Fast and Accurate Nearest Neighbor Search in the Manifolds of Symmetric Positive Definite Matrices

Ligang Zheng[1], *Guoping Qiu*[2] ,*Jiwu Huang*[3] and *Jiang Duan*[4]
[1] School of Computer Science, Guangzhou University, Guangzhou 510006, China,
[2] School of Computer Science, The University of Nottingham, Nottingham NG8 1BB, UK
[3] College of Information Engineering, Shenzhen University, Shenzhen 518060，China
[4]Southwestern University of Finance and Economics, Chengdu, China

ABSTRACT

In this paper, we present a fast and accurate Nearest Neighbor (NN) search method in the Riemannian manifolds formed by a kind of structured data - symmetric positive definite (SPD) matrices. We use an ensemble of vocabulary trees based on hierarchical k-means clustering and query these trees to find the NN candidates in sub-linear time. As generating these vocabulary trees with widely used affine-invariant Riemannian metric (AIRM) will be very time-demanding, we propose to use the second-order approximation to AIRM (SOA-AIRM). We evaluate the proposed NN search algorithm in the application scenario of near-duplicate image detection in a large database. Experimental results demonstrate that the proposed method significantly outperforms state of the art techniques in terms of both accuracy and speed.

Index Terms—near duplicate image detection, nearest neighbor search, Riemannian manifold, vocabulary forest.

1. INTRODUCTION

Recent times have seen a steep rise of data which are encoded symmetric positive definite (SPD) matrices. Covariance, correlation and kernel matrices are typical examples of SPD matrices. Some successful applications in computer vision related areas that make use of SPD matrices include, diffusion-tensor imaging [18], near-duplicate image/video detection [1, 17], object tracking [20], activity recognition [13] and many others [2, 5].

Generally speaking, there are two major issues affecting the application of SPD matrices to large-scale datasets. First, they lie in a Riemannian manifold and it is time consuming to compute the similarities (distances) between data points. Unlike in vector space where distances between data points can be measured by straight line metrics such as L_p-norm, in the Riemannian manifold, the distance between data points must be calculated following the Riemannian curvature which is computationally very expensive. Researchers have made much effort to tackle the problem of similarity computation for SPD matrices. The simple and naïve method is to vectorize SPD matrices and then use Euclidean distance. However, it is an inferior choice because it ignores the manifold structure. A more suitable choice is to use geodesic distance - affine invariant Riemannian metric (AIRM). AIRM has theoretically excellent properties but lead to complex algorithms as it involves intensive use of matrix inverse, square roots and logarithms [5] or generalized eigenvalues [6].

In [2], the authors proposed to use matrix logarithm to transform manifold into vector space and then use Euclidean norm as a metric. However, the matrix logarithm transformation is only approximate rather than exact, because in general there is no such a mapping that globally preserves the distance between the points on the manifold. As a result, when used for nearest neighbor (NN) search, log-Euclidean Riemannian metric (LERM) is not as accurate as widely used AIRM. In [3], the authors proposed *Jensen-Bregman LogDet Divergence* (JBLD) which improves *Bregman Divergence* as an efficient similarity measure. JBLD is fast to compute but does not satisfy the triangle inequality. Another alternative is the symmetrized KL-Divergence Measure (KLDM) [4], however in some cases its accuracy is poor [3].

Secondly, there are no effective fast NN search algorithms in the Riemannian manifold of SPD matrices. Researchers have attempted to modify existing fast NN search algorithms in vector space which cannot be directly applied to non-vector spaces for Riemannian manifolds. Zheng *et al* [21] proposed a coarse-to-fine strategy for efficient near-duplicate image detection. The searching efficiency will be largely improved when hashing algorithms are used in the coarse stage. However, there isn't a mechanism to decide how many candidates should be returned in the coarse stage. Chaudhry and Ivanov [13] approximated the manifold with a set of tangent hyper-planes first and then projected the data onto the tangent space using logarithmic mapping. As the tangent space is Euclidean, fast NN search algorithm such as hashing [7] can be used in this space. A similar method was chosen by Turaga and Chellappa [8] for computer vision applications. However, in a manifold, only area around the pole locally resembles Euclidean space of a specific dimension. As a result, there will be large projection errors for points that are far away from the poles [8, 11], causing major inaccuracies in NN search.

