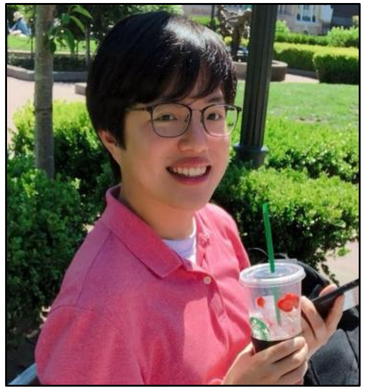
Set2Box: Similarity Preserving Representation Learning of Sets

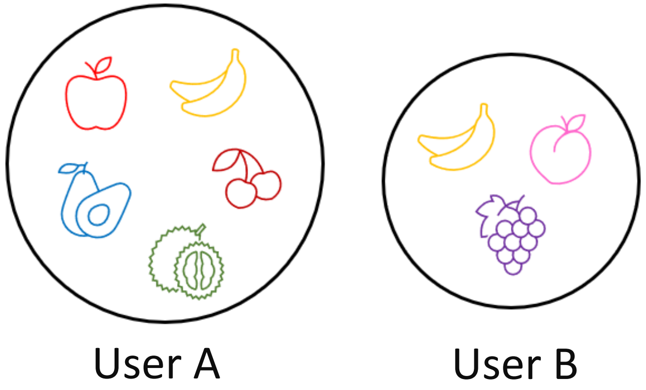

Geon Lee


Chanyoung Park

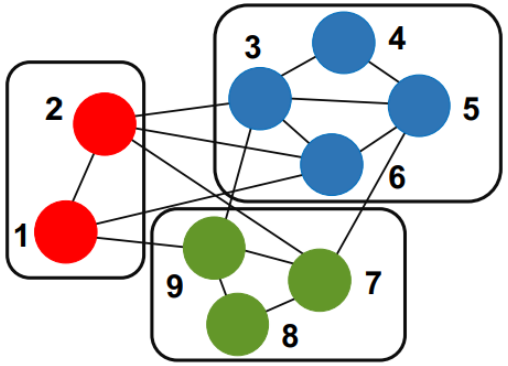

Kijung Shin

Similarity Between Sets is Used Everywhere

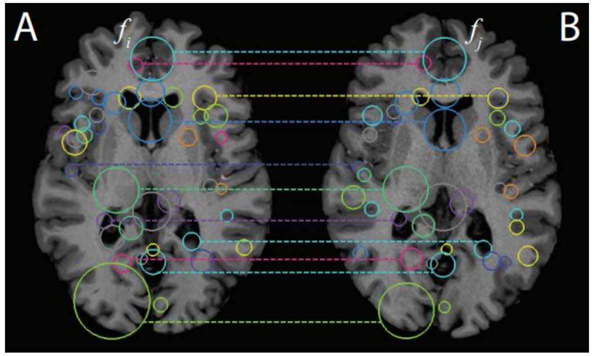
- **Similarity between sets** has been employed in many areas:
  - **Recommendation**
  - Graph compression
  - Medical analysis
  - Other examples include **plagiarism detection** and **gene expression**.



Do user A and user B have similar preferences?



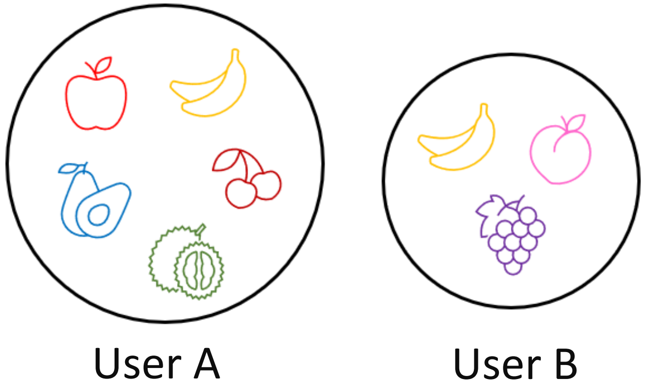
Do node 3 and node 4 have similar sets of neighbors?
Should we merge them as a supernode?



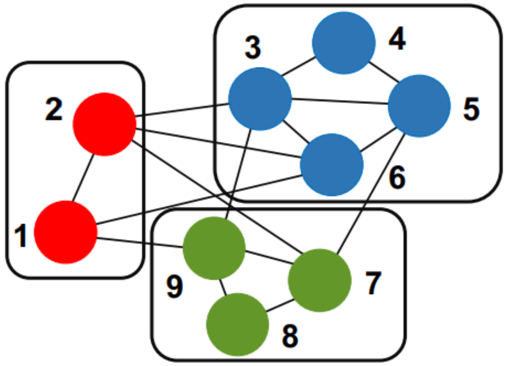
Do two MRI images have similar keypoints?

Similarity Between Sets is Used Everywhere

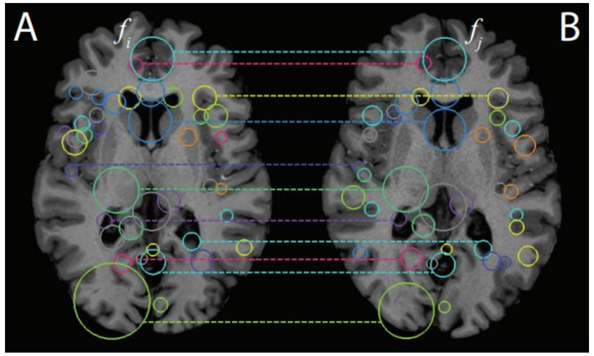
- **Similarity between sets** has been employed in many areas:
  - Recommendation
  - **Graph compression**
  - Medical analysis
  - Other examples include **plagiarism detection** and **gene expression**.



Do user A and user B have similar preferences?



