EasyChair Preprint

№ 1500

Forest-wise DSH: A Universal Hash Design for Discrete Probability Distributions

Arash Gholamidavoodi, Sean Chang, Hyun Gon Yoo, Mihir Mongia and Hosein Mohimani

September 12, 2019

Forest-wise DSH: A Universal Hash Design for Discrete Probability Distributions

Abstract

In this paper, we consider the problem of classification of M high dimensional queries $y^1, \cdots, y^M \in \mathcal{B}^S$ to N high dimensional classes $x^1, \cdots, x^N \in \mathcal{A}^S$ where $\mathcal{A}$ and $\mathcal{B}$ are discrete alphabets and the probabilistic model that relates data to the classes $\mathbb{P}(x, y)$ is known. This problem has applications in various fields including the database search problem in mass spectrometry. The problem is analogous to the nearest neighbor search problem, where the goal is to find the data point in a database that is the most similar to a query point. The state of the art method for solving an approximate version of the nearest neighbor search problem in high dimensions is locality sensitive hashing (LSH). LSH is based on designing hash functions that map near points to the same buckets with a probability higher than random (far) points. To solve our high dimensional classification problem, we introduce distribution sensitive hashes that map jointly generated pairs $(x, y) \sim \mathbb{P}$ to the same bucket with probability higher than random pairs $x \sim \mathbb{P}^x$ and $y \sim \mathbb{P}^y$, where $\mathbb{P}^x$ and $\mathbb{P}^y$ are the marginal probability distributions of $\mathbb{P}$. We design distribution sensitive hashes using a forest of decision trees and we show that the complexity of search grows with $O(N^{\lambda^*(\mathbb{P})})$ where $\lambda^*(\mathbb{P})$ is expressed in an analytical form. We further show that the proposed hashes perform faster than LSH-hamming and Minhash for various probability distributions, in both theory and simulations. Finally, we apply our method to the spectral library search problem in mass spectrometry, and show that it is an order of magnitude faster than the state of the art methods.

1 Introduction

Consider the problem of classifying a large number of high dimensional data $Y = \{y^1, \cdots, y^M\} \subset \mathcal{B}^S$ into high dimensional classes $X = \{x^1, \cdots, x^N\} \subset \mathcal{A}^S$, given a known joint probability distribution $\mathbb{P}(x, y)$, where $\mathcal{A}$ and $\mathcal{B}$ are discrete alphabets. Given a point $y \in Y$, the goal is to find the class $x \in X$ that maximize $\mathbb{P}(y \mid x)$;

$$Arg \max_{x \in X} \mathbb{P}(y \mid x) \tag{1}$$

where $\mathbb{P}(y \mid x)$ is factorizable to i.i.d components, i.e.,

$$\mathbb{P}(y \mid x) = \prod_{s=1}^{S} p(y_s \mid x_s) \tag{2}$$

In this paper, we refer to $X = \{x^1, \cdots, x^N\}$ as data base points and $Y = \{y^1, \cdots, y^M\}$ as queries. This problem has application in various fields, including the clustering of spectra generated by mass spectrometry instruments [1]. Consider the problem where there are billions of data points (mass spectra) and given a query spectrum y the goal is to find the spectrum x that maximize known

probability distribution $\mathbb{P}(y \mid x)$ [2, 3]. A naive approach to solve this problem is to compute $\mathbb{P}(y \mid x)$ for each $x \in X$, and find the maximum. The run time for this algorithm is $O(NS)$ which is very slow when the number of classes, N, is large.

In order to address this problem where the number of classes is massive, muticlass classification methods have been established [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16]. However, these methods fail to function efficiently for high dimensional data due to the curse of dimensionality, i.e., in the high dimensional setting, the number of $\mathbb{P}(y \mid x)$ computations required grows with the order of classes, $O(N)$, per query data point.

A similar problem has been investigated in the field of nearest neighbor search. In this problem, given a set of points in a database, the goal is to find the point in the database that is closest to a query point. A popular approach to this problem is locality sensitive hashing [17, 18]. This approach solves ϵ-approximate nearest neighbor search problem by designing a family of hashes in a way that near points are hashed to the same bucket with probability much higher than random points. In ϵ-approximate nearest neighbor search problem, given a query point y, the goal is to find $x \in X$ for which $d(x, y) \leq (1 + \epsilon)d(x', y)$ for all $x' \in X$ and X is the set of all feasible points [17, 19]. Currently, the locality sensitive hashing approach does not generalize to the cases where the triangle inequality, i.e., $d(x, y) \leq d(x, z) + d(z, y)$ does not hold [20, 17, 18, 21, 22, 23, 24, 25]. Recently, high dimensional approximate nearest neighbors in a known probabilistic distribution setting have been investigated [19, 26]. However, currently it is not possible to design efficient algorithms based on these methods, due to the large number of parameters involved.

The problem of finding high dimensional approximate nearest neighbors in a known probabilistic setting using a bucketing tree algorithm has been studied previously in [26, 28]. Dubiner uses a strategy to hash the data points from an arbitrary joint probability distribution into the leafs of the tree in a way that the paired data collide with a probability higher than the random pairs. However, the algorithm introduced in Dubiner requires solving computationally intractable optimizations, making it impossible to implement (e.g. see equation (126) from [26]). However, in this paper, for the specific case where $\mathbb{P}(p) = \begin{bmatrix} \frac{p}{2} & \frac{1-p}{2} \\ \frac{1-p}{2} & \frac{p}{2} \end{bmatrix}$, $0.5 \leq p \leq 1$, our theoretical and practical results are compared to [26]).

In this paper, we propose to solve (1) by defining a family of distribution sensitive hashes satisfying the following property. They hash the jointly-generated pairs of points to the same buckets with probabilities much higher than random pairs. We further design an algorithm to solve (1) in sub-linear time using these families of hashes, and present a method to find optimal family of hashes to achieve minimum search complexity using multiple decision trees where these decision trees have the same tree structure and defer from each other in that they apply to different permutations $\{1, 2, \cdots, S\} \to \{1, 2, \cdots, S\}$ of the data. This way, we design forest of decision trees where each decision tree captures a very small ratio α of true pairs for some $\alpha \in \mathbb{R}^+$ and by recruiting $b = O(\frac{1}{\alpha})$ independently permuted decision trees we can reach near perfect recovery of all the true pairs. In this paper, we refer to each decision tree as a band and b is referred to as the number of bands.

The main idea is that we construct the decision-tree hashes recursively, in a way that the odds of reaching leaf nodes under the true joint probability distribution $\mathbb{P}(x, y)$ is higher than an independent probability distribution $\mathbb{Q}(x, y) = \mathbb{P}^x(x)\mathbb{P}^y(y)$. The decision tree is built in a way that the ratio $\frac{\mathbb{P}(x,y)}{\mathbb{Q}(x,y)}$ is higher than a minimum threshold, while the ratios $\frac{\mathbb{P}(x,y)}{\mathbb{P}^x(x)}$ and $\frac{\mathbb{P}(x,y)}{\mathbb{P}^y(y)}$ and the number of nodes in the graph are lower than a maximum thresholds, see Algorithm 3. We further determined the optimal tree among many trees that can be constructed in this way. Two theorems are presented here on the complexity of the decision tree built in Algorithm 3.

1. No decision tree exists with the overall search complexity below $O(N^{\lambda^*})$ where λ^* is derived analytically from the probability distribution $\mathbb{P}(X, Y)$.

2. The decision tree construction of Algorithm 3 described in (33) results in the overall search with complexity $O(N^{\lambda^*})$.

Our results show that our approach, Forest-wise distribution sensitive hashing (DSH), provides a universal hash design for arbitrary discrete joint probability distributions, outperforming the existing approaches LSH-Hamming and Minhash in theory and practice. Moreover, we applied this method to the problem of clustering spectra generated from mass spectrometry instruments.

An alternative strategy for solving (2) is to reformulate the problem as minimum inner product search problem (MIPS) [27] by transferring the data points into a new space. However, as we show in Section 6 the transferred data points are nearly orthogonal to each other making it very slow to find maximum inner product using the existing method [27].

Note that, the algorithms presented in this paper are based on the assumption that the true pairs are generated from a known distribution $\mathbb{P}$. The distribution $\mathbb{P}$ can be learned from a training dataset of true pairs. In practice, training data can be collected by running a brute force search on smaller data sets or portion of the whole data. For example, in case of mass spectrometry search, we collect training data by running brute-force search on small portion of our data [1].

Notation: The cardinality of a set A is denoted as $|A|$. The sets $\mathbb{N}$ and $\mathbb{R}$ stand for the sets of natural and real numbers, respectively. We use $\mathbb{P}(\cdot)$ and $\mathbb{Q}(\cdot)$ to denote the probability function Prob$(\cdot)$. Moreover, we use the notation $f(x) = O(g(x))$, if $\limsup_{x\to\infty} \frac{|f(x)|}{g(x)} < \infty$ and the notation $f(x) = \Omega(g(x))$, if $\limsup_{x\to\infty} \frac{|f(x)|}{g(x)} > 0$. In this paper, $\log x$ is computed to the base e.

2 Definitions

Definition 1 *Define the S-dimensional joint probability distribution $\mathbb{P}$ as follows.*

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

where $\mathcal{A} = \{a_1, a_2, \cdots, a_k\}$, $\mathcal{B} = \{b_1, b_2, \cdots, b_l\}$, $x = (x_1, x_2, \cdots, x_S) \in \mathcal{A}^S$ and $y = (y_1, y_2, \cdots, y_S) \in \mathcal{B}^S$ where $k, l < \infty$. On the other hand, we assume that the probability distribution function $p(a, b)$ is independent of s and satisfies $\sum_{a\in[k],b\in[l]} p_{a,b} = 1$[1]. Similarly, we define the marginal probability distributions $\mathbb{P}^x : \mathcal{A}^S \to [0, 1]$, $\mathbb{P}^y : \mathcal{B}^S \to [0, 1]$ and $\mathbb{Q} : \mathcal{A}^S \times \mathcal{B}^S \to [0, 1]$ as $\mathbb{P}^x(x) = \prod_{s=1}^{S} p^x(x_s)$, $\mathbb{P}^y(y) = \prod_{s=1}^{S} p^y(y_s)$ and $\mathbb{Q}(x, y) = \prod_{s=1}^{S} q(x_s, y_s)$, where

$$p^x(x_s) = \sum_{j=1}^{l} p(x_s, b_j) \tag{4}$$

$$p^y(y_s) = \sum_{i=1}^{k} p(a_i, y_s) \tag{5}$$

$$q(x_s, y_s) = p^x(x_s) \times p^y(y_s) \tag{6}$$

[1]We use p_{ij} instead of $p(a_i, b_j)$ for simplicity. Moreover, p_i^x, p_j^y and q_{ij} are defined as $\sum_{j=1}^{l} p_{ij}$, $\sum_{i=1}^{k} p_{ij}$ and $q(a_i, b_j)$, respectively.

We define family of distribution sensitive hashes as follows.

Definition 2 (Family of Distribution Sensitive Hashes) *Assume that the two parameters α and β along with the probability distributions $\mathbb{P}$, $\mathbb{P}^x$, $\mathbb{P}^y$ and $\mathbb{Q}$ are given where $\mathbb{P}^x$ and $\mathbb{P}^y$ are marginal distributions of $\mathbb{P}$ and $\mathbb{Q} = \mathbb{P}^x\mathbb{P}^y$. A family of subsets (referred here as buckets) $U^x_{j,v} \subset \mathcal{A}^S$ and $U^y_{j,v} \subset \mathcal{B}^S, 1 \leq j \leq \#bands$ ($\#bands$ stands for the number of bands), $v \in V_{Buckets}$ is called $(\mathbb{P}, \alpha, \beta, \gamma_x, \gamma_y)$-distribution sensitive, if the following hold.*

$$\alpha = \sum_{v \in V_{Buckets}} Prob((x,y) \in U^x_{j,v} \times U^y_{j,v} \mid (x,y) \sim \mathbb{P}) \tag{7}$$

$$\beta = \sum_{v \in V_{Buckets}} Prob((x,y) \in U^x_{j,v} \times U^y_{j,v} \mid (x,y) \sim \mathbb{Q}) \tag{8}$$

$$\gamma_x = \sum_{v \in V_{Buckets}} Prob(x \in U^x_{j,v} \mid x \sim \mathbb{P}^x) \tag{9}$$

$$\gamma_y = \sum_{v \in V_{Buckets}} Prob(y \in U^y_{j,v} \mid y \sim \mathbb{P}^y) \tag{10}$$

$$\forall 1 \leq j \leq \#bands, v \neq v' : \quad U^x_{j,v} \times U^y_{j,v} \cap U^x_{j,v'} \times U^y_{j,v'} = \varnothing \tag{11}$$

Here $\#bands$ represents the number of bands, while $V_{Buckets}$ represent a set of indices for the buckets. We show how to choose #bands in (52) in Appendix C and how to select $V_{buckets}$ in Algorithm 3. Intuitively, α represents the chance of a true pair falling in the same bucket, while β represents the chance of random pairs falling in the same bucket. We will show that γ_x and γ_y represent the complexity of computing which buckets the data points fall into. In the next section, we will describe how the families of distribution sensitive hashes can be used to design an efficient solution for (1).

3 Distribution Sensitive Hashing Algorithm

In this section, we assume an oracle has given us a family of distribution sensitive hashes $U^x_{j,v}, U^y_{j,v}, \forall j : 1 \leq j \leq \#bands, v \in V_{Buckets}$ that satisfies ((7) − (11)). Inspired by LSH method, we present Algorithm 1 for solving (1) using this family.

Algorithm 1 Solving maximum likelihood classification (1) by DSH

Inputs: $X = \{x^1, \cdots, x^N\} \subset \mathcal{A}^S$, $y \in \mathcal{B}^S$ and threshold Δ.
Output: Classes $x \in X$ satisfying $P(y \mid x) > \Delta$.
For $j \in \{1, 2, \cdots, \#bands\}$
  For $v \in V_{Buckets}$
    For $x \in X \cap U^x_{j,v}, y \in Y \cap U^y_{j,v}$
      Call (x, y) a positive, compute $\mathbb{P}(y \mid x)$ and report x if $\mathbb{P}(y \mid x) > \Delta$.

Remark 1 *Note that, the number of times $\mathbb{P}(y \mid x)$ is computed in the brute force method to solve* (1) *is $|X||Y|$. The goal of Algorithm 1 is to solve* (1) *with a much smaller number of comparisons than the brute force.*

Remark 2 *In Appendix C, we show that the number of positive calls in Algorithm 1 is proportional to β, while the complexity of computing $|X \cap U^x_{j,v}|$ and $|Y \cap U^y_{j,v}|$ are proportional to γ_x and γ_y.*

Moreover, the chance of true pairs being called positive grows with α. Therefore, in the next sections, we attempt to design buckets such that α is maximized, while β, γ_x and γ_y are minimized.

Now, the question is how we can design these families in a way to minimize the complexity, and efficiently map data points to these families. We investigate these questions in Sections 4 and 4.1.

