Journal of Computational Science 23 (2017) 109–117

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Review of data structures for computationally efficient nearest-neighbour entropy estimators for large systems with periodic boundary conditions

Joshua M. Brown [a,*], Terry Bossomaier [b], Lionel Barnett [c]

[a] School of Computing & Mathematics, Charles Sturt University, Panorama Avenue, Bathurst, New South Wales 2795, Australia
[b] Centre for Research in Complex Systems, Charles Sturt University, Panorama Avenue, Bathurst, New South Wales 2795, Australia
[c] Sackler Centre for Consciousness Science, Department of Informatics, University of Sussex Brighton, United Kingdom

ARTICLE INFO

Article history:
Received 20 January 2017
Received in revised form 17 October 2017
Accepted 20 October 2017
Available online 26 October 2017

MSC:
68P05
68Q25
94A17

Keywords:
Information theory
Transfer entropy
Periodic boundary conditions
Spatial partitioning

ABSTRACT

Information theoretic quantities are extremely useful in discovering relationships between two or more data sets. One popular method—particularly for continuous systems—for estimating these quantities is the nearest neighbour estimators. When system sizes are very large or the systems have periodic boundary conditions issues with performance and correctness surface, however solutions are known for each problem. Here we show that these solutions are inappropriate in systems that simultaneously contain both features and discuss a lesser known alternative solution involving *Vantage Point* trees that is capable of addressing both issues.

1. Introduction

Information theory provides three useful metrics for determining relationships between data sets. *Mutual Information* (**I**) [25] measures the shared uncertainty between two variables and is often used as a marker for second-order phase transitions [21,12,30]. Meanwhile, *Transfer Entropy* (**T**) [24] and *Global Transfer Entropy* (**G**) [1] track the flow of information from one or more data sets onto another, with Barnett et al. demonstrating that **G** can be used as a predictor for an oncoming phase transition in the [15] spin model when system temperature is reduced over time.

If the underlying distribution for the data is known, these quantities can be straightforward to calculate. However, in general this distribution is unknown and can only be approximated from sampled data, thus requiring *Entropy Estimators*. Two well known approaches are the plug-in and nearest-neighbour (NN) estimators.

While simple, plug-in estimators tend to be less accurate, particularly for continuous variables [23] which are the focus of this paper. A high quality alternative is the Kozachenko–Leonenko entropy estimator [16] which uses nearest neighbour statistics of realisations of a data series. Kraskov et al. [17] extended this for estimating **I**, with **T** and **G** estimators further derived by [9]. For higher dimensional data, NN estimators become even more appropriate as plug-in estimators experience a combinatorial explosion in storage requirements.

On the other hand, the time complexity for the plug-in estimator scales linearly on the number of points, N, while NN estimators—using a direct method, where distances between each and every point are considered—scale on the order of N^2, which becomes significant for large N. This problem is widely encountered in the field of computer graphics and collision detection in computer games. The accepted solution for these scenarios are spatial partitioning data structures—such as well-known BSP Trees [6] and KD Trees [2]—resulting in a *"broad-phase"* collision detection step, which provides the ability to discard large swathes of points from consideration with relatively little computation.

* Corresponding author.
E-mail address: jbrown@csu.edu.au (J.M. Brown).

https://doi.org/10.1016/j.jocs.2017.10.019
1877-7503/

Table 1
Estimated and exact (italicised) values to 4 decimal places for **I**, **T**, and **G** using 1×10^4 realisations for variables X, Y, W drawn from the listed bi- and multi-variate distributions and auto-correlated processes. Evaluation performed over 500 repetitions to establish the mean estimate—and thus bias—and standard error of the estimator († Multiplied by 1×10^4 for readability). Bias is dependent on number of realisations and covariance of random variables, with improved bias as realisations increase and covariance decreases. *Gaussian, LogNormal and Cauchy distributions performed with covariance, $r = 0$ (i.e., $\Sigma = I$) and $r = 0.9$. Parameters for Von Mises and Hahs & Pethel evaluations found in text.

Distribution*		Metric					
		I	s. e. †	**T**	s. e. †	**G**	s. e. †
Gaussian	$r = 0.0$	-1×10^{-5} *0.0*	3.4	0.0001 *0.0*	3.5	−0.0004 *0.0*	3.2
	$r = 0.9$	0.8336 *0.8304*	5.1	0.1276 *0.1270*	3.9	0.1827 *0.1816*	4.0
LogNormal	$r = 0.0$	5×10^{-6} *0.0*	3.4	4×10^{-6} *0.0*	3.4	−0.0003 *0.0*	3.0
	$r = 0.9$	0.9773 *0.9741*	5.1	0.1326 *0.1314*	3.7	0.1906 *0.1871*	4.0
Cauchy	$r = 0.0$	0.1933 *0.2242*	4.6	0.0843 *0.0827*	4.1	0.1121 *0.1251*	4.0
	$r = 0.9$	1.0438 *1.0545*	7.2	0.1880 *0.2098*	4.6	0.2602 *0.3067*	4.9
Uniform		−0.0005 *0.0*	3.4	-9×10^{-5} *0.0*	3.4	−0.0003 *0.0*	3.2
Uniform wrapped		−0.0005 *0.0*	3.4	1×10^{-5} *0.0*	3.4	−0.0002 *0.0*	3.2
Hahs &Pethel		N/A	–	0.1659 *0.1651*	3.7	N/A	–
Von Mises	a	0.3489 *0.3488*	4.3	N/A	–	N/A	–
	b	0.4384 *0.4384*	4.5	N/A	–	N/A	–
	c	0.1940 *0.1928*	4.3	N/A	–	N/A	–

KD Trees are directly applicable to the nearest neighbour searches required by the NN estimators, and are commonly applied when data sets are large. However, when the data lies in a space with periodic boundaries—e.g., headings of observed objects, where $2\pi = 0$—the basic KD Tree fails. While there are modifications to overcome this, it is shown later that ultimately a more appropriate data structure is needed and available in the *Vantage Point* (VP) Trees developed—independently—by [26,32].

This paper serves to highlight the relative computational performance of the VP Tree compared with a modified KD Tree for the specific use case of measuring **I**, **T** and **G** of large periodic systems via a NN estimator. This use case also presents a unique constraint in that the search structures are typically constructed and used just one time—meaning construction costs are just as important as search costs.

To establish the accuracy of the VP Tree approach, tests are performed against periodic and aperiodic canonical distributions as done by [17]. To further analyse the use of the VP Tree for large data sets with periodic boundary conditions, data sets—generated by the widely-used Vicsek model of self-propelled particles [28]—are analysed. Lastly, work by [8] stresses that for accurate estimation, the NN estimator requires a number of points scaling exponentially with the true **I**. A decimation of the Vicsek data is employed to investigate this issue.

