On Obtaining Stable Ranking

ABOLFAZL ASUDEH
H. V. JAGADISH
GEROME MIKLAU
JULIA STOYANOVICH

VLDB 2018
Outline

➤ Motivation
➤ Problem setup
➤ Ranking algorithms
➤ Unbiased sampling
➤ Experimental results
Motivation
Motivation

➤ Rank items in a dataset

➤ Items have more than one attribute
  ➤ Combine values of multiple attributes into a scoring formula.

➤ Different ways of combining values for these attributes can lead to very different rankings.
  ➤ Significant consequences.
    ➤ Employee ranking: promotion/hiring
    ➤ University rankings: ranking formula has a significant effect on policies adopted by universities

➤ A robust method: small perturbations, such as small changes in parameter values, can’t change the rank order.
Motivation

➤ A linear combination of the attribute values is used for assigning a score to each item.

➤ Stability: captures the tolerance to changes in the weights.

➤ with respect to changes in the weights used in scoring function

➤ Each ranking can be produced by Many continuous scoring functions

➤ Larger region: higher stability
Motivation

➤ Producer (Data scientist)
  ➤ Justify their ranking methodology to gain the trust of consumers.

➤ Consumer
  ➤ Prioritize items and make decision
Acceptable Region

- Based on the domain knowledge of the producer:
  - Constraints defined by producer form a region of acceptable rankings

Stability:

Fraction of the acceptable region occupied by each ranking.
Contributions

➤ A novel notion of the stability of a ranking resulted from the linear weighting of item attribute values.

➤ Efficient stability test

➤ Efficient stability enumeration
Problem Setup
## Data Model And Ranking

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>a fixed database</td>
</tr>
<tr>
<td>n</td>
<td>number of items in D</td>
</tr>
<tr>
<td>t</td>
<td>an item in D</td>
</tr>
<tr>
<td>d</td>
<td>number of scalar scoring attributes of item</td>
</tr>
</tbody>
</table>
Data Model And Ranking

➤ t is a d length vector of attributes

\[ t \in D \rightarrow t = < t[1], t[2], ..., t[d] > \]

➤ More meaningful stability measure:

➤ scoring attributes:

➤ normalized (0,1)

➤ equivalent variance
**Definition 1 (Scoring Function).** A scoring function $f_{\vec{w}} : \mathbb{R}^d \rightarrow \mathbb{R}$, with weight vector $\vec{w} = \langle w_1, w_2, \ldots, w_d \rangle$, assigns the score $f_{\vec{w}}(t) = \sum_{j=1}^{d} w_j t[j]$ to any item $t \in D$.

$$w_j \in \vec{w} \geq 0$$

$f_{\vec{w}}(t)$ by $f(t)$

- $U$ : The set of all possible scoring functions.
- $\nabla_f(D)$ : the ranking of items in $D$ based on $f$. 
Geometry Of Ranked Items

➤ The original space
➤ The dual space
## Geometry Of Ranked Items

<table>
<thead>
<tr>
<th>id</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_1 + x_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_1$</td>
<td>0.63</td>
<td>0.71</td>
<td>1.34</td>
</tr>
<tr>
<td>$t_2$</td>
<td>0.83</td>
<td>0.65</td>
<td>1.48</td>
</tr>
<tr>
<td>$t_3$</td>
<td>0.58</td>
<td>0.78</td>
<td>1.36</td>
</tr>
<tr>
<td>$t_4$</td>
<td>0.70</td>
<td>0.68</td>
<td>1.38</td>
</tr>
<tr>
<td>$t_5$</td>
<td>0.53</td>
<td>0.82</td>
<td>1.35</td>
</tr>
</tbody>
</table>

\[
f(t) = x_1 + x_2.\]

\[
d = 2
\]

\[
n = 5
\]

\[
w = <1,1>\]
The Original Space

**Scoring function**: 
- A ray passing through the point of 
  \(<w[1], w[2], \ldots, w[d]>\)
- A set of d-1 angles : 
  \(\langle \theta_1, \theta_2, \ldots, \theta_{d-1} \rangle\)

**Ranking**: 
The projection of the points onto the vector of f(t) : 

- **further from the origin** 
- **higher the rank**
The Dual Space

**Dual hyperplane**

\[ d(t) : t[1] \times x_1 + \cdots + t[d] \times x_d = 1 \]

