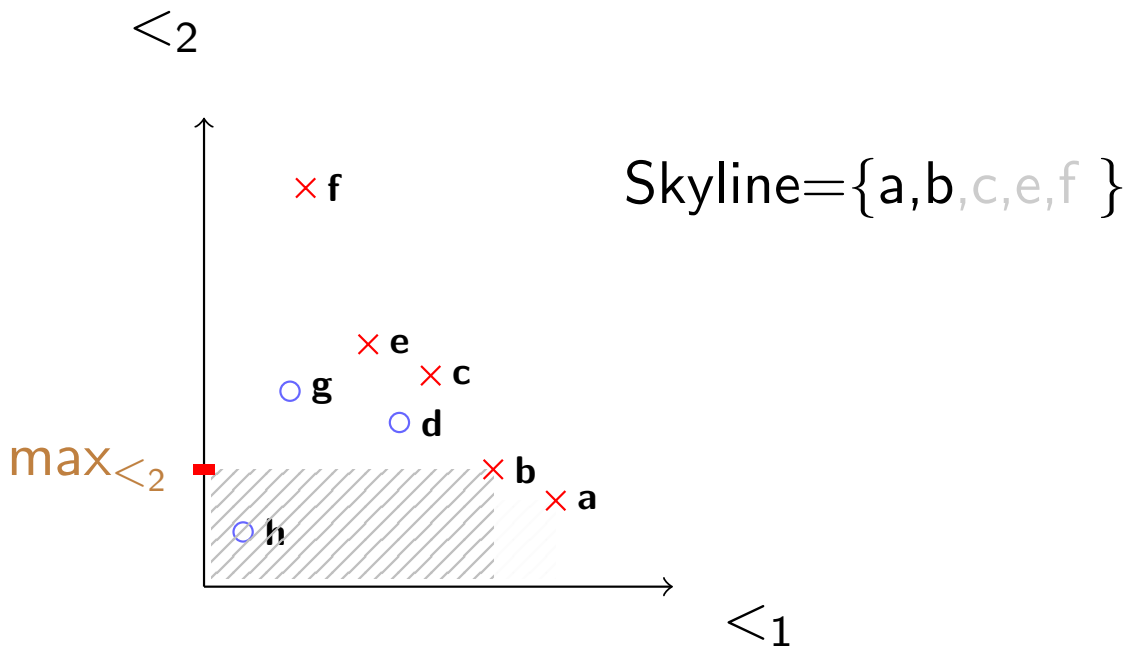
Res $\leftarrow \emptyset$

For $i = 1, \ldots, \infty$:

  Res $\leftarrow$ Res $\cup \max_{<_1}\{x \in S \setminus \text{Res} \mid x \not\prec \text{Res}\}$

Cost: $O(nk)$

for $d = 2$, $x \not\prec$ Res checked in $O(1)$ if we maintain $\max_{<_2}$ over Res



Noiseless case: skyline algorithm ($d = 2$)

Notation: $n = |S|$, $k = |\mathrm{Sky}(S)|$, d : dimension.