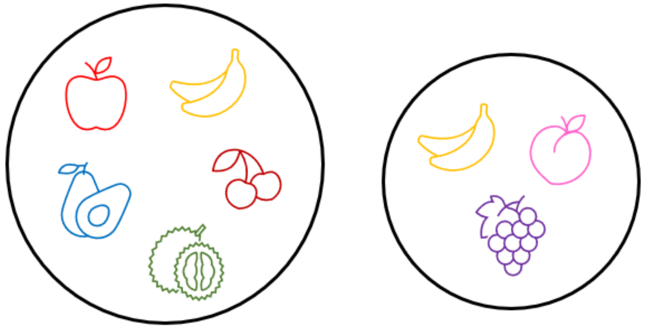
Do node 3 and node 4 have similar sets of neighbors? Should we merge them as a supernode?



Do two MRI images have similar keypoints?

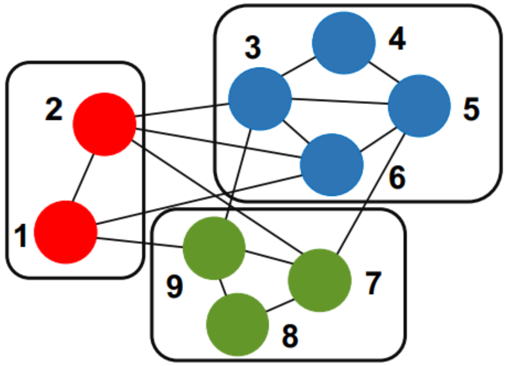
Similarity Between Sets is Used Everywhere

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

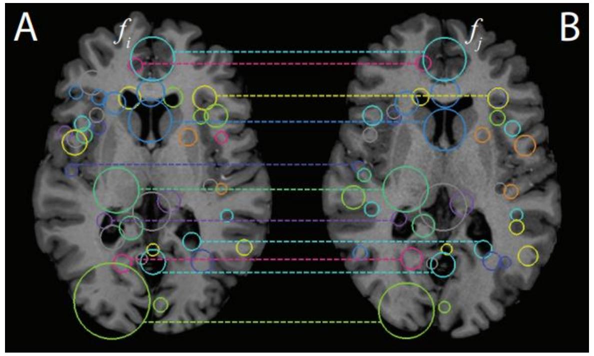


User A User B

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?

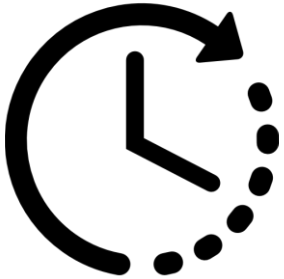


Do two MRI images have similar keypoints?

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.

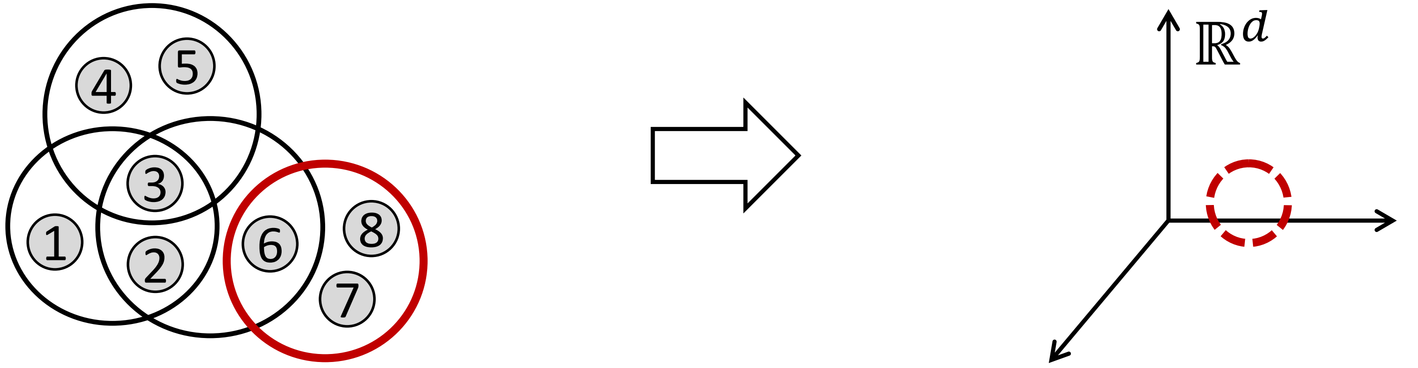
→ Computation of **set similarity** requires substantial **storage** and **time**.



*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:** $\left\| sim(s, s') - \widehat{sim}(z_s, z_{s'}) \right\|$ **Accuracy**
- **Subject to:** the total encoding cost $Cost(\{z_s : s \in S\}) \leq b$ **Conciseness**
- **Desired to:** compute set similarity in a constant time **Speed**



Set of Sets

Embedding Space

Similarity Preserving Set Embedding (cont.)

