Measuring Mutual Information Between All Pairs of Variables in Subquadratic Complexity

Mohsen Ferdosi
Carnegie Mellon University

Arash Gholami Davoodi
Carnegie Mellon University

Hosein Mohimani
Carnegie Mellon University

Abstract

Finding associations between pairs of variables in large datasets is crucial for various disciplines. The brute force method for solving this problem requires computing the mutual information between $\binom{N}{2}$ pairs. In this paper, we consider the problem of finding pairs of variables with high mutual information in sub-quadratic complexity. This problem is analogous to the nearest neighbor search, where the goal is to find pairs among N variables that are similar to each other. To solve this problem, we develop a new algorithm for finding associations based on constructing a decision tree that assigns a hash to each variable, in a way that for pairs with higher mutual information, the chance of having the same hash is higher. For any $1 \leq \lambda \leq 2$, we prove that in the case of binary data, we can reduce the number of necessary mutual information computations for finding all pairs satisfying $I(X,Y) > 2-\lambda$ from $O(N^2)$ to $O(N^\lambda)$, where $I(X,Y)$ is the empirical mutual information between variables X and Y. Finally, we confirmed our theory by experiments on simulated and real data. The implementation of our method and experiments is publicly available at https://github.com/mohimanilab/HashMI.

1 Introduction

Improvements in data collection and storage have led to the creation of massive datasets. However, currently there is no fast approach to find strong associations between all variables in these datasets. Finding interesting connections between sets of variables in large datasets is an essential tool in various domains [Reshef et al., 2011], including analyzing gene expression data for discovering interactions between genes based on multiple expression measurements [Friedman et al., 2000], connecting each marker with the disease phenotype in genome-wide association studies (GWAS) [Brinza et al., 2010], and finding associations between metabolomics and metagenomics datasets [Melnik et al., 2017, Cao et al., 2019].

To find dependencies between variables using a brute-force approach, one method is to compute the mutual information for each pair among N variables with a complexity of $O(N^2)$ which is impractical for massive datasets. In this paper, we introduce a new algorithm to find all discrete variable pairs X and Y satisfying $I(X,Y) > 2-\lambda$ with complexity $O(N^\lambda)$, where $I(X,Y)$ is the empirical mutual information computed on the empirical probability distribution of X and Y (See Definition 6). Our algorithm hashes variables in a way that pairs of variables with higher mutual information tend to have the same hash with higher probability. This way, one can avoid computing the mutual information for a large portion of pairs by limiting it to the pairs with the same hashes.

2 Related Work

Chow and Liu [1968] introduced an algorithm for fitting a multivariate distribution with a tree . In this model, they assume there are only pairwise dependencies between the variables, and the graph of these dependencies is a spanning tree. While there are $N^{(N-2)}$ tree structures for N variables, the Chow-Liu algorithm requires $O(N^2)$ computations of mutual information between pairs of variables, and currently is the state of the art method for learning the structure of graphical models [Friedman et al., 1997, Pearl, 2014, Heckerman et al., 1995]. The question we address in this paper is whether it is possible to find high mutual informations between pairs of variables in subquadratic runtime. Meila [1999] introduced an accel-

Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS) 2020, Palermo, Italy. PMLR: Volume 108. Copyright 2020 by the author(s).

erated method for computation of mutual information for sparse data, that avoids explicitly computing mutual information for each pair of data . Qiu et al. [2009] sped up the computation of pairwise mutual information by avoiding repeated calculations. However, this method remains quadratic in N and currently there is no sub-quadratic method in the general case.

Mutual information is not the only measure for the dependency of distributions. Distance correlation is another measure of dependency [Székely et al., 2009], and it is shown that for some distributions, it is more noise robust than the mutual information criteria [Simon and Tibshirani, 2014]. Currently, the runtime of computing distance correlations between N variables grow quadraticly with both N and the dimension. Chirigati et al. [2016] introduced DataPolygamy, a topology-based framework to find statistically significant relationships between spatio-temporal data sets. However this approach involves comparing all the pairs of variables. Boidol and Hapfelmeier [2017] provided an algorithm to estimate mutual information between a pair of data on data streams. Lin et al. [2014] used decision trees to provide supervised hashing for nearest neighbor search. Lin et al. [2010] used entropy to learn a hash function for nearest neighbor search which is not relevant to this paper.

In this paper, we propose a new algorithm based on hashing, that requires a subquadratic number of mutual information computations. The naive method for computing mutual information for each pair of data points has $O(S)$ complexity, where S is the dimension of data points. There are methods to compute mutual information of a pair of data points faster based on sampling instead of going over all the dimensions of each data points [Keller et al., 2015, Vollmer and Böhm, 2019]. Note that these methods are complementary to the approach proposed here, and they can be used to make our method even faster.

3 Definitions

Definition 1 (Alphabets) *Define two alphabets $\mathcal{A}$ and $\mathcal{B}$ as sets of characters:*

$$\begin{aligned} \mathcal{A} &= \{a_1, a_2, \cdots, a_k\} \quad (1) \\ \mathcal{B} &= \{b_1, b_2, \cdots, b_l\} \quad (2) \end{aligned}$$

for some natural numbers k and l. Moreover, define a "data point" as a sequence of S characters of an alphabet, i.e. $X \in \mathcal{A}^S$ or $Y \in \mathcal{B}^S$. Moreover, assume the product probability distribution:

$$\mathbb{P} : \mathcal{A}^S \times \mathcal{B}^S \to [0,1], \quad \mathbb{P}(X,Y) = \prod_{s=1}^{S} p(x_s, y_s) \quad (3)$$

for $X = (x_1, \cdots, x_S) \in \mathcal{A}^S$ and $Y = (y_1, \cdots, y_S) \in \mathcal{B}^S$. Note that, we assume the same probability distribution $p(x,y)$ for each dimension $1 \le s \le S$. Here we use i.i.d assumption to limit the number of parameters, which is widely used [Reshef et al., 2011]. Define: $p(x_s = i, y_s = j) = p_{i,j}$ satisfying:

$$\sum_{i=1}^{k} \sum_{j=1}^{l} p_{i,j} = 1 \quad (4)$$

Definition 2 (Empirical Mutual Information) *The empirical distribution $\hat{P}_{X,Y}$, is defined as a $k \times l$ matrix for a pair of (X,Y):*

$$[\hat{P}_{X,Y}]_{i,j} = \frac{m_{ij}}{S} \quad (5)$$

and $\hat{P}_x$ and $\hat{P}_y$ are the marginals of $\hat{P}_{X,Y}$, where m_i^x, m_j^y, and m_{ij} are defined as follow:

