Local Density Estimation in High Dimensions

Xian Wu [1] **Moses Charikar** [1] **Vishnu Natchu** [2]

Abstract

An important question that arises in the study of high dimensional vector representations learned from data is: given a set $\mathcal{D}$ of vectors and a query q, estimate the number of points within a specified distance threshold of q. Our algorithm uses locality sensitive hashing to preprocess the data to accurately and efficiently estimate the answers to such questions via an unbiased estimator that uses importance sampling. A key innovation is the ability to maintain a small number of hash tables via preprocessing data structures and algorithms that sample from multiple buckets in each hash table. We give bounds on the space requirements and query complexity of our scheme, and demonstrate the effectiveness of our algorithm by experiments on a standard word embedding dataset.

1. Introduction

In this work, we study a basic question that arises in the study of high dimensional vector representations: given a dataset $\mathcal{D}$ of vectors and a query q, estimate the number of points within a specified distance threshold of q. Such density estimates are important building blocks in non-parametric clustering, determining the popularity of topics, search and recommendation systems, the analysis of the neighborhoods of nodes in social networks, and in outlier detection, where geometric representations of data are frequently used. Yet for high dimensional datasets, we still lack simple, practical, experimentally verified and theoretically justified solutions to tackle this question.

Our questions have been studied in the context of *spherical range counting*. One class of solution methods arising in the computational geometry literature, such as hierarchical splitting via trees, (Arya et al., 2010) have performance guarantees that depend exponentially on dimension. These are unsuitable for the higher dimensional models that machine learning methods are increasingly shifting towards e.g. word embeddings (Pennington et al., 2014; Mikolov et al., 2013) and graph embeddings (Perozzi et al., 2014; Tang et al., 2015; Cao et al., 2015; Grover & Leskovec, 2016; Yang et al., 2016; Wang et al., 2017; Hamilton et al., 2017). Over-parameterized models are oftentimes easier to train (Livni et al., 2014), and perform just as well, if not better (Zhang et al., 2016). Word embeddings is one example where rigorous evaluation has shown increased performance with higher dimensionality (Melamud et al., 2016) (Lai et al., 2016).

In this paper, we develop an estimation scheme for high dimensional datasets to count the number of elements around a query that are in a given radius of cosine similarity. Angular distance, which corresponds to Euclidean distance for data points on the unit sphere is commonly used in applications related to word and document embeddings, and image and video search (Jegou et al., 2011) (Huang et al., 2012). Brute force search requires a linear scan over the entire dataset, which is prohibitively expensive. Our approach uses indexing and search via locality sensitive hashing (LSH) functions in order to estimate the size of the neighborhood in a more efficient manner than retrieving the neighbors within the given radius of similarity.

Recent work has also explored LSH techniques for spherical range counting and related questions around density estimation for high-dimensional models. For example (Aumüller et al., 2017) generalizes nearest neighbor LSH hash functions to be sensitive to custom distance ranges. (Ahle et al., 2017) builds many different parameterized versions of the prototypical LSH hash tables and adaptively probes them for spherical range reporting. The closest works to ours in terms of solution method that we are aware of is that of (Spring & Shrivastava, 2017), giving an LSH based estimator to compute the partition function of a log-linear model, and (Charikar & Siminelakis, 2017), adapting LSH to solve a class of kernel density estimation problems. Both works produce an unbiased estimator, using LSH to implement a biased sampling scheme that lowers the variance of this estimator. However their technique leverages only one hash bucket per table, and hence requires a large number of tables for an accurate estimate. The biggest drawback to these works is the very high storage (hash tables) and query complexities – their techniques, as presented, are impractical for

[1]Stanford University, USA [2]Laserlike Inc, USA. Correspondence to: Xian Wu <xwu20@stanford.edu>.

Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

adoption.

Our approach improves upon the storage and sample complexities of previous methods using a combination of extracting information from multiple buckets per table (hence reducing table complexity) and importance sampling (hence reducing sample complexity). As we show in our experimental study on GLOVE embeddings, our estimate of the number of elements that are 60 degrees from a query q (which corresponds to synonyms and/or related words to q in the English vocabulary), achieves multiple orders of magnitude improved accuracy over competing methods, subject to reasonable and practical resource constraints. Our theoretical analysis develops a rigorous understanding of our technique and offers practitioners further insight on optimizing our solution method for their particular datasets.

2. Problem Formulation and Approach

Given a dataset $\mathcal{D}$ of vectors $v_1, \ldots v_n \in \mathbb{R}^d$ on the unit sphere, a query $q \in \mathbb{R}^d$ also on the unit sphere, and a range of angles of interest $\mathcal{A}$, for example 0-60 degrees, how many elements v in $\mathcal{D}$ are such that the angle between q and v, denoted θ_{qv}, are within range $\mathcal{A}$? We use $\mathcal{A}_q$ to denote the set of data vectors v that are within angle $\mathcal{A}$ to q (that have angular distance to query q that is in the range of interest $\mathcal{A}$). Our goal is to preprocess $\mathcal{D}$ in order to estimate the cardinality of this set, denoted $|\mathcal{A}_q|$, efficiently for any given q.

One final note is that our scheme is conceptualized using bit-wise LSH functions; functions that hash vectors to 0-1 bits, and where the hamming distance between the hash sequences of two data points captures information about their angular distance. For their simplicity, easy implementation, and high performance in practice, bit hashes such as hyperplane LSH (Charikar, 2002) are the standard hash functions used in practice for angular distance (Andoni et al., 2015). Our technique and results can be extended for other hash functions; however, we will use hamming distance and other implementation details specific to bit-wise LSH functions in this work.

2.1. Approach Overview

Our overall estimation scheme is an implementation of importance sampling. It consists of two steps, a preprocessing step that applies locality sensitive hash functions to our dataset to produce hash tables. After this preprocessing step, we sample from our hash tables to produce our final estimate.

