Hierarchical Local Maps for Robust Approximate Nearest Neighbor Computation

Pratyush Bhatt and Anoop Namboodiri
Center for Visual Information Technology, IIIT, Hyderabad, 500032, India
{bhatt@research., anoop@}iiit.ac.in

Abstract

In this paper, we propose a novel method for fast nearest neighbors retrieval in non-Euclidean and non-metric spaces. We organize the data into a hierarchical fashion that preserves the local similarity structure. A method to find the approximate nearest neighbor of a query is proposed, that drastically reduces the total number of explicit distance measures that need to be computed. The representation overcomes the restrictive assumptions in traditional manifold mappings, while enabling fast nearest neighbor's search. Experimental results on the Unipen and CASIA Iris datasets clearly demonstrates the advantages of the approach and improvements over state of the art algorithms. The algorithm can work in batch mode as well as in sequential mode and is highly scalable.

1. Introduction

In various applications like multimedia search, biometric authentication, in order to retrieve the top K matches or nearest neighbors, the similarity of input sample with every sample in database should be found. This is a bottleneck in any online retrieval algorithm. As the similarity measure itself is not an absolute distance metric, a trade-off of accuracy of the nearest neighbor with speed or complexity of the algorithm can be done.

In biometric identification, a particular person's biometric sample is compared against all the registered samples in a database to identify the person. This process can be extremely time consuming in large databases even if the matching algorithm is extremely fast. For example, to do background check of a person who is crossing the border using the complete IAFIS system, one needs to do around 55 million comparisons. Even with the state of the art matching algorithms, this would take close to 10 minutes, which is not practical considering the millions of people who cross the border every month. Even for criminal investigations, it is desirable to get a quick and approximate search done immediately rather than the typical turn-around time of a few days for a search.

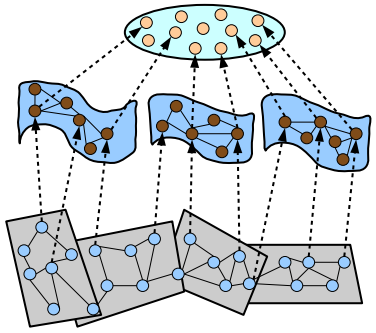


Figure 1. Hierarchical Local Maps.

To this end, various indexing methods such as hashing and tree structures are proposed [8, 7]. However, these approaches are applicable only in a metric space. Moreover, as the dimensions of input space increases, the performance degrades owing to curse of dimensionality. In this paper, we use manifold learning to approximate the computationally expensive distance measures such as Dynamic Time Warping (DTW). The mapping algorithm learns a lower dimensional embedding that best preserves the pairwise similarity between every point in input space [5, 15].

To compute the embedding of a new point into an existing manifold, one needs to recompute the entire manifold, with distances to all the existing points. This defeats the very purpose of our using manifolds to reduce the distance computations. To address this issue of scalability, scalable dimensionality reduction algorithms[5, 14, 12] were proposed. However, the goal here is to learn the nature of embedding and not to be able to add a new point quickly.

Online queries are handled well by Fast Map [6], Random Reference Objects [10], Random Line Projections [11], and VP-Tree[16]. These algorithms find the embedding of the query by computing only a few exact distances. Although these methods assume that triangular inequality holds, they work for non-metric distances as well with certain amount of distortion in embedding. Athitsos [3] framed

embedding construction as a machine learning task, where AdaBoost is used to combine many simple, 1D embeddings into a multidimensional embedding that preserves a significant amount of the proximity structure in original space. These five techniques are most related to our approach as their main target is to learn embeddings for fast retrieval of nearest neighbors. However, the existence of an embedding that is globally optimal is questionable. Moreover, the inherent local similarity in neighborhood is not exploited in these algorithms.

In this paper we propose the use of local manifold mappings for finding robust and approximate k-nearest neighbors for a given sample. The method, referred to as Hierarchical Local Maps (HLM) (see figure 1), arranges a set of simple local manifolds in a hierarchical fashion. As we move up in the hierarchy, the complexity of manifold increases as the data does not belong to a neighbourhood. At the top most level, no lower dimensional space can be found, where the pairwise similarity is preserved. Nearest neighbor retrieval is now framed as selecting correct path to traverse down the hierarchy that would give approximate nearest neighbors. We present experiments on two real world complex dataset: the UNIPEN dataset [9], and the CASIA Iris dataset [1]. The results show a considerable amount of computationally expensive measurements can be reduced without affecting the accuracy of nearest neighbors found. We also present comparison to state of the art algorithms.

