1 Introduction

We will start by considering the algorithmic task of finding similar items within a database. Variants of this problem occur in several natural settings such as:

- **Recommender Systems:** We want to find users that have similar buying patterns so we can recommend items to them
- **Data Imputation:** Given some missing entries in a database, we might want to fill them in based on their "nearest neighbor"
- **Document Collection:** Finding similar documents allows us to detect multiple versions of the same article, mirror websites, plagarism etc.

We will consider a special case of this general class of problems, called the closest pair problem.

2 Closest Pair Problem

2.1 Set-Up

In the closest pair problem, we are given n points in R^d, and we want to find the pair of points (p, q) with minimum distance. We will fix the distance we're using to to be the Euclidean distance, defined as

$$dist(p,q) = \left(\sum_{j=1}^{d} (p_j - q_j)^2 \right)^{1/2}$$

The brute force approach would involve iterating over all pairs of points, computing the distance, and outputting the minimum at the end. This takes $\Theta(n^2 d)$ time. This is not great when n is large, and also not great when d is large.

Recall that we've previously seen the $2-D$ version of the problem, where all points lie in a plane and we are given their (x, y) coordinates; we also saw an $O(n \log n)$ divide-and-conquer algorithm for this problem (in addition to an $O(n)$ randomized algorithm). These approaches can be generalized to d dimensions - however, the run-time will scale proportional to 2^d, which implies that this is only efficient when d is in $O(\log n)$.

We are instead interested in the case where d could be very large, and we want to focus on minimizing the dependence on d.

2.2 Embedding Paradigm

The key idea that we will use is to first reduce the dimension of the given points in a way that will allow us to approximately recover the distance between each pair. Once we have done this, we will then run the brute-force algorithm on the transformed points to find the closest pair and output it. That is, we perform the following three steps:

1. Choose a random $s \times d$ matrix S for a small value $s \ll d$

2. Replace the n points $p_1, \cdots, p_n \in R^d$ with n points $S \cdot p_1, \cdots, S \cdot p_n \in R^s$
3. Compute a function $f(S \cdot p_i, S \cdot p_j) \approx dist(p_i, p_j)$ between all pairs $S \cdot p_i$ and $S \cdot p_j$ and output the pair p_i and p_j for which $f(S \cdot p_i, S \cdot p_j)$ is minimal

The time complexity of the matrix multiplications in step 2 is $O(nd \cdot s)$. Assuming the function f can be computed in $O(s)$ time, the time complexity of the search in step 3 is $O(n^2 \cdot s)$. Assuming also that we can efficiently generate the matrix in the first step, this gives an overall time complexity of

$$O(nd \cdot s + n^2 \cdot s)$$

For example, if $n = d$ and $s = \Theta(\log n)$, we get an algorithm with $O(n^2 \log n)$ run-time, whereas our brute force algorithm would have taken $O(n^2 d) = O(n^3)$ time.

2.3 A Randomized Embedding

We will now more concretely specify the algorithm that outlined above. Our goal will be to output a pair of points such that the distance between them is within a factor of $(1 + \epsilon)$ of the overall minimum distance, where $\epsilon > 0$ is a constant accuracy parameter (which we can set to be as small as we want, depending on the level of precision we'd like).

We will let $s = O(\frac{1}{\epsilon^2})$. The matrix $S \in R^{s \times d}$ will be generated as follows: we pick each entry of S to be $1/\sqrt{s}$ with probability 1/2 and $-1/\sqrt{s}$ with probability 1/2. For a point $p \in R^d$, the vector $S \cdot p \in R^s$ is much lower dimensional.

Claim: $\mathbb{E}[|S \cdot p_2|_2^2] = |p|_2^2$

Proof: Let S_i be the i-th row of S. Since each row of S is identically distributed, $\mathbb{E}\left[|S.p_2|_2^2\right] = s \cdot \mathbb{E}[\langle S_1, p\rangle^2]$. Then

$$\mathbb{E}[\langle S_1, p\rangle^2] = \mathbb{E}\left[\left(\sum_{j=1}^{d} \sigma_j p_j\right)^2\right] = \sum_{j_1, j_2} \mathbb{E}[\sigma_{j_1}\sigma_{j_2}] \cdot p_{j_1} p_{j_2}$$

If $j_1 = j_2$, then $\mathbb{E}[\sigma_{j_1}\sigma_{j_2}] = 1/s$. Otherwise, $\mathbb{E}[\sigma_{j_1}\sigma_{j_2}] = \mathbb{E}[\sigma_{j_1}]\mathbb{E}[\sigma j_2] = 0$; the first equality follows from the independence of σ_{j_1} and σ_{j_2}, and the second equality comes from the fact that $\mathbb{E}[\sigma_i] = 0$ for all i. So $\mathbb{E}[\langle S_1, p\rangle^2] = \frac{|p|_2^2}{s}$. Therefore, $\mathbb{E}[|S \cdot p_2|_2^2] = |p|_2^2$.

