Asymmetric LSH (ALSH) for Sublinear Time Maximum Inner Product Search (MIPS)

Anshumali Shrivastava and Ping Li

Cornell University and Rutgers University

December 9th 2014

The Maximum Inner Product Search (MIPS) Problem

Given a query $q \in \mathbb{R}^D$ and a giant collection $\mathcal{C}$ of N vectors in $\mathbb{R}^D$, search for $p \in \mathcal{C}$ s.t.,

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

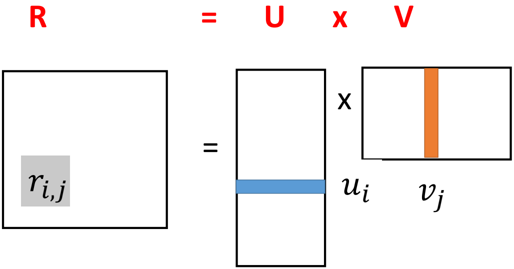
- Worst case O(N) for any query. N is huge.
- O(N) quite expensive for frequent queries.

Our goal is to solve this efficiently (something sub-linear)

Not same as the classical near-neighbor search problem.

$$\arg\min_{x \in \mathcal{C}} ||q - x||_2^2 = \arg\min_{x \in \mathcal{C}} (||x||_2^2 - 2q^T x) \neq \arg\max_{x \in \mathcal{C}} \quad q^T x$$

Scenario 1: User-Item Recommendations



- Matrix factorizations for collaborative filtering.
- Given a user u_i, the best item to recommend is a MIPS instance

$$Item = \arg\max_j \; r_{i,j} = \arg\max_j \; u_i^T v_j$$

Vectors u_i and v_j are learned. No control over norms.

Scenario 2: Multi-Class Prediction

Standard multi-class SVM in practice learns a weight vector w_i for each of the class label $i \in \mathcal{L}$.

Predicting a new x_{test}, is a MIPS instance:

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

w_is are learned and usually have varying norms.

Note: Fine grain ImageNet classification has 100,000 classes, in practice this number can be much higher.

Scenario 3: Web-Scale Deep Architectures

Activation of hidden node i monotonic in $x^T w_i$.

MAXOUT (Goodfellow et. al. 13) only requires updating hidden nodes having max activations.

$$h(x) = \max_j \quad x^T w_j;$$

Max-Product Networks (Gens & Domingos 12)
Adaptive Dropouts (Ba & Frey 13)

Problems:

- Each iteration, find max activation for every data point in every layer.
- **Networks with millions (or billions) of hidden nodes ?**

Efficient MIPS $\Longrightarrow$ Fast Training and Testing of Giant Networks

Note: No control over norms.

Many More !!

- Max of affine function approximations. (Prof. Nesterov's Talk)
- Active Learning
- Deformable parts model in vision
- Cutting plane methods and Greedy coordinate ascent.
- Greedy matching pursuit.
- ...

Outline of the talk

- Brief Introduction to Locality Sensitive Hashing (LSH).
- A Negative Result, Symmetry Cannot Solve MIPS.
- Construction of Asymmetric Hashing (ALSH) for MIPS
- Experiments.
- Extensions and More Connections.

Locality Sensitive Hashing (LSH)

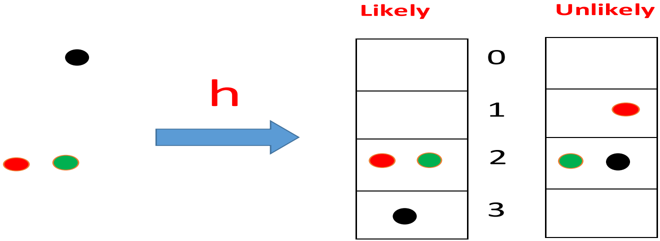
Hashing: Function (randomized) h that maps a given data vector $x \in \mathbb{R}^D$ to an integer key $h : \mathbb{R}^D \Rightarrow [0, 1, 2, ..., N]$

Locality Sensitive: Additional property

$$Pr_h\Big[h(x) = h(y)\Big] = f(sim(x, y)),$$

where f is monotonically increasing.

Similar points are more likely to have the same hash value (hash collision) compared to dissimilar points.



