INCREMENTAL UPDATING OF NEAREST NEIGHBOR-BASED HIGH-DIMENSIONAL ENTROPY ESTIMATION

Jan Kybic

Center for Machine Perception, Czech Technical University, Prague, Czech Republic

ABSTRACT

We present an algorithm for estimating entropy from high-dimensional data based on Kozachenko-Leonenko nearest neighbor estimator. The problem of finding all nearest neighbors is approximatively solved using a best-bin first (BBF) bottom-up k-D tree traversal. Our main application is evaluating higher-order mutual information (MI) image similarity criteria that, unlike standard scalar MI, are directly usable for vector (e.g. color) images and can take into account neighborhood information. As during the optimization the MI criterion is often evaluated for very similar images, it is advantageous to update the k-D tree incrementally. We show that the resulting algorithm is fast and accurate enough to be practical for the image registration application.

1. INTRODUCTION

Mutual information (MI) measures the amount of independence between two random variables F and G. It can be defined from the joint and marginal entropies

$$I(\mathsf{F},\mathsf{G}) = H(\mathsf{F}) + H(\mathsf{G}) - H(\mathsf{F},\mathsf{G}) \tag{1}$$

Besides numerous other applications, it has been used as an image similarity criterion for multimodal image registration [1, 2, 3, 4], with the samples of the random variables F, G usually corresponding to scalar ($d = 1$) pixel intensity values. Several attempts have been made to extend the MI registration framework to higher-dimensionality (vector, $d > 1$) data, especially in order to incorporate spatial relationships [5]. The main difficulty is overcomming the inadequacy of the standard and almost exclusively used histogram-based estimator for high-dimensional data, the "curse of dimensionality". The histogram estimator can be partly improved using vector quantization or adaptive binning [6], or alternative estimators can be used [7, 8, 9] that often work better for high-dimensional data.

In particular, the Kozachenko-Leonenko (KL) entropy estimator [10, 11] based on nearest neighbor (NN) distances works well for estimating not only standard scalar MI ($d = 1$) but also color MI ($d = 3$) and neighborhood MI ($d = 25$) [12] and these new criteria are superior to standard similarity measures such as SSD. Computational bottleneck of the KL estimator is the all-NN search, with complexity $O(dN^2)$ for the brute-force algorithm, which is unacceptably high for typical image registration application, where the number of samples (pixels) $N = 10^5 \sim 10^6$. Therefore, in [12], a relatively crude approximation is used, treating samples by batches.

The main contribution of this article is to present an efficient all-NN search algorithm based on k-D trees and the best-bin first (BBF) approach [13], and an incremental strategy of the tree update. Together with an improved robustification of the KL estimator, this leads to a practically usable entropy and MI estimation algorithm even for high-dimensional ($d \approx 20$) problems with large number of samples ($N \approx 10^6$).

Sponsored by the Grant Agency of the Czech Academy of Sciences, grant 1ET101050403.

2. ENTROPY ESTIMATION

Given a set of N samples $\mathbf{x}_i \in \mathbb{R}^d$ taken from an unknown pdf $p(\mathbf{x})$, we define the NN distance

$$\varrho_i = \min_{j \neq i} \|\mathbf{x}_i - \mathbf{x}_j\|_\infty$$

We have to adapt the KL estimator [10, 11] slightly for the ℓ_∞ (maximum) norm. (The maximum norm was chosen for better compatibility with the NN search algorithm that partitions the space into hyper-rectangles. However, the difference is small.) The probability $Q_i(r) = \mathcal{P}(\rho_i > r)$ of the neighborhood of $\mathbf{x}_i$ of size r being empty satisfies

$$Q_i(0) = 1 \quad \text{and} \quad Q_i(r + \mathrm{d}r) = Q_i(r)\left(1 - p(\mathbf{x}_i)\mathrm{d}V\right)^{(N-1)}$$

which leads to

$$Q_i(r) = \exp\left(-2(N-1)\,d\,(2r)^{d-1}p(\mathbf{x}_i)\right)$$

where $V = (2r)^d$ is the neighborhood size. The expected value of $\log \varrho$ is

$$\langle \log \varrho \rangle = \int_0^\infty \log r \frac{\mathrm{d}Q}{\mathrm{d}r}\,\mathrm{d}r = -\log\left(2(N-1)\,d\,p(\mathbf{x}_i)\right) - \gamma$$

where we have used the identity $\int_0^\infty \log(\alpha z)\,\mathrm{e}^{-z}\,\mathrm{d}z = \log\alpha - \gamma$ and where $\gamma \approx 0.577$ is the Euler constant. This directly yields the local density logarithm estimate

$$-\widehat{\log p(\mathbf{x}_i)} = d\log\varrho_i + \gamma + \log 2^d(N-1) \tag{2}$$

and consequently also the global entropy estimate

$$\widehat{H} = -\frac{1}{N}\sum_{i=1}^{N} \widehat{\log p(\mathbf{x}_i)} \tag{3}$$

Entropy estimation can be improved using k-th NN ($k > 1$). However, to find further NN is computationally expensive and thus was not used here.

