'N-Body' Problems in Statistical Learning

Alexander G. Gray
Department of Computer Science
Carnegie Mellon University
agray@cs.cmu.edu

Andrew W. Moore
Robotics Inst. and Dept. Comp. Sci.
Carnegie Mellon University
awm@cs.cmu.edu

Abstract

We present very fast algorithms for a large class of statistical problems, which we call all-point-pairs problems, or 'N-body'-like problems. These are problems which abstractly require a comparison of each of the N points in a dataset with each other point and would naively be solved using N^2 computations. Such algorithms cannot be applied to practical problems where N is large. In this paper we focus on several examples of all-point-pairs problems within nonparametric statistical learning: nearest-neighbor classification, kernel density estimation, outlier detection, the two-point correlation, multiple two-point correlation, and the n-point correlation. We give an asymptotic analysis and we show empirically that this algorithmic framework leads to several orders of magnitude in speedup over the naive computation, even for small datasets. We are aware of no exact algorithms for these problems which are faster either empirically or theoretically. The methodology presented in this paper is also applicable in principle to fast, large-scale parametric statistics such as RBF neural networks, mixtures of Gaussians, and Gaussian processes.

Keywords: nearest-neighbor classification, kernel density estimation, outlier detection, two-point correlation, n-point correlation, nonparametric statistics, machine learning, kd-trees, hyperstructure, cached sufficient statistics, N-body problem, computational geometry, all-nearest-neighbors, range-counting, range-searching.

1 Introduction

This paper is about accelerating a wide class of statistical algorithms that are naively quadratic in the number of datapoints. We introduce a class of dual kd-tree traversal algorithms for these problems. They are similar in spirit to powerful state-of-the-art N-body simulation algorithms [1, 5], but the full range of learning and inference tasks has many new opportunities for acceleration and techniques more general than those for potential-based problems.

We describe the dual-tree algorithm in detail for calculating the two-point correlation, the simplest case of the problems we consider; for the five other statistical problems we consider, we show only performance results for lack of space. These include a generalization of this approach to a class of point process methods called n-point correlation functions that compute statistics based on all n-tuples of datapoints, providing the first exact acceleration of this class and a new, faster, type of approximate acceleration. In addition, all the algorithms in this paper can be compactly defined and are easy to implement.

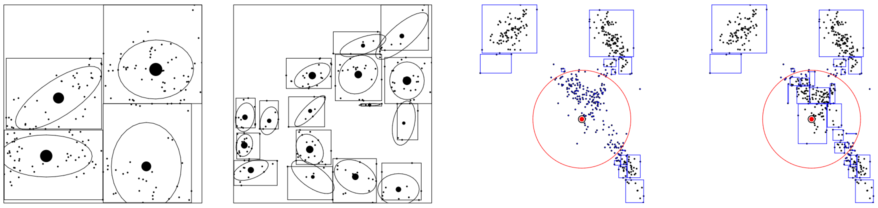

Figure 1: A kd-tree. (a) Nodes at level 3. (b) Nodes at level 5. The dots are the individual data points. The sizes and positions of the disks show the node counts and centroids. The ellipses and rectangles show the covariances and bounding boxes. (c) The rectanges show the nodes pruned during a RangeSearch for one (depicted) query and radius. (d) More pruning is possible using RangeCount instead of RangeSearch.

Statistics and geometry. Our algorithms proceed by viewing these statistical problems as geometric problems, exploiting the hyperstructure of the data space. Each algorithm utilizes dual Multiresolution kd-trees, one for each of the two sets of points, providing a geometric partitioning of the data space which is used to reason about entire chunks of the data simultaneously.

