Flexible Aggregate Similarity Search

Yang Li$^1$, Feifei Li$^2$, Ke Yi$^3$, Bin Yao$^2$, Min Wang$^4$

$^1$Department of Computer Science and Engineering
Shanghai Jiao Tong University, China

$^2$Department of Computer Science
Florida State University, USA

$^3$Department of Computer Science
Hong Kong University of Science and Technology

$^4$HP Labs China
Outline

1 Motivation and Problem Formulation

2 Basic Aggregate Similarity Search

3 Flexible Aggregate Similarity Search

4 Experiments
Similarity search (aka nearest neighbor search, NN search) is a fundamental tool in retrieving the most relevant data w.r.t. user input in working with massive data: extensively studied.
However, often time, users may be interested at retrieving objects that are similar to a group $Q$ of query objects, instead of just one.
However, often time, users may be interested at retrieving objects that are similar to a group $Q$ of query objects, instead of just one.

Given an aggregation $\sigma$, a similarity/distance function $d$, a dataset $P$, and any query group $Q$:

$$r_p = \sigma\{d(p, Q)\} = \sigma\{d(p, q_1), \ldots, d(p, q_{|Q|})\}, \text{ for any } p$$

aggregate similarity distance of $p$
However, often time, users may be interested at retrieving objects that are *similar* to a group $Q$ of query objects, *instead of just one*.

Given an aggregation $\sigma$, a similarity/distance function $d$, a dataset $P$, and any query group $Q$:

$$r_p = \sigma\{d(p, Q)\} = \sigma\{d(p, q_1), \ldots, d(p, q_{|Q|})\}, \text{ for any } p$$

Aggregate similarity distance of $p$

Find $p^* \in P$ having the smallest $r_p$ value ($r_{p^*} = r^*$).
However, often time, users may be interested at retrieving objects that are similar to a group $Q$ of query objects, instead of just one.

Figure: Aggregate similarity search in Euclidean space: max and sum.

- $\times$: group $Q$ of query points
- $\bullet$: dataset $P$
However, often time, users may be interested at retrieving objects that are similar to a group $Q$ of query objects, instead of just one.

$$\text{agg} = \text{max}, \ p^* = p_3, \ r^* = d(p_3, q_1)$$

$\times$ : group $Q$ of query points

$\bullet$ : dataset $P$

Figure: Aggregate similarity search in Euclidean space: max and sum.
However, often time, users may be interested in retrieving objects that are similar to a group $Q$ of query objects, instead of just one.

$$\text{agg} = \text{sum}, \ p^* = p_4, \ r^* = \sum_{q \in Q} d(p_4, q)$$

$x$ : group $Q$ of query points  
$ullet$ : dataset $P$

**Figure**: Aggregate similarity search in Euclidean space: max and sum.
Introduction and motivation

- However, often time, users may be interested at retrieving objects that are similar to a group $Q$ of query objects, instead of just one.
- Aggregate similarity search ($\text{ANN}$) may need to deal with data in high dimensions.
Outline

1. Motivation and Problem Formulation

2. Basic Aggregate Similarity Search

3. Flexible Aggregate Similarity Search

4. Experiments
Existing methods for ANN

- R-tree method: branch and bound principle [PSTM04, PTMH05].
- Some other heuristics to further improve the pruning.
- Can be extended to other metric space using M-tree [RBTFT08].
- Limitations:
  - No bound on the query cost.
  - Query cost increases quickly as dataset becomes larger and/or dimension goes higher.


Our approach for $\sigma = \max: \text{AMAX1}$

- We proposed $\text{AMAX1}$ (TKDE’10):

$\text{B}(c, r_c)$ is a ball centered at $c$ with radius $r_c$; $\text{MEB}(Q)$ is the minimum enclosing ball of a set of points $Q$; $\text{nn}(c, P)$ is the nearest neighbor of a point $c$ from the dataset $P$.

$\times$ : group $Q$ of query points
$\bullet$ : dataset $P$
Our approach for $\sigma = \max: A_{\text{MAX}1}$

- We proposed $A_{\text{MAX}1}$ (TKDE'10):
  - $B(c, r_c)$ is a ball centered at $c$ with radius $r_c$;
  - $\text{MEB}(Q)$ is the minimum enclosing ball of a set of points $Q$;

1. $B(c, r_c) = \text{MEB}(Q)$

$x$: group $Q$ of query points

$\bullet$: dataset $P$
Our approach for $\sigma = \max$: $A_{\text{MAX1}}$

- We proposed $A_{\text{MAX1}}$ (TKDE’10):
  - $B(c, r_c)$ is a ball centered at $c$ with radius $r_c$;
  - $\text{MEB}(Q)$ is the minimum enclosing ball of a set of points $Q$;
  - $\text{nn}(c, P)$ is the nearest neighbor of a point $c$ from the dataset $P$.

1. $B(c, r_c) = \text{MEB}(Q)$
2. return $p = \text{nn}(c, P)$

- $\times$: group $Q$ of query points
- $\bullet$: dataset $P$
Our approach for $\sigma = \max$: $\text{MAX}_1$

- An algorithm returns $(p, r_p)$ for $\text{ANN}(Q, P)$ is an $c$-approximation iff $r^* \leq r_p \leq c \cdot r_p$. 

\[\text{Theorem}\]

In any dimension $d$, given an $\alpha$-approximate $\text{MEB}$ algorithm and an $\beta$-approximate $\text{NN}$ algorithm, $\text{MAX}_1$ is an $\sqrt{\alpha^2 + \beta^2}$-approximation.

