Locality Sensitive Hashing and Nearest Neighbors[1]

- **Nearest Neighbor and Approximate Nearest Neighbor Problem.** In the nearest neighbor problem we are given a point set P which lies in a metric space (X, d). That is, $P \subseteq X$ where X is some ambient space with a distance function $d : X \times X \to \mathbb{R}_{\geq 0}$ satisfying three properties: (i) $d(x, x) = 0$ for all $x \in X$, (ii) (symmetry) $d(x, y) = d(y, x)$, and (iii) (triangle inequality) for any three points $x, y, z \in X$, $d(x, z) \leq d(x, y) + d(y, z)$. Examples of common metric spaces are $(\mathbb{R}^m, \ell_2)$ where the ambient space is all m-dimensional vectors and the distance between points is the usual Euclidean distance. There can be many other examples; for instance, X could be the set of all strings of a certain length, and $d(s, t)$ could be the Hamming distance between them which indicates the number of positions in which they differ. For now, let's stick to the abstract notion of (X, d) satisfying the three properties above.

  The goal in the nearest neighbor problem is to *preprocess* P and create a data-structure so that when *queried* with $x \in X$, it returns p in P that is closest to x. That is, $p = \arg\min_{q \in P} d(q, x)$. There are three parameters of interest: (a) preprocessing time, (b) space required by the data structure, and (c) the query time.

  Oftentimes, the *exact* nearest neighbor problem is too difficult to solve, and one is okay with an approximate solution. This is the c-approximate nearest neighbor, or simply the c-ANN problem. When queried with a point $x \in X$, the objective is to return a point p such that $d(p, x) \leq c \cdot \min_{q \in P} d(q, x)$. Furthermore, the algorithm is allowed to be randomized with error parameter δ, and so it can return either $\perp$ or a "far away" point, but that occurs with probability at most δ.

  In this lecture, we will see a technique to solve the ANN problem based on a particular kind of hashing called *locally sensitive hashing*. This is not the only application of this hashing technique, but it was the setting in which it was discovered, and in some sense the quintessential application. These lecture notes themselves are a sample/sketch/hash of these well[2] written[3] papers[4]. There are other techniques for solving the ANN problem especially in low-dimensional metric spaces (ie, when $X = \mathbb{R}^2$ or $\mathbb{R}^3$ and $d(\cdot, \cdot)$ is the Euclidean distance) but that will not be the focus of this lecture.

- **Approximate Near Neighbor Problem.** We will be focusing on a slightly different problem called the approximate *near* neighbor problem. In this problem, apart from the point set P, the approximation quality c and the failure probability δ, we are also given a parameter r. The goal is to create a data structure $\mathsf{NearNbr}^{(\delta)}_{r,c}$ which has the following property.

  When queried with a point $x \in X$, $\mathsf{NearNbr}^{(\delta)}_{r,c}(x)$ either returns a point $p \in P$ or $\perp$.
  - If it returns p, then $d(p, x) \leq cr$.
  - If there exists a point $q \in P$ such that $d(x, q) \leq r$, then $\mathbf{Pr}[\mathsf{NearNbr}^{(\delta)}_{r,c}(x) = \perp] \leq \delta$.

[1] *Lecture notes by Deeparnab Chakrabarty. Last modified : 30th April, 2023*
These have not gone through scrutiny and may contain errors. If you find any, or have any other comments, please email me at deeparnab@dartmouth.edu. Highly appreciated!

[2] S Har-Peled, P Indyk, R Motwani: *Approximate nearest neighbor: Towards removing the curse of dimensionality* , Theory of Computing, 2012. Journal version of Indyk-Motwani paper from STOC 1998.

[3] A. Z. Broder, *On the resemblance and containment of documents, Proceedings. Compression and Complexity of Sequences, 1997*

[4] M. Charikar, *Similarity estimation techniques from rounding algorithms*, Symposium on Theory of Computing (STOC), 2002.

In plain English, if there is a point in P which is "near" x, that is, within distance r, then with probability $\geq 1-\delta$, the data structure returns a point p which is within distance $\leq cr$. Note that it still may return a point within $\leq cr$ even if there is no point in P within distance r. If we use $\mathsf{NearNbr}_{r,c}$ without the subscript δ, then the assumption is that $\delta = 1/2$. Note that one can get $\mathsf{NearNbr}^{(\delta)}_{r,c}$ using $O(\log(1/\delta))$ independent instances of $\mathsf{NearNbr}_{r,c}$.