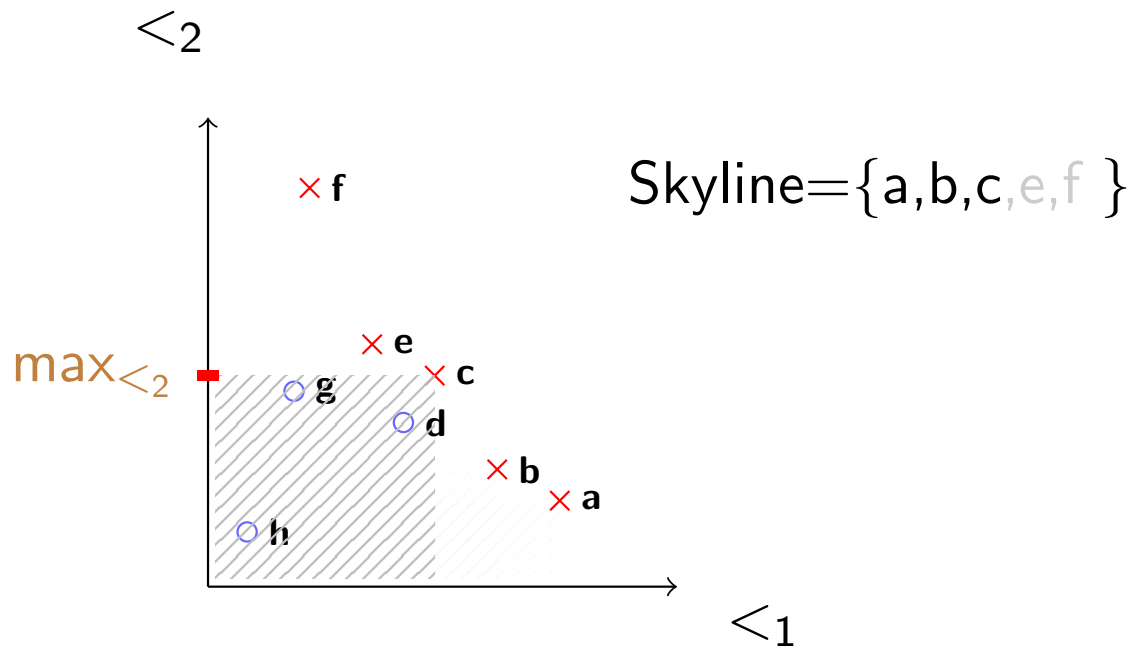
Res $\leftarrow \emptyset$

For $i = 1, \ldots, \infty$:

  Res $\leftarrow$ Res $\cup \max_{<_1}\{x \in S \setminus \text{Res} \mid x \not\prec \text{Res}\}$

Cost: $O(nk)$

for $d = 2$, $x \not\prec$ Res checked in $O(1)$ if we maintain $\max_{<_2}$ over Res



Noiseless case: skyline algorithm ($d = 2$)

Notation: $n = |S|$, $k = |\mathrm{Sky}(S)|$, d : dimension.

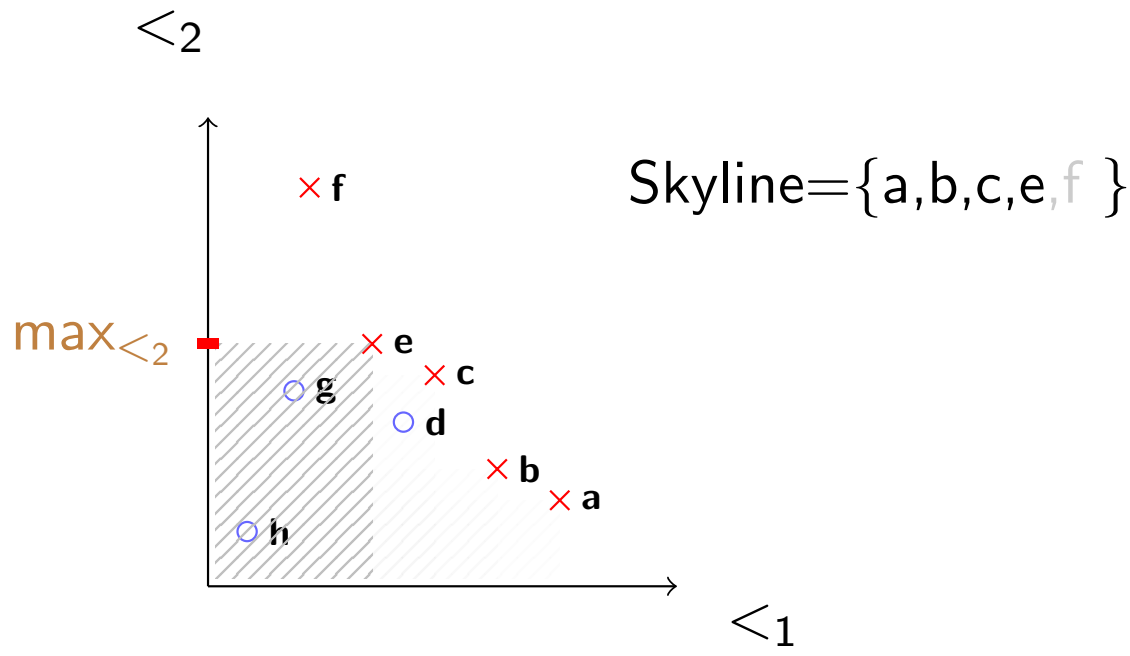
Res $\leftarrow \emptyset$

For $i = 1, \ldots, \infty$:

  Res $\leftarrow$ Res $\cup \max_{<_1}\{x \in S \setminus \text{Res} \mid x \not\prec \text{Res}\}$

Cost: $O(nk)$

for $d = 2$, $x \not\prec$ Res checked in $O(1)$ if we maintain $\max_{<_2}$ over Res



$d \prec c$

Noiseless case: skyline algorithm ($d = 2$)

Notation: $n = |S|$, $k = |\mathrm{Sky}(S)|$, d : dimension.