---


Our approach for $\sigma = \max$: $A_{\text{MAX}1}$

- An algorithm returns $(p, r_p)$ for $A_{\text{NN}}(Q, P)$ is an $c$-approximation iff $r^* \leq r_p \leq c \cdot r_p$.

Theorem

$A_{\text{MAX}1}$ is a $\sqrt{2}$-approximation in any dimension $d$ given (exact) $\text{nn}(c, P)$ and $\text{MEB}(Q)$. 

\[ \text{[AMNSW98]}: \text{An Optimal Algorithm for Approximate Nearest Neighbor Searching in Fixed Dimensions. In JACM, 1998.} \]
\[ \text{[TYSK10]}: \text{Efficient and Accurate Nearest Neighbor and Closest Pair Search in High Dimensional Space. In TODS, 2010.} \]
\[ \text{[KMY03]}: \text{Approximate Minimum Enclosing Balls in High Dimensions Using Core-Sets. In JEA, 2003.} \]
Our approach for $\sigma = \max$: $A_{\text{MAX1}}$

- An algorithm returns $(p, r_p)$ for $\text{ANN}(Q, P)$ is an $c$-approximation iff $r^* \leq r_p \leq c \cdot r_p$.

Theorem

$A_{\text{MAX1}}$ is a $\sqrt{2}$-approximation in any dimension $d$ given (exact) $\text{nn}(c, P)$ and $\text{MEB}(Q)$.

Theorem

In any dimension $d$, given an $\alpha$-approximate MEB algorithm and an $\beta$-approximate NN algorithm, $A_{\text{MAX1}}$ is an $\sqrt{\alpha^2 + \beta^2}$-approximation.
Our approach for $\sigma = \max$: $A_{\text{MAX1}}$

- An algorithm returns $(p, r_p)$ for $\text{ANN}(Q, P)$ is an $c$-approximation iff $r^* \leq r_p \leq c \cdot r_p$.
- In low dimensions, BBD-tree [AMNSW98] gives $(1 + \epsilon)$-approximate NN search; in high dimensions, LSB-tree [TYSK10] gives $(2 + \epsilon)$-approximate NN search with high probability; and $(1 + \epsilon) - \text{MEB}$ algorithm exists even in high dimensions [KMY03].

**Theorem**

$A_{\text{MAX1}}$ is a $\sqrt{2}$-approximation in any dimension $d$ given (exact) $\text{nn}(c, P)$ and $\text{MEB}(Q)$.

**Theorem**

In any dimension $d$, given an $\alpha$-approximate $\text{MEB}$ algorithm and an $\beta$-approximate NN algorithm, $A_{\text{MAX1}}$ is an $\sqrt{\alpha^2 + \beta^2}$-approximation.

Our approach for $\sigma = \text{sum}$: \textbf{Asum1}

- We proposed \textbf{Asum1} (TKDE’10):

\[ g_m \] be the geometric median of \( Q \);
\[ \text{return } 
\begin{align*}
nn(g_m, P)\end{align*} \]

\[ : \text{dataset } P \]
\[ : \text{group } Q \text{ of query points} \]
\[ : \text{dataset } P \]
Our approach for $\sigma = \text{sum}$: \textbf{Asum1}

- We proposed \textbf{Asum1} (TKDE’10):
  - let $g_m$ be the geometric median of $Q$;

1. $g_m$ is the geometric median of $Q$

$\times$ : group $Q$ of query points

$\bullet$ : dataset $P$
Our approach for \( \sigma = \text{sum} \): **Asum1**

- We proposed \textbf{Asum1} (TKDE’10):
  - let \( g_m \) be the geometric median of \( Q \);
  - return \( \text{nn}(g_m, P) \).

1. \( g_m \) is the geometric median of \( Q \)
2. return \( \text{nn}(g_m, P) \)

\( \times \) : group \( Q \) of query points  
\( \bullet \) : dataset \( P \)
Our approach for $\sigma = \text{sum}$: \texttt{Asum1}

- Using the Weiszfeld algorithm (iteratively re-weighted least squares), $g_m$ can be computed to an arbitrary precision efficiently.

**Theorem**

\texttt{Asum1} is a 3-approximation in any dimension $d$ given (exact) geometric median and $\text{nn}(c, P)$.
Our approach for $\sigma = \text{sum}$: $\text{Asum1}$

- Using the Weiszfeld algorithm (iteratively re-weighted least squares), $g_m$ can be computed to an arbitrary precision efficiently.

Theorem

$\text{Asum1}$ is a 3-approximation in any dimension $d$ given (exact) geometric median and $\text{nn}(c, P)$.

Theorem

In any dimension $d$, given an $\beta$-approximate NN algorithm, $\text{Asum1}$ is an $3\beta$-approximation.
Our approach for $\sigma = \text{sum}$: $\text{Asum1}$

- Using the Weiszfeld algorithm (iteratively re-weighted least squares), $g_m$ can be computed to an arbitrary precision efficiently.
- Both $\text{Amax1}$ and $\text{Asum1}$ can be easily extended to work for $k\text{ANN}$ search while the bounds are maintained.

**Theorem**

$\text{Asum1}$ is a 3-approximation in any dimension $d$ given (exact) geometric median and $\text{nn}(c, P)$.

