RHash: Robust Hashing via ℓ_∞-norm Distortion

Amirali Aghazadeh[1], **Andrew Lan**[2], **Anshumali Shrivastava**[1], **Richard Baraniuk**[1]
[1]Rice University, Houston, TX, USA
[2]Princeton University, Princeton, NJ, USA
{amirali,anshumali,richb}@rice.edu, andrew.lan@princeton.edu

Abstract

Hashing is an important tool in large-scale machine learning. Unfortunately, current data-dependent hashing algorithms are not robust to small perturbations of the data points, which degrades the performance of nearest neighbor (NN) search. The culprit is the minimization of the ℓ_2-norm, average distortion among pairs of points to find the hash function. Inspired by recent progress in robust optimization, we develop a novel hashing algorithm, dubbed RHash, that instead minimizes the ℓ_∞-norm, worst-case distortion among pairs of points. We develop practical and efficient implementations of RHash that couple the alternating direction method of multipliers (ADMM) framework with column generation to scale well to large datasets. A range of experimental evaluations demonstrate the superiority of RHash over ten state-of-the-art binary hashing schemes. In particular, we show that RHash achieves the same retrieval performance as the state-of-the-art algorithms in terms of average precision while using up to 60% fewer bits.

1 Introduction

Hashing, one of the primitive operations in large-scale machine learning systems, seeks a low-dimensional binary embedding of a high-dimensional dataset. Such a binary embedding can increase the computational efficiency of a variety of tasks, including nearest neighbor (NN) search.

The embedding of high-dimensional data into short binary codes naturally distorts the set of top NNs of each data point. To minimize this distortion, a range of hashing schemes have been developed. The simplest, *random projection*, projects the data onto a lower-dimensional random subspace and then quantizes to binary values. Random projection belongs to the family of probabilistic, locality sensitive hashing (LSH) algorithms that guarantee the preservation of the set of NN points with small distortion [Datar *et al.*, 2004]. The cost, however, is that LSH requires an impractically high number of bits to index real-world, large-scale datasets [Lv *et al.*, 2007]. In response, *data-dependent* binary hashing algorithms have been developed that require significantly fewer bits; see [Wang *et al.*, 2014] for a survey.

Data-dependent hashing algorithms learn a hash function by minimizing a measure of distortion between certain pairs of data points [Weiss *et al.*, 2009; 2012]. For the task of NN search, it is sufficient to control the pairwise distortion among the NNs of each point plus a few additional points [Kulis and Darrell, 2009]. Given a pair of data points $(\mathbf{x}_i, \mathbf{x}_j)$ in the ambient space and their corresponding embedded binary codes $(\mathbf{h}_i, \mathbf{h}_j)$ in the Hamming space, define the *pairwise distortion function* $\delta_{ij} = L(\mathbf{x}_i, \mathbf{x}_j; \mathbf{h}_i, \mathbf{h}_j)$, where $L(\cdot)$ is a chosen *loss function*. The lion's share of data-dependent hashing algorithms use an ℓ_2-norm loss function, which minimizes the *average* of the square of pairwise distortions $\|\boldsymbol{\delta}\|_2^2 = \sum_{(i,j)\in\Omega} \delta_{ij}^2$ over a set of selected pairs of points Ω [Jiang and Li, 2015; Kong and Li, 2012; Kulis and Darrell, 2009; Liu *et al.*, 2011; Weiss *et al.*, 2009; 2012].

Unfortunately, the loss functions employed by the hashing community to date, including the ℓ_2-norm loss, are not *robust* to perturbations of the data points. Indeed, the ℓ_2-norm average distortion is well-known to be sensitive to small disturbances and noise [Chatterjee and Hadi, 2009]. In this paper, we argue that alternative norms, in particular the ℓ_∞-norm (which measures worst-case distortion) [Du and Pardalos, 2013; Hartley and Schaffalitzky, 2004; Xu *et al.*, 2009], are more robust to small perturbations and thus more appropriate for designing data-dependent hash functions.

To elucidate the strong effect of the loss function on the robustness of a hashing function, we compare the hash functions learned using the ℓ_2- and ℓ_∞-norm losses on a subset of the *MNIST* dataset of handwritten digit images [LeCun and Cortes, 1998]. For visualization purposes, we project the data points onto the first two principal components and compute the optimal 1-dimensional binary embeddings generated by minimizing the ℓ_2-norm distortion and the ℓ_∞-norm distortion $\|\boldsymbol{\delta}\|_\infty = \max_{(i,j)\in\Omega} |\delta_{ij}|$ with $|\Omega| = 50$. Figure 1(a) shows the ℓ_2- and ℓ_∞-optimal embeddings that we learned from a grid search. Figure 1(b) summarizes an experiment where we add a small amount of white Gaussian noise to each data point before computing the optimal embeddings. We see that the ℓ_2-optimal embedding is increasingly less robust than the ℓ_∞-optimal embedding as the amount of noise grows.

The robustness of the ℓ_∞-optimal embedding translates directly to improved NN preservation. Continuing the *MNIST* example from above but with no added noise, Figure 1(c) il-

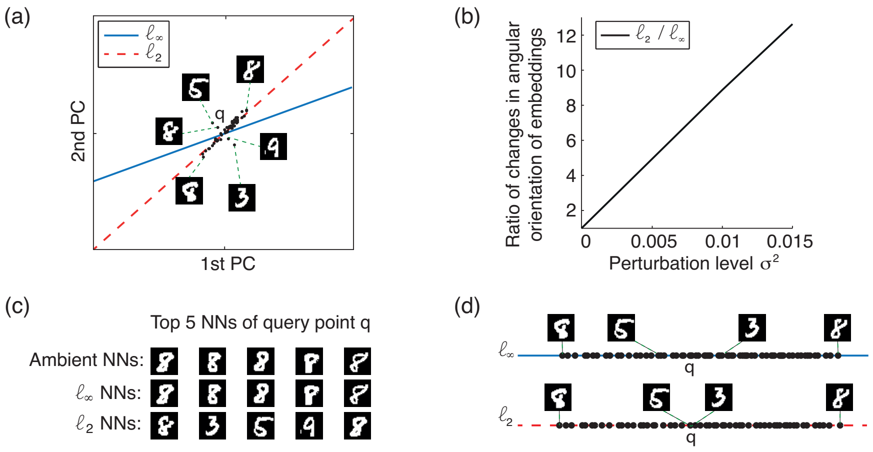

Figure 1: Comparison of the robustness and nearest neighbor (NN) preservation performance of embeddings based on minimizing the ℓ_2-norm (average distortion) vs. the ℓ_∞-norm (worst-case distortion) on a subset of the *MNIST* handwritten digit dataset projected onto its first two principal components. The subset consists of the 50 NN (in the 2-dimensional ambient space) of the centroid q of the cluster of "8" digits. (a) Optimal embeddings for both distortion measures computed using a grid search over the orientation of the line representing the embedding. (b) Robustness of the embedding orientations to the addition of a small amount of white Gaussian noise to each data point. This plot of the mean square error of the orientation of the ℓ_2-optimal embedding divided by the mean square error of the orientation of the ℓ_∞-optimal embedding indicates that the latter embedding is significantly more robust to perturbations in the data points. (c) Comparison of the top-5 NNs of the query point q obtained in the ambient space using the ℓ_∞- and ℓ_2-optimal embeddings (no added noise). (d) Projections of the data points onto the ℓ_∞- and ℓ_2-optimal embeddings (no added noise).

lustrates that the top-5 NNs of the query point q are severely damaged by the ℓ_2-optimal embedding but are left unscathed by the ℓ_∞-optimal embedding. Figure 1(d) provides some insight into the reason. The ℓ_2-norm (average distortion) optimal embedding does not preserve some of the distances between the digits "3", "5", and "9" and the digit "8" and thus projects some of them into the same neighborhood close to q. In contrast, the ℓ_∞-norm (worst-case distortion) optimal embedding preserves these *harder* distances and thus preserves the set of original NNs of q with minimal distortion.

