Dimension Reduction in Kernel Spaces from Locality-Sensitive Hashing

Alexandr Andoni Piotr Indyk

April 11, 2009

Abstract

We provide novel methods for efficient dimensionality reduction in kernel spaces. That is, we provide efficient and explicit randomized maps from "data spaces" into "kernel spaces" of *low dimension*, which approximately preserve the original kernel values. The constructions are based on observing that such maps can be obtained from *Locality-Sensitive Hash (LSH)* functions, a primitive developed for fast approximate nearest neighbor search. Thus, we relate the question of dimensionality reduction in kernel spaces to the already existing theory of LSH functions.

Efficient dimensionality reduction in kernel spaces enables a substantial speedup of kernel-based algorithms, as experimentally shown in Rahimi-Recht (NIPS'07). Our framework generalizes one of their constructions.

1 Introduction

Kernel functions are a fundamental tool for learning a non-linear classifier. For example, they form a key component of Support Vector Machines (SVM). A kernel function defines a scalar product in a high-dimensional Euclidean space. Alternatively, it can be viewed as a lifting of the data space $\mathcal{S}$ into a new feature space, called the *kernel space* $\mathcal{K} \subset L_2$. The lifting enables performing complex classification using only a simple linear separator.

However, the map ϕ lifting the original space $\mathcal{S}$ into the kernel space is usually not explicit and the dimensionality of the kernel space is very high (or even infinite). As a result, algorithms that use the mapping ϕ directly are very inefficient. The classical approach this problem (the *kernel trick*) is to design algorithms that rely only on the scalar product in $\mathcal{K}$, given by the kernel function $K(x, y) = \phi(x) \cdot \phi(y)$ for all $x, y \in \mathcal{S}$ (c.f. [16, 18]).

In this work we address this problem more directly, by constructing *explicit* and *efficient* maps of the data space into the kernel space of *low dimension*. Specifically, our goal is to construct a map $F : \mathcal{S} \to \mathbb{R}^k$, for some *small* value of k, such that, for any $x, y \in \mathcal{S}$, the scalar product $F(x) \cdot F(y)$ is (approximately) equal to $K(x, y)$. This approach, in various forms, has been proposed before, e.g., in [8, 1, 12, 7, 17].

The approach has multiple benefits (cf. [17]). First, one can compute the large-margin separator directly, using direct algorithms that are potentially more efficient. Second, the classification itself can be done much more efficiently. Specifically, in a standard approach, an SVM outputs a classifier[1] $f(x) = \sum_{i=1}^{S} \alpha_i K(x, x_i)$, where $\{x_1, \dots x_S\}$ are the support vectors. Evaluating $f(x)$ takes time that is linear in the number of support vectors, which in principle could be as large as the number

[1]An example x is classified as positive iff $f(x) > 0$.

of the data points. In contrast, using the explicit map F, one can compute the weights w of a linear separator explicitly by letting $w = \sum_{i=1}^{S} \alpha_i F(x_i)$. Then the classifier can be defined as $f(x) = F(x) \cdot w$. The latter classifier can be evaluated in only $O(k)$ time, which is independent of the number of the support vectors.

The *existence* of a map F into a low-dimensional kernel space for any kernel can be derived from the random dimension-reduction techniques, such as Johnson-Lindenstrauss lemma. Namely, if we project the high-dimensional kernel space into a random low-dimensional subspace, then the scalar product between any pair of unit vectors is preserved up to an additive term of ϵ. Then the map F is defined as a composition of the high-dimensional map ϕ and the random projection. Arriaga-Vempala [6] further prove that the resulting F also approximately preserves the separation margin between the two classes. Unfortunately, the aforementioned existential construction is highly inefficient, since it uses the original high-dimensional mapping $\phi : \mathcal{S} \to \mathcal{K}$. Instead, we would like to construct a map F directly.

Related work. The problem of designing efficient dimensionality reduction techniques of kernel spaces has been previously investigated in the literature. Some of the first results were obtained for a simpler problem of designing the map F that works for a particular purpose (e.g, linear classification) and for a given dataset. This question can be seen as approximating the kernel (Gram) matrix $M_{ij} = K(x_i, x_j)$ of some data set $D = \{x_1, \ldots x_n\}$ (see, e.g., [8, 7, 1, 12]). For example, [7] consider the question of constructing F after one draws a small number of samples from the dataset and has only *black-box access* to $K(x, y)$. Under this condition, they construct a low-dimensional map F that preserves *linear separability* of the kernelized dataset. However, the constructed F depends on the data distribution[2]. Furthermore, the constructed mapping preserves linear separability of the data, but it does not appear to approximate the kernel function itself. Our more strict condition guarantees usefulness of F for other applications of kernels, such as regression and clustering. Because of these reasons, [8, 7] asks if it is possible to construct data-independent F for specific kernels.