On the other hand, authors in [3] adopted Bregman Ball Tree (BBT) for fast NN search using JBLD as dissimilarity function. The BBT was built through top-down bi-partitioning the input data space by recursively applying

This work is supported in part by 973 Program (2011CB302204), National Science & Technology Pillar Program (No:2012BAK16B06), NSFC (U1135001, 61332012, 61300205), 2012jq0017.

JBLD-K-Means algorithm [3]. The construction of BBT is somewhat similar to that of vocabulary tree [10] using hierarchical k-means (in BBT algorithm, k = 2). Unlike L_2-norm K-Means in the Euclidean space which optimizing a quadratic cost function and is guaranteed to converge to a local minimum, the JBLD-K-means algorithm [3] has no guarantee of convergence. Even if the JBLD-K-Means based BBT converges, it still inherits the two well-known problems of K-Means [12]. One is that the clustering is sensitive the initial cluster centers and the other is that there is no systematic methods to determining the suitable value of k (the number of clusters).

In this paper, we present a novel fast and accurate NN search technique for SPD matrices and make two original contributions. First, we propose a second-order approximation to AIRM under the framework of Baker-Campbell-Hausdorff [14]. We show that LERM is just the first order approximation of AIRM and the Second Order Approximation of AIRM (SOA-AIRM) has similar computational complexity as LERM but is more accurate. Furthermore, we show that SOA-AIRM has competitive accuracies as the much more complex AIRM for NN search in the manifold of SPD matrices. Second, we introduced an ensemble of random vocabulary forest[1] as indexing structure for fast NN search. In the construction of vocabulary forest, we use two different strategies. The first built the forest with varied structure constituent trees while the other built the forest with the same structure for each component trees. Both methods have a better accuracy and speed performance than the state of the art NN search methods for SPD matrices.

Our new method is a generic one and can be applied to any applications that use SPD matrices as feature representations and there is a need to search for nearest neighbors of data points which include many applications in information retrieval, pattern recognition, machine learning and computer vision. In experimental section, we use content based near-duplicate image detection as an illustrative application example to make our contribution specific and concrete.

2. FAST NEAREST NEIGHBOUR SEARCH USING HIERACHICAL VOCABULARY TREES

2.1 Vocabulary Tree for Nearest Neighbor Search

To avoid an expensive linear search through database, tree-based data structures are widely used to improve the efficiency of nearest neighbor search in many applications. As a specific kind of tree-based structures, vocabulary tree has been widely used in large scale databases for nearest neighbor search [9, 10]. The vocabulary tree defines a hierarchical quantization of the feature space where the nodes are the centroids determined by hierarchical k-means clustering of feature descriptors.

In the online phase, each descriptor is simply propagated down the tree by comparing the descriptor to the k candidate cluster centers at each level and choosing the closest one. This can efficiently find a query's nearest neighbor in sub-linear time.

Currently, most of the existing hierarchical k-means clustering are based on Lp norms (e.g., vocabulary tree [10]). The problem is that existing hierarchical k-means clustering can't be directly applied to Riemannian manifolds which employed geodesic distance instead of Lp norms.

Clustering in Riemannian manifolds is not easy, which can be attributed to two reasons. The first is the aforementioned time-demanding similarity computation which will be a major issue when many iterations are needed in clustering. The second lies in the fact that, unlike arithmetical mean in Euclidean space, there is not a closed form solution for Karcher mean [15] in manifolds. For these reasons, it will be very time consuming to train a vocabulary tree for Riemannian manifolds of SPD matrices using the present methods.

