MMD_PnP2

Tobias Winter

March 2025

1 Task 1: Similarities for Min-hash Signatures I

1.1 (a)

In this exercise we study the effect of the number of min-hash functions used on the accuracy of estimating the Jaccard-similarity of documents. For that purpose, download the template code [4] and the prepared data [1]. Upon execution, the code loads data derived from the 20 newsgroups dataset and a pre-computed min-hash signature matrix for the dataset (using 10.000 hash-functions, so the corresponding matrix has 10.000 rows). Code for computing the Jaccard-similarity of the documents is already in place (the similarities are stored in the matrix sim which is a square matrix containing all pair-wise similarities).

(a) Compute approximations to the Jaccard-similarity from the signature matrix using 1, 100, 200, 300, . . . hash-functions (i.e., pick a subset of the rows of signature matrix for computing the similarities). Create a plot showing the sum of squared errors between sim and the approximate Jaccard-similarites for the different considered number of hash-functions. What do you observe?

You can use the functions pdist and squareform from scipy.spatial.distance.

Answer:

For this task we look at subsets of the signature matrix which was calculated using the jaccard simularitys. We want to know how the number of min-hash-functions effects the ssd. So we use the algorithm below "Compute SSE":

We get the plot from figure 1. Here we can see that we get a significant lower SSE after 200 hahs function used. After that not much changes. According to this plot, the sweet spot seems to be between 200 and 400 hash functions.

1.2 (b)

For b, the following additions need to be in the algo Compute SSE - Compute SSE, show 10-nearest-neighbors and we also get the table shown below with the 10-nn with the true sim and the approx sim and there union, if any.

Algorithm 1 Compute SSE

Input: Signature matrix `signature`, true similarity matrix `sim`
Output: List of SSE values `errors`
Initialize empty list `errors` $\leftarrow [\,]$
Set `num_hashes` $\leftarrow [1, 100, 200, \ldots, 1000]$
for each k in `num_hashes` **do**
    `sig_subset` $\leftarrow$ first k rows of `signature`
    `dist` $\leftarrow$ `pdist`(`sig_subset`^{T}, "hamming")
    `approx_sim` $\leftarrow 1 -$ `squareform`(`dist`)
    `sse` $\leftarrow \sum$(`sim` $-$ `approx_sim`)^2
    Append `sse` to `errors`
end for

Algorithm 2 Compute SSE, show 10-nearest-neighbors

Input: Signature matrix `signature`, true similarity matrix `sim`
Output: List of SSE values `errors`
Initialize empty list `errors` $\leftarrow [\,]$
Set `num_hashes` $\leftarrow [1, 100, 200, \ldots, 1000]$
for each k in `num_hashes` **do**
    `sig_subset` $\leftarrow$ first k rows of `signature`
    `dist` $\leftarrow$ `pdist`(`sig_subset`^{T}, "hamming")
    `approx_sim` $\leftarrow 1 -$ `squareform`(`dist`)
    `sse` $\leftarrow \sum$(`sim` $-$ `approx_sim`)^2
    Append `sse` to `errors`
    `true_neighbors` $\leftarrow$ indices of the 10 largest values in row 0 of `sim` (excluding index 0)
    `approx_neighbors` $\leftarrow$ indices of the 10 largest values in row 0 of `approx_sim` (excluding index 0)
    `union` $\leftarrow$ set intersection of `true_neighbors` and `approx_neighbors`
end for