More recently, Rahimi-Recht [17] provide the only currently known data-independent constructions. They give two constructions for maps F that approximate the kernel space. Their first construction works for the case when data live in the Euclidean space and the kernel is shift-invariant, i.e., $K(x, y) = K(\|x - y\|_2)$. For $\mathcal{S} = \mathbb{R}^d$, their function F maps the data points into a space of dimension $k = O(d \cdot \frac{\log 1/\epsilon}{\epsilon^2})$ and can be evaluated in a similar time. The construction proceeds by defining each feature as a sinusoid with a parameter drawn from a distribution defined by the Fourier transform of the kernel function. Their second construction is designed specifically for the Laplacian kernel $L(x, y) = e^{-\|x-y\|_1}$. The latter construction computes each feature in two steps. First, a randomly-shifted grid is imposed on the space $\mathbb{R}^d$. Then a point $x \in \mathbb{R}^d$ is encoded as the id of the grid cell containing x, represented in unary. Their experiments show that both methods compare favorably with standard methods for classification.

Our contribution. In this paper we propose a theoretical framework for obtaining the desired low-dimensional map F, which generalizes the second approach of [17]. The key idea is that one can obtain mappings F from *Locality-Sensitive Hashing (LSH)* functions, a hashing primitive that has been initially developed for the nearest-neighbor data structures [15, 5]. A family of LSH functions is a set of hash functions h such that, for any $x, y \in \mathcal{S}$ and a random h, the probability that x and y collide under h (i.e., $h(x) = h(y)$) is high when x and y are "close" in some underlying metric and the probability is low when x and y are "far". Several families of LSH functions for various

[2] In fact, as [7] prove, this condition is necessary if we have only a black-box access to the kernel function.

spaces $\mathcal{S}$ and distance measures have been developed so far (cf. the survey [5] and the references therein). We show that the existing LSH families yield efficient low-dimensional explicit maps F for corresponding spaces $\mathcal{S}$ and kernels K.

Our framework recovers the second construction of [17] by using one of the existing LSH families. However, our framework expresses a more general underlying phenomenon. As a result, we easily obtain mappings F for other similarity or dissimilarity functions.

2 Approximate Kernels from LSH families

In this section we construct the desired map $F : \mathcal{S} \to \mathbb{R}^k$ that approximates the kernel space for various data-spaces $\mathcal{S}$ and kernel functions K. Our approach is to utilize carefully constructed hash functions on $\mathcal{S}$, which are in essence "locality-sensitive hashing" functions as defined in [15].

2.1 Definition

We start by defining the notion of a family of kernel hash functions. Before giving a formal definition, we explain the intuition. Ideally, we would like a distribution over hash functions h such that $\Pr_h[h(x) = h(y)] = K(x, y)$ for any $x, y \in \mathcal{S}$. However, such a guarantee might be hard to obtain in some cases. Instead, we introduce a relaxed notion, which that we call *a family of ϵ-approximate kernel hash functions.*

Definition 2.1. *For $\epsilon > 0$ and kernel K,* a family of ϵ-approximate K-kernel hash functions (KHF) *is a set $\mathcal{H}$ of functions $h : \mathcal{S} \to U$ for some set U if, for any $x, y \in \mathcal{S}$, we have*

$$\left| \Pr_{h \in \mathcal{H}}[h(x) = h(y)] - K(x, y) \right| \le \epsilon.$$

To understand the definition, consider an example of such a family $\mathcal{H}$ for some specific K and $\epsilon = 0$. This family $\mathcal{H}$ is based on the original LSH scheme of [15]. Consider the hypercube $\mathcal{S} = \{0, 1\}^d$ with the kernel function $K(x, y) = (1 - \frac{\|x-y\|_1}{d})^p$, where $p \in \mathbb{N}$ is a fixed positive integer. We choose a hash function $h \in \mathcal{H}$ by taking a random set of coordinates $i_1 \ldots i_p \in [d]$ (with replacement), and setting $h(x) = x_{i_1} \ldots x_{i_p}$, i.e., h is a projection to a random set of p coordinates. It is immediate to see that $\mathcal{H}$ satisfies the above definition for $\epsilon = 0$.

