The Power of Asymmetry in Binary Hashing

Behnam Neyshabur    Yury Makarychev
Toyota Technological Institute at Chicago

Russ Salakhutdinov    Nati Srebro
University of Toronto    Technion/TTIC
Search by Image
Search by Image

\[ f(\text{image}) = 0110100011110 \]

Database of hashed image patches

- \( 1010100111100 \)
- \( 1011101001101 \)
- \( 0001111001001 \)
- \( 0110110010110 \)

\[ f(\text{image}) = 0110110010110 \]

"Brompton"

\[ f(\text{image}) = 0110110010110 \]
Binary Hashing

\[ \{\pm 1\}^k \cup \{\pm 1\}^k \cup 0/1 \]

\[ \left[ d_{\text{hamming}}(f(x), f(x')) < \theta \right] \approx \text{Sim}(x, x') \]
Binary Hashing

\[ \{\pm 1\}^k \cup \{\pm 1\}^k \cup 0/1 \]

\[ d_{\text{hamming}}( f(x), g(x') ) < \theta \] \approx \text{Sim} (x, x')

Even if \( \text{Sim}(x, x') \) is symmetric and “well behaved”:

- Use \( f(x) \) to hash objects in the database
- Use \( g(x) \) to hash queries

- No additional memory, communication or computation to perform query
- No increase in hash table size or lookup complexity
- We will show: shorter bit length; higher accuracy
Outline

- **Theoretical Observation:**
  Capturing similarity with arbitrary binary hashes

- **Empirical Investigation:**
  Database lookup with linear threshold hashes
Capturing Similarity with an Arbitrary Binary Code

- Given similarity $S(x,x')$ over objects $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$, want mapping $f: \mathcal{X} \rightarrow \pm 1^k$, s.t.
  $$S_{ij} = S(x_i,x_j) = [d(f(x_i), f(x_j)) < \theta]$$
- $f(\cdot)$ arbitrary, specified by $u_1, \ldots, u_n$, $u_i = f(x_i)$
- What is shortest $k$ possible?
  $$\exists \ u_1, \ldots, u_n \in \pm 1^k, \ \theta \in \mathbb{R} \ \ \ \forall_{ij} S_{ij} = [d(u_i, u_j) < \theta]$$

- What is shortest $k$ if we allow asymmetry?
  $$\exists \ u_1, \ldots, u_n, v_1, \ldots, v_n \in \pm 1^k, \ \theta \in \mathbb{R} \ \ \ \forall_{ij} S_{ij} = [d(u_i, v_j) < \theta]$$
  $$v_j = g(x_j)$$
The Power of Asymmetry for Arbitrary Binary Hashes

Theorem: For any $r$, there exists a set of points in Euclidean space, s.t. to capture

$$S(x_i,x_j) = [ |x_i-x_j| < 1 ]$$

using a symmetric binary hash we need $k_{sym} \geq 2^{r-1}$ bits, but using an asymmetric hash, we need only $k_{asym} \leq 2r$ bits.

$$X^T X = \frac{1}{n} \begin{bmatrix} n & -1 & \cdots & -1 & 1 & 1 & \cdots & 1 \\ -1 & n & \cdots & -1 & 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & n & 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 & n & -1 & \cdots & -1 \\ 1 & 1 & \cdots & 1 & -1 & n & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 & -1 & -1 & \cdots & n \end{bmatrix}$$

$n=2^r$
Binary Hashing as Matrix Factorization

\[ d_{\text{hamming}}(u_i, u_j) < \theta \] \approx \text{Sim}(x, x')
Binary Hashing as Matrix Factorization

\[
[ \langle u_i, u_j \rangle < \theta ] \approx \text{Sim}(x,x')
\]

Given similarity matrix \( S \in \{\pm 1\}^{n \times n} : 

\[
\begin{align*}
\min k & \quad \text{s.t.} \quad U \in \{\pm 1\}^{k \times n} \quad \text{symmetric} \\
\quad \text{sign}(U^T U - \theta) &= S
\end{align*}
\]

\[
\begin{align*}
\min k & \quad \text{s.t.} \quad U,V \in \{\pm 1\}^{k \times n} \quad \text{asymmetric} \\
\quad \text{sign}(U^T V - \theta) &= S
\end{align*}
\]
Bonus: Approximation Algorithm via SDP Relaxation

Given similarity matrix $S \in \{\pm 1\}^{n \times n}$:

$$\min k \quad \text{s.t.} \quad U \in \{\pm 1\}^{k \times n} \quad \text{symmetric}$$
$$\text{sign}(Y - \theta) = S$$
$$Y = U^T U$$

$$\min k \quad \text{s.t.} \quad U,V \in \{\pm 1\}^{k \times n} \quad \text{asymmetric}$$
$$\text{sign}(Y - \theta) = S$$
$$Y = U^T V$$
Bonus: Approximation Algorithm via SDP Relaxation

Given similarity matrix $S \in \{\pm 1\}^{n \times n}$:

\[
\begin{align*}
\min k \quad & \text{s.t.} \quad \|U_i\| \leq \sqrt{k} & \text{symmetric} \\
& S_{ij}(Y_{ij} - \theta) \geq 1 \\
& Y = U^T U
\end{align*}
\]

\[
\begin{align*}
\min k \quad & \text{s.t.} \quad \|U_i\|, \|V_j\| \leq \sqrt{k} & \text{asymmetric} \\
& S_{ij}(Y_{ij} - \theta) \geq 1 \\
& Y = U^T V
\end{align*}
\]
Bonus: Approximation Algorithm via SDP Relaxation