Claim: $\mathbf{Var}[|S \cdot p|_2^2] = O(|p|_2^4)$

Proof:

$$\mathbf{Var}[|S \cdot p|_2^2] = \mathbb{E}[|S \cdot p|_2^4] - \mathbb{E}[|S \cdot p|_2^2]^2$$

Expanding out the first term,

$$\begin{aligned}
\mathbb{E}[|S \cdot p|_2^4] &= \mathbb{E}\left[\left(\sum_{i=1}^{s} \langle S_i, p\rangle^2\right)^2\right] \\
&= \sum_{i,i'} \mathbb{E}[\langle S_i, p\rangle^2 \langle S_{i'}, p\rangle^2] \\
&= \sum_{i} \mathbb{E}[\langle S_i, p\rangle^4] + \sum_{i \neq i'} \mathbb{E}[\langle S_i, p\rangle^2]\mathbb{E}[\langle S_{i'}, p\rangle^2] \text{ by independence of rows } i \text{ and } i'
\end{aligned}$$

Therefore,

$$
\begin{aligned}
\mathbf{Var}[|S\cdot p|_2^2] &= \mathbb{E}[|S\cdot p|_2^4] - \mathbb{E}[|S\cdot p|_2^2]^2 \\
&= \sum_i \mathbb{E}[\langle S_i, p\rangle^4] + \sum_{i\neq i'} \mathbb{E}[\langle S_i, p\rangle^2]\mathbb{E}[\langle S_{i'}, p\rangle^2] - \mathbb{E}[|S\cdot p|_2^2]^2 \\
&\leq \sum_i \mathbb{E}[\langle S_i, p\rangle^4] \\
&= s\cdot \mathbb{E}\left[\left(\sum_{j=1}^d \sigma_j p_j\right)^4\right] \\
&= s\cdot \sum \mathbb{E}[\sigma_{j_1}\sigma_{j_2}\sigma_{j_4}\sigma_{j_4}]\cdot p_{j_1}p_{j_2}p_{j_3}p_{j_4}
\end{aligned}
$$

For $\mathbb{E}[\sigma_{j_1}\sigma_{j_2}\sigma_{j_3}\sigma_{j_4}] \neq 0$, the set $\{j_1, j_2, j_3, j_4\}$ needs to have either 4 equal indices, or 2 pairs of equal indices. This is because any index that appears an odd number of times would cause the expectation to be 0 - for instance, if $j_1 = j_2 = j_3 \neq j_4$, then $\mathbb{E}[\sigma_{j_1}\sigma_{j_2}\sigma_{j_3}\sigma_{j_4}] = \mathbb{E}[\sigma_{j_1}^3]\mathbb{E}[\sigma_{j_4}] = 0\times 0 = 0$ (where we used the independence of σ_{j_1} and σ_{j_4} to break up the expectation of a product into the product of expectations). We will now consider the cases where $\mathbb{E}[\sigma_{j_1}\sigma_{j_2}\sigma_{j_3}\sigma_{j_4}] \neq 0$.

If $j_1 = j_2 = j_3 = j_4$, then $\mathbb{E}[\sigma_{j_1}\sigma_{j_2}\sigma_{j_4}\sigma_{j_4}] = 1/s^2$. The total contribution of these terms is then

$$
s\cdot\frac{1}{s^2}\cdot\sum_j p_j^4 \leq s\cdot\frac{1}{s^2}\cdot\left(\sum_j p_j^2\right)^2 = \frac{1}{s}\cdot|p|_2^4
$$

If $j_1 = j_2$ and $j_3 = j_4$, then $\mathbb{E}[\sigma_{j_1}\sigma_{j_2}\sigma_{j_4}\sigma_{j_4}] = 1/s^2$. The contribution of these terms is then

$$
s\cdot\frac{1}{s^2}\cdot\left(\sum_j p_j^2\right)^2 = \frac{1}{s}\cdot|p|_2^4
$$

The case $j_1 = j_3$ and $j_2 = j_4$, and the case $j_1 = j_4$ and $j_2 = j_3$, are exactly the same.

Therefore, by summing the contributions from all 4 cases, we get

$$
\mathbf{Var}[|S\cdot p|_2^2] = \frac{4}{s}|p|_2^4 = O(|p|_2^4)
$$

2.4 Applying Chebyshev's Bound

Recall Markov's inequality on random variables X:

$$
\mathbf{Pr}[X \geq c\mathbb{E}[x]] \leq \frac{1}{c}
$$

We can use Markov's inequality to derive a stronger bound known as Chebyshev's inequality, which states that

$$
\mathbf{Pr}[|X - \mathbb{E}[X]| \geq \lambda(\mathbf{Var}[X])^{1/2}] \leq \frac{1}{\lambda^2}
$$

Proof:

$$
\mathbf{Pr}[|X - \mathbb{E}[X]| \geq \lambda(\mathbf{Var}[X])^{1/2}] = \mathbf{Pr}[(X - \mathbb{E}[X])^2 \geq \lambda^2(\mathbf{Var}[X])] \leq \frac{1}{\lambda^2}
$$

Here, the final step comes from applying Markov's inequality.