Inspired by the robustness of the ℓ_∞-norm distortion [Du and Pardalos, 2013; Hartley and Schaffalitzky, 2004; Xu *et al.*, 2009], we propose *Robust Hashing* (RHash), the first hashing algorithm that learns robust binary hash codes by minimizing the ℓ_∞-norm of the pairwise distances vector $\|\boldsymbol{\delta}\|_\infty = \max_{(i,j)\in\Omega} |d_H(\mathbf{h}_i, \mathbf{h}_j) - d(\mathbf{x}_i, \mathbf{x}_j)|$, where d and d_H denote the Euclidean and Hamming distances, respectively. The robustness of ℓ_∞-norm distortion stems from its dependence on only the pairwise distortions that are hardest to preserve. As long as the hardest pairwise distortions are not affected, small perturbations do not alter the outcome of the embedding. In other words, in the NN retrieval task, where we only care about the distortions of NN pairs, the hardest pair to preserve is the deal-breaker. We emphasize that the worst-case distorted NN pair (under an embedding) is the hardest NN pair to be preserved and not the farthest pair nor a pair of outliers.

1.1 Contributions

We make four distinct contributions in this paper. First, conceptually, we advocate minimizing the more robust worst-case ℓ_∞-norm distortion measure rather than the more prosaic ℓ_2-norm distortion.

Second, algorithmically, since ℓ_∞-norm minimization problems are computationally challenging, especially for large datasets, we develop two accelerated and scalable algorithms to find the optimal robust embedding. The first, *Robust Hashing* (RHash), is based on the alternating direction method of multipliers (ADMM) framework [Boyd *et al.*, 2011]. The second, *RHash-CG*, is based on an accelerated greedy extension of the RHash algorithm using the concept of *column generation* [Dantzig and Wolfe, 1960]. RHash-CG can rapidly learn hashing functions from large-scale data sets that require the preservation of *billions* of pairwise distances.

Third, theoretically, since current data-dependent hashing algorithms do not offer any probabilistic guarantees in terms of preserving NNs, we develop new theory to prove that, under natural assumptions regarding the data distribution and with a notion of hardness of NN search (dubbed the *k-order gap*), RHash preserves the top NNs with high probability.

Fourth, experimentally, we demonstrate the superior performance of RHash over ten state-of-the-art binary hashing algorithms using an exhaustive set of experimental evaluations involving six diverse datasets and three different performance metrics (distance preservation, Hamming distance ranking, and Kendall's τ ranking performance). In particular, we show that RHash achieves the same retrieval performance as state-of-the-art algorithms in terms of average precision *while using up to* 60% *fewer bits.* Our experiments clearly show the advantages of the ℓ_∞-norm formulation compared to the more classical ℓ_2-norm formulation that underlies many hashing algorithms [Kong and Li, 2012; Kulis and Darrell, 2009]. Our formulation also outperforms recently developed techniques that assume more structure in their hash functions [Heo *et al.*, 2012; Yu *et al.*, 2014].

We note that the ℓ_∞-norm has been exploited in the hashing literature before; however, these works are in sharp contrast to the RHash objective and algorithm. For example, antisparse coding [Jégou *et al.*, 2012] uses the ℓ_∞-norm to directly learn binary codes and bypass quantization. More recently, the ℓ_∞-norm has been used to train a deep *supervised* hashing algorithm [Wang *et al.*, 2016]. In contrast, we impose the ℓ_∞-norm on the *NNs of the dataset* in order to learn a robust *unsupervised* hashing algorithm, which is completely different from the above approaches.

2 Robust Hashing via ℓ_∞-norm Distortion

In this section, we formulate the RHash algorithm. We pick a simple formulation for data dependent binary hash function that embeds a data point $\mathbf{x} \in \mathbb{R}^N$ into the low-dimensional Hamming space $\mathcal{H} = \{0,1\}^M$ by first multiplying it by an *embedding matrix* $\mathbf{W} \in \mathbb{R}^{M \times N}$ and then quantizing the entries of the product $\mathbf{Wx}$ to binary values: $h(\mathbf{Wx}) = (1 + \text{sgn}(\mathbf{Wx}))/2$. The function $\text{sgn}(\cdot)$ operates element-wise on the entries of $\mathbf{Wx}$, transforming the real-valued vector $\mathbf{Wx}$ into a set of binary codes depending on the sign of the entries in $\mathbf{Wx}$.

2.1 Problem Formulation

Consider the design of a binary embedding f that maps Q high-dimensional data vectors $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_Q\}$ in the ambient space $\mathbb{R}^N$ into low-dimensional binary codes $\mathcal{H} = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_Q\}$ in the Hamming space with $\mathbf{h}_i \in \{0,1\}^M$, where $\mathbf{h}_i = f(\mathbf{x}_i)$, $i = 1, \ldots, Q$, and $M \ll N$. Define the distortion of the embedding by

$$\delta = \inf_{\lambda>0} \sup_{(i,j)\in\Omega} |\lambda d_H(\mathbf{h}_i, \mathbf{h}_j) - d(\mathbf{x}_i, \mathbf{x}_j)|, \qquad (1)$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ denotes the Euclidean distance between the data points $\mathbf{x}_i, \mathbf{x}_j$, $d_H(\mathbf{h}_i, \mathbf{h}_j)$ denotes the Hamming distance between the binary codes $\mathbf{h}_i$ and $\mathbf{h}_j$, Ω indexes the set containing a selected pair of points (more details in section 3), and λ is a positive scaling variable. The distortion δ measures the worst-case deviation from perfect distance preservation [Plan and Vershynin, 2014]. Define the *secant set* $\mathcal{S}(\mathcal{X})$ as $\mathcal{S}(\mathcal{X}) = \{\mathbf{x}_i - \mathbf{x}_j : (i,j) \in \Omega\}$, i.e., the set of pairwise difference vectors indexed by Ω. Let $|\mathcal{S}(\mathcal{X})| = |\Omega|$ denote the size of the secant set [Hegde *et al.*, 2015].

RHash minimizes the worst-case distortion parameter δ via the following optimization:

$$\underset{\mathbf{W},\lambda>0}{\text{minimize}} \sup_{(i,j)\in\Omega} \left| \lambda \|h(\mathbf{Wx}_i) - h(\mathbf{Wx}_j)\|_2^2 - \|\mathbf{x}_i - \mathbf{x}_j\|_2 \right|. \qquad (2)$$

Since the squared ℓ_2-distance between a pair of binary codes is equivalent to their Hamming distance up to a scaling factor that can be absorbed into λ, we can rewrite the above optimization as

$$(\text{P}^*) \quad \underset{\mathbf{W},\lambda>0}{\text{minimize}} \ \|\lambda \mathbf{v}'(\mathbf{W}) - \mathbf{c}\|_\infty, \qquad (3)$$

where $\mathbf{v}' \in \mathbb{N}^{|\Omega|}$ is a vector containing the pairwise Hamming distances between the embedded data vectors $\|h(\mathbf{x}_i) - h(\mathbf{x}_j)\|_2^2$, and $\mathbf{c}$ is a vector containing the pairwise ℓ_2-distances between the original data vectors. Intuitively, the ℓ_∞-norm objective optimizes the *worst-case* distortion among NN's distances.