2.2 Riemannian Metric: AIRM and Its Approximation

We denote $S(n)=\{S\in R^{dxd}, S^T=S\}$, the space of all $n\times n$ symmetric matrices and denote $P(n)=\{P(n)\in S(n), P>0\}$the set of all $n\times n$ symmetric positive definite (SPD) matrices. The symmetric, positive definite matrices in $P(n)$ forms a Riemannian manifold. The space is not closed under manipulation with negative scalars. According to [5], an affine-invariant Riemannian metric (AIRM) is given by,

$$\delta(X,Y) = \left\|\log\left(X^{-\frac{1}{2}}YX^{-\frac{1}{2}}\right)\right\|_F \quad (1)$$

where log(X) is the matrix logarithm which converts the manifold into vector space. Furthermore, (1) is equivalent to

$$\delta(X,Y) = \sqrt{\sum_{k=1}^{d} ln^2\lambda_k(X,Y)} \quad (2)$$

$$= \|\log(YX^{-1})\|_F \quad (3)$$

where λ (X, Y) is generalized eigenvalues of X and Y. This metric is perhaps the most widely used distance measure for SPD matrices. This metric is theoretically elegant but involves matrix inverse and matrix logarithm or generalized eigenvalues thus it is time-consuming. Using the Baker-Campbell-Hausdorff for non-commutative Lie groups [14], it has the following approximation,

$$\begin{aligned}\delta(X,Y) &= \|\log(YX^{-1})\|_F \\ &= \|\log(\exp(\log(Y)) * \exp(\log(X^{-1})))\|_F \\ &= \|\log(\exp(\log(Y)) * \exp(-\log(X)))\|_F \\ &= \left\|\log(Y) - \log(X) + \tfrac{1}{2}\big(\log(X) * (\log(Y) - \log(Y) * \log(X)) + O(\|\log(Y) * \log(X)\|^3)\big)\right\|_F \\ &\approx \left\|\log(Y) - \log(X) + \tfrac{1}{2}(\log(X) * \log(Y) - \log(Y) * \log(X))\right\|_F \quad (4)\end{aligned}$$

[1] Vocabulary forest and vocabulary trees are usually an ensemble of vocabulary trees, while vocabulary tree is a single tree.

It is noted that, as X and Y are SPD matrices, according to the matrix theory, $\log(X)$ and $\log(Y)$ are symmetric matrices. Algebra system ($S(n)$, *) is a unchangeable group. So,

$$\log(X) * \log(Y) - \log(Y) * \log(X) \neq \mathbf{0} \qquad (5)$$

Where $\mathbf{0}$ is a matrix with each entry is zero.

The popular log-Euclidean Riemannian metric (LERM) [2] that are often used in the literature to transform manifold to vector space is defined as follows,

$$\delta_{LERM}(X,Y) = \|\log(X) - \log(Y)\|_F \qquad (6)$$

Comparing (4) and (6), it is easily seen that LERM of (6) is a first order approximation to AIRM while (4) is a second order approximation to AIRM. LERM is easy to compute as it can be easily tackled in ordinary Euclidean space. However, when used for NN search it is not as accurate as AIRM because in general there is no such mapping that globally preserves the distance between the points on the manifold [18]. Generally speaking, when used for NN search, the second-order approximation (SOA-AIRM) can be much more accurate than first-order approximation (LERM). ROC curves in Figure 1 demonstrate this fact where it is seen that SOA-AIRM is much more accurate than LERM and is almost as accurate as AIRM. SOA-AIRM shares the same good merits with LERM as it can also be tackled in vectorial logarithm space.

2.3 Clustering using SOA-AIRM

Clustering is a key component in building vocabulary tree using hierarchical k-means. It is very time-demanding to train a vocabulary tree in Riemannian space because of complicated geometry and a lot of iterations involved in k-means. As stated in subsection 2.2, when using SOA-AIRM, the clustering can be done in vector space log(X), which will be more efficient. In this paper, we choose a specific neural network training algorithm, namely the frequency sensitive competitive learning (FSCL) [12] algorithm. FSCL is reported to be very efficient clustering algorithm that is insensitive to initial random cluster centroids, thus very suitable for hierarchical k-means clustering for a large database. The FSCL method for Riemannian manifold can be briefly described as follows:

1. Preprocessing all the data using matrix logarithm to transform the data into vector space.
2. Conduct k-means using FSCL in the vector space.
3. Transform the each centroid into Riemannian manifold using matrix exponential.

In our experiments, building a vocabulary tree using SOA-AIRM costs on average 2% of the time when using AIRM.

