Efficient kNN Join over Dynamic High-dimensional Data

2022 Australasian Database Conference

Never Stand Still | Faculty of Engineering | Computer Science and Engineering

The University of New South Wales, Sydney

Nimish Ukey, Zhengyi Yang, Guangjian Zhang, Boge Liu, Binghao Li, and Wenjie Zhang

Introduction

- **What is kNN-join?**
  - For every object of the query dataset R, finding the k nearest neighbors from another object dataset S

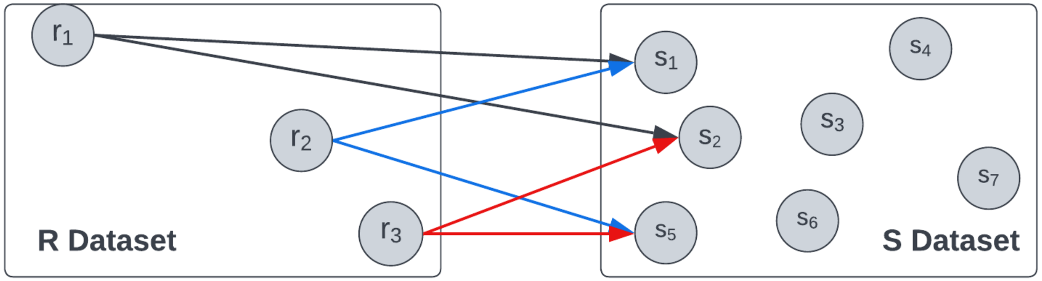
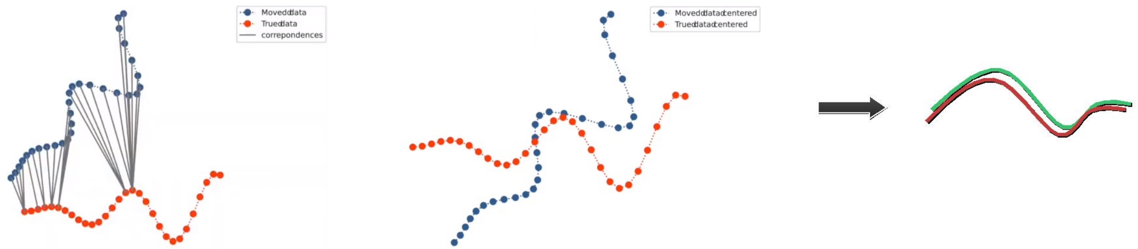


Fig. 1. An example of kNN Join with k = 2.

- **Applications**
  - kNN classification, k-means clustering, outlier detection, similarity search, etc.

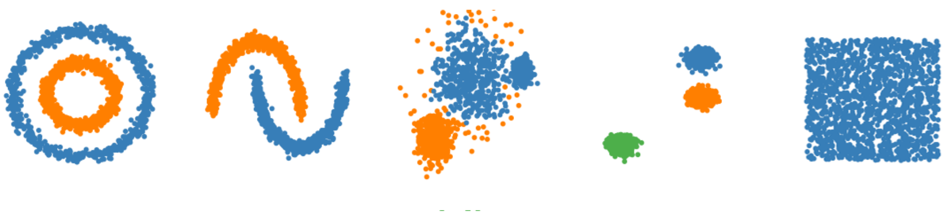
kNN Join Applications

- **Point Cloud Registration**
    - It is the process of finding a spatial transformation that aligns two-point clouds
    - Example: Iterative Closest Point



- **DBSCAN -** Density-Based Spatial Clustering of Applications with Noise

DBSCAN



Introduction

- **What's Dynamic kNN join?**
  - For any item updates, the affected user list must be updated efficiently

- **Applications of Dynamic kNN Join**
  - Recommendation system
  - Feature extraction
  - Video on-demand
  - Social network services, etc

UNSW THE UNIVERSITY OF NEW SOUTH WALES

Related Work

- Existing works focus on static data MuX[1], Gorder[2], iJoin[3].
- Yu, Cui, et al. proposes a high-dimensional kNN Join+ algorithm [4] for dynamically updating the new data points to allow incremental updates on kNN join results.
- Researchers propose the index structure of High-dimensional R-tree [5] on dynamic kNN join. It updates only the affected data points to avoid redundant computation

[1] Böhm, C., Krebs, F.: The k-nearest neighbour join: Turbo charging the kdd process. Knowledge and Information Systems 6(6), 728–749 (2004)

[2] Xia, C., Lu, H., Ooi, B.C., Hu, J.: Gorder: an efficient method for knn join process-ing. In: Proceedings of the Thirtieth international conference on Very large databases-Volume 30. pp. 756–767 (2004)

