We look at a hash scheme used to compute approximate nearest neighbors quickly in large, high-dimensional databases.

# 1 $p$-Stable distributions

Before turning to the nearest neighbor problem, we study the notion of $p$-stability. This property will be crucial to our hashing scheme and is a major ingredient in the proof of the Johnson-Lindentrauss lemma.

**def 1.** *Let $v \in \mathbb{R}^d$, and suppose $Z, X_1, \ldots, X_d$ are drawn iid from a distribution $\mathcal{D}$. Then $D$ is* $p$-stable *if* $\langle v, X \rangle \stackrel{\mathrm{d}}{=} \|v\|_p Z$.

It is known that $p$-Stable distributions exist for all $p \in (0, 2]$. Two examples are:

- The standard Cauchy distribution, with pdf $f(x) = \frac{1}{\pi(1+x^2)}$, is 1-stable.
- $\mathcal{N}(0,1)$—the standard Gaussian distribution—is 2-stable. This is quite easy to see:

  *Proof.* Let $v \in \mathbb{R}^d$, and $g_1, \ldots, g_d$ be iid samples from $\mathcal{N}(0,1)$. Then $\langle v, g \rangle$ is a linear combination of iid Gaussians, so is Gaussian. Let us compute the mean and variance.

$$\mathbb{E}\left[\langle v, g \rangle\right] = \sum v_i \mathbb{E}\left[g_i\right] = 0.$$
$$\mathbb{E}\left[\langle v, g \rangle^2\right] = \sum_{i \neq j} \mathbb{E}\left[v_i v_j g_i g_j\right] + \sum_i \mathbb{E}\left[v_i^2 g_i^2\right] = \sum_{i \neq j} v_i v_j \mathbb{E}\left[g_i\right]\mathbb{E}\left[g_j\right] + \sum_i v_i^2 = 0 + \|v\|_2^2.$$

  Thus $\langle v, g \rangle$ is $\mathcal{N}(0, \|v\|_2^2)$. □

The majority of the $p$-stable distributions do not have a simple closed-form pdf, but can be sampled from efficiently.

We can use $p$-stable distributions is to approximate the norms of high-dimensional vectors efficiently, or put differently, to reduce the dimension of high-dimensional points while preserving norms. We saw an example of this in the proof of the Johnson-Lindenstrauss lemma. There, we used a matrix of Gaussians to reduce the dimensionality while still approximately preserving the norms.

# 2 The nearest neighbor problem

We consider the nearest neighbor problem in $\mathbb{R}^D$, with the metric given by the $\ell_p$ norm, where $p \in (0, 2]$. Given a database of points $X \subset \mathbb{R}^D$, we hope to build a data structure that enables us to locate the $x \in X$ nearest to a provided query point $q \in \mathbb{R}^D$ quickly. Many researchers have derived data structures for this problem, but unfortunately all of them require either space or time (or both) that is exponential in the dimension.[1]

Since the problem of exact nearest neighbor search seems difficult, we look instead at an approximate version. To ease notation, let $x_q$ be the nearest neigbor to a query $q$ in $X$.

[1] Note that some of these schemes are only exponential in the doubling dimension of $X$, not $D$.

**def 2** ($\epsilon$NN)**.** *Suppose $q$ is a query point with nearest neighbor $x_q \in X$. Then a point $x \in X$ is an $\epsilon$-approximate nearest neighbor ($\epsilon$NN) if $\|q - x\|_p \leq (1 + \epsilon)\|q - x_q\|_p$.*

The major result of this paper is a polynomial-sized data structure that allows sublinear time retrievals of approximate nearest neighbors. It will be cleaner to work with an even easier problem.

**def 3** ($(R, \epsilon)$-NN)**.** *Let $q$ be a query point. If there is a point $x \in X$ such that $\|q - x\|_p \leq R$, then any point within distance $(1 + \epsilon)R$ of $q$ is called a $(R, \epsilon)$-near neighbor of $q$.*

Note that a $(R, \epsilon)$-NN is guaranteed to be an $\epsilon$NN only if $R = \|q - x_q\|_p$. Finding $(R, \epsilon)$-NNs seems to be an easier problem since we only need to search for nearby points at a fixed scale. However, it is shown in [IM98] that the $\epsilon$NN problem reduces to $O(\log(n/\epsilon))$ instances of the $(R, \epsilon)$-NN problem. Thus an efficient solution to the $(R, \epsilon)$-NN problem yields an efficient solution to $\epsilon$NN. We focus on the $(R, \epsilon)$-NN problem for the remainder of this paper.

# 3 Locality sensitive hashing

Locality sensitive hashing (LSH) is a hashing framework for the $(R, \epsilon)$-NN problem. Roughly, we wish to design a function $h$ with the following properties.

- If $x$ is a $(R, \epsilon)$-NN of $q$, then $h(x) = h(q)$.
- If $x$ is not a $(R, \epsilon)$-NN of $q$, then $h(x) \neq h(q)$ (for most $x$).
- $h$ requires polynomial storage and can be computed quickly.

These properties are a bit much to hope for, but we will be able to show corresponding statements that hold with some probability.

The rough idea behind the LSH scheme is to first project all of the data down to $O(\log n)$ dimensions and then to lay a grid down over this subspace. The hypercubes defined by the grid give us our bins.

More precisely, we setup our hashing scheme as follows.