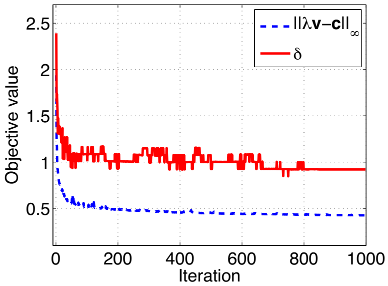

Figure 2: Empirical convergence behavior of the RHash algorithm. Both the maximum distortion parameter δ and the loss function $||\lambda\mathbf{v} - \mathbf{c}||_\infty$ that approximates δ decrease and converge as the number of iterations increases. We see that the RHash loss function closely matches the behavior of the actual distortion parameter in each iteration.

Problem (P*) has a combinatorial computational complexity of $\mathcal{O}(Q^{2M})$. Following the standard approach [Kulis and Darrell, 2009], we approximate the hash function $h(\cdot)$ by the sigmoid function $\sigma(x) = (1+e^{-x})^{-1}$. This enables us to approximate (P*) by the following optimization with a smooth objective:

$$(\text{P}) \quad \underset{\mathbf{W},\lambda>0}{\text{minimize}} \ \|\lambda \mathbf{v}(\mathbf{W}) - \mathbf{c}\|_\infty, \qquad (4)$$

where $\mathbf{v} \in \mathbb{R}_+^{|\Omega|}$ is a vector containing the pairwise ℓ_2 distances between the embedded data vectors after the sigmoid relaxation $\left\|(1 + e^{-\mathbf{Wx}_i})^{-1} - (1 + e^{-\mathbf{Wx}_j})^{-1}\right\|_2^2$. The sigmoid function operates element-wise on $\mathbf{Wx}_i$. In practice we use a more general definition of the sigmoid function, defined as $\sigma_\alpha(x) = (1 + e^{-\alpha x})^{-1}$, where α is the *rate* parameter that controls how closely it approximates the non-smooth function $h(\cdot)$. We formally characterize the approximation quality in Lem. 2 of the appendix A.

2.2 The RHash Algorithm

We now outline the steps that RHash takes to solve the optimization (P). We apply the alternating direction method of multipliers (ADMM) framework [Boyd *et al.*, 2011] to construct an efficient algorithm. We introduce an auxiliary variable $\mathbf{u}$ to arrive at the equivalent problem:

$$\underset{\mathbf{W},\mathbf{u},\lambda>0}{\text{minimize}} \ \|\mathbf{u}\|_\infty \quad \text{subject to} \quad \mathbf{u} = \lambda\mathbf{v}(\mathbf{W}) - \mathbf{c}. \qquad (5)$$

The augmented Lagrangian form of this problem can be written as $\underset{\mathbf{W},\mathbf{u},\lambda>0}{\text{minimize}} \ \|\mathbf{u}\|_\infty + \frac{\rho}{2}\|\mathbf{u} - \lambda\mathbf{v}(\mathbf{W}) + \mathbf{c} + \mathbf{y}\|_2^2$, where ρ is the scaling parameter in ADMM and $\mathbf{y} \in \mathbb{R}^{|\Omega|}$ is the Lagrange multiplier vector.

The RHash algorithm proceeds as follows. First, the variables $\mathbf{W}$, λ, $\mathbf{u}$, and Lagrange multipliers $\mathbf{y}$ are initialized randomly. Then, at each iteration, we optimize over each of the variables $\mathbf{W}$, $\mathbf{u}$, and λ while holding the other variables fixed.

More specifically, in iteration ℓ, we perform the following four steps until convergence:

- *Optimize over* $\mathbf{W}$ via $\mathbf{W}^{(\ell+1)} \leftarrow \arg\min_{\mathbf{W}} \frac{1}{2} \sum_{(i,j)\in\Omega} (u_{ij}^{(\ell)} - \lambda^{(\ell)} \| \frac{1}{1+e^{-\mathbf{W}\mathbf{x}_i}} - \frac{1}{1+e^{-\mathbf{W}\mathbf{x}_j}} \|_2^2 + \|\mathbf{x}_i - \mathbf{x}_j\|_2^2 + y_{ij}^{(\ell)})^2$, where $\lambda^{(\ell)}$ denotes the value of λ in the ℓ^{th} iteration. We also use $u_{ij}^{(\ell)}$ and $y_{ij}^{(\ell)}$ to denote the entries in $\mathbf{u}^{(\ell)}$ and $\mathbf{y}^{(\ell)}$ that correspond to the pair $(\mathbf{x}_i, \mathbf{x}_j)$ in the dataset $\mathcal{X}$. We show in our experiments below that using the accelerated first-order gradient descent algorithm [Nesterov, 2007] to solve this subproblem results in good convergence performance (see Figure 2).
- *Optimize over* $\mathbf{u}$ while holding the other variables fixed; this corresponds to solving the proximal problem of the ℓ_∞-norm $\mathbf{u}^{(\ell+1)} \leftarrow \arg\min_{\mathbf{u}} \|\mathbf{u}\|_\infty + \frac{\rho}{2} \|\mathbf{u} - \lambda^{(\ell)} \mathbf{v}^{(\ell+1)} + \mathbf{c} + \mathbf{y}^{(\ell)}\|_2^2$. We use the low-cost algorithm described in [Studer *et al.*, 2014] to perform the proximal operator update.
- *Optimize over* λ while holding the other variables fixed; this corresponds to a positive least squares problem, where λ is updated as $\lambda^{(\ell+1)} \leftarrow \arg\min_{\lambda>0} \frac{1}{2} \|\mathbf{u}^{(\ell+1)} - \lambda \mathbf{v}^{(\ell+1)} + \mathbf{c} + \mathbf{y}^{(\ell)}\|_2^2$. We perform this update using the non-negative least squares algorithm.
- *Update* $\mathbf{y}$ via $\mathbf{y}^{(\ell+1)} \leftarrow \mathbf{y}^{(\ell)} + \eta(\mathbf{u}^{(\ell+1)} - \lambda^{(\ell+1)} \mathbf{v}^{(\ell+1)} + \mathbf{c})$, where the parameter η controls the dual update step size.

2.3 Accelerated RHash for Large-scale Datasets

The ADMM-based RHash algorithm is efficient for small-scale datasets (e.g., for secant sets of size $|\mathcal{S}(\mathcal{X})| < 5000$ or so). However, the memory requirement of RHash is quadratic in $|\mathcal{X}|$, which could be problematic for applications involving large numbers of data points and secants. In response, we develop an algorithm that approximately solves (P) while scaling very well to large-scale problems. The key idea comes from classical results in optimization theory related to *column generation* (CG) [Dantzig and Wolfe, 1960; Hegde *et al.*, 2015].