4 Designing DSH Using a Decision Tree Structure

In the previous section, we assumed that an oracle has given us a family of distribution sensitive hashes. In this section, we design buckets that satisfy $((7)-(11))$ using a forest of decision tree with the same structure. Here, we focus on the probability distributions that can be factorized as the product of i.i.d components.

Each of our decision trees recovers ratio α of true pairs and by recruiting $b = O(\frac{1}{\alpha})$ decision trees we can recover nearly all true pairs. This can be more efficient than using a single decision tree classifier as achieving near perfect true pair recovery by the single decision tree would require near brute-force complexity. By allowing $\alpha < 1$, we can select decision tree that avoid paths with low $\mathbb{P}(x, y)$ resulting in complexities much lower than the brute-force search.

Assume that a decision tree $G = (V, E, f)$ is given where V is the set of nodes, E is the set of edges, $V_l \subset V$ is the set of leaf nodes in the decision tree and $f : V/V_l \times \mathcal{A} \times \mathcal{B} \to V$ is the decision function. For the two nodes $v_1, v_2 \in V$, v_1 is called an ancestor of v_2 if v_1 falls within the path from v_2 to root. In this case, v_2 is called a descendant of v_1. Furthermore, assume that a subset of leaf nodes $V_{Buckets} \subset V_l$ is given and at depth s in the decision tree, the decisions depend only on x_s and y_s where $x = (x_1, \cdots, x_S)$ and $y = (y_1, \cdots, y_S)$. We define functions $Seq^x : V \to \cup_{s=0}^{S} \mathcal{A}^s$ and $Seq^y : V \to \cup_{s=0}^{S} \mathcal{B}^s$ recursively as:

$$Seq^x(root) \leftarrow \varnothing \quad \& \quad Seq^y(root) \leftarrow \varnothing \tag{12}$$

$$Seq^x(f(v, a, b)) \leftarrow Seq^x(v) \quad \& \quad Seq^y(f(v, a, b)) \leftarrow Seq^y(v) \tag{13}$$

$$Seq^x(f(v, a, b)).append(a) \quad \& \quad Seq^y(f(v, a, b)).append(b) \tag{14}$$

$S.append(a)$ means that letter a is added to the end of the string S, i.e., for a string $S = s_1, \cdots, s_n$ of length n, $S.append(s)$ would be $s_1, \cdots, s_n, s$ which is a string of length $n + 1$. Then, given permutations $perm_j : \{1, \cdots, S\} \to \{1, \cdots, S\}$, $1 \le j \le \#bands$, the family of buckets $U^x_{j,v}$ and $U^y_{j,v}$ are defined as

$$U^x_{j,v} = \{X \in \mathcal{A}^S \mid Seq^x(v) \text{ is a prefix of } \mathrm{perm}_j(X)\} \tag{15}$$

$$U^y_{j,v} = \{Y \in \mathcal{B}^S \mid Seq^y(v) \text{ is a prefix of } \mathrm{perm}_j(Y)\} \tag{16}$$

Now, we show that these buckets are distribution sensitive.

Definition 3 *The functions $A : V \to \mathbb{R}$, $B_x : V \to \mathbb{R}$ and $B_y : V \to \mathbb{R}$ are defined as follows. At root, $A(root) = 1$, $B_x(root) = 1$ and $B_y(root) = 1$, and for $a \in \mathcal{A}$, $b \in \mathcal{B}$ and $v \in V$, $A(v)$, $B_x(v)$, and $B_y(v)$ are defined recursively as*

$$A(f(v, a_i, b_j)) = A(v).p_{ij}, \forall v \in V \tag{17}$$

$$B_x(f(v, a_i, b_j)) = B_x(v).p^x_i, \forall v \in V \tag{18}$$

$$B_y(f(v, a_i, b_j)) = B_y(v).p^y_j, \forall v \in V \tag{19}$$

Moreover, $B : V \to \mathbb{R}$ is defined as

$$B(v) \quad = \quad B_x(v).B_y(v), \forall v \in V \tag{20}$$

In Figure 1, the decision tree is sketched for probability distribution $\mathbb{P} = \begin{bmatrix} 0.4 & 0.3 \\ 0.1 & 0.2 \end{bmatrix}$ and is constructed based on Definition 3. Bold edges and the details of construction of the decision tree are explained in Section 6.

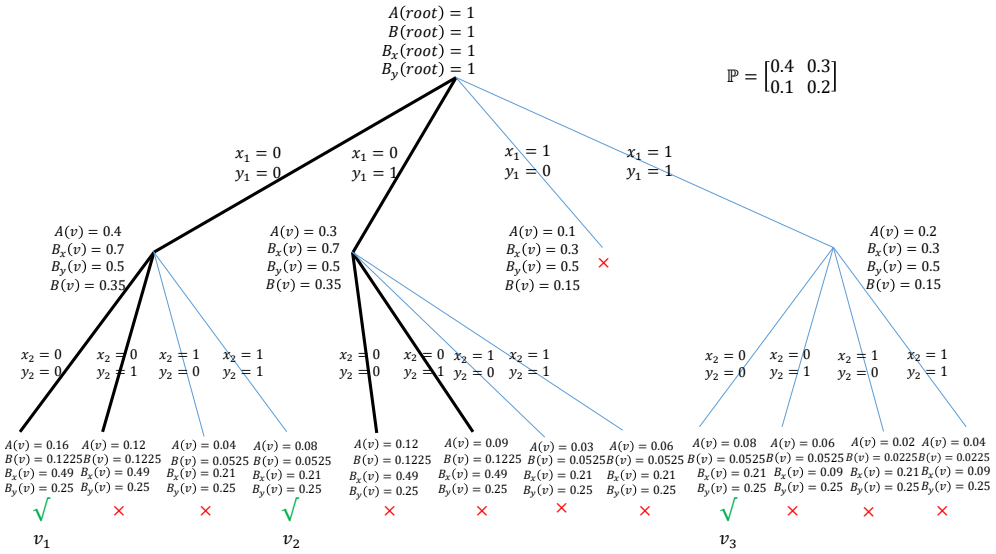


Figure 1: The decision tree and functions $A(v)$, $B_x(v)$, $B_y(v)$ and $B(v)$ are illustrated for $\mathcal{A} = \{0, 1\}, \mathcal{B} = \{0, 1\}$.

Lemma 1 *The following properties hold:*

$$A(v) \quad = \quad Prob((x, y) \in U^x_{j,v} \times U^y_{j,v} \mid (x, y) \sim \mathbb{P}) \tag{21}$$

$$B(v) \quad = \quad Prob((x, y) \in U^x_{j,v} \times U^y_{j,v} \mid (x, y) \sim \mathbb{Q}) \tag{22}$$

$$B_x(v) \quad = \quad Prob(x \in U^x_{j,v} \mid x \sim \mathbb{P}^x) \tag{23}$$

$$B_y(v) \quad = \quad Prob(y \in U^y_{j,v} \mid y \sim \mathbb{P}^y) \tag{24}$$

Lemma 2 *For any decision tree G, satisfying the condition that for any pair of buckets $v_1, v_2 \in V_{Buckets}$, v_1 is not an ancestor or descendant of v_2, $U^x_{j,v}$ and $U^y_{j,v}$ defined in (15) and (16) are $(\mathbb{P}, \alpha(G), \beta(G), \gamma_x(G), \gamma_y(G))$-sensitive where*

$$\alpha(G) \quad = \quad \sum_{v \in V_{Buckets}(G)} A(v) \tag{25}$$

$$\beta(G) \quad = \quad \sum_{v \in V_{Buckets}(G)} B(v) \tag{26}$$

$$\gamma_x(G) \quad = \quad \sum_{v \in V_{Buckets}(G)} B_x(v) \tag{27}$$

$$\gamma_y(G) \quad = \quad \sum_{v \in V_{Buckets}(G)} B_y(v) \tag{28}$$

Proofs of Lemmas 1 and 2 are relegated to Appendix A. So far, we showed how to design DSH using a decision tree structure. However, it is not yet clear how to map data points to these buckets. In Section 5, we investigate this and provide an algorithm to design optimal decision trees.

4.1 Mapping Data Points

In the previous sections, we presented Algorithm 1 for solving (1) where we need to compute $X \cap U^x_{j,v}$ and $Y \cap U^y_{j,v}$ by mapping data points to the buckets. We did not clarify how this mapping can be done efficiently. We present Algorithm 2 for mapping data points to the buckets using hash-table search.

Algorithm 2 Mapping data points to the buckets using hash-table search

Inputs: List of buckets $V_{Buckets}$, permutations $perm_j, 1 \leq j \leq \#bands$, and a set of data points $X = \{x^1, \cdots, x^N\}$.

Outputs: $X \cap U^x_{j,v}$ for each bucket $v \in V_{Buckets}$ and bands $1 \leq j \leq \#bands$.

Create an empty hash-table.

Initialize $W^x_{v,j} = \varnothing$ for all $v \in V_{buckets}$ and $1 \leq j \leq \#bands$. # At the end, $W^x_{v,j} = X \cap U^x_{j,v}$.

For $v \in V_{Buckets}$

Insert $Seq^x(v)$ into the hash-table.

For $j = 1$ to $\#bands$

For $x \in X$

Search $perm_j(x)$ in the hash-table to find all $v \in V_{Buckets}$ for which $Seq^x(v)$ is a prefix of $perm_j(x)$, and insert x into $W^x_{v,j}$.

Note that we slightly modify the hash-table to search for values that are prefix of a query, rather than being exactly identical. In Appendix C, we show that the complexity of Algotithms 1 and 2 can be formulated as:

$$c_t.|V(G)| + \left(\frac{c_h.N}{\alpha(G)} + \frac{c_h.M}{\alpha(G)} + \frac{c_i.N.\gamma_x(G)}{\alpha(G)} + \frac{c_i.M.\gamma_y(G)}{\alpha(G)} + \frac{c_p.MN.\beta(G)}{\alpha(G)}\right).\log\frac{1}{1-TP} \qquad (29)$$

where c_t, c_h, c_i and c_p are constants not depending on N. Note that, the first term $c_t|V(G)|$ stands for the time required for calculating and storing the tree. The second and third terms, i.e., $\left(\frac{c_h.N}{\alpha(G)} + \frac{c_h.M}{\alpha(G)}\right).\log\frac{1}{1-TP}$ denote the time needed for inserting data points to the hash-table. The fourth and fifth terms, i.e., $\left(\frac{c_i.N.\gamma_x(G)}{\alpha(G)} + \frac{c_i.M.\gamma_y(G)}{\alpha(G)}\right).\log\frac{1}{1-TP}$ stand for the time required for mapping the data points from the hash-table to buckets. Finally, the last term, i.e., $\left(\frac{c_p.MN.\beta(G)}{\alpha(G)}\right).\log\frac{1}{1-TP}$ is the time of brute-force checking within each bucket.

5 Constructing optimal decision trees for DSH

In this section, we present an algorithm to design decision trees with complexity $O(N^{\lambda^*})$, where λ^* is defined below, and we show that it is the optimal decision tree.

Definition 4 *Given probability distributions $p = [p_{ij}]$ and $q = [q_{ij}]$, $1 \leq i \leq k, 1 \leq j \leq l$, and number of queries and classes M and N define $\delta = \frac{\log M}{\log N}$ and*

$$\begin{aligned} \mathcal{I} &= \{(\mu,\nu,\eta) \in \mathbb{R}^3 | \min(\mu,\nu) \geq \eta \geq 0, \\ &\quad \sum_{1\leq i\leq k, 1\leq j\leq l} p_{ij}^{1+\mu+\nu-\eta} (p_i^x)^{-\mu} (p_j^y)^{-\nu} = 1\} \end{aligned} \tag{30}$$

$$(\mu^*,\nu^*,\eta^*) = Arg\max_{\mathcal{I}} \frac{\max(1,\delta)+\mu+\nu\delta}{1+\mu+\nu-\eta} \tag{31}$$

$$\lambda^* = \frac{\max(1,\delta)+\mu^*+\nu^*\delta}{1+\mu^*+\nu^*-\eta^*} \tag{32}$$

Remark 3 *For any probability distribution $\mathbb{P}$, the parameters μ^*, ν^*, η^* and λ^* can be derived numerically from Algorithm 4 in Appendix B. The intuition behind the definition of $\mathcal{I}$ and (μ^*,ν^*,η^*) is that in Lemma 4 in Appendix D we show that for any decision tree G and the variables $A(v)$, $B_x(v)$, $B_y(v)$ and $B(v)$ defined in (17)-(20), we have $\sum_{v\in V_{Buckets}(G)} A(v)^{1+\mu+\nu-\eta} B_x(v)^{-\mu+\eta} B_y(v)^{-\nu+\eta} B(v)^{-\eta} \leq 1$ if $(\mu,\nu,\eta) \in \mathcal{I}$. Moreover, in proof of Theorem 2 we show that (μ^*,ν^*,η^*) are Lagrangian multipliers in an optimization problem to minimize the search complexity in Algorithm 1 while retaining a nearly perfect recovery.*

In Algorithm 3, we provide an approach for designing decision trees with complexity $O(N^{\lambda^*})$. The algorithm starts with the root, and at each step, it either accepts a node as a bucket, prunes a node, or branches a node into kl children based on the following constraints:

$$\begin{cases} \frac{A(v)}{B(v)} \geq N^{1+\delta-\lambda^*} p_0 q_0 & : \text{Accept bucket} \\ \frac{A(v)}{B_x(v)} \leq N^{1-\lambda^*} p_0 q_0 & : \text{Prune} \\ \frac{A(v)}{B_y(v)} \leq N^{\delta-\lambda^*} p_0 q_0 & : \text{Prune} \\ A(v)^{1+\mu^*+\nu^*-\eta^*} B_x(v)^{-\mu^*} B_y(v)^{-\nu^*} \leq N^{-\lambda^*} \bar{p}_0 & : \text{Prune} \\ otherwise & : \text{Branch into the } kl \text{ children} \end{cases} \tag{33}$$

where p_0, q_0 and $\bar{p}_0$ are defined as $\prod_{i,j,p_{ij}\neq 0} p_{ij}$, $\min\left(\prod_{i,j,q_{ij}\neq 0} q_{ij}, \prod_{i,p_i^x\neq 0} (p_i^x)^l, \prod_{j,p_j{}^y\neq 0} (p_j^y)^k\right)$ and ${p_0}^{1+\mu^*+\nu^*-\eta^*} {q_0}^{\mu^*+\nu^*-\eta^*}$. Note that p_0, q_0 and $\bar{p}_0$ are constants not depending on N. In Appendix C, we prove that in order to bound the complexity in (29) with $O(N^\lambda)$ for some $\lambda \in \mathbb{R}^+$, it is necessary and sufficient to find a decision tree G that satisfies the constraints ((54) − (58)).

Theorem 1 *No decision tree exists with overall complexity below $O(N^{\lambda^*})$.*

Theorem 2 *The decision tree construction of Algorithm 3 described in (33) results in a tree with complexity $O(N^{\lambda^*})$.*

In the other words, Theorem 2 proves that the tree G constructed by Algorithm 3 satisfies ((54)-(58)), and Theorem 1 shows that this is the optimal decision tree. For proofs of Theorems 1 and 2, see Appendices D and E. Note that, not only Theorem 2 guarantees that the number of the nodes in our decision tree is bounded by $O(N^{\lambda^*})$ but also it guarantees that the run time for mapping the data points to the decision tree and the number of comparisons that we need to do for the nodes with the collision is bounded by $O(N^{\lambda^*})$, see complexity equation (29) is section 4.1.

