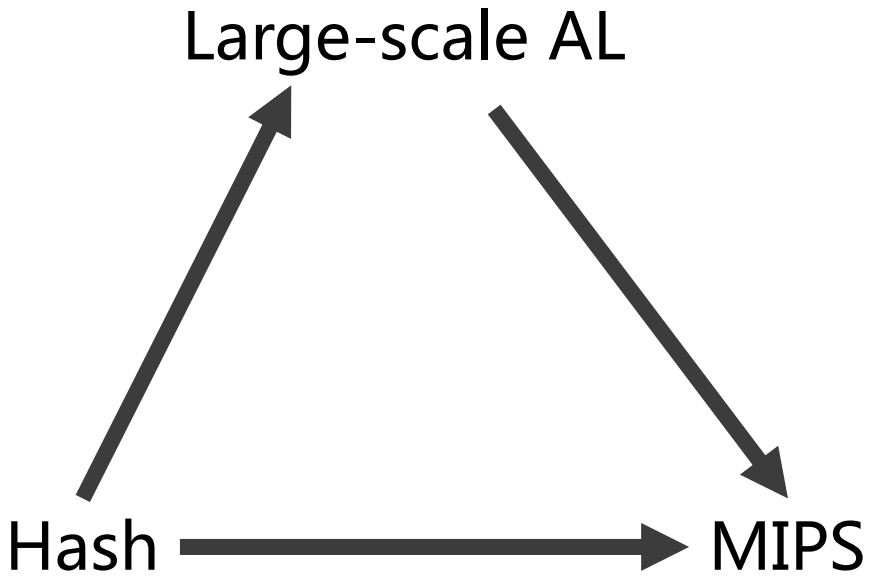


2017.10.13

Similarity Search in High Dimension via Hashing	VLDB 1999
Locality-Sensitive Hashing Scheme Based on p-Stable Distributions	ACM SCG 2004
Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS)	NIPS 2014
Accelerated Large Scale Optimization by Concomitant Hashing	ECCV 2012
Scalable and Sustainable Deep Learning via Randomized Hashing	SIG KDD 2017

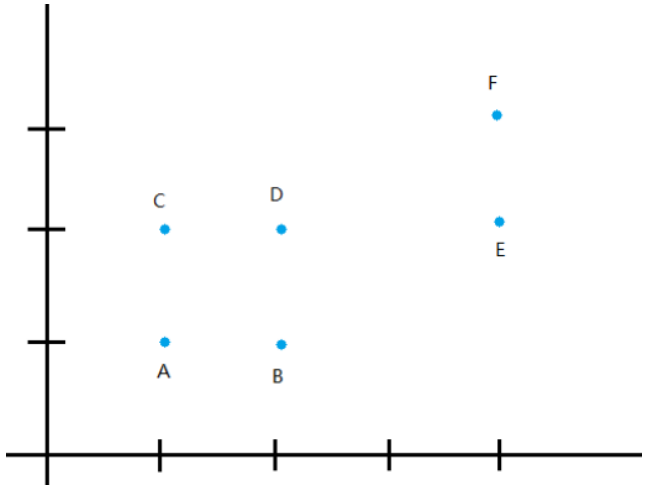
LSH

A=(1,1) B=(2,1) C=(1,2)

D=(2,2) E=(4,2) F=(4,3)