$$\begin{aligned} m_i^x &= |\{1 \le s \le S \mid X_s = a_i\}| \quad (6) \\ m_j^y &= |\{1 \le s \le S \mid Y_s = b_j\}| \quad (7) \\ m_{ij} &= |\{1 \le s \le S \mid X_s = a_i \text{ and } Y_s = b_j\}| \quad (8) \end{aligned}$$

for all $1 \le i \le k$ and $1 \le j \le l$, where X_s is the s'th character of sequence X, and $|V|$ is the size of the set V. Note that, $\sum_{i=1}^{k} \sum_{j=1}^{l} m_{ij}$, $\sum_{i=1}^{k} m_i^x$ and $\sum_{j=1}^{l} m_j^y$ are equal to S. Then, we define empirical entropies and empirical mutual information as

$$\begin{aligned} \mathcal{H}(X) &= \mathcal{H}(\hat{P}_x) \quad (9) \\ \mathcal{H}(Y) &= \mathcal{H}(\hat{P}_y) \quad (10) \\ \mathcal{I}(X,Y) &= \mathcal{I}(\hat{P}_{x,y}) \quad (11) \end{aligned}$$

For further details on the definitions of entropy $\mathcal{H}(p)$, mutual information $\mathcal{I}(p_{x,y})$ and Kullback-Leibler Divergence, i.e., $D_{KL}(p||q) = \mathcal{H}(p,q) - \mathcal{H}(p)$, see Appendix A.

Definition 3 (Bounded Density Distribution) *A random variable[1] has a bounded density distribution if there exists a finite positive constant $f_{\max}$ such that the probability distribution function exists everywhere and is bounded by $f_{\max}$.*

Notation $Dir(\boldsymbol{\alpha})$ stands for a Dirichlet distribution with parameter $\boldsymbol{\alpha}$ and $\beta(.)$ stands for beta function, see Appendix B for details.

4 Problem Statement and Model

Given $\mathcal{X} = \{X_1, X_2, ..., X_N\} \subseteq \mathcal{A}^S$ and $\mathcal{Y} = \{Y_1, Y_2, ..., Y_N\} \subseteq \mathcal{B}^S$, our goal is to design a fast algorithm to discover all pairs of data points (X,Y) with high empirical mutual information $I(X,Y)$ for $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$. Here our goal is to design a method for

[1]In this paper, we refer to variables as data points, which corresponds to a set of S samples from the variables.

finding all data points that have high mutual information in sub-quadratic runtime with respect to N. A similar problem has been investigated in the nearest neighbors search literature [Indyk and Motwani, 1998]. In this problem, we have a set of data points $\{X_1, X_2, ..., X_N\}$ and a data point Y, and the goal is to find all data points X_i that are most similar to Y, i.e. $d(X_i, Y)$ is less that a threshold, where d is a distance metric. Our problem is analogous to this problem with the difference that we attempt to maximize mutual information $I(X_i, Y)$ instead of minimizing a distance metric. We consider the following generative model for data points $X \in \mathcal{A}^S$ and $Y \in \mathcal{B}^S$. We assume (X, Y) are either generated from a joint (dependent) probability distribution $\mathbb{P}^{k\times l}$ with probability p_d, or generated independently from probability distributions ${\mathbb{P}_x}^{k\times 1}$ and ${\mathbb{P}_y}^{l\times 1}$ with probability $1-p_d$. $\mathbb{P}$ itself is generated from Dirichlet prior with parameter $\boldsymbol{\alpha}^{k\times l}$, while $\mathbb{P}_x$ and $\mathbb{P}_y$ are generated independently from Dirichlet priors $\boldsymbol{\alpha}_x^{k\times 1} = \boldsymbol{\alpha}\times\mathbf{1}$ and $\boldsymbol{\alpha}_y^{1\times l} = \mathbf{1}\times\boldsymbol{\alpha}$. Dirichlet prior is commonly used as a prior for probabilistic models [Blei et al., 2003]. Moreover, Dirichlet prior simplifies the computation of the posterior probabilities. Note that, while our model assumes a Dirichlet prior, in Section 5, we prove that presented algorithms also work for non-Dirichlet data (arbitrary prior with bounded density distribution).

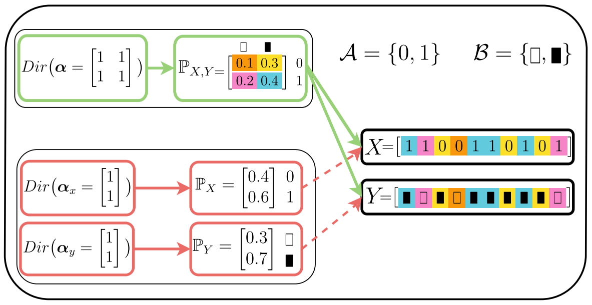
$Dir(\boldsymbol{\alpha} = \begin{bmatrix}1 & 1\\ 1 & 1\end{bmatrix})$
$\mathbb{P}_{X,Y} = \begin{bmatrix}0.1 & 0.3\\ 0.2 & 0.4\end{bmatrix}$ 0 1
$\mathcal{A} = \{0, 1\}$ $\mathcal{B} = \{\square, \blacksquare\}$
$X = [1\ 1\ 0\ 0\ 1\ 1\ 0\ 1\ 0\ 1]$
$Y = [\blacksquare\ \square\ \blacksquare\ \square\ \blacksquare\ \blacksquare\ \blacksquare\ \blacksquare\ \blacksquare\ \square]$
$Dir(\boldsymbol{\alpha}_x = \begin{bmatrix}1\\ 1\end{bmatrix})$
$\mathbb{P}_X = \begin{bmatrix}0.4\\ 0.6\end{bmatrix}$ 0 1
$Dir(\boldsymbol{\alpha}_y = \begin{bmatrix}1\\ 1\end{bmatrix})$
$\mathbb{P}_Y = \begin{bmatrix}0.3\\ 0.7\end{bmatrix}$ $\square$ $\blacksquare$

Figure 1: We assume the data points are coming either from a dependent (with probability p_d) or independent (with probability $1-p_d$) model. We model the dependent distribution with prior $Dir(\boldsymbol{\alpha})$ for generating a joint probability distribution $\mathbb{P}$, from which the data points are sampled. In the independent model, we assume marginal probability distributions $\mathbb{P}_x$ and $\mathbb{P}_y$ are sampled independently from $Dir(\boldsymbol{\alpha}_x)$ and $Dir(\boldsymbol{\alpha}_y)$, and data points X and Y are independently sampled from $\mathbb{P}_x$ and $\mathbb{P}_y$. We further show log-likelihood of Dirichlet distribution for each data point is proportional to their mutual information, While we use Dirichlet model for setting our parameters, in Section 8 we prove the guarantees hold for any distribution.