- One can use a binary search style idea to reduce the ANN problem to approximate near neighbor problem. Suppose we knew parameters r_{max} and r_{min} such that for any two distinct points p, q in P, we have $r_{\mathsf{min}} \leq d(p,q) \leq r_{\mathsf{max}}$. For instance, if $d(\cdot,\cdot)$ was the Hamming distance between strings of length m, then $r_{\mathsf{max}} = m$ and $r_{\mathsf{min}} = 1$ satisfy the above property. Let $\Delta_P := \frac{r_{\mathsf{max}}}{r_{\mathsf{min}}}$; this parameter is often called the *aspect ratio* of the point set P. One can use $O(\log\Delta)$ many $\mathsf{NearNbr}^{(\delta)}_{r,c}$'s to obtain an $2c$-ANN data structure. The formal statement is below.

  Lemma 1. *Suppose an $\mathsf{NearNbr}^{(\delta)}_{r,c}$-data structure can be constructed in $T_{r,c,\delta}(n)$ pre-processing time, uses space $S_{r,c,\delta}(n)$, and answers queries in time $Q_{r,c,\delta}(n)$. Assume these parameters are monotonically non-increasing in r and n, and non-decreasing in c and δ. Then a $2c$-ANN data structure can be constructed which takes pre-processing time $O(T_{r,c,\delta}(n)\cdot\log\Delta)$, uses space $O\left(S_{r,c,\delta}(n)\cdot\log\Delta\right)$, has query time $O(Q_{r,c,\delta}(n)\log\log\Delta)$ with error probability $O(\delta\log\Delta)$.*

  Exercise: *Prove the above lemma.*

  Remark: *The above lemma is unsatisfactory if Δ is very large. The paper[a] by Har-peled, Indyk, and Motwani does show a way how to get the reduction without any dependence on Δ.*

  [a]The statements in Corollary 2.6 and Theorem 10 have an unfortunate typographic error: the $\log^2 n$ should be in the numerator and not the denominator.

- **Locality Sensitive Hashing (LSH).** Given a metric space (X, d) and two parameters r, c, a family $\mathcal{H}_{r,c}$ of functions $h : X \to R$ for some range R is an (r, c, p_1, p_2)-locality sensitive hash (LSH) family has the following properties

  - For any two points $x, y \in X$ with $d(x,y) \leq r$, $\quad \mathbf{Pr}_{h\in_R\mathcal{H}}[h(x) = h(y)] \geq p_1$
  - For any two points $x, y \in X$ with $d(x,y) \geq cr$, $\quad \mathbf{Pr}_{h\in_R\mathcal{H}}[h(x) = h(y)] \leq p_2$

  In plain English, close by points are more likely to collide and far away points are less likely to collide. The parameter $\rho = \frac{\ln(1/p_1)}{\ln(1/p_2)}$ is of interest and it is desirable to have this as small as possible. Note that when $p_1 > p_2$, we have $\rho < 1$.

- **Using LSH to solve Approximate Near Neighbor Problem.** We now show how to create the $\mathsf{NearNbr}_{r,c}$ data structure using access to an (r, c, p_1, p_2)-LSH family. As you will see, the preprocessing times, space, and query times will all depend on the parameter ρ.

  The idea is quite simple. One first concatenates such hash-functions to amplify the probabilities; that is close by points get hashed together with probability $\geq p_1^k$ while far away points get hashed together with probability $\leq p_2^k$. If $p_1/p_2 > 1$, then this ratio is amplified. Next, one samples L many concatenated hash functions $g_1, \ldots, g_L$, and use them to construct L different hash tables T_1 to T_L, and each point $p \in P$ is stored in each of these tables. When queried with a point x, one "collects" the points from the tables by picking the points in the $g_i(x)$th location of table T_i, stopping once $16L$ points have been collected. One searches for a point in these which is within $\leq cr$ distance. If none found, $\perp$ is returned. The k and L are chosen such that the probability of error is $\leq 1/2$.

