Qs Rank and efficient ϵ-neighbor search

Arthur Darcet & Yohann Salaun

March 24, 2013

1 Overview

The QsRank algorithm, described in [1], is a method that allows a ranking for binary hash codes to efficiently perform ϵ-neighbors search in large data. The aim of this problem is to, for a given query q and a given radius ϵ, quickly finds the data subset $Y = (y_i)_i$ such that $\forall i,\ ||y_i - q|| < \epsilon$.
Methods based on the Hamming distance have been recently developed to solve efficiently such a problem for large data. However these latter methods often lack of precision due to their binarization of the query which induces a lack of speed in the performances of their algorithm. The methods presented here, performs a more accurate ranking for hash codes and thus allows a faster and more efficient method for ϵ-neighbors retrieval.

2 Theoretical Description

The algorithm of ϵ-neighbor search using the QsRank works in two times:

1. Hash code generation of the data for faster retrieval
2. Ranking of the hash code depending on a query

The first part can be computed before any requests whereas the second needs the parameters of the search (query q and neighbor distance ϵ).

2.1 Hash Codes generation

Such search algorithm are used for huge data with components in high dimensional space. Thus, in order to generate simple hash codes for each input data point $x \in \mathbb{R}^d$, a Principal Component Analysis (PCA) is computed.
The hash code h of x is then computed by taking the sign of the PCA-projections:

$$\forall j \in [1; d],\ h_j = \begin{cases} 1 \text{ if } (PCA(x, \{x_i\}_i))_j > 0 \\ 0 \text{ if } (PCA(x, \{x_i\}_i))_j \leq 0 \end{cases}$$

In practice, only the first components of the hash codes are computed.
The benefits given by the PCA are numerous:

1. The PCA allows a dimension decrease but keeps most of the information. This way, hash codes are shorter and the retrieval is still efficient.
2. The PCA is an orthogonal projection that preserves the L^2-norm. Thus, the ϵ-ball around a query is still meaningful after PCA.
3. The PCA values of the input data points are uncorrelated which will lead to an efficient ranking with Qs Rank.

2.2 QsRank for Hash Codes ranking

We suppose, that the projection y of the input data points x are distributed along a probability distribution function p. Then, with a given query q, a neighbor distance ϵ and a hash code h, the Qs Rank formula is defined by:

$$\text{QsRank}(q,h,\epsilon) = \frac{\int_{NN(q,\epsilon)\cap S(h)} p(y)dy}{\int_{NN(q,\epsilon)} p(y)dy}$$

where:

- $NN(q,\epsilon) = \{y \in \mathbb{R}^d \text{ s.t. } ||y-q|| < \epsilon\}$ is the ϵ-ball around the query q
- $S(h) = \{y \in \mathbb{R}^d \text{ s.t. } \forall i \in [1;d] y_i h_i > 0\}$ is the set described by the hash code h

The QsRank can only be seen as a probability, with Bayes rule:

$$\text{QsRank}(q,h,\epsilon) = \frac{\mathbb{P}(y \in NN(q,\epsilon)\cap S(h))}{\mathbb{P}(y \in NN(q,\epsilon))} = \mathbb{P}(y \in S(h) | y \in NN(q,\epsilon))$$

Thus, the QsRank only ranks hash codes with respect to their probability of containing many ϵ-neighbors.

2.3 QSRank approximation

In order to compute fast retrieval, the QsRank will be approximated by a lighter formula.
First, only the top k dimensions of the PCA projection will be used (k will be defined afterward in the next section). Thus, $NN(q,\epsilon)$ becomes $NN(q^k,\epsilon)$ and $S(h)$ becomes $S(h^k)$ where x^k is the k-top dimensions of a vector $x \in \mathbb{R}^d$. This approximation seems legit since the aim of the PCA is to find the dimensions where most of the information is kept.

$$\text{QsRank}(q,h,\epsilon) \approx \frac{\int_{NN(q^k,\epsilon)\cap S(h^k)} p(y^k)dy^k}{\int_{NN(q^k,\epsilon)} p(y^k)dy^k}$$

Another approximation is to replace the ϵ-ball by an ϵ-hypercube:

$$NN(q^k,\epsilon) \leftrightarrow HC(q^k,\epsilon) = \{y^k \in \mathbb{R}^k \text{ s.t. } \forall i \in [1;k], |y_i^k - q_i^k| < \epsilon\}$$

Moreover, since the PCA produces uncorrelated projections, each dimension of $p(y)$ is supposed to be independent. The QsRank approximation then becomes:

$$\text{QsRank}(q,h,\epsilon) \approx \prod_{i=1}^{k} \frac{\int_{|y_i^k - q_i^k|<\epsilon, y_i^k h_i^k > 0} p(y_i^k)dy_i^k}{\int_{|y_i^k - q_i^k|<\epsilon} p(y_i^k)dy_i^k} = \prod_{i=1}^{k} \mathbb{P}(y_i^k \in S(h_i^k) | y_i^k \in HC(q_i^k,\epsilon))$$

The last approximation is to consider that the y are generated from a uniform law. This assumption accelerates a lot the computation and seems to work quite well in accordance with the authors of [1]. The final formula thus becomes:

$$\begin{aligned}\text{QsRank}(q,h,\epsilon) &\approx \prod_{i=1}^{k} \frac{\int_{|y_i^k - q_i^k|<\epsilon, y_i^k h_i^k > 0} dy_i^k}{\int_{|y_i^k - q_i^k|<\epsilon} dy_i^k} \\ &\approx \prod_{i=1}^{k} \text{clamp}\left(\frac{1}{2}\left(1 + \frac{h_i^k q_i^k}{\epsilon}\right), [0;1]\right)\end{aligned}$$

Opposed to the Hamming distance, this measure has many substantial advantages :

- The radius ϵ is took into account
- Since it is a product, if one of the component of the hash code induces a null probability, the whole QsRank becomes null. Some sets are thus not explored whereas Hamming distance methods would have.

3 Implementation

3.1 Efficient computation

Once the query q is given, the logarithmic QsRank is computed for the 2^k different hash codes. This results in $\mathcal{O}(k)$ logarithm computation and $\mathcal{O}(k2^k)$ additions. Moreover, only hash codes with non zeros probabilities have to be computed, which accounts for only 15% of the data points according to the experiments made in [1].

3.2 Retrieval procedure

Input : input data $X \in \mathbb{R}^{n \times d}$, query q, radius ϵ
Parameters: hash codes length $K1$ and $K2$, number of buckets L.
Precomputation
$Y = PCA(X)$
for $i = 1..K1 + K2$ **do**
  Compute the logarithmic sub-QsRanks : $\log\left(\text{clamp}\left(\frac{1}{2}\left(1 + \frac{h_i^k q_i^k}{\epsilon}\right), [0; 1]\right)\right)$
end
ϵ-search
Sort the buckets by $K1$-QsRank.
Pick the L^{th} first buckets.
Sort the buckets hash codes by $(K1 + K2)$-QsRank.
Return the ϵ-neighbors.

4 Results

References

[1] L. Zhang X. Zhang and H-Y. Shum. Qsrank: Query-sensitive hash code ranking for efficient epsilon-neighbor search. *CVPR*, 2012.