2.1. MI estimation

MI can be estimated through entropy from (1). In image registration, F represents the fixed image and $H(\mathsf{F})$ is therefore constant. Often, also $H(\mathsf{G})$ corresponding to the moving image can be assumed approximately constant, saving computational effort and avoiding cancellation errors, at the expense of some accuracy loss. Estimating MI directly is possible [14] but the algorithm is less statistically and computationally efficient.

2.2. Robust KL entropy estimator

For real data identical samples can occur with non-zero probability and the estimator (2), $\rho_i \mapsto \widehat{\log p(\mathbf{x}_i)}$ must be modified appropriately. If $\rho_i < \varepsilon$ for an a priori chosen ε, we assume that quantization has taken place and the original samples share the same bin. The pdf is assumed constant inside each quantization bin. Furthermore, we require the estimate of $p(\mathbf{x}_i)$ to be continuous in ρ_i and proportional to k_i, the number of samples in the bin. This yields the following robustified estimate to be plugged into (3)

$$-\widehat{\log p(\mathbf{x}_i)} = \begin{cases} d \log \varrho_i + C & \text{for} \quad \rho_i \geq \varepsilon \\ \log\left(\varepsilon^d / k_i\right) + C & \text{for} \quad \rho_i < \varepsilon \end{cases}$$

where $C = \gamma + \log 2^d (N-1)$. Alternatively, a small amplitude perturbation can be added to the input data [14].

3. BBF K-D TREE WITH UPDATING

Finding the NN for a query point in 2D is a well-studied problem [15, 16, 17, 16], less so finding all-NN within a set. The available methods rarely have acceptable performance for $d \gg 2$. Dynamic methods exist [18] but no algorithm we are aware of can take advantage of our specific update pattern, considering instead inserts and deletes [19], or predetermined trajectories [20].

We construct a binary tree similar to the k-D tree. Each leaf contains at most L samples $\mathbf{x}_i$. Leaves are implemented using internal chaining hash-tables [17] for update efficiency, storing indices to the original data matrix. Each node contains the number of samples in the subtree of which it is a root and the corresponding hyper-rectangle (bounding box) $\mathcal{B}$. The bounding boxes are loose, filling the parent box completely and starting with $\mathbb{R}^d$ for the root. Non-leafs contain also the partitioning dimension m and value ξ; the m-th coordinate x_i^m of all points in the left resp. right subtree satisfies $x_i^m < \xi$, resp. $x_i^m \geq \xi$. In this way, identical samples end-up in the same leaf.

During recursive construction, tight bounding boxes are also maintained. The longest dimension of each box is chosen as the partitioning dimension m, with ξ calculated as a median (using incomplete ordering [17]) of the dimension m, so that each subtree receives half of the samples (except when many equal values are present). The tree construction complexity is $O\big(dN \log(N/L)\big)$.

3.1. All-NN search

The task is to find the NN distance or the multiplicity for all points in the tree. Unlike almost all other search methods that start at the root, we take advantage of the fact that all query points $\mathbf{q}$ are in the tree and that a NN is most likely in the same leaf, or some of the branches close by. For each query point $\mathbf{q}$ in leaf Q, we shall perform an A*-like search of the tree rooted at Q.

The nodes R to be visited are kept in a priority queue (PQ, implemented as a heap [17]), as in the best-bin-first (BBF) approach [13], partially ordered according to η_R, a minimum possible distance between q and any point of the subtree rooted in R. Note that η_R is the distance from q to $\mathcal{B}(R)$ when going from parent to son, and the distance to its complement $\mathbb{R}^d \backslash \mathcal{B}(R)$ otherwise. This way, closest boxes that are more likely to contain the NN are considered first.

The currently best NN distance ρ_{best} is maintained and used for pruning nodes with $\eta > \rho_{\text{best}}$ that cannot bring improvement. When a non-leaf is taken from the PQ head, the two nodes linked to it not yet visited are inserted to the PQ. When a leaf is taken, all points contained in it are examined. The search is terminated if the PQ is empty or if a predetermined number M of leaf points has been considered. In leaf Q, which is searched first, we also test whether q is multiple; if it is we terminate the search.