The optimization (5) is an ℓ_∞-norm minimization problem with an equality constraint on each secant. The Karush-Kuhn-Tucker (KKT) condition for this problem states that, if strong duality holds, then the optimal solution is entirely specified by a (typically very small) portion of the constraints. Intuitively, the secants corresponding to these constraints are the pairwise distances of kNNs that are harder to preserve in the low-dimensional Hamming space. We call this set of secants the *active* set. In order to solve (P), it suffices to find the active secants and solve the RHash optimization with that much smaller number of active constraints. To leverage the idea of the active set, we iteratively run RHash on a small subset of all of the secants that violate the distance preservation, as detailed below:

- Solve (P) with a small random subset $\mathcal{S}_0$ of all of the secants $\mathcal{S}(\mathcal{X})$ using RHash to obtain the initial estimates $\widehat{\mathbf{W}}$, $\hat{\delta}$, and $\hat{\lambda}$ of the parameters. Identify the active set $\mathcal{S}_{\text{a}}$. Fix $\lambda = \hat{\lambda}$ for the rest of the algorithm.
- Randomly select a new subset $\mathcal{S}_{\text{v}} \subset \mathcal{S}$ of secants that violate the distance preservation condition using the current estimates of $\widehat{\mathbf{W}}$, $\hat{\delta}$, and $\hat{\lambda}$. Then, form the augmented secant set $\mathcal{S}_{\text{aug}} = \mathcal{S}_{\text{a}} \cup \mathcal{S}_{\text{v}}$.
- Solve (P) with the secants in the set $\mathcal{S}_{\text{aug}}$ using the RHash algorithm.

We dub this approach *RHash-CG*. RHash-CG iterates over the above steps until no new violating secants are added to the active set. Since the algorithm searches over all secants for violations in each iteration, RHash-CG ensures that all of the constraints are satisfied when it terminates.

A key benefit of RHash-CG is that only the set of active secants (and not all of the secants) needs to be stored in memory. This leads to significant improvements in memory complexity over competing algorithms, since the size of the secant set grows quadratically with the number of data points and can quickly exceed the system memory capacity in large-scale applications. The ℓ_∞-norm distortion function enables RHash to leverage the simple CG approach in order to scale to larger datasets, another advantage of ℓ_∞-norm for large-scale hashing.

2.4 RHash and NN Preservation Guarantee

Now that we have fully described the RHash algorithm, we develop a theory that bounds the probability of preserving all kNNs as a function of the ℓ_∞-norm distortion measure δ minimized by RHash.

Inspired by the definition of *relative contrast* [He *et al.*, 2012], we define a more generalized measure of data separability to preserve kNN that we call the *k-order gap* $\Delta_k := d(\mathbf{x}_0, \mathbf{x}_{k+1}) - d(\mathbf{x}_0, \mathbf{x}_k)$, where $\mathbf{x}_0$ is a query point and $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ are its k^{th} and $k + 1^{\text{st}}$ NNs, respectively. We formally show that, if the data is sufficiently separable (Δ_k is large), then RHash preserves all kNNs with high probability under assumptions that are typical in the NN search literature [He *et al.*, 2012] (see the appendix A for the proof).

Theorem 1. *Assume that all of the data points are independently generated from a mixture of Gaussians distribution, i.e.,* $\mathbf{x}_i \sim \sum_{p=1}^{P} \pi_p \mathcal{N}(\mu_p, \Sigma_p)$. *Let* $\mathbf{x}_0 \in \mathbb{R}^N$ *denote a query data point in the ambient space, and let the other data points* $\mathbf{x}_i$ *be ordered such that* $d(\mathbf{x}_0, \mathbf{x}_1) < d(\mathbf{x}_0, \mathbf{x}_2) < \ldots < d(\mathbf{x}_0, \mathbf{x}_Q)$. *Let* δ *denote the* ℓ_∞*-norm distortion parameter in RHash, and let* c *denote a positive constant. Then, if* $\mathbb{E}_x[\Delta_k] \geq 2\delta + \sqrt{\frac{1}{c} \log \frac{Qk}{\epsilon}}$, *then RHash preserves all kNNs of the query point* $\mathbf{x}_0$ *with probability at least* $1 - \epsilon$.

3 Experiments

In this section, we validate the RHash and RHash-CG algorithms via experiments using a range of synthetic and real-world datasets, including three small-scale, three medium-scale, and one large-scale datasets. We use three metrics to compare RHash against ten state-of-the-art binary hashing algorithms, including binary reconstructive embedding (BRE) [Kulis and Darrell, 2009], spectral hashing (SH) [Weiss *et al.*, 2009], anchor graph hashing (AGH) [Liu *et al.*, 2011], multidimensional spectral hashing (MDSH) [Weiss *et al.*, 2012],

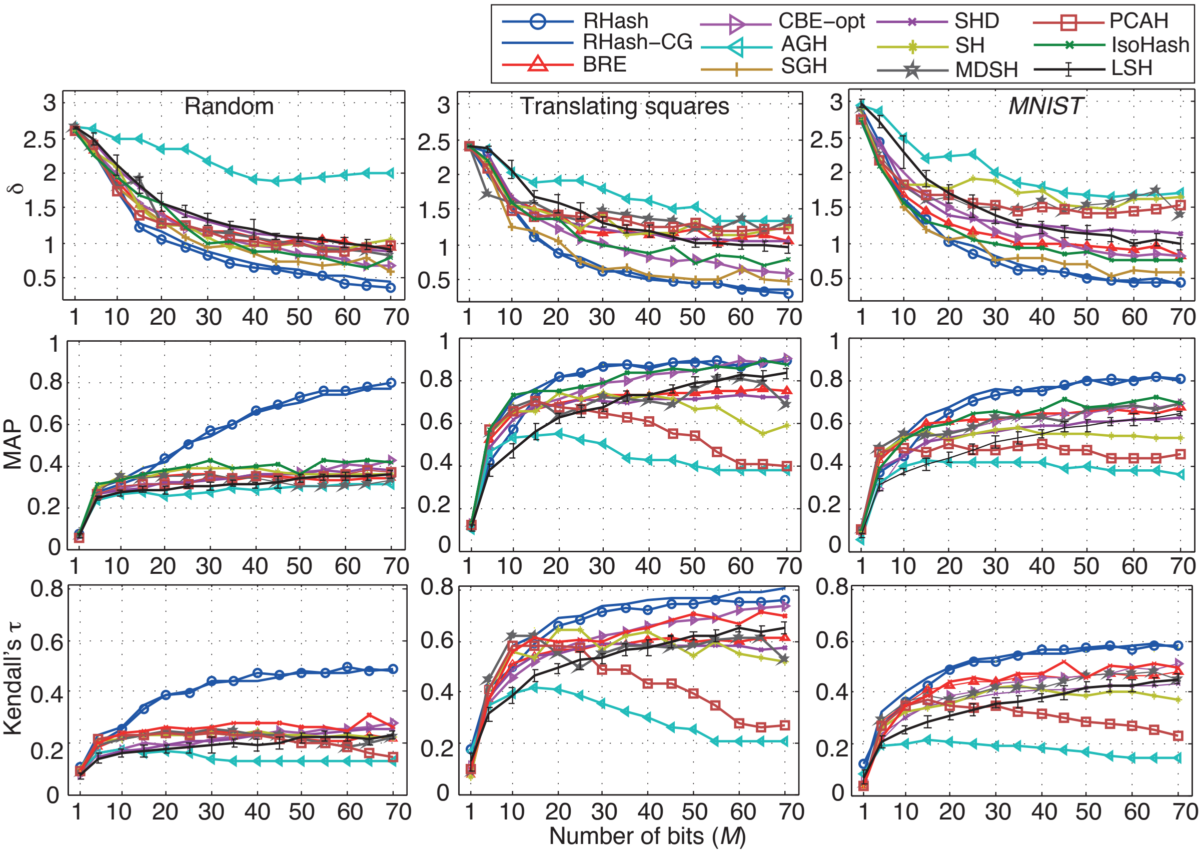

Figure 3: Comparison of the RHash and RHash-CG algorithms against several baseline binary hashing algorithms using three *small-scale datasets* ($Q = 100$). The performance of RHash-CG closely follows that of RHash, and both outperform all of the other algorithms in terms of the maximum distortion δ (superior NN distance preservation), mean average precision MAP of training samples (superior retrieval performance), and Kendall's τ rank correlation coefficient (superior ranking preservation).