2.2 Kernel maps from approximate kernel hash functions

We now prove how, given a family of ϵ-approximate kernel hash functions, we obtain the desired map F lifting data space into an (approximate) low-dimensional kernel space. Intuitively, we construct $F(x)$ by sampling many $h_i \in \mathcal{H}$, for some family $\mathcal{H}$ of approximate kernel hash functions, and then concatenating $h_i(x)$'s.

Lemma 2.2. *Let $\epsilon > 0$. Fix a space $\mathcal{S}$ that admits a family $\mathcal{H}$ of ϵ-approximate K-kernel hash functions, for a kernel function K. For any $\delta > 0$, there exists a randomized mapping $F : \mathcal{S} \to \mathbb{R}^k$, where $k = O(\frac{\log 1/\delta}{\epsilon^2})$, such that, for any $x, y \in \mathcal{S}$, we have $|F(x) \cdot F(y) - K(x, y)| < 2\epsilon$ with probability at least $1 - \delta$.*

The time to compute $F(x)$ is bounded by the time to evaluate functions from $\mathcal{H}$ times k.

We note that the image of the mapping F has a very simple form: it is a scaled hypercube $\left\{-\frac{1}{\sqrt{k}}, +\frac{1}{\sqrt{k}}\right\}^k$.

Proof. Draw k functions h from $\mathcal{H}$ and call them $h_1, \dots h_k$. Consider the function

$$F(x) = \frac{1}{\sqrt{k}}(E(h_1(x)), E(h_2(x)), \dots E(h_k(x))),$$

where $E : U \to \ell_2$ is an "encoding function", mapping the universe U into vectors of real numbers. For now we assume that E is such that $E(a) \cdot E(b) = 1$ when $a = b$ and $E(a) \cdot E(b) = 0$ when $a \neq b$; we will relax this assumption later in the proof. Let $\chi[A]$ be the indicator random variable for an event A, which is equal to 1 iff A is true. Then, we can see that $F(x) \cdot F(y) = \frac{\sum_{i=1}^k \chi[h_i(x)=h_i(y)]}{k}$. Furthermore, by Chernoff bound, we have

$$\left| F(x) \cdot F(y) - \Pr_h[h(x) = h(y)] \right| \leq \epsilon/2$$

with probability at least $1 - \delta/3$. Finally, using the definition of ϵ-approximate K-kernel hash functions $\mathcal{H}$, we deduce that $|F(x) \cdot F(y) - K(x, y)| \leq \epsilon + \epsilon/2$ with probability at least $1 - \delta/3$.

It remains to describe the encoding function E. A simple approach is to encode the universe U in a unary format, that is, map symbols $a \in U$ into a vectors of length U with exactly one coordinate equal to 1. However this is inefficient, since it multiplies the target dimension k by $|U|$. Instead, for each coordinate $i \in [k]$, we choose a random map $E_i : U \to \{-1, +1\}$, and take

$$F(x) = \frac{1}{\sqrt{k}}(E_1(h_1(x)), E_2(h_2(x)), \dots E_k(h_k(x))).$$

It is easy to see that even after this simplification, $|F(x) \cdot F(y) - K(x, y)| \leq 2\epsilon$ with probability at least $1-\delta$. Indeed, let c be the number of indexes i such that $h_i(x) = h_i(y)$. Then $F(x) \cdot F(y)$ is a sum of c ones and $k - c$ independent random variables chosen uniformly at random from $\{-1, +1\}$. We already showed that $|c/k - K(x, y)| \leq \epsilon + \epsilon/2$. By Chernoff bound, the sum of the other $k - c$ values is at most $\epsilon/2 \cdot k$ with probability at least $1 - \delta/3$. The conclusion follows from an application of the triangle inequality. □

2.3 Some families of kernel hash functions

We now show how known LSH families of functions yield families of (approximate) kernel hash functions for various kernels. By Lemma 2.2, we immediately obtain efficient maps F into low-dimensional kernel spaces, for corresponding kernels. Please refer to [5] for further information about the LSH functions.

We defer the proofs of the lemmas from this section to the appendix.