A review of kd-trees and $mrkd$-trees. A kd-tree [4] records a d-dimensional data set containing N records. Each node represents a set of data points by their bounding box. Non-leaf nodes have two children, obtained by splitting the widest dimension of the parent's bounding box. For the purposes of this paper, nodes are split until they contain only one point, where they become leaves. An $mrkd$-tree [3, 10, 7] is a conventional kd-tree decorated, at each node, with extra statistics about the node's data, such as their count, centroid, and covariance. They are an instance of the idea of cached sufficient statistics [9] and are quite efficient in practice. [1] Figure 1 shows an example.

2 The 2-point correlation function

The two-point correlation is a point process statistic which is of fundamental importance in many natural sciences, in particular astronomy and astrophysics. It can be thought of roughly as a measure of the clumpiness of a set of points. It is easily defined as the number of pairs of points in a dataset which lie within a given distance, or radius, r of each other.

2.1 Previous approaches

Quadratic algorithm. The most naive approach to computing this statistic is to simply compare each datum to each other one, incrementing a count if the distance between them is less than r. In this case the cost is $O(N^2)$, unacceptably high for problems of practical interest.

Binning and gridding algorithms. The algorithms in widespread use [14, 15] are mainly of this sort. The idea of binning is simply to divide the data space into a fine grid defining a set of bins, perform the quadratic algorithm on the bins as if they were individual data, then multiply by the bin sizes as appropriate to get an estimate of the total count. The idea of gridding, is to divide the data space into a coarse grid, perform the quadratic algorithm within each bin, and sum the results over all bins to get an estimate of the total

[1] $mrkd$-trees can be built quickly, in time $O(dN \log N + d^2 N)$. Although we have not needed to do so, they can modified to become disk-resident for data sets with billions of records, and they can be efficiently updated incrementally. They scale poorly to higher dimensions but recent work [8] partly remedies the dimensionality problem.

count. These are both of course very approximate algorithms. They are not usable when r is small or r is large, respectively.

Range-searching with a kd-tree. An approach to the two-point correlation computation that has been taken is to treat it as a *range-searching* problem [6, 12], since kd-trees have been historically almost synonymous with range-searching. The idea is that we will make each datapoint in turn a *query point* and then execute a range search of the kd-tree to find all other points within distance r of the query.

Each search proceeds by a depth-first traversal of the nodes in the kd-tree. There is frequently the opportunity for *exclusion-based* pruning of a node. The search always checks the minimum possible distance d_{min} between the query and the hyper-rectangle surrounding the current node. If $d_{\mathsf{min}} > r$ there is no point in visiting the node's children, and computation is saved.

The range searching avoids computing most of the distances between pairs of points further than r apart, which is a considerable saving if r is small. But is it the best we can do? And what if r is large? In this paper we propose several improvements to the previously-used geometric approaches.

2.2 Better geometric approaches: new algorithms

Single-tree search (Range-Counting Algorithm). A straightforward extension can exploit the fact that unlike conventional use of range searching, these statistics frequently don't need to retrieve all the points in the radius but merely to count them. The $mrkd$-tree has, in each node, the count of the number of data it contains—the simplest kind of cached sufficient statistic. At a given node, if the distance between the query and the farthest point of the bounding box of the data in the node is smaller than the radius r, clearly every datum in the node is within range of the query. We can then simply add the node's stored count to the total count. We call this *subsumption*. (Note that both exclusion and subsumption are simple computations because the geometric regions are always axis-parallel rectangles.) As we will see, subsumption can also be exploited when other *aggregate* statistics, such as centroids or covariances of sets of points in a range are required. This insight has been used for accelerated statistics by [3, 16, 11].

Dual-tree search. This is the primary topic of this paper. The benefits of single-tree search can be enhanced by simultaneously searching over query points and their nearby kd-tree nodes instead of iterating over the query points in an outer loop and only exploiting single-tree-search in the inner loop. Dual-tree search is based on node-node comparisons while Single-tree search was based on point-node comparisons.