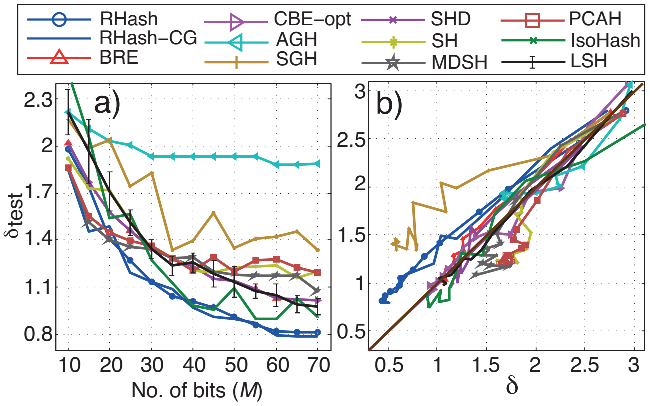

Figure 4: Comparison of RHash and RHash-CG against several state-of-the-art binary hashing algorithms in preserving distances on *MNIST* data. (a) RHash-CG outperforms the other algorithms in minimizing the distance distortion parameter on unseen data δ_{test}. (b) RHash and RHash-CG provide better distance preservation guarantee with a small sacrifice to universality.

scalable graph hashing (SGH) [Jiang and Li, 2015], PCA hashing (PCAH), isotropic hashing (IsoHash) [Kong and Li, 2012], spherical Hamming distance hashing (SHD) [Heo *et al.*, 2012], circulant binary embedding (CBE) [Yu *et al.*, 2014], and locality-sensitive hashing (LSH) [Indyk and Motwani, 1998].

3.1 Performance Metrics and Datasets

We compare the algorithms using the following three metrics:

Maximum distortion $\delta = \inf_{\lambda>0} ||\lambda\hat{\mathbf{v}} - \mathbf{c}||_{\infty}$, where the vector $\hat{\mathbf{v}}$ contains the pairwise Hamming distances between the learned binary codes. This metric quantifies the distance preservation after projecting the training data in the ambient space into binary codes. We also define the maximum distortion for unseen test data δ_{test}, which measures the distance preservation on a hold-out test dataset using the hash function learned from the training dataset.

Mean average precision (MAP) for NN preservation in the Hamming space. MAP is computed by first finding the set of kNNs for each query point in a hold-out test data set in the ambient space $\mathcal{L}^k$ and the corresponding set $\mathcal{L}^k_{\text{H}}$ in the Hamming space and then calculating the average precision $\text{AP} = |\mathcal{L}^k \cap \mathcal{L}^k_{\text{H}}|/k$. We then report MAP by calculating the

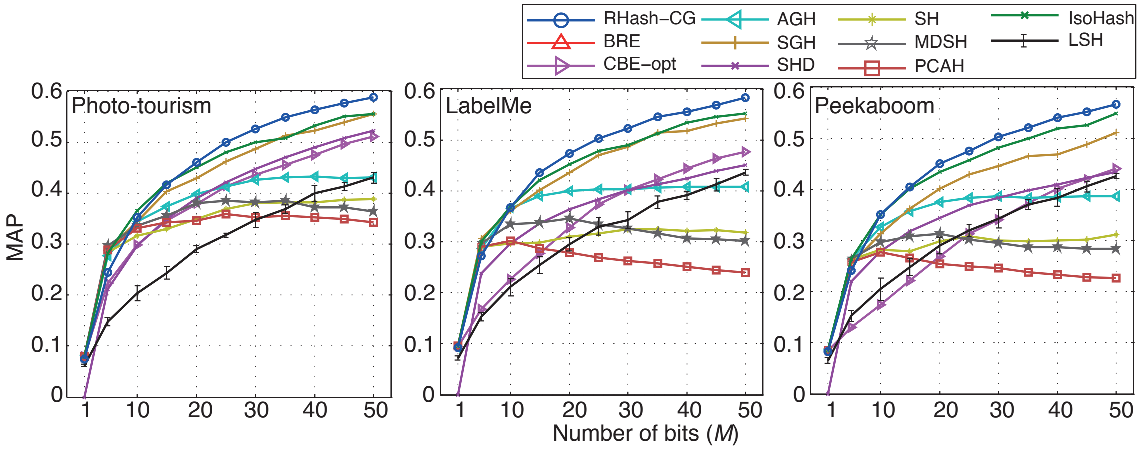

Figure 5: Hamming ranking performance comparison on three *medium-scale* datasets ($Q = 1000$). The top-50 neighbors are used to report MAP over a test data of same size.

mean value of AP across all data points.

Kendall's τ ranking correlation coefficient. We first rank the set of kNNs for each data point by increasing distance in the ambient space as $\mathcal{T}(\mathcal{L}^k)$ and in the Hamming space as $\mathcal{T}(\mathcal{L}^k_{\mathrm{H}})$. The Kendall τ correlation coefficient is a scalar $\tau \in [-1, 1]$ that measures the similarity between the two ranked sets $\mathcal{T}(\mathcal{L}^k)$ and $\mathcal{T}(\mathcal{L}^k_{\mathrm{H}})$ The value of τ increases as the similarity between the two rankings increases and reaches the maximum value of $\tau = 1$ when they are identical. We report the average value of τ achieved across all data points in the training dataset.

To compare the algorithms, we use the following standard datasets from computer vision: *Random* consists of independently drawn random vectors in $\mathbb{R}^{100}$ from a multivariate Gaussian distribution with zero mean and identity covariance matrix. *Translating squares* is a synthetic dataset consisting of 10×10 images that are translations of a 3×3 white square on a black background [Hegde *et al.*, 2015]. *MNIST* is a collection of 60,000 28×28 greyscale images of handwritten digits [LeCun and Cortes, 1998]. *Photo-Tourism* is a corpus of approximately 300,000 image patches represented using scale-invariant feature transform (SIFT) features [Lowe, 2004] in $\mathbb{R}^{128}$ [Snavely *et al.*, 2006]. *LabelMe* is a collection of over 20,000 images represented using GIST descriptors in $\mathbb{R}^{512}$ [Torralba *et al.*, 2008]. *Peekaboom* is a collection of 60,000 images represented using GIST descriptors in $\mathbb{R}^{512}$ [Torralba *et al.*, 2008]. Following the experimental approaches of the hashing literature [Kulis and Darrell, 2009; Norouzi and Fleet, 2011], we pre-process the data by subtracting the mean and then normalizing all points to lie on the unit sphere.

3.2 Small- and Medium-scale Experiments

We start by evaluating the performance of RHash and RHash-CG using a subset of the first three datasets. Small-scale datasets enable comparisons between the asymptotic behaviors of the algorithms in preserving distances in a regime where number of bits is high enough (compared to the total number of NNs) to preserve a large portion of NN distances.

Experimental Setup. We randomly select $Q = 100$ data points from the *Random*, *Translating squares*, and *MNIST* datasets. We then apply the RHash, RHash-CG, and all of the baseline algorithms on each dataset for different target binary code word lengths M from 1 to 70 bits. We set the RHash and RHash-CG algorithm parameters to the common choice of $\rho = 1$ and $\eta = 1.6$. To generate a hash function of length M for LSH, we draw M random vectors from a Gaussian distribution with zero mean and an identity covariance matrix. We use the same random vectors to initialize RHash and the other baseline algorithms. In the NN preservation experiments, to demonstrate the direct advantage of minimizing ℓ_∞-norm over ℓ_2-norm, we followed the exact procedure described in BRE [Kulis and Darrell, 2009] to select the training secants, i.e., we apply the RHash algorithm on the lowest 5% of the pairwise distances (which are set to zero as in BRE) combined with the highest 2% of the pairwise distances.

