22.1 Recap From Last Class

In the last class, we looked at Jaccard similarity as a metric to define similarity between two sets. Recall that Jaccard Similarity between two sets $A, B$ is given by $\text{SIM}(A, B) = \frac{|A \cap B|}{|A \cup B|}$. We then looked at the Characteristic matrix for a collection of sets and computing their min-hash values for a particular permutation $\pi$. We also looked at a theorem that relates min-hash values to Jaccard similarity. The theorem states that $\Pr\{\text{minhash}(S) = \text{minhash}(T)\} = \text{SIM}(S, T)$. We then looked at representing a set using a min-hash signature. Signature for a set $S$ is computed using $n$ min-hash functions, $\{h_1, h_2, \ldots, h_n\}$, each generated using an independent permutation, as $\text{SIG}(S) = [h_1(S), h_2(S), \ldots, h_n(S)]$.

From these signatures, we then constructed a signature matrix where rows are minhash functions and the columns are sets, with the value at a particular location $i, j$ being $h_j(S_i)$. We then looked at approximating $\text{SIM}(S, T)$ by the fraction of coordinates where $\text{SIG}(S)$ and $\text{SIG}(T)$ match.

We then looked at requirements for a Locality-Sensitive Hash (LSH) function. A good LSH function will divide inputs into a large number of buckets, and hashes similar inputs into the same bucket. Then, given a query item $q$, its nearest neighbors can be computed by just looking at all items in the same bucket as $\text{LSH}(q)$. We term these items as candidates. If $A$ and $B$ are similar, then we want $\text{hash}(A) = \text{hash}(B)$ with high probability. We also want fewer (1) false positives – items that are dissimilar getting hashed into the same bucket, and (2) false negatives – similar items are hashed into different buckets.

We consider construction of an LSH function designed for documents represented by shingle-sets and minhashed to short signatures. We start with the signature matrix and divide it up into $b$ bands of $r$ rows each. We then hash the columns in each bands using an off-the-shelf hash function to divide each band into buckets. If sets $S$ and $T$ have same values in a particular band, they’ll be hashed into the same bucket in the band-specific hashtable. We then use this to accelerate nearest neighbor queries by only considering items that are in the same bucket as the query item as candidates for expensive similarity computation.

22.2 Overview

In this lecture, we’ll analyze the above LSH function, then completely shift gears and look at two dimensionality reduction algorithms – Johnson-Lindenstrauss (JL) transform and
22.3 LSH Analysis

Let $Q, T$ be two sets with similarity $s = \text{sim}(Q, T)$. From previous results, we know that $\Pr\{h_j(Q) = h_i(T)\} = s = \text{sim}(Q, T)$. Let $b$ be the number of bands in the signature matrix, with $r$ rows in each band ($b \cdot r = n$). Then,

$$\Pr\{Q, T \text{ agree on all rows in a band}\} = s^r$$
$$\Pr\{Q, T \text{ disagree in at least one band}\} = (1 - s^r)$$
$$\Pr\{Q, T \text{ disagree in all bands}\} = (1 - s^r)^b$$

Using this, we get $p = \Pr\{T \text{ is chosen as a candidate for } Q\} = (1 - (1 - s^r)^b)$. Since our goal is to compare with only good candidates (in order to minimize computation), ideally, we would like to have candidates with high $s$ values to begin with. Plotting this probability against $s$, we get an $S$-curve (shown in slides) that allows us to evaluate quality of candidates produced by a scheme with $b$ bands of $r$ rows each. The curve plots the probability of choosing a set $T$ as a candidate for set $S$ ($p$) against the similarity between the two sets $s$. Notice that this curve is parameterized by $b, r$. The threshold for candidate selection is empirically set to be $s \approx \left(\frac{1}{b}\right)^\frac{1}{r}$. All sets above this threshold are deemed to be good candidates.

22.3.1 Tradeoffs

S-curves and thresholds are parameterized by $b, r$ with $n = b \cdot r$. Choosing a low threshold results in excessive computation, but better result quality as we evaluate more candidates. Conversely, choosing a high threshold gives us quick computation, but lower quality results. $b^*, r^*$ are chosen empirically to strike a balance between quality of results obtained and the amount of computation required. Once a set of candidates is chosen, higher precision, albeit more expensive, similarity computation can be used to further shrink the candidate set.