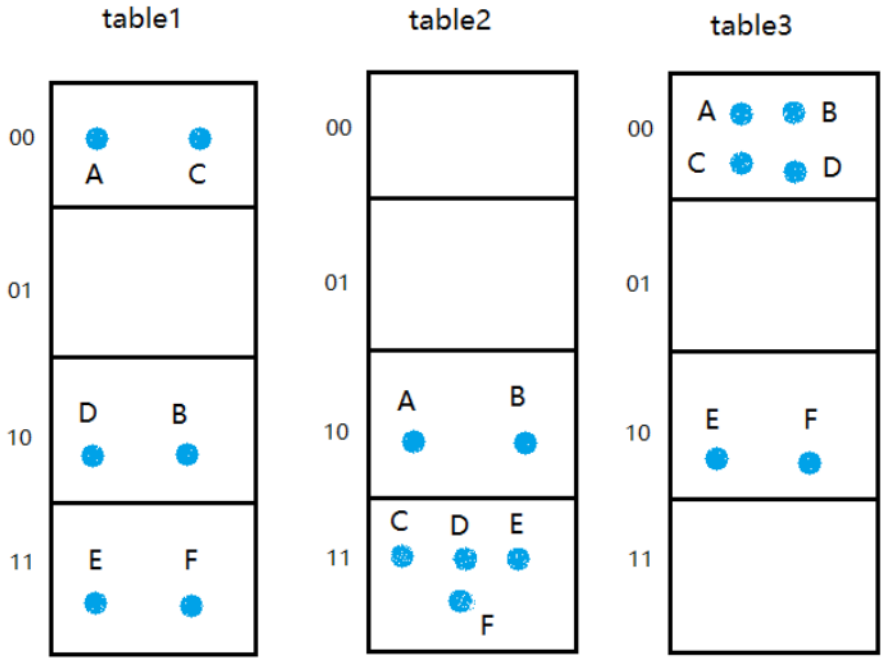
Hamming Embed

v(A) = 10001000
v(B) = 11001000
v(C) = 10001100
v(D) = 11001100
v(E) = 11111100
v(F) = 11111110



$K = 2, L = 3$

$g_1 = 2,4$
$g_2 = 1,6$
$g_3 = 3,8$



q = (4, 4)

$g_1(q) = [1,1]^T$
$g_2(q) = [1,1]^T$
$g_3(q) = [1,1]^T$

C D E F

p-stable LSH

p-stable

Stable Distribution: A distribution $\mathcal{D}$ over $\Re$ is called *p-stable*, if there exists $p \geq 0$ such that for any n real numbers $v_1 \ldots v_n$ and i.i.d. variables $X_1 \ldots X_n$ with distribution $\mathcal{D}$, the random variable $\sum_i v_i X_i$ has the same distribution as the variable $(\sum_i |v_i|^p)^{1/p} X$, where X is a random variable with distribution $\mathcal{D}$.

$$(a \cdot v_1 - a \cdot v_2) \longrightarrow \|v_1 - v_2\|_p X$$

$$h_{a,b}(v) = \left\lfloor \frac{a \cdot v + b}{r} \right\rfloor \qquad b \sim \mathrm{U}(0, r)$$

Asymmetric LSH

Given a giant data vector collection S of size N, where $S \subset \mathbb{R}^D$ and a given query point $q \in \mathbb{R}^D$. We are interested in searching for $p \in S$ which maximizes (or approximately maximizes) the inner product $q^T p$.

$$p = \arg\max_{x \in \mathcal{S}} \quad q^T x$$

The MIPS problem is related to near neighbor search (NNS), which instead requires computing

$$p = \arg\min_{x \in \mathcal{S}} \|q - x\|_2^2 = \arg\min_{x \in \mathcal{S}} (\|x\|_2^2 - 2q^T x)$$

Recommender systems

Large-scale object detection with DPM (Deformable Part Model)

Multi-class (and/or multi-label) prediction

Asymmetric LSH

The models for multi-class SVM (or logistic regression) learn a weight vector w_i for each of the class label i.

$$y_{test} = \arg\max_{i \in \mathcal{L}} \; x_{test}^T \, w_i$$

Theorem 1 *There cannot exist any LSH family for MIPS.*

$$Sim(x, x) = x^T x = ||x||_2^2 \qquad Sim(x, y) = y^T x > ||x||_2^2 + C$$

$$\{h(x) = h(x)\} \qquad \{h(x) = h(y)\} \qquad Sim(x, y) \; > Sim(x, x)$$

process

$$\|x_i\|_2 \leq U < 1 \qquad S(x) = \frac{U}{M} \times x; \qquad M = max_{x_i \in \mathcal{S}} \|x_i\|_2;$$

$$P : \mathbb{R}^D \mapsto \mathbb{R}^{D+m} \qquad Q : \mathbb{R}^D \mapsto \mathbb{R}^{D+m}$$

$$P(x) = [x; \|x\|_2^2; \|x\|_2^4; ....; \|x\|_2^{2^m}] \qquad Q(x) = [x; 1/2; 1/2; ....; 1/2]$$

$$Q(q)^T P(x_i) = q^T x_i + \frac{1}{2}(\|x_i\|_2^2 + \|x_i\|_2^4 + ... + \|x_i\|_2^{2^m})$$

$$\|P(x_i)\|_2^2 = \|x_i\|_2^2 + \|x_i\|_2^4 + ... + \|x_i\|_2^{2^{m+1}}$$

$$\|Q(q) - P(x_i)\|_2^2 = (1 + m/4) - 2q^T x_i + \|x_i\|_2^{2^{m+1}} \qquad \arg\max_{x \in \mathcal{S}} q^T x \simeq \arg\min_{x \in \mathcal{S}} \|Q(q) - P(x)\|_2$$

$$\mathcal{X} = \left\{ x \;\; | \;\; |\omega^T x| \leq |\omega^T x'|, \;\; \forall \;\; x' \notin \mathcal{X} \right\} \qquad \mathcal{X} = \left\{ x \;\; | \;\; |\omega^T x| \geq |\omega^T x'|, \;\; \forall \;\; x' \notin \mathcal{X} \right\}$$