We follow the *continuation* approach [Wen *et al.*, 2010] to set the value of α. We start with a small value of α (e.g., $\alpha = 1$), in order to avoid becoming stuck in bad local minima, and then gradually increase α as the algorithm proceeds. As the algorithm moves closer to convergence and has obtained a reasonably good estimate of the parameters $\mathbf{W}$ and λ, we set $\alpha = 10$, which enforces a good approximation of the sign function (see Lem. 2 in the appendix A for an analysis of the accuracy of this approximation).

Results. The plots in the top row of Figure 3 illustrate the value of the distortion parameter δ as a function of the number of projections (bits) M. The performance of RHash and RHash-CG closely follow each other, indicating that RHash-CG is a good approximation to RHash. Both RHash and RHash-CG outperform the other baseline algorithms in terms of the distortion parameter δ. Among these baselines, LSH has the lowest distance preservation performance, which should be expected, since random projections are oblivious to the intrinsic geometry of the training dataset.

To achieve $\delta = 1$, RHash(-CG) requires 60% fewer bits M than CBE and BRE. RHash(-CG) also achieves better dis-

Table 1: Comparison of RHash-CG against several baseline binary hashing algorithms on the *large-scale* MNIST+rotation dataset with over 5 billion secants $|\mathcal{S}(\mathcal{X})|$. We tabulate the Hamming ranking performance in terms of mean average precision MAP for different sizes of the dataset. All training times are in seconds.

	MAP / Top-500				Training time (s)
Training size Q	$1k$	$10k$	$100k$	$240k$	$240k$
Secant size $\lvert\mathcal{S}(\mathcal{X})\rvert$	$100k$	$10M$	$1B$	$5B$	$5B$
RHash-CG	**52.79** (±0.15)	**54.69** (±0.18)	**54.93** (±0.23)	**55.52** (±0.11)	541.43
CBE	38.70 (±1.18)	38.12 (±1.34)	38.50 (±2.05)	38.53 (±0.83)	68.94
SPH	44.33 (±0.74)	44.24 (±0.61)	44.37 (±0.71)	44.32 (±0.63)	184.46
SH	40.12 (±0.00)	39.37 (±0.00)	38.79 (±0.00)	38.26 (±0.00)	3.05
MDSH	41.06 (±0.00)	41.23 (±0.00)	40.80 (±0.00)	40.39 (±0.00)	15.00
AGH	45.81 (±0.34)	47.78 (± 0.38)	47.69 (±0.41)	47.38 (±0.32)	4.49
SGH	51.32 (±0.07)	51.33 (±0.20)	51.01 (±0.23)	50.66 (±0.76)	5.89
BRE	48.33 (±0.65)	50.67(±0.33)	–	–	18685.51
PCAH	39.90 (±0.00)	38.53 (±0.00)	38.81 (±0.00)	37.50 (±0.00)	0.08
IsoHash	50.91 (±0.00)	50.90 (±0.00)	50.72 (±0.00)	50.55 (±0.00)	2.82
LSH	33.69 (±0.94)	33.69 (±0.94)	33.69 (±0.94)	33.69 (±0.94)	2.29×10^{-4}

tance preservation performance asymptotically, i.e., up to $\delta \approx 0.5$, given a sufficient number of bits ($M \geq 70$), while for most of the other algorithms the performance plateaus after $\delta = 1$. RHash's superior distance preservation performance extends well to unseen data. RHash achieves the lowest distortion on a test dataset δ_{test} compared to the other hashing algorithms (Figure 4).

The plots in the middle and bottom row of Figure 3 show the average precision for retrieving training data and the Kendall's τ correlation coefficient respectively, as a function of the number of bits M. We see that RHash preserves a higher percentage of NNs compared to other baseline algorithms as M increases, with better average ranking among $k = 10$ NNs. We see that RHash preserves a higher percentage of NNs and achieves a better average ranking performance compared to other baseline algorithms.

Next, we showcase the performance of RHash-CG on the three medium-scale, real-world datasets used in [Kulis and Darrell, 2009; Norouzi and Fleet, 2011], including *Phototourism*, *LabelMe*, and *Peekaboom* for the task of *data retrieval*. From each dataset we randomly select $Q = 1000$ training points, following the setup in BRE [Kulis and Darrell, 2009], and use them to train RHash-CG and the other baseline algorithms. We then randomly select a separate set of $Q = 1000$ data points and use it to test the performance of RHash-CG and other baseline algorithms in terms MAP with $k = 50$. Figure 5 illustrates the performance of RHash-CG on these datasets. RHash-CG outperforms all of the baseline algorithms by large margins in Hamming ranking performance in terms of MAP with top-50 NNs.

3.3 Large-scale Experiments

To demonstrate how RHash-CG scales to large-scale datasets, we use the full *MNIST* dataset with 60,000 training images and augment it with three rotated versions of each image (rotations of 90°, 180°, and 270°) to create a larger dataset with $Q = 240{,}000$ data points. Next, we construct 4 training sets with 1,000, 10,000, 100,000, and 240,000 images out of this large set. We train all algorithms with $M = 30$ bits and compare their performance on a test set of 10,000 images. The results for all algorithms are given in Table 1; we tabulate their performance in terms of MAP for the top-500 NNs.[1] The performance of RHash-CG is significantly better than the baseline algorithms and, moreover, improves as the size of the training set grows.

4 Discussion

We have demonstrated that our worst-case, ℓ_∞-norm-based, Robust Hashing (RHash) algorithm is superior to a wide range of algorithms based on the more traditional average-case, ℓ_2-norm. Despite its non-convexity and non-smoothness, RHash admits an efficient optimization algorithm that converges to a high-performing local minimum. Moreover, RHash-CG, the accelerated version of RHash, provides significant memory advantages over existing algorithms; the memory requirement of RHash-CG is linear in the number of *active* secants rather than the total number of secants, a significant advantage over the existing reconstructive hashing methods which minimize the reconstruction error between the original distances in ambient space and the Hamming distances in embedded space. Our experiments with six datasets, three metrics, and ten algorithms have demonstrated the superiority of RHash over other state-of-the-art data-dependent hashing algorithms. Thus, our results provide a strong motivation for exploring ℓ_∞-norm and other worst-case formulations for robust binary hashing.

A Proof of Theorem 1

Here, we prove Thm. 1 on the performance of RHash for k nearest neighbor (kNN) preservation.

We start with a result that bounds the error of the sigmoid approximation in RHash.

Lemma 2. *Let x be a Gaussian random variable distributed as $x \sim \mathcal{N}(\mu, \sigma^2)$. Define the distortion of the sigmoid approximation at x by $|h(x) - \sigma_\alpha(x)|$. Then, the expected distortion is bounded by*

$$\mathbb{E}_x[|h(x) - \sigma_\alpha(x)|] \leq \frac{1}{\sigma\sqrt{2\pi\alpha}} + 2e^{-(\sqrt{\alpha}+c/\alpha\sigma^2)},$$

[1] BRE fails to execute on a standard desktop PC with 12 GB of RAM due to the size of the secant set.

where c is a positive constant. As $\alpha \to \infty$, the expected distortion $\to 0$.