To help guide the reader through the technical details of our implementation, we first offer an intuitive explanation of our approach. Our importance sampling scheme achieves 2 main objectives: we concentrate the elements of interest in our overall dataset into a few buckets that we can easily sample from, and we sample from these buckets to produce our estimate. In order to compensate for the concentrated sampling, we adjust the value of each sample by the inverse of the probability that the sample lands in the target buckets.

Our technique relies on the key insight that LSH functions can effectively implement both of these objectives. Using LSH functions to index our dataset ensures that for a given query q, elements that are close to q in angular distance have a comparative higher probability of hashing to q's bucket and to buckets that are of small hamming distance to q's bucket, thereby concentrating the elements of interest into certain buckets that we can selectively sample from.

Additionally, the hamming distance collision probabilities for bit-wise LSH functions are well expressed in terms of angular distance. Consider random hyperplane LSH (Charikar, 2002), where each hash vector is chosen uniformly at random from the d-dimensional unit sphere. Each hash vector r contributes one bit to the hash sequence of a data point v, based on the rule:

$$h_r(v) = \begin{cases} 0 & \text{if } r \cdot v \leq 0 \\ 1 & \text{otherwise.} \end{cases}$$

It is well-known that for any particular hamming distance i, and any data point x,

$$\mathbb{P}(d_{qx} = i | \theta_{qx}) = \binom{t}{i} \left(1 - \frac{\theta_{qx}}{\pi}\right)^{t-i} \left(\frac{\theta_{qx}}{\pi}\right)^{i}$$

where d_{qx} is the hamming distance between the hash for query q and the hash for data vector x, θ_{qx} denotes the angle between the 2 vectors, and t is the total number of bits in the hash sequence.

Thus, the choice of t affects the sensitivity of the LSH scheme – the correlation between the hamming distances of two hash sequences and the angle between the two underlying data points. Moreover, depending on the design choice for t, the set of hamming distances $\mathcal{I}$ that contains most of the probability mass for collision with elements of angular distance in range $\mathcal{A}$ is different. This is also a consideration in our sampling scheme; we want to sample from buckets of hamming distances $\mathcal{I}$ that have a high probability of containing elements that are within angle $\mathcal{A}$ of q.

Our sampling scheme picks elements over K hash tables from buckets that are at hamming distance $\mathcal{I}$ to the query, where $\mathcal{I}$ is tuned to $\mathcal{A}$. Given a sample, x, we compute the angular distance $\theta_{qx} = \cos^{-1}(q \cdot x)$. Let $p(x) = \mathbb{P}(d_{qx} \in \mathcal{I} | \theta_{qx})$, the collision probability that x lands in a bucket that is hamming distance $\mathcal{I}$ from q over the random choice of hash functions.

We define a random variable Z as a function of sample x as follows:

$$Z = \begin{cases} \frac{\sum_{k=1}^{K} C_q^k(\mathcal{I})}{K \cdot p(x)} & \text{if } \theta_{qx} \in \mathcal{A} \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

where $C_q^k(\mathcal{I})$ is the total number of elements in buckets of hamming distance $\mathcal{I}$ from q's bucket in table k.

We take S samples and construct $Z_1, Z_2, \dots Z_S$. We report $\frac{\sum_{i=1}^{S} Z_i}{S}$ as our estimate for $|\mathcal{A}_q|$.

Comparison to Related Work: Note that our problem can be viewed as kernel density estimation problem for a specific kernel function that has value 1 for pairs of points within the required angle range of interest and 0 outside. However the analysis of (Charikar & Siminelakis, 2017) does not apply to our setting because they need a scale free hash function (with collision probabilities related to the kernel value) and there is no such function for our 0-1 kernel. The work of (Spring & Shrivastava, 2017) does not make such an assumption on the hash function, but they do not give an analysis that gives meaningful bounds in our setting. As noted previously, both works only look at a single hash bucket in each hash table, leading to a high storage overhead.

2.2. Main Result

We establish the following theoretical bounds on the storage and sample complexity of our estimator in order to achieve a $(1 \pm \varepsilon)$-approximation to the true count with high probability.

Theorem 2.1 (Main Result)**.** For a given angular distance range of interest $\mathcal{A}$ and a given query q, with probability $1 - \delta$, our estimator returns a $(1 \pm \varepsilon)$-approximation to $|\mathcal{A}_q|$, the true number of elements within angle $\mathcal{A}$ to q using $O\left(\frac{1}{\varepsilon^2 \min\limits_{x \in \mathcal{A}_q} p(x)} \log(\frac{1}{\delta})\right)$ tables and $O\left(\frac{\mathbb{E}(C_q(\mathcal{I}))}{\varepsilon^2 |\mathcal{A}_q| \cdot \min\limits_{x \in \mathcal{A}_q} p(x)} \log(\frac{1}{\delta})\right)$ samples.

To help the reader digest this result, we briefly compare this statement to the sample complexity of naive random sampling. It can be shown through a standard Bernoulli-Chernoff argument that the sample complexity for random sampling is $O(\frac{n}{|\mathcal{A}_q|\varepsilon^2} \ln\left(\frac{1}{\delta}\right))$, where $\frac{n}{|\mathcal{A}_q|}$ is the inverse proportion of elements of interest in the overall population. Intuitively this says that you need to take more random samples if $|\mathcal{A}_q|$ is very small compared to n.

Our sample complexity replaces the $\frac{n}{|\mathcal{A}_q|}$ term with $\frac{\mathbb{E}(C_q(\mathcal{I}))}{|\mathcal{A}_q| \cdot \min\limits_{x \in \mathcal{A}_q} p(x)}$, where $|\mathcal{A}_q| \cdot \min\limits_{x \in \mathcal{A}_q} p(x)$ is a measure of the expected number of elements from the set of interest $\mathcal{A}_q$ that will land in hamming distance $\mathcal{I}$ to q, and $\mathbb{E}(C_q(\mathcal{I}))$ is the expected size of the overall sampling pool of elements in hamming distance $\mathcal{I}$. This ratio of expectations seems intuitive – one would expect to get such an expression if our scheme took one sample per table. Surprisingly, we achieve this same type of sample complexity bound while sampling from relatively few hash tables.