Laplacian kernel. Consider the d-dimensional Manhattan space $\mathcal{S} = \ell_1^d$ and the Laplacian kernel $L(x, y) = e^{-\|x-y\|_1/\sigma}$. We show that, for any $\epsilon > 0$, this space admits a family of ϵ-approximate L-kernel hash functions based on the LSH functions of [3, 2]. The final resulting map is similar to the second construction of [17].

We show how to pick a hash function $h \in \mathcal{H}$. Fix parameters $p = 2/\epsilon$ and $t = \sigma \cdot p$. Then construct p random functions f_i, $i = 1 \dots p$, by imposing a randomly shifted regular grid of side length t. Formally, we choose $s_1, \dots s_d$ at random from $[0, t)$, and define

$$f_i(x_1, \dots, x_d) \triangleq (\lfloor (x_1 - s_1)/t \rfloor, \dots, \lfloor (x_d - s_d)/t \rfloor).$$

The kernel hash function h is simply a concatenation of f_i chosen as above: $h(x) = (f_1(x), f_2(x), \dots f_p(x))$.

Lemma 2.3. *Let $\epsilon > 0$. Suppose h is a hash function chosen as above. Then, for any $x, y \in \ell_1^d$, we have $\left|\Pr_h[h(x) = h(y)] - L(x, y)\right| \le \epsilon$.*

We note that the same result holds for the Laplacian kernel in the Euclidean space (instead of ℓ_1), namely $L_2(x, y) = e^{-\|x-y\|_2/\sigma}$. To obtain the family, we use the exact same hash functions as above except that, for each f_i, we rotate its grid at random beforehand.

Near-Gaussian kernel. Consider the d-dimensional Euclidean space $\mathcal{S} = \ell_2^d$ and the kernel $K_{\text{erfc}}(x, y) = \frac{\text{erfc}(\|x-y\|_2/\sigma)}{2-\text{erfc}(\|x-y\|_2/\sigma)}$, where $\text{erfc}(x) = \frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2} dt$ is the *Gauss error function*. As we show in a moment, this function approximates well the Gaussian kernel $e^{-\|x-y\|_2^2/\sigma^2}$.

A KHF for K_{erfc} follows from the LSH family in [4], which generates a hash function $h(x)$ as follows. Set $t = O(\frac{1}{\epsilon^2}\log 1/\epsilon)$ and $w = \frac{1}{2\sqrt{2}}\sqrt{t}\sigma$. First pick a random projection from $\mathbb{R}^d$ to $\mathbb{R}^t$, denoted by the matrix A. Then, in the projected t-dimensional space, pick $U = 2^{O(t \log t)}$ grids of balls of radius w, where a grid $u \in [U]$ of balls is the (infinite) set of balls with centers at $4w \cdot \mathbb{Z}^d + s_u$ for a random translation $s_u \in [0, 4w)^d$. Finally, define $h(x)$ as the index of the ball with the smallest $u \in [U]$ that contains the point Ax, the projection of x.

Lemma 2.4. *Let $\epsilon > 0$. Suppose h is a hash function chosen as above. Then, for any $x, y \in \ell_2^d$, we have $\left|\Pr_h[h(x) = h(y)] - K_{\text{erfc}}(x, y)\right| \le \epsilon$. The function h can be evaluated in time $2^{\tilde{O}(1/\epsilon^2)}$.*

We note that this same family can be used for the Gaussian kernel $G(x, y) = e^{-\|x-y\|_2^2/\sigma^2}$, although we do not achieve an approximation for arbitrary value of $\epsilon > 0$. However, the following lemma proves that the above family is 0.16-approximate family of G-kernel hash functions.

Lemma 2.5. *Suppose h is a hash function chosen as above for fixed $t = O(1)$ and $w = O(1)$. Then, for any $x, y \in \ell_2^d$, we have $\left|\Pr_h[h(x) = h(y)] - G(x, y)\right| \le 0.16$. The function h can be evaluated in constant time.*

Jaccard kernel. Consider the space of sets over some universe W, namely $\mathcal{S} = \{A : A \subseteq W\}$, under the kernel $K_J(A, B) = \frac{|A \cap B|}{|A \cup B|}$.

Here, a KHF follows from the standard min-hash functions designed by [9, 10]. A hash function is chosen as follows. Pick a random permutation π on the ground universe W. Then, define $h_\pi(A) = \min\{\pi(a) \mid a \in A\}$. The family is 0-approximate kernel hash function.