The three aforementioned information theoretic metrics are widely used in the scientific community, where mutual information has been used in many applications, including the study of phase transitions in financial markets [12], while the range of applications for transfer entropy is building steadily, from the study of physics of fundamental systems, such as the Ising spin model [1] to cellular automata [20], random Boolean networks [19], and swarms [29]. Ensuring the accuracy of the VP Tree method in periodic systems is important as it will allow processing of larger data sets with fewer computational resources—which is particularly important in the age of big data.

2. Methods

The most simple of the three quantities, **I**, measures the amount of uncertainty common to two variables, say the headings of particles in a simulation. The input for an estimator for **I** is a set of data points, (x, y), drawn from the two variables. **T** instead measures the flow of information between two variables, that is, the reduction in uncertainty of X given past knowledge of X and Y. Estimators for **T** thus requires a time series of data points (x_t, x_{t-1}, y_{t-1}). Finally, **G** measures a similar statistic as **T**, but is used in the case where X is influenced by multiple variables Y^{d-2}, and therefore input to these estimators is d-dimensional data set, $(x_t, x_{t-1}, y^1_{t-1}, \ldots, y^{d-2}_{t-1})$.

The standard forms of the three metrics are [25,24,1]:

$$\mathbf{I}(X : Y) = \mathbf{H}(X) + \mathbf{H}(Y) - \mathbf{H}(X, Y), \tag{1}$$

$$\begin{aligned} \mathbf{T}_{Y \to X} &= -\mathbf{H}(X_t, Y_{t-1}, X_{t-1}) + \mathbf{H}(X_t, X_{t-1}) \\ &\quad + \mathbf{H}(X_{t-1}, Y_{t-1}) - \mathbf{H}(X_{t-1}), \end{aligned} \tag{2}$$

$$\mathbf{G} = \frac{1}{N} \sum_{i}^{N} \mathbf{T}_{Y'_i \to X_i}, \tag{3}$$

where $\mathbf{H}(Z) = -\sum p(z) \log_2 p(z)$. **I** and **T** are pairwise quantities, and as such, can be calculated using just two variables—and further averaged over every pairwise combination of variables if the data set contains many interacting variables. **G** on the other hand is a global quantity, so each random variable is considered in turn for the target variable. Y'_i is thus the set of all influencing variables on the chosen target, X_i.

Throughout this work, X and Y are drawn from a variety of sources where: (a) analytic results are available, (b) prior results for

Table 2
Closed form analytic expressions for **I**, **T**, and **G**. Note that closed form expressions for the Von Mises distribution and the Hahs & Pethel auto-regressive process require more context than can reasonably be provided here and as such the reader is referred to the original material for this context.

Distribution	Closed form
$\mathbf{I}_{Gauss}$, $\mathbf{I}_{LogNormal}$	$-\frac{1}{2}\log(1-r^2)$
$\mathbf{T}_{Gauss}$, $\mathbf{T}_{LogNormal}$	$\frac{1}{2}[-\log\lvert\Sigma_{xyw}\rvert+\log\lvert\Sigma_{xy}\rvert+\log\lvert\Sigma_{xw}\rvert]$
$\mathbf{G}_{Gauss}$, $\mathbf{G}_{LogNormal}$	$\frac{1}{2}[-\log\lvert\Sigma_{xy'w}\rvert+\log\lvert\Sigma_{xy'}\rvert+\log\lvert\Sigma_{xw}\rvert]$
$\mathbf{I}_{Cauchy}$	$\log(8\pi)-3-\frac{1}{2}\log r$
$\mathbf{T}_{Cauchy}$	$4-\log(16\pi)+\frac{1}{2}[-\log\lvert\Sigma_{xyw}\rvert+\log\lvert\Sigma_{xy}\rvert+\log\lvert\Sigma_{xw}\rvert]$
$\mathbf{G}_{Cauchy}$	$3+\log\frac{\Gamma\left(\frac{1+d}{2}\right)}{4\Gamma\left(\frac{d}{2}\right)\sqrt{\pi}}-\frac{d+1}{2}\psi\left(\frac{d+1}{2}\right)+\frac{d}{2}\psi\left(\frac{d}{2}\right)+\frac{1}{2}\psi(1)$ $+\frac{1}{2}[-\log\lvert\Sigma_{xyw}\rvert+\log\lvert\Sigma_{xy}\rvert+\log\lvert\Sigma_{xw}\rvert]$
Uniform	0
Uniform wrapped	0
Hahs & Pethel	See [10]
Von Mises	See [14]

Where r is the scalar covariance between X and Y and Σ is the covariance matrix between the subscripted variables. Additionally, $\Gamma(\cdot)$ is the Gamma function while $\psi(\cdot)$ is the digamma function. Information sharing and flow are non-existent for uniform distributions by definition and thus become 0.

metrics are available, or (c) statistics are unknown and hard to estimate. A collection of canonical distributions with analytic results and distributions with prior results are provided in Tables 1 and 2. Amongst these distributions is the ubiquitous Gaussian distribution, as well as the Von Mises distribution which is accepted as the circular analogue of the Gaussian distribution. The Vicsek Model (described in Section 2.2) is used to generate difficult-to-estimate data.

There are some optimisations available for **G** depending on the nature of the data. If the variables of the data set are indistinguishable then **G** can be calculated as a single ensemble calculation treating it as a multivariate **T** calculation, rather than N **T** calculations. This technique is employed in Section 3.1. A second such optimisation is dependent on how variables influence each other. In the case of the Vicsek model used later, variables influence one another in an aggregate fashion from which a consensus variable, Y^c, can be constructed. Thus **G** can be calculated as the 3-dimensional $\mathbf{G}_{Y^c\to X}$, rather than the above $d+2$-dimensional form. This reduction is utilised in Sections 3.2 and 3.3.

2.1. Nearest neighbour method

The plug-in estimation approach uses these data sets to determine the underlying distribution often via a histogram. As mentioned in the introduction, once this distribution is revealed it is often trivial to solve the above equations. However, working with continuous data presents certain issues for traditional plug-in histogram entropy estimation. For example, tuning the bin width parameter for discretising the continuous data is susceptible to the number of points available, which can introduce error at high or low noise if the bin width is too low or high, respectively. While adaptive binning can mitigate this somewhat, it is not always an option, particularly when the data covers the entire domain.

Instead, one can use Nearest-Neighbour Entropy Estimators. These estimators are based on the (continuous) statistics of the k nearest neighbours to each data point and the distribution of points along marginal dimensions as introduced by [16]. In these methods, the max-norm distance, $\epsilon(i)$, is found for each point i and its kth nearest neighbour (kNN search). A fixed range (FR) search is then performed to count the number of points within the (hyper-)*"ball"* defined by $\epsilon(i)$ from i along the marginal spaces. These values are then used to compute the above information theoretic quantities. See Fig. 1 for an example of collecting nearest neighbour information.