1. Pick $d = O(\log n)$ random directions $g_1, \ldots, g_d \in \mathbb{R}^D$ where $g_{ij}$ is chosen independently from $\mathcal{D}_p$.
2. Pick a bin width $w$.
3. For each direction $i$, pick an offset $o_i$ uniformly at random from $[0, w]$.
4. The hash value for vector $x$ is
$$h(x) = \left( \left\lfloor \frac{\langle g_1, x \rangle + o_1}{w} \right\rfloor, \quad \cdots \quad , \left\lfloor \frac{\langle g_d, x \rangle + o_d}{w} \right\rfloor \right).$$
5. We will need $L$ such structures ($L$ to be defined later): $h_1, \ldots, h_L$.

With this structure built for our database $X$, let us see how to use it on query $q$.

1. Compute $h_1(q), \ldots, h_L(q)$.
2. For $i = 1$ to $L$, retrieve the database points falling into the bin $h_i(q)$.
3. Compute the actual distances to these points. If we find an $x$ with $\|x - q\|_p \leq R$, return it. Otherwise, stop after we have tried $2L$ points.

### 3.1 Analysis

First, let us consider only a single direction $g$ with offset $o$. What is the probability that two points $x$ and $q$ fall into the same bin along $g$? To simplify notation, let $c = \|x - q\|_p$, and $f(\cdot)$ be the pdf of the absolute value of the $p$-stable distribution.

We wish to calculate the probability of a collision—*i.e.*

$$\mathbb{P}\left[\left\lfloor \frac{\langle g, x\rangle + o}{w} \right\rfloor = \left\lfloor \frac{\langle g, q\rangle + o}{w} \right\rfloor\right].$$

We get a collision if both $|\langle x, g\rangle - \langle q, g\rangle| < w$ and a divider does *not* fall between $\langle x, g\rangle$ and $\langle q, g\rangle$. We can rewrite the first condition as

$$|\langle x - q, g\rangle| < w.$$

Because of $p$-stability, this condition is equivalent to

$$|\|x - q\|Z| \equiv |cZ| < w, \text{ where } Z \sim \mathcal{D}_P.$$

The probability that the divider falls between $\langle x, g\rangle$ and $\langle x, q\rangle$ is simply $|\langle x - q, g\rangle|$ divided by the bin width $w$. Thus the probability of collision is

$$p(c) = \int_{r=0}^{w/c} f(r)\left(1 - \frac{rc}{w}\right) dr.$$

Let us change the variable of integration to $t \equiv rc$, yielding

$$p(c) = \int_{t=0}^{w} \frac{1}{c} f\left(\frac{t}{c}\right)\left(1 - \frac{t}{w}\right) dt.$$

Notice that this probability depends only on the distance $c$ and is monotonically decreasing in $c$.

We focus on two crucial values of $p$:

$p_1 \equiv p(R)$ (a good collision)
$p_2 \equiv p(R(1+\epsilon))$ (a bad collision).

Note that $p_1 > p_2$.

Now, let us look at the data structure as a whole. The only property we need from the previous discussion is that $p_1 > p_2$.

**Claim 1.** *Suppose that there is a $x^* \in X$ such that $\|q - x^*\| \le R$. Then with constant probability*

1. *for some $i \in \{1, \ldots, L\}$, $h_i(x^*) = h_i(q)$; and*
2. *the total number of collisions with points $x \in X$ such that $\|q - x\| > R(1+\epsilon)$ is bounded by $2L$.*

*Proof.* We show that each property holds with probability greater than $1/2$, giving the theorem. Let us consider (1) first. For a fixed $i$,

$$\mathbb{P}\left[h_i(x^*) = h_i(q)\right] \ge p_1^d.$$

Using $d = \log_{1/p_2} n$, we get

$$p_1^d = p_1^{\log_{1/p_2} n} = n^{-\frac{\log 1/p_1}{\log 1/p_2}}.$$

Set $\rho = \frac{\log 1/p_1}{\log 1/p_2}$. Now the probability that we collide for *some* $i$ is

$$\mathbb{P}\left[\exists i : h_i(x^*) = h_i(q)\right] \geq 1 - (1 - n^{-\rho})^L.$$

Now set $L = n^\rho$. Then the probability is

$$1 - (1 - n^{-\rho})^{n^\rho} \geq 1 - \frac{1}{e} > \frac{1}{2}.$$

Now let us consider (2). Suppose $x' \in X$ is such that $\|q - x'\| > (1 + \epsilon)R$. Then,

$$\mathbb{P}\left[h_i(q) = h_i(x')\right] \leq p_2^d = p_2^{\log_{1/p_2} n} = \frac{1}{n}.$$

Thus the expected number of collisions for a particular $i$ is at most 1 and the expected total number of collisions is at most $L$. By Markov's inequality,

$$\mathbb{P}\left[\#x' \text{colliding with } q \quad > 2L\right] < \frac{L}{2L} = \frac{1}{2}.$$

□

### 3.2 Time complexity

On query $q$, we must

- compute the projections, which takes time $O(L \cdot d \cdot D) = O(n^\rho \cdot D \cdot \log n)$; and
- compute the distances to at most $2L$ points, requiring time $O(L \cdot D) = O(n^\rho \cdot D)$.

Thus the key quantity is $n^\rho$, where again $\rho = \frac{\log(1/p_1)}{\log(1/p_2)}$. This quanitity depends on the distribution used and the value of $\epsilon$. Since calculating $p_1$ and $p_2$ requires evaluating a cdf, in many cases we must numerically compute these values. For the case of the $\ell_2$ norm and the Gaussian distribution, $\rho$ is roughly $1/1 + \epsilon$. Thus we get a sublinear (in $n$) time complexity for any $\epsilon > 0$.

## References


[DIIM04] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG '04: Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, New York, NY, USA, 2004. ACM Press.

[IM98] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *STOC '98: Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, New York, NY, USA, 1998. ACM Press.