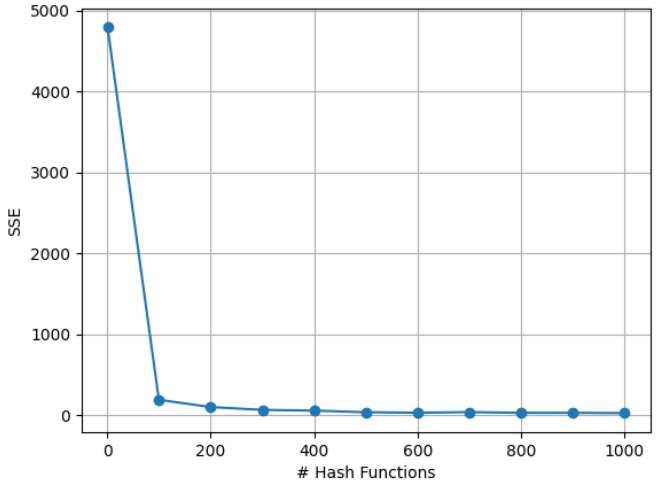

Figure 1: SSE vs. Hash Functions

k	True Neighbors	Approx Neighbors	Common Neighbors
1	74, 303, 199, 149, 370, 58, 36, 44, 367, 206	496, 495, 494, 493, 492, 491, 490, 489, 488, 487	–
100	74, 303, 199, 149, 370, 58, 36, 44, 367, 206	303, 74, 58, 149, 5, 386, 332, 360, 237, 106	74, 58, 149, 303
200	74, 303, 199, 149, 370, 58, 36, 44, 367, 206	303, 74, 149, 370, 36, 5, 199, 226, 48, 58	36, 199, 74, 303, 370, 149, 58
300	74, 303, 199, 149, 370, 58, 36, 44, 367, 206	303, 74, 199, 370, 149, 5, 58, 66, 345, 335	199, 74, 303, 370, 149, 58
400	74, 303, 199, 149, 370, 58, 36, 44, 367, 206	303, 74, 199, 370, 58, 149, 454, 66, 186, 102	199, 74, 303, 370, 149, 58
500	74, 303, 199, 149, 370, 58, 36, 44, 367, 206	303, 74, 199, 149, 58, 370, 66, 102, 367, 454	199, 74, 367, 303, 370, 149, 58
600	74, 303, 199, 149, 370, 58, 36, 44, 367, 206	74, 303, 199, 149, 58, 370, 367, 36, 22, 102	36, 199, 74, 367, 303, 370, 149, 58
700	74, 303, 199, 149, 370, 58, 36, 44, 367, 206	74, 303, 199, 149, 370, 58, 5, 445, 36, 66	36, 199, 74, 303, 370, 149, 58
800	74, 303, 199, 149, 370, 58, 36, 44, 367, 206	74, 303, 199, 149, 370, 58, 36, 445, 5, 102	36, 199, 74, 303, 370, 149, 58
900	74, 303, 199, 149, 370, 58, 36, 44, 367, 206	74, 199, 303, 149, 370, 58, 5, 36, 66, 445	36, 199, 74, 303, 370, 149, 58
1000	74, 303, 199, 149, 370, 58, 36, 44, 367, 206	74, 199, 303, 149, 370, 58, 5, 474, 36, 367	36, 199, 74, 367, 303, 370, 149, 58

2 Task 2 Similarities from Min-hash Signatures I

Assume that we want to approximate the Jaccard-similarity of two docu- ments from their min-hash signatures. We want to estimate how many hash functions

we need to use for the signature matrix such that with at least probability $\delta = 0.99$ the absolute error between the approximate Jaccard-similarity computed from the min-hash signature and the true Jaccard-similarity is at most $\epsilon = 0.01$. Compute a lower bound on the number of hash functions to use. Hint: Hoeffding's inequality.

Hoeffding's inequality

$$P(|\hat{\mu} - \mu| \geq \epsilon) \leq 2\exp(-2\epsilon^2 m) \tag{1}$$

where

$$\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} X_i \tag{2}$$

with μ beeing the true Jaccoard sim and $\hat{\mu}$ beeing the approximated jaccard sim. $\hat{\mu}$ should be with an abs. error ϵ of μ with at least δ prob.

$$P(|\hat{\mu} - \mu| \leq \epsilon) \geq \delta \Leftrightarrow P(|\hat{\mu} - \mu| > \epsilon) \leq 1 - \delta$$

$$\Rightarrow 2\exp(-2\epsilon^2 m) \leq 1 - \delta$$

$$= \exp(-2\epsilon^2 m) \leq \frac{1-\delta}{2}$$

take ln

$$\Rightarrow -2\epsilon m \leq \ln\left(\frac{1-\delta}{2}\right)$$

$$= m \geq \frac{1}{2\epsilon} \cdot \ln\left(\frac{2}{1-\delta}\right)$$

if we now plug in for ϵ and δ we get that we need 26492 functions.

3 Task 3: Min-Hashing Using MapReduce

Suppose we want to use a MapReduce framework to compute min-hash signatures of a set of documents represented by a binary matrix M, such that each column corresponds to a document and each row to a shingle. If the matrix is stored in chunks that correspond to some columns, then it is quite easy to exploit parallelism. Each map task gets some of the columns and all the hash functions, and computes the min-hash signatures of its given columns. However, suppose the matrix were chunked by rows, so that a map function is given the hash functions and a set of rows to work on. Design map and reduce functions to exploit MapReduce with data in this form. Credit: Exercise taken from the MMDS book (with slight adjustments).

We know $M_{r,c} = 1$ tells us that the r-th shingle is in column c present.