2.4 Building Vocabulary Trees and NN Search

The idea of building vocabulary trees is burrowed from random forests [21]. Vocabulary forest (trees) is an ensemble of single vocabulary tree. The vocabulary tree is built by hierarchical k-means. For n points in the dataset, it will take O(n $\log_k$ n) time to build a k-ary tree. So the time complexity for building N vocabulary trees is N×O(n $\log_k$ n). We create two sets of vocabulary trees using hierarchical k-means with fixed k and different k respectively. We stopped partitioning the nodes when the number of points goes below a threshold. Given a query point, it compares the node along the tree from the root. Once it reaches to the leaves of all the vocabulary trees, we combine all samples falling onto these leaves to do an exhaustive search. As the average number of samples in each leaf is small, it is very efficient to find the nearest neighbors.

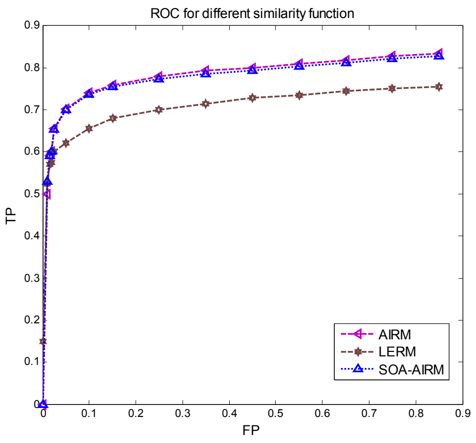

Figure 1: the ROC curve for LERM, AIRM and SOA-AIRM of near-duplicate image detection through brute-force search. The experiment setups are described in subsection 4.1 and 4.2.

3. EXPERIMENT AND RESULT ANALYSIS

In this part, we give our experiment of using vocabulary trees for near-duplicate image detection in Riemannian manifolds.

3.1 Experiment Setup

The evaluation is performed on datasets: 1)INRIA Copydays dataset [16] as testing dataset; 2)25,000 Flickr images and another 19,634 images (totally 44,634 images) as distracting image dataset[2]. We create totally 46 copies for each testing image. For each image, a SCOV[3] [1] is extracted to represent the image. The receiver operating characteristic (**ROC**) curve and mean average precision (**mAP**) is employed to evaluate the overall performance for each algorithm.

3.2 Experimental Results and Analysis

3.2.1 ROC Performance for Different NN Searching Methods

In our experiments, we give the ROC results for brute-force NN search using LERM and AIRM, single vocabulary tree using SOA-AIRM, multiple vocabulary trees using SOA-AIRM (10 trees) and LERM (10 trees), as well as the method proposed in [13].

[2] In order to get a comprehensive performance evaluation of our algorithm, besides the challenges mentioned in [16], we extend the transformation types, such as additive noise (salt & pepper, Gaussian), flipping, rotate, blur, illumination change, combination attacks, *etc*.

From the results shown in Figure 2, we can see brute-force NN search using ARIM clearly gives the best ROC performance. The proposed method (multiple vocabulary trees) can achieve nearly the same ROC performance as exhaustive brute force search using AIRM. The performance of using SOA-AIRM to build vocabulary forest is better than that of using LERM, which demonstrate that SOA-AIRM has a better accuracy than LERM. We also compare multiple vocabulary trees with the algorithm proposed in [13] and [21], this method also achieves a significant improvement in ROC performance.