In low dimensions d, often only a few neighboring nodes of the leaf Q need to be searched. For higher d the all-NN search complexity is $O(dNM)$, with the parameter M governing the trade-off between accuracy and speed.

Table 1. The tree construction (top) and all-NN search (bottom) times in seconds for $N = 10^5$ uniformly distributed samples as a function of the dimension d and the leaf size L. The best search times for each d are set in boldface.

d/L	5	10	20	30	40	50
1	4.8	4.2	3.7	3.3	3.3	3.0
2	5.8	5.1	4.7	4.2	4.2	3.8
3	6.5	5.8	5.2	4.7	4.7	4.3
5	7.7	6.9	6.2	5.7	5.6	5.1
10	11	10	9	8	8	7
15	13	12	11	10	10	9
20	16	15	14	12	12	11

d/L	5	10	20	30	40	50
1	1.9	**1.6**	2.1	2.9	2.9	5.8
2	6.5	**5.1**	5.4	6.2	6.2	10.1
3	17.6	14.2	**13.7**	14.4	14.4	20.4
5	75.0	62.0	**57.0**	57.8	57.9	71.7
10	873.6	671.8	588.0	**563.3**	563.5	632.0
15	4049	3130	2770	**2714**	2714	3086
20	18155	13479	11372	**10591**	10595	11166

3.2. Incremental update

Consider the task of updating a tree built from samples $\mathbf{x}_i$ to be valid also for samples $\mathbf{x}'_i$, assuming that the change $\mathbf{x}'_i - \mathbf{x}_i$ is small. To minimize changes in the tree structure, we will allow the sons of a parent containing N_p points to have up to $(1/2 + \delta)N_p$ points, as in the pseudo k-D tree [20].

The update algorithm first iterates over all points in the tree. If a point $\mathbf{q}$ is no longer contained in the bounding box of its original leaf, we go up the tree until an ancestor node is found that is big enough to contain $\mathbf{q}$. We then go down using the partitioning information (m,ξ) to find a leaf into which $\mathbf{q}$ belongs.

The second phase consist of traversing the tree depth-first, checking for unbalanced subtrees. If such a subtree is found, it is freshly rebuilt as described in Section 3.

If the changes are small, almost no rebuilding is needed. In the worst case, the whole tree needs to be rebuilt.

4. EXPERIMENTS

All experiments were run on a 2 GHz Pentium IV computer using an experimental Ocaml implementation. In the first experiment (Table 1) we attempt to determine the optimum leaf size (parameter L) Assuming that the all-NN search will prevail, the optimum L seems

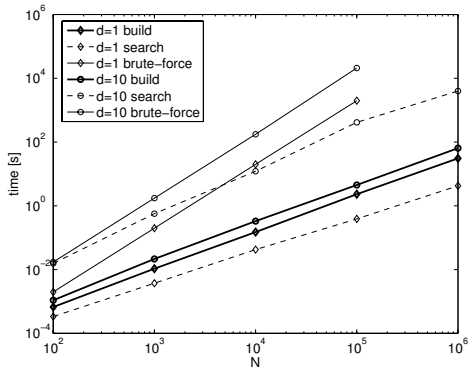

Fig. 1. Times needed for the tree construction, tree-based all-NN search and brute-force all-NN search as a function of N for $d = 1, 10$.

Table 2. The tree update time T_u and the all-NN search times on the updated tree (T_1) and on a freshly built tree (T_2) as a function of the perturbation amplitude σ and the parameter δ, for $d = 5$, $N = 10^5$.

σ	δ	T_u	T_1	T_2
0.01	0	5.9	57	57
0.01	0.1	0.3	57	57
0.01	0.2	0.3	57	57
0.01	0.3	0.3	57	57
0.1	0	6.3	59	59
0.1	0.1	0.9	59	60
0.1	0.2	0.7	59	60
0.1	0.3	0.7	59	60
1	0	7.3	69	69
1	0.1	3.2	70	69
1	0.2	2.3	76	69
1	0.3	1.7	82	69

to be around $L = 10$ for $d = 1$, gradually raising to $L = 30$ for $d = 20$.

The second experiment (Figure 1) compares the time needed for the tree construction and our all-NN search with the brute-force $O(N^2)$ all-NN search. The optimal L determined above was used. The results are identical since no search truncation was used, $M = \infty$. The tree approach is faster for all N. Brute-force search was not performed for the highest N since it would be too time consuming.