Just like random sampling, our sample complexity bound is also based on the proportion of elements of interest in hamming distance $\mathcal{I}$ to the total number of elements in hamming distance $\mathcal{I}$. However, it is easy to see that applying LSH to our dataset will increase this proportion to yield a smaller sample complexity. We choose $\mathcal{I}$ so that $\min\limits_{x \in \mathcal{A}_q} p(x)$ is high (this probability can be high even for a small set of hamming distances $\mathcal{I}$, since $p(x)$ is the cumulative probability mass of $\mathcal{I}$ successes in t trials, and binomial distributions in t concentrate in an $O(\sqrt{t})$ sized interval around the mean), and $\mathbb{E}(C_q(\mathcal{I}))$ to be small (to filter out elements that are not interesting).

There are certain tradeoffs to choosing $\mathcal{I}$. If more hamming distances are included in $\mathcal{I}$, then $\min\limits_{x \in \mathcal{A}_q} p(x)$ is higher, however, $\mathbb{E}(C_q(\mathcal{I}))$ is also larger. The optimal choice for $\mathcal{I}$ is to choose the hamming distances that substantially increase $\min\limits_{x \in \mathcal{A}_q} p(x)$ yet do not substantially increase $\mathbb{E}(C_q(\mathcal{I}))$ (so not too many uninteresting elements are infiltrating those buckets).

In the following sections, we explain our scheme further and present our experimental results.

3. Preprocessing

The preprocessing step contributes 3 key ingredients to the overall estimation scheme:

Hash Tables: Given a family of bit-wise hash functions $\mathcal{H}$, define a function family $\mathcal{G} = \{g : \mathcal{D} \to \{0,1\}^t\}$ such that $g(v) = (h_1(v), \dots h_t(v))$, where $h_j \in \mathcal{H}$. To construct K tables, we choose K functions $g_1, g_2, \dots g_K$ from $\mathcal{G}$ independently and uniformly at random. We store each $v \in \mathcal{D}$ in bucket $g_k(v)$ for $k = 1, 2 \dots K$. This step sets up the hash tables that we will sample from in our scheme.

Counts Vector: We create a counts vector, denoted $C_i^k \in \mathbb{R}^{t+1}$ for each hash address i^k for each table $k \in \{1, \dots, K\}$, where $C_i^k(d)$ is the count of the total number of items in buckets that are at hamming distance $d = 0, 1, \dots t$ away from i^k in table k.

Sampler: We create a sampler that given a separate hash address i^k for each table $k \in \{1, \dots, K\}$ and set of hamming distances $\mathcal{I}$, returns a data point uniformly at random from the union of elements that were hashed to buckets of hamming distance $\mathcal{I}$ from i^k across the K tables.

We describe in greater detail the 3 contributions of the preprocessing step. For the rest of this paper, all omitted proofs appear in Appendix C.

3.1. Hash Tables

Setting up quality hash tables to enable accurate and efficient importance sampling is vital to our scheme. Since we are importance sampling from buckets of hamming distance $\mathcal{I}$ across K tables, we need to make enough tables to guarantee unbiasedness or near-unbiasedness for our sampling-based estimator; due to the variance of the randomly generated hash functions, if we make too few tables we may not find enough elements of interest contained in those tables within hamming distance $\mathcal{I}$. We want to characterize the bias of our importance sampling scheme in relation to the contents of the buckets of our hash tables.

We let $\mathcal{B}_q^k(\mathcal{I})$ denote the set of hash buckets that are at hamming distance $\mathcal{I}$ from the hash address of query q for table k. Next, we introduce an intermediate random variable:

$$W = \frac{1}{K}\sum_{k=1}^{K}\sum_{x\in\mathcal{A}_q}\frac{\mathbb{1}(x\in\mathcal{B}_q^k(\mathcal{I}))}{p(x)} .$$

where $p(x) = \mathbb{P}(d_{qx}\in\mathcal{I}|\theta_{qx})$.

W is a random variable that represents the sum of the elements of interest $|\mathcal{A}_q|$ that are hashed to the buckets of sampling focus $\mathcal{B}_q^k(\mathcal{I})$, weighted by their probabilities $p(x)$. It is clear that once the set of hash functions is fixed, W becomes deterministic.

We first show that the random variable Z, as defined in Equation (1), is an unbiased estimator.

Lemma 3.1 (Expectation of Z)**.** *The expectation of Z over the random choice of hash functions is $|\mathcal{A}_q|$, i.e. $\mathbb{E}(Z) = |\mathcal{A}_q|$. The expectation of Z given a specific realization of hash functions, or equivalently, given W, is $\mathbb{E}(Z|W) = W$.*

As a consequence, it is immediately clear that $\mathbb{E}(W) = |\mathcal{A}_q|$. It is important to understand the implications of this lemma. In particular, the expression for $\mathbb{E}(Z|W)$ says that in a specific realization of a choice of hash functions (or a set of tables), the estimator Z is biased if $W \neq |\mathcal{A}_q|$. Therefore K is essential for helping concentrate the realized value of W around its mean.

Since in expectation, our estimator Z gives W, we want to understand how many tables K are required to ensure that W concentrates around its mean, $|\mathcal{A}_q|$. This is related to the variance of W.

We also introduce a new quantity $p(x,y) = \mathbb{P}(d_{qx}\in\mathcal{I}\cap d_{qy}\in\mathcal{I}|\theta_{qx},\theta_{qy})$, the collision probability that x and y both land in buckets that are hamming distance $\mathcal{I}$ from q over the random choice of hash functions.