We now apply Chebyshev's inequality to get

$$\mathbf{Pr}[|X - \mathbb{E}[X]| \geq \lambda(\mathbf{Var}[X])^{1/2}] \leq \frac{1}{\lambda^2}$$
$$\mathbf{Pr}[||S \cdot p|_2^2 - |p|_2^2| \geq \frac{20}{\sqrt{s}}|p|_2^2] \leq \frac{1}{100}$$
$$\mathbf{Pr}[||S \cdot p|_2^2 - |p|_2^2| \geq \epsilon|p|_2^2] \leq \frac{1}{100}$$

The last line follows by setting $s = \frac{400}{\epsilon^2}$

Thus, we have for any point p, $\mathbf{Pr}[||S \cdot p|_2^2 - |p|_2^2| \geq \epsilon|p|_2^2] \leq \frac{1}{100}$.

To compute $S(p_1 - p_2)$ for a pair of points p_1, p_2, notice that since we have Sp_1 and Sp_2 already computed, we can just compute $S(p_1 - p_2) = Sp_1 - Sp_2$. Therefore, we have for any pair of points,

$$\mathbf{Pr}[||S \cdot (p_1 - p_2)|_2^2 - |(p_1 - p_2)|_2^2| \geq \epsilon|(p_1 - p_2)|_2^2] \leq \frac{1}{100}$$

$$\mathbf{Pr}[||S \cdot (p_1 - p_2)|_2^2 - dist(p_1, p_2)^2| \geq \epsilon \cdot dist(p_1, p_2)^2] \leq \frac{1}{100}$$

This is great, but we're not done yet. Remember that we have $\binom{n}{2}$ pairs of points; taking the union bound won't be enough to guarantee that we succeed with any constant probability.

2.5 Amplifying the Probability

Instead, we generate $r = O(\log n)$ matrices $S_1, S_2, \ldots S_r$ independently. Then, for a pair of points p_1, p_2, we can take the median of $S_i(p_1 - p_2)$; in other words, we choose our function to be $f(p_1, p_2) = median_{i=1,2,\ldots,r}|S_i(p_1 - p_2)|_2^2$.

Previously, for any pair of points p_1, p_2, we had that $|S \cdot (p_1 - p_2)|_2^2 \in (1 \pm \epsilon)dist(p_1, p_2)^2$ with probability $\frac{99}{100}$. Now,

$$\mathbf{Pr}[|f(p_1, p_2) \cdot (p_1 - p_2)|_2^2 - dist(p_1, p_2)^2| \geq \epsilon \cdot dist(p_1, p_2)^2] \leq \frac{1}{n^3}$$

by picking an appropriate r. And thus, $f(p_1, p_2) \in (1 \pm \epsilon)dist(p_1, p_2)^2$ with probability $1 - \frac{1}{n^3}$.

Over $\binom{n}{2}$ pairs of points, by the union bound, we have $\forall i, j,\ f(p_i, p_j) \in (1 \pm \epsilon)dist(p_i, p_j)^2$ with probability $1 - \frac{1}{n}$. The time complexity overall is $O\left(nd \log\left(\frac{n}{\epsilon^2}\right) + n^2 \log\left(\frac{n}{\epsilon^2}\right)\right)$

2.6 Applications to Data Streams

Suppose we had a stream of elements from universe U where $|U| = u$ and a frequency vector f of length u where f_i is the number of times element i occurs in the stream. Now further suppose we want to compute $|f|_2^2 = \sum_i f_i^2$ which is called the *skew* of the stream. How could we approximate this?

We start by choosing a random $s \times u$ matrix S for $s = O(\frac{1}{\epsilon^2})$. We initialize a vector v of size s as the 0 vector. For each element in the stream, we update $v = v + S \cdot e_i$ where e_i is the i^{th} standard basis vector in R^u (i.e. an all zero vector except the i^{th} index). At the end, we have the guarantee that $|s|_2^2 \in (1 \pm \epsilon)|f|_2^2$ with high probability.

A further optimization could be applied where we generate the entries of S using a hash function h s.t. $S_{ij} = h(i, j) = \{-1, 1\}$, where h is drawn from a 4-universal hash family. This way, we only need to store the hash function in memory instead of the entire matrix, but we can still obtain the same guarantees as before. Further reading on how 4-universal hash functions can be expressed in $O(\log u)$ bits can be found at https://www.sciencedirect.com/science/article/pii/S0022000097915452.