Algorithm 3 Recursive construction of the decision tree

Inputs: δ, $\mathcal{A}$, $\mathcal{B}$, $\mathbb{P}$, M and N. # We use Algorithm 4 to derive $\mu^*, \nu^*, \eta^*, \lambda^*, p_0, q_0, \bar{p}_0$, δ.
Outputs: $G = (V, E, f)$ and a subset of leaf nodes of the decision tree $V_{Buckets}$.
Initialization:
$Seq^x(root) \leftarrow \varnothing$, $Seq^y(root) \leftarrow \varnothing$.
$A(root) \leftarrow 1$, $B_x(root) \leftarrow 1$, $B_y(root) \leftarrow 1$, $B(root) \leftarrow 1$.
Recursive $TreeConstruction(root)$.
Procedure $TreeConstruction(v)$.
For $a_i \in \mathcal{A}$
For $b_j \in \mathcal{B}$
Create a new node w. # The node is created only if $p_{ij} \neq 0$.
$A(w) \leftarrow A(v).p_{ij}$
$B_x(w) \leftarrow B_x(v).p_i^x$
$B_y(w) \leftarrow B_y(v).p_j^y$
$B(w) \leftarrow B_x(w)B_y(w)$
$f(v, a, b) \leftarrow w$
$Seq^x(w) \leftarrow Seq^x(v)$, $Seq^y(w) \leftarrow Seq^y(v)$
$Seq^x(w).append(a)$, $Seq^y(w).append(b)$
If $\frac{A(w)}{B(w)} \geq N^{1+\delta-\lambda^*} p_0 q_0$ #Accept bucket
$V_{Buckets}.insert(w)$
Else If $\frac{A(w)}{B_x(w)} \geq N^{1-\lambda^*} p_0 q_0$ and $\frac{A(w)}{B_y(w)} \geq N^{\delta-\lambda^*} p_0 q_0$
and $A(w)^{1+\mu^*+\nu^*-\eta^*} B_x(w)^{-\mu^*} B_y(w)^{-\nu^*} \geq N^{-\lambda^*} \bar{p}_0$ #Branch
$TreeConstruction(w)$
Else #Prune
$f(v, a, b) \leftarrow null$.

6 Examples and Experiments

Example 1 *In this example, we compare the complexity of ForestDSH with the algorithm proposed by Dubiner in [26]. Equation* (126) *in [26] is computationally intractable which makes it impossible to compute the complexity for the algorithm presented there for general probability distributions. However, in the specific case where* $\mathbb{P}(p) = \begin{bmatrix} \frac{p}{2} & \frac{1-p}{2} \\ \frac{1-p}{2} & \frac{p}{2} \end{bmatrix}$, $0.5 \leq p \leq 1$, *Dubiner shows that when* $S \to \infty$, *the complexity of the algorithm proposed there grows asymptotically with* $O(N^{\frac{1}{p}})$. *On the other hand, from* (30) *and* (31) *the complexity of ForestDSH is nearly* $O(N^{1+\log\left(\frac{1}{p}\right)})$ *for these* $\mathbb{P}(p)$ *matrices (see Appendix F). While asymptotically, Dubiner provides better bounds than ForestDSH and LSH, from [26] it remains unclear how their algorithm performs for* $S < \infty$. *We implemented the algorithm from [26] (Note that no implementation is provided in [26]) and compared it to the implementation of ForestDSH. Figure 2, shows that while asymptotically (when* $S \to \infty$*) Dubiner algorithm provides better guarantees than ForestDSH, in practice, when* S *is not infinity (*$S = 1000$ *in Figure 2) Dubiner algorithm's performance is worse than that of LSH and ForestDSH.*

Example 2 *In this example, we reformulate the problem of solving* (2) *to the minimum inner product search problem (MIPS) [27] by transferring the data points from* $\mathcal{A}^S$ *and* $\mathcal{B}^S$ *to* R^{klS} *in a*

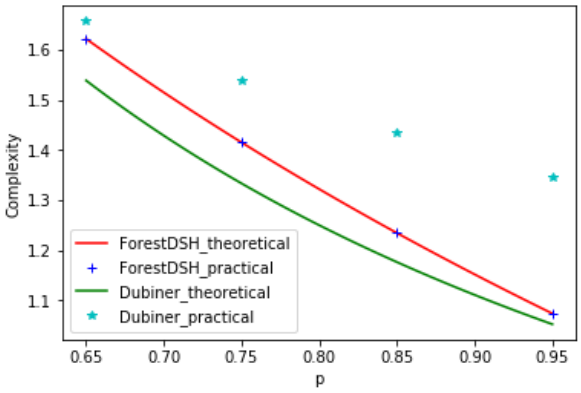


Figure 2: Comparing the theoretical and practical performances of Dubiner algorithm in [26] with ForestDSH for $S = 1000$.

way that $\log\left(\frac{\mathbb{P}(x,y)}{\mathbb{Q}(x,y)}\right)$ *is equal to the dot product in this new space. We transformed* $x \in \mathcal{A}^S$ *and* $y \in \mathcal{B}^S$ *to* $T(x) \in \mathbb{R}^{klS}$ *and* $T(y) \in \mathbb{R}^{klS}$ *as follows:*

$$T(x) = \big(f_{s,i,j}\big), 1 \le s \le S, 1 \le i \le k, 1 \le j \le l \tag{34}$$

$$f_{s,i,j} = \begin{cases} \frac{\log\left(\frac{p_{ij}}{q_{ij}}\right)}{w_{ij}} & \text{if } x_s = a_i \\ 0 & o.w. \end{cases} \tag{35}$$

$$T(y) = \big(g_{s,i,j}\big), 1 \le s \le S, 1 \le i \le k, 1 \le j \le l \tag{36}$$

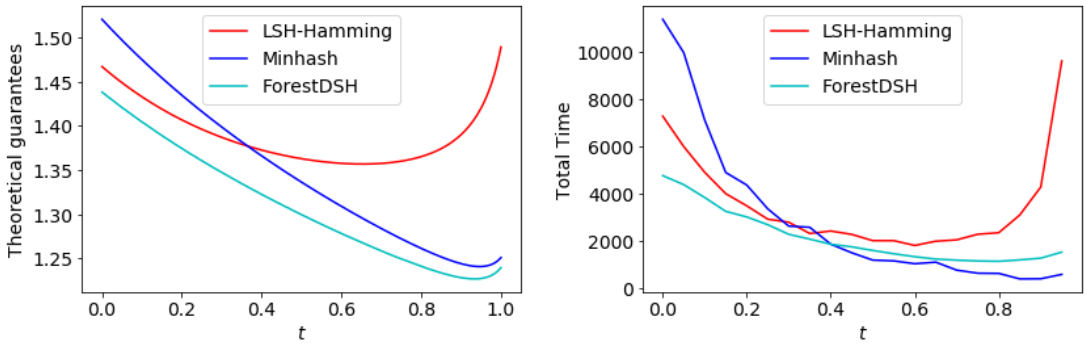
$$g_{s,i,j} = \begin{cases} w_{ij} & \text{if } y_s = b_j \\ 0 & o.w. \end{cases} \tag{37}$$

Then, we have $\log\left(\frac{\mathbb{P}(x,y)}{\mathbb{Q}(x,y)}\right) =< T(x), T(y) >$ *where* $< ., . >$ *stands for the inner product in* $\mathbb{R}^{klS}$*. In the other words, given any* $x = (x_1, \cdots, x_S)$*, for each* x_s *we transform it to a* $kl \times 1$ *vector with* l *non-zero elements. Similarly, given any* $y = (y_1, \cdots, y_S)$*, each* y_s *is transformed to a* $kl \times 1$ *vector with* k *non-zero elements. Therefore, finding pairs of data points with large* $\frac{\mathbb{P}(x,y)}{\mathbb{Q}(x,y)}$ *is equivalent to finding transformed data points with large dot product. Using this transformation, in Appendix G we show that the angle between both the true pairs and false pairs will be nearly* $\frac{\pi}{2}$ *for almost all the probability distributions (*$\frac{S_0}{M^2} \approx 0$*, using the notation from [27]). It is well known that MIPS performs poorly in detection of pairs that are nearly orthogonal [27]. Therefore, solving* (2) *by transforming it to a MIPS problem and using existing approaches fails.*

Example 3 *Here, we focus on the case where* $\mathcal{A} = \{0,1\}, \mathcal{B} = \{0,1\}$*,* $\mathbb{P} = \begin{bmatrix} 0.4 & 0.3 \\ 0.1 & 0.2 \end{bmatrix}$*,* $\mathbb{Q} = \begin{bmatrix} 0.35 & 0.35 \\ 0.15 & 0.15 \end{bmatrix}$*,* $\delta = 1$*, and* $M = N = 4$*. From Algorithm 4, we have* $\mu^* = 12.0791$*,* $\nu^* = 13.4206$*,* $\eta^* = 11.0959$*,* $\lambda^* = 1.7203$*. The decision tree is constructed from Algorithm 3 and is depicted in Figure 1. The nodes in the tree that are selected as bucket, i.e., satisfying* $\frac{A(v)}{B(v)} \ge N^{1+\delta-\lambda^*} p_0 q_0$*, are shown with a green check mark, and the nodes pruned out, satisfying either* $\frac{A(v)}{B_x(v)} \le N^{1-\lambda^*} p_0 q_0$ *or* $\frac{A(v)}{B_y(v)} \le N^{\delta-\lambda^*} p_0 q_0$ *or* $A(v)^{1+\mu^*+\nu^*-\eta^*} B_x(v)^{-\mu^*} B_y(v)^{-\nu^*} \le N^{-\lambda^*} \bar{p}_0$ *are shown with a red cross.*

For the non-leaf (intermediate) nodes, none of the above constraints holds. The bold edges show the paths in the tree corresponding to the data point $x = (0, 0)$. In this case, x falls into a single bucket v_1.

Experiment 1 *In this experiment, we compared the complexity for the three algorithms LSH-hamming, Minhash and ForestDSH for a range of probability distributions. We benchmark the three methods using matrices $\mathbb{P}(t) = \mathbb{P}_1(1-t) + \mathbb{P}_2 t$ where $0 \leq t \leq 1$, $\mathbb{P}_1 = \begin{bmatrix} 0.345 & 0 \\ 0.31 & 0.345 \end{bmatrix}$, $\mathbb{P}_2 = \begin{bmatrix} 0.019625 & 0 \\ 0.036875 & 0.9435 \end{bmatrix}$, and $\delta = 1$, i.e., $M = N$. The selection of $\mathbb{P}_1$ was such that the complexity of Minhash minus the complexity of LSH-hamming was maximized. $\mathbb{P}_2$ was selected such that the complexity of LSH-hamming minus the complexity of Minhash was maximized. Fig. 3 (a) shows the theoretical Complexities of Minhash, LSH-hamming and ForestDSH for each matrix. See Appendix H, for the details on the derivation of complexities for Minhash, LSH-hamming and ForestDSH. For instance, for $\mathbb{P}_1 = \begin{bmatrix} 0.345 & 0 \\ 0.31 & 0.345 \end{bmatrix}$, the theoretical per query complexities of Minhash, LSH-hamming and ForestDSH are equal to 0.5207, 0.4672 and 0.4384, respectively. We further consider N data points of dimension S, $\{x^1, \cdots, x^N\}$ and $\{y^1, \cdots, y^N\}$ where each (x^i, y^i) is generated from $\mathbb{P}(t)$, and x^i is independent from y^j for $i \neq j$ $(N = 2000, S = 6000)$. Then, we used ForestDSH, LSH-hamming and Minhash to find the matched pairs. In each case, we tuned r and #bands to achieve 99% true positive (recall) rate. Total simulation time for each of the three methods is plotted for each probability distribution in Figure 3 (b). The simulation times in Figure 3 (b) are consistent with the theoretical guarantees in Figure 3 (a). Figure 3 (b) shows that for sparse matrices, $(t \approx 1)$, Minhash and ForestDSH outperform LSH-hamming. In denser cases, $(t \approx 0)$, LSH-hamming and ForestDSH outperform Minhash. For $(t \leq 0.4)$, ForestDSH outperforms both Minhash and LSH-hamming. We further plotted $V(G(N))$, $\frac{\alpha(G(N))}{\beta(G(N))}$, $\frac{\alpha(G(N))}{\gamma_x(G(N))}$ and $\frac{\alpha(G(N))}{\gamma_y(G(N))}$ as a function of N for trees $G(N)$ constructed by Algorithm 3 for $\mathbb{P}(t = 0.25)$, where $M = N$. As predicted by Theorem 2, we observed that these quantities grow/decay proportional to N^{λ^*}, $N^{1+\delta-\lambda^*}$, $N^{1-\lambda^*}$ and $N^{\delta-\lambda^*}$, respectively.*



(a) (b)

Figure 3: Per query complexities of LSH-hamming, Minhash, and ForestDSH are plotted for all the probability distribution matrices $\mathbb{P}(t) = \mathbb{P}_1(1-t) + \mathbb{P}_2 t$ where $0 \leq t \leq 1$. (a) Theoretical guarantees, (b) Simulation time for $N = 2000$ and $S = 6000$.

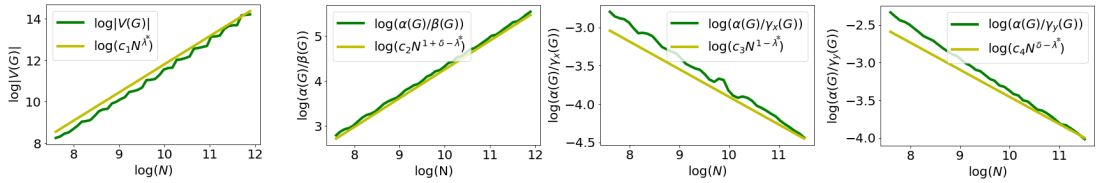


Figure 4: The bounds are sketched for c_1, c_2, c_3 and c_4 not depending on N confirming ((54)-(58)).

