Set2Box: Similarity Preserving Representation Learning of Sets

Geon Lee
Chanyoung Park
Kijung Shin
Similarity Between Sets is Used Everywhere

- **Similarity between sets** has been employed in many areas:
  - Recommendation
  - Graph compression
  - Medical analysis
  - Other examples include plagiarism detection and gene expression.

Do user A and user B have similar preferences?

Do node 3 and node 4 have similar sets of neighbors? Should we merge them as a supernode?

Do two MRI images have similar keypoints?
Similarity Between Sets is Used Everywhere

- **Similarity between sets** has been employed in many areas:
  - Recommendation
  - **Graph compression**
  - Medical analysis
  - Other examples include *plagiarism detection* and *gene expression*.

---

**Introduction**

- Do user A and user B have similar preferences?
  - User A
  - User B

- Do node 3 and node 4 have similar sets of neighbors? Should we merge them as a supernode?

- Do two MRI images have similar keypoints?
Similarity Between Sets is Used Everywhere

- **Similarity between sets** has been employed in many areas:
  - Recommendation
  - Graph compression
  - **Medical analysis**
  - Other examples include plagiarism detection and gene expression.

---

Do user A and user B have similar preferences?

Do node 3 and node 4 have similar sets of neighbors? Should we merge them as a supernode?

Do two MRI images have similar keypoints?
Why Do We Embed Sets?

• Sets **grow** in numbers and sizes.
  • **E.g., 1.** Millions of users rate tens of thousands of movies.
  • **E.g., 2.** Many nodes in graphs have thousands of neighbors.

→ Computation of **set similarity** requires substantial **storage** and **time**.

How can we represent sets **accurately**, **concisely**, and **fast**?
**Similarity Preserving Set Embedding**

- **Given:** (1) a set $S$ of sets and (2) a budget $b$
- **Find:** a latent representation $Z_S$ of each set $s \in S$
- **to Minimize:** $\|\text{sim}(s, s') - \hat{\text{sim}}(Z_s, Z_{s'})\|$  **Accuracy**
- **Subject to:** the total encoding cost $\text{Cost}(\{Z_s: s \in S\}) \leq b$  **Conciseness**
- **Desired to:** compute set similarity in a constant time  **Speed**
Similarity Preserving Set Embedding (cont.)

• There are diverse set similarity measures.
  • It is desirable to be used for various similarity measures. **Versatility**

<table>
<thead>
<tr>
<th>Similarity of Pair ((A, B)) of Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Jaccard Index</strong></td>
</tr>
<tr>
<td>( \frac{</td>
</tr>
<tr>
<td><strong>Overlap Coefficient</strong></td>
</tr>
<tr>
<td>( \frac{</td>
</tr>
<tr>
<td><strong>Dice Index</strong></td>
</tr>
<tr>
<td>( \frac{2 \cdot</td>
</tr>
<tr>
<td><strong>Cosine Similarity</strong></td>
</tr>
<tr>
<td>( \frac{</td>
</tr>
</tbody>
</table>
Roadmap

1. Concepts

2. Basic Method: Set2Box

3. Advanced Method: Set2Box+

4. Experimental Results

5. Conclusion
Box Embedding

- **Set2Box** is an accurate algorithm for similarity preserving set embedding.
  - We represent sets as **boxes (ranges)** instead of vectors (points).

**Box Embedding (Set2Box)**

**Vector Embedding**

Box $B_X$ of set $X$

Box $B_Y$ of set $Y$

Vector $V_X$ of set $X$

Vector $V_Y$ of set $Y$
**Box Embedding (cont.)**

A box $B_X$ consists of two vectors:

- **Center** $c_X = (4,3)$
- **Offset** $f_X = (3,2)$

From $c_X$ and $r_X$, we can obtain min/max vectors:

- **Min point** $m_X = c_X - f_X = (1,1)$
- **Max point** $M_X = c_X + f_X = (7,5)$

The **volume** of the box is computed by:

$$\forall(B_X) = \prod_{i=1}^{d}(M_X[i] - m_X[i]) = 6 \cdot 4 = 24$$
**Box Embedding (cont.)**

The min/max vectors of box $B_X$ are:

- Min point $m_X = c_X - f_X = (1,4)$
- Max point $M_X = c_X + f_X = (5,6)$

The min/max vectors of box $B_Y$ are:

- Min point $m_Y = c_Y - f_Y = (2,1)$
- Max point $M_Y = c_Y + f_Y = (6,5)$

The min/max vectors of box $B_X \cap B_Y$ are:

- Min point $m_{X\cap Y} = \max(m_X, m_Y) = (2,4)$
- Max point $M_{X\cap Y} = \min(M_X, M_Y) = (5,5)$
Box Embedding (cont.)