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

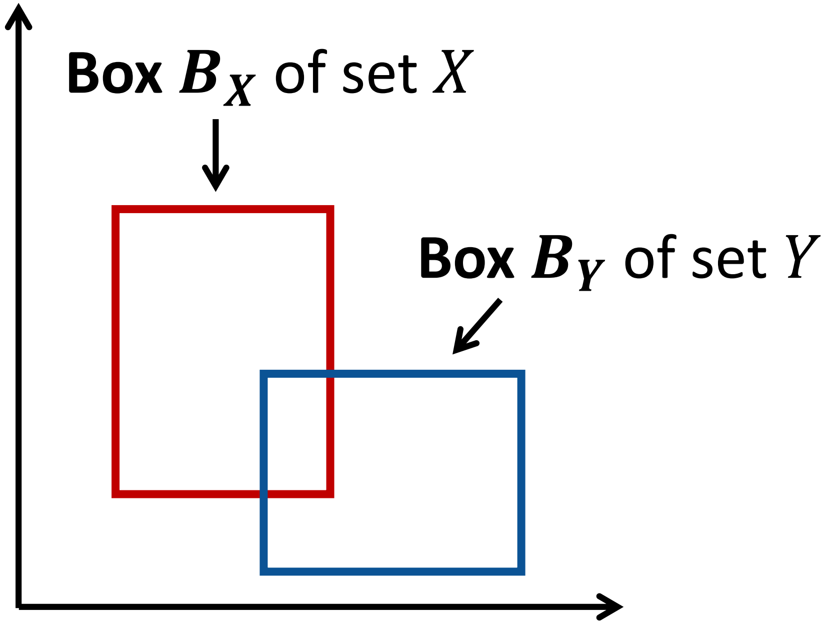
	Similarity of Pair (A, B) of Sets
Jaccard Index	$\frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$
Overlap Coefficient	$\frac{\lvert A \cap B \rvert}{\min(\lvert A \rvert, \lvert B \rvert)}$
Dice Index	$\frac{2 \cdot \lvert A \cap B \rvert}{\lvert A \rvert + \lvert B \rvert}$
Cosine Similarity	$\frac{\lvert A \cap B \rvert}{\sqrt{\lvert A \rvert \cdot \lvert B \rvert}}$

Roadmap

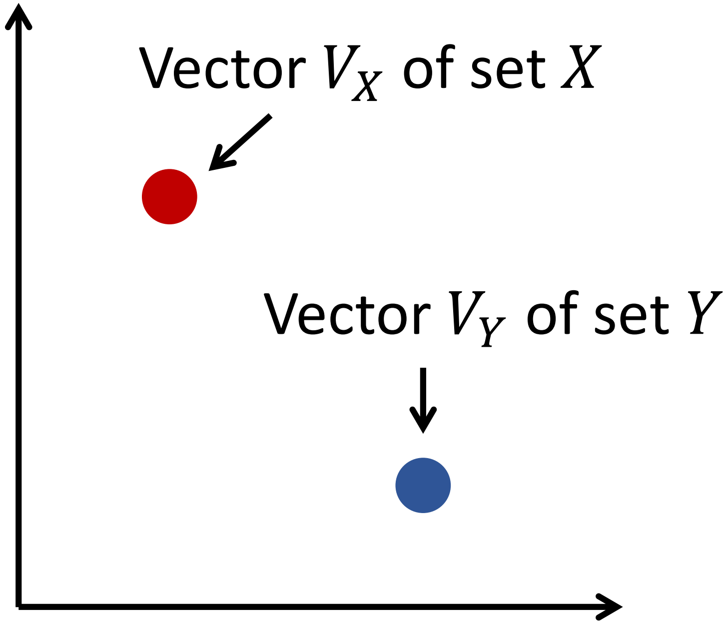
1. **Concepts**
2. Basic Method: Set2Box
3. Advanced Method: Set2Box$^+$
4. Experimental Results
5. Conclusion

Box Embedding

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

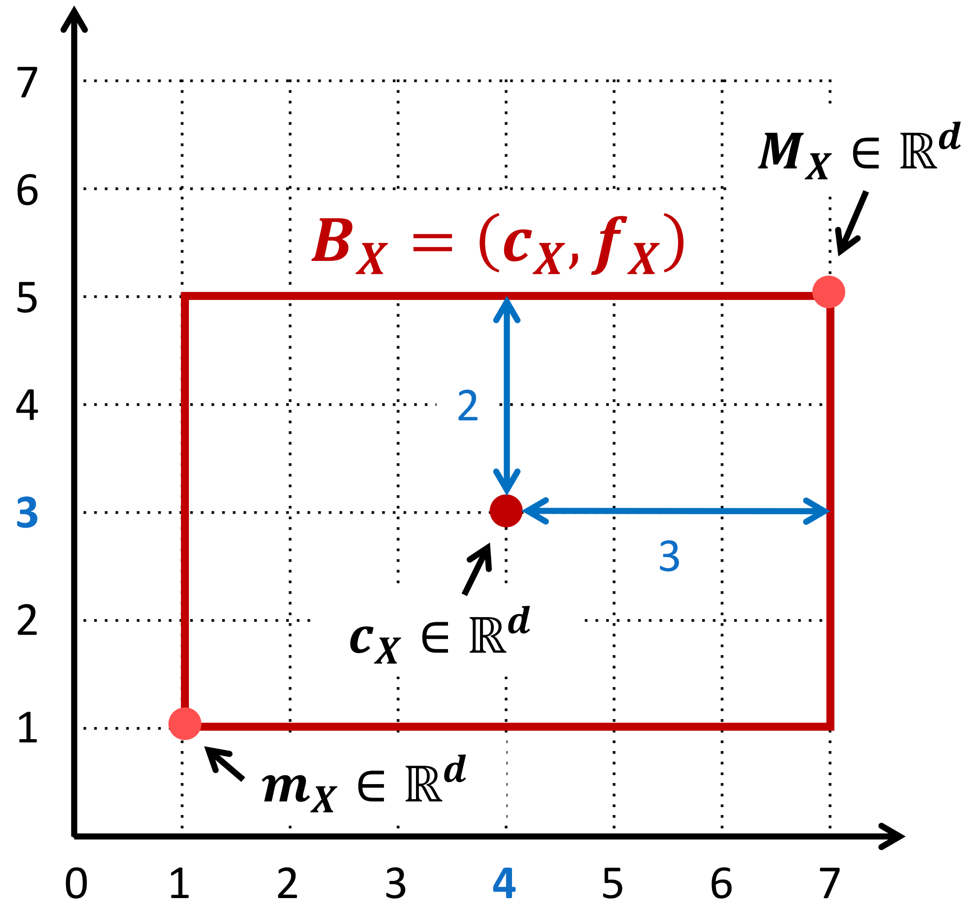


Box Embedding (**Set2Box**)



Vector Embedding

Box Embedding (cont.)