Now, we pose the following problem. Given X and Y, how can we decide whether the pair(X, Y) is being generated from the dependent model $\boldsymbol{\alpha}$ or the independent models $\boldsymbol{\alpha}^x$ and $\boldsymbol{\alpha}^y$? In order to answer this, a natural approach is to compute the conditional probability of the pair generated through each model:

$$\begin{aligned} A_{X,Y} &= P(X, Y \mid \boldsymbol{\alpha}) = \int P(X, Y \mid r)\, P(r \mid \boldsymbol{\alpha})\, dr \\ &= \int \frac{1}{\beta(\boldsymbol{\alpha})} \prod_{i=1}^{k}\prod_{j=1}^{l} {r_{ij}}^{m_{ij}} {r_{ij}}^{\alpha_{ij}-1} dr_{ij} \qquad (12) \\ &= \frac{\beta(\mathbf{m}+\boldsymbol{\alpha})}{\beta(\boldsymbol{\alpha})} \qquad (13) \\ B_{X,Y} &= P(X, Y \mid \boldsymbol{\alpha}^x, \boldsymbol{\alpha}^y) = \int P(X \mid r^x) P(Y \mid r^y) \\ & \quad P(r^x \mid \boldsymbol{\alpha}^x) P(r^y \mid \boldsymbol{\alpha}^y) dr^x dr^y \qquad (14) \\ &= \frac{\beta(\mathbf{m}^x + \boldsymbol{\alpha}^x)}{\beta(\boldsymbol{\alpha}^x)} \frac{\beta(\mathbf{m}^y + \boldsymbol{\alpha}^y)}{\beta(\boldsymbol{\alpha}^y)} \qquad (15) \end{aligned}$$

where

$$\mathbf{m} = [m_{ij}], \quad \mathbf{m}^x = [m_i^x], \quad \mathbf{m}^y = [m_j^y] \qquad (16)$$

where m_i, m_j, and m_{ij} are defined in (6), (7), and (8). From now on, we assume uniform Dirichlet prior, i.e., $\boldsymbol{\alpha}$ is a matrix of ones, $(\boldsymbol{\alpha} = \mathbf{1})$ and tune all the parameters based on it. Later, we show that for any arbitrary prior with bounded density distribution our algorithm works in sub-quadratic complexity. The posterior probability of model $\boldsymbol{\alpha}$ generating data points X and Y is computed using the Bayesian rule:

$$P(\boldsymbol{\alpha} \mid X, Y) = \frac{p_d A_{X,Y}}{p_d A_{X,Y} + (1-p_d) B_{X,Y}} \qquad (17)$$

Optimal classifier would classify (X, Y) as dependent if $P(\alpha \mid (X, Y))$ exceeds a threshold δ in (17), i.e.,

$$\frac{A_{X,Y}}{B_{X,Y}} > \frac{\delta(1-p_d)}{(1-\delta)p_d} \qquad (18)$$

The following lemma, shows the connection between this log likelihood ratio and mutual information.

Lemma 1 $\log\left(\frac{A_{X,Y}}{B_{X,Y}}\right)$ *is approximately equal to* $I(X;Y)S$. *More precisely, we have*

$$\begin{aligned} &\left| \log\left(\frac{A_{X,Y}}{B_{X,Y}}\right) - I(X;Y)S \right| \\ \leq\ & O(kl!) + |k + l - \frac{1}{2} - kl| \log S \qquad (19) \end{aligned}$$

Proof of Lemma 1 is relegated to Appendix C. A brute force approach to this problem computes $\frac{A_{X,Y}}{B_{X,Y}}$ for each pair of data points X and Y. However, this approach has a runtime of $O(MNS)$, where N and M are the numbers of data points and queries, and S is the dimension of the data points. [2] In order to find dependent pairs more efficiently, we use a hashing strategy

[2] In practice pairwise computation of mutual information can be done faster than $O(S)$ through sampling. Our algorithm can take advantage of these techniques as well.

based on a decision tree structure, to avoid comparing majority of the pairs and restrict the search to pairs that hash to the same indices. Our strategy favors pairs for which $\frac{A_{X,Y}}{B_{X,Y}}$ is large. While using only a single decision tree might result in loss of sensitivity (i.e. many of the dependent pairs might be missed), we show that it is possible to design sub-quadratic algorithms with near-perfect sensitivity (e.g. they capture nearly all the true pairs) by using multiple random permutations of data points, called bands.

5 Structure of the Decision Tree

Consider a decision tree $G(V, E, f)$, where V is the set of nodes, V_l is the set of leaf nodes , E is the set of edges, and $f : V/V_l \times \mathcal{A} \times \mathcal{B} \to V$ are decisions, where $f(v, a, b)$ is a child of v, for each $v \in V$, $a \in \mathcal{A}$ and $b \in \mathcal{B}$. Assume $V_s \subseteq V$ is the set of nodes at depth s in the decision tree. This introduces a natural mapping $S_x : V_s \to \mathcal{A}^s$ and $S_y : V_s \to \mathcal{B}^s$ defined in the following recursive way:

$$S_x(root) = \varnothing \tag{20}$$

$$S_y(root) = \varnothing \tag{21}$$

$$S_x(f(v,a,b)) = [S_x(v), a] \tag{22}$$

$$S_y(f(v,a,b)) = [S_y(v), b] \tag{23}$$

where $[S, a]$ stands for the concatenation of string S with character a. We further define depth(v) as the depth of v in the decision tree, and depth(G) = $\max_v$ depth(v).

Consider a subset $V_{buckets} \subseteq V_l$. Members of $V_{buckets}$ are referred as bucket nodes. Each decision tree G and each permutation $\pi : \{1, ..., S\} \to \{1, ..., S\}$ defines a natural hashing of data points to bucket nodes as follows. Given a data point $X \in \mathcal{A}^S$, X hashes to a bucket node $v \in V_{buckets}$ if and only if $S_x(v)$ is a prefix of $\pi(X)$, where $\pi(X) \in \mathcal{A}^S$ stands for the permutation of data point $X \in \mathcal{A}^S$. Hashing for data points $Y \in \mathcal{B}^S$ is defined similarly. Moreover, we define Prefix(S_1, S_2) as the event where string S_1 is a prefix of string S_2. Here, hashing is not unique, i.e. the same data point might hash to multiple buckets under the same decision tree and permutation, as multiple buckets might have the same string S_x.