Geodesic kernel. Consider a hypersphere in d-dimensional space $\mathcal{S} = \mathbb{S}^{d-1}$ with the kernel $K_\theta(x, y) = 1 - \frac{\theta(x,y)}{\pi}$, where $\theta(x, y)$ is the angle between vectors x and y which is proportional to the geodesic distance from x to y on the hypersphere.

Here, a KHF follows from the "random hyperplane" hash function designed by Charikar [11] (inspired by [14]). A hash function is chosen as follows. Pick a random unit-length vector $u \in \mathbb{R}^d$, and define $h_u(x) = \text{sign}(u \cdot x)$. The hash function can also be viewed as partitioning the space into two half-spaces by a randomly chosen hyperplane passing through the center. The resulting family is a family of 0-approximate K_θ-kernel hash functions.

Acknowledgments

We would like to thank Nina Balcan, Cy Chan and Michael Collins for several inspiring discussions.

References

[1] D. Achlioptas, F. McSherry, and B. Schölkopf. Sampling techniques for kernel methods. In S. B. Thomas, G. Dietterich, and Z. Ghahramani, editors, *Advances in Neural Information Processing Systems (NIPS)*, 2001.

[2] A. Andoni. Approximate nearest neighbor problem in high dimensions. *M.Eng. Thesis, Massachusetts Institute of Technology*, June 2005.

[3] A. Andoni and P. Indyk. Efficient algorithms for substring near neighbor problem. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, pages 1203–1212, 2006.

[4] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *Proceedings of the Symposium on Foundations of Computer Science*, 2006.

[5] A. Andoni and P. Indyk. Near-optimal hahsing algorithms for approximate nearest neighbor in high dimensions. *Communications of the ACM*, 51(1):117–122, 2008.

[6] R. I. Arriaga and S. Vempala. An algorithmic theory of learning: Robust concepts and random projection. *Proceedings of the Symposium on Foundations of Computer Science*, 1999.

[7] M.-F. Balcan, A. Blum, and S. Vempala. Kernels as features: On kernels, margins, and low-dimensional mappings. *Machine Learning*, 65(1):79 – 94, October 2006.

[8] A. Blum. Random projection, margins, kernels, and feature-selection. *LNCS 3940*, pages 52–68, 2006. Survey article based on an invited talk given at the 2005 PASCAL Workshop on Subspace, Latent Structure and Feature selection techniques.

[9] A. Broder. On the resemblance and containment of documents. *Proceedings of Compression and Complexity of Sequences*, pages 21–29, 1997.

[10] A. Broder, S. Glassman, M. Manasse, and G. Zweig. Syntactic clustering of the web. *Proceedings of the Sixth International World Wide Web Conference*, pages 391–404, 1997.

[11] M. Charikar. Similarity estimation techniques from rounding. *Proceedings of the Symposium on Theory of Computing*, 2002.

[12] D. DeCoste and D. Mazzoni. Fast query-optimized kernel machine via incremental approximate nearest support vectors. In *IEEE International Conference on Machine Learning*, 2003.

[13] P. Diaconis and D. Freedman. A dozen de finetti-style results in search of a theory. *Probabilités et Statistiques*, 23(2):397–423, 1987.

[14] M. Goemans and D. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42:1115–1145, 1995.

[15] P. Indyk and R. Motwani. Approximate nearest neighbor: towards removing the curse of dimensionality. *Proceedings of the Symposium on Theory of Computing*, pages 604–613, 1998.

[16] K. Muller, S. Mika, G. Ratsch, K. Tsuda, and B. Scholkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, March 2001.

[17] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2007.

[18] B. Schölkopf and A. J. Smola. *Learning with kernels. Support Vector Machines, Regularization, Optimization, and Beyond.* MIT University Press, Cambridge, 2002.