Lemma 3.2 (Variance of W)**.** $\sigma^2(W) = \frac{1}{K}\sum\limits_{x,y\in\mathcal{A}_q}\left(\frac{p(x,y)}{p(x)p(y)} - 1\right)$

We want to put these pieces together to make a statement about the number of tables K we should create to guarantee low inherent bias in our estimator. We use Chebyshev's Inequality to bound W's deviation from its mean as a function of K with a constant failure probability $\frac{1}{8}$. For simplicity, we fix a constant failure probability that we will boost later by average over several sets of estimators. This analysis is without loss of generality, as the bounds can be adjusted for any desired failure probability δ. We will use this piece again when we analyze our overall estimator.

Lemma 3.3 (Bound on Number of Tables)**.** *It suffices to make $K \geq \frac{8}{\varepsilon^2 \min\limits_{x\in\mathcal{A}_q} p(x)}$ tables to guarantee that W is within ε of $|\mathcal{A}_q|$ (relatively) with probability $\frac{7}{8}$.*

Proof. Chebyshev's inequality states: $\mathbb{P}(|W - |\mathcal{A}_q|| \geq \varepsilon|\mathcal{A}_q|) \leq \frac{\sigma^2(W)}{\varepsilon^2|\mathcal{A}_q|^2}$.

Therefore, to achieve a constant failure probability $\delta = \frac{1}{8}$, it suffices to create enough tables so that

$$\sigma^2(W) = \frac{1}{K}\sum_{x,y\in\mathcal{A}_q}\left(\frac{p(x,y)}{p(x)p(y)} - 1\right) \leq \frac{\varepsilon^2|\mathcal{A}_q|^2}{8}$$

Hence K needs to be large enough so that:

$$K \geq \frac{8\sum\limits_{x,y\in\mathcal{A}_q}\left(\frac{p(x,y)}{p(x)p(y)} - 1\right)}{\varepsilon^2|\mathcal{A}_q|^2}$$

Since $p(x,y) \leq \min\{p(x),p(y)\}$, we see that it is sufficient for K to satisfy

$$K \geq \frac{8|\mathcal{A}_q|^2\left(\min_{x\in\mathcal{A}_q}\frac{1}{p(x)} - 1\right)}{\varepsilon^2|\mathcal{A}_q|^2}$$

Therefore we conclude with the following bound on K:

$$K \geq \frac{8}{\varepsilon^2 \min\limits_{x\in\mathcal{A}_q} p(x)} \tag{2}$$

□

We emphasize that the joint probability $p(x,y) \leq \min\{p(x),p(y)\}$ is a very loose worst-case bound assuming high correlation between data points. The final bound for K, Equation (2), is also a worst-case bound in the sense that it is possible that a very minuscule fraction of $x\in\mathcal{A}_q$ have small values for $p(x)$. In the experimental section of the paper, we do an empirical analysis of the inherent bias for different values of K and demonstrate that for real datasets the number of tables needed can be far fewer than what is theoretically required in the worst case scenario.

3.2. Counts Vector

Query q maps to a bucket i^k for each table $k = 1, 2 \dots K$. The preprocessing step produces an average counts vector corresponding to bucket i^k, denoted C_q^k, where $C_q^k(i)$ is the count of the total number of items in buckets that are at hamming distance $i = 0, 1, \dots t$ away from the hash address for q in table k. For the hamming distances of interest $\mathcal{I}$, we let $C_q^k(\mathcal{I}) = \sum_{d \in \mathcal{I}} C_q^k(d)$.

$C_q^k(\mathcal{I})$ is an integral part of our weighted importance sampling scheme. In Appendix A, we show how to compute these vectors efficiently.

Theorem 3.1 (`Aggregate-Counts`)**.** Given a set of K hash tables, each with 2^t hash buckets with addresses in $\{0,1\}^t$, `Aggregate-Counts` (Algorithm 1) computes, for each hash address i, the number of elements in buckets that are hamming distance $0, 1, \dots t$ away from i, in each of the K tables, in time $O(Kt^2 2^t)$.

Note that the t in our hashing scheme is the length of the hash sequence; as a general rule of thumb, for bit-wise hash functions, implementers choose $t \approx \log(n)$, so as to average out to one element per hash bucket. Therefore, the preprocessing runtime of a reasonable hashing implementation for `Aggregate-Counts` (Algorithm 1) is approximately $O(nK \log^2(n))$.

The key benefit of `Aggregate-Counts` is that it computes via a message-passing or dynamic programming strategy that is much more efficient than a naive brute-force approach that would take time $O(K2^{2t})$, or $O(Kn^2)$ if $t \approx \log(n)$.

3.3. Sampler

We create a sampler that, given a hash address i^k for each table, and a set of hamming distances $\mathcal{I}$ that we want to sample from, generates a sample uniformly at random from the union of elements that were hashed to hamming distance $\mathcal{I}$ across the K tables. For an implementation and analysis, please consult Appendix B.

Theorem 3.2 (Sampler)**.** Given a set of K hash tables, each with 2^t hash buckets with addresses in $\{0,1\}^t$, a sampling scheme consisting of a data structure and a sampling algorithm can generate a sample uniformly at random from any fixed hash table k, an element at hamming distance d to hash address i. The data structure is a counts matrix that can be precomputed in preprocessing time $O(Kt^3 2^t)$, and the sampling algorithm `Hamming-Distance-Sampler` (Algorithm 2) generates a sample in time $O(t)$.

Again, if we follow $t \approx \log(n)$, the preprocessing time comes out to roughly $O(nK \log^3(n))$. Also we expect the $O(t)$ online sample generation cost to be negligible compared to, say, the inner product computation cost for $q \cdot x$, which our method and all competing methods use. We describe the importance sampling scheme in the next section.

4. Sampling

We now analyze our sampling algorithm. Recall that our sampling scheme works in the following way. Given query q, we generate the hash for q in each of our K tables, by solving for $i^k = g_k(q)$ for $k = 1, \dots K$. Given the hash for q in each of our K tables and the set of hamming distances $\mathcal{I}$ that we want to sample from, we invoke our sampler to generate a sample from across the K tables.