A **box** B_X consists of two vectors:

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

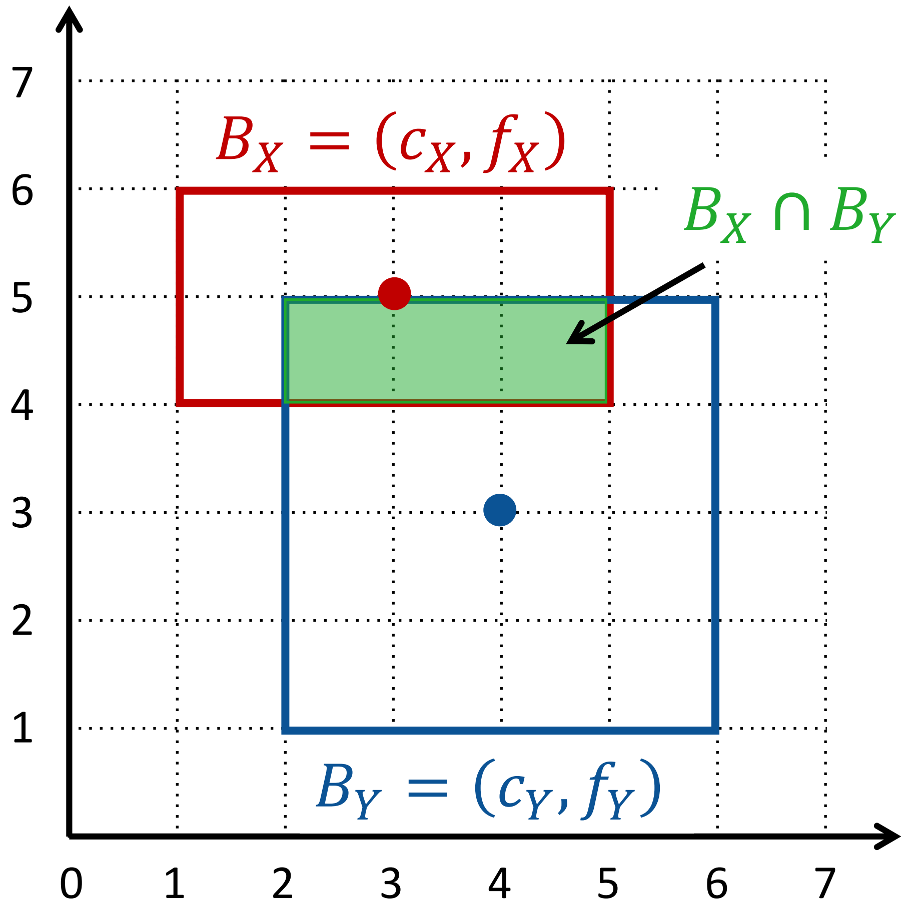
From c_X and r_X, we can obtain min/max vectors:

- 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:

$$\mathbb{V}(B_X) = \prod_{i=1}^{d}(M_X[i] - m_X[i]) = 6 \cdot 4 = 24$$

Box Embedding (cont.)



The min/max vectors of box B_X are:

- Min point $\boldsymbol{m}_X = \boldsymbol{c}_X - \boldsymbol{f}_X = (1,4)$
- Max point $\boldsymbol{M}_X = \boldsymbol{c}_X + \boldsymbol{f}_X = (5,6)$

The min/max vectors of box B_Y are:

- Min point $\boldsymbol{m}_Y = \boldsymbol{c}_Y - \boldsymbol{f}_Y = (2,1)$
- Max point $\boldsymbol{M}_Y = \boldsymbol{c}_Y + \boldsymbol{f}_Y = (6,5)$

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

- Min point $\boldsymbol{m}_{X \cap Y} = \mathbf{max}(\boldsymbol{m}_X, \boldsymbol{m}_Y) = (2,4)$
- Max point $\boldsymbol{M}_{X \cap Y} = \mathbf{min}(\boldsymbol{M}_X, \boldsymbol{M}_Y) = (5,5)$

Box Embedding (cont.)

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

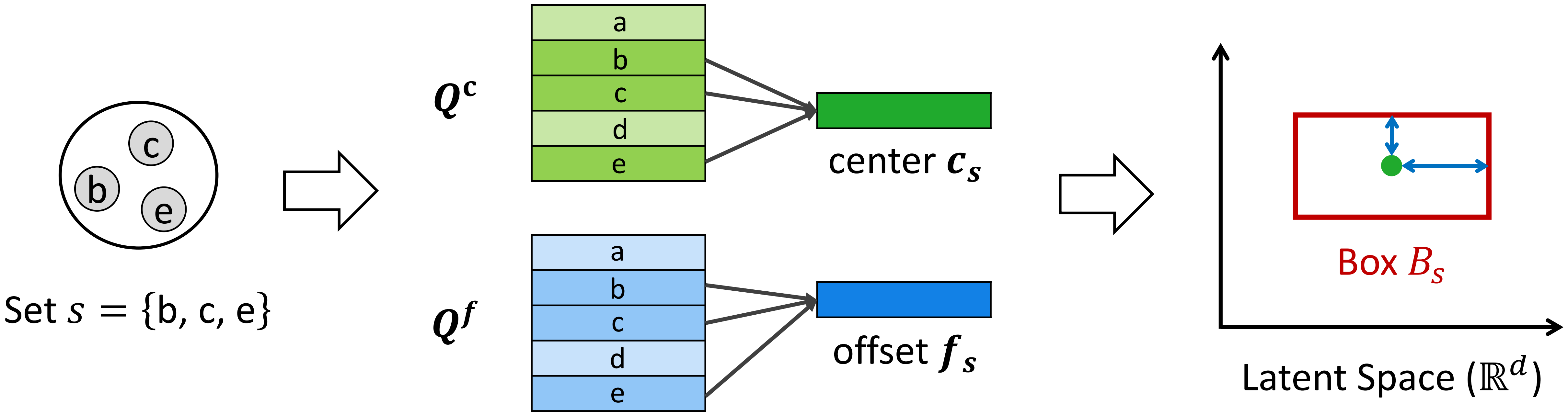
1. Transitivity Law	$A \subset B, B \subset C \rightarrow A \subset C$
2. Idempotent Law	$A \cup A = A$ $A \cap A = A$
3. Commutative Law	$A \cup B = B \cup A$ $A \cap B = B \cap A$
4. Associative Law	$A \cup (B \cup C) = (A \cup B) \cup C$ $A \cap (B \cap C) = (A \cap B) \cap C$
5. Absorption Law	$A \cup (A \cap B) = A$ $A \cap (A \cup B) = A$
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