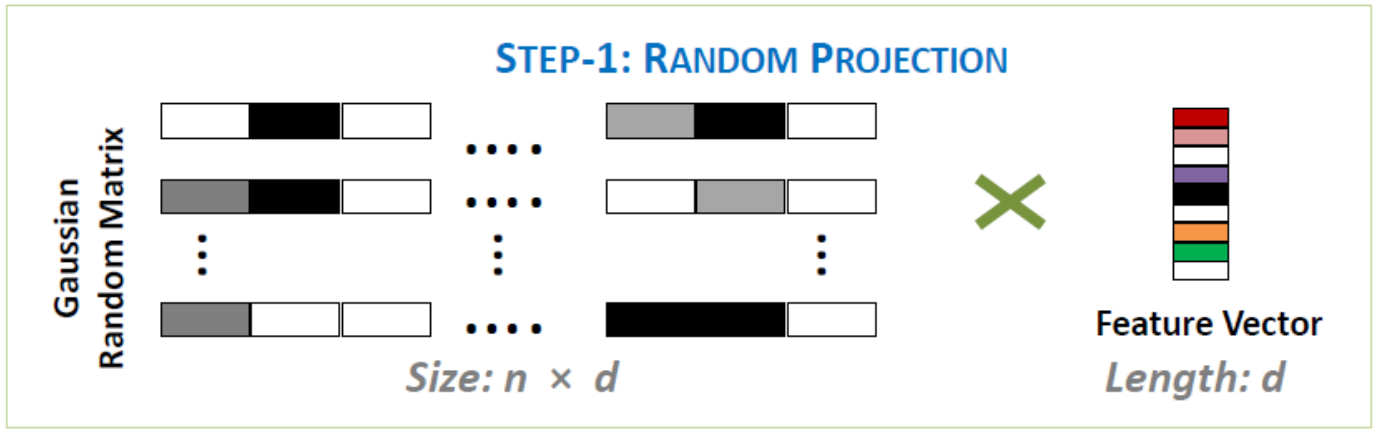
Min-Product: Active Learning SVM and Maximum Margin Clustering

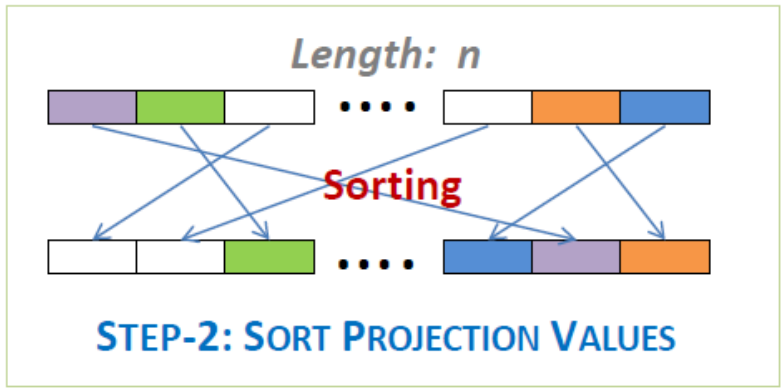
$$\min_{y \in \{-1,1\}^N, \; b, \; \xi \geq 0} \frac{1}{2} w^T w + C \sum_{i=1}^{N} \xi_i \;\; s.t. \;\; y_i \left(w^T x_i + b \right) \geq 1 - \xi_i, \;\; \forall i,$$

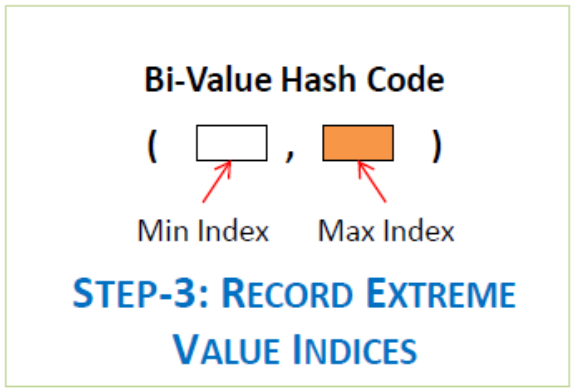
Max-Product: Sparse Optimization and Gaussian Process Regression