Algorithm 3 Task 3 map

```
1: Input: rows of M
2: Output: key = (column, index), value = hash_i(r)
3: for row r in rows do
4:     for column c in row where M[row, column] = 1 do
5:         k ← range of hash function
6:         for index ← 0,1,...,k do
7:             yield(key = (c, index), value = h_i(r)
8:         end for
9:     end for
10: end for
```

Algorithm 4 Task 3 reduce

```
1: Input: key = (column, index), list of hashes
2: Output: key = (column, index), min(list of hashes)
3: min_value ← min(list of hashes)
4: yield(key, min_value)
```

4 Task 4: LSH Using MapReduce

Suppose we wish to implement LSH by MapReduce. Specifically, assume chunks of the signature matrix consist of columns, and elements are key-value pairs where the key is the column number and the value is the signature itself (i.e., a vector of values).

4.1 (a)

Show how to produce the buckets for all the bands as output of a single MapReduce process. Hint: Remember that a map function can produce several key-value pairs from a single element.

Algorithm 5 Task 4 a) mapper

```
1: Input: key = document_id, value = signature vector
2: Output: key = (band_index, band_hash), value = document_id
3: r ← number of rows per band
4: b ← number of bands
5: for band_index ← 0, ... , b-1 do
6:     start ← band_index * r
7:     end ← start + r
8:     band ← signature[start:end]
9:     band_hash ← hash(band)
10:    yield((band_index, band_hash), doc_id)
11: end for
```

A band of signatures now hashes to a bucket. The reducer now only needs to group by Key.

Algorithm 6 Task 4 a) reducer

```
1: Input: key = key of mapper, value = document ID
2: yield (key, value)
```

4.2 (b)

Show how another MapReduce process can convert the output of (a) to a list of pairs that need to be compared. Specifically, for each column i, there should be a list of those columns j ¿ i with which i needs to be compared.

From a) we get as key the band index and band signatures and as the value we get a list of document IDs. key = (band_index, band_hash), val = (document_1,...document_n)

Algorithm 7 Task 4 b) mapper

```
1: Input: output of MapReduce process form a)
2: n ← length of columns
3: for i ← 0, 1, ..., n do
4:     for j ← i +1, ..., n do
5:         if i < j: yield((columnsID[i], columnsID[j]), 1)
6:     end for
7: end for
```

Algorithm 8 Task 4 b) reducer

```
1: yield((columnsID[i], columnsID[j]))
```

5 Task 5: LSH for Manhattan Distance

Let $x, y \in \{0,1\}^d$ be binary vectors of lenght d. The Hamming distance between vectors x, y is $d_H(x,y) = |\{j|j = 1, ..., d : x_j \neq y_j\}|$.

5.1 (a)

Show that d_H is indeed a distance. Consider the hash function family $H = \{h_i\}_{i=1}^d$, where $h_i(x) = x_i$. Prove that H defines an LSH family. What are its "sensitivity parameters"?

Prove that d_H is a distance:

1. Non- negativity:

The entrys in x and y can only take the values 0 and 1 of dim d. So $d_H(x,y) \geq 0$

1. Non- negativity:

1. d(x,x) = d(x,x)

Again, this is true since we have only 0 and 1 in d dims.

1. d(x,y) = d(y,x)

True for the same reason.

1. Triangle inequality

Lets say we have a 3rd point z with $x \neq z$ and $y \neq z$. So x and y differ in at least one entry from z.

Thats why $d(x,z) \leq d(x,y) + d(y,z)$ because of the Non-negativity axiom.

We now look at the Hash familiy H, with $h_i(x) = x_i$. The hash family H is (d_1, d_2, p_1, p_2)-senetive if

1. $d_H(x,y) \leq d_1 \Rightarrow Pr[h_i(x) = h_i(y)] \geq p_1$
2. $d_H(x,y) \geq d_2 \Rightarrow Pr[h_i(x) = h_i(y)] \leq p_2$