Experiment 2 *In this experiment, we applied DSH approach to the problem of spectral library search in mass spectrometry. In this problem, there are N spectra $X = x^1, \cdots, x^n$, and given a query spectrum y, our goal is to find the spectra x^i that maximize a probabilistic model $\mathbb{P}(y|x^i)$. $\mathbb{P}(y|x)$ is learned assuming that it can be factorized to i.i.d. components. To learn $p(y \mid x)$, each mass spectra is sorted based on the peak intensities, and in order to reduce the number of parameters we need to learn,* $\log Rank$ *of a peak is defined as the* $\log$ *of its rank. For any natural number n, the peaks at rank $\{2^n, \cdots, 2^{n+1}-1\}$ are associated with the rank $n+1$, e.g., $\log Rank(m) = 3$ for $m \in \{4,5,6,7\}$. Consider the training data from [1] shown in Figures 7 (a), (b) and (c) in Appendix I. Using these data, we learn the joint probability distribution $p(\log Rank(y_s) = i \mid \log Rank(x_s) = j)$. We applied DSH method to mass spectra after* $\log Rank$ *transformation, and we were able to speed up the search of* 20000 *mass spectra* 9 *times, in compare to brute force search. A true positive rate of* 90% *is obtained and DSH required* 9 *times less comparisons than the brute force. The pre-processing time is equal to* $22373ms$*, bringing the total time for ForestDSH under optimal parameters to* $705197ms$*. For the brute force, the total time would be* $22373 + 20000 * 20000 * 0.015$ *(preprocessing time + $N * M*$ time to check for false positive). For both LSH and MinHash approaches, the overall complexity was nearly the same as brute force. The amount of memory used peaks at 220MB. The mass spectrometry data for experiment 2, is shown in Figures 7 (a), (b) and (c) in case of* $\log Rank$ *at base* 4 *(a 4×4 matrix),* $\log Rank$ *at base* 2 *(an 8×8 matrix), and no* $\log Rank$ *transformation (a 51×51 matrix). For the mass spectrometry data shown in Figure 7 (a), the probability distribution $p(x, y)$ can be represented as*

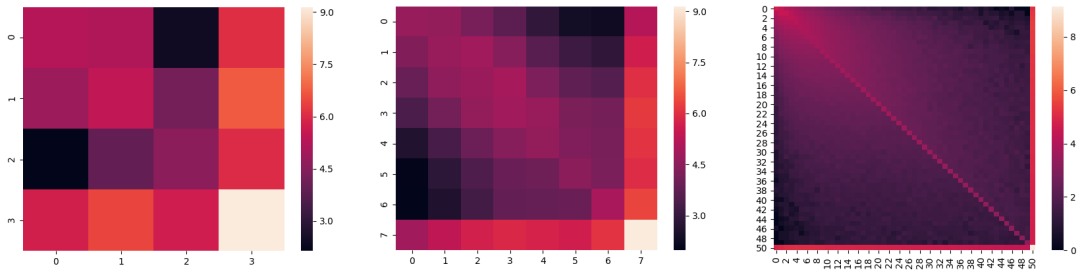
$$p_{4\times4}(x,y) = \begin{bmatrix} 0.000125 & 5.008081.10^{-5} & 9.689274.10^{-8} & 0.000404 \\ 5.008082.10^{-5} & 0.000209 & 6.205379.10^{-6} & 0.001921 \\ 9.689274.10^{-8} & 6.205379.10^{-6} & 2.688879.10^{-5} & 0.000355 \\ 0.000404 & 0.001921 & 0.000355 & 0.994165 \end{bmatrix} \tag{38}$$

From (32)*, $(\mu^*, \nu^*, \eta^*, \lambda^*)$ are derived as*

$$\mu^* = 1.151016 \tag{39}$$
$$\nu^* = 1.151016 \tag{40}$$
$$\eta^* = 0.813168 \tag{41}$$
$$\lambda^* = 1.326723 \tag{42}$$

6.1 Codes

For the codes, see https://github.com/ForestDSH.



(a) (b) (c)

Figure 5: Mass spectrometry joint probability distribution in the case of (a) $\log Rank$ at base 4, (b) $\log Rank$ at base 2, and (c) no $\log Rank$ transformation.

7 Conclusion

DSH algorithm proposed in this paper is comprehensive and efficient in the sense that for a wide range of probability distributions it enables us to capture the difference between pairs coming from a joint probability distributions and independent pairs. This algorithm is built upon a family of distribution sensitive hashes and is designed using a decision tree structure which is constructed recursively. Moreover, we prove that the decision tree introduced here has a complexity of $O(N^{\lambda^*})$ and there is no decision tree with overall complexity below it. Finally, we prove that this algorithm outperforms LSH-hamming and Minhash for various range of probability distributions in both the theory and simulations. Distribution sensitive hashing approach enabled speeding up the spectral library search in mass spectrometry by a factor of 9.

References

[1] A. M. Frank, M. E. Monroe, A. R. Shah, J. J. Carver, N. Bandeira, R. J. Moore, G. A. Anderson, R. D. Smith, and P. A. Pevzner, "Spectral archives: extending spectral libraries to analyze both identified and unidentified spectra," *Nature methods*, vol. 8, no. 7, p. 587, 2011.

[2] R. Aebersold and M. Mann, "Mass spectrometry-based proteomics," *Nature*, vol. 422, no. 6928, p. 198, 2003.

[3] S. Kim and P. A. Pevzner, "Ms-gf+ makes progress towards a universal database search tool for proteomics," *Nature communications*, vol. 5, p. 5277, 2014.

[4] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[5] Y. Prabhu and M. Varma, "Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning," pp. 263–272, 2014.

[6] J. Friedman, J. Bentley, and R. Finkel, "An algorithm for finding best matches in logarithmic time," *ACM Trans. Math. Software, 3(SLAC-PUB-1549-REV. 2*, pp. 209–226, 1976.

[7] H. Jain, Y. Prabhu, and M. Varma, "Extreme multi-label loss functions for recommendation, tagging, ranking & other missing label applications," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2016, pp. 935–944.

[8] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain, "Sparse local embeddings for extreme multi-label classification," in *Advances in neural information processing systems*, 2015, pp. 730–738.

[9] I. E.-H. Yen, X. Huang, P. Ravikumar, K. Zhong, and I. Dhillon, "Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification," in *International Conference on Machine Learning*, 2016, pp. 3069–3077.

[10] A. E. Choromanska and J. Langford, "Logarithmic time online multiclass prediction," in *Advances in Neural Information Processing Systems*, 2015, pp. 55–63.

[11] W. Liu and I. W. Tsang, "Making decision trees feasible in ultrahigh feature and label dimensions," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 2814–2849, 2017.

[12] J. Nam, E. L. Mencía, H. J. Kim, and J. Fürnkranz, "Maximizing subset accuracy with recurrent neural networks in multi-label classification," in *Advances in neural information processing systems*, 2017, pp. 5413–5423.

[13] P. Rai, C. Hu, R. Henao, and L. Carin, "Large-scale bayesian multi-label learning via topic-based label embeddings," in *Advances in Neural Information Processing Systems*, 2015, pp. 3222–3230.

[14] Y. Tagami, "Annexml: Approximate nearest neighbor search for extreme multi-label classification," in *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*. ACM, 2017, pp. 455–464.

[15] A. Niculescu-Mizil and E. Abbasnejad, "Label filters for large scale multilabel classification," in *Artificial Intelligence and Statistics*, 2017, pp. 1448–1457.

[16] W.-J. Zhou, Y. Yu, and M.-L. Zhang, "Binary linear compression for multi-label classification." in *IJCAI*, 2017, pp. 3546–3552.

[17] P. Indyk and R. Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," pp. 604–613, 1998.

[18] A. Gionis, P. Indyk, R. Motwani *et al.*, "Similarity search in high dimensions via hashing," vol. 99, no. 6, pp. 518–529, 1999.

[19] M. Bawa, T. Condie, and P. Ganesan, "Lsh forest: self-tuning indexes for similarity search," pp. 651–660, 2005.

[20] M. S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*. ACM, 2002, pp. 380–388.

[21] A. Andoni and I. Razenshteyn, "Optimal data-dependent hashing for approximate near neighbors," pp. 793–801, 2015.

[22] A. Andoni, P. Indyk, T. Laarhoven, I. Razenshteyn, and L. Schmidt, "Practical and optimal lsh for angular distance," pp. 1225–1233, 2015.

[23] A. Chakrabarti and O. Regev, "An optimal randomised cell probe lower bounds for approximate nearest neighbor searching," *In Proceedings of the Symposium on Foundations of Computer Science*, 2004.

[24] P. B. Miltersen, "Cell probe complexity-a survey," p. 2, 1999.

[25] A. Andoni, A. Naor, A. Nikolov, I. Razenshteyn, and E. Waingarten, "Data-dependent hashing via nonlinear spectral gaps," pp. 787–800, 2018.

[26] M. Dubiner, "A heterogeneous high-dimensional approximate nearest neighbor algorithm," *IEEE Transactions on Information Theory*, vol. 58, no. 10, pp. 6646–6658, 2012.

[27] A. Shrivastava and P. Li, "Asymmetric lsh (alsh) for sublinear time maximum inner product search (mips)," in *Advances in Neural Information Processing Systems*, 2014, pp. 2321–2329.

[28] M. Dubiner, "Bucketing coding and information theory for the statistical high-dimensional nearest-neighbor problem," *IEEE Transactions on Information Theory*, vol. 56, no. 8, pp. 4166 – 4179, Aug 2010.

A Proof of Lemmas 1 and 2

A.1 Proof of Lemma 1.

Lemma 1 is proved by induction on the depth of the node v. For example, consider v and its children $w = f(v, a_i, b_j)$. From (17), we have $A(w) = A(v).p_{ij}$. Therefore, (21) is concluded by induction.

A.2 Proof of Lemma 2.

Using ((21) − (24)), constraints ((7) − (10)) hold for $\alpha = \alpha(G)$, $\beta = \beta(G)$, $\gamma_x = \gamma_x(G)$ and $\gamma_y = \gamma_y(G)$. Since no two buckets are ancestor/descendant of each other, we have

$$\forall 1 \leq j \leq \#bands, v \neq v' : \qquad U^x_{j,v} \times U^y_{j,v} \cap U^x_{j,v'} \times U^y_{j,v'} = \varnothing \tag{43}$$

Therefore, using (43), (11) holds. This completes the proof that $U^x_{j,v}$ and $U^y_{j,v}$ are $(\alpha(G), \beta(G), \gamma_x(G), \gamma_y(G))$-sensitive.

B Deriving μ^*, ν^*, η^*, λ^*, p_0, q_0, $\bar{p}_0$ and δ for $\mathbb{P}$, M and N

In this section, an algorithm for deriving μ^*, ν^*, η^*, λ^*, p_0, q_0, $\bar{p}_0$ and δ for $\mathbb{P}$, M and N for the probability distribution $\mathbb{P}$ and δ is presented for a given probability distribution $\mathbb{P}$.

Algorithm 4 Deriving μ^*, ν^*, η^*, λ^*, p_0, q_0, $\bar{p}_0$ and δ for $\mathbb{P}$, M and N

Inputs: The probability distribution $\mathbb{P}$, $list_\mu$, $list_\nu$, $list_\eta$, threshold T, M and N.
Outputs: μ^*, ν^*, η^*, λ^*, p_0, q_0, $\bar{p}_0$ and δ.
Procedure
$\delta \leftarrow \frac{\log M}{\log N}$
$\lambda^* = 0$
For $\mu \in list_\mu$ # For some set of $list_\mu$, e.g., $\{0, 0.1, 0.2, \cdots, 10\}$.
For $\nu \in list_\nu$ # For some set of $list_\nu$, e.g., $\{0, 0.1, 0.2, \cdots, 10\}$.
For $\eta \in list_\eta$ # For some set of $list_\eta$, e.g., $\{0, 0.1, 0.2, \cdots, 10\}$.
If $|\sum_{i,j} p_{ij}^{1+\mu+\nu-\eta}(p_i^x)^{-\mu}(p_j^y)^{-\nu} - 1| \leq T$ and $\frac{\max(1,\delta)+\mu+\delta\nu}{1+\mu+\nu-\eta} > \lambda^*$. # For some small threshold T, e.g., 0.001
$\mu^* \leftarrow \mu$, $\nu^* \leftarrow \nu$, $\eta^* \leftarrow \eta$, $\lambda^* \leftarrow \frac{\max(1,\delta)+\mu+\delta\nu}{1+\mu+\nu-\eta}$
#p_0, q_0 and $\bar{p}_0$ are computed as follows.
$p_0 \leftarrow 1$, $q_0 \leftarrow 1$, $p_0^x \leftarrow 1$, $p_0^y \leftarrow 1$, $\bar{p}_0 \leftarrow 1$
For $a_i \in \mathcal{A}$
For $b_j \in \mathcal{B}$
$p_0 \leftarrow p_0.p_{ij}$
$q_0 \leftarrow q_0.q_{ij}$
$p_0^x \leftarrow p_0^x.p_i^x$
$p_0^y \leftarrow p_0^y.p_j^y$
$q_0 \leftarrow \min(q_0, p_0^x, p_0^y)$
$\bar{p}_0 \leftarrow {p_0}^{1+\mu^*+\nu^*-\eta^*} {q_0}^{\mu^*+\nu^*-\eta^*}$.

Remark 4 *In Algorithm 4, the parameters μ^*, ν^*, η^* and λ^* could be derived from newton method too.*

C Complexity Analysis

In this section, we bound the complexity of Algorithm 1. Note that, the complexity of Algorithm 1 is the summation of the following terms.

1. Tree Construction Complexity. $c_t.|V(G)|$ where c_t is a constant representing per node complexity of constructing a node and $|V(G)|$ is the number of nodes in the tree.

2. Data Mapping Complexity. The complexity of this hash-table search grows with

$$\begin{aligned} & \text{Complexity}_1 \\ = \; & c_h((\#bands).|X| + |V_{Buckets}|) + c_i \sum_{j=1}^{\#bands} \sum_{v \in V_{Buckets}} |X \cap U_{j,v}^x| \qquad (44) \\ & + c_h((\#bands).|Y| + |V_{Buckets}|) + c_i \sum_{j=1}^{\#bands} \sum_{v \in V_{Buckets}} |Y \cap U_{j,v}^y| \qquad (45) \end{aligned}$$

$\text{Complexity}_1 + (\#bands).(c_h.N + c_h.M + c_i.N.\gamma_x(G) + c_i.M.\gamma_y(G))$ where c_h, c_i represent complexity of insertion in the hash-table and insertion in the buckets, respectively. Note that, c_t, c_h, c_i and c_p are constants not depending on N.

3. Complexity of Checking Positive Calls, i.e., $(\#bands).c_p.\sum_{v\in V_{Buckets}(G)}|X\cap U^x_{j,v}|.|Y\cap U^y_{j,v}|\big)$ where c_p represents the complexity of computing $\mathbb{P}(y\mid x)$ for a positive.

