K-means Split Revisited: Well-grounded Approach and Experimental Evaluation

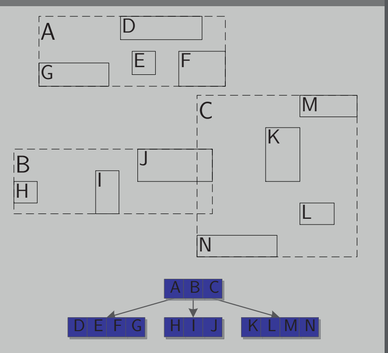
Valentin Grigorev[1] and George Chernishev[1, 2]

[1]Saint Petersburg State University, [2]JetBrains Research

{valentin.d.grigorev, chernishev}@gmail.com

1. R-tree

R-tree [5] and its variants [6] are popular data structures used for multidimensional indexing in many industrial applications [2].



2. R-tree split

- It is a very performance sensitive part of the R-tree;
- It can be performed in multiple ways, the quality of resulting tree may differ by several times;
- The problem was proved to be *NP*-hard, so all of the existing algorithms rely on some heuristic;
- Several dozens of algorithms exist.

In our study, we are interested in k-means split [3], based on idea of hyper-rectangle clustering.

3. Problem of hyper-rectangle clustering [4]

- X is the finite multiset of hyper-rectangles in $\mathbb{R}^d$
- $\{X_j\}_{j=1\ldots k}$ is a partition of X
- $X_j = \{\mathbf{x_1^j}, \ldots, \mathbf{x_{n_j}^j}\}$
- D is a distance between hyper-rectangles

Centroid $\mathbf{c}$ of cluster X_j is the hyper-rectangle that minimizes function:

$$S_p(X_j, \mathbf{c}) = \begin{cases} (\sum_{i=1}^{n_j} D(\mathbf{x_i^j}, \mathbf{c})^p)^{1/p} & for\ p < \infty \\ \max\limits_{i=1\ldots n_j} D(\mathbf{x_i^j}, \mathbf{c}) & for\ p = \infty \end{cases} \quad (1)$$

So, clustering of hyper-rectangles is the following optimization problem:

$$\arg\min_{X_j \subset X} \sum_{j=1}^{k} S_p(X_j, \mathbf{c_j}) \quad (2)$$

where $\mathbf{c_j}$ is a centroid of X_j.

4. K-means algorithm

Step 1 [Initialization]
Arbitrarily choose k objects as the initial cluster centroids.

Step 2 [Assign objects to clusters]
Assign each object to the cluster with the minimal distance to the corresponding centroid

Step 3 [Update cluster centroids]
Recalculate cluster centroids on the new entry distribution

Step 4 [Repeat] Repeat steps 2-3 until no change

5. K-means specialization

K-means is an iterative algorithm for solving clustering problem. It is general enough to work with abstract objects, for which distance and centroid functions are defined. As we can see, in case of hyper-rectangles we should choose parameter p and functions D and c in a consistent fashion.

6. Original approach [3]

- $p = 2$
- Distance function

$$D(\mathbf{x_1}, \mathbf{x_2}) = \sqrt{\sum_{i=1}^{d} (max(b_1^j, b_2^j) - min(a_1^j, a_2^j))^2}, \quad (3)$$

where $\mathbf{x_i} = \prod_{j=1}^{d} [a_i^j, b_i^j]$.

- Centroid function

$$\mathbf{c_j}(\mathbf{x_1}, \ldots, \mathbf{x_n}) = \frac{\sum_{i=1}^{n} \frac{a_i^j + b_i^j}{2} \mu(\mathbf{x_i})}{\sum_{i=1}^{n} \mu(\mathbf{x_i})}, \ j = 1..d \quad (4)$$

where μ is a volume of hyper-rectangle.

7. Problems of original approach

Distance function (3):

- isn't really a metric because it violates identity of indiscernibles axiom and triangle inequality.

Centroid function (4):

- is not consistent with centroid definition of a point set;
- is not consistent with degenerate rectangles, since their contribution to the formula is zero;
- if all rectangles are degenerate, the formula instructs us to divide by zero which is unacceptable.

In total:

- It is not clear whether this centroid function is consistent with chosen metric, i.e. whether it satisfies (2) or minimizes any other cost function.
- It is not proven that the k-means algorithm converges under these conditions.

8. Solution

To the best of our knowledge there were no clustering algorithms specialized for hyper-rectangles when original k-means split was being developed.

Proper solution of aforementioned problems was proposed by Marie Chavent et al. in the series of papers about clustering of hyper-rectangles, that was summarized in [4]. In our paper we adopt those results to k-means split.

There are a lot of metrics that could be used for hyper-rectangles. We are interested in two types:

- Hausdorff distance based on L_∞
- coordinate-wise distances, based on the following distances between intervals:
  - L_1 combination of Hausdorff distances
  - L_2 combination of L_2 distances
  - L_∞ combination of Hausdorff distances

In all these cases (with $p = \infty, 1, 2, \infty$, respectively) we have explicit formulas for distance and centroids, so they can be computed efficiently.

9. Test suite

We have implemented k-means split on the base of "cube" extension for the PostgreSQL DBMS. This R-tree implementation is based on the GiST interface, thus we were restricted to 2-means algorithm.
Experimental evaluation was performed on PC with following parameters:

- 4-core processor Intel®Core™i7 2.20GHz
- 8 Gb RAM
- Linux Ubuntu 14.04.3 LTS
- PostgreSQL 9.4.5

For performance evaluation we have used some of "rea" datasets from the benchmark [1] used in recent R-tree studies [2].