[3] Yu, C., Cui, B., Wang, S., Su, J.: Efficient index-based knn join processing for high-dimensional data. Information and Software Technology 49(4), 332–344 (2007)

[4] Yu, C., Zhang, R., Huang, Y., Xiong, H.: High-dimensional knn joins with incremental updates. Geoinformatica 14(1), 55–82 (2010)

[5] Yang, C., Yu, X., Liu, Y.: Continuous knn join processing for real-time recommendation. In: 2014 IEEE International Conference on Data Mining. pp. 640–649. IEEE (2014)

Research Problem

- **Efficient kNN Join over Dynamic High-dimensional Data**

- **Given** : In d-dimensional space, query point set/user dataset $U = \{u_0, u_1, u_{2} \ldots, u_n\}$ and object point/item dataset set $I = \{o_0, o_1, o_{2} \ldots, o_n\}$ and an integer k**.**

- **Goal** : For the given user dataset U and an item dataset I, our goal is to dynamically find the kNN join results of U in I upon every update of I (i.e., insertions and deletions).

HDR Tree

- **The process of computing PCA:**
  - Let $X[d \times N]$ be a point matrix
  - compute its covariance matrix Y, and then compute the eigenvalues and eigenvectors of Y.
  - Rank the eigenvalues and choose the eigenvectors corresponding to the r largest eigenvalues to form a transformation matrix $V[r \times d]$.
  - Finally, transform a point p to the r-dimensional space by multiplying it with V

Structure of HDR Tree

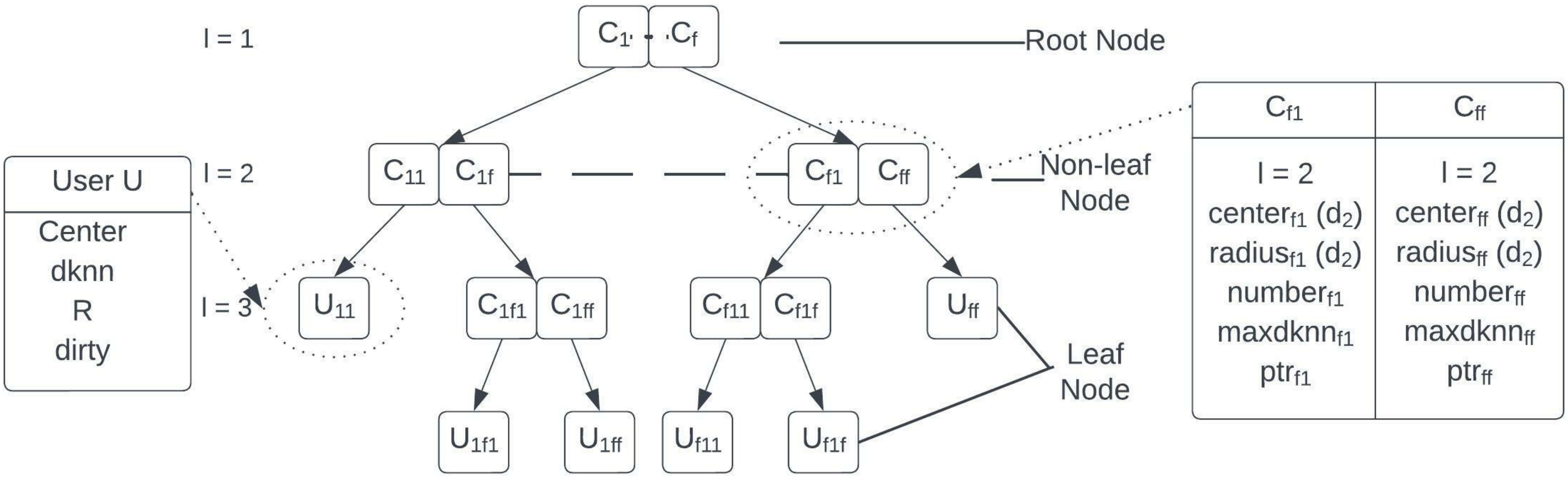


Fig. 2. HDR-tree structure

Our Contributions

- Limitations of the of existing solutions for kNN join over dynamic high-dimensional data
    1. Lack of Support for Batch Updates.
    2. Lack of Support for Deletions.
- Our solutions for the improvement of existing system
    1. We provided the Batch Operations
    2. Lazy Updates.
    3. Optimised Deletions

Lazy Updates

- **Identifying the affected users**
- **mark them as "dirty"**
- **Delaying updates until kNN values of affected users are needed**