Given this sample, x, we compute the angular distance $\theta_{qx} = \cos^{-1}(q \cdot x)$. Let $p(x) = \mathbb{P}(d_{qx} \in \mathcal{I} | \theta_{qx})$, the collision probability that x lands in a bucket that is hamming distance $\mathcal{I}$ from q over the random choice of hash functions; $p(x)$ is an endogenous property of an LSH function.

We score each sample as in Equation (1).

We take S samples and construct $Z_1, Z_2, \dots Z_S$. We report $\frac{\sum_{i=1}^S Z_i}{S}$ as our estimate for $|\mathcal{A}_q|$.

As an immediate consequence of Lemma 3.1, it is clear that

$$\mathbb{E}\left[\frac{\sum_{i=1}^S Z_i}{S}\right] = |\mathcal{A}_q| \,.$$

Now we analyze the variance of our estimator:

Lemma 4.1 (Variance of Estimator)**.**

$$\mathbb{E}\left[\left(\frac{\sum_{i=1}^S Z_i}{S} - |\mathcal{A}_q|\right)^2\right] \leq \frac{\mathbb{E}[Z^2]}{S} + \sigma^2(W)$$

This decomposition of the variance into the two terms indicates that the variance is coming from two sources. The first source is the variance of the samples, $\frac{\mathbb{E}[Z^2]}{S}$. If we don't take enough samples, we do not get a good estimate. The second source is the variance from the random variable W, $\sigma^2(W)$, which corresponds to the contents in the tables. As we have shown, it is crucial to create enough tables so that W is concentrated around its expectation, $|\mathcal{A}_q|$. Therefore, this second source of variance of the overall estimator comes from the variance of the hash functions that underlie table creation and composition.

The $\sigma^2(W)$ term has already been analyzed in Section 3.1, see Lemma 3.2. Now we analyze the second moment of Z.

Lemma 4.2 (Variance of Z)**.**

$$\mathbb{E}[Z^2] = \sum_{x \in \mathcal{A}_q} \sum_{y \in \mathcal{D}} \left[\frac{p(x,y)}{K \cdot p(x)^2} + \left(1 - \frac{1}{K}\right)\frac{p(y)}{p(x)}\right]$$

Now that we have all the components, we are ready to put together the final sample and storage complexities for our estimator. We want a final estimate that concentrates with at most ϵ error around its mean, $|\mathcal{A}_q|$ with probability $1-\delta$. To do this, we make several sets $1, 2, \ldots M$ of our estimator (one estimator consists of a set of K tables and S samples). We choose K and S so that the failure probability of our estimator is a constant, say $\frac{1}{4}$. Each estimator produces an estimate, call it E_m, for $m \in \{1, \ldots M\}$. We report our final estimate as the median of these estimates. This is the classic Median-of-Means technique.

Let F_m be the indicator variable indicating if the estimator E_m fails to concentrate. Clearly $\mathbb{E}(F_m) \leq \frac{1}{4}$. Moreover, $\mathbb{E}(F = \sum_{m=1}^{M} F_m) \leq \frac{M}{4}$. The probability that the median estimate is bad, $\mathbb{P}(\text{median of } E_m \text{fails}) \leq \mathbb{P}(\text{half of } E_m \text{ fails}) = \mathbb{P}(F \geq \frac{M}{2})$. By a simple Chernoff bound, we see that: $\mathbb{P}(F \geq \frac{M}{2}) \leq e^{-(2\ln 2-1)\frac{M}{4}} \leq e^{\frac{-M}{11}}$. So to satisfy a desired failure probability δ, it suffices to have $e^{\frac{-M}{11}} \leq \delta$, therefore $M \in O(\log(\frac{1}{\delta}))$.

In the rest of the section, we establish bounds on K and S so that one estimator fails with probability at most $\frac{1}{4}$. We appeal again to Chebyshev's Inequality:

$$\mathbb{P}\left(\left|\frac{\sum_{i=1}^{S} Z_i}{S} - |\mathcal{A}_q|\right| \geq \varepsilon|\mathcal{A}_q|\right) \leq \frac{\sigma^2(\frac{\sum_{i=1}^{S} Z_i}{S})}{\varepsilon^2|\mathcal{A}_q|^2}$$

In Lemma 4.1, we analyze the variance of our estimator, and show that $\sigma^2(\frac{\sum_{i=1}^{S} Z_i}{S}) \leq \frac{\mathbb{E}[Z^2]}{S} + \sigma^2(W)$. Therefore, in order so that the failure probability is less than $\frac{1}{4}$, it suffices to have $\sigma^2(\frac{\sum_{i=1}^{S} Z_i}{S}) \leq \frac{\varepsilon^2|\mathcal{A}_q|^2}{4}$, which can be obtained by letting $\frac{\mathbb{E}[Z^2]}{S} \leq \frac{\varepsilon^2|\mathcal{A}_q|^2}{8}$ and $\sigma^2(W) \leq \frac{\varepsilon^2|\mathcal{A}_q|^2}{8}$.

Focusing on the $\sigma^2(W)$ term, which depends on the number of tables K created, we show in Lemma 3.3 from Section 3.1 that it suffices to take $K \geq \frac{8}{\varepsilon^2 \min_{x\in\mathcal{A}_q} p(x)}$.

Now that we have our table complexity, we can analyze our sampling complexity S to bound $\frac{\mathbb{E}[Z^2]}{S}$.

Lemma 4.1. *Suppose* $K \geq \frac{8}{\varepsilon^2 \min_{x\in\mathcal{A}_q} p(x)}$. *Then* $S \in O\left(\frac{\mathbb{E}(C_q(\mathcal{I}))}{\varepsilon^2|\mathcal{A}_q| \cdot \min_{x\in\mathcal{A}_q} p(x)}\right)$ *suffices to achieve* $\frac{\mathbb{E}[Z^2]}{S} \leq \frac{\varepsilon^2|\mathcal{A}_q|^2}{8}$.