6 Searching for Dependent Pairs Using Decision Trees

Given a decision tree $G = (V, E, f)$, along with buckets $V_{buckets}$, Algorithm 1 present an approach for efficiently finding the pairs of data points with high dependency. In this algorithm, we first build a prefix tree for nodes $v \in V_{buckets}$ based on $S_x(v)$, and then given a data point $X \in \mathcal{A}^S$ and a permutation π, we search for bucket nodes v for which $S_x(v)$ is a prefix of $\pi(X)$ using the prefix tree, and insert X into those buckets. Similarly, we insert data points $Y \in \mathcal{B}^S$ into buckets. Finally, we report the pairs of data points X and Y falling into same buckets as positives (see Figure 2).

Algorithm 1 Matching Data Points to Buckets

Input: Decision tree $G = (V, E, f)$, buckets $V_{buckets}$, number of bands b, data points $\mathcal{X} = \{X_1, X_2, ..., X_N\}$, queries $\mathcal{Y} = \{Y_1, Y_2, ..., Y_M\}$

Output: Dependent pairs based on the tree and the buckets

```
procedure MATCH_TO_BUCKETS($V_{buckets}, b, \mathcal{X}, \mathcal{Y}$)
    Build prefix trees $T_1$ based on $v.S^x, v \in V_{buckets}$
    Build prefix trees $T_2$ based on $v.S^y, v \in V_{buckets}$
                                                    ▷ preprocessing
    for $1 \le i \le b$ do
        Select a random permutation $\pi$
        MATCHING($V_{buckets}, \mathcal{X}, \mathcal{Y}, \pi$)
procedure MATCHING($V_{buckets}, \mathcal{X}, \mathcal{Y}, \pi$)
    for $X \in \mathcal{X}$ do
        $m_X \leftarrow \{v \in V_{buckets} \mid \text{Prefix}(v.S_x, \pi(X))\}$
                                            ▷ using prefix tree $T_1$
        for $v \in m_X$ do
            $v.bucket_x$.insert($X$)
    for $Y \in \mathcal{Y}$ do
        $m_Y \leftarrow \{v \in V_{buckets} \mid \text{Prefix}(v.S_y, \pi(Y))\}$
                                            ▷ using prefix tree $T_2$
        for $v \in m_Y$ do
            $v.bucket_y$.insert($Y$)
    for $v \in V_{buckets}$ do hills
        for $X \in v.bucket_x$ do
            for $Y \in v.bucket_y$ do
                Call $(X, Y)$ a positive
```

Remark 1 *As a post processing step, we can filter positive calls using various criteria, e.g, only keeping pairs satisfying $I(X, Y)$ higher that a threshold.*

Note that Algorithm 1 requires a decision tree G, along with a list of buckets $V_{buckets}$. Care should be taken in selection of the tree and buckets in order to maximize the sensitivity and minimize the complexity of the algorithm. In Section 7, we compute sensitivity and complexity of algorithm 1, and in Appendix F we present an algorithm for designing decision trees with low complexity and high sensitivity.

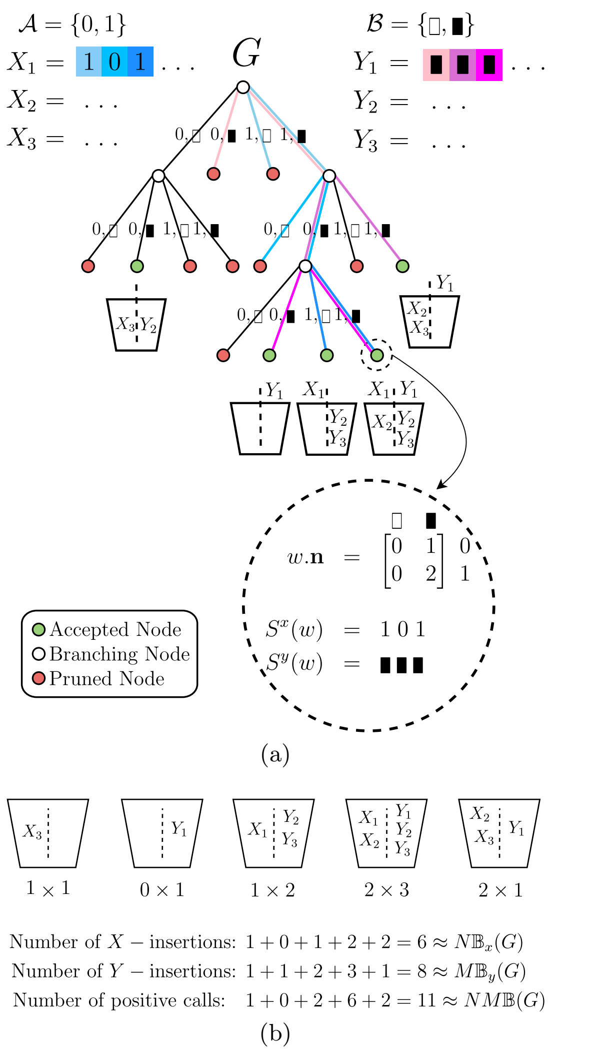
$\mathcal{A} = \{0,1\}$
$\mathcal{B}$
G
$X_1 = 1\ 0\ 1 \ldots$
$X_2 = \ldots$
$X_3 = \ldots$
$Y_1 = \ldots$
$Y_2 = \ldots$
$Y_3 = \ldots$
Accepted Node
Branching Node
Pruned Node
$w.\mathbf{n} = \begin{bmatrix} 0 & 1 \\ 0 & 2 \end{bmatrix} \begin{matrix} 0 \\ 1 \end{matrix}$
$S^x(w) = 1\,0\,1$
$S^y(w)$
(a)
1×1
0×1
1×2
2×3
2×1
Number of $X-$insertions: $1+0+1+2+2 = 6 \approx N\mathbb{B}_x(G)$
Number of $Y-$insertions: $1+1+2+3+1 = 8 \approx M\mathbb{B}_y(G)$
Number of positive calls: $1+0+2+6+2 = 11 \approx NM\mathbb{B}(G)$
(b)

Figure 2: The figure illustrates (a) mapping data points to buckets through the decision trees. The paths for data points $X_1 = 101...$ and $Y_1 =$ ▮▮▮... are shown in blue and purple, respectively. (b) checking data points in each bucket, in algorithm 1. The complexity of the algorithm involves insertion of data points $X \in \mathcal{X}$ (here 6 insertions), insertions of data points $Y \in \mathcal{Y}$ (here 8 insertions), and checking each pair X and Y in each bucket (here 11 checks).

7 Complexity Analysis of Algorithm 1

Complexity of Algorithm 1 consists of three parts:

- Matching data points $X \in \mathcal{X}$ to buckets.
- Matching data points $Y \in \mathcal{Y}$ to buckets.
- Verifying the dependency for each pair.

