Nearest Neighbor Search In Hyperspectral Data Using Binary Space Partitioning Trees

Evgeny Myasnikov
Geoinformatics and Information Security department
Samara National Research University
Samara, Russia
mevg@geosamara.ru

***Abstract*—Fast search of hyperspectral data is crucial in many practical applications ranging from classification to finding duplicate fragments in images. In this paper, we evaluate two space partitioning data structures in the task of searching hyperspectral data. In particular, we consider vp-trees and ball-trees, study several tree construction algorithms, and compare these structures with the brute force approach. In addition, we evaluate vp-trees and ball-trees with four similarity measures, namely, Euclidean Distance, Spectral Angle Mapper Bhattacharyya Angle, and Hellinger distance.**

Keywords—binary space partitioning trees; ball-tree; vp-tree; Spectral Angle Mapper; Bhattacharyya Angle, Hellinger distance; Spectral Information Divergence

I. Introduction

Nowadays hyperspectral data is extensively used in many areas such as agriculture, chemistry, medicine, etc. Its high spectral resolution allows solving many important problems with improved quality. However, the high dimensionality of hyperspectral data is also the reason for significant time casts for data processing.

Nearest neighbor search is one of the most often used procedures in hyperspectral image analysis, and the query processing time is crucial in many problems ranging from classification to finding duplicate fragments in hyperspectral images.

To improve nearest neighbor query processing, binary space partitioning trees are often used in computer science. Kd-tree [1] is one of the old and most well-known space partitioning structures based on the recursive splitting of space with hyperplanes orthogonal to the coordinate axes. However, there are other binary space partitioning trees, like ball-trees [2] and vp-trees [3, 4], which could have advantages over kd-trees in high dimensional spaces [5].

In this paper, we evaluate ball-trees and kd-trees, study several algorithms, which can be used in the construction of these structures, and compare them to the brute force approach.

Most often, the nearest neighbor search is performed using the Euclidean distance. Meanwhile, in hyperspectral image analysis, other dissimilarity measures, such as spectral angle mapper (SAM) [6] and spectral information divergence (SID) [7] are often used. Unfortunately, SID cannot be used in ball- or vp-trees due to its properties, and we consider Bhattacharyya Angle [9] and Hellinger distance [10] instead. So in this paper, we evaluate ball- and vp-trees with all four measures, namely, Euclidean distance, SAM, Bhattacharyya Angle, and Hellinger distance.

The reported study was funded by RFBR according to the research project no. 20-37-70053.

The paper is structured as follows. The next section introduces binary space partitioning trees and measures used in the study. Section 3 contains the results of experimental studies and discussion. The paper ends with the conclusion and the list of references.

II. Methods

A. Binary Space Partitioning Trees

As it was said in the introduction, in this paper, we evaluate two binary space partitioning trees: ball-tree and vp-tree.

Ball-tree [2] is a binary tree that splits a multidimensional space using hyperplanes orthogonal to a pair of chosen anchor points. The tree is built recursively. At each step, two anchor points v_1 and v_2 are selected from the entire set S of points corresponding to a current tree node. Then all points $v \in S$ of the set are divided into two subsets (child nodes) S_1 and S_2 according to the closeness to the selected anchor points:

$$S_1=\{p \in S \mid d(p,v_1) \le d(p,v_2)\},$$
$$S_2=\{p \in S \mid d(p,v_2) < d(p,v_1)\}.$$

The common approach to choose anchor points v_1 and v_2 is to maximize the distance between them. First, the centroid of the entire set of points is determined. The point furthest from the centroid is then selected as the first anchor point. The second anchor point is selected as the point farthest from the first one. In this paper, we stop the recursive partitioning of a particular tree node, if there are n (or fewer) objects in this node (we denote this approach as *Ball-<n>*).

Vp-tree [3, 4] is a binary tree splitting a multidimensional feature space using hyperspheres. The splitting is done in recursive order. At each stage, we choose one vantage point

$v \in S$ from the set of points S according to some algorithm, and the entire set of points is divided into two subsets S_1 and S_2 of the same cardinality:

$$S_1 = \{p \in S \setminus \{v\} \mid d(p,v) \le d_m\},$$
$$S_2 = \{p \in S \setminus \{v\} \mid d(p,v) > d_m\},$$

where d_m id the median value among distances $d(p,v)$, $p \in S$.