Pseudocode for a recursive procedure called **TwoPoint**(QNODE,DNODE,r) is shown in Figure 2. Its job is to count the number of pairs of points $(x_q \in \mathrm{QNODE}, x_d \in \mathrm{DNODE})$ such that $|x_q - x_d| < r$. Before doing any real work, the procedure checks whether it can perform an exclusion pruning (in which case the call terminates, returning 0) or subsumption pruning (in which case the call terminates, returning the product of the number of points in the two nodes). If neither of these prunes occur, then depending on whether QNODE and/or DNODE are leaves, various kinds of recursive calls are made.

Importantly, both kinds of prunings can now apply to a whole batch of query points, instead of each nearby query point rediscovering the same prune during the Single-tree search. The intuition behind Dual-tree's advantage can be seen by considering two extreme cases. First, if r is so large that all pairs of points are counted then the Single-Tree search will perform $O(N)$ operations, where each query point immediately prunes at the root, while Dual-Tree search will perform $O(1)$ operations. Second, if r is so small that no pairs of points are counted, Single-Tree search will run to one leaf for each query, meaning

```
TwoPoint(QNODE,DNODE,r)
  if excludes(QNODE,DNODE,r), return;

  if subsumes(QNODE,DNODE,r)
    total = total + ( count(QNODE) × count(DNODE) );
    return;

  if leaf(QNODE) and leaf(DNODE)
    if distance(QNODE,DNODE) < r, total = total + 1;

  if leaf(QNODE) and  leaf(DNODE)
    TwoPoint(QNODE,leftchild(DNODE),r);
    TwoPoint(QNODE,rightchild(DNODE),r);

  if  leaf(QNODE) and leaf(DNODE)
    TwoPoint(leftchild(QNODE),DNODE,r);
    TwoPoint(rightchild(QNODE),DNODE,r);

  if  leaf(QNODE) and  leaf(DNODE)
    TwoPoint(leftchild(QNODE),leftchild(DNODE),r);
    TwoPoint(leftchild(QNODE),rightchild(DNODE),r);

    TwoPoint(rightchild(QNODE),leftchild(DNODE),r);
    TwoPoint(rightchild(QNODE),rightchild(DNODE),r);
```

Figure 2: The **TwoPoint** recursive function.

total work $O(N \log N)$ whereas Dual-tree search will in total visit each leaf once, meaning $O(N)$ work. Note, however, that these extreme cases are not representative of the general case of a medium size r, in which Dual-tree is theoretically only a constant-factor superior to Single-tree. [2]

Non-redundant dual-tree search. So far, we have discussed two operations which cut short the need to traverse the tree further - exclusion and subsumption. Another form of pruning is to eliminate node-node comparisons which have been performed already in the reverse order. This can be done [13] simply by (virtually) ranking the datapoints according to their position in a depth-first traversal of the tree, then recording for each node the minimum and maximum ranks of the points it owns, and pruning whenever QNODE's maximum rank is less than DNODE's minimum rank. This is useful for all-pairs problems, but will later be seen to be *essential* for all-n-tuples problems. This kind of pruning is not practical for Single-tree search. Figure 2.2 shows the performance of a two-point correlation algorithm using all the aforementioned pruning methods.

Multiple radii simultaneously. Most often in practice, the two-point is computed for many successive radii so that a curve can be plotted, indicating the clumpiness on different scales. Though the method presented above is fast, it may still be quite painful to run it, say, 1000 times. It is possible to perform the computation for many radii simultaneously, scaling much less than linearly in the number of radii. The function takes as arguments indices into an array or other structure mapping indices to real-valued radius values. The algorithm starts with the low index set to the minimum and the high value to the maximum.