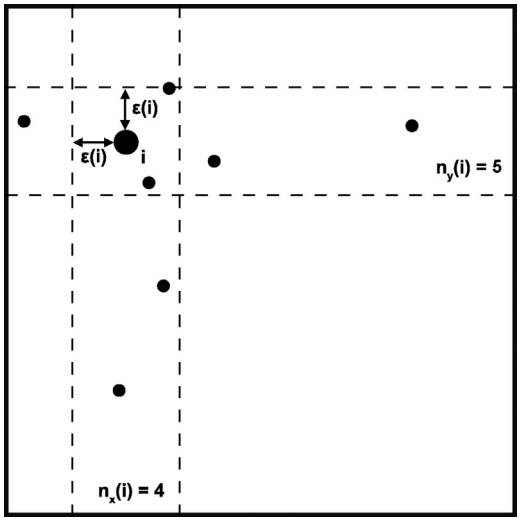


Fig. 1. Determining the kth nearest neighbour—$k=2$ in this case—and $\epsilon(i)$ for some point i and then count the number of points in the marginals strictly within these bounds, with $n_x(i)=4$ and $n_y(i)=5$, after [17].

Krasov et al. [17] provide the nearest neighbour estimator for **I** as:

$$\mathbf{I}(X:Y)=\psi(k)+\psi(N)-\langle\psi(n_x(i)+1)+\psi(n_y(i)+1)\rangle, \tag{4}$$

where $\psi(\cdot)$ is the digamma function, $n_z(i)$ is the count of points within the distance $\epsilon(i)$ from point i in the marginal space z (either x or y), and $\langle\ldots\rangle$ is the average over all points.

The extension to this for **T** and **G** is provided by [9], and requires computing:

$$\mathbf{T}_{Y\to X}=\psi(k)-\langle\psi(n_{xw}(i)+1)+\psi(n_{xy}(i)+1)-\psi(n_x(i)+1)\rangle, \tag{5}$$

$$\mathbf{G}=\psi(k)-\langle\psi(n_{xw}(i)+1)+\psi(n_{xy'}(i)+1)-\psi(n_x(i)+1)\rangle, \tag{6}$$

where x and y represent the state at $t-1$, w represents the current state of x at t and y' represents the set of Y_{t-1}^{d-2} variables used for the **G**. n_{xw} is thus the number of points that exist within the distance $\epsilon(i)$ from point i when only considering the x_{t-1} and x_t coordinates of the data set.

Each of the above quantities requires N kNN searches and either $2N$ or $3N$ FR searches. In a naïve approach, each of these searches is a $\mathcal{O}(N^2)$ operation, clearly making this approach more computationally expensive than simple plug-in estimators. However, by choosing more appropriate searching algorithms we can significantly reduce the cost of these estimators such that they become a feasible approach.

2.1.1. Underlying data structure

The naïve approach for finding the kth nearest neighbours and the marginal neighbour counts for point i is to calculate the distance between i and every other point, and only keep those points passing the criteria—i.e., amongst the k closest points, or within distance $\epsilon(i)$. This approach requires N comparisons for all N points, resulting in a computational complexity of $\mathcal{O}(N^2)$ and as such is not a suitable approach when N is large.

A common solution to this problem in computer graphics are data structures such as the KD Tree [2], which subdivide a space using lines to define front/back areas, as seen in Fig. 2. When considering point i, one can quickly cull large sections of space—those containing points which cannot possibly be among the k nearest neighbours, or within the bounds $\epsilon(i)$—from further consideration. This approach reduces the complexity to $\mathcal{O}(N\log_2 N)$ [5]. The subdivision process creates a tree hierarchy which can be seen in Fig. 3.

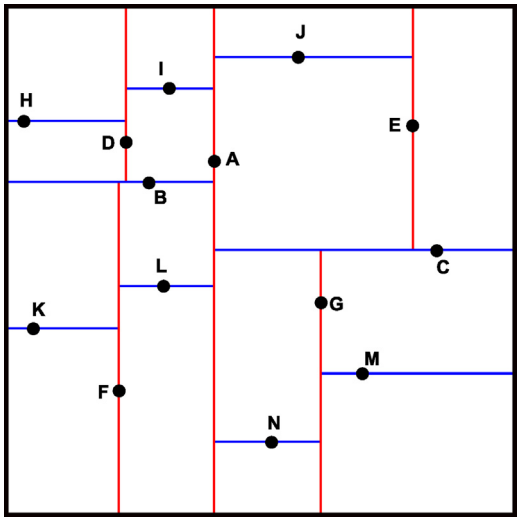


Fig. 2. KD Tree constructed via lines intersecting each point segregate the local area of the point into so-called *front* and *back* areas. In this example, point A is the root node in the binary tree, with points B and C being its immediate child nodes, as seen in Fig. 3.

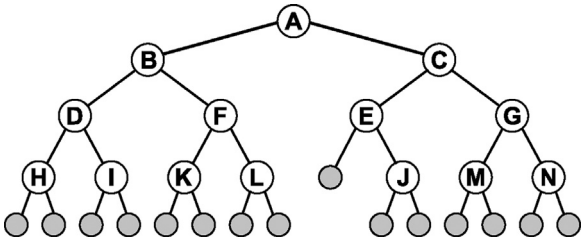


Fig. 3. Tree structure created by subdivision of space in KD Tree. All nodes descending from the left of A appear in front of A in the space, while those descending from the right of A appear behind A. Grey leaf nodes represent the areas enclosed by the subdivisions. An algorithm for traversing this tree, presented by [5], allows finding the kth nearest neighbour in $\log_2 N$ time, rather than N^2.

However, when the data lies in a space with periodic boundary conditions a structure such as a KD Tree will incorrectly cull points. Since the space wraps, a point is both in front of and behind any other point, while the KD Tree only considers points up to the boundary and not beyond. Three potential methods for correcting this—using Images, a hybrid involving naïve search and KD Trees, and VP Trees—are discussed below.

The images method duplicates and shifts the data—thus creating cloned images—such that when the KD Tree is constructed from the combination of the original data set and all images, it has the illusion of periodicity. This requires $3^d - 1$ duplications for a d-dimensional space which can quickly become intractable, especially for **G**. Note however, that typically only the *shortest* distance between points i and j are considered and as such this can be reduced to $2^d - 1$ by only duplicating the closest region (half/quarter/eighth/etc.) as needed. Some specific cases—perhaps where direction is a factor—might require $3^d - 1$ full duplications, however, this will not be considered here, due to both the difficulty in imagining such a scenario—implying the data is both circular and *pairwise* ordered—as well as the fact that the two following approaches are only capable of solving the shortest distance form.

The second method—a hybrid of duplication and naïve search—requires only a single duplication, but can degenerate to a naïve search for select points. The image in this method—queried separately—is shifted such that points originally at the corners are now in the centre of the space, and vice versa—i.e., each dimension is shifted by half of its width. When querying x_i (kNN or FR), if it is closer to the boundary than $\epsilon(i)$—i.e., there are possibly points just on the other side of the boundary closer than $\epsilon(i)$—the query is repeated in the image. The boundary check is repeated on the image result, and if it fails again, the query steps down to a naïve search. This method requires less duplication than the previous method but requires more computation per point.