The standard approach to choose vantage points is based on random sets. At each stage, two random subsets of points are formed. The first subset contains potential vantage point candidates. The second one contains test points by which candidates are evaluated. A candidate point, for which the variance of distances to test points is maximized, becomes a vantage point. Hereinafter, this approach is denoted as *VP-RS<n>*, where *<n>* is the cardinality of the random subsets used to choose vantage points.

Besides the standard approach, three simple additional approaches to vantage point selection were also used in this paper. The simplest one is just to select the first point from the set as a vantage point. This approach is denoted as *VP-FP* further. Another algorithm is to select the point with the greatest length of the corresponding vector (denoted as *VP-LV*). Finally, we selected the farthest point from the centroid of a point set as a vantage point (denoted as *VP-FFC*).

Despite the construction algorithm, the vp-tree is balanced, while the ball-tree is not necessarily balanced and its structure depends on the distribution of data. Note that search speed is determined not only by balance but also by the ability of the data structure to take into account the distribution of data.

To execute a query by the nearest neighbor, we traverse the created tree structure, starting from its root, and visiting first those nodes where the nearest neighbor points are most likely to appear. During the traversal, we can skip nodes or subtrees if they are further than the distance to the nearest point already found. To make a decision in such cases, the triangle inequality is used.

If it is necessary to find k nearest neighbors, the candidates are stored in an ordered list and the distance to the k-th element is taken as the current estimate of the distance to the nearest neighbors.

B. Hyperspectral Similarity Measures

Usually, Euclidean distance is used as a metric in the described above space partitioning data structures. But in hyperspectral image analysis, other similarity measures are often used. The most well-known measure among them is the spectral angle mapper (SAM) measure [6]:

$$\theta(x_i, x_j) = \arccos\left(\frac{x_i^T x_j}{\|x_i\| \|x_j\|}\right).$$

Here x_i and x_j are two vectors corresponding to i-th and j-th pixels in a hyperspectral image. Thus, the SAM measures the spectral dissimilarity as the angle between two spectral signatures x_i and x_j in amplitude independent manner that gives some independence to the variations in illumination.

Another well-known non-Euclidean measure used with hyperspectral images is the spectral information divergence (SID) measure, which is also called Jeffrey's divergence or symmetric Kullback-Leibler divergence. This measure is given by the formulae [7]:

$$SID(x_i, x_j) = D(x_i \| x_j) + D(x_j \| x_i),$$
$$D(x_i \| x_j) = \sum_{k=1}^{M} p_k(x_i) \log\big(p_k(x_i) / p_k(x_j)\big),$$
$$p_k(x_i) = \frac{x_{ik}}{\sum_{l=1}^{M} x_{il}}.$$

It is worth noting that the measure should be a true metric to be used in the described binary space partitioning trees. It means that positivity, identity, symmetry, and triangle inequality conditions should be satisfied:

$$d(x, y) \ge 0,$$
$$d(x, y) = 0 \Leftrightarrow x = y,$$
$$d(x, y) = d(y, x),$$
$$d(x, y) \le d(x, z) + d(z, y).$$

Unfortunately, Jeffrey's divergence is not a metric since it does not satisfy the triangle inequality [8], while other requirements are satisfied. The violation of the triangle inequality makes it impossible to perform exact nearest neighbor queries as the correct points can be mistakenly discarded during the search.

For the above reason, in this paper, we adapt the Bhattacharyya angle [9]:

$$BCa(x_i, x_j) = \arccos\left(\sum_{k}^{M} \sqrt{p_k(x_i) p_k(x_j)}\right).$$

and Hellinger distance [10]:

$$HD(x_i, x_j) = \sqrt{1 - \sum_{k}^{M} \sqrt{p_k(x_i) p_k(x_j)}}.$$

in binary space partitioning trees construction and study the structures built with the described metrics.

III. Experiments

To evaluate the described space partitioning data structures and metrics, we performed several experiments. We used probably the most well-known open hyperspectral scene Indian Pines [11] as a source of hyperspectral data. All the algorithms were implemented in C++. The research was carried out using an ordinary laptop based on Intel Core i3-6100U CPU @ 2.3 GHz. Due to a limited volume, we restrict the reported results with the scene [11] and time characteristics, while other indicators can be used, such as the number of comparisons, distance calculations, etc.