Proof. It is easy to see that the maximum distortion $|h(x) - \sigma_\alpha(x)|$ occurs at $x = 0$. Therefore, among different values of μ, $\mu = 0$ gives the largest distortion, since the density of x peaks at $x = 0$. Therefore, we bound the distortion at setting $\mu = 0$, which is an upper bound of the distortion when $\mu \neq 0$. By definition (1) above, $h(x)$ can be written as

$$h(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{otherwise.} \end{cases}$$

When $x \sim \mathcal{N}(0, \sigma^2)$ and we set $x_0 = \frac{1}{\sqrt{\alpha}}$, we can write

$$\begin{aligned}
&\mathbb{E}_x[|h(x) - \sigma_\alpha(x)|] = \int_{-\infty}^{\infty} |h(x) - \sigma_\alpha(x)| \mathcal{N}(x; 0, \sigma^2) \mathrm{d}x \\
&= 2\int_0^{\infty} (h(x) - \sigma_\alpha(x)) \mathcal{N}(x; 0, \sigma^2) \mathrm{d}x \qquad (6) \\
&= 2\int_0^{x_0} (h(x) - \sigma_\alpha(x)) \mathcal{N}(x; 0, \sigma^2) \mathrm{d}x \\
&\quad + 2\int_{x_0}^{\infty} (h(x) - \sigma_\alpha(x)) \mathcal{N}(x; 0, \sigma^2) \mathrm{d}x \\
&\leq 2\int_0^{x_0} \frac{1}{2} \mathcal{N}(x; 0, \sigma^2) \mathrm{d}x + 2\int_{x_0}^{\infty} \frac{1}{1 + e^{\alpha x_0}} \mathcal{N}(x; 0, \sigma^2) \mathrm{d}x \qquad (7) \\
&\leq \frac{x_0}{\sqrt{2\pi}\sigma} + 2\frac{e^{-c x_0^2/\sigma^2}}{1 + e^{\alpha x_0}} \qquad (8) \\
&\leq \frac{1}{\sigma\sqrt{2\pi\alpha}} + 2e^{-(\sqrt{\alpha} + c/\alpha\sigma^2)}, \qquad (9)
\end{aligned}$$

where c is a positive constant. In (6), we used the fact that $\sigma_\alpha(x)$ and $h(x)$ are symmetric with respect to the point $(0, \frac{1}{2})$. (7) follows from the properties of the sigmoid function; (8) follows from the Gaussian concentration inequality [Ledoux and Talagrand, 1991]; and (9) follows from the inequality $1/(1 + e^{\alpha x_0}) \leq e^{-\alpha x_0}$. The fact that $\mathbb{E}_x[|h(x) - \sigma_\alpha(x)|] \to 0$ as $\alpha \to \infty$ is obvious from the bound above. □

Proof of Thm. 1

In order to prove Thm. 1, we need the following Lemma:

Lemma 3. *Let $\mathbf{x}_0, \ldots, \mathbf{x}_N$ and Δ_k be defined as in Thm. 1. Then, there exists a constant c such that $P(\Delta_k - \mathbb{E}_x[\Delta_k] < t) \leq e^{-ct^2}$ for $t > 0$.*

Proof. Since the data points $\mathbf{x}_0$, $\mathbf{x}_k$ and $\mathbf{x}_{k+1}$ are independently generated from a finite mixture of Gaussian distributions, the random variable of their concatenation $\mathbf{y} = [\mathbf{x}_0^T, \mathbf{x}_k^T, \mathbf{x}_{k+1}^T]^T \in \mathbb{R}^{3N}$ is sub-Gaussian [Wainwright, 2009]. Then, we have

$$\begin{aligned}
\Delta_k(\mathbf{y}) &= \|\mathbf{x}_0 - \mathbf{x}_{k+1}\|_2 - \|\mathbf{x}_0 - \mathbf{x}_k\|_2 \\
&= \|\begin{pmatrix} \mathbf{I} & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & -\mathbf{I} \end{pmatrix} \mathbf{y}\|_2 - \|\begin{pmatrix} \mathbf{I} & 0 & 0 \\ 0 & -\mathbf{I} & 0 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{y}\|_2 \\
&\leq \|\begin{pmatrix} 2\mathbf{I} & 0 & 0 \\ 0 & -\mathbf{I} & 0 \\ 0 & 0 & -\mathbf{I} \end{pmatrix} \mathbf{y}\|_2 \leq 2\|\mathbf{y}\|_2,
\end{aligned}$$

where we have used the triangular inequality in the second to last step, the Rayleigh-Ritz theorem, and the fact that the maximum singular value of the diagonal matrix above is 2. This means that $\Delta_k(\mathbf{y})$ is a Lipschitz function of $\mathbf{y}$. Thus, by Talagrand's inequality [Ledoux and Talagrand, 1991], we have that $P(\Delta_k - \mathbb{E}_x[\Delta_k] < -t) \leq e^{-ct^2}$ for some positive constant c and $t > 0$, since $\mathbf{y}$ is sub-Gaussian. □

We are now ready to prove Thm. 1.

Proof. Let E denote the event that the set of top-k NNs is not preserved in the Hamming space. Then, we have $E = \cup e_{m,n}$, where $e_{m,n}$ denotes the event that $d_H(\mathbf{x}_0, \mathbf{x}_m) > d_H(\mathbf{x}_0, \mathbf{x}_n)$, with $m \in \{1, \ldots, k\}$ and $n \in \{k+1, \ldots, Q\}$. Then, using the union bound, we have

$$\begin{aligned}
P(E) &\leq \sum_{m,n} P(e_{m,n}) \leq k(Q-k)P(e_{k,k+1}) \\
&= k(Q-k)P(d_H(\mathbf{x}_0, \mathbf{x}_k) > d_H(\mathbf{x}_0, \mathbf{x}_{k+1})) \\
&= k(Q-k)P(d_H(\mathbf{x}_0, \mathbf{x}_{k+1}) < d_H(\mathbf{x}_0, \mathbf{x}_k)),
\end{aligned}$$

where we have used the fact that the most probable among all $e_{m,n}$ events is the one corresponding to the order mismatch between the k^{th} and $k + 1^{\text{st}}$ NN. Now, note that the RHash output δ satisfies $\max_{i,j} |d_H(\mathbf{x}_i, \mathbf{x}_j) - d(\mathbf{x}_i, \mathbf{x}_j)| \leq \delta^2$. Observe that $\Delta_k = d(\mathbf{x}_0, \mathbf{x}_{k+1}) - d(\mathbf{x}_0, \mathbf{x}_k) \geq 2\delta$ is a sufficient condition for $d_H(\mathbf{x}_0, \mathbf{x}_{k+1}) \geq d_H(\mathbf{x}_0, \mathbf{x}_k)$, since

$$\begin{aligned}
&d_H(\mathbf{x}_0, \mathbf{x}_{k+1}) - d_H(\mathbf{x}_0, \mathbf{x}_k) \\
&\quad \geq d(\mathbf{x}_0, \mathbf{x}_{k+1}) - \delta - d(\mathbf{x}_0, \mathbf{x}_k) - \delta \\
&\quad \geq 2\delta - 2\delta = 0
\end{aligned}$$

by the triangular inequality. This leads to

$$\begin{aligned}
&P(d_H(\mathbf{x}_0, \mathbf{x}_{k+1}) < d_H(\mathbf{x}_0, \mathbf{x}_k)) \\
&\quad = 1 - P(d_H(\mathbf{x}_0, \mathbf{x}_{k+1}) \geq d_H(\mathbf{x}_0, \mathbf{x}_k)) \\
&\quad \leq 1 - P(\Delta_k \geq 2\delta) = P(\Delta_k < 2\delta).
\end{aligned}$$