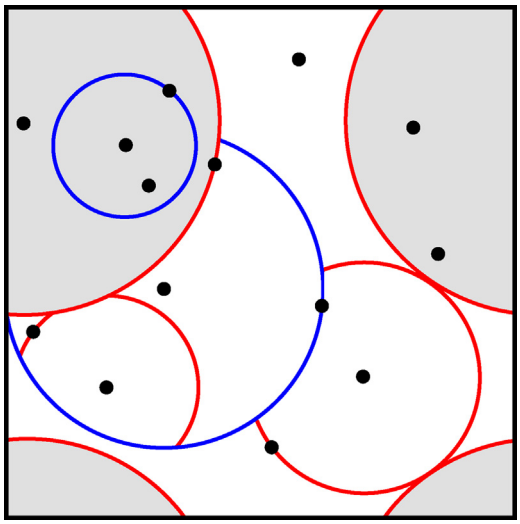


Fig. 4. Circles centred on vantage points delineating near/far subsets. Note that child circles do not cross the boundaries of parent circles, as vantage points only consider points in the same subset as itself. Shaded region represents the near subset of the root node (top left), accounting for periodic boundary conditions.

Instead of wrangling the KD Tree to handle periodicity, the third approach is to use a different structure. As already discussed, KD Trees cannot deal with periodic conditions because the nature of front and back areas in periodic spaces is ill-defined. VP Trees work instead by choosing a point, v (the vantage point), and then subdividing the space into those points nearer to v and those points farther from v than a given threshold. Since distances—and thus near/far points—are trivial to calculate in a periodic space, this structure can be used to attain efficient searching without resorting to any tricks. To create a balanced tree, the threshold is chosen such that it divides the points in half. The subdivision process is repeated on each new subset with new vantage points and thresholds chosen from only those points in the subregion. Fig. 4 shows how this subdivision strategy works in an example space. Note that the hierarchical structure of this tree is the same as the KD Tree, just with the left children being near—instead of in front—and the right children being far.

One drawback to this near/far dichotomy is that a new tree is needed for each set of marginals as the dimension reduction can change the distance between points. That is, if the max-norm distance between d-dimensional realisations, $||x_i - x_j||_\infty$, is the distance along dimension d_α, then a marginal subset that does not include d_α will have a different distance between x_i and x_j. Similar arguments apply to other norms.

Thus a separate tree is needed for each kNN and FR search—totaling three for **I** and four for **T** and **G**. However, the FR searches are independent of each other, thus if the kNN search is performed and the ϵ-ball sizes stored the remaining trees can be used in serial. Since each tree is only used once (with all N points processed), only one tree is ever needed at any given time, reducing the space requirements for this approach to $2N$, rather than $3N$ or $4N$.

When performing an FR search in just one dimension (i.e., for counting n_x and n_y), it was found that performing a binary search to find the range of elements covered by $\epsilon(i)$—and thus the neighbour count—was faster than VP Trees. As such in the one dimensional

searches, all test cases revert to this simpler algorithm. This also reduces the number of VP Trees required to one for **I** and three for **T** and **G**.

The KD Tree implementation used in this paper was provided by the ANN library [22], which is a standard package in the area of nearest neighbour searches with over 330 citations on *Google Scholar*, while the VP Tree implementation is a slightly modified version of the public domain code by [11]. Our software is provided online. Both implementations are written in C++.

It should also be noted that other structures and approaches exist for solving this issue. For example, the periodic KD Tree library [27] (written in Python) instead duplicates the query points rather than the data points. While this halves the storage requirements, each query point needs to be processed 2^d times (in the typical case) or $3^d - 1$ times (in the general, if not odd, case), rather than 1–3 times as in the hybrid model. This can be an obstacle in the case of measuring information theoretic quantities where each data point is also a query point (i.e., $Q = N$ rather than $Q \ll N$). Another two structures capable of addressing the kNN problem are Locality Sensitive Hashing (LSH) [33] and higher order Voronoi Diagrams (specifically, k-order) [18,4]. For the purposes of estimating information theory quantities however, these algorithms will not perform significantly better than the KD Tree or VP Tree due to space requirements, the one-shot nature of the estimation (that is, only using each structure once), and the distribution of data points. As such, the authors shall simply note their existence (and better performance in other use cases) and consider just the KD Tree and VP Tree structures for the remainder of this paper.

2.2. Vicsek model

The standard Vicsek Model (SVM) [28] is a simple model that is widely used in collective motion studies. The simplicity of the model allows it to easily scale to large system sizes (both in terms of time and space). As such, it easily generates a large number of points—with periodic boundary conditions—perfect for testing VP trees. In the model, a flock of M particles move in a continuous space (off-lattice) with periodic boundary conditions of linear size L. Particle density is defined as $\rho = \frac{N}{L^2}$. Particles move with constant speed, s, and are simulated for τ discrete time steps. Formally, the particles are updated according to:

$$\boldsymbol{x}_i(t + \Delta t) = \boldsymbol{x}_i(t) + \boldsymbol{v}_i(t)\Delta t, \tag{7}$$

$$\theta_i(t + \Delta t) = \langle \theta_i(t) \rangle + \omega_i(t), \tag{8}$$

where $\boldsymbol{x}_i(t)$ is the position of particle i at time t, $\boldsymbol{v}_i(t)$ is its velocity, which is constructed with speed s and heading $\theta_i(t)$. $\langle \theta_i(t) \rangle$ defines the average angle of all particles within $r = 1$ units of i (including itself) and $\omega_i(t)$ is a realisation of uniform noise on the range $\left[-\frac{\eta}{2}, \frac{\eta}{2}\right]$, where η is our temperature variable and $0 \leq \eta \leq 2\pi$.

Particle alignment is measured via the instantaneous order (or disorder) using the parameter:

$$\varphi(t) = \frac{1}{M} \left| \sum_i^M \boldsymbol{v}_i(t) \right|, \tag{9}$$

where $0 \leq \varphi(t) \leq 1$. When $\varphi(t) = 1$, the flock is completely aligned and collective motion has been attained, while $\varphi(t) = 0$ represents the disordered, chaotic state of particles moving with no alignment. $\varphi(t) = 1$ is associated with no noise in the system—i.e., $\eta \to 0$—while $\varphi(t) = 0$ occurs at high noise, $\eta \to 2\pi$.

Note that while particle positions are indeed in a wrapped space, the random variables evaluated below are actually the heading *angles* of neighbouring (interacting) particles. That is, the quantities describe the various reductions in uncertainty of a particle's heading given knowledge of its past heading and its neighbours' headings, not their positions. To measure the information theoretic quantities—using the notation from Eqs. (4)–(6))—X is defined as the heading of particle i, Y as the heading of an interacting neighbour particle j, and W as the next heading of particle i. In the case of **I** and **T**, which are pairwise quantities, this is repeated for each interacting neighbour j for every time step t. On the other hand, **G** measures all j interacting neighbours in one multi-dimensional variable Y', and thus generates just a single point. Particle indistinguishability is employed and repeated the above for all i, coalescing all points into a single point cloud. The pairwise quantities generate a separate data point for all (i, j) interacting pairs, which is not constant over time or noise. See Fig. 5 for a visualisation of how the number of interacting pairs changes as a function of noise temperature, η. **G** on the other hand generates a single point per i for all of its interacting neighbours at time t and thus contains precisely τM data points at all noise values.