Maximal absolute correlation greedy selection:

$$\begin{aligned}
&(\mathbf{Lasso}): \;\; \min_{\alpha} \; \|x - D\alpha\|^2 + \lambda \|\alpha\|_1. \\
&(\mathbf{OMP}): \;\; \min_{\alpha} \; \|x - D\alpha\|^2, \;\; s.t. \; \|\alpha\|_0 \leq k. \\
&(\mathbf{GPR}): \;\; \min_{\alpha} \; \frac{1}{2} \alpha^T \left(K + \sigma^2 I \right) \alpha - y^T \alpha.
\end{aligned}$$

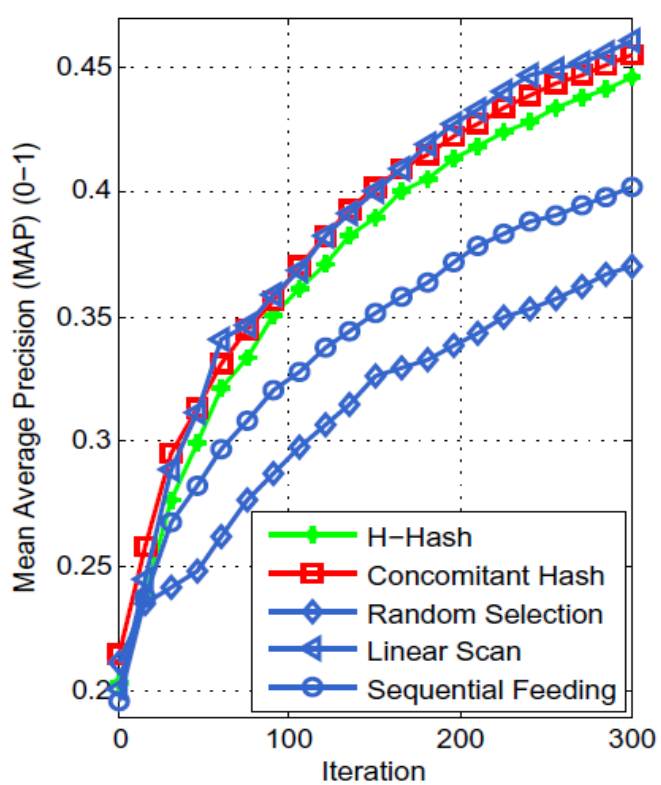

STEP-1: RANDOM PROJECTION
Gaussian Random Matrix
Size: n × d
Feature Vector
Length: d

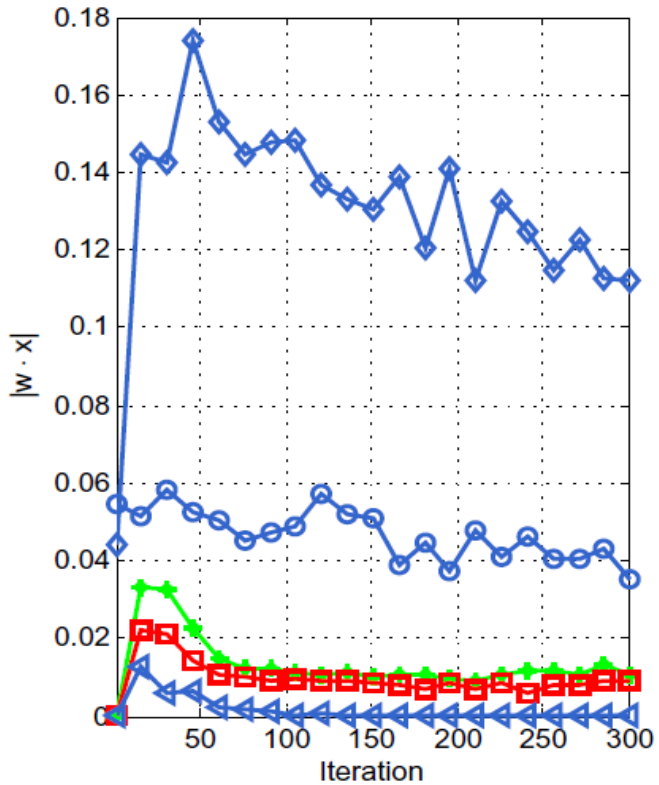

Length: n
Sorting
STEP-2: SORT PROJECTION VALUES


Bi-Value Hash Code
(,)
Min Index
Max Index
STEP-3: RECORD EXTREME VALUE INDICES

$$h(x) = \{h_1(x), h_2(x)\}$$

Proposition 2. *For any pair of normalized vectors* (x_1, x_2)*, the collision probability* $P[h(x_1) = h(x_2)]$ *depends only on their inner product:* $P[h(x_1) = h(x_2)] = g(x_1^T x_2)$*. The function* $g(\rho)$ *is symmetric about* $\rho = 0$*, and monotonically increasing on* $(0, 1)$*.*