**Scoring function**:

- A ray passing through the point of \( \langle w[1], w[2], \ldots, w[d] \rangle \)
- A set of \( d-1 \) angles:
  \[ \langle \theta_1, \theta_2, \cdots, \theta_{d-1} \rangle \]

**Ranking**:

The intersection of \( d(t) \) and \( f(t) \):
- closer to the origin
- higher the rank
The Dual Space

Every point on $f(t)$ is a linear scaling of $\vec{w}$

$$t[1] \times a \times w_1 + \cdots + t[d] \times a \times w_d = 1$$

$$\sum t[j]w_j = 1/a$$

$$f(t) = 1/a$$

The intersection of $f(t)$ and $d(t)$:

$$a \times \vec{w}.$$ 

score of an item and intersection distance are in an inverse relationship.
Stability Of A Ranking

Rankings

R1
R2
R3
RN

Scoring functions

f1
f2
f3
fN
Stability Of A Ranking

\( \mathcal{R}_D : \)

the set of rankings over the items in D by at least one scoring function \( f \in U \)

for each ranking \( r \) : 
\( r \in \mathcal{R}_D : \)

\( R_D(r) : \)

Region of \( r \) : the set of functions that generate \( r \) :

\[
R_D(r) = \{ f \mid \nabla_f(D) = r \}
\]

\( \nabla_f(D) : \) ranking of items in D based on f
Stability Of A Ranking

➤ Example:

➤ Boundaries of the regions

➤ 11 feasible rankings
Stability Of A Ranking

Larger region for a ranking

Small changes in the weight vector are not likely to cross the boundary of a region.

More stable ranking
**Definition 2** (Stability of \( r \) at \( D \)). Given a ranking \( r \in \mathcal{R}_D \), the stability of \( r \) is the proportion of ranking functions in \( \mathcal{U} \) that generate \( r \). That is, stability is the ratio of the volume of the ranking region of \( r \) to the volume of \( \mathcal{U} \). Formally:

\[
S_D(r) = \frac{\text{vol}(R_D(r))}{\text{vol}(\mathcal{U})}
\]  

\( \text{vol}(R) \):  
The volume of a region is the area of the space carved out in the unit d-sphere by the set of functions in the region.
Acceptable Scoring Functions

➤ Producers can constrain the scoring function by specifying an acceptable region $U^* \subseteq U$

➤ **A vector and angle distance**

➤ a hypercone around the central ray defined by the weight vector.

➤ e.g: 95% cosine similarity to $f(t)$

➤ **A set of constraints**

➤ e.g: $w_2 \leq w_1$
Acceptable Scoring Functions

➤ New definition of stability:

\[ r \in \mathcal{R}^* : \text{an acceptable ranking} \]

\[ R^*(r) = \{ f \in \mathcal{U}^* | \nabla_f(D) = r \} \]

\[ S(r) = \frac{\text{vol}(R^*(r))}{\text{vol}(\mathcal{U}^*)} \]
Stability Problem Definitions

➤ Consumer’s stability problem
  ➤ Seeks to validate the stability of a given ranking

  Stability verification

➤ Producer’s stability problems
  ➤ Enumerate rankings and prioritizing them regarding to stability

  Stability Enumeration
Problem 1 (Stability verification). For dataset $\mathcal{D}$ with $n$ items over $d$ scoring attributes, and ranking $r \in \mathcal{R}$ of the items in $\mathcal{D}$, compute the ranking region $R_\mathcal{D}(r)$ and its stability $S_\mathcal{D}(r)$. 
Problem 2 (Batch Stable-Region Enumeration). For a dataset \( \mathcal{D} \) with \( n \) items over \( d \) scoring attributes, a region of interest \( \mathcal{U}^* \) (specified either by a set of constraints or by a vector-angle), and a stability threshold \( s \) (resp. a value \( h \)), find all rankings \( r \in \mathbb{R}^* \) such that \( S(r) \geq s \) (resp. the top-\( h \) stable rankings).