Proof. By Lemma 4.2 we have:

$$\frac{\mathbb{E}[Z^2]}{S} = \frac{1}{S}\sum_{x\in\mathcal{A}_q}\sum_{y\in\mathcal{D}}\left[\frac{p(x,y)}{K\cdot p(x)^2} + \left(1-\frac{1}{K}\right)\frac{p(y)}{p(x)}\right]$$

Substituting for $K \geq \frac{8}{\varepsilon^2 \min_{x\in\mathcal{A}_q} p(x)}$ gives:

$$\begin{aligned}\frac{\mathbb{E}[Z^2]}{S} &\leq \frac{1}{S}\sum_{x\in\mathcal{A}_q}\sum_{y\in\mathcal{D}}\left[\frac{\varepsilon^2 p(x,y)\min_{x\in\mathcal{A}_q} p(x)}{8p(x)^2} + \frac{p(y)}{p(x)}\right] \\ &\leq \frac{1}{S}\sum_{x\in\mathcal{A}_q}\sum_{y\in\mathcal{D}}\left[\frac{\varepsilon^2 p(x,y)}{8p(x)} + \frac{p(y)}{p(x)}\right] \\ &\leq \frac{1}{S}\sum_{x\in\mathcal{A}_q}\sum_{y\in\mathcal{D}}\left[(1+\varepsilon^2)\frac{p(y)}{p(x)}\right]\end{aligned}$$

In order to guarantee $\frac{\mathbb{E}[Z^2]}{S} \leq \frac{\varepsilon^2|\mathcal{A}_q|^2}{8}$, we need:

$$\begin{aligned}S &\geq \frac{\sum_{x\in\mathcal{A}_q}\sum_{y\in\mathcal{D}}\left[(1+\varepsilon^2)\frac{p(y)}{p(x)}\right]}{\varepsilon^2|\mathcal{A}_q|^2} \\ &= (1+\varepsilon^2)\frac{\sum_{x\in\mathcal{A}_q}\frac{1}{p(x)}\sum_{y\in\mathcal{D}} p(y)}{\varepsilon^2|\mathcal{A}_q|^2} \\ &= (1+\frac{1}{\varepsilon^2})\frac{\sum_{x\in\mathcal{A}_q}\frac{1}{p(x)}\mathbb{E}(C_q(\mathcal{I}))}{|\mathcal{A}_q|^2}\end{aligned}$$

Therefore, we conclude that

$$S \in O\left(\frac{\mathbb{E}(C_q(\mathcal{I}))}{\varepsilon^2|\mathcal{A}_q| \cdot \min_{x\in\mathcal{A}_q} p(x)}\right)$$

is sufficient. □

Putting together Lemmas 3.3 and 4.1 with the median of means strategy yields our main result, Theorem 2.1.

In the rest of this paper we discuss the results of our experiments on real datasets.

5. Experiments

We describe our experiments using the GLOVE dataset. We use the set of 400,000 pre-trained 50-dimensional word embedding vectors trained from Wikipedia 2014 + Gigaword 5, provided by (Pennington et al., 2014). We normalize the embeddings, as is standard in many word embedding applications (Sugawara et al., 2016) We choose 3 query words with different neighborhood profiles: "venice", "cake", "book". Venice has the smallest neighborhood, with 206 elements with angular distance less than 60 degrees, cake has a medium sized neighborhood with about 698 elements, book has the largest neighborhood with 1275 elements. The histogram for these 3 queries are shown in Figure 1.

We also choose our angle range of interest, $\mathcal{A}$, to be 0-60 degrees. A search through our dataset gave "florence", "cannes", "rome" as representative elements that are 40-50

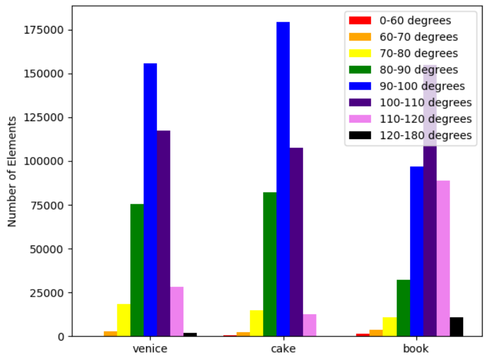

Figure 1: The statistical profiles of angular distance between our 3 queries and the rest of the dataset $\mathcal{D}$. "venice" has the smallest neighborhood with only about 206 points in the 0-60 range, followed by "cake" with 698, followed by "book" with 1275. We plot the number of elements at different intervals of cosine similarity from the 3 queries. Notice that most (more than 370,000 out of the total 400,000) embeddings in the dataset fall at about 60-120 degrees to the query. The leftmost blue bin that represents the number of elements between 0-60 degrees to our 3 queries are barely noticeable.

degrees from "venice", and "renaissance", "milan", "tuscany", "italy" in the 50-60 degree range. Terms such as "cheesecake", "desserts", "ganache", and "bakes" appear in the 40-50 degree annulus around "cake", while terms such as "fruitcake", "cupcake", "confections", "poundcake", and "eggs" appear in the 50-60 degree histogram. For "book": "character", "chronicles", "paperback", "authors", and "text" are in the 40-50 degree range while "bestseller", "protagonist", "publishers", "booklet", "publishes", "editing", "monograph", and "chapter" are in the 50-60 degree range. This particular experiment shows that while elements in the 40-50 degree range are extremely related, words in the 50-60 degree range are also relevant, and so we fix $\mathcal{A}$ to be 0-60 degrees in all of our experiments. We also fix $t = 20$ in all of our experiments, since we have 400,000 embeddings in total and $20 \approx \log_2(400,000)$.

As Table 1 illustrates, the biggest challenge for this estimation problem is the fact that the count of the number of elements within 0-60 degrees is dwarfed by the number of elements 60-120 degrees away from the queries. This issue makes locality sensitive techniques necessary for efficient search and retrieval in high dimensions.