Given similarity matrix $S \in \{-1, 1\}^{n \times n}$:

\[
\begin{align*}
\min & \quad \|Y\|_{\max} \\
\text{s.t.} & \quad S_{ij}(Y_{ij} - \theta) \geq 1 \\
& \quad Y \succeq 0
\end{align*}
\]

\[
\min \|Y\|_{\max} \quad \text{s.t.} \quad S_{ij}(Y_{ij} - \theta) \geq 1
\]

\[
\|Y\|_{\max}^2 = \min_{Y = LR} \|L\|_{\infty,2} \|R\|_{\infty,2}
\]
SDP Relaxation—Rounding

\[ \|Y\|_{\text{max}}^2 = \min_{Y=LR} \|L\|_{\infty,2} \|R\|_{\infty,2} \]

Using random vectors \( z_1, \ldots, z_k \):

\[ u_i = \text{sign}(L_i^T z_i) \]
\[ v_i = \text{sign}(R_i^T z_i) \]

\[ k = O( (k_{\text{opt}})^2 \log n ) \]
The Power of Asymmetry for Arbitrary Binary Hashes

**Theorem:** For any $r$, there exists a set of points in Euclidean space, s.t. to capture

$$S(x_i, x_j) = [ |x_i - x_j| < 1 ]$$

using a symmetric binary hash we need $k_{\text{sym}} \geq 2^{r-1}$ bits, but using an asymmetric hash, we need only $k_{\text{asym}} \leq 2r$ bits.

$$X^T X = \frac{1}{n} \begin{bmatrix} n & -1 & \cdots & -1 & 1 & 1 & \cdots & 1 \\ -1 & n & \cdots & -1 & 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & \cdots & n & 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 & n & -1 & \cdots & -1 \\ 1 & 1 & \cdots & 1 & -1 & n & \cdots & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 & -1 & -1 & \cdots & n \end{bmatrix}$$

$n = 2^r$
Arbitrary Binary Hashes: Empirical Evaluation

**10D Uniform**

- Symmetric
- Asymmetric

**LabelMe**

- Symmetric
- Asymmetric
Parametric Mappings

• \( f(x) = \phi_F(x) \) has some parametric form
  – E.g. \( \phi_F(x) = \text{sign}(Fx), \ F \in \mathbb{R}^{k \times d} \)
  – Could be more complex, e.g. multi-layer network, kernel based, etc.

• Why restrict \( f(\cdot) \)?
  – Generalization: learn \( F \) using objects \( x_1, \ldots, x_n \), then receive new objects \( x \) as queries
  – Compactness of representation

• Asymmetric extension:
  \[ S(x,x') \approx \left[ d(\phi_F(x), \phi_G(x')) < \theta \right] \]
  – Learn params \( F, G \) from training data (= pairs of database objects)
  – Hash database objects using \( \phi_G \)
  – Hash queries using \( \phi_F \)
A bit on optimization

• Loss function : $L(\phi_F(x), \phi_G(x'), S(x,x'))$

• Parameters $F, G$

• For $\phi_F(x) = \text{sign}(Fx)$:
  – Updating single row of $F$ (responsible for single bit in the hashing) entails solving a single weighted binary classification problem
Empirical Results using Asymmetric Linear Threshold Hashes

Comparison with learning symmetric linear threshold hashes using Minimum Loss Hashing [Norouzi Fleet ICML 2011]

Training on all pairs of database objects, average precision measured on held out query objects.
Even More Power:
Searching a Static Database

\[ S(x_{\text{query}}, x_i) \approx \left[ d( f(x_{\text{query}}), g(x_i) ) < \theta \right] \]
Even More Power: Searching a Static Database

\[ S(x_{\text{query}}, x_i) \approx \left[ d(\phi_F(x_{\text{query}}), v_i) \right) < \theta \]

- No need to generalize: only used on database objects
- Stored in database hash, no need for compactness
- Can use arbitrary mapping \( g(x_i) = v_i \)

- Learn parametric \( \phi_F(\cdot) \) and arbitrary vectors \( v_1, \ldots, v_n \) such that on \( x_1, \ldots, x_n \):
  \[ S(x_i, x_j) \approx \left[ d(\phi_F(x_i), v_j) \right) < \theta \]

- For query \( x \), use \( d(\phi_F(x), v_j) \) to aprox \( S(x, x_j) \)
e.g. find \( v_j \) in DB with small \( d(\phi_F(x), v_j) \)
Empirical Results using Linear-Threshold-to-Arbitrary Hashes

- **LabelMe**
  - Symmetric (MLH)
  - Asym Lin:Lin
  - Asym Lin:Arb

- **MNIST**
  - Lin:Arb 64bits
  - MLH 64bits
  - MLH 256bits

- **Recall**
  - Lin:Arb 64bits
  - MLH 64bits
  - MLH 256bits
Summary


• Binary Hashing
  – Fast similarity approximation
  – Approximate retrieval
  – Nearest Neighbor search
  – Locality Sensitive Hashing (LSH)

• In many applications: want short bit-length
  – Smaller hash tables
  – Super-fast hamming distance eval on short words
  – Fewer bits to transmit

• **Asymmetric hashes can enable better approximation with shorter bit-length!**
  – Even if similarity function symmetric and well-behaves
  – In most applications: no additional computational or memory costs