   From $((7)-(11))$, we have

$$\begin{aligned} & E\big(\sum_{j=1}^{\#bands}\sum_{v\in V_{Buckets}(G)}|X\cap U^x_{j,v}|\big) \\ = \ & (\#bands).N.\sum_{v\in V_{Buckets}(G)}B_y(v)=(\#bands).N.\gamma_x(G) \qquad (46) \\ & E\big(\sum_{j=1}^{\#bands}\sum_{v\in V_{Buckets}(G)}|Y\cap U^y_{j,v}|\big) \\ = \ & (\#bands).M.\sum_{v\in V_{Buckets}(G)}B_x(v)=(\#bands).M.\gamma_y(G) \qquad (47) \end{aligned}$$

   Similarly, we conclude that

$$\begin{aligned} & E\big(\sum_{j=1}^{\#bands}\sum_{v\in V_{Buckets}(G)}|X\cap U^x_{j,v}||Y\cap U^y_{j,v}|\big) \\ = \ & (\#bands).M.N.\sum_{v\in V_{Buckets}(G)}B(v)=(\#bands).M.N.\beta(G) \qquad (48) \end{aligned}$$

Now, the question is how we can select $\#bands$ such that the true positive rate, defined as the ratio of true pairs that are called positive is high. In each band, the chance of a pair $(x,y)\sim\mathbb{P}$ being called positive is computed as $\alpha(G)=\sum_{v\in V_{Buckets}(G)}A(v)$. Therefore, the overall true positive rate can be computed as:

$$\begin{aligned} TP &= Prob((x,y)\text{ called positive in Algorithm 1 }\mid(x,y)\sim\mathbb{P}) \qquad (49) \\ &= 1-\prod_{j=1}^{\#bands}\left(1-\sum_{v\in V_{Buckets}}Prob(x\in U^x_{v,j}\&y\in U^y_{v,j}\mid(x,y)\sim\mathbb{P})\right) \qquad (50) \\ &= 1-(1-\alpha(G))^{\#bands} \qquad (51) \end{aligned}$$

Using (51), and the inequality $(1-x)^{\frac{c}{x}}<e^{-c}$, the minimum possible value of $\#bands$ to ensure true positive rate TP can be computed as

$$\#bands = \lceil\frac{\log\frac{1}{1-TP}}{\alpha(G)}\rceil \qquad (52)$$

where $\lceil r\rceil$ stands for the smallest integer greater than or equal to r. Therefore, the total complexity is computed as

$$c_t.|V(G)|+\left(\frac{c_h.N}{\alpha(G)}+\frac{c_h.M}{\alpha(G)}+\frac{c_i.N.\gamma_x(G)}{\alpha(G)}+\frac{c_i.M.\gamma_y(G)}{\alpha(G)}+\frac{c_p.MN.\beta(G)}{\alpha(G)}\right).\log\frac{1}{1-TP} \qquad (53)$$

In order to bound (53) with $O(N^{\lambda})$ for some $\lambda \in \mathbb{R}^{+}$, it is necessary and sufficient to find a tree G that satisfies the following constraints:

$$
\begin{aligned}
|V(G)| &= O(N^{\lambda}) && (54)\\
\frac{\alpha(G)}{\beta(G)} &= \Omega(N^{1+\delta-\lambda}) && (55)\\
\frac{\alpha(G)}{\gamma_x(G)} &= \Omega(N^{1-\lambda}) && (56)\\
\frac{\alpha(G)}{\gamma_y(G)} &= \Omega(N^{\delta-\lambda}) && (57)\\
\alpha(G) &= \Omega(N^{\max(1,\delta)-\lambda}) && (58)
\end{aligned}
$$

where $\delta = \frac{\log M}{\log N}$.

D Proof of Theorem 1

In order to prove Theorem 1, we first state the following two lemmas.

Lemma 3 *The function* $f(\theta, \theta_1, \theta_2, \theta_3) = \theta^{1+\rho_1+\rho_2+\rho_3}{\theta_1}^{-\rho_1}{\theta_2}^{-\rho_2}{\theta_3}^{-\rho_3}$ *is a convex function on the region* $(\theta, \theta_1, \theta_2, \theta_3) \in \mathbb{R}^{4+}$ *where* $(\rho_1, \rho_2, \rho_3) \in \mathbb{R}^{3+}$. [2]

Lemma 4 $\sum_{v \in V_{Buckets}(G)} A(v)^{1+\mu+\nu-\eta} B_x(v)^{-\mu+\eta} B_y(v)^{-\nu+\eta} B(v)^{-\eta} \leq 1$ *for any* $(\mu, \nu, \eta) \in \mathcal{I}$.

Proof of Lemma 3 and Lemma 4, are relegated to Appendices D.1 and D.2, respectively. Consider (μ^*, ν^*, η^*) that satisfy (31). For any decision tree satisfying ((54) − (58)), we have:

$$
\begin{aligned}
& \Big(\frac{\sum_{v \in V_{Buckets}(G)} A(v)}{|V_{Buckets}(G)|}\Big)^{1+\mu^*+\nu^*-\eta^*} \Big(\frac{\sum_{v \in V_{Buckets}(G)} B_x(v)}{|V_{Buckets}(G)|}\Big)^{-\mu^*+\eta^*}\\
& \times \Big(\frac{\sum_{v \in V_{Buckets}(G)} B_y(v)}{|V_{Buckets}(G)|}\Big)^{-\nu^*+\eta^*} \Big(\frac{\sum_{v \in V_{Buckets}(G)} B(v)}{|V_{Buckets}(G)|}\Big)^{-\eta^*}\\
\leq\ & \frac{\sum_{v \in V_{Buckets}(G)} A(v)^{1+\mu^*+\nu^*-\eta^*} B_x(v)^{-\mu^*+\eta^*} B_y(v)^{-\nu^*+\eta^*} B(v)^{-\eta^*}}{|V_{Buckets}(G)|} && (59)\\
\leq\ & \frac{1}{|V_{Buckets}(G)|} && (60)
\end{aligned}
$$

where (59) holds to the convexity of $f(\theta, \theta_1, \theta_2, \theta_3) = \theta^{1+\rho_1+\rho_2+\rho_3}{\theta_1}^{-\rho_1}{\theta_2}^{-\rho_2}{\theta_3}^{-\rho_3}$ in Lemma 3 and (60) follows from Lemma 4. Therefore, we have

$$
\begin{aligned}
& \Big(\sum_{v \in V_{Buckets}(G)} A(v)\Big)^{1+\mu^*+\nu^*-\eta^*} \Big(\sum_{v \in V_{Buckets}(G)} B_x(v)\Big)^{-\mu^*+\eta^*}\\
& \times \Big(\sum_{v \in V_{Buckets}(G)} B_y(v)\Big)^{-\nu^*+\eta^*} \Big(\sum_{v \in V_{Buckets}(G)} B(v)\Big)^{-\eta^*}\\
\leq\ & 1 && (61)
\end{aligned}
$$

[2] For any natural number n, $\mathbb{R}^{n+}$ denotes as the set of all n-tuples non-negative real numbers.

On the other hand, using (60) and the definitions of $\alpha(G)$, $\beta(G)$, $\gamma_x(G)$ and $\gamma_y(G)$ in $((25)-(28))$, we have

$$\big(\alpha(G)\big)^{1+\mu^*+\nu^*-\eta^*}\big(\gamma_x(G)\big)^{-\mu^*+\eta^*}\big(\gamma_y(G)\big)^{-\nu^*+\eta^*}\big(\beta(G)\big)^{-\eta^*} \quad \leq \quad 1 \tag{62}$$

Therefore, from $((54)-(58))$ and (62) we have

$$\begin{aligned}
& 1 \\
\geq \quad & \big(\alpha(G)\big)^{1+\mu^*+\nu^*-\eta^*}\big(\gamma_x(G)\big)^{-\mu^*+\eta^*}\big(\gamma_y(G)\big)^{-\nu^*+\eta^*}\big(\beta(G)\big)^{-\eta^*} \\
= \quad & \alpha(G)\left(\frac{\alpha(G)}{\gamma_x(G)}\right)^{\mu^*-\eta^*}\left(\frac{\alpha(G)}{\gamma_y(G)}\right)^{\nu^*-\eta^*}\left(\frac{\alpha(G)}{\beta(G)}\right)^{\eta^*} && (63) \\
\geq \quad & \left(N^{\max(1,\delta)-\lambda}\right)\left(N^{1-\lambda}\right)^{\mu^*-\eta^*}\left(N^{\delta-\lambda}\right)^{\nu^*-\eta^*}\left(N^{1+\delta-\lambda}\right)^{\eta^*} && (64) \\
= \quad & N^{\max(1,\delta)+\mu^*+\delta\nu^*-(1+\mu^*+\nu^*-\eta^*)\lambda} && (65)
\end{aligned}$$

Therefore, we have

$$\lambda \quad \geq \quad \lambda^* = \frac{\max(1,\delta)+\mu^*+\delta\nu^*}{1+\mu^*+\nu^*-\eta^*} \tag{66}$$

D.1 Proof of Lemma 3

The Hessian matrix for $f(\theta,\theta_1,\theta_2,\theta_3)$ is represented as

$$\begin{aligned}
& H(\theta,\theta_1,\theta_2,\theta_3) \\
= \quad & f(\theta,\theta_1,\theta_2,\theta_3) \\
& \times \begin{bmatrix}
\frac{(1+\rho_1+\rho_2+\rho_3)(\rho_1+\rho_2+\rho_3)}{\theta^2} & \frac{-\rho_1(1+\rho_1+\rho_2+\rho_3)}{\theta\theta_1} & \frac{-\rho_2(1+\rho_1+\rho_2+\rho_3)}{\theta\theta_2} & \frac{-\rho_3(1+\rho_1+\rho_2+\rho_3)}{\theta\theta_3} \\
\frac{-\rho_1(1+\rho_1+\rho_2+\rho_3)}{\theta\theta_1} & \frac{\rho_1(\rho_1+1)}{\theta_1^2} & \frac{\rho_1\rho_2}{\theta_1\theta_2} & \frac{\rho_1\rho_3}{\theta_1\theta_3} \\
\frac{-\rho_2(1+\rho_1+\rho_2+\rho_3)}{\theta\theta_2} & \frac{\rho_1\rho_2}{\theta_1\theta_2} & \frac{\rho_2(\rho_2+1)}{\theta_2^2} & \frac{\rho_2\rho_3}{\theta_2\theta_3} \\
\frac{-\rho_3(1+\rho_1+\rho_2+\rho_3)}{\theta\theta_3} & \frac{\rho_1\rho_3}{\theta_1\theta_3} & \frac{\rho_2\rho_3}{\theta_2\theta_3} & \frac{\rho_3(\rho_3+1)}{\theta_3^2}
\end{bmatrix} \\
= \quad & f(\theta,\theta_1,\theta_2,\theta_3) \\
& \times \begin{bmatrix}
W^2+\frac{\rho_1+\rho_2+\rho_3}{\theta^2} & WW_1-\frac{\rho_1}{\theta\theta_1} & WW_2-\frac{\rho_2}{\theta\theta_2} & WW_3-\frac{\rho_3}{\theta\theta_3} \\
W_1W-\frac{\rho_1}{\theta\theta_1} & W_1^2+\frac{\rho_1}{\theta_1^2} & W_1W_2 & W_1W_3 \\
W_2W-\frac{\rho_2}{\theta\theta_2} & W_2W_1 & W_2^2+\frac{\rho_2}{\theta_2^2} & W_2W_3 \\
W_3W-\frac{\rho_3}{\theta\theta_3} & W_3W_1 & W_3W_2 & W_3^2+\frac{\rho_3}{\theta_3^2}
\end{bmatrix}
\end{aligned} \tag{67}$$

where $W=\frac{\rho_1+\rho_2+\rho_3}{\theta}$, $W_i=\frac{\rho_i}{\theta_i}$ for any $i\in\{1,2,3\}$. In order to show that the function $f(\theta,\theta_1,\theta_2,\theta_3)$ is a convex function it is necessary and sufficient to prove that $H(\theta,\theta_1,\theta_2,\theta_3)$ is positive semidefinite on $\mathbb{R}^{4+}$ On the other hand, for positive semidefinite matrices we have

1. For any non-negative scalar a and positive semidefinite matrix M, aM is positive semidefinite.
2. For positive semidefinite matrices M_1 and M_2, M_1+M_2 is positive semidefinite.

As $f(\theta,\theta_1,\theta_2,\theta_3)>0$ for any $\theta,\theta_1,\theta_2,\theta_3$, it is sufficient to prove that $\frac{H(\theta,\theta_1,\theta_2,\theta_3)}{f(\theta,\theta_1,\theta_2,\theta_3)}$ is positive semidefinite. Define, $M_1=\begin{bmatrix} W^2 & WW_1 & WW_2 & WW_3\\ W_1W & W_1^2 & W_1W_2 & W_1W_3\\ W_2W & W_2W_1 & W_2^2 & W_2W_3\\ W_3W & W_3W_1 & W_3W_2 & W_3^2\end{bmatrix}$ and $M_2=\begin{bmatrix} \frac{\rho_1+\rho_2+\rho_3}{\theta^2} & -\frac{\rho_1}{\theta\theta_1} & -\frac{\rho_2}{\theta\theta_2} & -\frac{\rho_3}{\theta\theta_3}\\ -\frac{\rho_1}{\theta\theta_1} & \frac{\rho_1}{\theta_1^2} & 0 & 0\\ -\frac{\rho_2}{\theta\theta_2} & 0 & \frac{\rho_2}{\theta_2^2} & 0\\ -\frac{\rho_3}{\theta\theta_3} & 0 & 0 & \frac{\rho_3}{\theta_3^2}\end{bmatrix}$.
The matrices M_1 and M_2 are positive semidefinite as for any non-zero vector $z=\begin{bmatrix}a & b & c & d\end{bmatrix}$, we have $zM_1z^T\geq 0$ and $zM_2z^T\geq 0$, i.e.,

$$\begin{aligned}
&\begin{bmatrix}a & b & c & d\end{bmatrix}\begin{bmatrix} W^2 & WW_1 & WW_2 & WW_3\\ W_1W & W_1^2 & W_1W_2 & W_1W_3\\ W_2W & W_2W_1 & W_2^2 & W_2W_3\\ W_3W & W_3W_1 & W_3W_2 & W_3^2\end{bmatrix}\begin{bmatrix}a\\b\\c\\d\end{bmatrix}\\
=\;& (Wa+W_1b+W_2c+W_3d)^2 && (68)\\
\geq\;& 0 && (69)\\
&\begin{bmatrix}a & b & c & d\end{bmatrix}\begin{bmatrix} \frac{\rho_1+\rho_2+\rho_3}{\theta^2} & -\frac{\rho_1}{\theta\theta_1} & -\frac{\rho_2}{\theta\theta_2} & -\frac{\rho_3}{\theta\theta_3}\\ -\frac{\rho_1}{\theta\theta_1} & \frac{\rho_1}{\theta_1^2} & 0 & 0\\ -\frac{\rho_2}{\theta\theta_2} & 0 & \frac{\rho_2}{\theta_2^2} & 0\\ -\frac{\rho_3}{\theta\theta_3} & 0 & 0 & \frac{\rho_3}{\theta_3^2}\end{bmatrix}\begin{bmatrix}a\\b\\c\\d\end{bmatrix}\\
=\;& \rho_1\big(\frac{a}{\theta}-\frac{b}{\theta_1}\big)^2+\rho_2\big(\frac{a}{\theta}-\frac{c}{\theta_2}\big)^2\\
&+\rho_3\big(\frac{a}{\theta}-\frac{d}{\theta_3}\big)^2 && (70)\\
\geq\;& 0 && (71)
\end{aligned}$$

where (71) is concluded as $\rho_1,\rho_2,\rho_3\geq 0$.