2. Hierarchical Local Maps

Goal: Given a set S of N points and an arbitrary distance measure (F) between them, construct a data structure that helps us to compute an approximate list of nearest neighbors of a query point q.

In most of the real-world datasets, there is no low-dimensional single manifold that spans the whole dataset. However, parts of the dataset may lie on a manifold. For example, each handwritten digit lies on a single manifold, but a smooth manifold covering all the digits does not exist. Since distance metric is non-metric, even cluster based approaches cannot guarantee that points similar to each other will fall in the same cluster.

We propose a way to split the data into a multi-level hierarchy, so that, local similarity property of dataset can be exploited to direct the search to correct branch of the tree while traversing it in top-down fashion. We call this representation as *Hierarchical Local Maps(HLM)*. Using a tree structure for representation, helps in reducing the search space at each level and makes the algorithm scalable and incremental. New samples can be added in the hierarchy without modifying the existing tree structure. Such a tree representation, combined with a way to exploit local similarity, drastically reduces the explicit distance computations.

Input : S, F, T
Output: $Tree, Num\_Levels$

$l = 1$; $N_1 = n(S)$; $New\_Set = \{\ \}$
while $N_l > T$ **do**
  $bf_{++l} = \lfloor P \log_{10}(N_l) \rfloor$
  while $\exists x \in S\ s.t.\ seen(x) == FALSE$ **do**
    Add an unseen point, $Seed$, to New_Set
    $nn \leftarrow$ bf-NN of $Seed$ in S according to F
    Mark $Seed$ and nn as seen
    Make $Seed$ parent of nn
  end
  $S = New\_Set$; $New\_Set = \{\ \}$; $N_l = n(S)$
end
$Num\_Levels = l$

Algorithm 1: Construction of HLM.

2.1 Construction of the Hierarchy

In a non-metric space, the computation of an optimal hierarchy for a given set of points, that minimizes the number of comparison needed for finding nearest neighbors is extremely difficult. Thus we use a greedy method to construct the hierarchy such that local neighborhood information gets embedded in a tree, which could be used later to direct the search to correct local maps. Let N denote the number of samples in training set S. The similarity function is denoted by F ; N_l and bf_l are the number of points and branching factor for each level, respectively. T denotes the minimum number of samples need to be present at top most level. Algorithm 12 describes the way to build the Hierarchical Local Maps.

In such a representation, a single point may lie on multiple nodes at a level in the hierarchy. Thus, if a point is overlooked at any level during a serach, it could still be included in the levels to follow. This is one of the major advantages this representation holds over the traditional tree-based search, where a misdirected search cannot be corrected. If the input space is metric, then this operation would preserve topology with zero distortion. However, in a non-metric space one might be able to find metric approximations of data points lying in a small neighbourhood.

2.2 Nearest Neighbors Retrieval

To carry out a search, we need to describe a way to traverse down in the hierarchy. At the topmost level, we explicitly find the P_1 nearest neighbors of a query sample q using the similarity function F. The search process as described in algorithm 12, requires only $P_1 \times P_2$ ex-

plicit distance computations at any level. The ISOMAP algorithm [13] is used to learn the manifolds formed by the nearest neighbors at each level. The embedding of a point is given by [5].

$$y = L_k^{\#}\left(\vec{\delta_a} - \vec{\delta_\mu}\right) \quad (1)$$

$$L_k^{\#} = \left[\frac{v_1^t}{\sqrt{\lambda_1}}\frac{v_2^t}{\sqrt{\lambda_2}}\ \frac{v_k^t}{\sqrt{\lambda_k}}\right]^t, \quad (2)$$

where $\vec{\delta_a}$ is squared distance between q and P_1 points, $\vec{\delta_a}$ is the mean of columns of the $P_1 \times P_1$ squared distance matrix. At every stage, the nearest $P_1 \times P_2$ points are determined in the low dimension, which are further reduced to P_1 using explicit computation of F. One can also include the similarity measures computed in the previous level for further refinement. If the search finds only $K1$ points at the last level of the hierarchy, where $K1 < K$, we expand the list by backtracking and finding more points at the previous level.