- *Pre-processing.* Throughout, we use n to denote $|P|$, the size of the data set. Note that X itself could be infinite. Nevertheless, we do assume that every point $x \in X$ can be represented in $O(1)$-word of memory and arithmetic is possible in $O(1)$ time.
  - Given the family $\mathcal{H}$, construct the family $\mathcal{G}$ of hash functions $g : X \to R^k$ where we sample g by first sampling, with replacement, k independent h_i's from $\mathcal{H}$, and defining
  $$\forall x \in X : \quad g(x) := (h_1(x), h_2(x), \ldots, h_k(x))$$
  The parameter k will be chosen as $k := \left\lceil \frac{\ln n}{\ln(1/p_2)} \right\rceil$. We will soon see the reason of this choice, and for now the reader can keep this k as a parameter.
  - We now choose L independent g_j's, with replacement, and initialize L different hash-tables $T_1, \ldots, T_L$ each with range $[n]$. We sample $\jmath$ from a universal hash family of functions with domain R^k and range $[n]$. For every point $p \in P$, for every $1 \leq j \leq L$, we compute $g_j(p)$ and add p in the $\jmath(g_j(p))$th "bucket" (list) at T_j.
  The parameter L will be chosen[5] as $L := \lceil (1/p_1)^k \rceil \leq \lceil n^\rho / p_1 \rceil$. Once again, the reason for this choice will be clear. This completes the preprocessing step.

  For every point p, we compute L different $g_j(p)$'s, and for each $g_j(p)$ we compute k different $h_i(p)$'s. Assuming that the h-computations (and $\jmath$-computations) take $O(1)$ time, the pre-processing time is $O(nkL)$. Using the parameters that we will set, the preprocessing time is
  $$T_{r,c,0.5}(n) = O\left(n^{1+\rho} \cdot \frac{\ln n}{p_1 \ln(1/p_2)}\right) = O(n^{1+\rho} \ln n) \text{ if } p_1 \text{ and } p_2 \text{ are constants}$$
  The space is the dominated by the size of the buckets. Each point p is placed in L buckets, and therefore, the space required to save the points is $O(nL)$ assuming the points in P can be stored in $O(1)$ words of memory. This is also the size of the hash tables. The space required to store the various hash functions is $O(kL)$ assuming each $h \in \mathcal{H}$ can be stored in $O(1)$ words of memory. When p_1 is reasonable, $k \ll n$, and therefore the total space is
  $$S_{r,c,0.5}(n) = O\left(n^{1+\rho} \cdot \frac{1}{p_1}\right) = O(n^{1+\rho}) \text{ if } p_1 \text{ is a constant}$$
- *Query.* Upon a query $x \in X$, we do the following.
  - (Computation) Compute $g_j(x)$ for all $1 \leq j \leq L$. Initialize $B \leftarrow \emptyset$.
  - (Point collection) Starting with $j = 1$, keep adding points from $\jmath(g_j(x))$th bucket of T_j into B, stopping once $|B|$ becomes $16L$.
  - (Selection) If there exists a point $q \in B$ with $d(x, q) \leq cr$, return q. Otherwise return $\perp$.

[5]if we ignored ceilings, L would precisely be n^ρ.

The query time is $O(kL)$ has function computations, $\leq 16L$ point collection and distance computations. Therefore, using the k and L values, the total time complexity of the query process is

$$Q_{r,c,0.5}(n) = O\left(n^{\rho} \cdot \frac{\ln n}{p_1}\right) = O(n^{\rho} \ln n) \text{ if } p_1 \text{ is a constant}$$

- *Correctness.* It should be clear that when the query algorithm called with $x \in X$ returns a q then, by design, $d(q,x) \leq cr$. We need to prove that if there exists $p \in P$ such that $d(p,x) \leq r$, then the $\mathbf{Pr}[\text{return } \perp] \leq 0.5$. To see this, note that there are *two* bad events.
  - $\mathcal{B}_1$. The point p is never hashed with x. More precisely, for all $1 \leq j \leq L$, $g_j(p) \neq g_j(x)$.
  - $\mathcal{B}_2$. Too many far-away points are hashed with x. To make this precise, let $F := \{p \in P : d(p,x) > cr\}$. This event occurs if $\sum_{j=1}^{L} f_j > 16L$ where f_j are the points in F that are in the same location of T_j as where x lands.