The third experiment (Table 2) studies the effect of the maximum permitted tree unbalancement δ for $d = 5$ and $N = 10^5$. We performed the tree update after adding random perturbation uniformly distributed in $[-\sigma, \sigma]^d$ to the original uniform samples from $[-1, 1]^d$. We measure the update time and the all-NN search time on the updated tree. Updating the tree with $\delta \geq 0.1$ is always faster than building it anew, which takes 5 s. Since only small and medium amplitude updates are expected in our application, we have chosen $\delta = 0.3$ which does not increase the all-NN search time significantly.

Reducing the parameter M can reduce the all-NN search time at the expense of accuracy of the NN distance estimation. We have investigated this trade-off for multidimensional unit variance normally distributed data with known entropy $H = (d/2)\log(2\pi\mathrm{e})$. (Table 3). The value of M needed for a given accuracy increases with the dimensionality d: for $d \leq 10$ examining $M = 10^3$ points yields a relative MSE smaller than 1 %.

Table 3. The all-NN search times (top row) and MSE of the entropy estimator (bottom row) as a function of the dimension d and the maximum number of visited points M for $N = 10^5$ for normally distributed data. Bold entries correspond to relative accuracy better than 2 %.

d/M	10	10^2	10^3	10^4	10^5
1	5.9	6.2	6.2	6.2	6.2
	0.05	**0.0069**	**0.0059**	**0.0043**	**0.0036**
2	7.5	10.2	10.2	10.2	10.2
	0.18	**0.006**	**0.0058**	**0.0045**	**0.0035**
3	8.8	19.6	20.0	20.0	20.0
	0.39	**0.097**	**0.083**	**0.0070**	**0.0055**
5	11.3	52.8	83.3	83.4	83.5
	1.03	**0.097**	**0.024**	**0.020**	**0.020**
10	17.7	108	556	1181	1198
	3.02	1.14	**0.076**	**0.053**	**0.049**
15	24.0	156	909	4902	6114
	4.67	2.42	0.72	**0.14**	**0.12**
20	45.8	176	960	7893	17280
	6.03	4.67	2.05	0.69	**0.52**

4.1. Image similarity criterion evaluation

Finally, we apply the new estimator to two image similarity evaluation tasks, using the parameters determined in the previous experiments. First, we take the red channel of the Lena image (Figure 2, left) and calculate the scalar MI with respect to the rotated version of the green channel of the same image.

Second, we shall the color Mandrill image (Figure 2, right) and the a colormap-rotated version of the same image [12], using a color MI criterion [12] (Figure 3).

This corresponds to entropy estimation with $d_{\mathsf{FG}} = 2$ in the first case and $d_{\mathsf{FG}} = 6$ in the second case. All images were pre-rotated by a small amount to avoid singularity effects due to discretization. In each case, images were cropped after rotation to avoid boundary issues. The results were not much sensitive to the parameter ε; value 1 was used, corresponding to the quantization. We also found that it is not necessary to update the moving image entropy and it even leads to better (less noisy) results. Both experiments were repeated for different sizes of the original images. To mimic the conditions of image registration, while maintaining reproducibility, we chose to evaluate the criteria for angles from 10° to 0° decreasing by 0.1°. Average evaluation time is reported to distribute the cost of the initial tree construction (Table 4). We see that the evaluation times are quite acceptable for off-line use.

5. CONCLUSIONS

Estimating high-dimensional MI and thus entropy using the KL estimator needs a fast all-NN search algorithm. We have shown that such all-NN search performed using a k-D tree can be accelerated, especially in higher dimensions and at the expense of a small loss of accuracy, by considering the k-D tree nodes in the order of the

Table 4. Average times in seconds to evaluate the scalar and color MI criteria depending on the image size. Percentage of the MI evaluation time with respect to the total evaluation time (including image rotation).

size	N_{pix}	T_{scalar}	%	T_{color}	%
104×104	10816	0.2	75	2.1	91
206×206	42436	1.0	77	12.1	93
410×410	168100	6.2	83	61.7	94

likelihood (best-bin-first) and truncating the search as needed. Second major improvement concerns dropping the requirement of exact balancing of the k-D tree and updating it instead of rebuilding each time.

The resulting estimator keeps the good properties of the KL entropy estimator, the suitability of which for image registration application has been shown previously [12]. However, unlike [12], almost exact NN search has now become computationally feasible, improving the accuracy of the estimator. It is a practically usable method of estimating high-dimensional entropy and MI. Further acceleration is likely by reimplementing the presented estimator in C and combining it with the batch idea [12].

6. REFERENCES

[1] Paul Viola and William M. Wells III, "Alignment by maximization of mutual information," *International Journal of Computer Vision*, , no. 2, pp. 137–154, 1997.