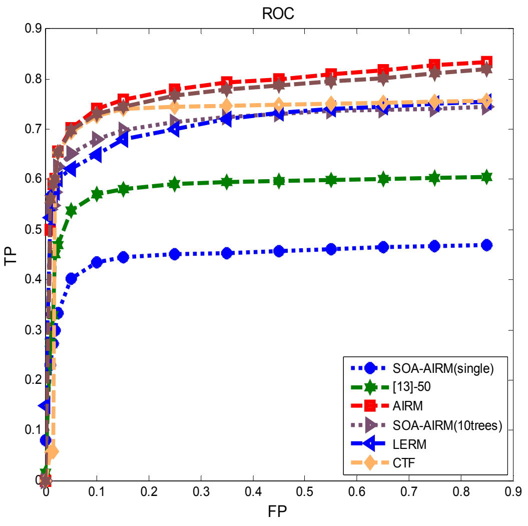

Figure 2. ROC curves for different methods. [13]-50 is the method used in [13] and the number of poles is 50. LERM and AIRM are the brute-force search results. LERM (10 trees) is the result of 10 trees using LERM to build vocabulary forest. SOA-AIRM (single) is the result of using one tree while SOA-AIRM (10 trees) is the result of using 10 trees each with a different k. Both SOA-AIRM (single) and SOA-AIRM (10 trees) used SOA-AIRM to build vocabulary forest. CTF is coarse-to-fine strategy proposed in [21].

3.2.2 Time Efficiency Evaluation

Table 1: speed and mAP for different methods

methods	***Time per query***	**mAP**
AIRM	17s	0.7654
LERM	16.5s	0.6841
10 trees(k not fixed)	0.7s	0.7509
[13]-50	0.71s	0.5992

AIRM means the brute force search using AIRM. LERM means the brute force search using LERM. [13]-50 means using the method in [13] and the number of poles is 50. 10 trees (k not fixed) means the vocabulary forest built using 10 vocabulary trees with different k, details refer to subsection 4.3.3.

Table 1 shows the time efficiency for the proposed methods (multiple vocabulary trees), the brute-force method using AIRM and LERM, and the method proposed in [13]. From table 1, we can see that our method is as fast as the method in [13] which is over 20 times faster than using brute force searching in a 50 k image dataset. However, it is also seen that the accuracy of our method is over 22% better than the method of [13].

3.2.3 Comparison of Different Vocabulary Forest Building Strategy

In our experiments, we compare the time efficiency and mean average precision (mAP) of different strategies to choose k in hierarchical clustering. Specifically, we build two sets of vocabulary forests. One set is built with the same k, which means that each component tree in the forest is k-ary (k fixed). The other set of forest is built with different k, which means that the component trees in the forest are 2-ary, 3-ary,…, and 11-ary respectively.

From Table 2, we can see that the proposed methods (both fixed k and different k) outperform the method in [13]. What is more, creating the vocabulary forest using the same k may result in different results for different k. Because, there is no systematic method existed for choosing the best k – this is a well-known problem of k-means. On the other hand, using hierarchical k-means with a set of different k (k = 2, 3,…,K) to create a vocabulary forest can successfully address this issue.

Table 2: the comparison of vocabulary forest.

methods	**Time/query**	**mAP**
10 trees(K not fixed)	0.7	0.7509
10 trees(K=5)	0.7	0.7289
10 trees(K=6)	0.7	0.7278
10 trees(K=7)	0.7	0.7380
10 trees(K=8)	0.7	0.6814
10 trees(K=9)	0.7	0.7367
10 trees(K=10)	0.7	0.7376

Each forest contains 10 vocabulary component trees. The vocabulary forest in the first row is created using hierarchical k-means with k=2, 3, 4, 5, 6, 7, 8, 9, 10, 11. The others are created with the same k as listed.

4. CONCLUSION

In this paper, we propose to use vocabulary forest (an ensemble of vocabulary trees) as our indexing structure for fast and accurate NN search in Riemannian manifolds. In order to efficiently train the vocabulary trees in Riemannian manifolds, we propose to use the second order approximation of AIRM (SOA-AIRM) as similarity measure for SPD matrices. This approximation can substantially reduce the training time while maintaining the NN search accuracy. We also evaluated the different strategies to construct the vocabulary trees and found that by varying the k in the component trees of the forest can achieve robust results.

[3]SCOV is a covariance matrix (SPD matrix. More information please refers to [1].

5. REFERENCES

[1] L. G. Zheng, G. P. Qiu, J. W. Huang and H. Fu, “Salient Covariance for Near Duplicate Image and Video Detection,” In Proceedings of International Conference on Image Processing, pp.2585-2588, 2011.