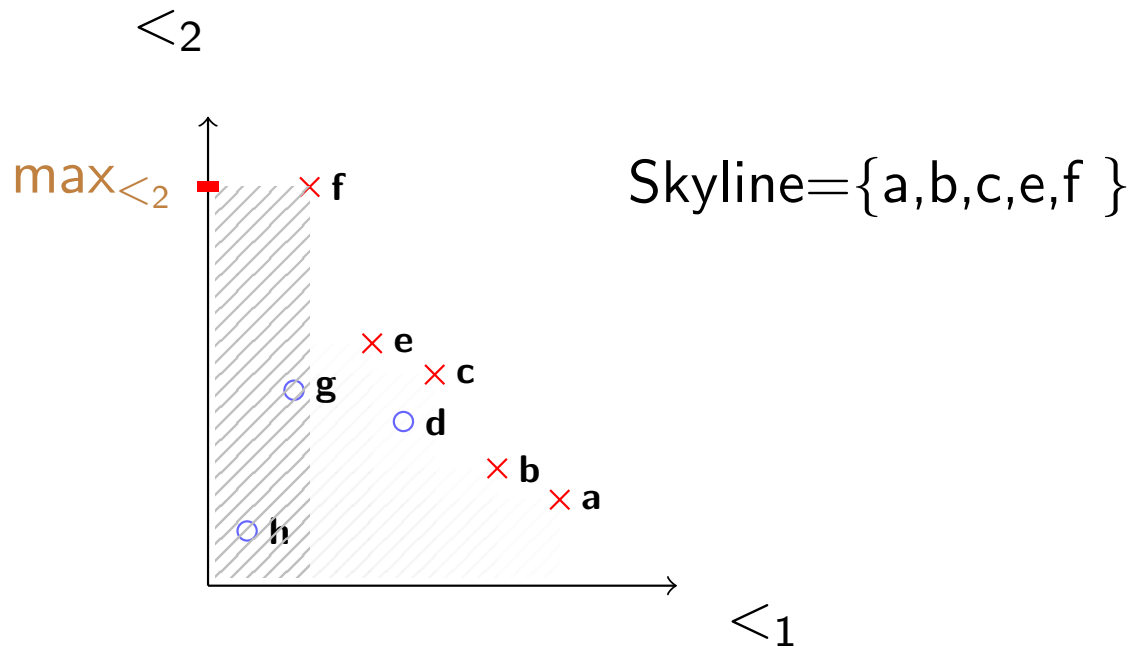
Res $\leftarrow \emptyset$

For $i = 1, \ldots, \infty$:

  Res $\leftarrow$ Res $\cup \max_{<_1}\{x \in S \setminus \text{Res} \mid x \not\prec \text{Res}\}$

Cost: $O(nk)$

for $d = 2$, $x \not\prec$ Res checked in $O(1)$ if we maintain $\max_{<_2}$ over Res



