Pseudo-randomness, Hash functions and Min-Hash for document comparison

Yoav Freund

This lecture is based on:

- “Mining Massive Datasets” by Jure Leskovec, Anand Rajaraman and Jeffrey D. Ullman
- Description is in section 11.6 in the lecture notes.
Random vs. pseudo-random numbers

1. **Random:**
   
   ![Random outcomes with thumbs up and down]

2. **Pseudo-Random:**
   ```python
   In [1]: import random
   
   In [8]: random.seed(a=1550)
   [random.randint(0,1) for i in range(10)]
   Out[8]: [1, 1, 0, 0, 0, 1, 0, 0, 1, 0]
   
   In [7]: random.seed(a=22)
   [random.randint(0,1) for i in range(10)]
   Out[7]: [1, 0, 0, 1, 0, 0, 1, 0, 1, 0]
   
   In [9]: random.seed(a=1550)
   [random.randint(0,1) for i in range(10)]
   Out[9]: [1, 1, 0, 0, 0, 1, 0, 0, 1, 0]
   ```
Comparing Random with Pseudo Random

Compare two sources: random vs. pseudo-random. Suppose the seed consists of $k$ bits, and the length of the generated binary sequence $n \gg k$.

1. A true coin flip assigns equal probability to each of the $2^n$ binary sequences.

2. The pseudo-random number generator assigns non-zero probability to at most $2^k$ sequences.

3. There exists an algorithm that can distinguish between the two distributions.

4. There is no efficient (poly-time in $n$) algorithm that can distinguish between the sources.
Random Hash Function

- The fact that the sequence is a function of the seed is a deficiency of the pseudo random generator.
- However the same fact is a *feature* when using PRNG’s to define Hash Functions
- A hash function $h_{seed}$ maps from some large domain $X$ to a small set $1, 2, \ldots, n$
- If $seed$ is chosen uniformly at random, we can $h_{seed}(x_1), h_{seed}(x_2) \ldots, h_{seed}(x_n)$ are (pseudo) IID draws from the uniform distribution over $1, 2, \ldots, n$.
- If $R(seed)$ is a PRNG then $h_{seed}(x) = R(seed + x)$ is a random hash function.
Random Hash Function

- The fact that the sequence is a function of the seed is a deficiency of the pseudo random generator.
- However the same fact is a *feature* when using PRNG’s to define **Hash Functions**
- A hash function $h_{seed}$ maps from some large domain $X$ to a small set $1, 2, \ldots, n$
- If $seed$ is chosen uniformly at random, we can $h_{seed}(x_1), h_{seed}(x_2) \ldots, h_{seed}(x_n)$ are (pseudo) IID draws from the uniform distribution over $1, 2, \ldots, n$.
- If $R(seed)$ is a PRNG then $h_{seed}(x) = R(seed + x)$ is a random hash function.
• Suppose we are writing a compiler and we need to keep the memory address for each variable name. We typically use a Hash Table.

• Hash Tables are an implementation of map data structures that allows insertion, deletion and retrieval in $O(1)$ time.

• Suppose table has $n$ slots.

• Given $(key, value)$ pair. Place pair in slot $h_{seed}(key)$.

• Unless collision: there is already an item in that slot.

• Pick at another randomly picked slot, repeat until empty slot found.

• A random hash function will guarantee that the probability of a collision, if $m$ slots are occupied, is $m/n$.

• Therefore, if $m/n < 1/2$ then the expected number of collisions before we find an empty slot is 1.

• $E(#\text{collisions}) = \frac{1}{2} 0 + \frac{1}{2} (1 + E(#\text{collisions})) \Rightarrow E(#\text{collisions}) = 1$
Finding Similar Items

Based on chapter 3 of the book “Mining Massive Datasets” by Jure Leskovec, Anand Rajaraman and Jeffrey D. Ullman

Suppose we receive a stream of documents.

We want to find sets of documents that are very similar

Reasons: Plagiarism, Mirror web sites, articles with a common source.
Measuring the distance between sets

1. Suppose we consider the set of words in a document. Ignoring order and number of occurrences.
2. We will soon extend this assumption.
3. If two documents define two sets $S$, $T$, how do we measure the similarity between the two sets?
4. Jaccard similarity: $\frac{|S \cap T|}{|S \cup T|}$

Figure 3.1: Two sets with Jaccard similarity 3/8
Let $X$ be a finite (but large) set

Let $N = \{1, 2, \ldots, n\}$ be a (very large) set of numbers.