  Observe that $\mathbf{Pr}[\perp] \leq \mathbf{Pr}[\mathcal{B}_1 \cup \mathcal{B}_2]$. To see this, suppose neither bad events occur. Then, $\neg\mathcal{B}_1$ implies there exists j such that $g_j(p) = g_j(x)$, and therefore, at the end of the point collection, either $p \in B$ or $|B| = 16L$. $\neg\mathcal{B}_2$ implies that if $|B| = 16L$, then some point in B must be $\leq cr$ distance away, since the number of far away points that could end up in B is $\leq \sum_{j=1}^{L} f_j < 16L$. And so, we would be returning that point instead of $\perp$. It's important to note that we are *not* claiming $p \in B$ (otherwise, we would have found a point within distance r from x).

- We bound each $\mathbf{Pr}[\mathcal{B}_1]$ and $\mathbf{Pr}[\mathcal{B}_2]$ individually and this analysis explains the choice of k and L. As argued earlier, since $d(x,p) \leq r$, for each individual g_j, $\mathbf{Pr}[g_j(x) = g_j(p)] \geq p_1^k$. Therefore,

  $$\mathbf{Pr}[\mathcal{B}_1] \leq \mathbf{Pr}[\forall 1 \leq j \leq L \ : \ g_j(x) \neq g_j(p)] \leq \left(1 - p_1^k\right)^L \leq e^{-Lp_1^k}$$

  We would want this to be $\leq 1/e$, and thus $Lp_1^k \geq 1$ explaining our choice of L (in terms of k).

  To argue about $\mathcal{B}_2$, let's fix a point $q \in F$ and a j between 1 and L. Let $X_{q,j}$ be the indicator that q lies in $\jmath(g_j(x))$th position of T_j. Note that $\mathcal{B}_2$ implies that $\sum_{q \in F} \sum_{j=1}^{L} X_{q,j} \geq 16L$. Next note

  $$\mathbf{Pr}[X_{q,j} = 1] = \mathbf{Pr}[\jmath(g_j(q)) = \jmath(g_j(x))] < \mathbf{Pr}[g_j(q) = g_j(x)] + \frac{1}{n} \leq p_2^k + \frac{1}{n}$$

  where the $1/n$ follows from the UHF-ness of $\jmath$, and, since $d(q,x) > cr$, $\mathbf{Pr}[g_j(q) = g_j(x)] \leq p_2^k$.

  Therefore,

  $$\mathbf{Exp}[\sum_{q \in F} \sum_{j=1}^{L} X_{q,j}] \leq |F| \cdot L \cdot \left(p_2^k + \frac{1}{n}\right) < L + Lnp_2^k$$

  where we have used the trivial inequality $|F| < n$. Now we see the choice of k: we want $np_2^k \leq 1$ which is what $k = \lceil \ln n / \ln(1/p_2) \rceil$ gives us. With this, we get

  $$\mathbf{Exp}[\sum_{q \in F} \sum_{j=1}^{L} X_{q,j}] \leq 2L \ \Rightarrow \ \mathbf{Pr}[\mathcal{B}_2] = \mathbf{Pr}[\sum_{q \in F} \sum_{j=1}^{L} X_{q,j} \geq 16L] < \frac{1}{8}$$

  by Markov's inequality.

  Thus, $\mathbf{Pr}[\perp] \leq \mathbf{Pr}[\mathcal{B}_1] + \mathbf{Pr}[\mathcal{B}_2] \leq \frac{1}{e} + \frac{1}{8} < 0.5$.

In the remainder of these notes we look at three LSH families for certain metric spaces (X, d). In fact, all three constructions are stronger than what is needed; we show construction of a family $\mathcal{H}$ of functions such that for any two points

$$x, y \in X, \; \mathop{\mathbf{Pr}}_{h \in \mathcal{H}}[h(x) = h(y)] = 1 - \frac{d(x,y)}{D} \tag{*}$$

where $D = \max_{x,y \in X} d(x,y)$.