Combining all of the above with Lem. 3, the probability that the k NN is not preserved is bounded by

$$\begin{aligned}
P(E) &\leq k(Q-k)P(d_H(\mathbf{x}_0, \mathbf{x}_{k+1}) < d_H(\mathbf{x}_0, \mathbf{x}_k)) \\
&\leq k(Q-k)P(\Delta_k < 2\delta) \\
&= k(Q-k)P(\Delta_k - \mathbb{E}_x[\Delta_k] < -(\mathbb{E}_x[\Delta_k] - 2\delta)) \\
&\leq k(Q-k)e^{-c(\mathbb{E}_x[\Delta_k] - 2\delta)^2} \\
&\leq kQe^{-c(\mathbb{E}_x[\Delta_k] - 2\delta)^2}.
\end{aligned}$$

Now, by letting $kQe^{-c(\mathbb{E}_x[\Delta_k] - 2\delta)^2} \leq \epsilon$, we have that the requirement for the top-k NNs to be exactly preserved with probability at least $1 - \epsilon$ is $\mathbb{E}_x[\Delta_k] \geq 2\delta + \sqrt{\frac{1}{c} \log \frac{Qk}{\epsilon}}$. □

[2] Here we assume $\lambda = 1$ without loss of generality.

References

[Boyd *et al.*, 2011] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, Jan. 2011.

[Chatterjee and Hadi, 2009] S. Chatterjee and A. S. Hadi. *Sensitivity Analysis in Linear Regression*, volume 327. John Wiley & Sons, 2009.

[Dantzig and Wolfe, 1960] G. B. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8(1):101–111, Feb. 1960.

[Datar *et al.*, 2004] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proc. Annu. Symp. Comput. Geometry*, pages 253–262, June 2004.

[Du and Pardalos, 2013] D. Du and P. M. Pardalos. *Minimax and Applications*, volume 4. Springer, 2013.

[Hartley and Schaffalitzky, 2004] R. Hartley and F. Schaffalitzky. L_∞ minimization in geometric reconstruction problems. In *Proc. Conf. Comput. Vision Pattern Recogn. (CVPR)*, pages 504–509, June 2004.

[He *et al.*, 2012] J. He, S. Kumar, and S. Chang. On the difficulty of nearest neighbor search. In *Proc. Int. Conf. Mach. Learn.*, pages 1127–1134, June 2012.

[Hegde *et al.*, 2015] C. Hegde, A. C. Sankaranarayanan, W. Yin, and R. G. Baraniuk. Numax: A convex approach for learning near-isometric linear embeddings. *IEEE Trans. Signal Process.*, 63(22):6109–6121, Nov. 2015.

[Heo *et al.*, 2012] J. Heo, Y. Lee, J. He, S. Chang, and S. Yoon. Spherical hashing. In *Proc. Conf. Comput. Vision Pattern Recogn.*, pages 2957–2964, June 2012.

[Indyk and Motwani, 1998] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proc. Annu. Symp. Theory Comput.*, pages 604–613, May 1998.

[Jégou *et al.*, 2012] H. Jégou, T. Furon, and J. Fuchs. Anti-sparse coding for approximate nearest neighbor search. In *Proc. Intl. Conf. Acoustics Speech Signal Process. (ICASSP)*, pages 2029–2032. IEEE, 2012.

[Jiang and Li, 2015] Q. Y. Jiang and W. J. Li. Scalable graph hashing with feature transformation. In *Proc Intl. Joint Conf. Artif. Intell.*, Sep. 2015.

[Kong and Li, 2012] W. Kong and W. Li. Isotropic hashing. In *Adv. Neural Info. Process. Syst.*, pages 1646–1654, Dec. 2012.

[Kulis and Darrell, 2009] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. In *Adv. Neural Inf. Process. Syst.*, pages 1042–1050, Dec. 2009.

[LeCun and Cortes, 1998] Y. LeCun and C. Cortes. The MNIST database of handwritten digits, 1998.

[Ledoux and Talagrand, 1991] M. Ledoux and M. Talagrand. *Probability in Banach Spaces: Isoperimetry and Processes*. Springer-Verlag, 1991.

[Liu *et al.*, 2011] W. Liu, J. Wang, S. Kumar, and S. Chang. Hashing with graphs. In *Proc. Intl. Conf. Mach. Learn.*, pages 1–8, July 2011.

[Lowe, 2004] D. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, Nov. 2004.

[Lv *et al.*, 2007] Q. Lv, W. Josephson, Z. Wang, M. Charikar, and K. Li. Multi-probe lsh: efficient indexing for high-dimensional similarity search. In *Proc. Int. Conf. Very Large databases*, pages 950–961, Sep. 2007.

[Nesterov, 2007] Y. Nesterov. Gradient methods for minimizing composite objective function. Technical report, Université Catholique de Louvain, Sep. 2007.

[Norouzi and Fleet, 2011] M. Norouzi and D. J. Fleet. Minimal loss hashing for compact binary codes. In *Proc. Int. Conf. on Mach. Learn.*, pages 353–360, June 2011.

[Plan and Vershynin, 2014] Y. Plan and R. Vershynin. Dimension reduction by random hyperplane tessellations. *Discrete Comput. Geom.*, 51(2):438–461, Mar. 2014.

[Snavely *et al.*, 2006] N. Snavely, S. Seitz, and R. Szeliski. Photo tourism: Exploring photo collections in 3D. *ACM Trans. Graphics*, 25(3):835–846, July 2006.

[Studer *et al.*, 2014] C. Studer, T. Goldstein, W. Yin, and R. G. Baraniuk. Democratic representations. *arXiv preprint arXiv:1401.3420*, 2014.

[Torralba *et al.*, 2008] A. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *Proc. Conf. Comput. Vision Pattern Recogn.*, pages 1–8, June 2008.

[Wainwright, 2009] M. J. Wainwright. Sharp thresholds for high-dimensional and noisy sparsity recovery using ℓ_1 -constrained quadratic programming (Lasso). *IEEE Trans. Info. Theory*, 55(5):2183–2202, May 2009.

[Wang *et al.*, 2014] J. Wang, H. T. Shen, J. Song, and J. Ji. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927*, 2014.

[Wang *et al.*, 2016] Z. Wang, Y. Yang, S. Chang, Q. Ling, and T. S. Huang. Learning a deep ℓ_∞ encoder for hashing. In *Proc. Intl. Joint Conf. Artif. Intell.*, pages 2174–2180. AAAI Press, 2016.

[Weiss *et al.*, 2009] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *Adv. Neural Inf. Process. Syst.*, pages 1753–1760, Dec. 2009.

[Weiss *et al.*, 2012] Y. Weiss, R. Fergus, and A. Torralba. Multidimensional spectral hashing. In *European Conf. Comput. Vision*, pages 340–353, Oct. 2012.

[Wen *et al.*, 2010] Z. Wen, W. Yin, D. Goldfarb, and Y. Zhang. A fast algorithm for sparse reconstruction based on shrinkage, subspace optimization, and continuation. *SIAM J. Sci. Comput.*, 32(4):1832–1857, June 2010.

[Xu *et al.*, 2009] H. Xu, C. Caramanis, and S. Mannor. Robustness and regularization of support vector machines. *J. Mach. Learn. Research*, 10:1485–1510, 2009.

[Yu *et al.*, 2014] F. X. Yu, S. Kumar, Y. Gong, and S. Chang. Circulant binary embedding. In *Proc. Int. Conf. Mach. Learn.*, pages 946–954, June 2014.