In our first experiment, we studied the time required to build the trees from the above dataset using different settings, algorithms, and metrics. The results are depicted in Fig.1.

As can be seen, the construction time substantially depends (grows) on the number of data points (10, 20, 50, 100) used in vantage point selection, when the standard approach (*VP-RS*) is used. The minimal construction time is provided by the *VP-FP* algorithm as there is no need to look through data to select a vantage point. *VP-LV* algorithm is a bit slower as it needs one pass through data. *VP-FFC* needs several passes and still fast compared to other techniques.

The construction time of the ball-tree decreases with the growth of tree nodes that is expected. This time changes relatively slow and it is comparable to that of *VP-RS* technique with small random sets.

In the next experiment, we measured the average time taken by the structures to perform the nearest neighbor search. To conduct this experiment, we took the constructed trees, searched for the second nearest neighbor (the first nearest neighbor was the query point itself), and averaged the timings. The results are shown in Fig.2.

As you can see, the VP-tree shows almost two times better timings than the ball-tree despite algorithms used in tree construction. It is worth noting that fast and simple construction approaches (*VP-FP*, *VP-LV*) are comparable to the standard approach and outperform ball-tree. For the ball-tree, the node size of ten points seems to be a good option as it is a near-optimal value for all described metrics. The minimal timings for the vp-tree correspond to relatively large test and candidate sets in *VP-RS* algorithm. It is difficult to make the recommendation to use large sets due to much higher construction time casts.

To compare the described data structures to alternative solutions, we provide the results for the kd-tree [1] (with standard construction algorithm and Euclidean distance) and brute force approach. The results of this experiment are shown in Fig.3.a. Please, note the logarithmic time scale.

As can be seen, the vp-tree provides minimal timings among all the considered algorithms and with all four considered metrics. It is also obvious that any considered structure allows getting considerable performance gain over the brute force approach.

To give a simple practical example of the studied data structures, we conducted an experiment with per-pixel classification using the nearest neighbor classifier. We split the overall set of points having the ground-truth classification into train and test sets in 60:40 proportions. We used the training set to construct considered data structures, and for each point from the test set, we performed the classification by the nearest neighbor. We used the accuracy defined as the fraction of the correctly classified pixels from the test set as a quality indicator. The result of this experiment is shown in Fig. 3.b.