Problem 3 (Iterative Stable-Region Enumeration). For a dataset \( \mathcal{D} \) with \( n \) items over \( d \) scoring attributes, a region of interest \( \mathcal{U}^* \), specified either by a set of constraints or by a vector-angle, and the top-\((h - 1)\) stable rankings in \( \mathcal{U}^* \), discovered in the previous iterations of the problem, find the \( h \)-th stable ranking \( r \in \mathbb{R} \). That is, find:

\[
\arg\max_{r \in \mathbb{R} \setminus \text{top-}(h-1)} (S(r)) \quad (4)
\]
Ranking algorithms
Two Dimensional Ranking

\[ d(t) : t[1] \times x_1 + t[2] \times x_2 = 1 \]

Ordering exchange \( X(i,j) \):

- intersection of \( d(t_i) \) and \( d(t_j) \)

![Diagram showing intersection of lines]

\( d(t1) \)
\( d(t2) \)
\( d(t3) \)
Two Dimensional Ranking

Each regions identified by the two ordering exchanges that form its borders:

R : collection of rankings defined by the regions

|R|: the number of the regions
Two Dimensional Ranking

t1 and t2 do not dominate each other:

The polar coordinates of the intersection:

\[ \theta_{t,t'} = \arctan \left( \frac{t'[1] - t[1]}{t[2] - t'[2]} \right) \]
Stability Verification

t1 and t2 do not dominate each other:

If \( t[1] < t'[1] \)

\[
\frac{t[1]}{t[2]} > \frac{t'[1]}{t'[2]}
\]

\( t[1] \) has larger slope

If \( \theta < \theta_{t,t'} \)

\( t \) is ranked higher than \( t' \)
Stability Verification

\[ r = \langle t_3, t_1, t_2 \rangle \]
Stability Verification

**Algorithm 1** \(SV_{2D}\)

**Input:** Two dimensional dataset \(\mathcal{D}\) with \(n\) items and the ranking \(\tau\)

**Output:** The stability and the region of \(\tau\)

1: \((\theta_1, \theta_2) = (0, \pi/2)\)
2: for \(i = 1\) to \(n - 1\) do
3: \(t = \tau[i]; t' = \tau[i + 1]\)
4: if \(t\) dominates \(t'\) then continue
5: if \(t'\) dominates \(t\) then return null
6: \(\theta = \arctan \frac{t'[1] - t[1]}{t[2] - t'[2]}\)
7: if \((t[1] < t'[1] \text{ and } \theta > \theta_1)\) then \(\theta_1 = \theta\)
8: if \((t[1] > t'[1] \text{ and } \theta < \theta_2)\) then \(\theta_2 = \theta\)
9: if \(\theta_1 > \theta_2\) then return null
10: end for
11: return \(\frac{\theta_2 - \theta_1}{\pi/2}, (\theta_1, \theta_2)\)

**Complexity:** \(O(n)\)
1. $U^* = [U^*[1], U^*[2]]$

2. **RAYSWEEPING**: stores the regions in $U^*$, along with the stability of their rankings, in a heap data structure.

3. **GET-NEXT2D**: returns the most stable ranking $\nabla f (D)$, along with its stability from the heap.
Raysweeping

$r_1 = \langle t_1, t_2, t_3 \rangle, S_1$

$r_2 = \langle t_1, t_3, t_2 \rangle, S_2$

$r_3 = \langle t_3, t_1, t_2 \rangle, S_3$
Algorithm 1 RAYSWEEPING

Input: Two dimensional dataset $\mathcal{D}$ with $n$ items and the region of interest in the form of an angle range $[\mathcal{U}^*[1], \mathcal{U}^*[2]]$

Output: A heap of ranking regions and their stability

1: sweeper = new min-heap([\mathcal{U}^*[2]])
2: $\vec{w} = (\cos \mathcal{U}^*[1], \sin \mathcal{U}^*[1])$
3: $L = \nabla f(\mathcal{D})$
4: for $i = 1$ to $n - 1$ do
5:  $\theta = \arctan(L_{i+1}[1] - L_i[1]) / (L_i[2] - L_{i+1}[2])$
6:  if $\mathcal{U}^*[1] < \theta < \mathcal{U}^*[2]$ then sweeper.push((\theta, L_i, L_{i+1}))
7: end for
8: h = new max-heap(); $\theta_p = \mathcal{U}^*[1]$
9: while sweeper is not empty do
10:  $(\theta, t, t') = \text{sweeper.pop()}$
11:  $i, j = \text{index of } t, t' \text{ in } L$
12:  h.push($\theta_p - \theta_p \frac{\theta - \theta_p}{\mathcal{U}^*[2] - \mathcal{U}^*[1]}, (\theta_p, \theta)$)
13:  swap $L_i$ and $L_j$ and add the ordering exchanges between the new adjacent items to the sweeper
14:  $\theta_p = \theta$
15: end while
16: return h
**Algorithm 3** GET-\(NEXT_{2D}\)

