SparseHash: Embedding Jaccard Coefficient between Supports of Signals

MM-SPARSE
Seattle, 15 July 2016

Diego Valsesia    Sophie Marie Fosson    Chiara Ravazzi
Tiziano Bianchi   Enrico Magli

Politecnico di Torino

www.crisp-erc.eu
Jaccard similarity

- **Jaccard coefficient**: set similarity metric

\[ J(S_1, S_2) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} \]

- Used in several applications: search of near-duplicate documents, image retrieval, genome clustering, ...

- The support of a \( k \)-sparse signal living in an \( n \)-dimensional space is a set with \( k \ll n \) elements

\[ S_1 = \{ 2, 5, 6 \} \]
\[ S_2 = \{ 1, 2, 5, 8 \} \]
\[ J = \frac{2}{5} \]
Several information processing tasks boil down to comparing similarity metrics between a large number of signals living in a high-dimensional space.

To do this efficiently:

- A compact (low-dimensional) representation is calculated...
- ... which approximately preserves the desired similarity metric.

Example (Gaussian random projections):

- \( y = \Phi x \), \( \Phi \) made of i.i.d. random Gaussian entries.
- \( l_2 \) norms are approximately preserved:

\[
(1 - \epsilon)\|u - v\|_2^2 \leq \|\Phi u - \Phi v\|_2^2 \leq (1 + \epsilon)\|u - v\|_2^2
\]
How do you create a compact representation of a set to preserve the Jaccard coefficient?

Classic method: **MinHash**
- Apply a random universal hash function to map each element of the set to an integer
- Choose the minimum as your measurement
- Repeat
- Variant: quantize by keeping only the LSB (1-bit MinHash)

This work: **SparseHash**
- Can random projections be used to preserve the Jaccard coefficient?
SparseHash

Sparse random projections

- Let \( x \) be a \( k \)-sparse signal
- Let \( \Phi \) be an \( m \times n \) matrix where each entry is non-zero with probability \( \gamma \)

- Measurements are computed as

\[
y = 1(\{\Phi x = 0\})
\]

- Binary-valued measurements form the SparseHash \( y \)
SparseHash

Equivalent computation via hash functions

- The **SparseHash** can be equivalently computed without having to generate the matrix \( \Phi \)
- This allows to
  - generalize to any set (not just supports of signals)
  - compute it more efficiently

Computing SparseHash

- Define random hash functions \( f_i : S \rightarrow [0, 2^b - 1], \ i \in [1, m] \)
- Define threshold \( \tau = \gamma 2^b \)
- For \( i = 1, \ldots, m \)
  - For each entry in the set, apply \( f_i \) to get integer \( h_{ij} \)
  - If \( h_{ij} < \tau \)
    - Set \( y_i = 1 \) and break
How do we choose $\gamma$?

- Parameter $\gamma$ is important because it controls the sparsity of the $\Phi$ matrix.
- $\gamma$ should be chosen to maximize the entropy of the SparseHash, i.e. to yield equiprobable zero and nonzero measurements.
- For sets of cardinality $k$ choose $\gamma = 1 - 2^{-\frac{1}{k}}$. 
How do we compute similarity between SparseHashes?

- Method 1: compute the following metric:

\[
\text{sim}_{sh}(y, z) = \frac{\log(\text{sim}_\cap(y, z))}{\log(\text{sim}_\cup(y, z))}
\]

where:

\[
\text{sim}_\cup(y, z) = \frac{1}{m} \sum_{i=1}^{m} 1(y_i = 0, z_i = 0)
\]

\[
\text{sim}_\cap(y, z) = \frac{\sum_{i=1}^{m} 1(y_i = 0) \sum_{j=1}^{m} 1(z_j = 0)}{m \sum_{i=1}^{m} 1(y_i = 0, z_i = 0)}
\]
You can prove that the similarity metric concentrates around the original Jaccard coefficient of the two sets

**Theorem 1: sim_sh concentration**

Let \( \mathcal{X} = \{x_i \in \mathbb{R}^n : \text{supp}(x_i) \in [k_{min}, k_{max}]\}_{i=1}^{N} \), for any \( \epsilon > 0 \), \( \beta > 2 \) and any integer \( n \), let \( m \) be a positive integer such that

\[
m > 32 \frac{\log 4 + \beta \log N}{\gamma^2 k_{min}^2 e^{-\gamma k_{max} \epsilon^2}}
\]

then

\[
\mathbb{P}\left[ \bigcup_{(u,v) \in \mathcal{X}} \{|\text{sim}_{sh}(\Phi u, \Phi v) - J(u, v)| > \epsilon\} \right] \leq N^{-\beta+2}
\]
SparseHash

Metrics in the embedded space: 2) Hamming distance

- Method 2: Hamming distance

\[ d_H(y, z) = \sum_{i=1}^{m} 1 \{ y_i \neq z_i \} \]

- Much faster to compute
- Non-linearly related to the original Jaccard
- Useful for Locality Sensitive Hashing (LSH):
  - One can actually prove that it is advantageous for LSH, requiring fewer hash tables than MinHash (ongoing work)
**Theorem 2: Hamming distance concentration**

\[
\mathbb{E} \left[ \frac{d_H(\Phi u, \Phi v)}{m} \right] = 1 - (1 - \gamma)^{k_1} - (1 - \gamma)^{k_2} + 2(1 - \gamma)^{\frac{k_1 + k_2}{1 + J(u,v)}}
\]

\[\trianglerighteq P_{sh}\]

\[
\text{Var} \left[ \frac{d_H(\Phi u, \Phi v)}{m} \right] = \frac{(1 - P_{sh})P_{sh}}{m}
\]
Example: $k = 230$, $\gamma = 3 \cdot 10^{-3}$, $m = 50$, $n = 1000$

SparseHash with sim_sh metric

SparseHash with Hamming distance
Experiment

Near-duplicate text documents

- UCI dataset of $N = 300,000$ New York Times articles
- Bag of words representation of each article
- Mean sparsity $k = 232$
- Identify near-duplicate documents as those exceeding a preset threshold on the Jaccard coefficient

Precision vs recall, with ground truth threshold 0.5 (left) and 0.6 (right)
Experiment
Near-duplicate text documents

Precision and recall, with ground truth threshold 0.5

Precision and recall, with ground truth threshold 0.6
Conclusions

- **SparseHash** uses random projections to embed the Jaccard coefficient between sets.
- It is more efficient than MinHash: more compact representations.
- Future work: proving that the Hamming distance metric in the compressed space is even more efficient and enables faster LSH.
C-language library with multithread support available on:

https://github.com/diegovalsesia/sparsehash

Follow future publications on:

www.crisp-erc.eu