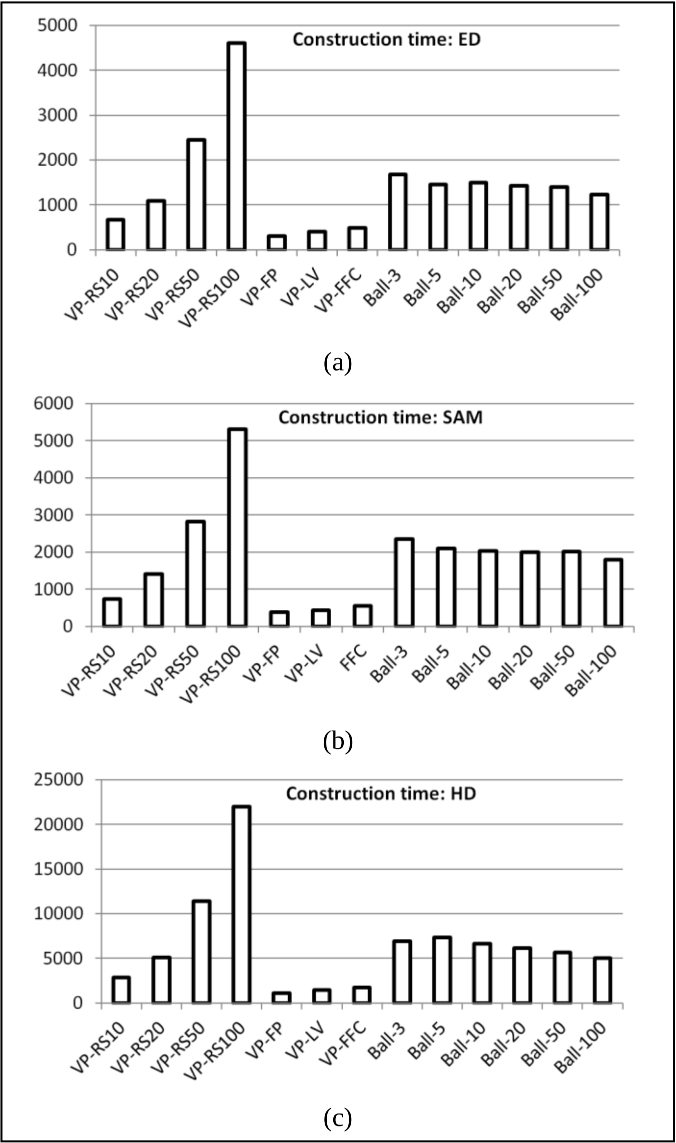

Fig. 1. The construction time (in milliseconds) of different binary space partitioning trees and settings for the Euclidean distance (a); Spectral angle (b), and Hellinger distance (c), in milliseconds. We do not show the results for the Bhattacharyya angle here as they are similar to (c).

As it can be seen, the best results are obtained by Bhattacharyya angle (*BCa*), Hellinger distance (*HD*), and Spectral information divergence (*SID*). While the latter could not be used with the ball- or vp-trees, we included it into comparison to see if it can be replaced by *BCa* and *HD* metrics in such cases.

According to the presented results, *BCa* and *HD* provided similar results both in timings and classification accuracy. The accuracy of *SID* is imperceptibly above *BCa* and *HD*, hundredths of a percent differ. According to this experiment, there is a sense to use *BCa* or *HD* instead of *SID* as one can obtain performance gain by accelerating data search using vp-trees. We included the timing for *SID* using the brute force approach (the rightmost column in Fig.3.a). This result can be compared to the vp-tree search using *BCa* or *HD* (third and fourth columns in the leftmost group in Fig. 3.a). As can be seen, the vp-tree with *BCa* or *HD* is approximately two orders of magnitude faster than the brute-force approach with *SID*.

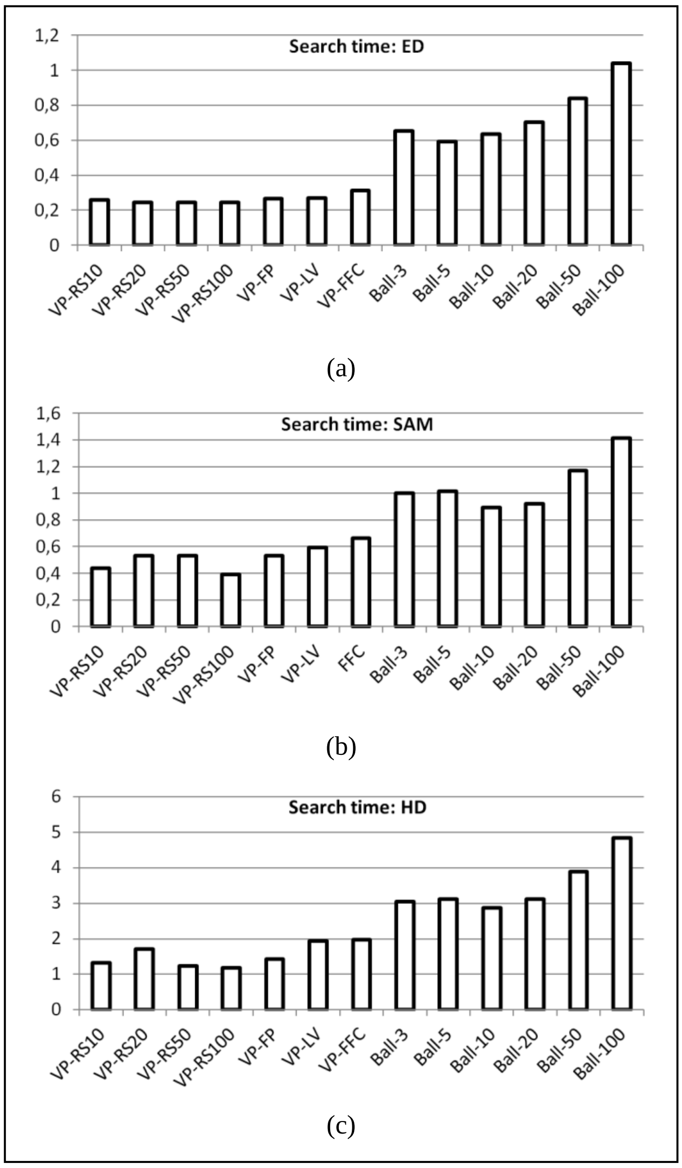

Fig. 2. Search time (in milliseconds, per one query) using different binary space partitioning trees and settings: Euclidean distance (a); Spectral angle (b), and Hellinger distance (c), in milliseconds. We do not show the results for the Bhattacharyya angle here as they are similar to (c).