**Input:** The heap of regions, with the top-\(h\) regions being removed from it

**Output:** the top-\(h + 1\) ranking, along with its stability

1. \(S, (\theta_1, \theta_2) = \text{heap.pop}()\)
2. \(w = (\cos \frac{\theta_1 + \theta_2}{2}, \sin \frac{\theta_1 + \theta_2}{2})\)
3. \(L = \nabla_f(D)\)
4. **return** \(L, S, (\theta_1, \theta_2)\)
Stability Enumeration

Complexity:

<table>
<thead>
<tr>
<th></th>
<th>RAYSWEEPING</th>
<th>GET-NEXT2D</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$O(n^2 \log n)$</td>
<td>first iteration: $O(n^2 \log n)$, other iterations: $O(n \log n)$</td>
</tr>
</tbody>
</table>

Store the ordered list $L$ for every region in RAYSWEEPING

GET-NEXT2D with order of $O(\log n)$ and memory cost of $O(n^3)$
Multi Dimensional (Md) Ranking

Every hyperplane \( h = X(t_i, t_j) \) partitions the function space \( U \) in two “half-spaces”

\[
X(t_i, t_j) = \sum_{k=1}^{d} (t_i[k] - t_j[k])x_k = 0
\]

- \( h^- : \sum_{k=1}^{d} (t_i[k] - t_j[k])x_k < 0 \): for the functions in \( h^- \), \( t_j \) outranks \( t_i \).
- \( h^+ : \sum_{k=1}^{d} (t_i[k] - t_j[k])x_k > 0 \): for the functions in \( h^+ \), \( t_i \) outranks \( t_j \).
Stability Verification In (Md) Ranking

➤ The intersection of half-spaces:
  open-ended d-dimensional cone (d-cone) with base of a (d−1) dimensional convex polyhedron.

➤ The region of ranking is a polyhedron:
  ➤ computing its volume is #P-hard
    ➤ numeric methods and sampling for estimating this quantity
Stability Oracle

➤ A stability oracle $S(R, U^*)$:

➤ Given a convex region $R$ in the form of an intersection of half-spaces and a region of interest

➤ returns the stability of $R$ in $U^*$

➤ Monte-Carlo:

➤ Uniform sampling

---

Algorithm 12 $S$ // the stability oracle

**Input:** A set $R$ of halfspaces defining a region and a set $S$ of unbiased samples taken from $U^*$

**Output:** The stability of $R$ in $U^*$

1: count = 0
2: for sample $w$ in $S$ do
3:   flag = true
4:   for halfspace $(h, \text{sign})$ in $R$ do
5:     sum = $\sum_{i=1}^{d} h[i]w_i$
6:     if (sign is + and sum < 0) or (sign is - and sum > 0) then
7:       flag = false; break
8:     end if
9:   end for
10: if flag = true then count = count + 1
11: end for
12: return $\frac{\text{count}}{|S|}$
Stability Enumeration (Md) Ranking

3D Ranking region definition

➤ Arrangement of ordering exchange hyperplanes:

➤ Connected convex d-cones (angle in 2D): The set of ordering exchanges intersecting $U^*$ each showing a ranking region.

**Theorem 1.** Every ranking $\mathbf{v} \in \mathcal{R}^*$ is provided by the functions in exactly one convex region in the arrangement of ordering exchange hyperplanes in $U^*$. 
Stability Enumeration (Md) Ranking

- **2D**: The number of ranking regions is bounded by \( O(n^2) \)
- **3D**: The arrangement can contain \( O(n^{2d}) \) regions.