[2] J.P.W. Pluim, J. B. A. Maintz, and M. A. Viergever, "Mutual-information-based registration of medical images: A survey," *IEEE Transactions on Medical Imaging*, vol. 22, no. 8, pp. 986–1004, Aug. 2003.

[3] F. Maes, A. Collignong, D. Vandermeulen, G. Marchal, and P. Suetens, "Multimodality image registration by maximization of mutual information," *IEEE Transactions on Medical Imaging*, vol. 16, no. 2, pp. 187–198, Apr. 1997.

[4] Philippe Thévenaz and Michael Unser, "Optimization of mutual information for multiresolution image registration," *IEEE Transactions on Image Processing*, vol. 9, no. 12, pp. 2083–2099, Dec. 2000.

[5] D. Rueckert, M. J. Clarkson, D. L. G. Hill, and D. J. Hawkes, "Non-rigid registration using higher-order mutual information," in *Proceedings of SPIE Medical Imaging 2000: Image Processing*, 2000, pp. 438–447.

[6] Georges A. Darbellay and Igor Vajda, "Estimation of the information by an adaptive partitioning of the observation space," *IEEE Transactions on Information Theory*, vol. 45, no. 4, pp. 1315–1321, May 1999.

[7] J. Beirlant, E. J. Dudewicz, Györfi L., and E. C. van der Meulen, "Nonparametric entropy estimation: an overview," *International J. Math. Stat. Sci.*, , no. 6, pp. 17–39, 1997.

[8] Erik G. Miller, "A new class of entropy estimators for multi-dimensional densities," in *Proceedings of ICASSP2003*, 2003.

[9] O. Michel A. O. Hero, B. Ma and J. Gorman, "Applications of entropic spanning graphs," *IEEE Signal Proc. Magazine*, vol. 19, no. 5, pp. 85–95, sep 2002.

[10] L. F. Kozachenko and N. N. Leonenko, "On statistical estimation of entropy of random vector," *Probl. Inf. Trans.*, vol. 23, no. 9, 1987, (in Russian).

[11] Jonathan D. Victor, "Binless strategies for estimation of information from neural data," *Physical Review E*, vol. 66, no. 5, pp. 051903(15), Nov. 2002.

Fig. 2. Lena and Mandrill images used in the experiments.

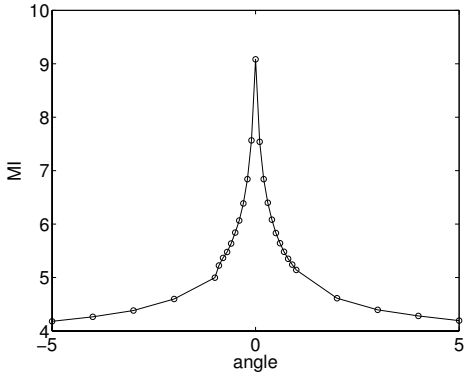

Fig. 3. Color MI criterion between the original and colormap-rotated versions of Mandrill as a function of the rotation angle.

[12] Jan Kybic, "High-dimensional mutual information estimation for image registration," in *ICIP'04: Proceedings of the 2004 IEEE International Conference on Image Processing*, 445 Hoes Lane, Piscataway, NJ, U.S.A., October 2004, IEEE Computer Society.

[13] Jeffrey S. Beis and David G. Lowe, "Shape indexing using approximate nearest-neighbour search in high-dimensional spaces," in *Proceedings of Conference on Computer Vision and Pattern Recognition*, June 1997, pp. 1000–1006.

[14] A. Kraskov, H. Stögbauer, and P. Grassberger, "Estimating mutual information," *Physical Review E*, , no. 69 066138, 2004.

[15] M. Smid, "Closest-point problems in computational geometry," 1997.

[16] F. P. Preparata and M. I. Shamos, *Computational geometry: An introduction*, Texts and Monographs in Computer Science. Springer-Verlag, 1985.

[17] Robert Sedgewick, *Algorithms*, Addison Wesley, 1989.

[18] Y. Chiang and R. Tamassia, "Dynamic algorithms in computational geometry," *Proc. of the IEEE*, vol. 80, no. 9, pp. 362–381, 1992, Special Issue on Computational Geometry.

[19] P. Bozanis, A. Nanopoulos, and Y. Manopoulos, "Lr-tree: a logarithmic decomposable spatial index method," *The computer journal*, vol. 46, no. 3, 2003.

[20] P. K. Agarwal, J. Gao, and L. J. Guibas, "Kinetic medians and *kd*-trees," in *Proc. 10th European Sympos. Algorithms*, Sept. 2002, pp. 5–16.