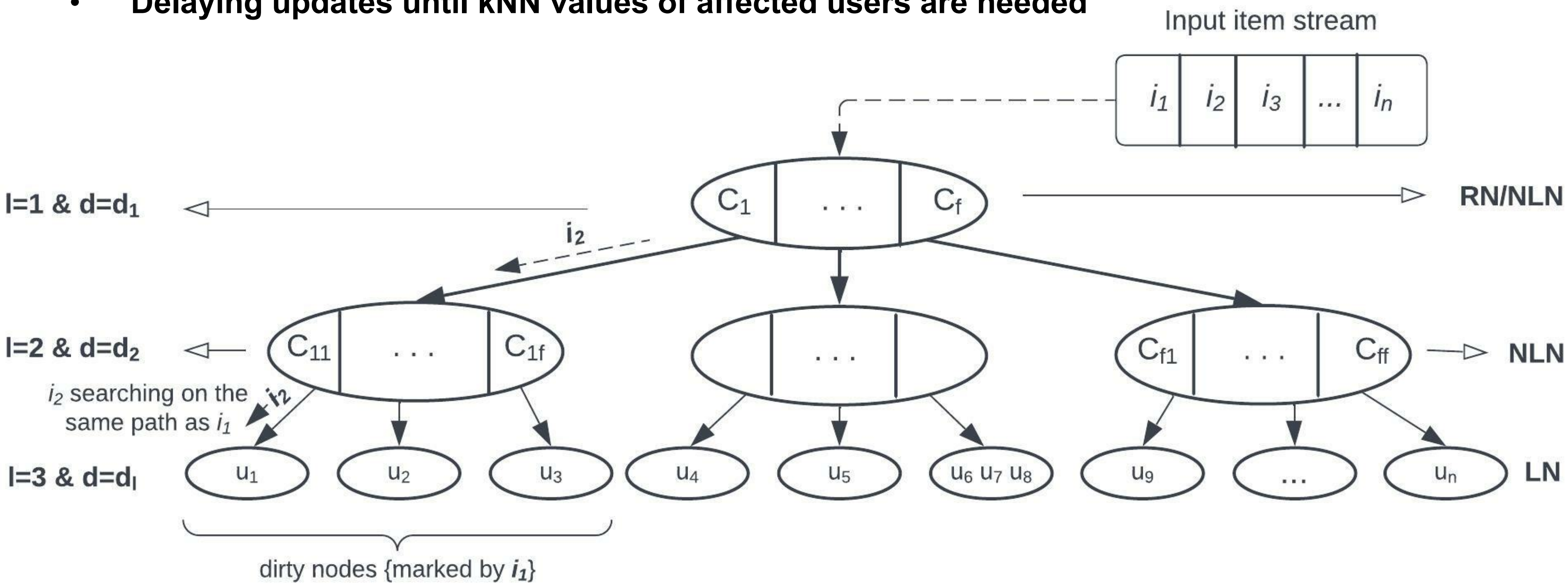


Fig. 2: Example of Lazy Updates on HDR-Tree

Batch Update for Insertion

- Identify the affected users but didn't update them for each of the new items
- We process all updates at the node left before updating the parameters on the internal level to help save the cost.
- Example –
  - Input item stream = $\{i_1, i_2, \ldots, i_n\}$ & user set = $\{u_1, u_2, \ldots, u_m\}$

$i_1 = \{ u_1, u_8, u_{11} \}$

$i_2 = \{ u_8, u_{11}, u_{18}, u_{25} \}$

➢ batch update helps us to save the cost

Batch Update for Deletion

- Example – Lets suppose we consider a batch of 4
  - Input item stream = $\{i_1, i_2, \ldots, i_n\}$ & user set = $\{u_1, u_2, \ldots.., u_m\}$

$i_1 = \{ u_9, u_{10} \}$

$i_2 = \{ u_8, u_{21} \}$

$i_3 = \{ u_8, u_9 \}$

$i_4 = \{ u_9, u_{17} \}$

- ➢ Performing batch update helps us to avoid repetitive computation costs. As shown in the above example, we had to update 3 times earlier for user u_9.