There is one major difficulty in calculating **G** however. **G** uses all interacting neighbours, N_I, to particle i at time step t to define the multivariate Y'—i.e., $d = N_I$. N_I is not constant which causes issues when collecting all realisations into a single point cloud for calculation. One solution is to pick an arbitrary value of d—say the minimum, maximum or mean of N_I—however these options will miss pertinent variables and thus give an inaccurate result, or will include irrelevant particles and thus increase the complexity of the problem. The alternative is to note that particles only influence each other via the consensus heading, $\langle \theta_i(t) \rangle$, and to use it instead of each particle individually. The latter approach is used for all Vicsek results below.

Finally, to provide an example for the earlier claim regarding adaptive binning techniques for plug-in estimators, note that the average heading of Vicsek flocks will proceed on a random walk about the unit circle over time. As such, the angles generated by each particle will be roughly uniform over each 2π marginal (although in the joint space, accumulates around the $x \approx y \approx w$ diagonal), degenerating adaptive bins to simple fixed-width bins.

3. Results

3.1. Accuracy of NN estimator

To ensure the integrity of the VP Tree implementation of the NN estimator, **I**, **T** and **G** are estimated from data series sampled from distributions with known closed form entropies. Krasov et al. [17] performed this test for **I** against Gaussian data with multiple covariances, which is repeated and extended here. This establishes the accuracy of the estimator for the **T** and **G** as well. The estimator is also tested against the Log-Normal, Cauchy (Via a Student's t distribution with $\nu = 1$ degree of freedom) and Uniform distributions. For these tests—and all tests below—k is set to 3 as per the suggestion of [17], who propose the choice of $k = 2$–4 as it provides a good balance between statistical and systematic errors in the NN estimator. For larger systems, larger k is allowed as systematic errors tend to 0. Simulations with $k = 30$ confirmed no change in estimate, and so only $k = 3$ is presented below.

To extend the test to account for periodic cases, all three metrics are tested against a wrapped uniform distribution and **I** against the von Mises distribution (sine variant). Exact **I** for the von Mises distribution is provided in Table 1 of [14], for three parameter sets: (a) $\kappa_1 = 10$, $\kappa_2 = 15$, $\lambda = 10$, (b) $\kappa_1 = 15$, $\kappa_2 = 12$, $\lambda = 12$, and (c) $\kappa_1 = 12$, $\kappa_2 = 14$, $\lambda = -8$, with $\mu_1 = \mu_2 = \pi$ for all three sets.

Lastly, the estimator is tested against the autoregressive process described by [10] (Example 1) which includes an analytical expression for **T**. Variances of the Gaussian noise terms in each process were $Q = 1$, $R = 2$ for X and Y, respectively, with $a = 0.9$ and $h_c = 1$, and an initial x value of $x_0 = 0$.

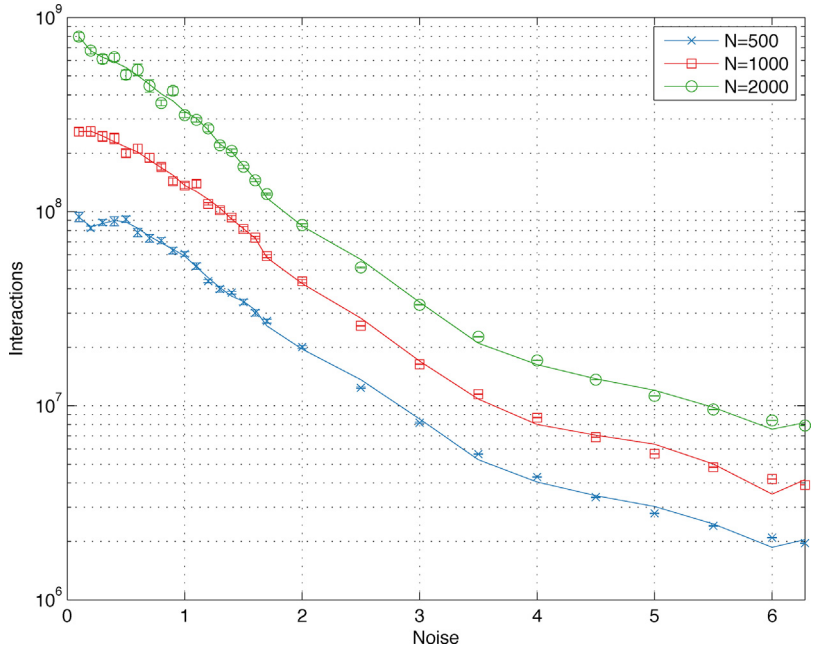


Fig. 5. Logarithmic scale of number of interactions, N_I, for Vicsek system with parameters $\tau = 5000$, $\rho = 0.25$, $s = 0.1$ for $N = 500, 1000, 2000$. Error bars represent standard deviation over 10 repetitions.

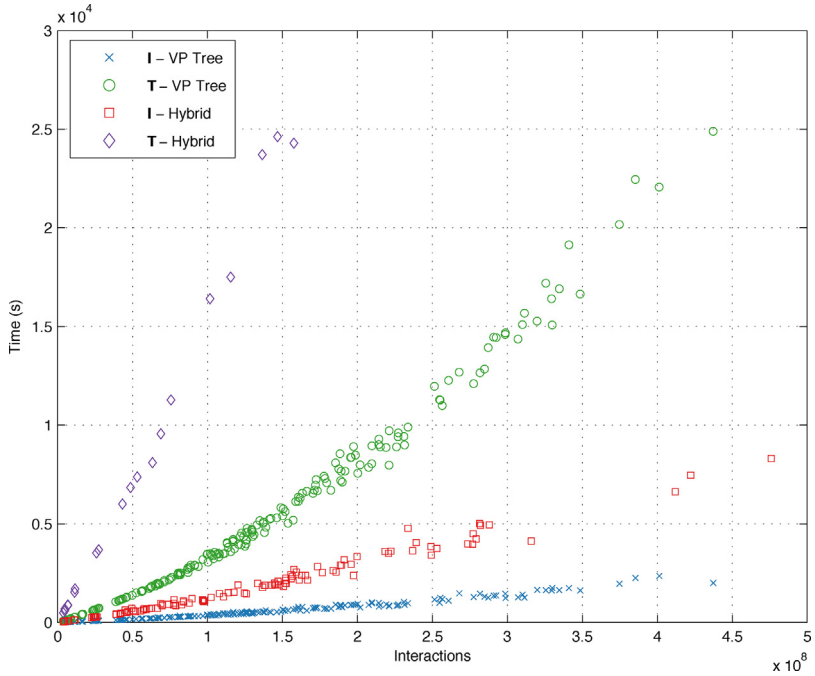


Fig. 6. Time (s) vs N_I measured for a Vicsek system with parameters: $N = 1000$, $\tau = 5000$, $\rho = 0.25$, $s = 0.1$ using VP Tree and hybrid methods.