No need to construct the full arrangement:

- Objective: finding stable rankings, not all possible rankings
- User satisfaction: observing a few rankings we propose

Idea:

- Construction of only the next stable ranking
- Delays the construction of other rankings
Get-NextMD

- Arrangement construction is an iterative process
  - Breaks down the largest region at every iteration
  - Delaying the construction of the arrangement in all other regions.
- “region” data structure: record each region
  - C: the set of half-spaces defining the regions
  - S: the stability of the region
  - pending: the index of the next hyperplane to be added to the region
Randomized Get-Next

- Worst case: all regions are equally stable
  - order: $O(n^{2d})$

- Randomized algorithm:
  - Does not depend on the arrangement construction
  - Does not suffer its high complexity
Randomized Get-Next

➤ The user is interested in the head (the top-k)
  ➤ Randomized get-Next:
    ➤ Acalable for large settings
    ➤ Applicable for enumerating the top-k items
➤ Every ranking is generated by continuous ranges of functions
  ➤ Every function f generates only one ranking of items
  ➤ The larger the volume of a ranking region, the higher the chance of choosing a random function from it.
  ➤ Uniform sampling of the function space allows sampling of rankings based on their stability distribution.
Randomized Get-Next

Existence of a suitable sampler

Monte-Carlo methods for Bernoulli random variables
Randomized Get-Next

- also valid for top-k
- law of large numbers
- **sampling** for:
  - Discovering the rankings
  - Estimating stability of discovered ranking
- Randomized get-next:
  - Specify the sampling budget (fixed sample number)
    - Fixed The running time
    - Variable error
  - The confidence interval (fixed error)
    - Guarantees of output quality
    - Non deterministic running time
Randomized Get-Next

- Each ranking $r \in R$ : a bernoulli random variable

- Consider the distribution of drawing a function that generate it.

- The probability mass function of this distribution:

\[
p(\Theta; S(r)) = \begin{cases} 
S(r) & \nabla_{f(\Theta)}(D) = r \\
1 - S(r) & \nabla_{f(\Theta)}(D) \neq r
\end{cases}
\]

\[
\mu_r = S(r)
\]

\[
\sigma_r = S(r)(1 - S(r)).
\]
Randomized Get-Next

\[ m_r \quad : \text{the average of a set of } N \text{ samples of the random variable} \]

\[ E[m_r] = S(r) \]

\[ s_r = m_r(1 - m_r) \]

Central limit theorem:
\[ N\left(\mu_r, \frac{\sigma_r}{\sqrt{N}}\right) \]

For a large \( N \) : estimate standard deviation with estimated mean
Randomized Get-Next

➤ Confidence interval :
  ➤ Confidence level : $\alpha$,
  ➤ Confidence error : $e$ 
    $[m_r - e, m_r + e]$

$$p(m_r - e \leq \mu_r \leq m_r + e) = 1 - \alpha$$

$$e = Z(1 - \frac{\alpha}{2}) \frac{s_r}{\sqrt{N}} = Z(1 - \frac{\alpha}{2}) \sqrt{\frac{m_r(1 - m_r)}{N}}$$
Randomized Get-Next

Algorithm 2 GET-NEXT$_r$

Input: $\mathcal{D}$, $\mathcal{U}^*$, previous stable rankings $\mathcal{R}_{h-1}$, hash of previous aggregates $cnts$, total number of previous samples $N'$, confidence level $\alpha$, and sampling budget $N$

Output: Next stable ranking and its stability measures

1: for $k = 1$ to $N$ do
2: $w = \text{Sample}\mathcal{U}^*(\mathcal{U}^*)$
3: $r = \nabla_f(\mathcal{D})$
4: if $r$ is in $cnts.\text{keys}$ then $cnts[r] += 1$ else $cnts[r] = 1$
5: end for
6: if $cnts.\text{keys}\setminus\mathcal{R}_{h-1} = \emptyset$ then return null
7: $r_h = \underset{r \in cnts.\text{keys}\setminus\mathcal{R}_{h-1}}{\text{argmax}} (cnts[r])$
8: $S(r_h) = \frac{cnts[r_h]}{N+N'}$; $e = Z(1 - \frac{\alpha}{2}) \sqrt{\frac{S(r_h)(1-S(r_h))}{N+N'}}$
9: return $r_h, S(r_h), e$
Randomized Get-Next

➤ In addition to the N new samples, it uses the aggregates of its previous runs.