Skyline={a,b,c,e,f }

$d \prec c$

$g \prec f$

$h \prec f$

Outline

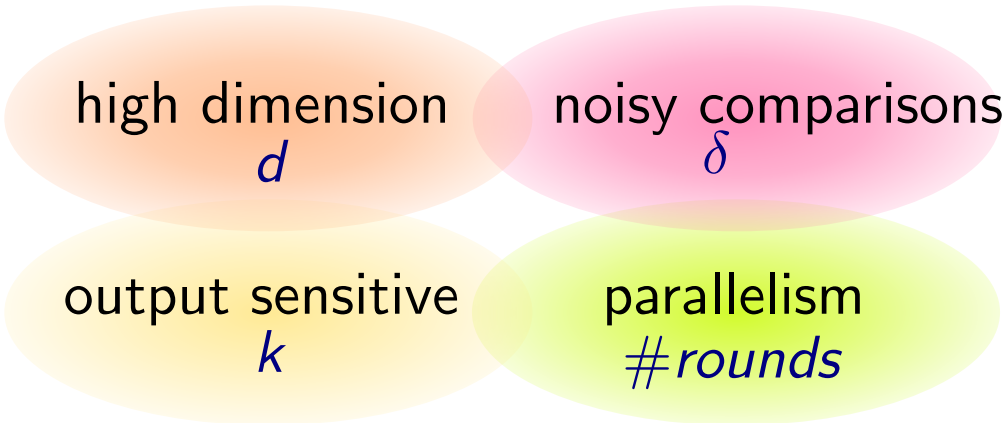
State of the art: noiseless skylines

Computational complexity

Notation: $n = |S|$, $k = |\mathrm{Sky}(S)|$, d : dimension.

Worst-case	Algorithm	author
$O(dnk)$	naïve	
$O(n \log^{d-2} n)$	Divide&Conquer	[Kung et al.'75+Sheng&Tao'12]
$O(d^2 n \log^{d-2} k)$	Divide&Conquer+median	[Kirkpatrick et al.'85]
When $d = 2, 3$:		
$O(n\mathcal{H}(S))$	Divide&Conquer+median	[Afshani et al.'09]
$\Omega(n\mathcal{H}(S))$	k-d tree+ Info. theory	[Afshani et al.'09]
Other bounds for I/O, RAM models, average complexity, etc.		

Our challenges (oracle complexity) :



State of the art: noiseless skylines

high d

sensitive k

Oracle complexity

Notation: $n = |S|$, $k = |\mathrm{Sky}(S)|$, d : dimension.

Oracle complexity:

$$\left.\begin{array}{l} n \log k + O(n\sqrt{\log k}) \\ n \log n + O(n) \end{array}\right\} \text{ when } d = 2$$

$2n \log n + O(n)$ when $d = 3$

$O(dn \log n)$, $O(dnk)$

$\Omega(dn)$, $n \log k$ when $d = 2$

Computational compl. (reminder):

$O(dn \log^{d-2} k)$

$O(n \log^{d-2} n)$

✎ *Open: oracle complexity for large d? $O(dn \log n)$ vs $|\Omega(\max(n \log n, dn))|$.*

State of the art: sort& search with noise

noise

Notation: $n = |S|$, δ=err. pr.

MAX	$O(n \log \frac{1}{\delta})$
SORTING	$O(n \log \frac{n}{\delta})$
Binary search	$O(\log \frac{n}{\delta})$
TOP-k	$O(n \log \frac{\min(k,n-k)}{\delta})$ $\rightarrow$ like sorting if $k = \frac{n}{2}$
OR	$O(n \log \frac{1}{\delta})$

These bounds are tight ! [Feige et al'94]

That's theory! Constants like "repeat 100 times".

State of the art: sort& search with noise

noise

Notation: $n = |S|$, δ=err. pr.

MAX	$O(n \log \frac{1}{\delta})$
SORTING	$O(n \log \frac{n}{\delta})$
Binary search	$O(\log \frac{n}{\delta})$
TOP-k	$O(n \log \frac{\min(k,n-k)}{\delta})$ $\rightarrow$ like sorting if $k = \frac{n}{2}$
OR	$O(n \log \frac{1}{\delta})$ $\rightarrow$ returns first true variable

These bounds are tight ! [Feige et al'94]

That's theory! Constants like "repeat 100 times".

State of the art: sort& search with noise

noise

Notation: $n = |S|$, δ=err. pr.

MAX	$O(n \log \frac{1}{\delta})$
SORTING	$O(n \log \frac{n}{\delta})$
Binary search	$O(\log \frac{n}{\delta})$
TOP-k	$O(n \log \frac{\min(k,n-k)}{\delta})$ $\rightarrow$ like sorting if $k = \frac{n}{2}$
OR	$O(n \log \frac{1}{\delta})$ $\rightarrow$ returns first true variable

These bounds are tight ! [Feige et al'94]

That's theory! Constants like "repeat 100 times".