The estimated and exact values for these distributions can be seen in Table 1. Estimates were measured from 1×10^4 realisations and repeated 500 times. Note that as the number of realisations increases, the estimate values converge to the exact values, as observed in [17]. Closed form expressions for the information theoretic quantities can seen in Table 2.

Given the results in Table 1, and the converging nature as the number of realisations is increased, it is determined that the VP Tree implementation of the NN estimator is accurate.

3.2. Comparison of underlying data structure

Yianlilos [32] and Friedman et al. [5] provide *Big* $\mathcal{O}$ analysis of VP Trees and KD Trees, respectively, showing both can be constructed and searched in $\mathcal{O}(N\log_2 N)$ time and use $\mathcal{O}(N)$ space. Here, estimation of the complexity of the hybrid KD Tree approach is provided, followed by empirical data comparing computation and space of the VP Tree and hybrid approaches.

As mentioned above, the metrics for the Vicsek model are calculated from a two or three dimensional space with periodic boundary

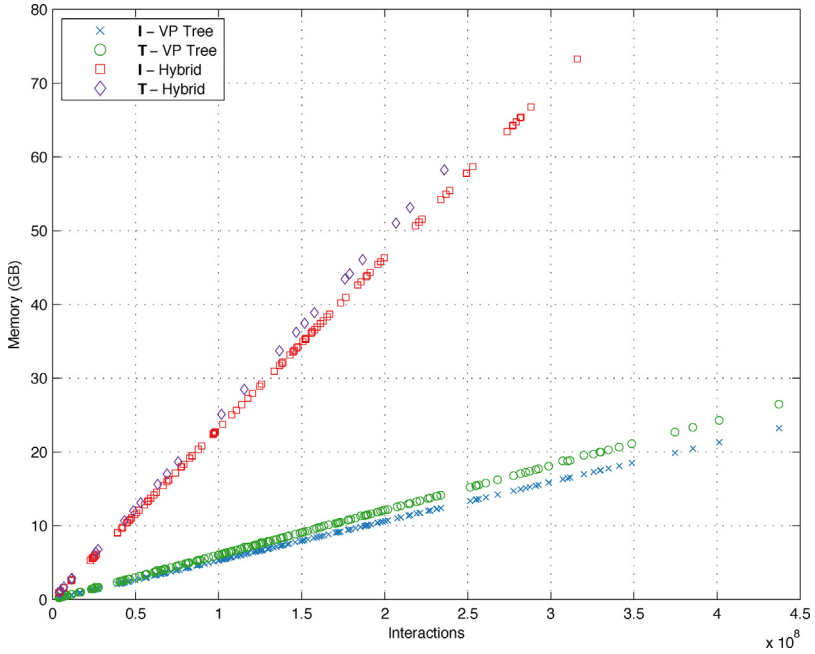


Fig. 7. Memory usage (GB) vs N_I measured for a Vicsek system with parameters: $N = 1000$, $\tau = 5000$, $\rho = 0.25$, $s = 0.1$ using VP Tree and hybrid methods.

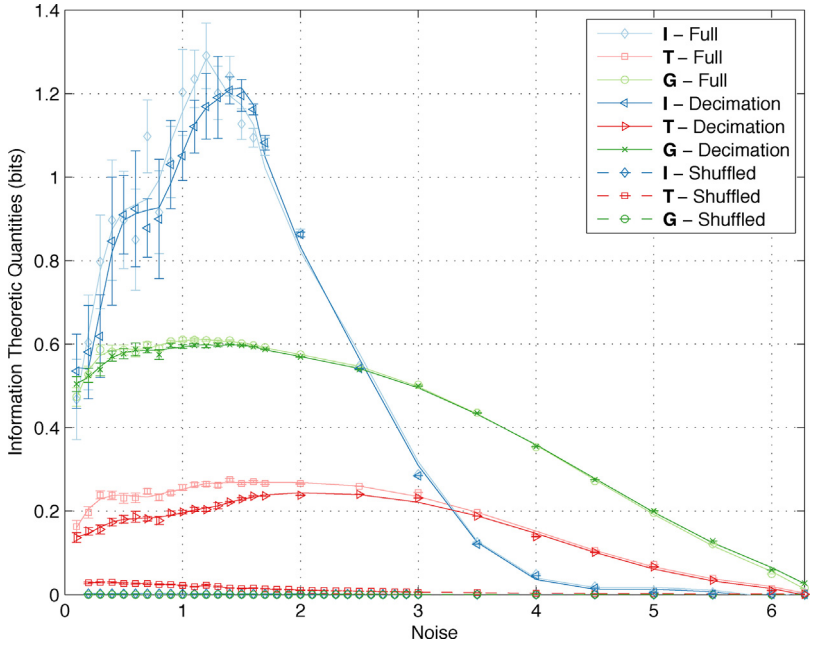


Fig. 8. Results of random shuffle (dashed) and decimation (solid bold) for a Vicsek system with parameters: $N = 1000$, $\tau = 5000$, $\rho = 0.25$, $s = 0.1$. Results for unmodified data shown washed out. Shuffling has removed almost all information sharing between variables as evidenced by the near zero value for all three metrics over all noise. Decimation of data (using random 10% of available data) has little impact on results.

conditions, with sides $S = 2\pi$. To perform kNN searches the hybrid method constructs two KD Trees: κ partitions the points N_I, while κ' partitions the points N_I'—the original points shifted by π along every dimension, with wrapping.

To refresh, the hybrid search works as follows: search the first tree, κ, for the kth nearest neighbour to i, which gives a distance, $\epsilon(i)$. If i is closer than $\epsilon(i)$ units to the boundary then the search progresses to the equivalent point i' in the second tree, κ', to find a corresponding distance, $\epsilon'(i)$. If i' is also closer than $\epsilon'(i)$ units to the boundary, the search reverts to a naïve linear search over all N points.

Thus 4 areas can be defined. First is the total area, $A_T = (2\pi)^D$, containing all points. Second is the inner area, A_I given by the box of sides $2\pi - 2\epsilon$ centred in the space, containing all points successfully processed by κ in the first step. Following this is the border area, $A_B = A_T - A_I$, covering those points that κ' will process (successfully or otherwise). Finally, the naïve area, A_N, contains those points that will ultimately be processed using the linear search, and is defined as the area within a max-norm distance of ϵ from the boundary midpoints, such that $A_N = D2^D\epsilon^2$.

Due to the tiered approach of the hybrid method, the time complexity will follow the form

$$N\log_2 N + \alpha N\log_2 N + \beta N^2, \tag{10}$$

where α is the proportion of points in the border area and β is the proportion of points in the naïve area. Note that $0 \leq \beta \leq \alpha \leq 1$.