Exercise: *Show that any family of functions satisfying (*) is an (r,c)-LSH family for any r and c satisfying $rc < D$, with $\rho \leq \frac{1}{c}$.*

- **LSH for Hamming Metric.** Suppose $X = \Sigma^m$ where Σ is an alphabet and X is the set of all m-length strings with characters from this alphabet. For $x, y \in X$, let $d_{\mathsf{Ham}}(x,y)$ denote the *Hamming Distance* indicating the number of coordinates at which x and y differ. That is,

  $$d_{\mathsf{Ham}}(x,y) := \Big|\{i \in [m] \; : \; x_i \neq y_i\}\Big|$$

  The LSH family for this (X, d_{Ham}) is rather simple. It's simply

  $$\mathcal{H} := \{h_i : X \to \Sigma \mid 1 \leq i \leq m\}, \qquad \forall i \in [m], h_i(x) = x_i$$

  That is $\mathcal{H}$ contains m function where the ith function simply outputs the ith character of the string. Note that this family satisfies (*) with $D = m$.

  As a corollary one gets

  Theorem 1 (c-ANN for Hamming metrics)**.** *Given a subset $P \subseteq \Sigma^m$ with $|P| = n$ for any parameter $c > 1$, there exists a randomized data-structure with space $\widetilde{O}(n^{1+\frac{1}{c}})$ which, when given a query string x, can return an $O(c)$-approximately nearest neighbor in P in time $\widetilde{O}(n^{\frac{1}{c}})$.*

  The $\widetilde{O}()$ notation hides logarithmic factors. By querying every point $x \in P$ (to be fair, we need to delete x itself from P, but all that can be done), in $O(n^{1+1/c})$-time we can get an $O(c)$-approximation to the closest pair of strings. This is, at the time of writing and to the best of my knowledge, still the fastest time to get any constant approximation. In particular, to obtain an $O(1)$ approximation to the closest pair of strings in $\widetilde{O}(n)$ time is still an open question!

  There are many follow up results on LSH families for $(\mathbb{R}^m, \ell_1)$ and $(\mathbb{R}^m, \ell_2)$ metrics. At some point we will add pointers to these in this lecture notes, but for now we point to a recent (at least at the time of writing) survey talk by Moses Charikar.

- **LSH for Jaccard Distance.** We now state an LSH family which pre-dates the definition of LSH, and in fact possibly inspired the above definition. This is a hashing family due to Andrei Broder from a paper which was interested in trying to quickly figure out similarities between documents, which were (are?) thought of as a set of objects where each object is a consecutive string of words.

  Abstractly, let U be a universe of n elements. The set X is the power set of U, that is, the collection of all subsets of U. Given $A, B \subseteq U$, we define the Jaccard similarity between A and B, and the Jaccard distance

  $$\mathsf{JS}(A,B) := \frac{|A \cap B|}{|A \cup B|} \quad \text{and} \quad d_{\mathsf{Jacc}}(A,B) := 1 - \mathsf{JS}(A,B)$$

Exercise: ◕ *Prove that* d_{Jacc} *satisfies triangle inequality. Try proving this before reading ahead. You will appreciate the "book proof" of this fact that's coming up.*

- *The Family.* Suppose π is a permutation of U picked uniformly at random from all permutations. We imagine $\pi : U \to [n]$ where $\{\pi(x) \ : \ x \in U\}$ forms a random permutation of $\{1, 2, \ldots, n\}$. Given a subset $A \subseteq U$,define

$$h_\pi(A) := \min_{a \in A} \pi(a)$$

The following observation shows that this family satisfies (*). This hashing scheme is called $\mathsf{MinHash}$ in the field.

Claim 1. For any two sets A and B, $\mathbf{Pr}_\pi[h_\pi(A) = h_\pi(B)] = \mathsf{JS}(A, B) = 1 - d_{\mathsf{Jacc}}(A, B)$.