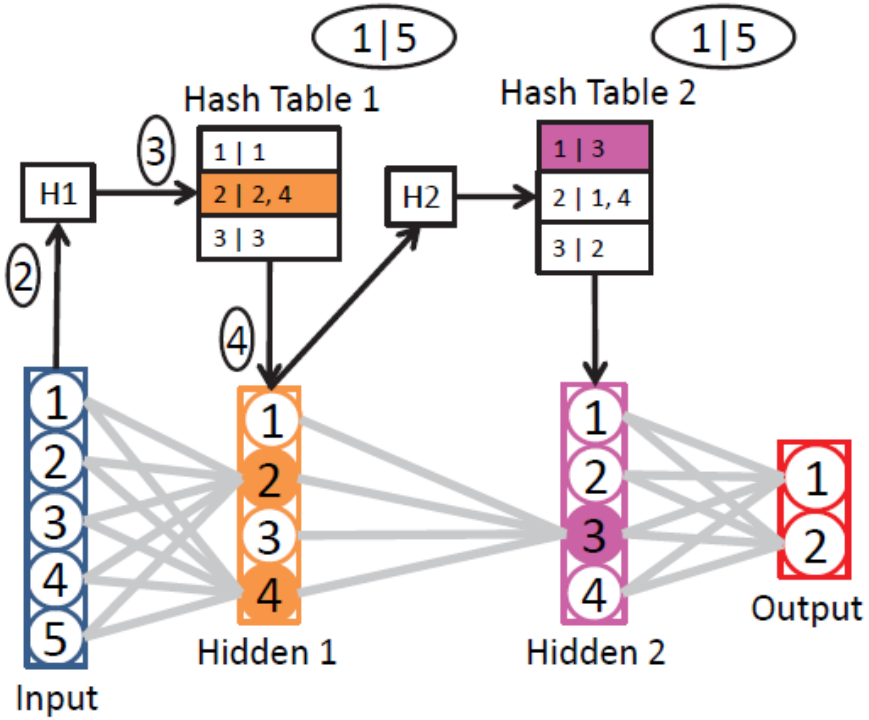


Figure 2: A visual representation of a neural network using randomized hashing (1) Build the hash tables by hashing the weights of each hidden layer (one time operation) (2) Hash the layer's input using the layer's randomized hash function (3) Query the layer's hash table(s) for the active set AS (4) Only perform forward and back-propagation on the neurons in the active set. The solid-colored neurons in the hidden layer are the active neurons. (5) Update the AS weights and the hash tables by rehashing the updated weights to new hash locations.

Algorithm 1 Deep Learning with Randomized Hashing

// HF_l - Layer l Hash Function
// HT_l - Layer l Hash Tables
// AS_l - Layer l Active Set
// $\theta^l_{AS} \in W^l_{AS}$, b^l_{AS} - Layer l Active Set parameters
Randomly initialize parameters W^l, b^l for each layer l
HF_l = constructHashFunction(k, L)
HT_l = constructHashTable(W^l, HF_l)
while not stopping criteria **do**
  for each training epoch **do**
    // Forward Propagation
    for layer $l = 1 \ldots N$ **do**
      $\text{fingerprint}_l = \text{HF}_l(a_l)$
      AS_l = collectActiveSet(HT_l, fingerprint_l)
      for each node i in AS_l **do**
        $a_i^{l+1} = f(W_i^l a_i^l + b_i^l)$
      end for
    end for
    // Backpropagation
    for layer $l = 1 \ldots N$ **do**
      $\Delta J(\theta^l_{AS})$ = computeGradient(θ^l_{AS}, AS_l)
      θ^l_{AS} = updateParameters(θ^l_{AS}, $\Delta J(\theta_{AS})$)
    end for
    for each Layer l -> updateHashTables(HF_l, HT_l, θ^l)
  end for
end while