Lemma 2 describes the complexity and true positive rate of Algorithm 1 for Dirichlet inputs, and Lemma 3 reports the complexity for data coming from arbitrary prior with bounded density distribution.

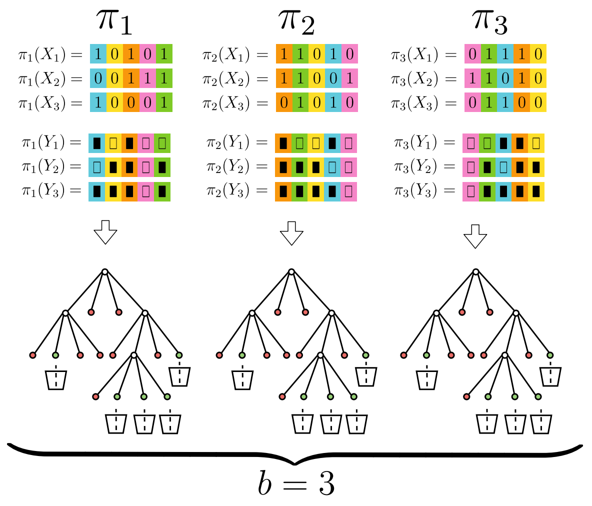
π_1
π_2
π_3
$\pi_1(X_1) = 1\,0\,1\,0\,1$
$\pi_1(X_2) = 0\,0\,1\,1\,1$
$\pi_1(X_3) = 1\,0\,0\,0\,1$
$\pi_2(X_1) = 1\,1\,0\,1\,0$
$\pi_2(X_2) = 1\,1\,0\,0\,1$
$\pi_2(X_3) = 0\,1\,0\,1\,0$
$\pi_3(X_1) = 0\,1\,1\,1\,0$
$\pi_3(X_2) = 1\,1\,0\,1\,0$
$\pi_3(X_3) = 0\,1\,1\,0\,0$
$\pi_1(Y_1) =$
$\pi_1(Y_2) =$
$\pi_1(Y_3) =$
$\pi_2(Y_1) =$
$\pi_2(Y_2) =$
$\pi_2(Y_3) =$
$\pi_3(Y_1) =$
$\pi_3(Y_2) =$
$\pi_3(Y_3) =$
$b = 3$

Figure 3: Running Algorithm 1 for each band usually results in a large number of false negatives. In order to reduce the number of false negatives, we randomly permute elements of data points b times (one per each band). Then, we insert the permuted data points into decision trees, and check all collisions in each band.

Lemma 2 *Under Dirichlet prior, the expected overall complexity of Algorithm 1 for b bands is:*

$$\begin{aligned} E(Complexity_b(G)) \leq\ & c_{prefix}.|V_{buckets}|.depth(G) \\ & + c_{search}.b.(N+M)depth(G) \\ & + c_{insert}.b.\big(N\mathbb{B}_x(G) + M\mathbb{B}_y(G)\big) \\ & + c_{check}.b.N.M.\mathbb{B}(G) \qquad (24) \end{aligned}$$

and the chance of dependent pairs falling in the same bucket in a single band, referred as $TP_1(G)$ is:

$$TP_1(G) = \mathbb{A}(G) \qquad (25)$$

c_{prefix}, c_{search}, c_{insert}, and c_{check} are constants, and $\mathbb{A}(G)$, $\mathbb{B}(G)$, $\mathbb{B}_x(G)$, and $\mathbb{B}_y(G)$ are defined as follows:

$$\mathbb{A}(G) = \sum_{v \in V_{buckets}(G)} A(v) \qquad (26)$$

$$\mathbb{B}(G) = \sum_{v \in V_{buckets}(G)} B(v) \qquad (27)$$

$$\mathbb{B}_x(G) = \sum_{v \in V_{buckets}(G)} B_x(v) \qquad (28)$$

$$\mathbb{B}_y(G) = \sum_{v \in V_{buckets}(G)} B_y(v) \qquad (29)$$

$$A(v) = \frac{\beta(v.\mathbf{n} + \mathbf{1})}{\beta(\mathbf{1})} \qquad (30)$$

$$B_x(v) = \frac{\beta(v.\mathbf{n}^x + \mathbf{1}^x)}{\beta(\mathbf{1}^x)} \qquad (31)$$

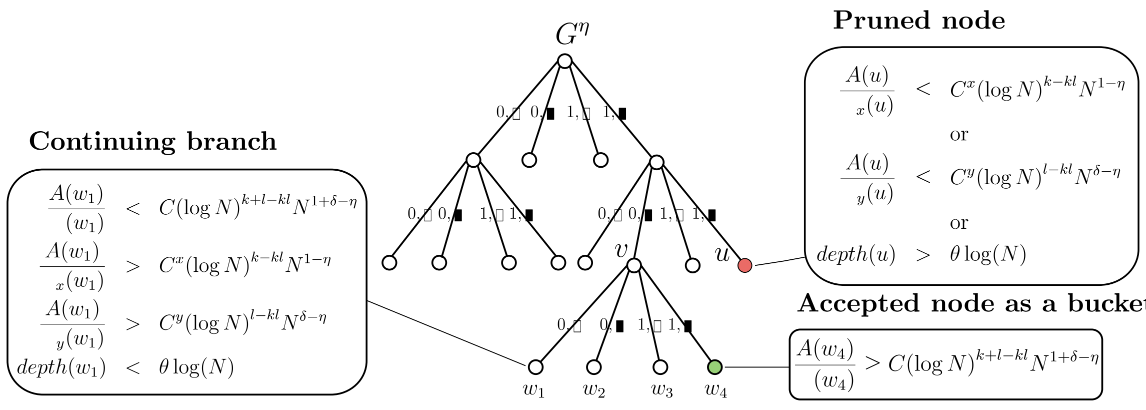
Pruned node
$\frac{A(u)}{{}_x(u)} < C^x(\log N)^{k-kl}N^{1-\eta}$
or
$\frac{A(u)}{{}_y(u)} < C^y(\log N)^{l-kl}N^{\delta-\eta}$
or
$depth(u) > \theta\log(N)$
G^η
Continuing branch
$\frac{A(w_1)}{(w_1)} < C(\log N)^{k+l-kl}N^{1+\delta-\eta}$
$\frac{A(w_1)}{{}_x(w_1)} > C^x(\log N)^{k-kl}N^{1-\eta}$
$\frac{A(w_1)}{{}_y(w_1)} > C^y(\log N)^{l-kl}N^{\delta-\eta}$
$depth(w_1) < \theta\log(N)$
0,□ 0,▮ 1,□ 1,▮
v
u
Accepted node as a bucket
$\frac{A(w_4)}{(w_4)} > C(\log N)^{k+l-kl}N^{1+\delta-\eta}$
w_1
w_2
w_3
w_4