D.2 Proof of Lemma 4

First of all, note that $B(v)=B_x(v)B_y(v)$. Let us define

$$D(v)=\sum_{v\in V_{Buckets}(G)} A(v)^{1+\mu+\nu-\eta}B_x(v)^{-\mu}B_y(v)^{-\nu} \tag{72}$$

We show that

$$\sum_{v\in V_{Buckets}(G)} D(v)\leq 1 \tag{73}$$

by induction on the number of nodes in the tree. If the tree has only one node, i.e., *root*, then (73) is holds as $A(root)=1$, $B_x(root)=1$ and $B_y(root)=1$ from the definition of $A(v)$, $B_x(v)$ and $B_y(v)$ in $((17)-(19))$. Assume that (73) holds for any decision tree with $|G|<Z$. Our goal is to prove that (73) holds for a decision tree with $|G|=Z$. Assume v_{11} is the node with maximum length in G and consider a tree G' constructed by removing v_{11} and all its siblings $v_{ij}, 1\leq i\leq k, 1\leq j\leq l$

belonging to the same parent v. In the other words, for the tree G' we have[3]

$$V(G') = V(G) - \{w_{11}, \cdots, w_{kl}\} \tag{74}$$
$$V_b(G') = V_b(G) - \{w_{11}, \cdots, w_{kl}\} + \{v\} \tag{75}$$

Then, we have

$$\begin{aligned}
& \sum_{v \in V_{Buckets}(G)} D(v) \\
\leq & \sum_{v \in V_{Buckets}(G')} D(v) - A(v)^{1+\mu+\nu-\eta} B_x(v)^{-\mu} B_y(v)^{-\nu} \\
& + \sum_{i,j} A(v_{ij})^{1+\mu+\nu-\eta} B_x(v_{ij})^{-\mu} B_y(v_{ij})^{-\nu} && (76) \\
= & \sum_{\in V_{Buckets}(G')} D(v) - A(v)^{1+\mu+\nu-\eta} B_x(v)^{-\mu} B_y(v)^{-\nu} \\
& + \sum_{i,j} \Big(A(v)^{1+\mu+\nu-\eta} B_x(v)^{-\mu+\eta} B_y(v)^{-\nu} \\
& \times {p_{ij}}^{1+\mu+\nu-\eta} (p_i^x)^{-\mu} (p_i^y)^{-\nu} \Big) && (77) \\
= & \sum_{v \in V_{Buckets}(G')} D(v) - A(v)^{1+\mu+\nu-\eta} B_x(v)^{-\mu} B_y(v)^{-\nu} \\
& \times \left(1 - \sum_{i,j} {p_{ij}}^{1+\mu+\nu-\eta} (p_i^x)^{-\mu} (p_j^y)^{-\nu} \right) && (78) \\
= & \sum_{v \in V_{Buckets}(G')} D(v) && (79) \\
= & \ 1 && (80)
\end{aligned}$$

where (76) holds from the definition of tree G', (77) follows from the recursive definition of $A(v)$, $B_x(v)$ and $B_y(v)$ in $((17) - (19))$ and (79) holds, note that from the definition of μ, ν and η in (30), i.e.,

$$\sum_{i,j} {p_{ij}}^{1+\mu+\nu-\eta} (p_i^x)^{-\mu} (p_j^y)^{-\nu} = 1 \tag{81}$$

Therefore, we conclude that

$$\sum_{v \in V_{Buckets}(G)} D(v) \leq 1 \tag{82}$$

Note that, the inequality (76) becomes an equality only in cases where the tree is homogeneous and none of the children are pruned.

[3]Note that, $V_b(G')$ satisfies bucket-list property, e.g., there is no bucket in the tree that is ancestor of another bucket.

E Proof of Theorem 2

In order to prove Theorem 2, we first present the following lemma.

Lemma 5 *Given a fixed $N \in \mathbb{N}$ and probability distribution $\mathbb{P}$, consider the following region R_λ*

$$\mathcal{R}(\lambda, r_{ij}, n) = \Bigg\{ \lambda, r_{ij}, n \ s.t. \ \ \lambda \geq 0, \sum_{i,j} r_{ij} = 1, r_{ij} \geq 0, r_{ij} \in \mathbb{R}^+, n \in \mathbb{R}^+, \tag{83}$$

$$\sum_{i,j} r_{ij} \log p_{ij} - \sum_{i,j} r_{ij} \log r_{ij} \geq \frac{(\max(1,\delta) - \lambda) \log N}{n}, \tag{84}$$

$$\sum_{i,j} r_{ij} \log p_{ij} \geq \frac{(\max(1,\delta) - 2\lambda) \log N}{n}, \tag{85}$$

$$\sum_{i,j} r_{ij} \log p_{ij} - \sum_{i,j} r_{ij} \log p_i^x \geq \frac{(1-\lambda) \log N}{n}, \tag{86}$$

$$\sum_{i,j} r_{ij} \log p_{ij} - \sum_{i,j} r_{ij} \log p_j^y \geq \frac{(\delta-\lambda) \log N}{n}, \tag{87}$$

$$\sum_{i,j} r_{ij} \log p_{ij} - \sum_{i,j} r_{ij} \log q_{ij} \geq \frac{(1+\delta-\lambda) \log N}{n} \Bigg\} \tag{88}$$

[4] *Then, $(\lambda^*, r_{ij}^*, n^*)$ defined in Definition 4 is a member of $\mathcal{R}(\lambda, r_{ij}, n)$.*

The proof of Lemma 5 is relegated to Appendix E.2. From Lemma 5, $(\lambda^*, r_{ij}^*, n^*)$ has the following properties:

$$-r_{ij}^* \leq 0 \tag{89}$$

$$\sum_{i,j} r_{ij}^* - 1 = 0 \tag{90}$$

$$\sum_{i,j} r_{ij}^* \log p_{ij} - \sum_{i,j} r_{ij}^* \log r_{ij}^* \geq \frac{(\max(1,\delta) - \lambda^*) \log N}{n^*} \tag{91}$$

$$\sum_{i,j} r_{ij}^* \log p_{ij} \geq \frac{(\max(1,\delta) - 2\lambda^*) \log N}{n^*} \tag{92}$$

$$\sum_{i,j} r_{ij}^* \log p_{ij} - \sum_{i,j} r_{ij}^* \log p_i^x \geq \frac{(1-\lambda^*) \log N}{n^*} \tag{93}$$

$$\sum_{i,j} r_{ij}^* \log p_{ij} - \sum_{i,j} r_{ij}^* \log p_j^y \geq \frac{(\delta-\lambda^*) \log N}{n^*} \tag{94}$$

$$\sum_{i,j} r_{ij}^* \log p_{ij} - \sum_{i,j} r_{ij}^* \log q_{ij} \geq \frac{(1+\delta-\lambda^*) \log N}{n^*} \tag{95}$$

$$\tag{96}$$

[4] Recall that, for simplicity we use the notation $\sum_{i,j}$ and $\prod_{i,j}$ instead of $\sum_{1 \leq i \leq k, 1 \leq j \leq l}$ and $\prod_{1 \leq i \leq k, 1 \leq j \leq l}$, respectively.

Let us prove that the following tree construction steps in Algorithm 3 result in a tree that satisfies ((54) − (58)).

$$\begin{cases} \frac{A(v)}{B(v)} \geq N^{1+\delta-\lambda^*} p_0 q_0 & : accept\ bucket \\ \frac{A(v)}{B_x(v)} \leq N^{1-\lambda^*} p_0 q_0 & : prune \\ \frac{A(v)}{B_y(v)} \leq N^{\delta-\lambda^*} p_0 q_0 & : prune \\ A(v)^{1+\mu^*+\nu^*-\eta^*} B_x(v)^{-\mu^*} B_y(v)^{-\nu^*} \leq N^{-\lambda^*} \bar{p}_0 & : prune \\ otherwise & : branch\ into\ the\ kl\ children \end{cases} \tag{97}$$

E.1 Proof of Theorem 2

Consider the set of $r^*_{ij} = p_{ij}^{1+\mu^*+\nu^*-\eta^*} (p_i^x)^{-\mu^*} (p_j^y)^{-\nu^*}$ and $n^* = \frac{(\max(1,\delta)-\lambda^*)\log N}{\sum r^*_{ij} \log \frac{p_{ij}}{r^*_{ij}}}$. Note that we assume p_{ij} and q_{ij} are non-zero[5]. Consider $n_{ij} = \lceil n^* r^*_{ij} \rceil$ if $r^*_{ij} > \frac{1}{2}$ and $n_{ij} = \lfloor n^* r^*_{ij} \rfloor$ if $r^*_{ij} \leq \frac{1}{2}$. Therefore, we have $n^* - kl < \sum_{ij} n_{ij} \leq n^*$. For any $v \in V(G)$, define the set $S_{ij}(v)$ as follows

$$S_{ij}(v) \stackrel{\triangle}{=} \{s \mid 1 \leq s \leq depth(v), seq^x_s(v) = a_i \& seq^y_s(v) = b_j\} \tag{98}$$

where $depth(v)$ is the depth of node v in the tree, $seq^x_s(v)$ and $seq^y_x(v)$ stand for position s in the strings $seq^x(v)$ and seq^y(v), respectively. Now, consider a node v in the graph that satisfies the following constraints:

$$|S_{ij}(v)| \quad = \quad n_{ij}, \forall 1 \leq i \leq k, 1 \leq j \leq l \tag{99}$$

The number of nodes v that satisfy this constraint $\binom{n^*}{n_{11},\cdots,n_{kl}}$. Moreover, define

$$|V_{n_{11},\cdots,n_{kl}}| \quad \stackrel{\triangle}{=} \quad \{v \in V(G) \mid |S_{ij}(v)| = n_{ij}, \forall 1 \leq i \leq k, 1 \leq j \leq l\} \tag{100}$$

First, we prove that the node v, or one of its ancestors, yet designated as a bucket by Algorithm 3. In order to show this, we need to prove that:

$$A(v) \quad \geq \quad e^{\sum_{i,j} n^* r^*_{ij} \log p_{ij}} p_0 \geq N^{\max(1,\delta)-2\lambda^*} p_0 \tag{101}$$

$$\frac{A(v)}{B(v)} \quad \geq \quad e^{\sum n^* r^*_{ij} \log p_{ij}} e^{-\sum n^* r^*_{ij} \log q_{ij}} p_0 q_0 \geq N^{1+\delta-\lambda^*} p_0 q_0 \tag{102}$$

$$\frac{A(v)}{B_x(v)} \quad \geq \quad e^{\sum n^* r^*_{ij} \log p_{ij}} e^{-\sum n^* r^*_{ij} \log p^x_i} p_0 q_0 \geq N^{1-\lambda^*} p_0 q_0 \tag{103}$$

$$\frac{A(v)}{B_y(v)} \quad \geq \quad e^{\sum n^* r^*_{ij} \log p_{ij}} e^{-\sum n^* r^*_{ij} \log p^y_j} p_0 q_0 \geq N^{\delta-\lambda^*} p_0 q_0 \tag{104}$$

[5]Note that in the cases where q_{ij} is zero, then from the definition of q_{ij}, p_{ij} would also be equal to zero. Therefore, we will ignore those branches during the tree construction.

where p_0 and q_0 are defined as $\prod_{i,j} p_{ij}$ and $\min(\prod_{i,j} q_{ij}, \prod_i (p_i^x)^l, \prod_j (p_j^y)^k)$. However, $A(v)$, $B(v)$, $B_x(v)$ and $B_y(v)$ can be computed from

$$\begin{aligned}
A(v) &= \prod_{i,j} p_{ij}^{n_{ij}} = e^{\sum_{i,j} n_{ij} \log p_{ij}} \geq e^{\sum_{i,j} (n^* r_{ij}^* + 1) \log p_{ij}} \\
&\geq e^{\sum_{i,j} n^* r_{ij}^* \log p_{ij}} (\prod_{i,j} p_{ij}) = e^{\sum_{i,j} n^* r_{ij}^* \log p_{ij}} p_0 \qquad (105) \\
B(v) &= \prod_{i,j} q_{ij}^{n_{ij}} = e^{\sum_{i,j} n_{ij} \log q_{ij}} \\
&\leq e^{\sum_{i,j} (n^* r_{ij}^* - 1) \log q_{ij}} \leq \frac{e^{\sum_{i,j} n^* r_{ij}^* \log q_{ij}}}{\prod_{i,j} q_{ij}} = \frac{e^{\sum_{i,j} n^* r_{ij}^* \log q_{ij}}}{q_0} \qquad (106) \\
B_x(v) &= \prod_{i,j} (p_i^x)^{n_{ij}} = e^{\sum_{i,j} n_{ij} \log p_i^x} \\
&\leq e^{\sum_{i,j} (n^* r_{ij}^* - 1) \log (p_i^x)} \leq \frac{e^{\sum_{i,j} n^* r_{ij}^* \log p_i^x}}{\prod_{i,j} p_i^x} = \frac{e^{\sum_{i,j} n^* r_{ij}^* \log p_i^x}}{q_0} \qquad (107) \\
B_y(v) &= \prod_{i,j} (p_j^y)^{n_{ij}} = e^{\sum_{i,j} n_{ij} \log p_j^y} \\
&\leq e^{\sum_{i,j} (n^* r_{ij}^* - 1) \log p_j^y} \leq \frac{e^{\sum_{i,j} n^* r_{ij}^* \log p_j^y}}{\prod_{i,j} p_j^y} = \frac{e^{\sum_{i,j} n^* r_{ij}^* \log p_j^y}}{q_0} \qquad (108)
\end{aligned}$$

Then, from ((91) − (95)) and ((105) − (108)) we have

$$\begin{aligned}
A(v) &\geq N^{\max(1,\delta) - 2\lambda^*} p_0 \qquad (109) \\
\frac{A(v)}{B(v)} &\geq N^{1+\delta-\lambda^*} p_0 q_0 \qquad (110) \\
\frac{A(v)}{B_x(v)} &\geq N^{1-\lambda^*} p_0 q_0 \qquad (111) \\
\frac{A(v)}{B_y(v)} &\geq N^{\delta-\lambda^*} p_0 q_0 \qquad (112)
\end{aligned}$$

Moreover, we have

$$\begin{aligned}
& A(v)^{1+\mu^*+\nu^*-\eta^*} B_x(v)^{-\mu^*} B_y(v)^{-\nu^*} \\
= \; & A(v) \left(\frac{A(v)}{B_x(v)}\right)^{\mu^*-\eta^*} \left(\frac{A(v)}{B_y(v)}\right)^{\nu^*-\eta^*} \left(\frac{A(v)}{B(v)}\right)^{\eta^*} \qquad (113) \\
\geq \; & \left(N^{\max(1,\delta)-2\lambda^*}\right) \left(N^{1-\lambda^*}\right)^{\mu^*-\eta^*} \left(N^{\delta-\lambda^*}\right)^{\nu^*-\eta^*} \left(N^{1+\delta-\lambda^*}\right)^{\eta^*} \bar{p}_0 \qquad (114) \\
= \; & N^{\max(1,\delta)-\lambda^*+\mu^*+\delta\nu^*-(1+\mu^*+\nu^*-\eta^*)\lambda^*} = N^{-\lambda^*} \bar{p}_0 \qquad (115)
\end{aligned}$$

where $\bar{p}_0$ is defined as ${p_0}^{1+\mu^*+\nu^*-\eta^*} {q_0}^{\mu^*+\nu^*-\eta^*}$. Therefore, every node v satisfying (99) is an accepted bucket (or one of its ancestors is an accepted bucket). Now, we derive a lower bound on $\alpha(G)$ as

follows.