Lets now choose at random one h from H. (one hash function gives us the coordinates for $i \in \{1, ..., d\}$ We know that we have d indices in x_i and d indices in y_i from h_i. We also know that the coordinates where $x_i \neq y_i = d_H(x,y) \Rightarrow x_i = y_i = d - d_H(x,y)$ Thus:

$$Pr[h_i(x) = h_i(y)] = Pr[x_i = y_i] = 1 - \frac{d_H(x,y)}{d}$$

$$\Rightarrow Pr[h_i(x) \neq h_i(y)] = \frac{d_h(x,y)}{d}$$

So H is an LSH family with d_H. We also know that $d_1 < d_2$ ($d_1 \rightarrow$ two objects are similar $d_2 \rightarrow$ two objects are dissimilar) So, if $d_H(x,y) = d_1$

$$Pr[h_i(x) = h_i(y)] = 1 - \frac{d_H(x,y)}{d} \Rightarrow p_1 = 1 - \frac{d_1}{d}$$

if $d_H(x,y) = d_2$

$$Pr[h_i(x) = h_i(y)] = 1 - \frac{d_H(x,y)}{d} \Rightarrow p_2 = 1 - \frac{d_2}{d}$$

So the (d_1, d_2, p_1, p_2)-sensitivity is $(d_1, d_2, 1 - \frac{d_1}{d}, 1 - \frac{d_2}{d})$

5.2 (b)

For d = 10, can you boost the hash family such that for $d_H(x, y) \leq 2$, the probability of collision is at least 95% and for $d_H(x, y) \geq 4$ at most 5%? If not, prove your statement, if yes, provide a possible construction.

We know that for d=10 and $d_H(x, y) = 2$

$$Pr[h_i(x) = h_i(y)] = 1 - \frac{d_H(x, y)}{d} = 1 - \frac{2}{10} = \frac{4}{5} = 0.8 = p_1$$

for $d_H(x, y) = 4$

$$1 - \frac{4}{10} = \frac{3}{5} = 0.6 = p_2$$

With AND construction we get

$$p_1^r, p_2^r$$

and OR

$$1 - (1 - p_1^r)^b, 1 - (1 - p_2^r)^b$$

Proof that we cant construct this conditions:
let f be a function of p with

$$f(p) = 1 - (1 - p^r)^b$$

if $p_1 < p_2$ (which there are in our case) then $f(p_1) < f(p_2)$
but we want that $p_2 \leq 0.05$ and $p_1 \geq 0.95$ so $f(p_2)$ can never be less then $f(p_1)$

6 Task 6: LSH for the Euclidean Distance

Suppose we have points in a 3-dimensional Euclidean space: p1 = (1, 2, 3), p2 = (0, 2, 4), and p3 = (4, 3, 2). Consider the three hash functions defined by the three axes (to make our calculations very easy). Let buckets be of length a, with one bucket the interval $[0, a)$ (i.e., the set of points x such that $0 \leq x < a$), the next $[a, 2a)$, the previous one $[-a, 0)$, and so on.

6.1 (a)

For each of the three lines, assign each of the points to buckets, assuming a = 1
the buckets here are [0, 1), [1, 2) and also in the minus direction from [-1, 0), [-2, -1) and so on.

$$\begin{bmatrix} \textbf{Point} & h_1(p) & h_2(p) & h_3(p) \\ p_1 = (1, 2, 3) & [1, 2) & [2, 3) & [3, 4) \\ p_2 = (0, 2, 4) & [0, 1) & [2, 3) & [4, 5) \\ p_3 = (4, 3, 2) & [4, 5) & [3, 4) & [2, 3) \end{bmatrix}$$

6.2 (b)

$$\begin{bmatrix} \textbf{Point} & h_1(p) & h_2(p) & h_3(p) \\ p_1 = (1,2,3) & [0,2) & [2,4) & [2,4) \\ p_2 = (0,2,4) & [0,2) & [2,4) & [4,6) \\ p_3 = (4,3,2) & [4,6) & [2,4) & [2,4) \end{bmatrix}$$

6.3 (c)

What are the candidate pairs for the cases a = 1 and a = 2?

For case a = 1 we only have to look where the 3 points match.

1. p_1 and p_2 math in [2,3)
2. p_2 and p_3 math in [2,3)
3. p_1 and p_3 math in [2,3)
4. p_1 and p_3 math in [3,4)
5. p_2 and p_3 math in [4,5)

For a = 2

1. p_1 and p_2 math in [2,4)
2. p_2 and p_3 math in [2,4)
3. p_1 and p_3 math in [2,4)
4. p_1 and p_3 math in [3,4)
5. p_2 and p_3 math in [4,6)

6.4 (d)

For each pair of points, for what values of a will that pair be a candidate pair? Here we can read it out of the table.

7 Task 7: LSH for the Cosine Distance

Consider LSH for the cosine distance, i.e., read and understand section 3.7.2 of the MMDS book. e prepared to explain how hashing works and explain how this can define a locality-sensitive family of functions. Explain the corresponding proof Hint: It might be useful to prepare several sketches that help you explain your thoughts in the exercise session