Input : $Query\ q, Num\_Levels, P_1, P_2$
Output: $NN : Nearest\ Neighbors\ List$

$level = Num\_Levels$
Put points of topmost level in S
$NN \leftarrow P_1$-NN of q in S
while $level > 1$ **do**
    Store children of NN in S_{child}
    Run Landmark Isomap to embed S_{child} with NN as landmarks
    Let Emb_{newpt} be embedding of q
    $K_1 = P_1 * P_2$
    Find K1-NN of Emb_{newpt} in low dimensional embedding
    Filter P_1 from $K1$ and update NN set
    Decrement level
end

Algorithm 2: Nearest Neighbor Retrieval.

2.3 Parameter Selection

While constructing a hierarchy, T and bf_l needs to be tuned for any dataset. The T is a constant that is set emperically, based on the overall similarity of the points in the dataset (set at 50 in our experiments). As the total number of points in a level increases, the number of points similar to any given point also increases. Thus the branching factor at a level, is determined by the number of points in the level below. For our experiments, we set bf_l as:

$$bf_{l+1} = Plog_{10}N_l; \quad (3)$$

The NN retrieval algorithm has two parameters, P_1 and P_2. The application for which retrieval is being used determines their values. If for any application we need lesser percentage of nearest neighbors to be correct, then a lower value of P_1 can be chosen. However, if aim is at higher accuracy then a bigger value for P_1 and P_2 should be chosen. P_2 determines the weight given to distance computed in low dimensions.

2.4 Computational Complexity

In most applications, the non-metric distance computation is the most expensive operation to be performed. In the proposed search, the number of distance computations to be performed at the top-most level is around T. Further, at each lower level, we need to perform $P_1 \times P_2$ distance computations. Hence the overall computational complexity is $O(Num\_Levels \times P_1 \times P_2)$.

In addition to the above, we also need to compute $P_! \times P_2$ nearest neighbors from $P_1 \times bf_l$ samples in fixed dimensions at each level. The complexity of this process is $O(Num\_Levels \times P_1 \times bf)$, assuming the branching factor to be bf at all levels. The computation of the low dimensional space manifold requires the singular value decomposition of a P_1x$(P_1 \times bf)$ matrix, which is $O(P_1^2 \times bf)$ operations. The embedding process requires $O(P_1 \times d)$ multiplications, where d is the dimensionality of the manifold space. Note that our aim is to reduce the number of non-metric distances computed.

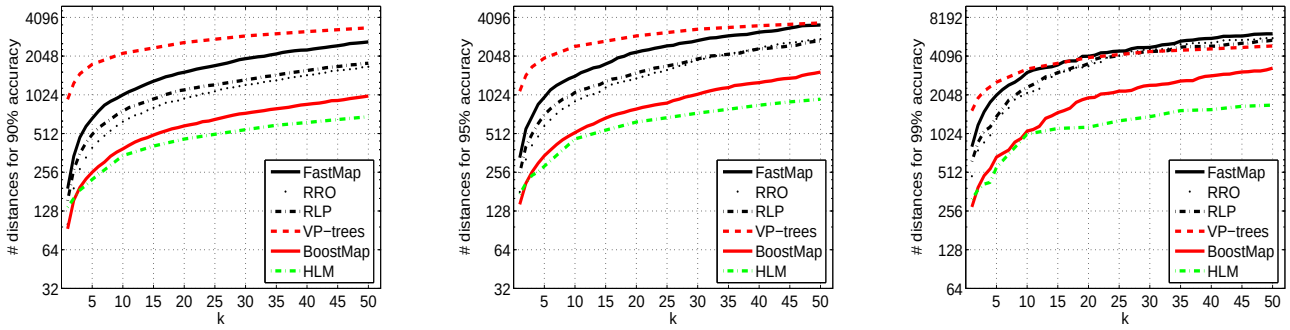
3. Experimental Results and Discussion

3.1 Unipen Handwriting Database

Experiments are conducted on the Unipen train-R01/V07 online handwriting database [9]. It contains 15953 digit examples of which randomly selected 10630 are treated as training data and rest 5323 as testing data. To compare our results with BoostMap, we downloaded two distance matrices, one with DTW score between each pair of database object and other with DTW score between each test object and database object along with class labels for training and testing set from [2].