22.3.2 Summary

Using a min-hash based LSH, we can build a system to quickly find similar documents from a corpus, for a given document. Further, the system performance can be easily tuned to balance computational complexity with quality of results obtained.

22.4 Dimensionality Reduction

High-dimensional vectors are common in many real-world applications like information retrieval, product recommendation, web search, etc. Working with high-dimensional vectors is made difficult due to the curse of dimensionality. To overcome this, one prefers to work in
a low-dimensional space by transforming vectors into lower dimensions while retaining some useful properties.

In this lecture, we take a look at two such transformation techniques – (1) Johnson-Lindenstrauss (JL) Transform, and (2) Principal Component Analysis.

### 22.4.1 Johnson-Lindenstrauss (JL) Transform

JL transform is a linear transform. Given a vector $v$, its JL transform is computed as $Sv$, where $S$ is the JL transform matrix. JL transform preserves pairwise $\ell_2$ distances between the given input vectors. JL Lemma (stated and proved later) provides bounds on the quality of the preservation.

#### Lemma 22.1. (JL Lemma)

Let $\epsilon \in (0, \frac{1}{2})$. Given a set of points $X = \{x_1, x_2, \ldots, x_n\}$, with $x_i \in \mathbb{R}^D$, there exists a map $S : \mathbb{R}^D \to \mathbb{R}^k$ with $k = \mathcal{O}(\epsilon - 2 \log n)$ such that

$$(1 - \epsilon)||x_i - x_j||_2 \leq ||Sx_i - Sx_j||_2 \leq (1 + \epsilon)||x_i - x_j||_2$$

Two things are very peculiar about the JL transform-

- The projection space dimension, $k$, is independent of the original dimension, $D$.
- The projection space dimension depends only on the number of points, $n$, and an accuracy parameter, $\epsilon$ that determines the quality of the projection.

#### Construction

Let $M$ be a $k \times D$ matrix whose entries are drawn from $\mathcal{N}(0, 1)$. Then, the JL transform matrix $S$ is given by

$$S = \frac{1}{\sqrt{k}} M$$

Given a vector $v \in \mathbb{R}^D$, its transform is now computed as $Sv \in \mathbb{R}^k$. It is interesting to note that the JL transform is a simple matrix-vector multiplication, and does not take into account the set of input points $X$.

#### Proof:

We will assume the following lemma to be true without proof.

**Lemma 22.2.** Let $\epsilon \in (0, \frac{1}{2})$. If $S$ is constructed as above with $k = \mathcal{O}(\epsilon - 2 \log \delta - 1)$, and $x \in \mathbb{R}^D$ is a unit vector ($||x||_2 = 1$), then

$$\Pr\{||Sx||_2 \in (1 \pm \epsilon)\} \geq 1 - \delta$$

We’re done if we can choose a proper substitution for $\delta$. Setting $\delta = \frac{1}{n^2}$, makes $k = \mathcal{O}(\epsilon - 2 \log n)$.
From this lemma, $\forall x_i, x_j \in X$, we get that the squared length of unit vector $(x_i - x_j) \in (1 \pm \epsilon)$ with probability $\geq 1 - \frac{1}{n^2}$. We know that $S$ is a linear map $\implies S \cdot (\alpha x) = \alpha \cdot Sx$ for any constant $\alpha$. Therefore, $\|S(x_i - x_j)\|_2 \in (1 \pm \epsilon)\|x_i - x_j\|_2$ with probability $\geq 1 - \frac{1}{n^2}$.

There are a total of $\binom{n}{2}$ such pairs to which we can apply this bound, with each pair failing with a probability of $\frac{1}{2}$. By union bound, the probability that all pairs fail is $\leq \frac{n(n-1)}{n^2} = \frac{1}{2}$. Hence, the probability that all pairs succeed is $\geq 1 - \frac{1}{2} = \frac{1}{2}$. Thus, we’ve obtained a randomized construction algorithm for $S$ that succeeds with probability $\geq \frac{1}{2}$.

**Variants of JL**

There exist variants of the JL transform that fill the matrix entries from $\{-1, 1\}$ with equal probability, instead of using $\mathcal{N}(0, 1)$. As it turns out, the above lemma is applicable for such constructions as well with no modifications. In other work, people have explored variants that operate on sparse vectors to reduce computational complexity of the transform.