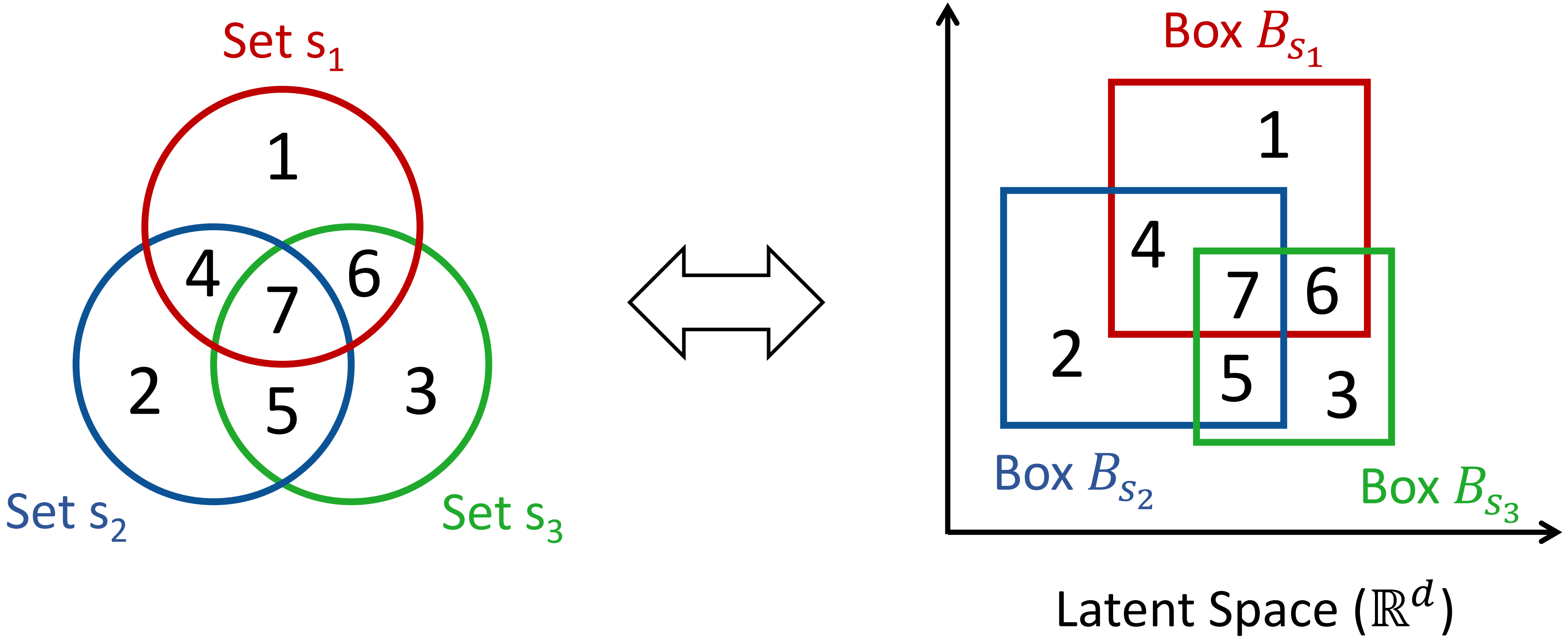
Set2Box: Representing Sets as Boxes

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



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:

$$\underbrace{|s_1|, |s_2|, |s_3|}_{\text{Singlewise overlaps}}, \underbrace{|s_1 \cap s_2|, |s_2 \cap s_3|, |s_3 \cap s_1|}_{\text{Pairwise overlaps}}, \text{ and } \underbrace{|s_1 \cap s_2 \cap s_3|}_{\text{Triplewise overlap}}$$

  - The objective is to preserve the sizes by the **box volumes**:

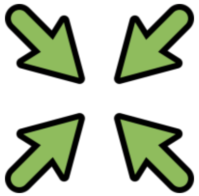
$$\boxed{\begin{array}{ll} |s_1| \propto \mathbb{V}(\boldsymbol{B}_{\boldsymbol{s_1}}) & |s_1 \cap s_2| \propto \mathbb{V}(\boldsymbol{B}_{\boldsymbol{s_1}} \cap \boldsymbol{B}_{\boldsymbol{s_2}}) \\ |s_2| \propto \mathbb{V}(\boldsymbol{B}_{\boldsymbol{s_2}}) & |s_2 \cap s_3| \propto \mathbb{V}(\boldsymbol{B}_{\boldsymbol{s_2}} \cap \boldsymbol{B}_{\boldsymbol{s_3}}) \\ |s_3| \propto \mathbb{V}(\boldsymbol{B}_{\boldsymbol{s_3}}) & |s_3 \cap s_1| \propto \mathbb{V}(\boldsymbol{B}_{\boldsymbol{s_3}} \cap \boldsymbol{B}_{\boldsymbol{s_1}}) \\ \multicolumn{2}{c}{|s_1 \cap s_2 \cap s_3| \propto \mathbb{V}(\boldsymbol{B}_{\boldsymbol{s_1}} \cap \boldsymbol{B}_{\boldsymbol{s_2}} \cap \boldsymbol{B}_{\boldsymbol{s_3}})} \end{array}} \Rightarrow \text{MSE Loss}$$