[2] V. Arsigny, P. Fillard, X. Pennec, and N. Ayache, “Fast and simple calculus on tensors in the log-Euclidean framework,” Medical Image Computing and Computer-Assisted Intervention , pp. 115-122, 2005

[3] A. Chedian, S. Sra, A. Banerjee and N. Papanikolopoulos, “Efficient similarity search for Covariance matrices via the Jensen-Bregman LogDet Divergence,” 2011 International Conference on Computer Vision, pp.2399-2406 2011

[4] M. Moakher, and P. Batchelor, “Symmetric positive-definite matrices: From Geometry to applications and visualization,” Springer Berlin Heidelberg, Chapter 17, 2006

[5] X. Pennec, P. Fillard, and N. Ayache,”A Riemannian Framework for Tensor Computing,” International Journal of Computer Vision, Volume 66 Issue 1, January 2006.

[6] W. Förstner, B. Moonen, F. Gdp and C. F. Gauss, “A Metric for Covariance Matrices,” Technical report, Dept. of Geodesy and Geoinformatics, Stuttgart University (1999)

[7] Piotr Indyk, Rajeev Motwani, “Approximate Nearest Neighbors: Towards Removing the Curse of Dimensionality,” In Proceedings of the thirtieth annual ACM symposium on Theory of computing (1998), pp.604-613.

[8] P. Turaga and R. Chellappa,”Nearest-neighbor search algorithms on non-Euclidean manifolds for computer vision applications ,” Proceedings of the Seventh Indian Conference on Computer Vision, Graphics and Image Processing, ISBN: 978-1-4503-0060-5, 2010

[9] D. Nister and H. Stewenius, “Scalable Recognition with a Vocabulary Tree,” International conference on Computer Vision and Pattern Recognition, pp. 2161- 2168, 2006

[10] T. Yeh, J. Lee and T. Darrell, “Adaptive Vocabulary Forests for Dynamic Indexing and Category Learning,” IEEE 11th International Conference on Computer Vision, pp.1-8, 2007

[11] J. Sivic and A. Zisserman. “Video Google: A text retrieval approach to object matching in videos,” 2003 International Conference on Computer Vision, pp.1470-1477, 2003.

[12] Stanley C. Ahalt , Ashok K. Krishnamurthy, “Competitive learning algorithms for vector quantization,” Neural Networks, Volume 3, pp. 277-290, 1990

[13] R. Chaudhry and Y. Ivanov, “Fast Approximate Nearest Neighbor Methods for Non-Euclidean Manifolds with Applications to Human Activity Analysis in Videos,” 2010 European Conference on Computer Vision, pp.735-748, 2010

[14] Roger Godement, “Introduction a la theorie des groupes de Lie,” Springer (November 2003)

[15] Xavier Pennec,” Probabilities and Statistics on Riemannian Manifolds: A Geometric Approach,” In IEEE Workshop on Nonlinear Signal and Image Processing, pp: 194-198.2004

[16] M. Douze, H. Jegou, H. Sandhawalia, L. Amsaleg and C. Schmid, “Evaluation of Gist Descriptors for Web-scale Image Search,” ACM CIVR, pp.19.1-19.8, 2009.

[17] L. G. Zheng, G. P. Qiu, J. W. Huang and Y Lei, “Near-duplicate Image Detection in Visually Salient Riemannian Space,” In IEEE Transaction on Information Forensics and Security, 7(5): 1578-1593, (2012)

[18] P. Thomas Fletcher and Sarang Joshi, “ Riemannian Geometry for the Statistical Analysis of Diffusion Tensor Data,” Signal Processing, vol.87,pp.250-260, 2007

[19] Weiming Hu and Xi Li and Wenhan Luo and Xiaoqin Zhang and Stephen J. Maybank and Zhongfei Zhang, “Single and Multiple Object Tracking Using Log-Euclidean Riemannian Subspace and Block-Division Appearance Model,” IEEE Trans. Pattern Anal. Mach. Intell., pp. 2420-2440, 2012

[20] Breiman, Leo, “Random Forests,” Machine Learning, vol. 45, pp. 5–32, 2001

[21] Ligang Zheng, Guoping Qiu and Jiwu Huang, “Efficient coarse-to-fine near-duplicate image detection in Riemannian manifold.” ICASSP 2012, pp.977-980.