- Several **set operations** hold in box embedding.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Transitivity Law</td>
<td>( A \subset B, B \subset C \rightarrow A \subset C )</td>
</tr>
</tbody>
</table>
| 2. Idempotent Law | \( A \cup A = A \)  
\( A \cap A = A \) |
| 3. Commutative Law | \( A \cup B = B \cup A \)  
\( A \cap B = B \cap A \) |
| 4. Associative Law | \( A \cup (B \cup C) = (A \cup B) \cup C \)  
\( A \cap (B \cap C) = (A \cap B) \cap C \) |
| 5. Absorption Law | \( A \cup (A \cap B) = A \)  
\( A \cap (A \cup B) = A \) |
| 6. Distributive Law | \( A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \)  
\( A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \) |
Roadmap

1. Concepts

2. Basic Method: Set2Box

3. Advanced Method: Set2Box+

4. Experimental Results

5. Conclusion
Set2Box: Representing Sets as Boxes

- We learn a pair of embedding matrices of entities $\mathcal{E}$:
  - $Q^c \in \mathbb{R}^{|\mathcal{E}| \times d}$: centers of entities
  - $Q^f \in \mathbb{R}^{|\mathcal{E}| \times d}$: offsets of entities
- We aggregate (i.e., pool) entities’ embeddings to obtain the box $B_s$ of the set $s$.

![Diagram](image)
Set2Box: Representing Sets as Boxes (cont.)

- We aim to preserve relations among triple \( \{ s_1, s_2, s_3 \} \) of sets.
  - Preserve the **cardinalities of the subsets** by the **volumes of the boxes**.
Set2Box: Representing Sets as Boxes (cont.)

- We aim to preserve relations among triple \( \{ s_1, s_2, s_3 \} \) of sets.
  - Learn the relative sizes of the following seven subsets:
    \[
    |s_1|, \quad |s_2|, \quad |s_3|, \quad |s_1 \cap s_2|, \quad |s_2 \cap s_3|, \quad |s_3 \cap s_1|, \text{ and } |s_1 \cap s_2 \cap s_3|
    \]
    - Singlewise overlaps
    - Pairwise overlaps
    - Triplewise overlap

- The objective is to preserve the sizes by the box volumes:
  \[
  \begin{align*}
  |s_1| & \propto V(B_{s_1}) & |s_1 \cap s_2| & \propto V(B_{s_1} \cap B_{s_2}) \\
  |s_2| & \propto V(B_{s_2}) & |s_2 \cap s_3| & \propto V(B_{s_2} \cap B_{s_3}) \\
  |s_3| & \propto V(B_{s_3}) & |s_3 \cap s_1| & \propto V(B_{s_3} \cap B_{s_1}) \\
  |s_1 \cap s_2 \cap s_3| & \propto V(B_{s_1} \cap B_{s_2} \cap B_{s_3})
  \end{align*}
  \]

\[\Rightarrow\] MSE Loss
Roadmap

1. Concepts
2. Basic Method: Set2Box
3. Advanced Method: Set2Box$^+$
4. Experimental Results
5. Conclusion
Set2Box$^+$: Even More Concise & Accurate

- We propose Set2Box$^+$ to derive better conciseness and accuracy.
  - Set2Box$^+$ consists of two effective schemes:
    - **Box quantization** makes boxes more concise.
    - **Joint training** improves the accuracy.
**Set2Box+: Even More Concise & Accurate (cont.)**

- **Box quantization (BQ)** compresses boxes.
  - Divide the box $B \in \mathbb{R}^d$ into $D$ subspaces where each dimension is $\mathbb{R}^{d/D}$.
  - In each subspace, there are $K$ key boxes.