Lemma (trust-preserving OR/MAX)

If the comparison oracle has error probability $\leq \delta$, $O(n)$ comparisons are enough for OR and MAX.

State of the art: sort& search with noise

noise

Notation: $n = |S|$, δ=err. pr.

MAX	$O(n \log \frac{1}{\delta})$
SORTING	$O(n \log \frac{n}{\delta})$
Binary search	$O(\log \frac{n}{\delta})$
TOP-k	$O(n \log \frac{\min(k,n-k)}{\delta})$ $\rightarrow$ like sorting if $k = \frac{n}{2}$
OR	$O(n \log \frac{1}{\delta})$ $\rightarrow$ returns first true variable

These bounds are tight ! [Feige et al'94]

Our main result:

Check Skyline	$O(dn \log \frac{dk}{\delta})$
Skyline	$O(dkn \log \frac{dk}{\delta})$

We believe this is not tight

Oracle complexity

noise

Upper bound

Any problem can be solved with error probability $< \delta$ in $O(dn \log \frac{dn}{\delta})$.

For constant d when $k = n$, this provides a tight bound on skyline computation/checking.

Proof: Sort along each dimension with err. pr. δ/d.

Lower bound for checking skylines: $n \log_2 n$ from noiseless case, and $\Omega(n \log \frac{1}{\delta})$ from OR. □

Also, any algorithm can be adapted to noise by repeating comparisons, but at high cost: $f(x) \mapsto f(x) \log(f(x)/\delta)$

Reminder:

OR,MAX	SORTING	Binary search
$O(n)$ trust preserving	$O(n \log \frac{n}{\delta})$	$O(\log \frac{n}{\delta})$

Oracle complexity

noise

Upper bound

Any problem can be solved with error probability $< \delta$ in $O(dn \log \frac{dn}{\delta})$.

For constant d when $k = n$, this provides a tight bound on skyline computation/checking.

Proof: Sort along each dimension with err. pr. δ/d.

Lower bound for checking skylines: $n \log_2 n$ from noiseless case, and $\Omega(n \log \frac{1}{\delta})$ from OR. □

Also, any algorithm can be adapted to noise by repeating comparisons, but at high cost: $f(x) \mapsto f(x) \log(f(x)/\delta)$

Reminder:

OR,MAX	SORTING	Binary search
$O(n)$ trust preserving	$O(n \log \frac{n}{\delta})$	$O(\log \frac{n}{\delta})$

Outline

Checking candidate skyline

high d noise sensitive k

Theorem: checking skyline

Given $C \subseteq S$, we can check if $C = \mathrm{Sky}(S)$ in

1. $O(dn \log \frac{d|C|}{\delta})$
2. $O(dn|C| \log \frac{1}{\delta})$.

Proof: Let $C = \{w_1, \ldots, w_{|C|}\}$, $S = \{v_1, \ldots, v_n\}$. Check with err. pr. $\delta/2$:

- $C \supseteq \mathrm{Sky}(S)$: $\bigwedge_{i \le n} v_i \preccurlyeq C$
- $C = \mathrm{Sky}(C)$: $\bigwedge_{w_i \neq w_j} w_j \not\preccurlyeq w_i$

We check $v \preccurlyeq w = \bigwedge_{i \le d} v \le_i w$ with err. pr. $1/3$ using OR. Cost: $O(d)$

Reminder:

OR,MAX	SORTING	Binary search
$O(n \log \frac{1}{\delta})$	$O(n \log \frac{n}{\delta})$	$O(\log \frac{n}{\delta})$

Checking candidate skyline

high d noise sensitive k

Theorem: checking skyline

Given $C \subseteq S$, we can check if $C = \mathrm{Sky}(S)$ in

1. $O(dn \log \frac{d|C|}{\delta})$
2. $O(dn|C| \log \frac{1}{\delta})$.