$$\begin{aligned}
\alpha(G) &= \sum_{v\in V_{Buckets}(G)} A(v) \\
&\geq \sum_{v\in V_{n_{11},\cdots,n_{kl}}} A(v) && (116)\\
&\geq |V_{n_{11},\cdots,n_{kl}}|A(v) && (117)\\
&\geq |V_{n_{11},\cdots,n_{kl}}|e^{\sum_{i,j} n^* r^*_{ij}\log p_{ij}}p_0 && (118)
\end{aligned}$$

where $V_{n_{11},\cdots,n_{kl}}$ is the set of nodes that satisfies (99). $|V_{n_{11},\cdots,n_{kl}}|$ is lower bounded as

$$\begin{aligned}
|V_{n_{11},\cdots,n_{kl}}| &= \binom{n}{n_{11},\cdots,n_{kl}} \\
&\geq \frac{n!}{(kl)!\prod_{i,j} n_{ij}!} \geq \frac{(\frac{n}{e})^n\sqrt{2\pi n}}{(kl)!\prod_{i,j}(\frac{n_{ij}}{e})^{n_{ij}}\sqrt{2\pi n_{ij}}e} && (119)\\
&\geq \prod_{i,j}(\frac{n_{ij}}{n})^{-n_{ij}} n^{\frac{1-kl}{2}}\frac{(2\pi)^{\frac{1-kl}{2}}e^{-kl}}{(kl)!} && (120)\\
&= ce^{-\sum_{i,j} n_{ij}\log(\frac{n_{ij}}{n})}n^{\frac{1-kl}{2}} && (121)\\
&\geq ce^{-n\sum_{i,j} r^*_{ij}\log r^*_{ij}} && (122)
\end{aligned}$$

for some constant $c = \frac{(2\pi)^{\frac{1-kl}{2}}e^{-kl}}{(kl)!}n^{\frac{1-kl}{2}}$ depending on n, k and l. (119) is true as for any natural number n we have $\sqrt{2\pi n}(\frac{n}{e})^n \leq n! < \sqrt{2\pi n}(\frac{n}{e})^n e$. (122) follows as $a\log\frac{1}{a}$ is an increasing function for $0 \leq x \leq 0.5$, and a decreasing function for $0.5 \leq x \leq 1$. Therefore, from (118) and (122), $\alpha(G) \geq N^{\max(1,\delta)-\lambda^*}$ is concluded. Similarly, $((54)-(58))$ are proved as follows.

$$\begin{aligned}
\frac{\alpha(G)}{\beta(G)} &= \frac{\sum_{v\in V_{Buckets}(G)} A(v)}{\sum_{v\in V_{Buckets}(G)} B(v)} \geq N^{1+\delta-\lambda^*}p_0q_0 && (123)\\
\frac{\alpha(G)}{\gamma_x(G)} &= \frac{\sum_{v\in V_{Buckets}(G)} A(v)}{\sum_{v\in V_{Buckets}(G)} B_x(v)} \geq N^{1-\lambda^*}p_0q_0 && (124)\\
\frac{\alpha(G)}{\gamma_y(G)} &= \frac{\sum_{v\in V_{Buckets}(G)} A(v)}{\sum_{v\in V_{Buckets}(G)} B_y(v)} \geq N^{\delta-\lambda^*}p_0q_0 && (125)
\end{aligned}$$

where (123) and (125) is concluded from $((109)-(112))$ and the fact that $\frac{\sum_i a_i}{\sum_i b_i} \geq c$ if $\frac{a_i}{b_i} \geq c$ and $b_i > 0$ for any i.

Now, we need to provide an outer bound on the number of nodes $|V(G)|$ to show (54).

$$\begin{aligned}
&\geq 1 \\
&\geq \sum_{v\in V(G)} A(v)^{1+\mu^*+\nu^*-\eta^*}B_x(v)^{-\mu^*+\eta^*}B_y(v)^{-\nu^*+\eta^*}B(v)^{-\eta^*} && (126)\\
&\geq \sum_{v\in V(G)} N^{-\lambda^*}\bar{p}_0 && (127)\\
&= |V(G)|N^{-\lambda^*}\bar{p}_0 && (128)
\end{aligned}$$

where (126) follows from Lemma 4 and (127) is true from (115). Therefore, we conclude that $|V(G)| = O(N^{\lambda^*})$.

E.2 Proof of Lemma 5

Consider the optimization problem of finding the member of $(\lambda, r_{ij}, n) \in \mathcal{R}(\lambda, r_{ij}, n)$ with minimum λ. This optimization problem is a convex optimization problem[6]. Therefore, writing the KKT conditions, we have

$$\begin{aligned}
& F(r_{ij}, n, \lambda) \\
= \; & \lambda + \sum_{i,j} \mu_{1ij}(-r_{ij}) + \mu_2\big(\frac{(\max(1,\delta)-\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} + \sum_{i,j} r_{ij}\log r_{ij}\big) \\
& + \mu_2'\big(\frac{(\max(1,\delta)-2\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij}\big) \\
& + \mu_3\big(\frac{(1-\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} + \sum_{i,j} r_{ij}\log p_i^x\big) \\
& + \mu_4\big(\frac{(\delta-\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} + \sum_{i,j} r_{ij}\log p_j^y\big) \\
& + \mu_5\big(\frac{(1+\delta-\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} + \sum_{i,j} r_{ij}\log q_{ij}\big) + \mu_6\big(\sum_{i,j} r_{ij} - 1\big) \qquad (132)
\end{aligned}$$

[6]Note that $f_1(n) = \frac{1}{n}$, $f_2(r) = r\log r$ and $f_3(r) = ar$ are convex functions. Therefore, as the sum of the convex functions is a convex function, the optimization problem ((83) − (88)) is in the form of convex optimization problem

$$\begin{aligned}
\underset{\lambda}{\text{Minimize}} \;\; & f(x) && (129)\\
\text{subject to} \;\; & g_i(x) \leq 0, i \in \{1,\cdots,m\} && (130)\\
& h_j(x) = 0, j \in \{1,\cdots,p\} && (131)
\end{aligned}$$

where $x \in \mathbb{R}^n$, $f(x)$ and $g_i(x)$ are convex functions and $h_j(x)$ is affine functions.

where

$$-r_{ij} \leq 0, \mu_{1ij} r_{ij} = 0 \tag{133}$$

$$\frac{(\max(1,\delta)-\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} + \sum_{i,j} r_{ij}\log r_{ij} \leq 0 \tag{134}$$

$$\mu_2\left(\frac{(\max(1,\delta)-\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} + \sum_{i,j} r_{ij}\log r_{ij}\right) = 0 \tag{135}$$

$$\frac{(1-\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} + \sum_{i,j} r_{ij}\log p_i^x \leq 0 \tag{136}$$

$$\mu_2'\left(\frac{(\max(1,\delta)-2\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij}\right) = 0 \tag{137}$$

$$\frac{(\max(1,\delta)-2\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} \leq 0 \tag{138}$$

$$\mu_3\left(\frac{(1-\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} + \sum_{i,j} r_{ij}\log p_i^x\right) = 0 \tag{139}$$

$$\frac{(\delta-\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} + \sum_{i,j} r_{ij}\log p_j^y \leq 0 \tag{140}$$

$$\mu_4\left(\frac{(\delta-\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} + \sum_{i,j} r_{ij}\log p_j^y\right) = 0 \tag{141}$$

$$\frac{(1+\delta-\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} + \sum_{i,j} r_{ij}\log q_{ij} \leq 0 \tag{142}$$

$$\mu_5\left(\frac{(1+\delta-\lambda)\log N}{n} - \sum_{i,j} r_{ij}\log p_{ij} + \sum_{i,j} r_{ij}\log q_{ij}\right) = 0 \tag{143}$$

$$\sum_{i,j} r_{ij} - 1 = 0 \tag{144}$$

$$\mu_2, \mu_3, \mu_4, \mu_5 \geq 0 \tag{145}$$

From (133), μ_{1ij} is zero if r_{ij} is a non-zero number. Therefore, we only keep i and j where $r_{ij} \neq 0$ and $\mu_{1ij} = 0$.

$$\frac{dF(r_{ij},n,\lambda)}{dr_{ij}} = 0 \rightarrow \mu_2 + \mu_2\log r_{ij} + \mu_3\log p_i^x + \mu_4\log p_j^y + \mu_5\log q_{ij} + \mu_6$$

$$-(\mu_2 + \mu_2' + \mu_3 + \mu_4 + \mu_5)\log p_{ij} = 0 \tag{146}$$

$$\rightarrow r_{ij}^{\mu_2} = p_{ij}^{\mu_2+\mu_2'+\mu_3+\mu_4+\mu_5}(p_i^x)^{-\mu_3}(p_j^y)^{-\mu_4}q_{ij}^{-\mu_5}e^{-\mu_2-\mu_6} \tag{147}$$

Consider the following two cases.

1. $\mu_2 = 0$. In this case, all the constraints are affine functions and therefore we have a linear programming problem and the feasible set of this linear programming problem is a polyhedron.

From (83), the polyhedron is bounded, i.e., $0 \le r_{ij} \le M$ for some constant M. Assume that the polyhedron is nonempty, otherwise the solution is ∞. Moreover, a nonempty bounded polyhedron cannot contain a line, thus it must have a basic feasible solution and the optimal solutions are restricted to the corner points.

2. $\mu_2 \neq 0$. From (133), μ_{1ij} is zero if r_{ij} is a non-zero number. Therefore, we only keep i and j where $r_{ij} \neq 0$ and $\mu_{1ij} = 0$.

$$\frac{dF(r_{ij}, n, \lambda)}{dr_{ij}} = 0 \rightarrow \mu_2 + \mu_2 \log r_{ij} + \mu_3 \log p_i^x + \mu_4 \log p_j^y + \mu_5 \log q_{ij} + \mu_6$$

$$-(\mu_2 + \mu_2' + \mu_3 + \mu_4 + \mu_5) \log p_{ij} = 0 \tag{148}$$

$$\rightarrow r_{ij}^{\mu_2} = p_{ij}^{\mu_2+\mu_2'+\mu_3+\mu_4+\mu_5} (p_i^x)^{-\mu_3} (p_j^y)^{-\mu_4} q_{ij}^{-\mu_5} e^{-\mu_2-\mu_6} \tag{149}$$

$$\rightarrow r_{ij} = c p_{ij}^{\frac{\mu_2+\mu_2'+\mu_3+\mu_4+\mu_5}{\mu_2}} (p_i^x)^{-\frac{\mu_3}{\mu_2}} (p_j^y)^{-\frac{\mu_4}{\mu_2}} q_{ij}^{-\frac{\mu_5}{\mu_2}} \tag{150}$$

$$\frac{dF(r_{ij}, n, \lambda)}{dn} = 0$$

$$\rightarrow -\big(\mu_2(\max(1, \delta) - \lambda) + \mu_2'(\max(1, \delta) - 2\lambda) + \mu_3(1 - \lambda) + \mu_4(\delta - \lambda)$$

$$+\mu_5(1 + \delta - \lambda)\big) \frac{\log N}{n^2} = 0 \tag{151}$$

$$\rightarrow (\mu_2 + \mu_2')(\max(1, \delta) - \lambda) - \mu_2' \lambda + \mu_3(1 - \lambda)$$

$$+\mu_4(\delta - \lambda) + \mu_5(1 + \delta - \lambda) = 0 \tag{152}$$

$$\rightarrow \lambda = \frac{(\mu_2 + \mu_2') \max(1, \delta) + \mu_3 + \mu_4 \delta + \mu_5(1 + \delta)}{\mu_2 + 2\mu_2' + \mu_3 + \mu_4 + \mu_5} \tag{153}$$

Summing (135), (139), (141) and (143), we have

$$\mu_2\Big(\frac{(\max(1, \delta) - \lambda) \log N}{n} - \sum_{i,j} r_{ij} \log p_{ij} + \sum_{i,j} r_{ij} \log r_{ij}\Big)$$

$$+\mu_3\Big(\frac{(1 - \lambda) \log N}{n} - \sum_{i,j} r_{ij} \log p_{ij} + \sum_{i,j} r_{ij} \log p_i^x\Big)$$

$$+\mu_4\Big(\frac{(\delta - \lambda) \log N}{n} - \sum_{i,j} r_{ij} \log p_{ij} + \sum_{i,j} r_{ij} \log p_j^y\Big)$$

$$+\mu_5\Big(\frac{(1 + \delta - \lambda) \log N}{n} - \sum_{i,j} r_{ij} \log p_{ij} + \sum_{i,j} r_{ij} \log q_{ij}\Big) = 0 \tag{154}$$

Using $q_{ij} = p_i^x p_j^y$ we have $r_{ij} = c p_{ij}^{\frac{\mu_2+\mu_2'+\mu_3+\mu_4+\mu_5}{\mu_2}} (p_i^x)^{-\frac{\mu_3+\mu_5}{\mu_2}} (p_j^y)^{-\frac{\mu_4+\mu_5}{\mu_2}}$ in (154), we have

$$\begin{aligned} & \big((\mu_2 + \mu_2')(\max(1, \delta) - \lambda) - \mu_2' \lambda + \mu_3(1 - \lambda) \\ & +\mu_4(\delta - \lambda) + \mu_5(1 + \delta - \lambda)\big) \frac{\log N}{n} \\ = \quad & \mu_2 \log c \end{aligned} \tag{155}$$

On the other hand, as $\mu_2 \neq 0$, we conclude that $c = 1$. Moreover, we have $\lambda = \frac{(1+\epsilon)\max(1,\delta)+\mu+\nu\delta}{1+\epsilon+\mu+\nu+\eta}$, where μ, ν and η are defined as:

$$\mu = \frac{\mu_3 + \mu_5}{\mu_2} \tag{156}$$

$$\nu = \frac{\mu_4 + \mu_5}{\mu_2} \tag{157}$$

$$\eta = \frac{\mu_2' - \mu_5}{\mu_2} \tag{158}$$

$$\epsilon = \frac{\mu_2'}{\mu_2} \tag{159}$$

From (135), as $\mu_2 \neq 0$ we conclude that $n = \frac{(\max(1,\delta)-\lambda)\log N}{\sum r_{ij} \log \frac{p_{ij}}{r_{ij}}}$. Since $q_{ij} = q_i q_j$, we have

$$r_{ij} = p_{ij}^{1+\mu+\nu+\eta}(p_i^x)^{-\mu}(p_j^y)^{-\nu} \tag{160}$$

$$\text{s.t.} \qquad \nu, \epsilon \geq 0, \eta \geq -\min(\mu,\nu), \sum_{i,j} r_{ij} = 1 \tag{161}$$