In the high noise Vicsek case, where N_I is uniformly distributed with density $\rho_I = N/(2\pi)^D$ the average distance to the kth nearest neighbour is [3,7,13]:

$$\begin{aligned}\epsilon &= c\left(\frac{k}{\rho_I}\right)^{\frac{1}{D}} \\ &= c\left(\frac{k(2\pi)^D}{N}\right)^{\frac{1}{D}},\end{aligned} \tag{11}$$

where $c = \frac{7}{15}$. This allows further expansion of A_I, A_B and A_N.

For the high noise uniform case the proportions are simply $\alpha = A_B/A_T$ and $\beta = A_N/A_T$, which gives time complexity of

$$(N + 4c\sqrt{kN} - 4c^2k)\log_2 N + 8ckN \tag{12}$$

for the two-dimensional **I** case and

$$(N + 6ck^{1/3}N^{2/3} - 12c^2k^{2/3}N^{1/3} + 8c^3k)\log_2 N + 24c^3kN \tag{13}$$

for the three-dimensional **T** case.

In other distributions, such as the low noise Vicsek model, α and β will change. In the specific case of low noise Vicsek data, points will accumulate along the $x = y$ and $x = y = w$ diagonals for **I** and **T**, respectively. This is ideal as it results in minimal points in A_N—i.e., $\beta \approx 0$. Furthermore, the amount of points in A_B will also be significantly reduced since only points near the corners—that is near $x \approx y \approx w \approx 0 = 2\pi$—will be within the required threshold, with $\alpha \approx \frac{5\epsilon^D}{N}$, noting that ϵ will not match Eq. (11) due to the distribution change.

Given that $k \ll N$, it is clear that Eqs. (12) and (13) are of order $\mathcal{O}(N\log_2 N)$ and thus equivalent to the VP Tree. In practical terms, however, a non-zero α and β as well as requiring twice the construction time—which is $\mathcal{O}(N\log_2 N)$ itself—will result in lower processing throughput.

To demonstrate the difference between the two approaches, empirical data is provided below where Vicsek data sets are analysed against the number of interactions generated. Only **I** and **T** are investigated as these have variable N_I, while **G** has constant $N_I = \tau M$ and thus does not provide any additional insight. As mentioned earlier, both algorithm implementations are provided with off the shelf software—with slight modification of the VP Tree implementation such that it uses contiguous memory allocation similar to the KD Tree implementation.

Fig. 6 shows the time taken to calculate **I** and **T** for a range of N_I. While all four measurements seem to scale according to $N_I \log_2 N_I$—as described above—the VP Tree methods are faster. The difference in running time for calculation of **I** is strictly in finding the ϵ-ball sizes, as both FR searches are performed with the same binary search function. Calculation of **T** is worse still due to relying on the hybrid method more, with computations taking almost 10 h with the hybrid method compared to only 30 min under the VP Tree method.

Fig. 7 shows the memory usage for the same calculations as above. All four metrics are rigidly linear in their scaling—as expected—but there is a marked increase in memory usage between the two methods, with the hybrid method performing definitively worse with both metrics.

3.3. Additional checks of numerical stability

Two additional checks for numerical stability were performed with the Vicsek data—a random shuffle and a data decimation. These additional checks were not performed for the canonical distributions as exact results are known for these.

In the first check, a random shuffle is employed on the coordinates corresponding to the Y variable(s) for all three metrics. This should eliminate most of the information sharing between data sets and thus result in $\mathbf{I} \approx \mathbf{T} \approx \mathbf{G} \approx 0$. Results can be seen in Fig. 8.

The decimation method is widespread and is employed to test the precision of the NN estimator implementation in a manner analogous to cross-validation [31]. Specifically, data is generated from the Vicsek simulations, and a random subset containing 10% of the entire population is chosen, which were then processed with the NN estimators. This was repeated 10 times per run. Fig. 8 shows that there was not a significant impact from this decimation, in alignment with assertions by [8].

3.4. Periodicity in data structure

Hnizdo et al. [14] uses the ANN library for their KD Tree implementation, as here, in which they measure **I** for the circular von Mises distribution. Interestingly however, they make no mention of the inability for KD Trees to handle periodic situations. Furthermore, their parameters for the von Mises distribution use a mean of π for both random variables—with a periodic range of $[-\pi, \pi)$—meaning one would expect periodic artefacts to be quite pronounced as the distribution is centred on the border.

However, this is not the case. 500 sets of 1×10^4 realisations of parameter set c (Table 1) were tested with and without wrapping enabled in the VP Tree implementation. The estimated **I** for the two tests were 0.194013 and 0.194037 for wrapped and non-wrapped VP trees, respectively, with an exact result of 0.1928.

To see why the difference in results is not larger, consider not accounting for periodic conditions. This will only affect points near the boundary, where instead of counting the neighbours between $\pi \pm \epsilon(i)$—with wrapping handled—the process counts the neighbours in the range $[\pi - \epsilon(i)', \pi]$ (or $[-\pi, -\pi + \epsilon(i)']$), noting that $\epsilon(i)'$ will not necessarily be the same as $\epsilon(i)$ since it is likely the kth nearest neighbour will be a different point. In a data set centred around π, even though many points are near the boundary, only a small subset will cross the boundary in either the kNN or FR searches. Additionally, the mirrored density on either side of the boundary means that $n'_x \approx n_x$ for these points—likewise for n_y. Note that the actual distance to the kth nearest neighbour is not used in Eqs. (4)–(6), only the neighbour counts to which the digamma function is applied—which is approximately logarithmic for values over 10. These facts, combined with the result being averaged over all N leads to the small change in result when periodic boundaries are ignored.

To test for the mirrored density assertion above, the data set is shifted such that the data on the boundary is asymmetric. This is achieved using $\mu_1 = \mu_2 = \frac{7\pi}{8}$. The estimate for wrapped trees is (rightfully) unaffected, however the non-wrapped trees estimate worsens, to 0.194050.

4. Conclusion

In this paper, we discuss estimating information theoretic quantities for continuous variables using nearest neighbour estimators, specifically for systems with a large amount of circular data. The direct method of calculation is shown to be infeasible as it scales proportional to N^2. While spatial partitioning structures can reduce

this to $N \log_2 N$, special care must be taken when dealing with periodic boundary conditions.

We discuss three data structures which are capable of handling these boundary conditions and have demonstrated that choosing a more appropriate data structure like the VP Tree can significantly increase performance in terms of time and memory used. Furthermore, as the VP Tree is fundamentally similar to the KD Tree, instead relying on distance between points rather than their specific topology, it should perform no worse than the KD Tree in any metric space on any manifold of a closed form, such as a sphere or *Klein* bottle.

We have also shown that employing a decimation technique to reduce the number of points processed does not have a significant impact on estimation.

By combining the considerations put forth in this paper, estimation of **I**, **T**, and **G** for large systems with periodic boundary conditions is feasible and will show the same performance characteristics as systems with fewer constraints.