Table 1: Statistics of Queries

QUERY	# WITHIN 60 DEGREES	% OF POPULATION
VENICE	206	.0515
CAKE	698	.1745
BOOK	1275	.31875

As we have previously mentioned in section 3.1, the number of tables K theoretically required for (near) unbiased estimation relies on a worst-case variance bound; real-world data do not necessarily exhibit worst-case behavior. In our studies of our 3 queries see Figures 2, the inherent bias of our estimator decreases as we increase the sampling hamming threshold. This is as expected, using a larger range of hamming distances helps concentrate the count of the elements of interest $\mathcal{A}_q$ that fall into the specified range of hamming distances around the mean, which means that a smaller K is required to achieve small bias.

Moreover, the empirical bias of our estimator at hamming threshold 5 is around 5% for 20 hash tables, with very little improvement with 40 hash tables. This is consistent with our 3 queries. With this in mind, we compare our estimator against the benchmark estimator introduced by (Spring & Shrivastava, 2017). Though their work originally intended to solve a different problem, their technique can solve our problem by adapting the weight function appropriately. The key differences between their work and ours is that they only probe the 0 hamming distance bucket in each table, similar to the classic LSH literature, and instead of sampling, they simply enumerate the elements in the hamming distance 0 bucket for each table. For higher values of K, which our experiments demonstrate that their estimator needs in order to get good results, enumeration might not be so efficient.

In Figure 3, we compare (Spring & Shrivastava, 2017)'s technique of enumerating and importance-weighting hamming distance 0 elements to our technique of importance sampling from different hamming thresholds. Our experiments use random hyperplane LSH and we report relative error averaged over 25 trials, where in each trial we generate a new set of K tables. Panel (b) experiments with (Spring & Shrivastava, 2017)'s technique for the 3 queries, with different choices of K, the number of tables. Our results show that even for $K = 40$ tables, the relative error of their technique can still be higher than 50%, particularly for queries with small neighborhoods such as "venice". For "venice" the increase in table allocation from 20 to 40 made a very small difference to the overall estimation error. "book" and "cake" fared better at 40 tables, however, the error was still around 25 %, while our estimator (panel a) estimated to within about 10% error using only 20 tables.

Panel (a) of Figure 3 shows that utilizing any hamming threshold greater than 0 gives superior estimation performance to staying only within the 0 hamming distance bucket. In this experiment, we fix our sampling budget to 1000 samples and the table budget to 20 tables. The hamming distance 0 error reported in this figure uses enumeration; all other hamming thresholds use the 1000 sampling budget. In our experiments for the 3 queries, one can expect about 80 points in total in the hamming distance 0 buckets across 20 tables. In this experiment, our technique uses 1000 samples vs 80 points, however, this (somewhat negligible in today's computing infrastructure) sample complexity trades off against a large improvement in precision, as well as a much lower storage cost in the number of tables K.

Finally, we note that panel (a) of Figure 3 shows the smallest

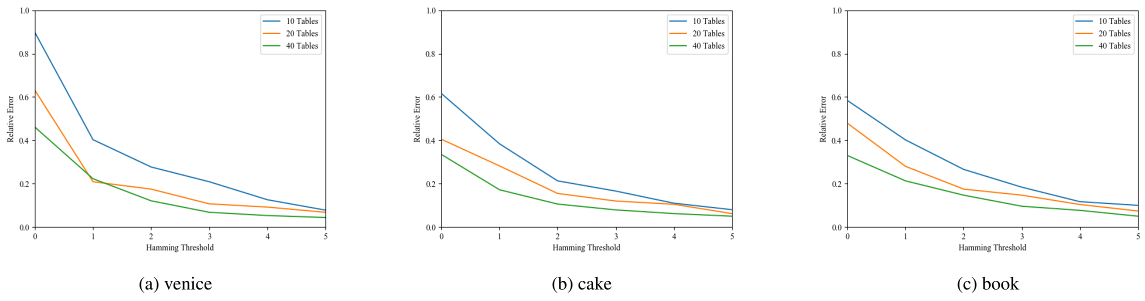

Figure 2: The empirical bias for different values of K for queries "venice", "cake" and "book". For each hamming threshold, the relative bias is averaged over 50 sets of K tables, using random hyperplane hash as the LSH function. The bias decreases as the hamming threshold increases and the bias decreases with more hash tables, keeping hamming threshold fixed. The bias does not drop significantly from 20 to 40 hash tables. At 20 hash tables, the bias at hamming threshold 5 is around 5%. This demonstrates that for real datasets the number of tables needed can be far fewer than what is theoretically required in the worst case scenario.

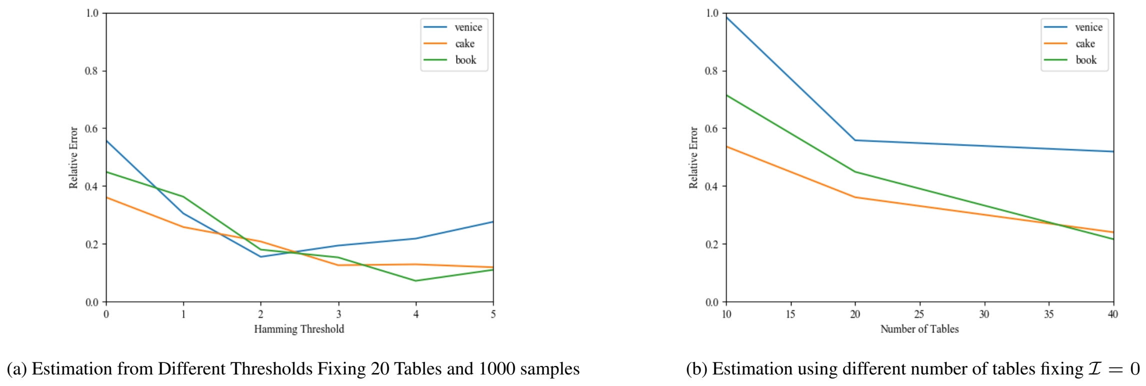

Figure 3: Comparison of our estimator against the benchmark LSH estimator adapted from ideas introduced in (Spring & Shrivastava, 2017). In our experiments, panel (b), even after 40 hash tables, the error remained above 20%, and above 50% for queries with very small neighborhoods, such as "venice". In contrast, panel (a) illustrates the relative accuracy for our estimator. We fix 20 hash tables and takes 1000 samples from different hamming thresholds. Note that for queries like "venice", which has a very small neighborhood, taking 1000 samples at hamming threshold 5 performed worse than at hamming threshold 3; this is likely because 1000 samples was too few for hamming threshold 5 – the ratio of total elements to elements of interest in hamming threshold 5 is high.