Roadmap

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

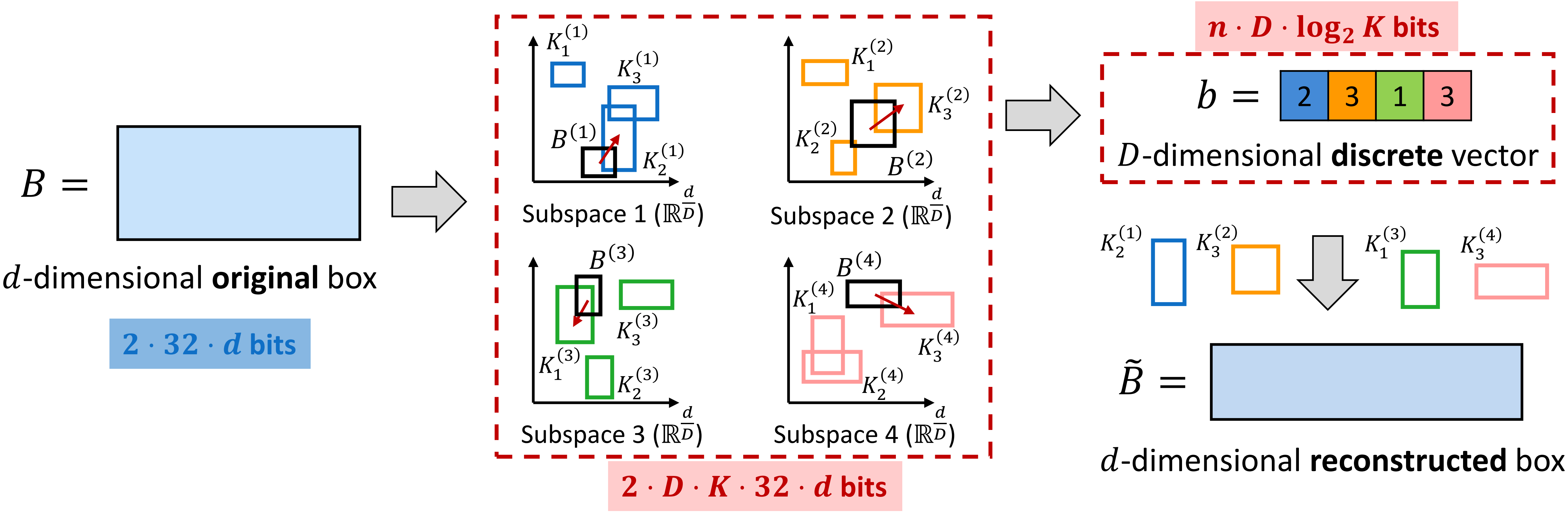
Set2Box⁺: Even More Concise & Accurate

- 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.



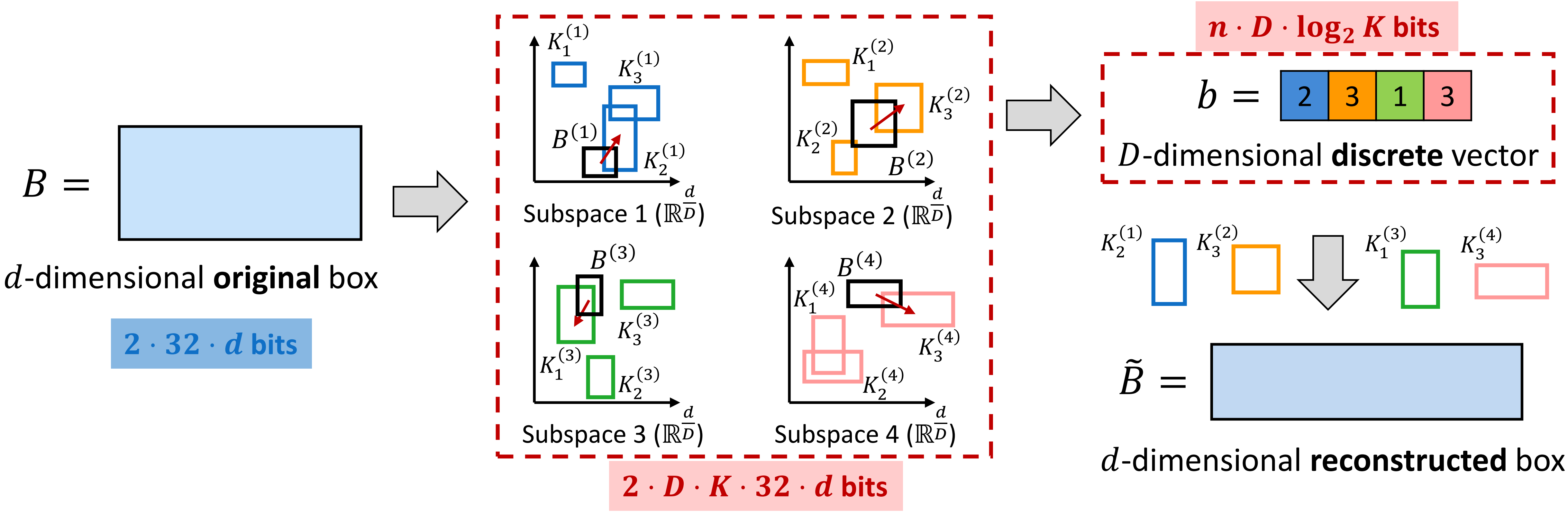
Set2Box+: Even More Concise & Accurate (cont.)