**Theorem**

In any dimension $d$, given an $\beta$-approximate NN algorithm, $\text{Asum1}$ is an $3\beta$-approximation.
Outline

1. Motivation and Problem Formulation
2. Basic Aggregate Similarity Search
3. Flexible Aggregate Similarity Search
4. Experiments
Definition of flexible aggregate similarity search

Flexible aggregate similarity search ($FANN$): given support $\phi \in (0, 1]$ and find an object in $P$ that has the best aggregate similarity to (any) $\phi \mid Q \mid$ query objects (our work in SIGMOD’11).
Definition of flexible aggregate similarity search

- Flexible aggregate similarity search (FANN): given support $\phi \in (0, 1]$ and find an object in $P$ that has the best aggregate similarity to (any) $\phi|Q|$ query objects (our work in SIGMOD'11).

$$\sigma = \max, \phi = 40\%, p^* = p_4, r^* = d(p_4, q_3)$$

$\times$: group $Q$ of query points

$\bullet$: dataset $P$

Figure: FANN in Euclidean space: max, $\phi = 0.4$. 
For $\forall p \in P$, $r_p = \sigma(p, Q^p_\phi)$, where $Q^p_\phi$ is $p$'s $\phi|Q|$ NNs in $Q$. 
For $\forall p \in P$, $r_p = \sigma(p, Q^p_\phi)$, where $Q^p_\phi$ is $p$'s $\phi|Q|$ NNs in $Q$.

$Q^p_\phi$ : top $\phi|Q|$ NNs of $p$ in $Q$

\[ \phi = 0.4, \ |Q| = 5, \ \phi|Q| = 2 \]
Exact methods for FANN

- For $\forall p \in P$, $r_p = \sigma(p, Q_p^\phi)$, where $Q_p^\phi$ is $p$'s $\phi|Q|$ NNs in $Q$.
- R-tree method, with the branch and bound principle, can still be applied based on this observation.
- In high dimensions, take the brute-force-search (BFS) approach:
  - For each $p \in P$, find out $Q_p^\phi$ and calculate $r_p$. 
Approximate methods for $\sigma = \text{sum}$: \textbf{Asum}

$x$: group $Q$ of query points

$\bullet$: dataset $P$

$\phi = 0.4$, $|Q| = 5$, $\phi|Q| = 2$, $\sigma = \text{sum}$
Approximate methods for $\sigma = \text{sum}$: **Asum**

$p_2 = \text{nn}(q_1, P)$

\(q_1\) \(\times\) : group \(Q\) of query points

\(\bullet\) : dataset \(P\)

\(\phi = 0.4, \ |Q| = 5, \ \phi|Q| = 2, \ \sigma = \text{sum}\)
Approximate methods for $\sigma = \text{sum}$: \textit{Asum}

$Q^P_\phi$ : top $\phi|Q|$ NNs of $p$ in $Q$

$p_2 = \text{nn}(q_1, P)$

$\times$ : group $Q$ of query points

$\bullet$ : dataset $P$

$\phi = 0.4$, $|Q| = 5$, $\phi|Q| = 2$, $\sigma = \text{sum}$
Approximate methods for $\sigma = \text{sum}$: \textbf{Asum}

$Q^P_\phi$: top $\phi|Q|$ NNs of $p$ in $Q$

\[ p_2 = \text{nn}(q_1, P) \]

\[ r_{p_2} = d(p_2, q_1) + d(p_2, q_2) \]

$\times$: group $Q$ of query points

$\bullet$: dataset $P$

$\phi = 0.4$, $|Q| = 5$, $\phi|Q| = 2$, $\sigma = \text{sum}$

- Repeat this for every $q_i \in Q$, return the $p$ with the smallest $r_p$. 
Approximation quality of Asum

Theorem

In any dimension $d$, given an exact NN algorithm, Asum is an 3-approximation.

Theorem

In any dimension $d$, given an $\beta$-approximate NN algorithm, Asum is an $(\beta + 2)$-approximation.
Approximation quality of Asum

Theorem

In any dimension $d$, given an exact NN algorithm, Asum is an 3-approximation.

Theorem

In any dimension $d$, given an $\beta$-approximate NN algorithm, Asum is an $(\beta + 2)$-approximation.

- Asum only needs $|Q|$ times of NN search in $P$. ...
Approximation quality of \texttt{Asum}

\textbf{Theorem}

\textit{In any dimension $d$, given an exact NN algorithm, \texttt{Asum} is an 3-approximation.}

\textbf{Theorem}

\textit{In any dimension $d$, given an $\beta$-approximate NN algorithm, \texttt{Asum} is an $\left(\beta + 2\right)$-approximation.}

- \texttt{Asum} only needs $|Q|$ times of NN search in $P$...
- \texttt{Asum} still needs $|Q|$ times of NN search in $P$!
An improvement to ASUM

randomly select a subset of $Q$

$x$ : group $Q$ of query points

$\bullet$ : dataset $P$
randomly select a subset of $Q$!

$\times$: group $Q$ of query points

$\bullet$: dataset $P$

Theorem

For any $0 < \varepsilon, \lambda < 1$, executing Asum algorithm only on a random subset of $f(\phi, \varepsilon, \lambda)$ points of $Q$ returns a $(3 + \varepsilon)$-approximate answer to Fann search in any dimensions with probability at least $1 - \lambda$, where

$$f(\phi, \varepsilon, \lambda) = \frac{\log \lambda}{\log(1 - \phi \varepsilon / 3)} = O(\log(1/\lambda) / \phi \varepsilon).$$
An improvement to **Asum**

randomly select a subset of $Q$!