During the construction of HLM, the value of P is set emperically as 2.5. We conduct several experiments for different values of accuracy one aims to achieve for different set of values for P_1 and P_2. The optimal value is obtained for $P_1 = 15$ and $P_2 = 1$. This means that the nearest neighbors computed in lower dimension are good enough and thus no refinement process is required.

The results of our algorithm is shown in figure 2. As there is no external conditions or parameters of the dataset used, we directly used the values reported in the BoostMap



(a) For 90% of the 5323 test samples (b) For 95% of the 5323 test samples (c) For 99% of the 5323 test samples

Figure 2. Number of DTW computations for K nearest neighbor retrieval.

paper [3] for other algorithms namely RRO, RLP, FastMap, VP-Trees[1]. Each subplot shows the exact number of DTW distances that needs to be computed against different values of nearest neighbors to be retrieved, for different accuracies on input dataset.

For lower values of K, the difference in the number of distance computations is not significant but as K increases, HLM starts out-performing the other algorithms. Note that the values in Y-axis is in logarithm scale, so even small difference along Y-axis for higher K, signifies much more savings in terms of actual distances to be computed. To be precise, for 99% accuracy, BoostMap required 3302 exact distance computations, whereas HLM required only 1704 explicit distance measurements.

To study the nature of graph for $K > 50$, we find the number of nearest neighbors that can be retrieved with respective accuracy for the same number of distance computations required for $K = 50$ in BoostMap. For 90% and 95% accuracy, HLM can find 127-NN which is 2.5 times the nearest neighbors found by BoostMap using same number of distance computations. For 99% accuracy, HLM can find 110-NN. Thus we can say that the nature of the graph would be the same for higher values of K.

Another measure of the approximate nearest neighbors computed is their similarity to the query point. To evaluate this, we perform k-NN classification using the approximate NNs and compared with accuracy achieved using exact NN from direct DTW measurements. For the chosen parameter, we are left with around 30 points in last level after calculating around 130 DTW distances. Hence, the total number of DTW distances to be computed in order to find the first nearest neighbor (approximate) is around 160. Table 1, shows the classification accuracies for different values of k, when we use HLM instead of computing DTW distances to all points. As indicated by the first column, our approach can classify a sample within 0.5% accuracy of the ideal case, while achieving a 98.5% savings on DTW distance computations (160 instead of $10,630$).

k	1	5	10	15	20
DTW	98.1	97.73	97.41	96.99	96.83
HLM	97.65	97.27	97.08	96.69	96.44

Table 1. Classification Accuracy on UNIPEN Dataset using exact and approximate k-NN.

3.2 CASIA Iris Database

However, the nature of matching score using for biometric identification or verification is highly discriminative, with small intra-class distances and larger inter-class distances. Hence the matching score cannot be used for computing approximate nearest neighbor. For strong biometric traits such as the iris pattern, the similarity score between any two samples belonging to different classes will be close to each other, making the hierarchical search almost random. To solve this problem, one should use a smoother distance function in the HLM construction and retrieval. Note that the problem in the case of iris based identification is not that of high computational cost of the distance metric, but the sheer size of the database, which makes explicit comparison with every sample, impractical.

For iris based person identification, we segment the iris pattern into a set of concentric circles, and each circle further into sectors. We characterize each segment using the average gray value, after normalization of the whole image, resulting in a 160 dimensional feature vector. Euclidean distance between two such vectors is used to find the ap-

[1]We would like to thank Dr. Vassilis Athitsos, University of Texas, for providing the BoostMap Code and results for comparison.

proximate nearest neighbor. We use the term *soft metric* to refer to this distance measure, as opposed to the matching score used for the biometric. To compare the results, we also perform the experiments with the matching score computed using the 20x240 feature vector, proposed in [4].