\[
B = \begin{array}{c}
\text{d-dimensional original box}
\end{array}
\]

\[
2 \cdot 32 \cdot d \text{ bits}
\]

\[
\begin{array}{c}
\text{Subspace 1 (}\mathbb{R}^{d/D}\text{)}
\end{array}
\]

\[
\begin{array}{c}
K_1^{(1)}
\end{array}
\]

\[
\begin{array}{c}
K_3^{(1)}
\end{array}
\]

\[
\begin{array}{c}
B^{(1)}
\end{array}
\]

\[
\begin{array}{c}
K_2^{(1)}
\end{array}
\]

\[
\begin{array}{c}
2 \cdot D \cdot K \cdot 32 \cdot d \text{ bits}
\end{array}
\]

\[
\begin{array}{c}
\text{Subspace 2 (}\mathbb{R}^{d/D}\text{)}
\end{array}
\]

\[
\begin{array}{c}
K_1^{(2)}
\end{array}
\]

\[
\begin{array}{c}
K_3^{(2)}
\end{array}
\]

\[
\begin{array}{c}
B^{(2)}
\end{array}
\]

\[
\begin{array}{c}
K_2^{(2)}
\end{array}
\]

\[
\begin{array}{c}
\text{Subspace 3 (}\mathbb{R}^{d/D}\text{)}
\end{array}
\]

\[
\begin{array}{c}
K_1^{(3)}
\end{array}
\]

\[
\begin{array}{c}
K_3^{(3)}
\end{array}
\]

\[
\begin{array}{c}
B^{(3)}
\end{array}
\]

\[
\begin{array}{c}
K_2^{(3)}
\end{array}
\]

\[
\begin{array}{c}
\text{Subspace 4 (}\mathbb{R}^{d/D}\text{)}
\end{array}
\]

\[
\begin{array}{c}
K_1^{(4)}
\end{array}
\]

\[
\begin{array}{c}
K_3^{(4)}
\end{array}
\]

\[
\begin{array}{c}
B^{(4)}
\end{array}
\]

\[
\begin{array}{c}
K_2^{(4)}
\end{array}
\]

\[
\begin{array}{c}
\text{D-dimensional discrete vector}
\end{array}
\]

\[
b = \begin{array}{c}
2 \ 3 \ 1 \ 3
\end{array}
\]

\[
\begin{array}{c}
\text{n} \cdot D \cdot \log_2 K \text{ bits}
\end{array}
\]

\[
\begin{array}{c}
\text{d-dimensional reconstructed box}
\end{array}
\]

\[
\tilde{B} = \begin{array}{c}
\end{array}
\]
Set2Box+: Even More Concise & Accurate (cont.)

- **Box quantization (BQ)** compresses boxes.
  - To encode \( n \) number of \( d \)-dimensional boxes:
    - (Original) \( 64nd \) bits \( \gg \) (BQ) \( 64DKd + nD \log_2 K \) bits

\[
B = \begin{pmatrix}
K_1^{(1)} & K_2^{(1)} & K_3^{(1)} \\
B^{(1)} & K_2^{(1)} & 0
\end{pmatrix}
\]

\( d \)-dimensional original box

\( 2 \cdot 32 \cdot d \) bits

\[
\tilde{B} = \begin{pmatrix}
K_1^{(4)} & K_2^{(4)} & K_3^{(4)} \\
K_1^{(3)} & K_2^{(3)} & K_3^{(3)}
\end{pmatrix}
\]

\( d \)-dimensional reconstructed box

\( b = [2 \ 3 \ 1 \ 3] \)

\( D \)-dimensional discrete vector

\( 2 \cdot D \cdot K \cdot 32 \cdot d \) bits
Set2Box+: Even More Concise & Accurate (cont.)

- How does box quantization find the closest key box?

To compute box similarities, we define Box Overlap Ratio:

$$\text{BOR}(B_X, B_Y) = \frac{1}{2} \left( \frac{\mathbb{V}(B_X \cap B_Y)}{\mathbb{V}(B_X)} + \frac{\mathbb{V}(B_X \cap B_Y)}{\mathbb{V}(B_Y)} \right)$$

$$2 = \arg \max_i \text{BOR}(x^{(2)}, K_i^{(2)})$$
Set2Box+: Even More Concise & Accurate (cont.)