### 22.4.2 Principal Component Analysis

In the JL transform, we derived a transform that did not depend on the input or its structure. Although this is a benefit in cases where the structure in the input is not known ahead of time, it is a sub-optimal transform when there is some structure in the input. PCA approximates the dataset $X$ using vectors $U = \{u_1, u_2, \ldots, u_k\} \in \mathbb{R}^D$ such that every $x_i$ is close to the span of the vectors in $U$. PCA has been found to be particularly useful when the data is inherently low-dimensional and/or contains a small amount of noise that is independent of the data. The goal of PCA is to find $U$ i.e. $k$ orthonormal vectors such that the points in $X$ have a good approximation in the subspace generated by the vectors in $U$. We’re interested in obtaining an orthonormal set of vectors since they form a basis for the low-dimensional subspace span($U$). The quality of approximation is determined using the $\ell_2$ distance between the original vector and its approximation in span($U$). This metric is also commonly known as the Mean Squared Error (MSE).

**Preprocessing**

PCA is very sensitive to scaling, and so data must be preprocessed in the following way before invoking PCA –

- Convert data to have mean zero - This is achieved by simply subtracting the sample mean from the data.
- Convert data to have standard deviation 1 in each coordinate. This is done so that different coordinates are comparable, and can be achieved by dividing each coordinate by the sample standard deviation in that coordinate.
It is easy to see that minimizing the error in approximation equates to maximizing the length of the projected vectors. Let \( \hat{x} \) be an approximation of \( x \), and \( e = x - \hat{x} \) be the error vector. Then, error is minimized when \( e \perp (x - \hat{x}) \) as that gives the minimum \( ||e||_2 \). Coincidentally, this also maximizes \( ||\hat{x}||_2 \). This is the primary reason for sensitivity of PCA to coordinate scaling since the coordinates with large values tend to dominate the length of the projected distances and errors. One could also look at it as PCA maximizing the variance of the projected points. Let us take a look at the case of PCA applied on 1-dimensional data.

**PCA for 1-dimensional data**

Given a unit vector \( u \) and a point \( x \), projection of \( x \) on \( u \) is given by \( \langle x, u \rangle = x^T u \). To maximize the projected distances,

\[
\sum_{i=1}^{n} (x_i^T u)^2 = \sum_{i=1}^{n} u^T x_i x_i^T u \\
= \sum_{i=1}^{n} u^T (x_i x_i^T) u \\
= u^T (\sum_{i=1}^{n} x_i x_i^T) u \\
= u^T M u
\]

where \( M = \sum_{i=1}^{n} x_i x_i^T \) is a symmetric matrix. Then, \( \underset{u \in \mathbb{R}^D}{\text{argmax}} \ u^T M u \) is the principal eigenvalue of \( M \).

**PCA for D-dimensional data**

Let \( M = \sum_{i=1}^{n} x_i x_i^T \) be a \( D \times D \) symmetric matrix. Then, the eigenvalue decomposition of \( M \), \( \text{evd}(M) \) is given by \( \text{evd}(M) = V \Lambda V^T \) where \( \Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D) \), and \( \{\lambda_i\} \) is the set of eigenvalues ordered in a decreasing order. Let \( V_k = \{v_1, v_2, \ldots, v_k\} \) be the first \( k \) columns of \( V \). Then, \( V_k \) is the set of \( k \) orthonormal vectors that maximize the projected distances (and thereby variance).

**PCA Algorithm**

Putting the above steps together gives us a 4-step process to compute the k-rank transformation matrix using PCA.

1. Preprocess the data to have mean zero and standard deviation 1 in each coordinate. This process is also called *centering* the data.
2. Compute the covariance matrix \( M = \sum_{i=1}^{n} x_i x_i^T \) using the centered data.
3. Compute the eigenvalue decomposition of $M$. Let $\text{evd}(M) = V\Lambda V^T$ where $V = [v_1, v_2, \ldots, v_D], \Lambda = \text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D)$ with $\lambda_1 \geq \lambda_2 \geq \ldots \lambda_D$.

4. Compute the linear transformation matrix $S$ as the following –

$$S = \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{bmatrix}$$