$x$: group $Q$ of query points

$\bullet$: dataset $P$

---

**Theorem**

*For any $0 < \varepsilon, \lambda < 1$, executing Asum algorithm only on a random subset of $f(\phi, \varepsilon, \lambda)$ points of $Q$ returns a $(3 + \varepsilon)$-approximate answer to FANN search in any dimensions with probability at least $1 - \lambda$, where*

$$f(\phi, \varepsilon, \lambda) = \frac{\log \lambda}{\log(1 - \phi \varepsilon / 3)} = O(\log(1/\lambda)/\phi \varepsilon).$$

- For $|Q| = 1000$, $\phi = 0.4$, $\lambda = 10\%$, $\varepsilon = 0.5$, only needs 33 NN search in any dimension.
An improvement to **Asum**

randomly select a subset of $Q!$

$x$: group $Q$ of query points

$\bullet$: dataset $P$

**Theorem**

*For any $0 < \varepsilon, \lambda < 1$, executing **Asum** algorithm only on a random subset of $f(\phi, \varepsilon, \lambda)$ points of $Q$ returns a $(3 + \varepsilon)$-approximate answer to **Fann** search in any dimensions with probability at least $1 - \lambda$, where*

$$f(\phi, \varepsilon, \lambda) = \frac{\log \lambda}{\log(1 - \phi \varepsilon / 3)} = O(\log(1/\lambda)/\phi \varepsilon).$$

- For $|Q| = 1000$, $\phi = 0.4$, $\lambda = 10\%$, $\varepsilon = 0.5$, only needs 33 NN search in any dimension. (much less in practice, $\frac{1}{\phi}$ is enough!)
An improvement to Asum

```
randomly select a subset of Q!

X: group Q of query points
•: dataset P
```

**Theorem**

For any $0 < \varepsilon, \lambda < 1$, executing Asum algorithm only on a random subset of $f(\phi, \varepsilon, \lambda)$ points of Q returns a $(3 + \varepsilon)$-approximate answer to FANN search in any dimensions with probability at least $1 - \lambda$, where

$$f(\phi, \varepsilon, \lambda) = \frac{\log \lambda}{\log(1 - \phi\varepsilon/3)} = O(\log(1/\lambda)/\phi\varepsilon).$$

- For $|Q| = 1000, \phi = 0.4, \lambda = 10\%, \varepsilon = 0.5$, only needs 33 NN search in any dimension. (much less in practice, $\frac{1}{\phi}$ is enough!)
- Independent of dimensionality, $|P|$, and $|Q|$!
Approximate methods for σ = max: \textbf{A\text{MAX}}

$\phi = 0.4$, $|Q| = 5$, $\phi|Q| = 2$, $\sigma = \text{max}$

\textbf{x} : group $Q$ of query points

\textbf{•} : dataset $P$

$q_1$
Approximate methods for \( \sigma = \text{max} \): \text{A\textsc{MAX}}

\[ Q^q_\phi : \text{top } \phi |Q| \text{ NNs of } q \text{ in } Q, \text{ including } q \]

\( \times \) : group \( Q \) of query points
\( \bullet \) : dataset \( P \)
\( \phi = 0.4, \ |Q| = 5, \ \phi |Q| = 2, \ \sigma = \text{max} \)
Approximate methods for $\sigma = \max$: $A_{\text{MAX}}$

$Q^q_{\phi}$: top $\phi|Q|$ NNs of $q$ in $Q$, including $q$

- $\text{MEB}(Q^q_{\phi}) = \text{MEB}\{q_1, q_2\}$

$\times$: group $Q$ of query points

$\bullet$: dataset $P$

$\phi = 0.4$, $|Q| = 5$, $\phi|Q| = 2$, $\sigma = \max$

Yang Li, Feifei Li, Ke Yi, Bin Yao, Min Wang
Approximate methods for $\sigma = \max$: $A_{\text{MAX}}$

$Q^q_\phi$: top $\phi|Q|$ NNs of $q$ in $Q$, including $q$

- $\text{MEB}(Q^q_\phi) = \text{MEB} \{q_1, q_2\}$

$p_3 = \text{nn}(c_1, P)$

\[\text{x} : \text{group } Q \text{ of query points}\]

\[\text{\bullet : dataset } P\]

$\phi = 0.4$, $|Q| = 5$, $\phi|Q| = 2$, $\sigma = \max$
Approximate methods for \( \sigma = \max \): \textit{Amax}

\[ Q^{p_3}_\phi : \text{top } \phi|Q| \text{ NNs of } p_3 \text{ in } Q \]

\[ p_3 = \text{nn}(c_1, P) \]

\( \times \): group \( Q \) of query points

\( \bullet \): dataset \( P \)

\( \phi = 0.4, |Q| = 5, \phi|Q| = 2, \sigma = \max \)
Approximate methods for $\sigma = \max$: A\text{MAX}

$Q^{p_3}_\phi$: top $\phi|Q|$ NNs of $p_3$ in $Q$

$\times$: group $Q$ of query points

$\bullet$: dataset $P$

$\phi = 0.4$, $|Q| = 5$, $\phi|Q| = 2$, $\sigma = \max$
Approximate methods for $\sigma = \max$: $A_{\text{MAX}}$

$Q_{\phi}^{p_3}$ : top $\phi|Q|$ NNs of $p_3$ in $Q$

- $p_3 = \text{nn}(c_1, P)$

$x$ : group $Q$ of query points

$\bullet$ : dataset $P$

$\phi = 0.4, \ |Q| = 5, \ \phi|Q| = 2, \ \sigma = \max$

Repeat this for every $q_i \in Q$, return the $p$ with the smallest $r_p$. 
Approximate methods for $\sigma = \max$: $\textbf{A}_{\text{MAX}}$

$Q^P_{p_3}$: top $\phi|Q|$ NNs of $p_3$ in $Q$

$\times$: group $Q$ of query points

$\bullet$: dataset $P$

$\phi = 0.4$, $|Q| = 5$, $\phi|Q| = 2$, $\sigma = \max$

- Repeat this for every $q_i \in Q$, return the $p$ with the smallest $r_p$.
- Identical to $\textbf{A}_{\text{SUM}}$, except using $p = \text{nn}(c_i, P)$ instead of $p = \text{nn}(q_i, P)$, where $c_i$ is the center of $\text{MEB}(q_i, Q^q_{\phi})$. 