A Hash-Function $h : X \rightarrow N$ is a function that “can be seen as” a mapping from each element of $X$ to a an independently and uniformly chosen random element of $N$. 
Min-Hash

1. Choose a random hash function $h_i$
2. Given a set of elements $S$ in the domain $X$
3. $\text{min-}H_i(S) = \min_{s \in S} h_i(s)$
4. A min-hash signature for a document is the vector of numbers $\langle \text{min-}H_1(S), \text{min-}H_2(S), \ldots, \text{min-}H_k(S) \rangle$
5. Signature also called a “sketch”: Any length document is represented by $k$ numbers.
6. A lot of information is lost, but enough is retained to approximate the Jaccard similarity.
Visualizing Min-Hash

- We can represent the set of words in each document as a matrix.
- Rows $a, b, c, \ldots$ correspond to words.
- Columns $S_1, S_2, \ldots$ correspond to documents.
- A “1” in row $b$, column $S_i$ means that document $S_i$ contains the word $b$.
- Hashing corresponds to randomly permuting the rows.
- Min-hashing a document corresponds to identifying the first “1” starting from the top of the column.

<table>
<thead>
<tr>
<th>Element</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$b$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$c$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$d$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$e$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Element</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$b$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$e$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$a$</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$d$</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$c$</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Understanding Min-Hash

- For any set $S$ of size $|S|$, the probability that any particular element $s \in S$ is the min-hash is $1/|S|$
- Fix two documents $S_i, S_j$ (columns) and partition the rows that contain at least a single “1” in those columns
- Denote by $X$ rows that contain 1,1 (both documents contain the word.)
- Denote by $Y$ rows that contain 1,0 or 0,1 (only one document contains the word)
- Permuting the rows does not change which rows are $X$ and which are $Y$
- The min-hash of $S_i, S_j$ agree if and only if first row that is not 0,0 is an $X$
- The probability that the min-hash of $S_i, S_j$ agree is exactly $\frac{\#X}{\#X + \#Y}$ which is equal to $JS(S_i, S_j) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|}$
We can use min-hash to estimate Jaccard similarity (JS):

\[
\frac{|S_i \cap S_j|}{|S_i \cup S_j|}
\]

For each min hash function \( MH_i \) we have that

\[
P_i [\text{min-}H_i(S) = \text{min-}H_i(T)] = \frac{|S \cap T|}{|S \cup T|}
\]

A single comparison yields only true (1) or false (0)

Taking the average of \( k \) independent hash functions we can get an accurate estimate.
How many hash functions do we need? (1)

1. From a statistics point of view we have $k$ independent binary random variables:

$$X_i = \begin{cases} 
1 & \text{if } \min-H_i(S) = \min-H_i(T) \\
0 & \text{otherwise}
\end{cases}$$

2. We seek the expected value: $p \doteq E(X_i) = \frac{|S \cap T|}{|S \cup T|}$

3. We have to overcome the large std: $\sigma(X_i) = \sqrt{p(1-p)}$

4. Averaging gives a random variable with the same expected value but a smaller variance.

$$Y = \frac{1}{k} \sum_{i=1}^{k} X_i; \quad E(Y) = p \quad \sigma(Y) = \sqrt{\frac{p(1-p)}{k}}$$

5. $\sigma(Y) \leq \sqrt{\frac{1}{2}(1-\frac{1}{2})(1/k)} = \frac{1}{2\sqrt{k}}$
Using a z-Scores to calculate the minimal number of hash functions.

1. Suppose we want our estimate of JS to be within $\pm 0.05$ of the Jaccard distance with probability at least 95%.

2. The fraction of min-hashes matches is the average of $k$ independent binary random variables.

3. Let's assume $k$ is large enough so that the central limit theorem holds.

4. We want a confidence of 95% that the estimate is within $\pm 0.05$ of the true value. In other words, we want

   $$2\sigma(Y) \leq 0.05$$

5. Using the bound

   $$\sigma(Y) \leq \frac{1}{2\sqrt{k}}$$

   we find that it is enough if $\frac{1}{k} \leq 0.05$ or if $k \geq 20$. 
Introducing Order

1. So far, we represented each document by the set of words it contains.
2. This removes the order in which the words appear: “Sampras beat Nadal” is the same as “Nadal beat Sampras.”
3. We can add order information to the set representation using Shingles.
Consider the sentence: ”the little dog loughed to see such craft“

Word set representation: \{ “the”, ”little”, ”dog”, 
“loughed”, ”to”, ”see”, ”such”, ”craft” \}

2-shingle representation: \{ “the little”, ”little dog”, “dog 
loughed”, ”loughed to”, ”to see”, ”see such”, ”such craft” \}

3-shingle representation: \{ “the little dog”, ”little dog loughed” ,...

And so on

The number of shingles of length \( k \) from a document of length \( n \) is?

\( n + 1 − k \) - largest for single words!

On the other hand, there is a much larger number of different items.

\( k \) too small - documents judged similar too often.

\( k \) too large - documents judged dissimilar too often