Figure 4: An illustration of the decision tree constructed for $k = l = 2$ and binary alphabets ($\mathcal{A} = \{0, 1\}$ and $\mathcal{B} = \{□, ▮\}$). Examples of an accepted, a pruned, and a branched bucket are shown.

$$B_y(v) = \frac{\beta(v.\mathbf{n}^y + \mathbf{1}^y)}{\beta(\mathbf{1}^y)} \tag{32}$$

$$B(v) = B_x(v).B_y(v) \tag{33}$$

where

$$v.n^x_i = |\ \{1 \le s \le depth(v) \mid S_x(v,s) = a_i\}\ | \tag{34}$$

$$v.n^y_j = |\ \{1 \le s \le depth(v) \mid S_y(v,s) = b_j\}\ | \tag{35}$$

$$v.n_{ij} = |\ \{1 \le s \le depth(v) \mid S_x(v,s) = a_i, S_y(v,s) = b_j\}\ | \tag{36}$$

$$v.\mathbf{n} = [v.n_{ij}], \quad v.\mathbf{n}^x = [v.n^x_i], \quad v.\mathbf{n}^y = [v.n^y_j] \tag{37}$$

and $S_x(v,s)$ and $S_y(v,s)$ are the s'th characters in $S_x(v)$ and $S_y(v)$ respectively, for $1 \le s \le depth(v)$. Note that, $\sum_{i=1}^k \sum_{j=1}^l v.n_{ij} = \sum_{i=1}^k v.n^x_i = \sum_{j=1}^l v.n^y_j = depth(v)$.

Lemma 3 *For any arbitrary priors with bounded density distribution, the expected overall complexity of Algorithm 1 for b bands is bounded by*

$$\begin{aligned} &E(Complexity_b(G)) \\ \le &c_{prefix}|V_{buckets}|depth(G) \\ &+ f_{\max}c_{search}.b.(N+M)depth(G) \\ &+ f_{\max}c_{insert}.b.\big(N\mathbb{B}_x(G) + M\mathbb{B}_y(G)\big) \\ &+ f^2_{\max}c_{check}.b.N.M.\mathbb{B}(G) \end{aligned} \tag{38}$$

Proof of Lemmas 2 and 3 are relegated to Appendices D and E, respectively.

Now, the question is how we should design decision trees in a way that the complexity is minimized while the true positive rate is high, e.g., nearly 99%. Assuming $M = N^\delta$, Algorithm 2 provides an approach for designing decision trees, and in the next section we prove these decision trees are highly sensitive and have low complexity. Starting from the root, at each step, Algorithm 2 either accepts a node as a bucket, prunes a node, or branches a node into kl children based on specific constraints. The intuition behind these constraints are described in Appendix F.

Algorithm 2 Constructing the Buckets

```
Input:   $\eta, N, M, \theta$
Output:  Decision tree, $V_{buckets}$
procedure CONSTRUCT_BUCKETS($\eta, N, M, \theta$)
    Make a new node $root$
    $root.\mathbf{n} = [0]^{k\times l}$, $root.S^x = root.S^y = \varnothing$      ▷ (20)
    CONSTRUCT_TREE ($root, \eta, \frac{\log(N)}{\log(M)}, \theta$)
    return $root, V_{buckets}$
procedure CONSTRUCT_TREE($v, \eta, \delta, \theta$)
    for any $1 \le i \le k$ do
        for any $1 \le j \le l$ do
            Make a new node $w$
            $w.\mathbf{n} = v.\mathbf{n}$, $w.S^x = v.S^x$ and $w.S^y = v.S^y$
            $f(v, a_i, b_i) \leftarrow w$      ▷ Make $w$ child of $v$
            $w.n_{ij} \leftarrow w.n_{ij} + 1$
            $A_w \leftarrow \frac{\beta(w.\mathbf{n}+\mathbf{1})}{\beta(\mathbf{1})}$      ▷ See (29)
            $B^x_w \leftarrow \frac{\beta(w.\mathbf{n}^x+\mathbf{1}^x)}{\beta(\mathbf{1}^x)}$      ▷ See (31)
            $B^y_w \leftarrow \frac{\beta(w.\mathbf{n}^y+\mathbf{1}^y)}{\beta(\mathbf{1}^y)}$      ▷ See (32)
            $B_w \leftarrow B^x_w.B^y_w$      ▷ See (33)
            $w.S_x.append(a_i)$      ▷ See (20)
            $w.S_y.append(b_j)$      ▷ See (21)
            if $\frac{A_w}{B_w} \ge C(\log N)^{k+l-2kl}N^{1+\delta-\eta}$ then
                $V_{buckets}$.insert($w$)      ▷ See (94)
            else if $\frac{A_w}{B^x_w} \le C^x(\log N)^{k-kl}N^{1-\eta}$
                or $\frac{A_w}{B^y_w} \le C^y(\log N)^{l-kl}N^{\delta-\eta}$
                or $depth(w) > \theta\log(N)$ then
                Prune $w$      ▷ See (94)
            else
                CONSTRUCT_TREE($w, \eta, \delta, \theta$)
```

8 Theoretical Guarantees

In this section, we first provide theoretical guarantees on runtime and true positive rate of Algorithm 1 and 2 in the case of single band ($b = 1$). We then present a new algorithm for achieving nearly perfect true positive rate using multiple bands, and provide theoretical guarantees for it.

Definition 4 *Pairs of data points X and Y are called λ-associative if the following holds:*

$$\lambda(X,Y) = \min_{0 \leq \eta} \epsilon(X,Y,\eta) + \eta \leq \lambda \tag{39}$$

where

$$\epsilon(X,Y,\eta) = \min_{n_{ij}} \quad \epsilon \tag{40}$$

$$\begin{cases} n.D_{KL}(\frac{n_{ij}}{n} || \frac{m_{ij}}{S}) \\ +(S-n)D_{KL}(\frac{m_{ij}-n_{ij}}{S-n} || \frac{m_{ij}}{S}) \geq -\epsilon \log N \\ -\mathcal{H}(\frac{n_{ij}}{n}) + \mathcal{H}(\frac{n_i}{n}) + \mathcal{H}(\frac{n_j}{n}) \geq \frac{(1+\delta-\eta)\log N}{n} \\ -\mathcal{H}(\frac{n_{ij}}{n}) + \mathcal{H}(\frac{n_i}{n}) \geq \frac{(1-\eta)\log N}{n} \\ -\mathcal{H}(\frac{n_{ij}}{n}) + \mathcal{H}(\frac{n_j}{n}) \geq \frac{(\delta-\eta)\log N}{n} \\ n \leq \theta \log(N) \\ \sum n_{ij} = n \end{cases}$$

where m_{ij} is the frequency matrix as defined in (8). The optimization is done on η and n_{ij}. If no η satisfies (40), we define $\epsilon(X,Y,\eta) = \infty$.