Yang Li, Feifei Li, Ke Yi, Bin Yao, Min Wang

Flexible Aggregate Similarity Search
Approximation quality of $A_{\text{MAX}}$

Theorem

In any dimension $d$, given an exact NN algorithm, $A_{\text{MAX}}$ is an $(1 + 2\sqrt{2})$-approximation.

Theorem

In any dimension $d$, given an $\beta$-approximate NN algorithm, $A_{\text{MAX}}$ is an $((1 + 2\sqrt{2})\beta)$-approximation.
Approximation quality of \textsc{Amax}

\textbf{Theorem}

\textit{In any dimension }d\textit{, given an exact NN algorithm, \textsc{Amax} is an }\left(1 + 2\sqrt{2}\right)\text{-approximation.}

\textbf{Theorem}

\textit{In any dimension }d\textit{, given an }\beta\text{-approximate NN algorithm, \textsc{Amax} is an }\left((1 + 2\sqrt{2})\beta\right)\text{-approximation.}

- \textsc{Amax} only needs \(|Q|\) times of MEB and \(|Q|\) NN search in }P\ldots
Approximation quality of $A_{\text{MAX}}$

**Theorem**

*In any dimension $d$, given an exact NN algorithm, $A_{\text{MAX}}$ is an $(1 + 2\sqrt{2})$-approximation.*

**Theorem**

*In any dimension $d$, given an $\beta$-approximate NN algorithm, $A_{\text{MAX}}$ is an $((1 + 2\sqrt{2})\beta)$-approximation.*

- $A_{\text{MAX}}$ only needs $|Q|$ times of MEB and $|Q|$ NN search in $P$...
- $A_{\text{MAX}}$ still needs $|Q|$ times of MEB and $|Q|$ NN search in $P$!
An improvement to Amax

randomly select a subset of $Q$!

$\times$ : group $Q$ of query points
$\bullet$ : dataset $P$

Theorem

For any $0 < \lambda < 1$, executing Amax algorithm only on a random subset of $f(\phi, \lambda)$ points of $Q$ returns a $(1 + 2\sqrt{2})$-approximate answer to the Fann query with probability at least $1 - \lambda$ in any dimensions, where $f(\phi, \lambda) = \log \frac{\lambda}{\log(1 - \phi)} = O\left(\frac{\log(1/\lambda)}{\phi}\right)$.

For $|Q| = 1000$, $\phi = 0.4$, $\lambda = 10\%$, only needs 5 MEB and NN search in any dimension.

Independent of dimensionality, $|P|$, and $|Q|$!
An improvement to **Amax**

randomly select a subset of $Q$!

![Diagram: dots represent dataset $P$, crosses represent group $Q$ of query points]

**Theorem**

For any $0 < \lambda < 1$, executing **Amax** algorithm only on a random subset of $f(\phi, \lambda)$ points of $Q$ returns a $(1 + 2\sqrt{2})$-approximate answer to the **Fann** query with probability at least $1 - \lambda$ in any dimensions, where

$$f(\phi, \lambda) = \frac{\log \lambda}{\log(1 - \phi)} = O(\log(1/\lambda)/\phi).$$
An improvement to \texttt{Amax}

randomly select a subset of $Q$!

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figures/dataset.png}
\caption{Group $Q$ of query points and dataset $P$.}
\end{figure}

$\times$ : group $Q$ of query points

$\bullet$ : dataset $P$

\textbf{Theorem}

\textit{For any $0 < \lambda < 1$, executing \texttt{Amax} algorithm only on a random subset of $f(\phi, \lambda)$ points of $Q$ returns a $(1 + 2\sqrt{2})$-approximate answer to the \texttt{Fann} query with probability at least $1 - \lambda$ in any dimensions, where}

\[ f(\phi, \lambda) = \frac{\log \lambda}{\log(1 - \phi)} = O(\log(1/\lambda)/\phi). \]

\begin{itemize}
  \item For $|Q| = 1000$, $\phi = 0.4$, $\lambda = 10\%$, only needs 5 MEB and NN search in any dimension.
\end{itemize}
randomly select a subset of Q!

\[ \times : \text{group } Q \text{ of query points} \]
\[ \bullet : \text{dataset } P \]

**Theorem**

For any \(0 < \lambda < 1\), executing \textsc{Amax} algorithm only on a random subset of \(f(\phi, \lambda)\) points of \(Q\) returns a \((1 + 2\sqrt{2})\)-approximate answer to the \textsc{Fann} query with probability at least \(1 - \lambda\) in any dimensions, where

\[
f(\phi, \lambda) = \frac{\log \lambda}{\log(1 - \phi)} = O(\log(1/\lambda)/\phi).\]

For \(|Q| = 1000, \phi = 0.4, \lambda = 10\%\), only needs 5 MEB and NN search in any dimension. (even less in practice, \(\frac{1}{\phi}\) is enough!)
randomly select a subset of $Q$!

\[
\begin{align*}
\times & : \text{group } Q \text{ of query points} \\
\bullet & : \text{dataset } P
\end{align*}
\]