➤ Running time: $O(N \times n \log n)$
STABLE TOP-K ITEMS

- A company interview: top-k candidates
- A student that wants to apply for the top colleges: the top-k ranking of the colleges
- Different ranking regions may share the same top-k items:
  - MD algorithm not applicable here
  - Randomized Get-next can work
UNBIASED FUNCTION
SAMPLING
Monte-Carlo needs Uniform sampler from the function space:

Generating angle vectors at random:

No uniform random functions sampled from the function space, except for 2D
Sampling From The Function Space

➤ One-to-one mapping between the points on the surface of the unit d-sphere and the unit origin-starting rays

➤ The normal distribution function has a constant probability on the surfaces of d-spheres with common centers.

➤ Samples from the surface of a d-sphere using the normal distribution, and normalizes them.
Unbiased Function Sampling

➤ Generate a random function in $U$:

$$w_i = |\mathcal{N}(0, 1)|.$$
Sampling From The Region Of Interest

➤ Draw samples from U*:

1. An acceptance-rejection method:
   ➤ It is efficient if the volume of U* is not small

2. Inverse CFD (when U* is small relative to U)
Sampling From The Region Of Interest

Inverse CDF

- One-to-one mapping between the rays in $U^*$ and the points on the surface unit hyper-spherical cap

\[ \text{Cap :} \]

the intersection of:

- The set of the (d-th axis orthogonal) planes ((d−1)-spheres)
  with

- The d-sphere

\[ \cos \theta \leq x_d \leq 1 \]
The area of the unit \( d \)-spherical cap:

the integral over the surface areas of the \((d - 1)\)-spheres, defined by the intersection of the planes \( \cos \theta \leq xd \leq 1 \) with the \( d \)-sphere

The surface area of a \( \delta \)-sphere with the radius \( r \):

\[
A_{\delta}(r) = \frac{2\pi^{\delta/2}}{\Gamma(\delta/2)} r^{\delta-1}
\]

\[
A_{d}^{cap}(1) = \int_{0}^{\theta} A_{d-1} \sin \phi d\phi = \frac{2\pi^{d/2}}{\Gamma(d/2)} \int_{0}^{\theta} \sin^{d-2}(\phi) d\phi
\]

\[
F(x) = \frac{\int_{0}^{x} \sin^{d-2}(\phi) d\phi}{\int_{0}^{\theta} \sin^{d-2}(\phi) d\phi}
\]
Sampling From The Region Of Interest

Inverse CDF

➤ $d = 3$:

$$F(x) = \frac{1 - \cos x}{1 - \cos \theta} \Rightarrow F^{-1}(x) = \arccos \left( 1 - (1 - \cos \theta)x \right)$$

➤ For a general $d$ : numerical methods for finding the integrals

➤ Riemann sums for

$$F(x) = \frac{\int_0^x \sin^{d-2}(\phi) \, d\phi}{\int_0^\theta \sin^{d-2}(\phi) \, d\phi}$$
Algorithm 3 Sample $\mathcal{U}^*$

**Input:** The ray $\rho$, angle $\theta$

1. $y = U[0, 1]$ // draw a uniform sample in range $[0, 1]$
2. $x = F^{-1}(y)$
3. for $i = 1$ to $d - 1$ do $\hat{\omega}_i = \mathcal{N}(0, 1)$
4. $\langle \theta_1, \cdots, \theta_{d-2} \rangle = \text{the angles in polar representation of } \hat{\omega}$
5. $w = \text{toCartesian}(1, \langle \theta_1, \cdots, \theta_{d-2}, x \rangle)$
6. return $\text{Rotate}(w, \rho)$
Experimental Results
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Dimensionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSMetrics</td>
<td>2</td>
</tr>
<tr>
<td>FIFA</td>
<td>4</td>
</tr>
<tr>
<td>Blue Nile</td>
<td>5</td>
</tr>
<tr>
<td>Depart of Transportation</td>
<td>3</td>
</tr>
</tbody>
</table>
Stability Investigation

**CSMetrics 2D**

**FIFA 4D**

0.998 cosine sim

0.999 cosine sim
Algorithm Performance-2D

Stability Verification

Get-next
Algorithm Performance-Md

Stability Verification

![Graph showing time (sec) vs. stability and number of items (n)]
Getnext-Md

Impact of dataset size

Impact of number of attributes

Impact of width of region of interest
Randomized Get-Next (Top-K)
The Effect Of Attribute Correlation
Any Question?

Thank you :)

73