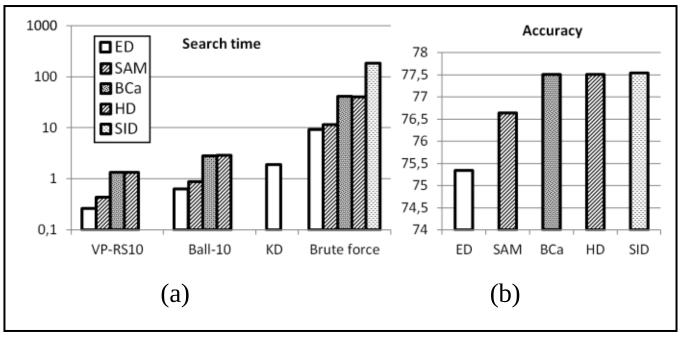

Fig. 3. Search time (in milliseconds, per one query) for different nearest neighbor search algorithms (a); classification accuracy for Indian Pines hyperspectral image and different similarity measures (b).

Conclusion

In this paper, we studied two space partitioning data structures in the task of nearest neighbor search in hyperspectral data. In particular, we considered vp-trees and ball-trees, evaluated the construction algorithms, and compared these structures to the k-d tree and brute force approach. We noted that the SID measure cannot be used with the considered data structures and proposed to use Bhattacharyya Angle and Hellinger distance in addition to common Euclidean Distance and Spectral Angle Mapper. Experimental studies showed that the use of the proposed approach allowed improving the nearest neighbor search time up to several orders of magnitude.

References

[1] J.L. Bentley, “Multidimensional binary search trees used for associative searching,” Communications of the ACM, vol. 18(9), pp. 494-546, 1975.

[2] Linde, Y., Buzo, A., Gray, R.M.: An algorithm for vector quantizer design. IEEE Transactions on Communications 28(1) (January 1980) 84–94

[3] Uhlmann J. Satisfying General Proximity/Similarity Queries with Metric Trees // Information Processing Letters, 40 (4). 1991.

[4] Yianilos Data structures and algorithms for nearest neighbor search in general metric spaces // Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms. 1993. pp. 311–321.

[5] N. Kumar, L. Zhang, S. Nayar, “What Is a Good Nearest Neighbors Algorithm for Finding Similar Patches in Images?” Lecture Notes in Computer Science, vol. 5303, pp. 364-378, 2008.

[6] Kruse, F.A., Boardman, J.W., Lefko, A.B., Heidebrecht, K.B., Shapiro, A.T., Barloon, P.J., Goetz, A.F.H.: The Spectral Image Processing System (SIPS) { Interactive Visualization and Analysis of Imaging Spectrometer Data. Remote Sensing of Environment 44, 145{163 (1993)

[7] Chang, C.-I.: Hyperspectral Data Processing: Algorithm Design and Analysis. John Wiley & Sons (2013)

[8] K.T. Abou-Moustafa, F.P. Ferrie, “Note on Metric Properties for Some Divergence Measures: The Gaussian Case,” JMLR: Workshop and Conference Proceedings, vol. 25, pp. 1-15, 2012.

[9] A.K. Bhattacharyya, "On a measure of divergence between two statistical populations defined by their probability distributions,".Bulletin of the Calcutta Mathematical Society, vol. 35, pp. 99–109, 1943.

[10] E. Hellinger, "Neue Begründung der Theorie quadratischer Formen von unendlichvielenVeränderlichen,” Journal für diereine und angewandte Mathematik, vol. 136, pp. 210–271, 1909 (in German).

[11] M.F. Baumgardner, L.L. Biehl, D.A. Landgrebe, "220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992, Indian Pine Test Site 3," Purdue University Research Repository, 2015.