**Theorem**

For any $0 < \lambda < 1$, executing \texttt{Amax} algorithm only on a random subset of $f(\phi, \lambda)$ points of $Q$ returns a $(1 + 2\sqrt{2})$-approximate answer to the \texttt{Fann} query with probability at least $1 - \lambda$ in any dimensions, where

\[
f(\phi, \lambda) = \frac{\log \lambda}{\log(1 - \phi)} = O(\log(1/\lambda)/\phi).
\]

- For $|Q| = 1000$, $\phi = 0.4$, $\lambda = 10\%$, only needs 5 MEB and NN search in any dimension. (even less in practice, $\frac{1}{\phi}$ is enough!)
- Independent of dimensionality, $|P|$, and $|Q|$!
All algorithms for $\text{FANN}$ can be extended to work for top-$k$ $\text{FANN}$. 
All algorithms for FANN can be extended to work for top-\(k\) FANN.
Most algorithms work for any metric space, except A_{\text{max}} which works for metric space when MEB is properly defined.
Outline

1. Motivation and Problem Formulation
2. Basic Aggregate Similarity Search
3. Flexible Aggregate Similarity Search
4. Experiments
Experiments: setup and datasets

- Experiments are performed in a Linux machine with 4GB of RAM and an Intel Xeon 2GHz CPU.
- Datasets:
  - 2-dimension: Texas ($TX$) points of interest and road-network dataset from the Open Street Map project: 14 million points (we have other 49 states as well).
  - 2-6 dimensions: synthetic datasets of random clusters ($RC$).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Number of Points</th>
<th>Dimensionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX</td>
<td>14,000,000</td>
<td>2</td>
</tr>
<tr>
<td>RC</td>
<td>synthetic</td>
<td>2–6</td>
</tr>
<tr>
<td>Color</td>
<td>68,040,32</td>
<td>4</td>
</tr>
<tr>
<td>MNIST</td>
<td>60,000</td>
<td>50</td>
</tr>
<tr>
<td>Cortina</td>
<td>1,088,864</td>
<td>74</td>
</tr>
</tbody>
</table>

Report the average of 40 independent queries, as well as the 5%-95% interval.
Experiments: setup and datasets

- Experiments are performed in a Linux machine with 4GB of RAM and an Intel Xeon 2GHz CPU.
- Datasets:
  - 2-dimension: Texas (Tx) points of interest and road-network dataset from the Open Street Map project: 14 million points (we have other 49 states as well).
  - 2-6 dimensions: synthetic datasets of random clusters (RC).
Experiments: setup and datasets

- Experiments are performed in a Linux machine with 4GB of RAM and an Intel Xeon 2GHz CPU.
- Datasets:
  - 2-dimension: Texas (TX) points of interest and road-network dataset from the Open Street Map project: 14 million points (we have other 49 states as well).
  - 2-6 dimensions: synthetic datasets of random clusters (RC).
  - High dimensions: datasets from
    - and [http://www.scl.ece.ucsb.edu/datasets/index.htm](http://www.scl.ece.ucsb.edu/datasets/index.htm)

<table>
<thead>
<tr>
<th>dataset</th>
<th>number of points</th>
<th>dimensionality</th>
</tr>
</thead>
<tbody>
<tr>
<td>TX</td>
<td>14,000,000</td>
<td>2</td>
</tr>
<tr>
<td>RC</td>
<td>synthetic</td>
<td>2 – 6</td>
</tr>
<tr>
<td>Color</td>
<td>68,040</td>
<td>32</td>
</tr>
<tr>
<td>MNIST</td>
<td>60,000</td>
<td>50</td>
</tr>
<tr>
<td>Cortina</td>
<td>1,088,864</td>
<td>74</td>
</tr>
</tbody>
</table>

- report the average of 40 independent queries, as well as the 5%-95% interval.
- sampling rate of $\frac{1}{\phi}$ is enough for both $A_{sum}$ and $A_{max}$!
High dimensions: query cost, all datasets

![Graph showing query cost for MNIST, Color, and Cortina datasets with ASUM and AMAX methods.]

- **MNIST**
  - ASUM:
    - \( r_p / r^* \) = 1.4
  - AMAX:
    - \( r_p / r^* \) = 1.2

- **Color**
  - ASUM:
    - \( r_p / r^* \) = 1.6
  - AMAX:
    - \( r_p / r^* \) = 1.8

- **Cortina**
  - ASUM:
    - \( r_p / r^* \) = 1.8
  - AMAX:
    - \( r_p / r^* \) = 1.6

Authors: Yang Li, Feifei Li, Ke Yi, Bin Yao, Min Wang

Flexible Aggregate Similarity Search
High dimensions: query cost, all datasets

![Graph showing query cost for different datasets and methods. The x-axis represents datasets (MNIST, Color, Cortina) and the y-axis represents query cost.

- **BFS**
- **ASUM**
- **AMAX**

The graph indicates a comparison of query costs across different datasets using various methods.
Thank You

Q and A
Existing methods for ANN

- R-tree method: brunch and bound principle [PSTM04, PTMH05].