Remark 2 *The intuition behind this definition is that in our decision tree, a node is pruned if $\frac{A(v)}{B_x(v)}$ and $\frac{A(v)}{B_y(v)}$ are small and accepted if $\frac{A(v)}{B(v)}$, is large. On the other hand, A, B_x, and B_y are beta functions. Therefore, using sterling inequality, $\frac{A}{B_x}$ is equivalent to $\exp(\mathcal{H}(\frac{n_{ij}}{n}) - \mathcal{H}(\frac{n_i}{n}))$. Similar equivalences hold for $\frac{A}{B_y}$ and $\frac{A}{B}$. Taking logarithms from both sides results in conditions of the definition.*

The following lemma provides lower and upper bounds on λ.

Lemma 4 $\max(1,\delta) \leq \lambda(X,Y) \leq 1+\delta$

Remark 3 *Intuitively, X and Y have a higher association with each other if they are λ-associated with a lower λ. Later, we show that λ-associated pairs can be discovered with complexity $O(N^{\lambda})$.*

Remark 4 *Note that λ and ϵ depend on θ and δ, and we should show them as $\lambda_{\theta,\delta}$ and $\epsilon_{\theta,\delta}$. However, we drop θ and δ subscripts, and in the rest of the paper we assume θ and δ are constants.*

The following theorem provides lower-bound guarantees on sensitivity of Algorithm 1, where tree and buckets are constructed using Algorithm 2.

Theorem 1 *Consider a tree G^{η} constructed by Algorithm 2 with parameter η. Then any pair (X,Y) can be discovered by algorithm 1 (using decision tree G^{η}) with probability at least $C_0(\log N)^{\frac{kl-1}{2}} N^{-\epsilon(X,Y,\eta)}$ and the time complexity is $O(N^{\eta}\log(N)^2)$.*

Proof of Theorem 1 is relegated to Appendix H.1. Theorem 1 above provides lower bounds on true-positive rate of Algorithm 1 and Algorithm 2 in the case when a single band is used ($b = 1$). Currently the relationship between λ-associativity and mutual information is not clear. The following theorem provides guarantees on finding pairs with high mutual information using Algorithm 3 in Appendix I.

Here, we present Algorithm 3, which is based on Algorithm 1 and 2, and given a value $\lambda > 0$ it finds (nearly) all λ-associated pairs with complexity $O(N^{\lambda}\log(N)^2)$. The algorithm is based on constructing decision trees with parameter η for various values of η, $0 < \eta < \lambda$. Then multiple bands from each tree is recruited, in a way that all λ-associated pairs are discovered, while the complexity remains $O(N^{\lambda}\log(N)^2)$

Definition 5 *For $\mathbf{H} = \{\eta_0, ..., \eta_z\}$, (X,Y) is called $(\lambda, \mathbf{H})$-associated if*

$$\min_{\eta \in \mathbf{H}} \epsilon(X,Y,\eta) + \eta \leq \lambda \tag{41}$$

Theorem 2 *Consider $M = N(\delta = 1)$, $\mathcal{A} = \mathcal{B}$, and $\mathcal{X} = \mathcal{Y}$. Then for any pairs of data points satisfying $I(X,Y) \geq \log(k)(2-\lambda)$, Algorithm 3 in Appendix I with parameters λ, $z = 1$, $\mathbf{H} = \{\lambda\}$, and $\theta = 1$ can discover (X,Y) with probability nearly one. The complexity for this case is $O(N^{\lambda}\log(N)^2)$*

For proof of Theorem 2, see Appendix J.2.

Remark 5 *When $\mathcal{A} = \mathcal{B} = \{0,1\}$, all pairs with $I(X;Y) \geq 2-\lambda$ can be discovered in $O(N^{\lambda}\log(N)^2)$.*

In the next section, we experimentally evaluate the theoretical guarantees provided in this section.

9 Experiments

In this section, we run algorithm 1, 2 and 3 in Appendix I on various simulated datasets, and compare the results to guarantees given in Theorems 1 and 2. **Experiment 1.** We computed the tree G^{η} for $\eta = 1.1$, 1.3, and 1.5 and $N = 1000$ using Algorithm 2. Then we sampled 1000 probability distribution $\mathbb{P}$ from $Dir(\boldsymbol{\alpha})$

where $\boldsymbol{\alpha} = \mathbf{1}$, and for each probability distribution $\mathbb{P}$, we sampled a pair of 1000×1 data points X and Y. For each tree G^η and each pair of data points X and Y, we computed empirical true positive rate by selecting random permutations $\pi : \{1, ..., S\} \to \{1, ..., S\}$, and computing the ratio of permutation for which data points X and Y map to the same bucket using Algorithm 1 ($b = 1$). The expected true positive rates is:

$$TP_1^{X,Y}(G^\eta) = \sum_{v \in V_{buckets}(G)} P(\text{Prefix}(S_x(v), \pi(X)), \text{Prefix}(S_y(v), \pi(Y))) \quad (42)$$

$$= \sum_{v \in V_{buckets(G^\eta)}} \frac{\prod_{i=1}^{k} \prod_{j=1}^{l} \frac{(m_{ij})!}{(m_{ij} - v.n_{ij})!}}{\frac{S!}{(S - v.n)!}} \quad (43)$$

where probability is computed over random permutation π. We also computed the lower-bounds on the true positive rates predicted by Theorem 1. Figure 5 illustrates empirical, expected and theoretical lower-bounds on true positive rates as a function of $\epsilon(X, Y, \eta)$ for $\eta = 1.3$. The figure shows that empirical and expected true positive rates are very similar. Moreover these rates are higher than the lower-bounds predicted by Theorem 1, as the bounds used for deriving Theorem 1 are not tight.