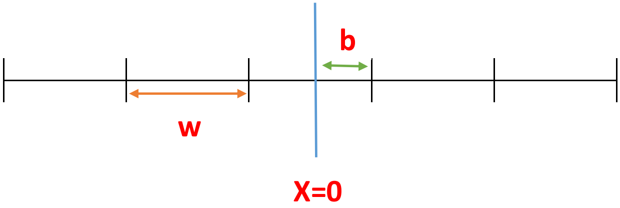
Hashing for L_2 Distance

L2-LSH

$$h_w(x) = \left\lfloor \frac{r^T x + b}{w} \right\rfloor$$

where $r \in R^D$ drawn independently from $N(0, \mathcal{I})$, b is drawn uniformly from $[0, w]$. w is a parameter. $\lfloor\rfloor$ is the floor operation.

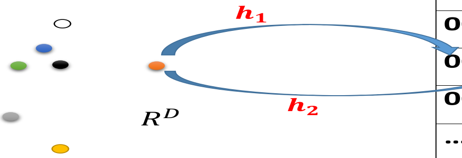
It can be shown that $Pr(h_w(x) = h_w(y))$ **is monotonic in** $||x - y||_2$



---

b **hurts**. 1-bit (Sign) or 2-bit hashing is preferred (Li et. al. ICML 14)

Sub-linear Near Neighbor Search: Idea



$h_1, h_2 : R^D \rightarrow \{0, 1, 2, 3\}$

h_1	h_2	Buckets (pointers only)
00	00	...
00	01	...
00	10	Empty
...	...	...
11	11	...

- Given query q, if $h_1(q) = 11$ **and** $h_2(q) = 01$, then probe bucket with index **1101**. It is a good bucket !!
- (LSH Property) $h_i(q) = h_i(x)$ is an indicator of high similarity between q and x for all i.

The Classical LSH Algorithm

Table 1

h_1^1	...	h_K^1	Buckets
00	...	00	...
00	...	01	...
00	...	10	Empty
...	...	...	...
11	...	11	...

...

Table L

h_1^L	...	h_K^L	Buckets
00	...	00	...
00	...	01	...
00	...	10	
...	...	...	...
11	...	11	Empty

Querying: Report union of L buckets.

- We can use K concatenation. To improve recall we can repeat the process L times and take union.
- K and L are two knobs.

Theory says we have a sweet spot. Provable sub-linear algorithm.

Negative Result: LSH Cannot Solve MIPS !!

- For inner products, we can have x and y, s.t $x^T y > x^T x$. Self similarity is not the highest similarity.

- Under any hash function $Pr(h(x) = h(x)) = 1$. But we need

$$Pr(h(x) = h(y)) > Pr(h(x) = h(x)) = 1$$

We cannot have Locality Sensitive Hashing for inner products !

Extend Framework: Asymmetric LSH (ALSH)

Main concern: $Pr(h(x) = h(x)) = 1$, an obvious identity.

How about asymmetry ?

- We can use $P(.)$ for creating buckets.
- While querying probe buckets using $Q(.)$ with $P \neq Q$.

$$Pr(Q(x) = P(x)) \neq 1$$

All we need is $Pr(Q(q) = P(x))$ to be monotonic in $q^T x$.

- Same proofs work !!
- Symmetry in hashing is unnecessary part of LSH definition.

Fine ! How do I construct P and Q ?

Reduce the problem to known domain we are comfortable with.
(Tea-kettle principle in mathematics)

Construction

Known: $Pr\big(h(q) = h(x)\big) = f(||q||_2^2 + ||x||_2^2 - 2q^Tx)$ **(L2 LSH)**

Idea: Construct P and Q s.t. $||Q(q)||_2^2 + ||P(x)||_2^2 - 2Q(q)^TP(x)$ is monotonic (or approximately) in q^Tx. This is also sufficient !!
(We can expand dimensions as part of P and Q.)