- An overview of **box quantization (BQ)**.
  - $\mathcal{L}$: Similarity preserving MSE loss

---

**Box Quantization**

- **center** $\mathbf{c}_s$
- **offset** $\mathbf{f}_s$

---

**Set2Box**

- Introduction
- Concepts
- Basic Method
- Advanced Method
- Experimental Results
- Conclusion
**Speed of Set2Box**+

- **Set2Box**+ computes estimated set similarity sets in a **constant** time.

**Lemma (Time Complexity of Similarity Estimation)**

Given a pair of sets $s$ and $s'$ and their boxes $B_s$ and $B_{s'}$, respectively, it takes $O(d)$ time to compute the estimated similarity $\hat{\text{sim}}(B_s, B_{s'})$, where $d$ is a user-defined **constant** that does not depend on the sizes of $s$ and $s'$. 

---

**Advanced Method**

---

**Introduction**

---

**Concepts**

---

**Basic Method**

---

**Experimental Results**

---

**Conclusion**
Other Details

- In the paper, you can find:
  - ✓ Set context pooling
  - ✓ End-to-end discrete code learning
  - ✓ Joint training original and reconstructed boxes
  - ✓ Box smoothing for effective learning
Roadmap

1. Concepts

2. Basic Method: Set2Box

3. Advanced Method: Set2Box^*

4. Experimental Results

5. Conclusion
Accuracy & Conciseness of Set2Box⁺

- Set2Box⁺ preserves set similarities most accurately compared to baselines.
- Set2Box⁺ gives up to 40.8X smaller estimation error while requiring about 60% fewer bits to encode sets.

---

**Experimental Results**

**Conclusion**
Accuracy & Conciseness of Set2Box\(^+\) (cont.)

- For example, Set2Box\(^+\) preserves the Overlap Coefficient between sets more accurately with smaller encoding cost.

![Comparison of methods](image)

(a) Random Hashing
MSE = 0.0884
Cost = 77.312 KB

(b) Vector Embedding
MSE = 0.0495
Cost = 77.312 KB

(c) Set2Box\(^+\)
MSE = 0.0125
Cost = 15.695 KB
Effects of Box Quantization & Joint Training

- We compare following variants:
  - **Set2Box-PQ**: Product quantization for center & offset
  - **Set2Box-BQ**: Box quantization without joint training
  - **Set2Box^+**: The proposed method with box quantization and joint training

<table>
<thead>
<tr>
<th>Method</th>
<th>OC</th>
<th>CS</th>
<th>JI</th>
<th>DI</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set2Box-PQ</td>
<td>0.0129</td>
<td>0.0028</td>
<td>0.0012</td>
<td>0.0023</td>
</tr>
<tr>
<td>Set2Box-BQ</td>
<td>0.0106 (-17%)</td>
<td>0.0023 (-17%)</td>
<td>0.0009 (-26%)</td>
<td>0.0019 (-17%)</td>
</tr>
<tr>
<td>Set2Box^+</td>
<td><strong>0.0077 (-40%)</strong></td>
<td><strong>0.0016 (-44%)</strong></td>
<td><strong>0.0007 (-41%)</strong></td>
<td><strong>0.0013 (-42%)</strong></td>
</tr>
</tbody>
</table>

**Box quantization** and **joint training** of **Set2Box^+** incrementally improves the accuracy (in terms of MSE) averaged over all datasets.
Roadmap

1. Concepts
2. Basic Method: Set2Box
3. Advanced Method: Set2Box+
4. Experimental Results
5. Conclusion
Conclusion

- We propose Set2Box\(^+\), an effective and efficient representation learning method for preserving similarities between sets.

**Set2Box\(^+\)** is:

- **Accurate**: yields smaller estimation error while requiring smaller encoding cost.
- **Concise**: requires smaller encoding cost to achieve the same performance.
- **Fast**: computes set similarities in a constant time.
- **Versatile**: estimates various set similarity measures with a single set embedding.

Code & datasets: [https://github.com/geon0325/Set2Box](https://github.com/geon0325/Set2Box)
Set2Box: Similarity Preserving Representation Learning of Sets

Geon Lee
Chanyoung Park
Kijung Shin