Existing methods for ANN

- **R-tree method**: brunch and bound principle [PSTM04, PTMH05].
  - For a query point $q$ and a MBR node $N_i$:
    \[ \forall p \in N_i, \text{mindist}(q, N_i) \leq d(p, q) \leq \text{maxdist}(q, N_i). \]

Existing methods for ANN

- **R-tree method**: branch and bound principle \([\text{PSTM04, PTMH05}]\).
  - For a query group \(Q\) and \(\sigma = \max\),

\[
\forall p \in N_i, \max_{q \in Q} (\text{mindist}(q, N_i)) \leq r_p \leq \max_{q \in Q} (\text{maxdist}(q, N_i)).
\]

Existing methods for ANN

- R-tree method: branch and bound principle [PSTM04, PTMH05].
  - For a query group $Q$ and $\sigma = \text{sum}$,
    \[
    \forall p \in N_i, \sum_{q \in Q} \text{mindist}(q, N_i) \leq r_p \leq \sum_{q \in Q} \text{maxdist}(q, N_i).
    \]

The List algorithm for any dimensions:

\[
\begin{array}{cccc}
q_1 & q_2 & q_3 & q_{|Q|} \\
\hspace{1em} a_1 & \hspace{1em} a_2 & \hspace{1em} a_3 & \hspace{1em} a_{|Q|} \\
\end{array}
\]

For \( \forall p \in P, a_i = d(p, q_i) \)
The List algorithm for any dimensions:

| $q_1$ | $q_2$ | $q_3$ | $q_{|Q|}$ |
|-------|-------|-------|-----------|
| $a_1$ | $a_2$ | $a_3$ | $a_{|Q|}$ |

For $\forall p \in P$, $a_i = d(p, q_i)$

$r_p = \sigma(p, Q^p_\phi)$ is monotone w.r.t. $a_i$'s
The List algorithm for any dimensions:

| $q_1$ | $q_2$ | $q_3$ | $q_{|Q|}$ |
|-------|-------|-------|-----------|
| $a_1$ | $a_2$ | $a_3$ | $a_{|Q|}$ |

For $\forall p \in P$, $a_i = d(p, q_i)$

$r_p = \sigma(p, Q^p_\phi)$ is monotone w.r.t. $a_i$'s

apply the *TA algorithm* [FLN01]

---

The List algorithm for any dimensions:

For $\forall p \in P$, $a_i = d(p, q_i)$

$r_p = \sigma(p, Q^p_\phi)$ is monotone w.r.t. $a_i$’s

apply the TA algorithm [FLN01]

$a_{2,j}$ is the $j$th NN of $q_2$ in $P$!

The List algorithm for any dimensions:

For $\forall p \in P$, $a_i = d(p, q_i)$

$r_p = \sigma(p, Q^p_\phi)$ is monotone w.r.t. $a_i$’s

apply the **TA algorithm** [FLN01]

$a_{2,j}$ is the $j$th NN of $q_2$ in $P$!

Experiments: defaults

Default values:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>$</td>
<td>Q</td>
</tr>
<tr>
<td>$\phi$</td>
<td>support</td>
<td>0.5</td>
</tr>
<tr>
<td>$\mathcal{A}$</td>
<td>query group volume</td>
<td>5% (of the entire data space)</td>
</tr>
<tr>
<td></td>
<td>points in a query group</td>
<td>random cluster distribution</td>
</tr>
</tbody>
</table>
Default values:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>$</td>
<td>Q</td>
</tr>
<tr>
<td>$\phi$</td>
<td>support</td>
<td>0.5</td>
</tr>
<tr>
<td>$A$</td>
<td>query group volume</td>
<td>5% (of the entire data space)</td>
</tr>
<tr>
<td></td>
<td>points in a query group</td>
<td>random cluster distribution</td>
</tr>
</tbody>
</table>

Default values for low dimensions:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>$</td>
<td>P</td>
</tr>
<tr>
<td>$d$</td>
<td>dimensionality</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>dataset</td>
<td>$TX$, when $d = 2$</td>
</tr>
<tr>
<td></td>
<td>dataset</td>
<td>$RC$, when vary $d$ from 2 to 6</td>
</tr>
</tbody>
</table>
Experiments: defaults

- **Default values:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>$</td>
<td>Q</td>
</tr>
<tr>
<td>$\phi$</td>
<td>support</td>
<td>0.5</td>
</tr>
<tr>
<td>$\mathcal{A}$</td>
<td>query group volume</td>
<td>5% (of the entire data space)</td>
</tr>
<tr>
<td></td>
<td>points in a query group</td>
<td>random cluster distribution</td>
</tr>
</tbody>
</table>

- **Default values for low dimensions:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>$</td>
<td>P</td>
</tr>
<tr>
<td>$d$</td>
<td>dimensionality</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>dataset</td>
<td>$TX$, when $d = 2$</td>
</tr>
<tr>
<td></td>
<td>dataset</td>
<td>$RC$, when vary $d$ from 2 to 6</td>
</tr>
</tbody>
</table>

- **Values for high dimensions:**

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>$</td>
<td>P</td>
</tr>
<tr>
<td>$d$</td>
<td>dimensionality</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>default dataset</td>
<td>$Cortina$</td>
</tr>
</tbody>
</table>
Experiments: defaults

- Default values:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>$</td>
<td>Q</td>
</tr>
<tr>
<td>$\phi$</td>
<td>support</td>
<td>0.5</td>
</tr>
<tr>
<td>$A$</td>
<td>query group volume</td>
<td>5% (of the entire data space)</td>
</tr>
<tr>
<td></td>
<td>points in a query group</td>
<td>random cluster distribution</td>
</tr>
</tbody>
</table>

- Default values for low dimensions:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>$</td>
<td>P</td>
</tr>
<tr>
<td>$d$</td>
<td>dimensionality</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>dataset</td>
<td>$TX$, when $d = 2$</td>
</tr>
<tr>
<td></td>
<td>dataset</td>
<td>$RC$, when vary $d$ from 2 to 6</td>
</tr>
</tbody>
</table>

- Values for high dimensions:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N$</td>
<td>$</td>
<td>P</td>
</tr>
<tr>
<td>$d$</td>
<td>dimensionality</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>default dataset</td>
<td>$Cortina$</td>
</tr>
</tbody>
</table>