Proof: Let $C = \{w_1, \ldots, w_{|C|}\}$, $S = \{v_1, \ldots, v_n\}$. Check with err. pr. $\delta/2$:

- $C \supseteq \mathrm{Sky}(S)$: $\bigwedge_{i \le n} v_i \preccurlyeq C$
- $C = \mathrm{Sky}(C)$: $\bigwedge_{w_i \neq w_j} w_j \not\preccurlyeq w_i$

We check $v \preccurlyeq w = \bigwedge_{i \le d} v \le_i w$ with err. pr. $1/3$ using OR. Cost: $O(d)$

Proof of 2: Use the OR algorithm ($v_i \preccurlyeq C$ iff $\bigvee_j v_i \preccurlyeq w_j$):
$C \supseteq \mathrm{Sky}(S) : \bigwedge_{i \le n} \bigvee_j v_i \preccurlyeq w_j$. Cost: $O(n|C| \log \frac{1}{\delta}) \times O(d)$ □

Reminder:

OR,MAX	SORTING	Binary search
$O(n \log \frac{1}{\delta})$	$O(n \log \frac{n}{\delta})$	$O(\log \frac{n}{\delta})$

Checking candidate skyline

high d noise sensitive k

Theorem: checking skyline

Given $C \subseteq S$, we can check if $C = \mathrm{Sky}(S)$ in

1. $O(dn \log \frac{d|C|}{\delta})$
2. $O(dn|C| \log \frac{1}{\delta})$.

Proof of 1:

- Sort C with err. pr. $\delta/(2d)$ in every dimension. Error $\leq \delta/2$.
- Check $C = \mathrm{Sky}(C)$. Cost: 0. Error: 0.
- Check $\bigwedge_{i \leq n} v_i \preccurlyeq C$ with err. pr. $\delta/2$ Cost: $O(d|C| \log(d|C|/\delta))$:
  - Obtain an oracle for $v \preccurlyeq C$ with err. pr. $\delta/2$ by binary insertion of v in C with err. pr. $\delta/(2d)$ on each dimension.
    Cost per oracle call: $O(d \log(d|C|/\delta))$.
  - We use trust preserving OR with $O(n)$ calls to oracle $v \preccurlyeq C$. □

Reminder:

OR,MAX	SORTING	Binary search
$O(n)$ trust preserving	$O(n \log \frac{n}{\delta})$	$O(\log \frac{n}{\delta})$

Checking candidate skyline

high d noise sensitive k

Theorem: checking skyline

Given $C \subseteq S$, we can check if $C = \mathrm{Sky}(S)$ in

1. $O(dn \log \frac{d|C|}{\delta})$ tight for constant d
2. $O(dn|C| \log \frac{1}{\delta})$. tight for constant k

Proof of 1*:*

- Sort C with err. pr. $\delta/(2d)$ in every dimension. Error $\leq \delta/2$.
- Check $C = \mathrm{Sky}(C)$. Cost: 0. Error: 0.
- Check $\bigwedge_{i \leq n} v_i \preccurlyeq C$ with err. pr. $\delta/2$ Cost: $O(d|C| \log(d|C|/\delta))$:
  - Obtain an oracle for $v \preccurlyeq C$ with err. pr. $\delta/2$ by binary insertion of v in C with err. pr. $\delta/(2d)$ on each dimension.
    Cost per oracle call: $O(d \log(d|C|/\delta))$.
  - We use trust preserving OR with $O(n)$ calls to oracle $v \preccurlyeq C$. □

Reminder:

OR,MAX	SORTING	Binary search
$O(n)$ trust preserving	$O(n \log \frac{n}{\delta})$	$O(\log \frac{n}{\delta})$

Computing skyline (output-sensitive)

high d noise sensitive k

Theorem: computing skyline

We can compute the skyline $\mathrm{Sky}(S)$ in

1. $O(dkn \log \frac{dk}{\delta})$
2. $O(dk^2 n \log \frac{k}{\delta})$.

Computing skyline (output-sensitive)

high d noise sensitive k

Theorem: computing skyline

We can compute the skyline $\mathrm{Sky}(S)$ in

1. $O(dkn \log \frac{dk}{\delta})$
2. $O(dk^2 n \log \frac{k}{\delta})$.

Algorithm sketch (1/2): *Assume we know k in advance.*

SkySample(S, k, δ)

$Res \leftarrow \emptyset$.

for $i = 1, \ldots, \infty$

compute $w = MAX_{\leq_{\text{lex}}}\{v \in S \mid v \not\preccurlyeq Res\}$ with err. pr. $\frac{\delta}{k}$.

if $w = \emptyset$ **return** Res

else $Res \leftarrow Res \cup w$

Error: $\leq \delta$. Cost: k calls to "modified" MAX.