[2] We'll summarize the asymptotic analysis briefly. If the data is uniformly distributed in d-dimensional space, the cost of computing the n-point correlation function on a dataset with N points using the Dual-tree (n-tree) algorithm is $O(N^{\alpha_{nd}})$ where α_{nd} is the dimensionality of the manifold of n-tuples that are just on the border between being matched and not-matched, and is $\alpha_{nd} = n' \left(1 - \frac{n'-1}{2d}\right)$ where $n' = \min(n, d)$ For example, the 2-point correlation function in two dimensions is $O(N^{3/2})$, considerably better than the $O(N^2)$ naive algorithm. Disappointingly, for 2-point, this performance is asymptotically the same cost as Singe-tree. For $n > 2$ our algorithm *is* better. Furthermore, if we can accept an approximate answer, the cost is $\left(\frac{n \alpha n d}{\epsilon}\right)^{(\alpha_{nd}/(n-\alpha_{nd}))}$ which is independent of N.

Algorithm	# Data	Quadratic	Single-tree	Dual-tree	ST Speedup	DT Speedup
twopoint	10,000	132	2.2	1.2	60	110
twopoint	20,000	528 est.	4.8	2.8	110	189
twopoint	50,000	3300 est.	11.8	7.0	280	471
twopoint	150,000	30899 est.	37	20	835	1545
twopoint	300,000	123599 est.	76	40	1626	3090
nearest	10,000	139	2.0	1.4	70	99
nearest	20,000	556 est.	11.6	9.8	48	57
nearest	50,000	3475 est.	30.6	26.4	114	132
outliers	10,000	141	2.3	1.2	61	118
outliers	20,000	564 est.	4.7	2.6	120	217
outliers	50,000	3525 est.	12	6.5	294	542
outliers	150,000	33006 est.	36	21	917	1572
outliers	300,000	132026 est.	72	44	1834	3001

Figure 3: Our experiments timed our algorithms on large astronomical datasets of current scientific interest, consisting of x-y positions of sky objects from the Sloane Digital Sky Survey. All times are given in seconds, and runs were performed on a Pentium III-500 MHz Linux workstation. The larger runtimes for the quadratic algorithm were estimated based on those for smaller datasets. The dual kd-tree method is about a factor of 2 faster than the single kd-tree method, and both are 3 orders of magnitude faster than the quadratic method for a medium-sized dataset of 300,000 points.

# Data	1	100	1000	Speedup
10,000	1.2	1.8	2.4	500
20,000	2.8	6.4	6.6	424
50,000	7.0	31	31	226
150,000	20	133	146	137

# Data	Quadratic	10^{-2}	10^{-4}	10^{-6}	Speedup
10,000	226	1.2	2.2	3.0	188
20,000	904 est.	2.6	4.0	4.4	348
50,000	5650 est.	10.4	14.4	16.8	543
150,000	50850 est.	32	59	65	1589
300,000	203400 est.	73	105	151	2786

Figure 4: (a) Runtimes for multiple 2-point correlation with increasing number of radii, and the speedup factored compared to 1,000 separate Dual-tree 2-point correlations. (b) Runtimes for kernel density estimation with decreasing levels of approximation, controlled by parameter ϵ, and speedup over quadratic. The datasets are again from the Sloane Digital Sky Survey.

As it proceeds, the range of indices is narrowed. First the closest and farthest distances between the nodes is determined. If the closest is outside of the radii being considered, the call is terminated. If the farthest is inside of all the radii the total for the low index is incremented by the number of data in the data node. If the closest is outside some of the radii or the farthest is inside some of the radii, narrow the range to consider before recursing. The results in Figure 3 confirm that the algorithm quickly focuses on the radii of relevance.

3 Kernel density estimation

Approximation accelerations. A fourth major type of pruning opportunity is *approximation.* This arises particularly in all-point-pairs computations which involve computing some real-valued function $f(x, y)$ between every pair of points x and y. An example is kernel density estimation [2]. For small kernel bandwidths, an analog of Range-Searching can be used for considerable acceleration over the quadratic algorithm. But [3] showed how a method closer to Range-Counting, can offer dramatic accelerations even for values of r sufficiently large to cause each query point to be impacted by a large fraction of all other points. We now a Dual-tree approach yielding yet greater acceleration.