10. Experiments

The k-means split variations with different metrics were evaluated. Version with coordinate-wise distance based on L_2 combination of L_2 distance between intervals was found to be the most efficient:

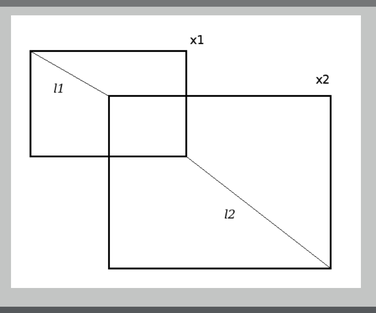
$$D(\mathbf{x_1}, \mathbf{x_2}) = \sqrt{\sum_{j=1}^{d} (a_1^j - a_2^j)^2 + (b_1^j - b_2^j)^2}$$

$$\widetilde{\mathbf{c}}_\mathbf{j} = \left[\frac{1}{n}\sum_{i=1}^{n} a_i^j, \frac{1}{n}\sum_{i=1}^{n} b_i^j\right], \ j = 1..d$$

where $\mathbf{x_i} = \prod_{j=1}^{d} [a_i^j, b_i^j]$.

Finally, we compared the winning variant with the PostgreSQL implementation of Guttman's Quadratic Split [5]. The results are presented in the tables 12 and 13.
For each dataset/query/split combination we executed benchmark 10 times and calculated 95% confidence interval.

11. Formula illustration



$$D(\mathbf{x_1}, \mathbf{x_2}) = \sqrt{l_1^2 + l_2^2}$$

12. Experimental results: build time (seconds)

dataset	PG default	k-means	speedup
rea02	$60.5 \pm 1.3\%$	$36.8 \pm 0.5\%$	1.6
rea03	$519.5 \pm 1.6\%$	$197.6 \pm 0.5\%$	2.6
rea05	$20.3 \pm 0.6\%$	$8.6 \pm 0.9\%$	2.4
rea09	$1.9 \pm 0.5\%$	$0.8 \pm 0.6\%$	2.4

13. Experimental results: queries (seconds)

		PG default	k-means	speedup
rea02	QR1	$16.8 \pm 1.3\%$	$16.1 \pm 1.3\%$	1.0
	QR100	$7.4 \pm 1.3\%$	$7.3 \pm 0.2\%$	1.0
	QR1000	$19.7 \pm 0.6\%$	$19.8 \pm 0.7\%$	1.0
rea03	QR1	$1317.9 \pm 6.2\%$	$838.6 \pm 3.9\%$	1.6
	QR100	$297.6 \pm 3.9\%$	$180.9 \pm 2.5\%$	1.6
	QR1000	$290.0 \pm 2.0\%$	$216.5 \pm 1.0\%$	1.3
rea05	QR1	$31.5 \pm 3.1\%$	$11.9 \pm 5.8\%$	2.7
	QR100	$14.7 \pm 4.0\%$	$4.6 \pm 2.9\%$	3.2
	QR1000	$12.5 \pm 3.3\%$	$6.0 \pm 1.5\%$	2.1
rea09	QR1	$1.9 \pm 3.3\%$	$1.7 \pm 3.0\%$	1.1
	QR100	$1.0 \pm 0.9\%$	$1.0 \pm 2.1\%$	1.1
	QR1000	$1.3 \pm 0.8\%$	$1.2 \pm 0.9\%$	1.1

14. Results

Results show that our k-means generally outperforms standard Guttman's quadratic split in terms of build time as well as query processing. The best improvement on the processing stage was achieved on datasets of medium dimensionality (3 and 5). Build time improved almost twice on all datasets.

15. Summary

- We presented a well-grounded algorithm for R-tree split based on k-means algorithm
- We have approached the rectangles clustering problem as the problem of selection of consistent configuration of p, distance and centroid functions
- We have selected a number of such configurations using the recent advances in geometry
- We have evaluated the found configurations on real data distributions of the standard multidimensional indexing benchmark
- Notable improvement was achieved in both building and querying time

16. Acknowledgments

We would like to thank Kirill Smirnov for his valuable comments on this work.

17. References

[1] N. Beckmann and B. Seeger. A benchmark for multidimensional index structures. http://www.mathematik.uni-marburg.de/~rstar/benchmark/distributions.pdf, 2008.

[2] N. Beckmann and B. Seeger. A revised R*-tree in comparison with related index structures. ACM SIGMOD, pages 799–812, 2009.

[3] S. Brakatsoulas, D. Pfoser, Y. Theodoridis. Revisiting R-Tree Construction Principles. ADBIS, pages 149–162, 2002.

[4] M. Chavent and J. Saracco. On central tendency and dispersion measures for intervals and hypercubes. *Communications in Statistics—Theory and Methods*, 37(9):1471–1482, 2008.

[5] A. Guttman. R-trees: a dynamic index structure for spatial searching. *SIGMOD Record*, 14(2):47–57, 1984.

[6] A. N. Papadopoulos et al. R-Tree (and Family). In L. Liu and M. T. Özsu, editors, *Encyclopedia of Database Systems*, pages 2453–2459. 2009.