error for "venice" at hamming threshold 3. This is related to the characteristics of this query and the sampling budget. We see in this example that for "venice", which is a fairly isolated data point compared to the other 2 queries, going to further hamming distances actually hurts the quality of the estimate because we actually dilute the proportion of interesting elements. Using higher thresholds typically requires more samples, as shown in Figure 4. However, higher thresholds typically lowers the inherent bias in the importance sampling scheme, as demonstrated in Figure 2. Implementers should consider this tradeoff in their algorithmic design choices.

6. Discussion

Given the case study of our estimator achieving the smallest estimation error for "venice" at hamming threshold 3, whereas for the more popular queries "cake" and "book" performance improves steadily at higher hamming thresholds, it would be interesting to, from the practitioner's point of view, understand what is the best hamming threshold to sample from, and given a hamming threshold, how many samples should be taken for a quality estimate. The optimal

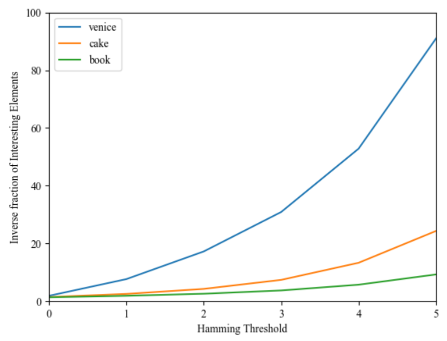

Figure 4: The inverse proportion of relative elements of interest in the overall subsampling pool for various hamming thresholds, averaged over 50 trials of sets of 20 hash tables. This figure shows that using higher thresholds typically requires more samples. However, higher thresholds typically have less inherent bias in the importance sampling scheme, as demonstrated in Figure 2. Implementers should consider this tradeoff in their algorithmic design choices.

sample complexity is data-dependent, and cannot be known without a sense of $|\mathcal{A}_q|$, the very quantity we aim to estimate. But instead of fixing the sample complexity up-front, is there a way we can iteratively, in an on-line fashion, determine whether we should keep sampling or stop, based on a current belief of $|\mathcal{A}_q|$?

Acknowledgements

This work was initiated while the authors were visiting Laserlike, Inc. Xian Wu was supported by a Harold Thomas Hahn Jr. Fellowship from the Department of Management Science and Engineering at Stanford University. Moses Charikar was supported by NSF grant CCF-1617577 and a Simons Investigator Award.

References

Ahle, T. D., Aumüller, M., and Pagh, R. Parameter-free locality sensitive hashing for spherical range reporting. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 239–256. Society for Industrial and Applied Mathematics, 2017.

Andoni, A., Indyk, P., Laarhoven, T., Razenshteyn, I., and Schmidt, L. Practical and optimal lsh for angular distance. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 1225–1233. 2015.

Arya, S., da Fonseca, G. D., and Mount, D. M. A unified approach to approximate proximity searching. *European Symposium on Algorithms*, 2010.

Aumüller, M., Christiani, T., Pagh, R., and Silvestri, F. Distance-sensitive hashing. 03 2017. URL https://arxiv.org/abs/1703.07867.

Cao, S., Lu, W., and Xu, Q. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pp. 891–900. ACM, 2015.

Charikar, M. and Siminelakis, P. Hashing-based-estimators for kernel density in high dimensions. *IEEE Symposium on Foundations of Computer Science*, 2017.

Charikar, M. S. Similarity estimation techniques from rounding algorithms. In *Proceedings of the Thiry-fourth Annual ACM Symposium on Theory of Computing*, STOC '02, pp. 380–388, New York, NY, USA, 2002. ACM. URL http://doi.acm.org/10.1145/509907.509965.

Grover, A. and Leskovec, J. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864. ACM, 2016.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1025–1035, 2017.

Huang, E. H., Socher, R., Manning, C. D., and Ng, A. Y. Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pp. 873–882, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Jegou, H., Douze, M., and Schmid, C. Product quantization for nearest neighbor search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33, Jan 2011.

Lai, S., Liu, K., He, S., and Zhao, J. How to generate a good word embedding. *IEEE Intelligent Systems*, 31, 2016.

Livni, R., Shalev-Shwartz, S., and Shamir, O. On the computational efficiency of training neural networks. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 855–863. Curran Associates, Inc., 2014.

Melamud, O., McClosky, D., Patwardhan, S., and Bansal, M. The role of context types and dimensionality in learning word embeddings. 01 2016. URL https://arxiv.org/abs/1601.00893.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. 01 2013. URL https://arxiv.org/abs/1301.3781.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, 2014.

Perozzi, B., Al-Rfou, R., and Skiena, S. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710. ACM, 2014.

Spring, R. and Shrivastava, A. A new unbiased and efficient class of lsh-based samplers and estimators for partition function computation in log-linear models. 03 2017. URL https://arxiv.org/abs/1703.05160.

Sugawara, K., Kobayashi, H., and Iwasaki, M. On approximately searching for similar word embeddings. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., and Mei, Q. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pp. 1067–1077. International World Wide Web Conferences Steering Committee, 2015.

Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., and Yang, S. Community preserving network embedding. In *AAAI*, pp. 203–209, 2017.

Yang, Z., Cohen, W., and Salakhudinov, R. Revisiting semi-supervised learning with graph embeddings. In *International Conference on Machine Learning*, pp. 40–48, 2016.

Zhang, C., Bengio, S., Hardt, M., Recht, B., and Vinyals, O. Understanding deep learning requires rethinking generalization. 11 2016. URL https://arxiv.org/abs/1611.03530.