Assume that $\eta > 0$, therefore $r_{ij} < p_{ij}$ as $p_{ij} \leq \min(q_i, q_j)$. On the other hand, $\sum_{i,j} r_{ij} = \sum p_{ij} = 1$ which contradicts our assumption that $\eta > 0$. Thus, $\eta \leq 0$. Therefore, in order to derive λ we should derive

$$\max_{\epsilon,\mu,\nu\geq 0, 0\geq\eta\geq -\min(\mu,\nu), \sum_{i,j} p_{ij}^{1+\mu+\nu+\eta}(p_i^x)^{-\mu}(p_j^y)^{-\nu}=1} \min_{\lambda} F(r_{ij}, n, \lambda)$$

$$= \max_{\epsilon,\mu,\nu\geq 0, 0\geq\eta\geq -\min(\mu,\nu), \sum_{i,j} p_{ij}^{1+\mu+\nu+\eta}(p_i^x)^{-\mu}(p_j^y)^{-\nu}=1} \frac{(1+\epsilon)\max(1,\delta)+\mu+\nu\delta}{1+\epsilon+\mu+\nu+\eta} \tag{162}$$

$$= \max_{\mu,\nu\geq 0, 0\geq\eta\geq -\min(\mu,\nu), \sum_{i,j} p_{ij}^{1+\mu+\nu+\eta}(p_i^x)^{-\mu}(p_j^y)^{-\nu}=1} \max_{\epsilon\geq 0} \frac{(1+\epsilon)\max(1,\delta)+\mu+\nu\delta}{1+\epsilon+\mu+\nu+\eta} \tag{163}$$

$$= \max_{\mu,\nu\geq 0, 0\geq\eta\geq -\min(\mu,\nu), \sum_{i,j} p_{ij}^{1+\mu+\nu+\eta}(p_i^x)^{-\mu}(p_j^y)^{-\nu}=1} \max\left(\frac{\max(1,\delta)+\mu+\nu\delta}{1+\mu+\nu+\eta}, \max(1,\delta)\right) \tag{164}$$

$$= \max_{\min(\mu,\nu)\geq\eta\geq 0, \sum_{i,j} p_{ij}^{1+\mu+\nu-\eta}(p_i^x)^{-\mu}(p_j^y)^{-\nu}=1} \max\left(\frac{\max(1,\delta)+\mu+\nu\delta}{1+\mu+\nu-\eta}, 1, \delta\right) \tag{165}$$

(164) is true as $\max_{\epsilon\geq 0} \frac{a+c\epsilon}{b+\epsilon} = \max(c, \frac{a}{b})$ for $b > 0$. (165) follows from replacing η with $-\eta$. On the other hand, $(\mu, \nu, \eta) = (0, 0, 0)$ satisfies the conditions of maximization, resulting in $\frac{\max(1,\delta)+\mu+\nu\delta}{1+\mu+\nu-\eta} = \max(1, \delta)$. Therefore, we have

$$\max_{\min(\mu,\nu)\geq\eta\geq 0, \sum_{i,j} p_{ij}^{1+\mu+\nu-\eta}(p_i^x)^{-\mu}(p_j^y)^{-\nu}=1} \frac{\max(1,\delta)+\mu+\nu\delta}{1+\mu+\nu-\eta} \geq \max(1,\delta) \tag{166}$$

which ensures that the term $\max\left(\frac{\max(1,\delta)+\mu+\nu\delta}{1+\mu+\nu-\eta}, 1, \delta\right)$ can be replaced with $\frac{\max(1,\delta)+\mu+\nu\delta}{1+\mu+\nu-\eta}$. Define (μ^*, ν^*, η^*) as follows

$$\begin{aligned}(\mu^*, \nu^*, \eta^*) &= Arg \max_{\min(\mu,\nu)\geq\eta\geq 0, \sum_{i,j} p_{ij}^{1+\mu+\nu-\eta}(p_i^x)^{-\mu}(p_j^y)^{-\nu}=1} \\ & \frac{\max(1,\delta)+\mu+\nu\delta}{1+\mu+\nu-\eta} \end{aligned} \tag{167}$$

Therefore, we set:

$$r_{ij}^* = p_{ij}^{1+\mu^*+\nu^*-\eta^*}(p_i^x)^{-\mu^*}(p_j^y)^{-\nu^*} \tag{168}$$

$$\lambda^* = \frac{\max(1,\delta)+\mu^*+\nu^*\delta}{1+\mu^*+\nu^*-\eta^*} \tag{169}$$

$$n^* = \frac{(\max(1,\delta)-\lambda^*)\log N}{\sum r_{ij}^* \log \frac{p_{ij}}{r_{ij}^*}} \tag{170}$$

$((167)-(170))$ result in a local minimum for the optimization problem of finding the minimum the member of $(\lambda, r_{ij}, n) \in \mathcal{R}(\lambda, r_{ij}, n)$ with minimum λ. It is easy to check that this optimum point satisfies $((83)-(88))$.

F Approximation of the theoretical complexity of ForestDSH

In this section, for the probability matrices $\mathbb{P}(p) = \begin{bmatrix} \frac{p}{2} & \frac{1-p}{2} \\ \frac{1-p}{2} & \frac{p}{2} \end{bmatrix}$, $0.5 \leq p \leq 1$, ForestDSH theoretical complexity and LSH theoretical complexity, i.e., $1+\log\left(\frac{1}{p}\right)$ are compared.

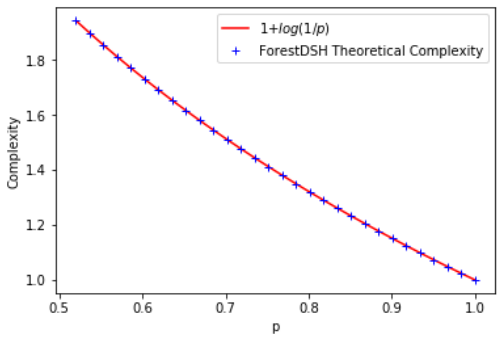


Figure 6: ForestDSH theoretical complexity almost overlaps with the LSH theoretical complexity, i.e., $1+\log\left(\frac{1}{p}\right)$ for the probability matrices $\mathbb{P}(p)$ where $0.5 \leq p \leq 1$.

G Further Discussion on MIPS

In order to use MIPS to solve this problem, i.e., (2), we need to derive optimal weights w_{ij} to minimize the norm M^2 in [27]. The term M stands for the radius of the space which is computed as follows: $M^2 = E\big(||x||\big)^2 + E\big(||y||\big)^2$. Therefore, from (34)-(37) we conclude that

$M^2 = \sum_{ij} \left(p_j^y w_{ij}^2 + \frac{p_i^x \log^2(\frac{p_{ij}}{q_{ij}})}{w_{ij}^2} \right)$ which results in optimal $w_{ij} = \left(\frac{p_i^x}{p_j^y}\right)^{0.25} \left(\mid \log \frac{p_{ij}}{q_{ij}} \mid \right)^{0.5}$. On the other hand, for $(x, y) \sim Q(x, y)$ we have

$$E\big(||x||||y||\big) \geq E(< T(x), T(y) >) = S \sum_{ij} q_{ij} \mid \log(\frac{p_{ij}}{q_{ij}}) \mid \tag{171}$$

In order to have nearly one true positive rate and sub-quadratic complexity we need $S_0 \leq Sd_{KL}(p_{ij}||q_{ij})$ and $cS_0 \geq -Sd_{KL}(q_{ij}||p_{ij})$ where d_{KL} stands for kullback leibler divergence. Moreover, we should have $M^2 \geq S \sum_{ij} \sqrt{q_{ij}} |\log(\frac{p_{ij}}{q_{ij}})|$. Setting $c = 0$, S_0 and M as above, the complexity will be more than 1.9 for any 2×2 probability distribution matrix. The reason is that the transferred data points are nearly orthogonal to each other and this makes it very slow to find maximum inner product using the existing method [27].

H Complexities of Minhash, LSH-hamming and ForestDSH

In this section, we derive the complexities of Minhash, LSH-hamming and ForestDSH for $\mathbb{P}_1 = \begin{bmatrix} 0.345 & 0 \\ 0.31 & 0.345 \end{bmatrix}$ for instance. Complexities are computed for the other probability distributions similarly.

H.1 Complexity of Minhash

For Minhash the query complexity is

$$N^{\min(mh_1, mh_2, mh_3, mh_4)} \tag{172}$$

where $mh_1 = \frac{\log \frac{p_{00}}{1-p_{11}}}{\log \frac{q_{00}}{1-q_{11}}}$, $mh_2 = \frac{\log \frac{p_{01}}{1-p_{10}}}{\log \frac{q_{01}}{1-q_{10}}}$, $mh_3 = \frac{\log \frac{p_{10}}{1-p_{01}}}{\log \frac{q_{10}}{1-q_{01}}}$ and $mh_4 = \frac{\log \frac{p_{11}}{1-p_{00}}}{\log \frac{q_{11}}{1-q_{00}}}$. Similarly for $\mathbb{P}_1$, the per query complexity is derived and is equal to 0.5207.

H.2 Complexity of LSH-hamming

In the case of LSH-hamming, the query complexity is

$$O(N^{\min(\frac{\log(p_{00}+p_{11})}{\log(q_{00}+q_{11})}, \frac{\log(p_{01}+p_{10})}{\log(q_{01}+q_{10})})}) \tag{173}$$

and the storage required for the algorithm is $O(N^{1+\min(\frac{\log(p_{00}+p_{11})}{\log(q_{00}+q_{11})}, \frac{\log(p_{01}+p_{10})}{\log(q_{01}+q_{10})})})$. Similarly for $\mathbb{P}_1$, the per query complexity is derived and is equal to 0.4672.

H.3 Complexity of ForestDSH

From Definition 4, we derive λ^* as follows

$$\begin{aligned}
(\mu^*, \nu^*, \eta^*) &= Arg \max_{\min(\mu,\nu) \geq \eta > 0, \sum_{i,j} p_{ij}^{1+\mu+\nu-\eta} (p_i^x)^{-\mu} (p_j^y)^{-\nu} = 1} \frac{1+\mu+\nu}{1+\mu+\nu-\eta} && (174) \\
&= (4.6611, 4.6611, 3.1462) && (175) \\
\lambda^* &= \frac{1+\mu^*+\nu^*}{1+\mu^*+\nu^*-\eta^*} && (176) \\
&= 1.4384 && (177)
\end{aligned}$$

Note that $\delta = 1$ and the per query complexity is equal to 0.4384.

I Joint Probability Distributions Learned on Mass Spectrometry Data

The mass spectrometry data for experiment 2, is shown in Figures 7 (a), (b) and (c) in case of $\log Rank$ at base 4 (a 4×4 matrix), $\log Rank$ at base 2 (an 8×8 matrix), and no $\log Rank$ transformation (a 51×51 matrix). For the mass spectrometry data shown in Figure 7 (a), the probability distribution $p(x, y)$ can be represented as

$$p_{4\times4}(x,y) = \begin{bmatrix} 0.000125 & 5.008081.10^{-5} & 9.689274.10^{-8} & 0.000404 \\ 5.008082.10^{-5} & 0.000209 & 6.205379.10^{-6} & 0.001921 \\ 9.689274.10^{-8} & 6.205379.10^{-6} & 2.688879.10^{-5} & 0.000355 \\ 0.000404 & 0.001921 & 0.000355 & 0.994165 \end{bmatrix} \quad (178)$$

From (32), $(\mu^*, \nu^*, \eta^*, \lambda^*)$ are derived as

$$\mu^* = 1.151016 \quad (179)$$
$$\nu^* = 1.151016 \quad (180)$$
$$\eta^* = 0.813168 \quad (181)$$
$$\lambda^* = 1.326723 \quad (182)$$

For the mass spectrometry data shown in Figure 7 (b), the probability distribution $p(x, y)$ is represented as

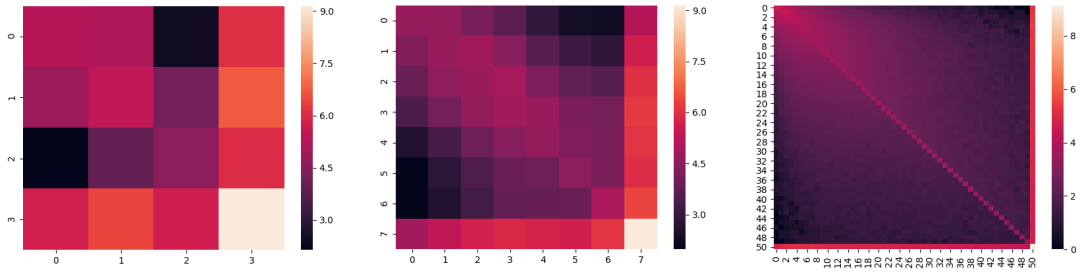
$$p_{8\times8}(x,y) = \begin{bmatrix} 3.458.10^{-5} & 1.442.10^{-5} & 5.434.10^{-6} & 1.723.10^{-6} \\ 1.442.10^{-5} & 3.708.10^{-5} & 2.550.10^{-5} & 8.706.10^{-6} \\ 5.434.10^{-6} & 2.550.10^{-5} & 3.907.10^{-5} & 2.948.10^{-5} \\ 1.723.10^{-6} & 8.706.10^{-6} & 2.948.10^{-5} & 4.867.10^{-5} \\ 2.921.10^{-7} & 1.561.10^{-6} & 6.442.10^{-6} & 1.813.10^{-5} \\ 7.496.10^{-8} & 4.809.10^{-7} & 2.008.10^{-6} & 6.098.10^{-6} \\ 6.718.10^{-8} & 2.680.10^{-7} & 1.251.10^{-6} & 4.531.10^{-6} \\ 5.023.10^{-5} & 1.574.10^{-4} & 3.671.10^{-4} & 5.539.10^{-4} \end{bmatrix} \quad (183)$$

$$\begin{bmatrix} 2.920.10^{-7} & 7.496.10^{-8} & 6.718.10^{-8} & 5.023.10^{-5} \\ 1.561.10^{-6} & 4.809.10^{-7} & 2.680.10^{-7} & 1.575.10^{-4} \\ 6.442.10^{-6} & 2.008.10^{-6} & 1.251.10^{-6} & 3.672.10^{-4} \\ 1.813.10^{-5} & 6.098.10^{-6} & 4.532.10^{-6} & 5.539.10^{-4} \\ 2.887.10^{-5} & 6.892.10^{-6} & 5.309.10^{-6} & 4.138.10^{-4} \\ 6.892.10^{-6} & 2.123.10^{-5} & 5.826.10^{-6} & 3.246.10^{-4} \\ 5.309.10^{-6} & 5.826.10^{-6} & 6.411.10^{-5} & 8.364.10^{-4} \\ 4.138.10^{-4} & 3.246.10^{-4} & 8.364.10^{-4} & 0.994 \end{bmatrix} \quad (184)$$

From (32), $(\mu^*, \nu^*, \eta^*, \lambda^*)$ are derived as

$$\mu^* = 0.871147 \quad (185)$$
$$\nu^* = 0.871147 \quad (186)$$
$$\eta^* = 0.624426 \quad (187)$$
$$\lambda^* = 1.294837 \quad (188)$$

For the mass spectrometry data shown in Figure 7 (c), $(\mu^*, \nu^*, \eta^*, \lambda^*)$ are derived as



(a) (b) (c)

Figure 7: Mass spectrometry joint probability distribution in the case of (a) $\log Rank$ at base 4, (b) $\log Rank$ at base 2, and (c) no $\log Rank$ transformation.

$$\mu^* = 0.901208 \tag{189}$$
$$\nu^* = 0.901208 \tag{190}$$
$$\eta^* = 0.615797 \tag{191}$$
$$\lambda^* = 1.281621 \tag{192}$$