Pre-Processing
Scale data x, such that

$$||x_i|| < 1 \quad \forall x_i \in \mathcal{C}$$

$$P(x_i) = [x_i; ||x_i||_2^2; ||x_i||_2^4; ....; ||x_i||_2^{2^m}]$$

Querying q

$$Q(q) = [q; 1/2; 1/2; ....; 1/2]$$

$$||Q(q) - P(x_i)||_2^2 = (||q||^2 + m/4) - 2q^Tx_i + ||x_i||_2^{2^{m+1}}$$

$||x_i||^{2^{m+1}} \to 0$, and m is constant. We therefore have

$$\arg\max_{x \in \mathcal{S}} q^Tx \simeq \arg\min_{x \in \mathcal{S}} ||Q(q) - P(x)||_2$$

Main Result: First Provable Hashing Algorithm for MIPS

Theorem

*(**Approximate MIPS is Efficient**) For the problem of c-approximate MIPS, one can construct a data structure having $O(n^{\rho^*}\log n)$ query time and space $O(n^{1+\rho^*})$, where $\rho^* < 1$.*

$$\rho_u^* = \min_{0<U<1, m\in N, r} \frac{\log F_r\left(\sqrt{m/2 - 2S_0\left(\frac{U^2}{M^2}\right) + 2U^{2^{m+1}}}\right)}{\log F_r\left(\sqrt{m/2 - 2cS_0\left(\frac{U^2}{M^2}\right)}\right)}$$

$$s.t. \quad \frac{U^{(2^{m+1}-2)}M^2}{S_0} < 1 - c,$$

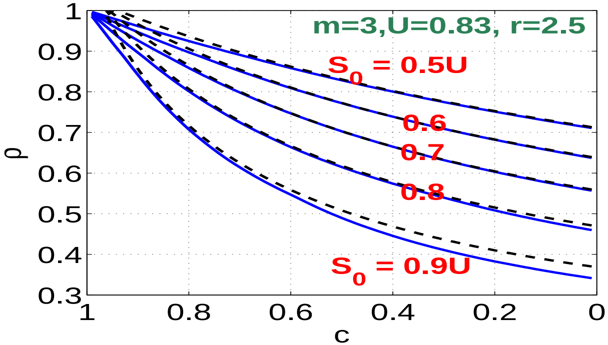
The only assumption needed is bounded norms, i.e. M is finite.

Why parameters do not bother us ?

Theory

- Even classical LSH requires computing K and L for a given c-approximate instance, and there is no fixed universal choice.
- Not hyper-parameters, can be exactly computed.
- In theory, **we do not lose any properties.**

In practice, there is a good choice: $m = 3, \quad U = 0.83, \quad r = 2.5$



The Final Algorithm

Preprocessing: Scale $\implies$ Append 3 Numbers $\implies$ Usual L2 - LSH

- Scale $x \in \mathcal{C}$ to have norm ≤ 0.83
- Append $||x_i||^2$, $||x_i||^4$, and $||x_i||^8$ to vector x_i. (just 3 scalars)
- Use standard L2 hash to create hash tables.

Querying: Append 3 Numbers $\implies$ Usual L2 - LSH

- Append 0.5 three times to the query q. (just three 0.5s)
- Use standard L2 hash on the transformed query to probe buckets.

- **A surprisingly simple algorithm.**
- Trivial to implement.

How much benefit compared to standard hash functions ?

Datasets and Settings

Datasets

- Movielens (10M)
- Netflix

Latent User Item Features:

- Matrix factorization to generate user and item latent features.
- Dimension: 150 for Movielens and 300 for Netflix.

Given a user as query, finding the best item is a MIPS instance.
Aim: Evaluate and compare computational savings.

Competing Hash Functions

- ALSH (proposed)
- L2-LSH (LSH for L2 distance)
- Signed Random Projections (SRP)

