# Efficient KNN Join Algorithm for spark

THE UNIVERSITY OF SYDNEY

Ramanathan Sivagurunathan

Dr. Ying Zhou

School of Information Technologies

## Background

### KNN Join

- KNN Join is a classification / regression algorithm
- For every vector in R dataset, [$V_{t1}$, $V_{t2}$ … $V_{tn}$], we find K nearest vectors based on distance in S dataset [$v_1$, $v_2$ … $v_k$] and classify $V_{tx}$ based on neighbour vectors' class.
- Complexity = $O(|R| * |S|)$

### Spark

- Spark is large scale data processing framework with in memory primitives
- Designed for high scalability and performance
- About 100 times faster than Hadoop Mapreduce when the data can fit into memory

## Problem

- High Complexity => Unusable for Large Dataset
- How to improve the overall running time of the algorithm ?

## Previous Works

- Focused only on reducing comutational complexity
- Designed only for single node system
- Algorithms were focusing on creating a spatial index for the data. This means there is a huge initial cost and not scalable for a large dataset which cannot fit into memory.
- Only one research work done on creating a distributed algorithm. It uses hadoop Mapreduce as its framework.

  **Disadvantage:**
  1. High data replication
  2. High disk usage
  3. Does not scale for high dimensions due to number of replications