A Proofs of the KHF property of families from Section 2.3

Proof of Lemma 2.3. The proof follows directly from Lemma 4.1.1 in [2], which states that for any f_i, $i = 1 \dots p$,

$$1 - \|x-y\|_1/t \le \Pr_{f_i}[f_i(x) = f_i(y)] \le e^{-\|x-y\|_1/t}.$$

Since $\Pr_h[h(x) = h(y)] = \prod_{i=1}^p \Pr_{f_i}[f_i(x) = f_i(y)]$, the probability of collision under h is

$$(1 - \|x-y\|_1/t)^p \le \Pr_h[h(x) = h(y)] \le e^{-\|x-y\|_1 p/t}.$$

If we let $\Delta = \|x-y\|_1/\sigma$, then $\|x-y\|_1/t = \Delta/p$. We use the approximation $1 - \xi \ge e^{-\xi} e^{\frac{-\xi^2}{1-\xi}}$ for $\xi \le \frac{1}{2}$ and $e^{-\xi} \ge 1 - \xi$. Then, for $\Delta/p \le 1/2$, we obtain

$$\left|\Pr_h[h(x) = h(y)] - e^{-\|x-y\|_1/\sigma}\right| \le e^{-\Delta} - e^{-\Delta} \cdot e^{-p\frac{(\Delta/p)^2}{1-\Delta/p}} \le e^{-\Delta}\left(1 - \left(1 - p\frac{(\Delta/p)^2}{1-\Delta/p}\right)\right) \le \frac{2}{p}\max_{\Delta \ge 0}\frac{\Delta^2}{e^{\Delta}}.$$

Since $\max_{\Delta \ge 0} \Delta^2/e^{\Delta} \le 1$ and $p = 2/\epsilon$, the above quantity is upper-bounded by ϵ. For $\Delta > p/2 = 1/\epsilon$, the conclusion follows immediately since, in this case, $e^{-\|x-y\|_1/\sigma} < \epsilon$. □

Proof of Lemma 2.4. We use the analysis of this hash function presented in [4]. First, Lemma 3.1 of [4] proves that the entire space $\mathbb{R}^t$ will indeed be covered by balls with probability at least $1 - 2^{-\Omega(t \log t)} \ge 1 - \epsilon/4$. Second, we argue that after the projection into $\mathbb{R}^t$ is performed, the incurred distortion of $\|x-y\|_2$ is negligible. Indeed, let $\Delta = \|x-y\|_2$. Johnson-Lindenstrauss lemma says that $\Delta' = \|Ax - Ay\|_2$ is within a multiplicative factor of $1 + \epsilon/8$ of Δ. Then, $|K_{\text{erfc}}(x,y) - K_{\text{erfc}}(Ax,Ay)| = |K_{\text{erfc}}(\Delta) - K_{\text{erfc}}(\Delta')| = |K'_{\text{erfc}}(\xi)| \cdot |\Delta - \Delta'| \le \epsilon/3$ where $\xi \in [\Delta(1-\epsilon/8), \Delta(1+\epsilon/8)]$.

Finally, Eqn. (1) in [4] states that

$$\Pr[h(Ax) = h(Ay) | \|Ax - Ay\|_2 = \Delta'] = \frac{I(\Delta'/2, w)}{1 - I(\Delta'/2, w)},$$

where $I(\Delta'/2, w)$ is the probability that a random point chosen from a ball of radius w has its first coordinate at least as big as $\Delta'/2$. We can approximate the distribution of the first coordinate of a random point from a ball as a Gaussian of variance $\frac{1}{t}w^2$ (cf. [13]). The probability that a random Gaussian of variance $\frac{1}{t}w^2$ is greater than $\Delta'/2$ is precisely $\frac{1}{2}\operatorname{erfc}(\Delta' \cdot \frac{\sqrt{t}}{2\sqrt{2}w})$. Thus, we obtain that $|I(\Delta'/2, w) - \frac{1}{2}\operatorname{erfc}(\Delta' \cdot \frac{\sqrt{t}}{2\sqrt{2}w})| \le O(1/t) \le \epsilon/20$ (cf. [13]). This further implies that $\left|\frac{I(\Delta'/2,w)}{1-I(\Delta'/2,w)} - K_{\text{erfc}}(\Delta')\right| \le \epsilon/4$.

In the end, we conclude that $|\Pr_h[h(x) = h(y)] - K_{\text{erfc}}(x,y)| \le \epsilon$. □

Proof of Lemma 2.5. We observe that $\max_{\Delta \in [0,\infty)} \left| e^{-\Delta^2} - \frac{\operatorname{erfc}(0.4\Delta)}{2 - \operatorname{erfc}(0.4\Delta)} \right| \le 0.158$. The lemma then follows by Lemma 2.4 for $\epsilon = 0.001$. □