Proof. This is simply because π restricted to any fixed set is a distribution over random permutations of that set, and that every element of a permutation is equally likely to be the minimum. $h_\pi(A) = h_\pi(B)$ occurs if and only if the first element of the random permutation of $A \cup B$ lies in $A \cap B$. And this probability is precisely the ratio of the sizes of these two sets. □

Indeed, the above claim gives a simple proof that $d_{\mathsf{Jacc}}(A, B)$ satisfies triangle inequality. Let $\mathbf{1}_\pi(A, B)$ be the indicator variable if $h_\pi(A) \neq h_\pi(B)$; thus, $d_{\mathsf{Jacc}}(A, B)$ is precisely $\mathbf{Exp}_\pi[\mathbf{1}_\pi(A, B)]$. Now since these indicators are $0, 1$ random variables, it's next to trivial to argue for *any* π,

$$\mathbf{1}_\pi(A, B) \leq \mathbf{1}_\pi(A, C) + \mathbf{1}_\pi(C, B)$$

Indeed, the only way this can be violated is if the LHS is 1 and both RHS are 0. But both RHS are 0 implues $h_\pi(A) = h_\pi(C) = h_\pi(B)$ which would imply the LHS is 0 too. Thus, the above inequality holds and the triangle inequality of d_{Jacc} follows by taking expectations on both sides.

- *Min-Wise Independent Family of Functions.* The big issue with the above family is that π needs to be a permutation picked uniformly at random. The size of storing h_π is therefore $O(n \log n)$ which is undesirable. Motivated by this, Broder, Charikar, Frieze, and Mitzenmacher[6] defined the notion of ε-approximate min-wise independent family of functions.

Definition 1. A family $\mathcal{H}$ of functions $h : [n] \to [n]$ is said to be ε-approximate min-wise independent if for *any* $X \subseteq [n]$ and for any $x \in X$,

$$\mathop{\mathbf{Pr}}_{h \in \mathcal{H}}[h(x) = \min_{y \in X} h(y)] \in (1 \pm \varepsilon) \cdot \frac{1}{|X|}$$

Exercise: *Prove that for any two sets* A *and* B*, if we define* $h(A) := \min_{a \in H} h(a)$*, then* $\mathbf{Pr}_{h \in H}[h(A) = h(B)] \in \mathsf{JS}(A, B) \pm \varepsilon$.

[6]Andrei Z. Broder, Moses Charikar, Alan M. Frieze, Michael Mitzenmacher: Min-Wise Independent Permutations. J. Comput. Syst. Sci. 60(3): 630-659 (2000)

Broder et al. showed that there exist ε-approximate min-wise independent families of size $O(n^2/\varepsilon^2)$. Unfortunately, this is only an "existence" result shown using the probabilistic method. Indyk[7] showed that in fact a ℓ-wise independent family of hash-functions satisfies the above condition (but only for smallish sets X with $|X| \leq \varepsilon n$) when $\ell \approx \log(1/\varepsilon)$.

- **LSH for Angular Distance.** The final distance we look at is called *angular distance*. Once again, assume our data is m-dimensional vectors. We use the *angle* between these vectors as our notion of similarity/distance. In particular, assume by scaling that all the vectors $\mathbf{v}_i$'s lie on the unit sphere, that is, $\|\mathbf{v}_i\|_2 = 1$. We define

$$d_{\mathsf{ang}}(\mathbf{v}_i, \mathbf{v}_j) := \frac{\angle \mathbf{v}_i, \mathbf{v}_j}{\pi}$$

where the angle is measured in radians.

The RHS above is closely related to the dot product between the two vectors. Indeed, since the vectors are unit vectors $d_{\mathsf{ang}}(\mathbf{v}_i, \mathbf{v}_j) = \cos^{-1}(\mathbf{v}_i \cdot \mathbf{v}_j)$. One may wonder if choosing $1 - \mathbf{v}_i \cdot \mathbf{v}_j$ may not be a better choice for a distance measure...but this doesn't even satisfy triangle inequality.

Exercise: *Come up with three unit vectors $\mathbf{v}_i, \mathbf{v}_j, \mathbf{v}_k$ such that $(1 - \mathbf{v}_i \cdot \mathbf{v}_j) > (1 - \mathbf{v}_i \cdot \mathbf{v}_k) + (1 - \mathbf{v}_k \cdot \mathbf{v}_j)$.*

Hint: $\|\mathbf{v}_i - \mathbf{v}_j\|_2^2 = \|\mathbf{v}_i\|_2^2 + \|\mathbf{v}_j\|_2^2 - 2\mathbf{v}_i \cdot \mathbf{v}_j = 2(1 - \mathbf{v}_i \cdot \mathbf{v}_j)$ since the vectors are unit-vectors.

