IT University of Copenhagen
Johan Siversten

IT UNIVERSITY OF COPENHAGEN

Lecture notes - Locality Sensitive Hashing

Lecture: February 22, 8:00–10:00

0.1 The Nearest Neighbor problem

Perhaps the most central problem in similarity search is the *nearest neighbor* (NN) problem. Let $D(x,y)$ be the distance between x and y.

Nearest Neighbor (NN) Given $S \subseteq X$ and $q \in X$. Return $x \in S$, such that $D(q,x)$ is minimized.

As we will discuss in the lecture, this problem is difficult(impossible) to solve exactly in polynomial time and space. To approach it we relax the problem in two ways: First, we will accept an answer x' if it is a c-approximate nearest neighbor. That is, we will require only that $D(q,x') \leq cD(q,x)$, where $x \in S$ is the actual nearest neighbor. Since we are searching for similar, but not necessarily equal things, the most similar and the *almost* most similar will often be equally useful.
We can also consider cases where the similar thing is much closer (more than a factor c) to the query than the rest of the dataset. In such settings the returned c-approximate nearest neighbor is also the actual nearest neighbor.
Secondly we will allow the distance, r, to be a parameter to the problem. We say that x' is r-near if $D(q,x') \leq r$. The relaxed approximate near neighbor problem (ANN) is then stated as:

Definition 1 ((c,r)-Approximate Near Neighbor)**.** For $c > 1$, $r > 0$. If there exists a point $x \in S$ such that $D(x,q) \leq r$, report some point $x' \in S$ where $D(x',q) \leq cr$, otherwise report nothing.

These relaxations were first introduced by Indyk and Motwani in [8]. They also show that we can use (c,r)-approximate near neighbor to find the c-approximate nearest neighbor by searching over settings of r. In many applications achieving a fixed similarity might also suffice on its own, regardless of the existence of closer points.

0.2 Locality Sensitive Hashing

Locality Sensitive Hashing(LSH) is the current state of the art for solving the ANN problem(Definition 1). The technique was first introduced by Indyk, Gionis and Motwani [8, 6] with an implementation that is still the best know for Hamming space. Since then it has been a subject of intense research. See [1] for an overview. The basic idea is to partition the input data using a hash function, H, that is sensitive to the metric space location of the input. This means that the collision probability is larger for inputs close to each other than for inputs that are far apart. This requirement is normally formalized as:

$$\Pr[H(u) = H(v)] \begin{cases} \geq P_1 \text{ when } D(u,v) \leq r \\ \leq P_2 \text{ when } D(u,v) \geq cr \end{cases} \tag{1}$$

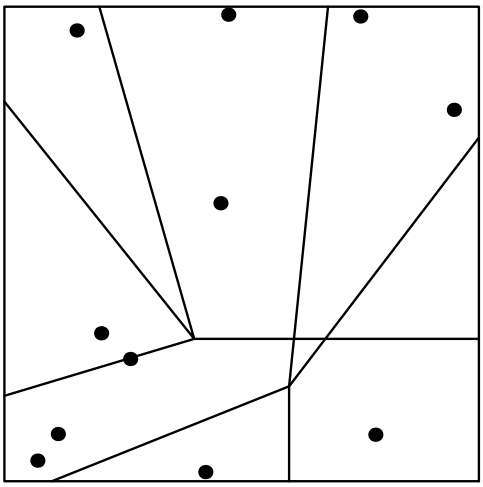
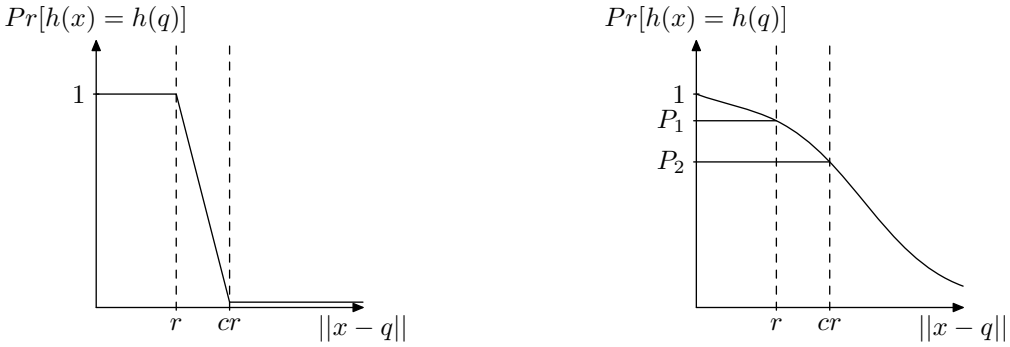