Acknowledgements

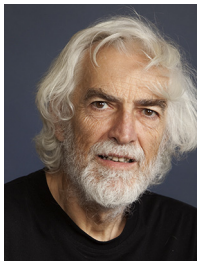
Joshua Brown would like to acknowledge the support of his Ph.D. program and this work from the Australian Government Research Training Program Scholarship.

The National Computing Infrastructure (NCI) facility provided computing time for the simulations under project e004, with part funding under Australian Research Council Linkage Infrastructure grant LE140100002.

References

[1] L. Barnett, M. Harré, J. Lizier, A.K. Seth, T. Bossomaier, Information flow in a kinetic Ising model peaks in the disordered phase, Phys. Rev. Lett. 111 (2013) 177203.
[2] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (1975) 509–517.
[3] P. Bhattacharyya, B.K. Chakrabarti, The mean distance to the nth neighbour in a uniform distribution of random points: an application of probability theory, Eur. J. Phys. 29 (2008) 639.
[4] R.A. Dwyer, Higher-dimensional Voronoi diagrams in linear expected time, Discrete Comput. Geom. 6 (1991) 343–367.
[5] J.H. Friedman, J.L. Bentley, R.A. Finkel, An algorithm for finding best matches in logarithmic expected time, ACM Trans. Math. Softw. 3 (1977) 209–226.
[6] H. Fuchs, Z.M. Kedem, B.F. Naylor, On visible surface generation by a priori tree structures, in: ACM Siggraph Computer Graphics, vol. 14, ACM, 1980, pp. 124–133.
[7] B. Gaboune, G. Laporte, F. Soumis, Expected distances between two uniformly distributed random points in rectangles and rectangular parallelepipeds, J. Oper. Res. Soc. 44 (1993) 513–519.
[8] S. Gao, G. Ver Steeg, A. Galstyan, Efficient estimation of mutual information for strongly dependent variables, AISTATS (2015) 277–286.
[9] G. Gómez-Herrero, W. Wu, K. Rutanen, M.C. Soriano, G. Pipa, R. Vicente, Assessing coupling dynamics from an ensemble of time series, Entropy 17 (2015) 1958–1970.
[10] D.W. Hahs, S.D. Pethel, Transfer entropy for coupled autoregressive processes, Entropy 15 (2013) 767–788, http://dx.doi.org/10.3390/e15030767.
[11] S. Hanov, VP Trees: A Data Structure for Finding Stuff Fast, 2012 http://stevehanov.ca/blog/index.php?id=130.
[12] M. Harré, T. Bossomaier, Phase-transition – behaviour of information measures in financial markets, Europhys. Lett. 87 (2009) 18009.
[13] P. Hertz, Über den gegenseitigen durchschnittlichen abstand von punkten, die mit bekannter mittlerer dichte im raume angeordnet sind, Math. Ann. 67 (1909) 387–398.
[14] V. Hnizdo, J. Tan, B.J. Killian, M.K. Gilson, Efficient calculation of configurational entropy from molecular simulations by combining the mutual-information expansion and nearest-neighbor methods, J. Comput. Chem. 29 (2008) 1605–1614.
[15] E. Ising, Beitrag zur theorie des ferromagnetismus, Z. Phys. 31 (1925) 253–258.
[16] L. Kozachenko, N.N. Leonenko, Sample estimate of the entropy of a random vector, Probl. Pered. Inf. 23 (1987) 9–16.
[17] A. Kraskov, H. Stögbauer, P. Grassberger, Estimating mutual information, Phys. Rev. E 69 (2004) 066138–066153, http://dx.doi.org/10.1103/PhysRevE.69.066138.
[18] D.-T. Lee, On k-nearest neighbor Voronoi diagrams in the plane, IEEE Trans. Comput. 100 (1982) 478–487.
[19] J.T. Lizier, F.M. Atay, J. Jost, Information storage, loop motifs, and clustered structure in complex networks, Phys. Rev. E 86 (2012) 026110+, http://dx.doi.org/10.1103/physreve.86.026110.
[20] J.T. Lizier, M. Prokopenko, A.Y. Zomaya, Local information transfer as a spatiotemporal filter for complex systems, Phys. Rev. E 77 (2008) 026110+.
[21] H. Matsuda, K. Kudo, R. Nakamura, O. Yamakawa, T. Murata, Mutual information of Ising systems, Int. J. Theor. Phys. 35 (1996) 839–845.
[22] D.M. Mount, S. Arya, ANN: A Library for Approximate Nearest Neighbor Searching, 2010.
[23] B.C. Ross, Mutual information between discrete and continuous data sets, PLOS ONE 9 (2014) e87357.
[24] T. Schreiber, Measuring information transfer, Phys. Rev. Lett. 85 (2000) 461–464.
[25] C. Shannon, A mathematical theory of communication, Bell Syst. Tech. J. 27 (379–423) (1948) 623–656.
[26] J.K. Uhlmann, Satisfying general proximity/similarity queries with metric trees, Inf. Process. Lett. 40 (1991) 175–179, http://dx.doi.org/10.1016/0020-0190(91)90074-R.
[27] P. Varilly, Periodic KD Tree, 2014 https://github.com/patvarilly/periodic_kdtree.
[28] T. Vicsek, A. Czirók, E. Ben-Jacob, I. Cohen, O. Shochet, Novel type of phase transition in a system of self-driven particles, Phys. Rev. Lett. 75 (1995) 1226–1229, http://dx.doi.org/10.1103/PhysRevLett.75.1226.
[29] X.R. Wang, J.M. Miller, J.T. Lizier, M. Prokopenko, Rossi R L.F, Quantifying and tracing information cascades in swarms, PLoS ONE 7 (2012) e40084+, http://dx.doi.org/10.1371/journal.pone.0040084.
[30] R.T. Wicks, S.C. Chapman, R. Dendy, Mutual information as a tool for identifying phase transitions in dynamical complex systems with limited data, Phys. Rev. E 75 (2007), http://dx.doi.org/10.1103/PhysRevE.75.051125.
[31] I.H. Witten, E. Frank, Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, 2005.
[32] P.N. Yianilos, Data structures and algorithms for nearest neighbor search in general metric spaces, SODA, vol. 93 (1993) 311–321.
[33] Y.-m. Zhang, K. Huang, G. Geng, C.-l. Liu, Fast kNN graph construction with locality sensitive hashing, in: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Springer, 2013, pp. 660–674.

Joshua Brown received his Bachelor's degree (Hons) in Computer Science (Games Technology) in 2010 from Charles Sturt University, Bathurst, Australia. Currently he is a PhD candidate at Charles Sturt University, Bathurst, Australia. His research interests are computational modelling, optimisation, and information theory.



Terry Bossomaier is professor of computer systems at Charles Sturt University. His research interests range from complex systems to computer games. He has published numerous research articles in journals and conferences and several books, the most recent, on information theory appearing in December 2016.

Lionel Barnett is a member of the Sackler Centre of Consciousness Science at the University of Sussex, United Kingdom. He has a number of publications with particular interest in evolutionary computation and Granger causality. He is also co-author with Terry Bossomaier of a Transfer Entropy book published in 2016.