Do the angles satisfy triangle inequality? It's not obvious. See here for an argument on Math Stack-exchange. What is coming up is another proof a la Jaccard distance in the previous bullet point.

- *The Family.* Let $\mathbf{r}$ be a *random* unit vector in m dimensions. To define this precisely, we need to recall normal/Gaussian random variables. A standard Gaussian random variable $Z \sim N(0,1)$ with mean 0 and standard deviation 0 has a probability distribution function of $f(z) = \frac{1}{\sqrt{2\pi}} e^{-z^2/2}$. Construct $\mathbf{r}$ by choosing each $\mathbf{r}_i$ independently as a standard Gaussian. Note that for any point $\mathbf{x} = (x_1, \ldots, x_m)$, the

$$\mathbf{Pr}[\mathbf{r} = \mathbf{x}] = \frac{1}{(2\pi)^{m/2}} e^{-\|\mathbf{x}\|_2^2/2}$$

and thus is equiprobable if $\|\mathbf{x}\| = \|\mathbf{y}\|$. Therefore, scaling $\mathbf{r}$ by $\|\mathbf{r}\|$ gives a point uniform on the unit sphere.

Given $\mathbf{r}$, we define the following hash function.

$$h_{\mathbf{r}}(\mathbf{v}) := \operatorname{sgn}(\mathbf{v}_i \cdot \mathbf{r})$$

where $\operatorname{sgn}(x) = 1$ if $x \geq 0$ and -1 otherwise.

The following observation shows that this family satisfies (*). This hashing scheme is called $\mathsf{SimHash}$ in the field. As in the case of MinHash, the claim below proves that d_{ang} satisfies triangle inequality.

Claim 2. For any two unit vectors $\mathbf{v}$ and $\mathbf{w}$, $\mathbf{Pr}_{\mathbf{r}}[h_{\mathbf{r}}(\mathbf{v}) = h_{\mathbf{r}}(\mathbf{w})] = 1 - d_{\mathsf{ang}}(\mathbf{v}, \mathbf{w})$

[7] Piotr Indyk, A Small Approximately Min-Wise Independent Family of Hash Functions, Journal of Algorithms, 2000

Proof. Fix the plane spanned by $\mathbf{v}$ and $\mathbf{w}$. Let $\mathbf{r}'$ be the projection of $\mathbf{r}$ on to this plane which is dilated so as to lie on the unit (great) circle on which $\mathbf{v}$ and $\mathbf{w}$ lie. Note, $\mathrm{sgn}(\mathbf{v}, \mathbf{r}) = \mathrm{sgn}(\mathbf{v}, \mathbf{r}')$ and $\mathrm{sgn}(\mathbf{w}, \mathbf{r}) = \mathrm{sgn}(\mathbf{w}, \mathbf{r}')$. Also note that due to the fact that $\mathbf{r}$ was uniform on the sphere, $\mathbf{r}'$ is uniformly distributed on the circumference of the great circle. Therefore, the probability $\mathrm{sgn}(\mathbf{v}, \mathbf{r}') \neq \mathrm{sgn}(\mathbf{w}, \mathbf{r}')$ is the probability that $\mathbf{r}'$ lies in the minor arc between $\mathbf{v}$ and $\mathbf{w}$ (or $-\mathbf{v}$ and $-\mathbf{w}$). This is precisely $\frac{\angle \mathbf{v}_i, \mathbf{v}_j}{\pi}$ proving the claim. $\square$

As in the case of MinHash, it takes space to store m Gaussian random variables up to high precision. Indeed, as described, the family $\mathcal{H}$ of functions is infinite in size. It can be shown, however, if there are only n vectors, then the hash functions can be chosen using only $O(\log^2 n)$ bits, that is, the family $\mathcal{H}$ contains only $2^{O(\log^2 n)}$ different $\mathbf{r}$'s. This is connected to deterministic algorithms for the Johnson-Lindenstrauss lemma, which is the subject of the next lecture, and is connected to the study of *psuedorandom generators* and is beyond the scope of today's lecture.