Deletion Optimization

- **Generally, reverse kNN need to perform for delete operation.**
  - Computationally expensive operation

- **Maintained RkNN table for all items to speed up the searching process of affected users**
  - Reduce search cost

Experimental Setup

- **Language:** C++
- **RAM:** 12GB
- **Processor:** Intel Core i5-4210U 2.4GHz
- **OS:** Windows 10
- **Dataset:** NUS-WIDE Image Dataset [6]

- **Default values:**
  - window size - 200,000
  - size of user set - 50,000
  - K - 10
  - Dimension - 128

[6] Chua, T.S., Tang, J., Hong, R., Li, H., Luo, Z., Zheng, Y.: Nus-wide: a real-world web image database from national university of Singapore. In: Proceedings of the ACM International Conference on Image and Video Retrieval, pp. 1–9 (2009)

Experimental Results

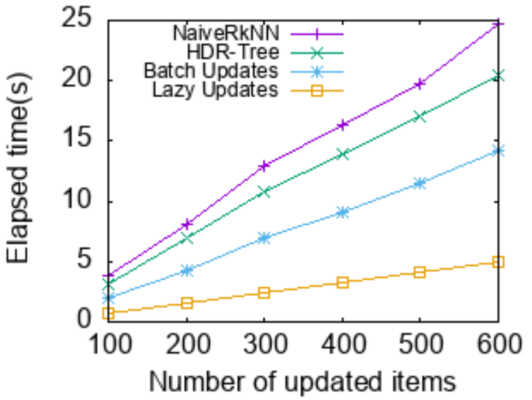


Fig. 3. Vary number of updated items

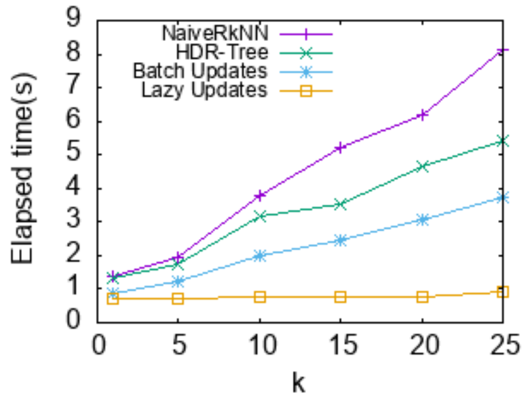


Fig. 4. Vary number of k

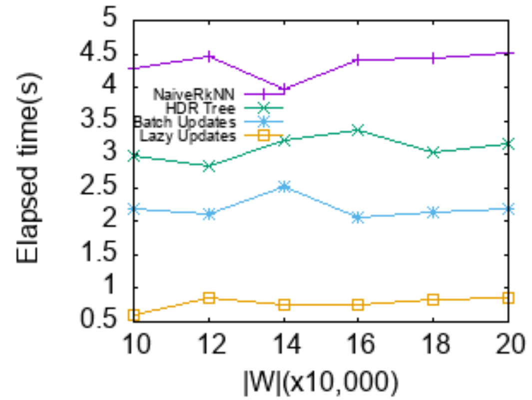


Fig. 5. Vary number of |W|

UNSW
THE UNIVERSITY OF NEW SOUTH WALES

Experimental Results

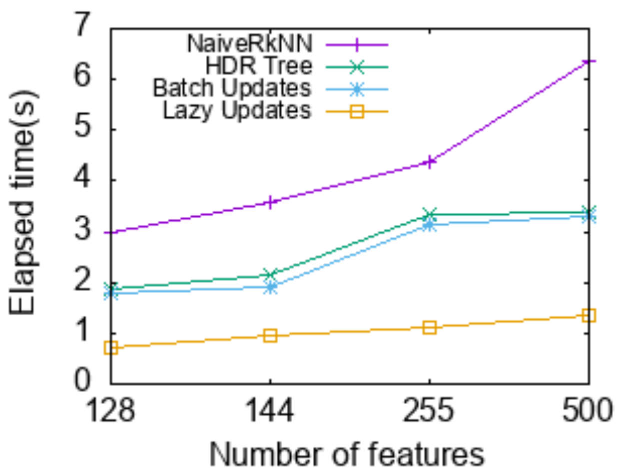


Fig. 6. Vary number of features

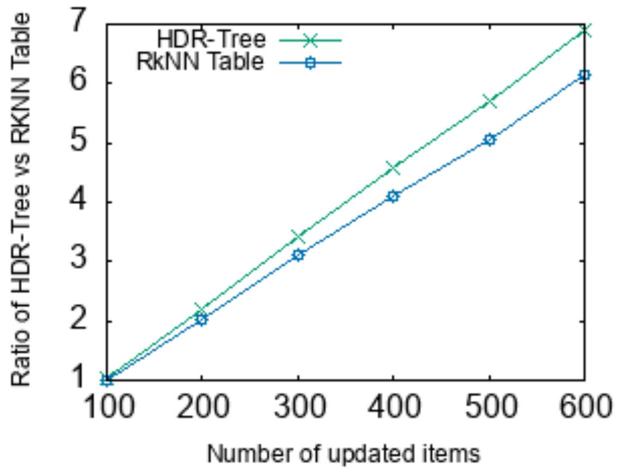


Fig. 7: HDR Tree vs RkNN Table in Deletion

Thank You