Ranking Quality Based on Hash Collisions

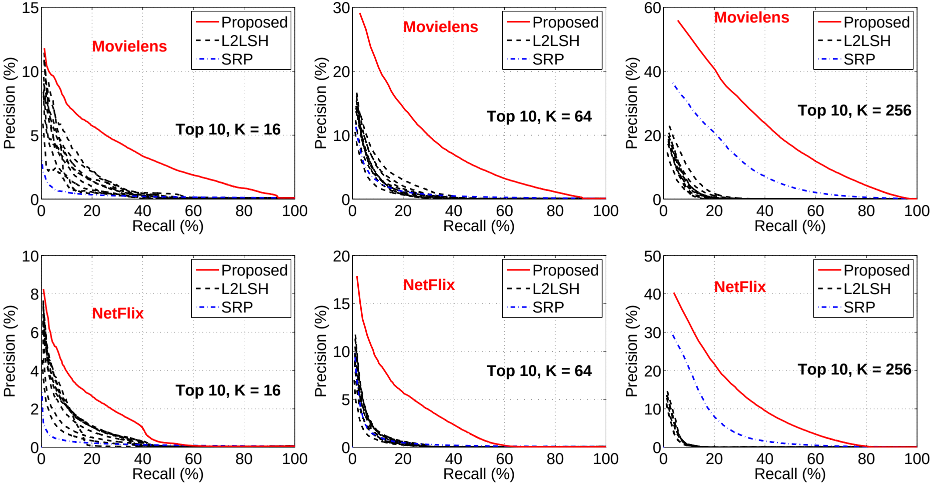


Figure: Precision-Recall curves (**higher is better**), for top-10 items.

- Previous evaluations are not true indicators of computational savings.
- **Bucketing Experiments:** We need to ensure comparison is fair.
- Find best $K \in [1 - \mathbf{40}]$ and $L \in [1 - \mathbf{400}]$ for each recall.
- Summary of **16000** experiments. Averaged over 2000 queries.

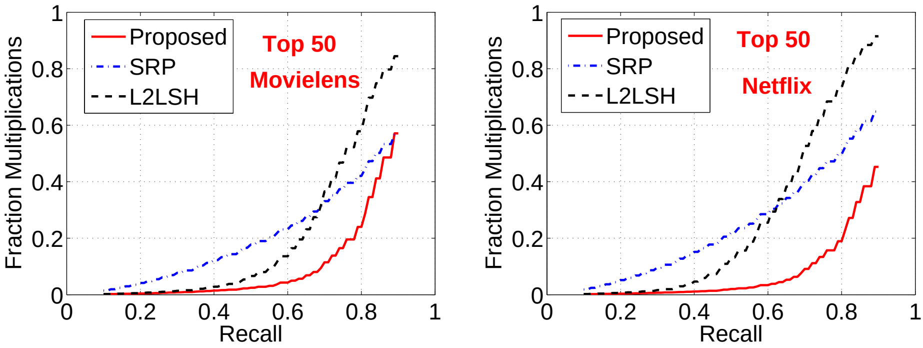


Figure: Mean inner products computed, relative to a linear scan. **Lower is better**.

A Generic Recipe for Constructing New ALSHs

In this work

- LSH for $\mathcal{S}'(q,x) = ||q - x||_2$. $\implies$ ALSH for $S(q,x) = q^Tx$.

Can we do better ? (Yes !)

- Signed Random Projections (SRP) leads to more informative hashes than L2-LSH (Li et. al. ICML 2014). **SRP as Base LSH** ?
- Use $\mathcal{S}'(q,x) = \frac{q^Tx}{||q||_2||x||_2}$, we can construct a different set of P and Q.
- **For binary data:** Minwise hashing and $\mathcal{S}'(q,x) = \frac{|q \cap x|}{|q \cup x|}$. (Jaccard)

Asymmetric Minwise Hashing for Set Intersection

(Shrivastava & Li 2014) "Asymmetric Minwise Hashing"

For sparse data (not necessarily binary):

- Minwise hashing is better than SRP (Shrivastava & Li AISTATS 14)
- $\mathcal{S}'(q,x) = \frac{|q \cap x|}{|q \cup x|}$ (Jaccard Similarity).
- Another P and Q with minwise hashing, **significant improvements**.

$$P(x) = [x; M - f_x; 0] \qquad Q(x) = [x; 0; M - f_x]$$

f_x is number of non-zeros in x, M is maximum sparsity.

Excercise in Construction of P and Q: **Expand dimensionality** and cancel out effect of terms we do not want in $\mathcal{S}'(q,x)$.

Conclusions

- We provide the first provable and practical hashing solution to MIPS.
- MIPS occurs as a subroutine in many machine learning application. All those applications will directly benefit from this work.
- Constructing asymmetric transformations to reduce MIPS to near-neighbor search was the key realization.
- In the task of item recommendation, we obtain significant savings compared to well known heuristics.
- Idea behind the ALSH construction is general and it connects MIPS to many known similarity search problems.

Acknowledgments

- **Funding Agencies:** NSF, ONR, and AFOSR.
- Reviewers of KDD 2014 and NIPS 2014
- Prof. Thorsten Joachims and Class of CS 6784 at Cornell University (Spring 2014) for feedbacks and suggestions.

Thanks for Your Attention !!