For each query point x_q we need to accumulate $K \sum_i w(|x_q - x_i|)$ where K is a normalizing constant and w is a weighting function (which we will need to assume is monotonic). A recursive call of the Dual-tree implementation has the following job: for $x_q \in$ QNODE compute the contribution to x_q's summed weights that are due to all points in DNODE.

n	Naive	n-tree	Speedup
2	528	2	264
3	12133	186	65
4	278784	14441	19

# Data	Time
1000	1
2000	13
5000	176
10000	1470
20000	14441

d	$n = 2$	$n = 3$	$n = 4$
1	< 1	< 1	< 1
2	< 1	3	23
3	< 1	6	57
4	< 1	7	73

Figure 5: (a) Runtimes for approximate n-point correlation with $\epsilon = 0.02$ and 20,000 data. (b) Runtimes for approximate 4-point correlation with $\epsilon = 0.02$ and increasing data size. (c) Runtimes for exact n-point correlation, run on 2000 datapoints of galaxies in d-dimensional color space.

Once again, before doing any real work we use simple rectangle geometry to compute the shortest and furthesh possible distances between any (x_q, x_d) pair. This bounds the minimum and maximum possible values of $Kw(|x_q - x_d|)$. If these bounds are tight enough (according to an approximation parameter ϵ) we prune by simply distributing the midpoint weight to all the point in QNODE. This distribution can be implemented semi-efficiently by directly incrementing all the partial scores of all points in QNODE.

4 The n-point correlation, for $n > 2$

The n-point correlation is the generalization of the 2-point correlation, which counts the number of n-tuples of points constrained to be within radius r of each other, or more generally, between some $r_{\min}$ and $r_{\max}$. [3] The implementation is entirely analogous to the 2-point function, except that there is more benefit in being careful about which of 2^n possible recursive calls to choose in the cases where you cannot prune, the approximation versions are harder, there is no immediately analagous Single-tree version of the algorithm, and anti-redundancy pruning is much more important. In general, the algorithm uses n trees.

Approximating 'exact' computations. Even for algorithms such as 2-point, that return exact counts, approximations may be desirable. Suppose the true value of the 2-point function is $V^\star$ but that we are prepared to accept a fractional error of ϵ: we will be happy with any value $\hat{V}$ such that $|\hat{V} - V^\star| < \epsilon V^\star$. It is possible to adapt the dual-tree algorithm using a best-first iterative deepening search strategy to guarantee this result while exploiting permission to approximate effectively by building the count as much as possible from "easy-win" node pairs while doing approximation at hard deep node-pairs. The details are omitted for space.

5 Outlier detection, nearest neighbors, and others

One of the main intents of this paper is to point out the broad applicability of this type of algorithm within statistical learning. Figure 6 lists many all-point-pairs problems which are clear candidates for acceleration in future work. Figure 2.2 shows performance results for our outlier detection and nearest neighbors algorithms. [4]

[3] So far, this paper has only dealt with pairs of points, but now we consider arbitrary tuples. the n-point value is a certain kind of point process statistic that is useful for characterizing mass distributions (including stars, galaxies and biomasses). Higher-order n-point correlations detect increasingly subtle differences in mass distribution, and are also useful for assessing variance in the lower-order n-point statistics. For example, the three-point correlation function, which measures the number of triplets of points meeting the specified geometric constraints, can distinguish between two distributions that have the same 2-point correlations but differ in their degree of "stripyness" versus "spottiness".