- report the average of 40 independent, randomly generated queries, as well as the 5%-95% interval.
- sampling rate of $\frac{1}{\phi}$ is enough for both $A_{SUM}$ and $A_{MAX}$!
Low dimensions: approximation quality

![Graph showing the relationship between $r_p/r^*$ and $\phi$ for different methods]

- **ASUM**
- **AMAX**

Authors: Yang Li, Feifei Li, Ke Yi, Bin Yao, Min Wang
Low dimensions: approximation quality

\[
\frac{r_p}{r^*}
\]

ASUM
AMAX

Yang Li, Feifei Li, Ke Yi, Bin Yao, Min Wang
Flexible Aggregate Similarity Search
Low dimensions: query cost, vary $M$

![Diagram showing query cost variation for different methods: BFS, R-treeSUM, R-treeMAX, ASUM, AMAX. The x-axis represents $M \times 10^2$, and the y-axis shows the query cost in a log scale. The methods are compared for varying M values.]
Low dimensions: query cost, vary $M$

![Graph showing running time for different algorithms](graph.png)

- BFS
- R-treeSUM
- R-treeMAX
- ASUM
- AMAX

Running time (seconds) vs. $M$: $10^{2}$

(Yang Li, Feifei Li, Ke Yi, Bin Yao, Min Wang) Flexible Aggregate Similarity Search
Low dimensions: query cost, vary $N$

- BFS
- R-treeSUM
- R-treeMAX
- ASUM
- AMAX

IO

10^4

10^3

10^2

10^1

10^0

$N:X10^6$

1 2 3 4 5

Yang Li, Feifei Li, Ke Yi, Bin Yao, Min Wang

Flexible Aggregate Similarity Search
Low dimensions: query cost, vary $N$

Running time (seconds)

BFS, R-tree\text{SUM}, R-tree\text{MAX}, ASUM, AMAX

N:X10^6

Yang Li, Feifei Li, Ke Yi, Bin Yao, Min Wang
Flexible Aggregate Similarity Search
Low dimensions: query cost, vary $\phi$

- BFS
- R-tree SUM
- R-tree MAX
- ASUM
- AMAX

$IO$ vs $\phi$
Low dimensions: query cost, vary $\phi$

Yang Li, Feifei Li, Ke Yi, Bin Yao, Min Wang
Flexible Aggregate Similarity Search
Low dimensions: query cost, vary $d$

![Graph](graph.png)

- BFS
- R-treeSUM
- R-treeMAX
- ASUM
- AMAX

Yang Li, Feifei Li, Ke Yi, Bin Yao, Min Wang
Flexible Aggregate Similarity Search
Low dimensions: query cost, vary $d$

- BFS
- R-treeSUM
- R-treeMAX
- ASUM
- AMAX

Running time (seconds) vs. $d$
High dimensions: approximation quality

![Graph showing the relationship between $r_p/r^*$ and $\phi$. The graph compares ASUM (squares) and AMAX (triangles). The x-axis represents $\phi$ values from 0.1 to 1, and the y-axis represents $r_p/r^*$ values from 1 to 1.3. The data points indicate a trend where ASUM generally has a higher $r_p/r^*$ value than AMAX, with some variation at different $\phi$ values.]
High dimensions: approximation quality

\[ \frac{r_p}{r^*} \]

\[ N \times 10^5 \]

Graph showing the ratio of \( r_p \) to \( r^* \) for different values of \( N \times 10^5 \). The graph compares two methods: ASUM (squares) and AMAX (triangles).
High dimensions: approximation quality

\[ \frac{r_p}{r^*} \]

\( d \)

\( \square \) ASUM

\( \triangle \) AMAX
High dimensions: query cost, vary $M$

- BFS
- List−iDistSUM
- List−iDistMAX
- List−LsbSUM
- List−LsbMAX
- ASUM
- AMAX

IO

$10^2$ $10^3$ $10^4$

$M$

$4$ $8$ $16$ $32$ $64$ $128$ $256$ $512$
High dimensions: query cost, vary $M$

- BFS
- List–iDistSUM
- List–iDistMAX
- List–LsbSUM
- List–LsbMAX
- ASUM
- AMAX

Running time (seconds) vs. $M$
High dimensions: query cost, vary $N$
High dimensions: query cost, vary $N$

- BFS
- ASUM
- AMAX

Running time (seconds)

- $10^{-4}$
- $10^{-2}$
- $10^{0}$
- $10^{2}$

$N: \times 10^5$

Yang Li, Feifei Li, Ke Yi, Bin Yao, Min Wang
Flexible Aggregate Similarity Search
High dimensions: query cost, vary $\phi$

![Graph showing query cost variations with $\phi$ values.](image)
High dimensions: query cost, vary $\phi$

![Graph showing running time vs. $\phi$]

- BFS
- ASUM
- AMAX

Running time (seconds)

- $10^{-4}$
- $10^{-2}$
- $10^{0}$
- $10^{2}$

$\phi$

0.1 0.3 0.5 0.7 0.9 1

**Flexible Aggregate Similarity Search**

Yang Li, Feifei Li, Ke Yi, Bin Yao, Min Wang
High dimensions: query cost, vary $d$

![Graph showing query cost for varying dimensions ($d$). The graph plots the query cost against $d$, with three different methods: BFS, ASUM, and AMAX. The y-axis represents query cost on a logarithmic scale, ranging from $10^1$ to $10^4$, and the x-axis represents $d$ ranging from 10 to 50.]
High dimensions: query cost, vary \(d\)

![Graph showing running time vs. \(d\) for BFS, ASUM, and AMAX methods.](image_url)