Figure 1: A non-perfect partitioning of points in $\mathbb{R}^2$



(a) Ideal (b) Achievable

Figure 2: Ideal vs. achievable LSH function.

where $P_1 > P_2$. So the points in S close to q are more likely to hash to the same location as q under H than other points. The key to success for this method is in achieving a large gap between P_1 and P_2, quantified as $\rho = \frac{\ln P_1}{\ln P_2}$(See Figure 2). Ideally, P_2 would be 0, for example by the hash function returning the cell of the voronoi diagram associated with a point. But that would trap the function in the curse of dimensionality, either taking up too much space or time. So instead we use several functions that each return imperfect partitioning, as illustrated in Figure 1, but are fast to evaluate.
Using a hash function with these properties the (c,r)-ANN problem can be solved using $n^{1+\rho+o(1)}$ extra space with $dn^{\rho+o(1)}$ query time [7]. The techniques in [8, 6] extend to ℓ_1 with $\rho = 1/c$, for ℓ_2 a result with $\rho = 1/c^2$ can be found in [2]. Recently lower bounds have been published on ρ for the ℓ_1 [7] and ℓ_2 [9] norm showing these upper bounds to be essentially tight[1].

0.3 Distance functions and similarity measures

We will be formalizing "similarity" through the inverse notion of distance. Given a point in space, similar things will be close to each other, differing things far away. But we will also sometimes use

[1] In the setting where we cannot look at the data before choosing our hash function.

direct similarity measures. Table 1 some common distance functions and similarity measures we will be using throughout.

The distance functions are central in geometry, dating back to the ancient Greeks. Most commonly used are the ℓ_p norms, in particular the Euclidean distance ℓ_2. For a thorough discussion of the ℓ_p norms we refer to [11].

The similarity measures originated in biology where they where developed to compare subsets of a bounded set, like $[d]$ or the set of all flowers. The most common is Jaccard similarity and variations of it like the Braun-Blanquet variation. These measures range between 0 and 1, with 0 being no common elements and 1 being duplicate sets.

The odd space out is the Hamming space. We could define the Hamming similarity as $(d - H(x,y))/d$, but it is standard in the literature to use Hamming distance.

0.4 Minwise Hashing

Let S_n be the set of all permutations of $[n]$. We say that a family of permutations $\mathscr{F} \subseteq S_n$ is min-wise independent if for any $X \subseteq [n]$ and any $x \in X$, when π is chosen at random from $\mathscr{F}$ we have

$$\Pr[\min(\pi(X)) = \pi(x)] = \frac{1}{|X|} \ .$$

That is, every element of X is equally likely to permute to the smallest value. We call $\mathscr{H}$ a familiy of MINHASH functions if for a random $h \in \mathscr{H}$, $h(X) = min(\pi(X))$ where π is a random permutation from a min-wise independent family of permutations.