[4] In our nearest neighbors algorithm we consider the problem of finding, for each query point, its single nearest neighbor among the data points. (This is exactly the all-nearest-neighbors problem

Statistical Operation	Results in this paper?	Approximation?	What is N?
2-point function	Yes	Optional	# Datapoints
n-point function	Yes	Optional	# Datapoints
Multiple 2-point function	Yes	Optional	# Datapoints
Batch k-nearest neighbor classify	Yes	Optional	# Datapoints
Batch k-nearest neighbor regression	No	Optional	# Datapoints
Non-parameteric outlier detection/denoising	Yes	Optional	# Datapoints
Batch Kernel density/classify/regression	Yes	Yes	# Datapoints
Multiple-bandwidth kernel density/class./reg.	Yes	Yes	# Datapoints
Batch locally weighted regression	No	Yes	# Datapoints
Batch kernel PCA	No	Yes	# Datapoints
Gaussian process learning and prediction	No	Yes	# Datapoints
K-means iterations with large K	No	Optional	# Datapoints
Mixture model with many components	No	Yes	# Datapoints
Finding pairs of correlated attributes	No	Optional	# Attributes
Finding n-tuples of correlated attributes	No	Optional	# Attributes
Dependency-tree learning	No	Optional	# Attributes

Figure 6: A very brief summary of applicability of Dual-tree search methods.

References

[1] J. Barnes and P. Hut. A Hierarchical $O(NlogN)$ Force-Calculation Algorithm. *Nature*, 324, 1986.

[2] C. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, 1995.

[3] K. Deng and A. W. Moore. Multiresolution instance-based learning. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 1233–1239, San Francisco, 1995. Morgan Kaufmann.

[4] J. H. Friedman, J. L. Bentley, and R. A. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3(3):209–226, September 1977.

[5] L. Greengard and V. Rokhlin. A Fast Algorithm for Particle Simulations. *Journal of Computational Physics*, 73, 1987.

[6] D. E. Knuth. *Sorting and Searching.* Addison Wesley, 1973.

[7] A. W. Moore. Very fast mixture-model-based clustering using multiresolution kd-trees. In M. Kearns and D. Cohn, editors, *Advances in Neural Information Processing Systems 10*, pages 543–549, San Francisco, April 1999. Morgan Kaufmann.

[8] A. W. Moore. The Anchors Hierarchy: Using the triangle inequality to survive high dimensional data. In *Twelfth Conference on Uncertainty in Artificial Intelligence (to appear)*. AAAI Press, 2000.

[9] A. W. Moore and M. S. Lee. Cached Sufficient Statistics for Efficient Machine Learning with Large Datasets. *Journal of Artificial Intelligence Research*, 8, March 1998.

[10] A. W. Moore, J. Schneider, and K. Deng. Efficient locally weighted polynomial regression predictions. In D. Fisher, editor, *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 196–204, San Francisco, 1997. Morgan Kaufmann.

[11] D. Pelleg and A. W. Moore. Accelerating Exact k-means Algorithms with Geometric Reasoning. In *Proceedings of the Fifth International Conference on Knowledge Discovery and Data Mining.* AAAI Press, 1999.

[12] F. P. Preparata and M. Shamos. *Computational Geometry.* Springer-Verlag, 1985.

[13] A. Szalay. Personal Communication. 2000.

[14] I. Szapudi. A New Method for Calculating Counts in Cells. *The Astrophysical Journal*, 1997.

[15] I. Szapudi, S. Colombi, and F. Bernardeau. Cosmic Statistics of Statistics. *Monthly Notices of the Royal Astronomical Society*, 1999.

[16] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: An Efficient Data Clustering Method for Very Large Databases. In *Proceedings of the Fifteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems : PODS 1996.* Assn for Computing Machinery, 1996.

---

of computational geometry.) The methods are easily generalized to the case of finding the k nearest neighbors, as in k-NN classification and locally weighted regression. Outlier detection is one of the most common statistical operations encountered in data analysis. The question of which procedure is most correct is an open and active one. We present here a natural operation which might be used directly for outlier detection, or within another procedure: for each of the points, find the number of other points that are within distance r of it - those having zero neighbors within r are defined as outliers. (This is exactly the all-range-count problem.)