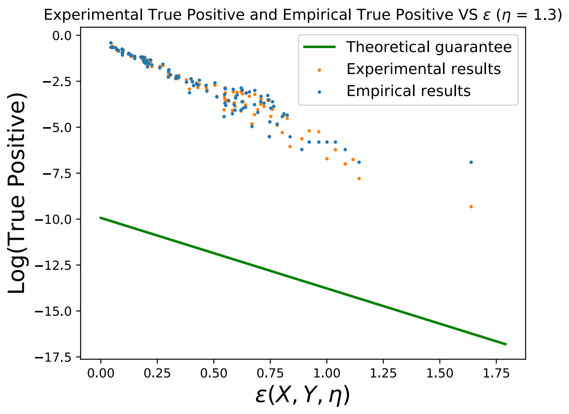

Figure 5: Experimental, empirical and theoretical lower-bounds on true positives rates $TP_1^{X,Y}(G^\eta)$ for various data points $(X, Y) \sim P$, $P \sim Dir(\boldsymbol{\alpha})$ for $\eta = 1.3$. Here, G^η is computed using Algorithm 2, and true positives are calculated using algorithm 1 where $b = 1$. Similar data for $\eta = 1.1$ and $\eta = 1.5$ is shown in Figure 7 in Appendix K.1.

Experiment 2. In this experiment, we use Algorithm 3 to find all pairs with mutual information above 0.5. To this end, we use parameter $\lambda = 2 - 0.5 = 1.5$, and use the brute force method and Algorithm 3 for finding associated pairs. We repeat this experiment for $M = N = 1000, 3000, 10000, 100000$. The following figure shows the runtime of Algorithm 3 versus brute force for each N. Algorithm 3 is capable of finding about 95% of all true positives for $N = 1000, 3000, 10000, 100000$, respectively.

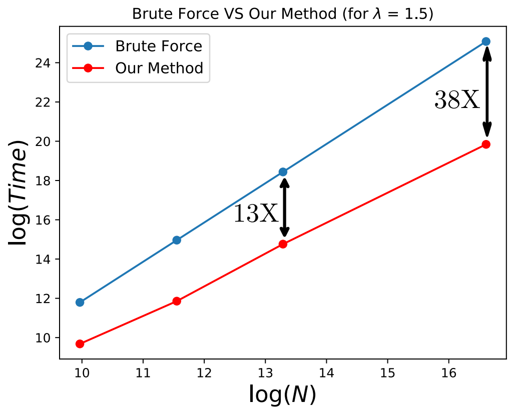

Figure 6: Comparing the runtime of our method with brute force, for various values of N. Our method reaches 95% true positive rate in each case and is ~ 40 times faster than brute force, when $N = 100K$.

Experiment 3. In this experiment, we used our method on a single cell dataset [Alavi et al., 2018] with 8424 genes and 1000 cells to find pairs of genes (G_i, G_j) with mutual information higher than $I(G_1, G_2) > 0.8$. This is crucial step for reconstructing the gene regulatory network from the data. We compared our method to brute force, and the result is as follows: Our method

	Number of Pairs	Time(s)
Brute Force	154	25200
Our Method	153	99

results in $250X$ speedup compared to brute force while maintaining similar sensitivity.

10 Conclusion

In this paper, we propose a new method for computing mutual information between all pairs of data points in a large database. Our method is based on hashing data points to leaf nodes of decision trees, in a way that data points with higher mutual information are mapped to the same bucket with higher probability. We use Dirichlet prior assumption for setting parameters. However, our theoretical results show these parameters work for any bounded density distribution.

11 Acknowledgement

This work was supported by a research fellowship from the Alfred P. Sloan Foundation and a National Institutes of Health New Innovator Award (DP2GM137413).

References

A. Alavi, M. Ruffalo, A. Parvangada, Z. Huang, and Z. Bar-Joseph. A web server for comparative analysis of single-cell rna-seq data. *Nature communications*, 9 (1):4768, 2018.

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

J. Boidol and A. Hapfelmeier. Fast mutual information computation for dependency-monitoring on data streams. pages 830–835, 2017.

D. Brinza, M. Schultz, G. Tesler, and V. Bafna. Rapid detection of gene–gene interactions in genome-wide association studies. *Bioinformatics*, 26(22): 2856–2862, 2010.

L. Cao, E. Shcherbin, and H. Mohimani. A metabolome- and metagenome-wide association network reveals microbial natural products and microbial biotransformation products from the human microbiota. *mSystems*, 4(4), 2019. doi: 10.1128/mSystems.00387-19. URL https://msystems.asm.org/content/4/4/e00387-19.

F. Chirigati, H. Doraiswamy, T. Damoulas, and J. Freire. Data polygamy: the many-many relationships among urban spatio-temporal data sets. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1011–1025. ACM, 2016.

C. Chow and C. Liu. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3):462–467, 1968.

N. Friedman, D. Geiger, and M. Goldszmidt. Bayesian network classifiers. *Machine learning*, 29 (2-3):131–163, 1997.

N. Friedman, M. Linial, I. Nachman, and D. Pe'er. Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620, 2000.

D. Heckerman, D. Geiger, and D. M. Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 20 (3):197–243, 1995.

P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

F. Keller, E. Müller, and K. Böhm. Estimating mutual information on data streams. In *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*, page 3. ACM, 2015.

G. Lin, C. Shen, Q. Shi, A. Van den Hengel, and D. Suter. Fast supervised hashing with decision trees for high-dimensional data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1963–1970, 2014.

R.-S. Lin, D. A. Ross, and J. Yagnik. Spec hashing: Similarity preserving algorithm for entropy-based coding. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 848–854. IEEE, 2010.

M. Meila. An accelerated chow and liu algorithm: fitting tree distributions to high dimensional sparse data. 1999.

A. V. Melnik, R. R. da Silva, E. R. Hyde, A. A. Aksenov, F. Vargas, A. Bouslimani, I. Protsyuk, A. K. Jarmusch, A. Tripathi, T. Alexandrov, et al. Coupling targeted and untargeted mass spectrometry for metabolome-microbiome-wide association studies of human fecal samples. *Analytical chemistry*, 89(14): 7549–7559, 2017.

J. Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Elsevier, 2014.

P. Qiu, A. J. Gentles, and S. K. Plevritis. Fast calculation of pairwise mutual information for gene regulatory network reconstruction. *Computer methods and programs in biomedicine*, 94(2):177–180, 2009.

D. N. Reshef, Y. A. Reshef, H. K. Finucane, Grossman, M. S. R., P. J. G., Turnbaugh, and Sabeti. Detecting novel associations in large data setss. *Science*, 334(6062):1518–1524, 2011.

N. Simon and R. Tibshirani. Comment on" detecting novel associations in large data sets" by reshef et al, science dec 16, 2011. *arXiv preprint arXiv:1401.7645*, 2014.

G. J. Székely, M. L. Rizzo, et al. Brownian distance covariance. *The annals of applied statistics*, 3(4): 1236–1265, 2009.

M. Vollmer and K. Böhm. Iterative estimation of mutual information with error bounds. In *EDBT*, pages 73–84, 2019.