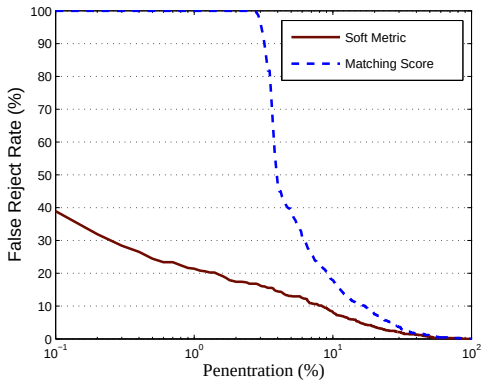


Figure 3. FRR vs Penetration (CASIA Iris).

Experiments are conducted on the CASIA Iris Image Database V3.0 [1]. For experimental evaluation, the CASIA-IrisV3-Interval was used as it contains the larger number of images, captured in two different sessions. Database consists of left and right eye images of 249 subjects. Six images per eye of a subject are randomly chosen and divided equally in training and testing set. We discard those users for which less than six images per eye were present. In total 855 images were present in training and testing set, corresponding to 285 eyes, with three samples per eye. We construct the HLM structure using the distance measure mentioned above.

For the construction of HLM, as samples of most of the class will not be present in the top most level, we set T to be a constant: 50 (irrespective of number of classes). Figure 3 shows the variation of False Reject Rate(FRR) with Penetration Rate for $P_1 = 15$ and $P_2 = 1$. We note that one can find a matching pattern at the first nearest neighbor in around 60% of the case, which requires around 40 soft metric computations and 75 distances in a 15-dimensional subspace. Note that this is very small compared to the 855 matching scores to be computed for the brute force approach. Moreover, finding of the following nearest neighbors take only around 1 comparison on an average.

On the downside, a large number of similarity measures need to be evaluated during offline stage of constructing the HLM. The storage complexity is also linear in the number of training samples. We note that the search works better when used with smoother similarity measures, which are also well correlated with the original distance metric, F.

4. Conclusions and Future Work

We presented a novel approach for robust approximate nearest neighbor retrieval. A way to arrange the data in multiple levels of hierarchy is proposed so that local neighborhood information can be utilized during search to guide it to correct local map. Retrieval as well as classification results on UNIPEN dataset shows the advantages of using HLM over state-of-the-art approximate nearest neighbor retrieval algorithms. One of the potential directions of improving the algorithm would look into optimal construction of HLM. One could also extend the applicability of the approach by defining similarity measures that allow hierarchical representation.

References

[1] $http://www.cbsr.ia.ac.cn/IrisDatabase.htm.$
[2] $csr.bu.edu/asl/data/bm_datasets/unipen/.$
[3] V. Athitsos, J. Alon, S. Scarloff, and G. Kollis. Boostmap: An embedding method for efficient nearest neighbor retrieval. *PAMI*, January 2008.
[4] J. Daugman. How iris recognition works. *IEEE Transactions on Circuits and Systems for Video Technology*, 14:21–30, 2004.
[5] V. de Silva and J. Tenenbaum. Sparse multidimensional scaling using landmark points. *Stanford Mathematics Technical Report*, 2004.
[6] C. Faloutsos and K. Lin. Fastmap: A fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets. *In ACM SIGMOD International Conference on Management of Data*, pages 163–174, 1995.
[7] P. Franco, P. Shamos, and M. Ian. *Computational Geometry: An Introduction*. Springer-Verlag, 1985.
[8] A. Gionis, P. Indyk, and R.Motwani. Similarity search in high dimensions via hashing. *In International Conference on Very Large Databases*, pages 518–529, 1999.
[9] I. Guyon, L. Schomaker, and R. Plamondon. Unipen project of on-line data exchange and recognizer benchmarks. *Proc. of ICPR*, pages 29–33, 1994.
[10] G. Hjaltason and H. Samet. Properties of embedding methods for similarity searching in metric spaces. *IEEE Transactions on PAMI*, 25(5):530–549, 2003.
[11] G. Hristescu and M. Farach-Colton. Cluster-preserving embedding of proteins. *Technical Report 99-50*, 1999.
[12] S. Ketpreechasawat. Hierarchical landmark charting. *Master's. thesis*, 2006.
[13] J. C. Langford, B. Tenenbaum, and V. de Silva. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
[14] J. Li and P. Hao. Hierarchical structuring of data on manifolds. In *CVPR07*, pages 1–8, 2007.
[15] S. Roweis and L. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.
[16] P. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. *ACM-SIAM Symposium on Discrete Algorithms*, pages 311–321, 1993.