To guess the value of k, use Chan's trick:

compute SkySample(S, k_i, δ_i) where $k_i = 2^{2^i}, \delta_i = \frac{\delta}{2^i}$ for $i = 1, 2....$

Error: $\leq \delta$. Cost: we need efficient procedure for "modified" MAX. □

Computing skyline (output-sensitive)

high d noise sensitive k

Theorem: computing skyline

We can compute the skyline $\mathrm{Sky}(S)$ in

1. $O(dkn \log \frac{dk}{\delta})$
2. $O(dk^2 n \log \frac{k}{\delta})$.

Objective: answer the k queries $MAX_{\leq_{\text{lex}}}\{v \in S \mid v \not\preccurlyeq Res\}$ with err. pr. δ/k.

Algorithm sketch (2/2):

Return MAX on order $\lessdot$: $v \lessdot v'$ iff $v \preccurlyeq Res \vee (v' \not\preccurlyeq Res \wedge v \leq_{lex} v')$.

Lexicographic oracle with err. pr. $\frac{\delta}{2k}$ in $O(d \log \frac{k}{\delta})$

$\hookrightarrow$ *first true comparisons in* $\left(\bigvee_{i \leq d} v <_i w\right)$ *vs* $\left(\bigvee_{i \leq d} w <_i v\right)$

Computing skyline (output-sensitive)

high d noise sensitive k

Theorem: computing skyline

We can compute the skyline $\mathrm{Sky}(S)$ in

1. $O(dkn \log \frac{dk}{\delta})$
2. $O(dk^2 n \log \frac{k}{\delta})$.

Objective: answer the k queries $MAX_{\leq_{\text{lex}}}\{v \in S \mid v \not\preccurlyeq Res\}$ with err. pr. δ/k.

Algorithm sketch (2/2):

Return MAX on order $\lessdot$: $v \lessdot v'$ iff $v \preccurlyeq Res \vee (v' \not\preccurlyeq Res \wedge v \leq_{lex} v')$.

Lexicographic oracle with err. pr. $\frac{\delta}{2k}$ in $O(d \log \frac{k}{\delta})$

Two options to check dominance ($v \preccurlyeq Res$):

1. *Sort Res once in $O(dk \log dk/\delta)$ with err. pr. $\frac{\delta}{2k}$. Then check in $O(d \log \frac{dk}{\delta})$ if $v \preccurlyeq Res$ with err. pr. $\frac{\delta}{2k}$ (binary+OR).*
   Cost: $O(n \times d \log \frac{dk}{\delta})$ per MAX query.
2. *Check in $O(dk \log \frac{k}{\delta})$ if $v \preccurlyeq Res$ with err. pr. $\frac{k}{\delta}$ (OR).*
   Cost: $O(n \times dk \log \frac{k}{\delta})$ per MAX query. □

Number of comparison rounds

noise

#rounds

In crowdsourcing scenarios, rounds = delay.

	OR	MAX	SORTING	Binary search
cost	$O(n)$	$O(n)$	$O(n \log \frac{n}{\delta})$	$O(\log \frac{n}{\delta})$
#rounds	$\log^* n$	$\log \log n$	$\log n$	$\log n$

- $\bigwedge_{i=1}^{n} \bigvee_{j=1}^{k} \bigwedge_{h=1}^{d} v_i \leq_h w_j$: $\log^* n \times \log^* k \times \log^* d$
- All dimensions can be sorted independently.
- δ has no impact on #rounds.

Conclusion

Upper bound for all pbs: $O(dn \log \frac{dn}{\delta})$

Theorem: checking skyline

Given $C \subseteq S$, we can check if $C = \mathrm{Sky}(S)$ in

1. $O(dn \log \frac{d|C|}{\delta})$
2. $O(dn|C| \log \frac{1}{\delta})$.

$\Omega(n \log k + dn \log \frac{1}{\delta})$

Theorem: computing skyline

We can compute the skyline $\mathrm{Sky}(S)$ in

1. $O(dkn \log \frac{dk}{\delta})$
2. $O(dk^2 n \log \frac{k}{\delta})$.

- *non constant d?*
- *computing output-sensitive skylines with noise (even constant d)?*

Special thanks to: