Learning to Hash for Indexing Big Data -- A Survey

Paper: Wang et al. Learning to Hash for Indexing Big DataVA Survey. Proceedings of the IEEE, 2016.

Background

Pipeline of Hashing-Based ANN Search

- Desiging Hash Functions: For example: $h_k(x) = \mathrm{sgn}(f(w_k^T x + b_k)) \in \{-1, 1\}$
- Building Reverse Table: For every hash value $v = \{h_k(x)\}_k$, list all x s.t. $f(x) = v$
- Querying: Given query q, enumerate all possible p s.t. $|p - q| \leq r$

Randomized Hashing Methods

A local sensitive hashing function should fulfill

$$P\{h(x_i) = h(x_j)\} = \mathrm{sim}(x_i, x_j)$$

- Random Projection Hashing

  Random choose w, $h_w(x) = \mathrm{sgn}(w^T x)$, then

$$P\{h(x_i) = h(x_j)\} = 1 - \frac{\theta_{ij}}{\pi}$$

- Random Permutation Hashing

  To estimate **Jaccard Similarity**:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

Random choose permutation π, $h_\pi(S) = \min_{a \in S} \pi(a)$

Categories of Learning-Based Hashing

- Data Dependent?
  - Independent: Random Local Sensitive
- Unsupervised, Supervised and Semisupervised
  - Unsupervised: Using distribution of data
  - Supervised: Metric Learning~Kernel Learning
  - Semisupervised: Using both unlabeled and labeled data
- N-Points-wise: Pointwise, Pairwise, Tripletwise...

- Linear or Nonlinear:
  - Linear hash function need less computation time.
- Single-Shot v.s. Multiple-Shot
- Weighted Hashing?

$$H : X \to \{\alpha_i h_i(\mathbf{x})\}_i$$

Methods

Spectral Hashing

- Minimize: $\frac{1}{2}\sum A_{ij}|y_i - y_j|^2$
- Bit Balance: $\mathbf{1}^T y_k = 0$
- Uncorrelated: $Y^T Y = nI_{K\times K}$

Anchor Graph Hashing

Angular-Quantization-Based Hashing

Try to hash to a binary code with closest angular:

$$y = \arg\max_b \cos(\angle bx) = \arg\max_b \frac{b^T x}{|b|}$$

Or including rotation R of data:

$$(y, R) = \arg\max_{(y,R)} \sum_i \frac{b^T R^T x_i}{|b|}$$

Binary Reconstructive Embedding

The distance of data and the distance of hash value define as

$$d_M(x_i, x_j) = \frac{1}{2}|x_i - x_j|^2$$

$$d_R(x_i, x_j) = \frac{1}{K}\sum_{k=1}^{K} |y_{ki} - y_{kj}|^2$$

We try to minimize the disntance between d_M and d_R:

$$W^* = \arg\min_W \sum_{(i,j)\in \text{Sample}} (d_M(x_i, x_j) - d_R(x_i, x_j))^2$$

Metric-Learning-Based Hashing

$$\text{sim}(x_i, x_j) = x_i^T M x_j = (Gx_i)^T (Gx_j)$$

Semisupervised Hashing

$$\sum_k \left[\sum_{(i,j)\in +} y_{ki} y_{kj} - \sum_{(i,j)\in -} y_{ki} y_{kj} \right]$$

Column Generation Hashing

Triplewise or listwise give more information.

Let $T = \{(q, x^+, ^-)|sim(q, x^+) > sim(q, x^-)\}$ be a set of triplet.

The problem can formulate as:

$$\arg\min_{w,t} \sum_i^{|T|} t_i + C|w|$$

Subject to:

$$w \succeq \mathbf{0}, t \succeq \mathbf{0}$$
$$d(q_i, x_i^+) - d(q_i, x_i^-) \geq 1 - t_i, \ \forall i$$

Ranking Supervised Hashing

Given some query point $\{q_i\}$, try to learn

$$d(q_m, x_j) < d(q_m, x_i) \Leftrightarrow H(q_m)^T H(x_j) < H(q_m)^T H(x_i)$$

Circulant Binary Embedding

Deep Learning for Hashing

> [Table 3] Cropped from paper

Hashing Methods	Data Domain	Learning Paradigm	Feature?
Semantic Hashing	Text	Unsupervised	No
RBM	Text, Image	Supervised	No
Deep Hashing	Image	Unsupervised	No
Supervised Deep Hashing	Image	Supervised	No
CNN Hashing	Image	Supervised	Yes
Deep Semantic Ranking Hashing	Image	Supervised	Yes
DNN Hashing	Image	Supervised	Yes

Open Issue and Future Direction

- Learning-Based hashing methods lack the theoretical guarantees
- Semantic gap between x and $h(x)$: May integral with binary code and representation learning