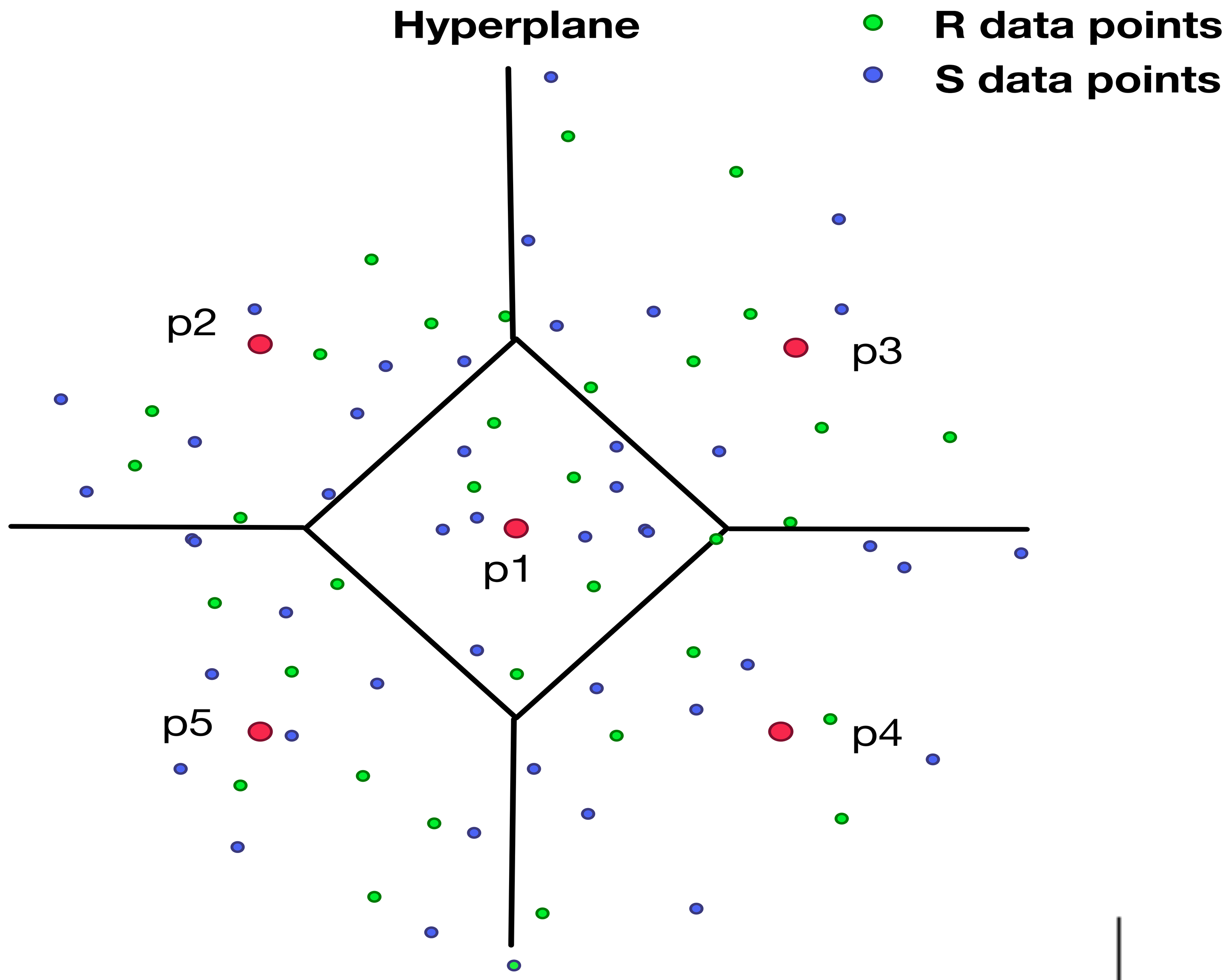
## Solution & Contribution

- Designed a Voronoi Partition based distributed algorithm for Spark

**Advantages**

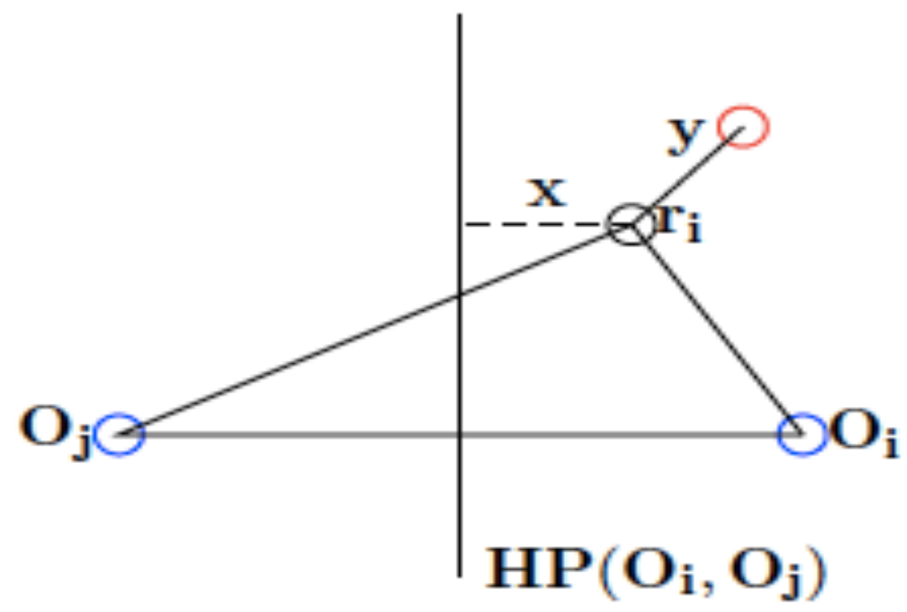
1. Lower computational complexity
2. Distributed algorithm
3. Iterative and Incremental Algorithm
4. Uses Spark framework
5. Scalable for large datasets
6. Minimal data replication and hence minimal usage of disk & memory

## Voronoi Partition




$$x = \frac{|p, O_i|^2 - |p, O_j|^2}{2 \times |O_i, O_j|}$$

y = distance of farthest neighbour




## Algorithm

1. Select Random Pivots
2. Closest Vector to a pivot form a partition in both R and S.
3. Self Join: For any partition in R, find KNN S in the same partition.
4. If distance to farthest neighbour for a vector *y* is less than the distance to HP*(x)* then K Found Else Nearest partition found. (Refer fig above)
5. Move the vector to first nearest partition and find & update KNN.
6. Check if the new distance *y* is less than the distance to HP(x) and Update the nearest partition list.
7. Repeat Step 5 and 6 until KNN found for all Vectors
8. *Additional Optimization*: If the number of replication is less then we can replicate to all the partition and then combine the results after wards. This speeds up the process significantly

## Experiment Results(On a 10 Node 8core 16G Ram Cluster)

1. About 100 times faster than Brute force for 1M x 1M dataset and Estimated 800-1000 times faster for 11M x 11M
2. 1M x 1M join takes 2 minutes and 11M x 11M Join takes 26 mins
3. For dimensions more than 12 performance drops, this is because of curse of dimensionality

## Future Work

Improve performance of the algorithm at higher dimensions

**References**

Index-driven similarity search in metric spaces http://dl.acm.org/citation.cfm?id=958948

When is "nearest neighbor" meaningful? http://link.springer.com/chapter/10.1007/3-540-49257-7_15

Efficient parallel kNN joins for large data in MapReduce http://dl.acm.org/citation.cfm?id=2247602