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



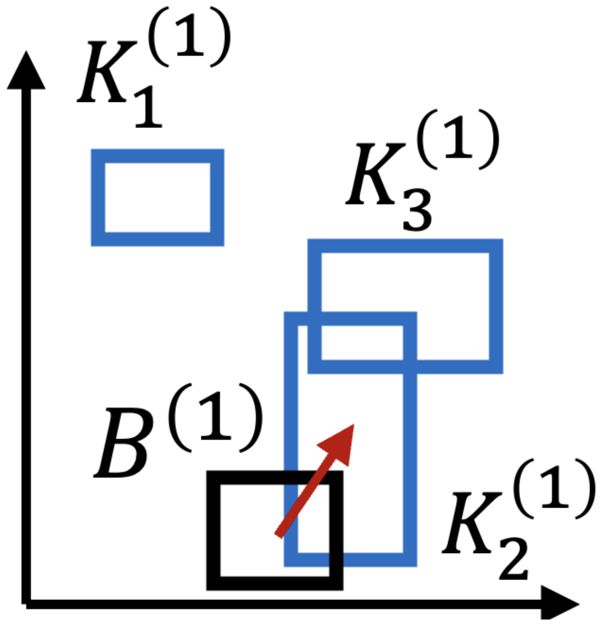
Set2Box⁺: Even More Concise & Accurate (cont.)

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



Set2Box^{+}: Even More Concise & Accurate (cont.)

- How does **box quantization** find the **closest** key box?



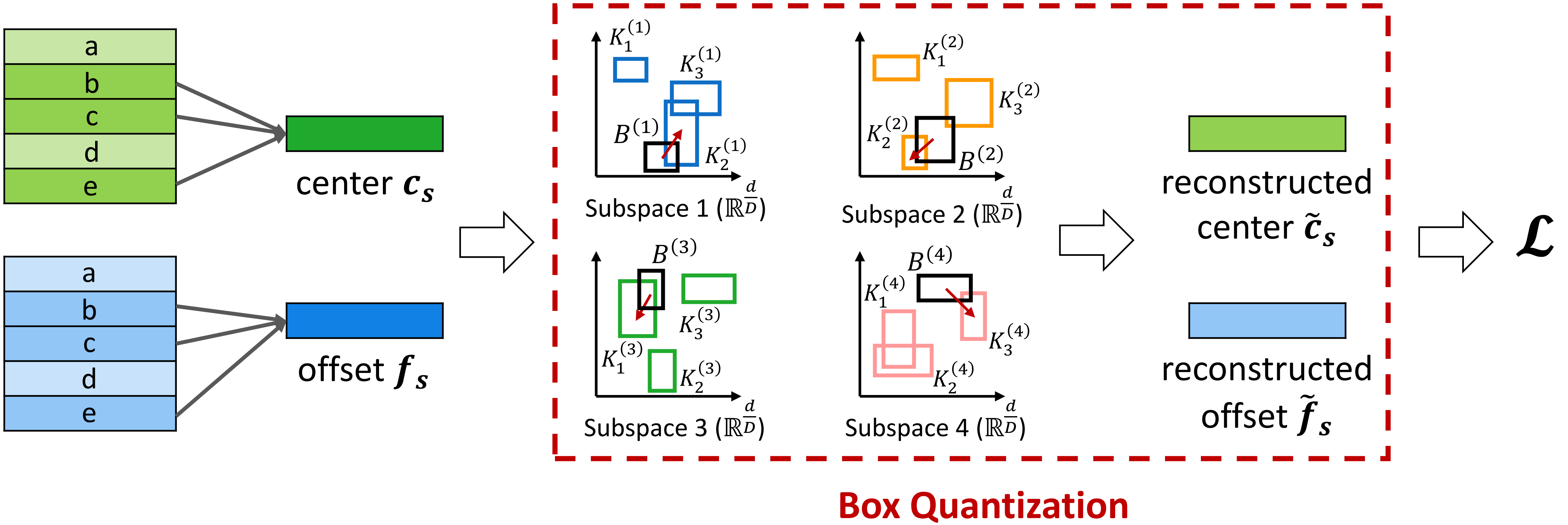
$$\boxed{2 = \arg\max_i \mathbf{BOR}\left(x^{(2)}, K_i^{(2)}\right)}$$

To compute box similarities, we define **Box Overlap Ratio**:

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

Set2Box⁺: Even More Concise & Accurate (cont.)

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



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 $\boldsymbol{O(d)}$ **time** to compute the estimated similarity $\widehat{\mathrm{sim}}(B_s, B_{s'})$, where $\boldsymbol{d}$ **is a user-defined constant** that does not depend on the sizes of s and s'.