MINHASH functions are very useful in Set Similarity because

$$\Pr[h(x) = h(y)] = \frac{|x \cap y|}{|x \cup y|} = J(x,y) \ .$$

Let $X_i = 1$ if $h_i(x) = h_i(y)$ and 0 otherwise. A Chernoff bound tells us that if $X = \frac{1}{t}\sum_i^t X_i$,

$$\Pr[|X - J(x,y)| \geq \sqrt{\frac{3\ln t}{t}} J(x,y)] \leq 2e^{-J(x,y)\ln t} = \frac{2}{t^{J(x,y)}} \ . \tag{2}$$

Name	Input	Distance function
Hamming distance	$x,y \in X^d$	$H(x,y) = \sum_i^d \begin{cases} 1 \text{ if } x_i = y_i \\ 0 \text{ else} \end{cases}$
Minkowski distance	$x,y \in X^d$	$\ell_p(x,y) = \left(\sum_i^d \lvert x_i - y_i\rvert^p\right)^{1/p}$
Euclidian distance	$x,y \in X^d$	$\ell_2(x,y) = \sqrt{\sum_i^d \lvert x_i - y_i\rvert^2}$
Jaccard similarity	$A,B \subseteq X$	$J(A,B) = \frac{\lvert A \cap B\rvert}{\lvert A \cup B\rvert}$
Braun-Blanquet similarity	$A,B \subseteq X$	$BB(A,B) = \frac{A \cap B}{max(\lvert A\rvert,\lvert B\rvert)}$

Table 1: Distance functions and Similarity measures

So we can get precise estimates of the Jaccard similarity from a small number of hash functions.
Of course the number of permutations of $[n]$ is $n!$ so in practice we allow $\mathscr{F} \subseteq S_n$ to be $\varepsilon-$*min-wise independent*:

$$\Pr[\min(\pi(X)) = \pi(x)] \in \frac{1 \pm \varepsilon}{|X|}$$

In practice we also want hash functions that are fast to evaluate and easy to implement. Zobrist hashing, or *simple tabulation hashing*, fits this description. It is $\varepsilon-$min-wise independent with ε shrinking polynomially in $|X|$ [10], $3-$independent and fast in practice [12]. Tabulation hashing works by splitting keys $x = (x_0, \cdots, x_{c-1})$ into c parts. Each part is treated individually by mapping it to $[M]$, say with a table of random keys$t_o, \cdots, t_{v-1} : \mathscr{U} \to [M]$. Finally $h : \mathscr{U}^c \to [M]$ is computed by:

$$h(x) = \oplus_{i \in [c]} t_i(x_i)$$

Where $\oplus$ denotes the bit-wise XOR operation.

References

[1] A. Andoni. *Nearest Neighbor Search: the Old, the New, and the Impossible*. PhD thesis, MIT, 2009.

[2] A. Andoni and P. Indyk. Near-Optimal Hashing Algorithms for Approximate Nearest Neighbor in High Dimensions. *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 459–468, 2006.

[3] A. Broder. On the resemblance and containment of documents. In *Proceedings of the Compression and Complexity of Sequences 1997*, SEQUENCES '97, pages 21–. IEEE Computer Society, 1997.

[4] A. Z. Broder, M. Charikar, A. M. Frieze, and M. Mitzenmacher. Min-wise independent permutations. *Journal of Computer and System Sciences*, 60(3):630–659, June 2000.

[5] J. Carter and M. N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143 – 154, 1979.

[6] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *Proceedings of 25th Conference on Very Large Data Bases (VLDB)*, pages 518–529, 1999.

[7] S. Har-Peled, P. Indyk, and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. *Theory of Computing*, 8(1):321–350, 2012.

[8] P. Indyk and R. Motwani. Approximate nearest neighbors: Towards removing the curse of dimensionality. *Proceedings of the 30th ACM Symposium on Theory of Computing (STOC)*, 8:321–350, 1998.

[9] R. O'Donnell, Y. Wu, and Y. Zhou. Optimal lower bounds for locality sensitive hashing (except when q is tiny). *ACM Transactions on Computation Theory*, 2014.

[10] M. Pătraşcu and M. Thorup. The power of simple tabulation hashing. *Journal of the ACM*, 59(3):14:1–14:50, June 2012.

[11] W. Rudin. *Real and complex analysis*. McGraw-Hill Book Company, 1987.

[12] M. Thorup. Fast and Powerful Hashing using Tabulation. *ArXiv e-prints*, May 2015.

[13] M. N. Wegman and L. Carter. New classes and applications of hash functions. In *Proceedings of the 20th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 175–182. IEEE Computer Society, 1979.