Other Details

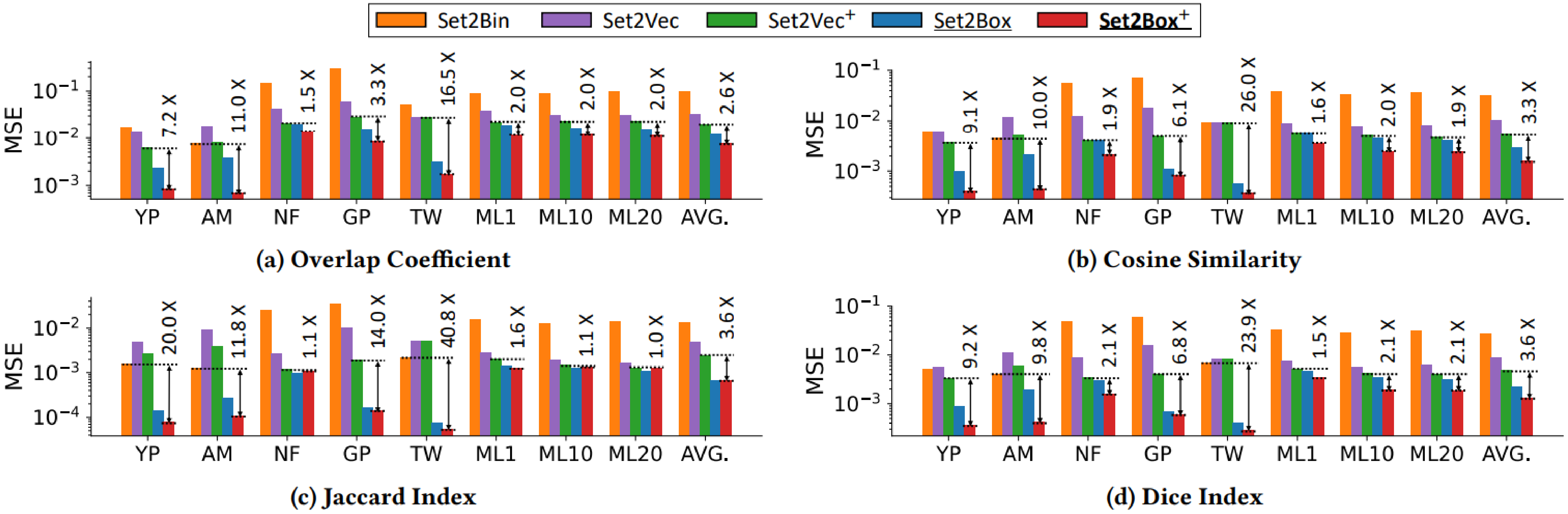
- In the paper, you can find:
  - ✓ Set context pooling
  - ✓ End-to-end discrete code learning
  - ✓ Joint training original and reconstructed boxes
  - ✓ Box smoothing for effective learning

Roadmap

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

Accuracy & Conciseness of Set2Box$^+$

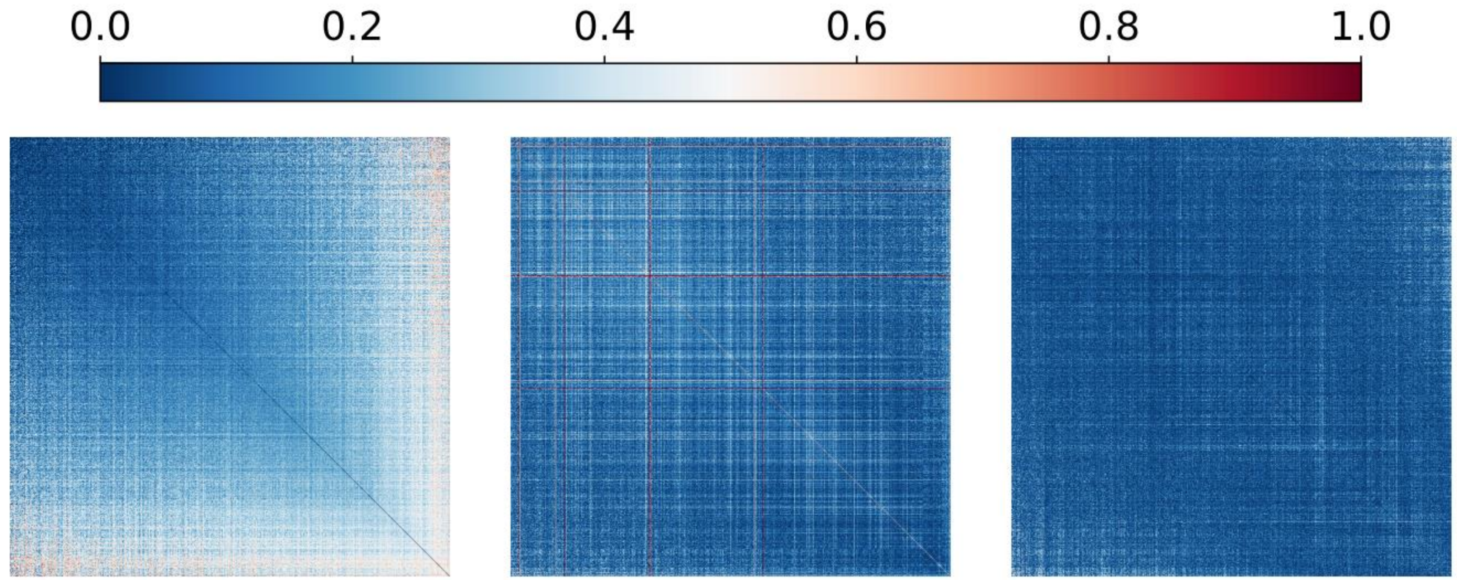
- **Set2Box$^+$** preserves set similarities most accurately compared to baselines.
  - **Set2Box$^+$** gives up to 40.8X smaller estimation error while requiring about 60% fewer bits to encode sets.



(a) Overlap Coefficient (b) Cosine Similarity (c) Jaccard Index (d) Dice Index

Accuracy & Conciseness of Set2Box$^+$ (cont.)

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



(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

Method	OC	CS	JI	DI
Set2Box-PQ	0.0129	0.0028	0.0012	0.0023
Set2Box-BQ	0.0106 (-17%)	0.0023 (-17%)	0.0009 (-26%)	0.0019 (-17%)
Set2Box$^+$	**0.0077** (**-40%**)	**0.0016** (**-44%**)	**0.0007** (**-41%**)	**0.0013** (**-42%**)

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.

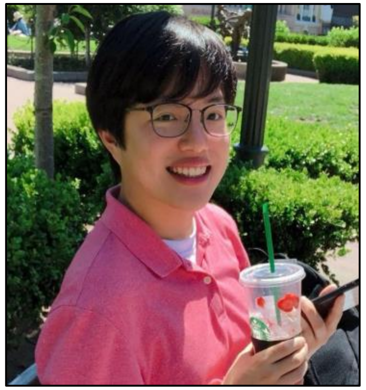
✓ **Versatile**: estimates various set similarity measures with a single set embedding.

Code & datasets: https://github.com/geon0325/Set2Box

Set2Box: Similarity Preserving Representation Learning of Sets

Geon Lee

Chanyoung Park



Kijung Shin