Asymmetric LSH for sublinear time complexity MIPS

Group 3 - Karthik, Shubham, Krishna Priya, Saurabh

Problem Statement

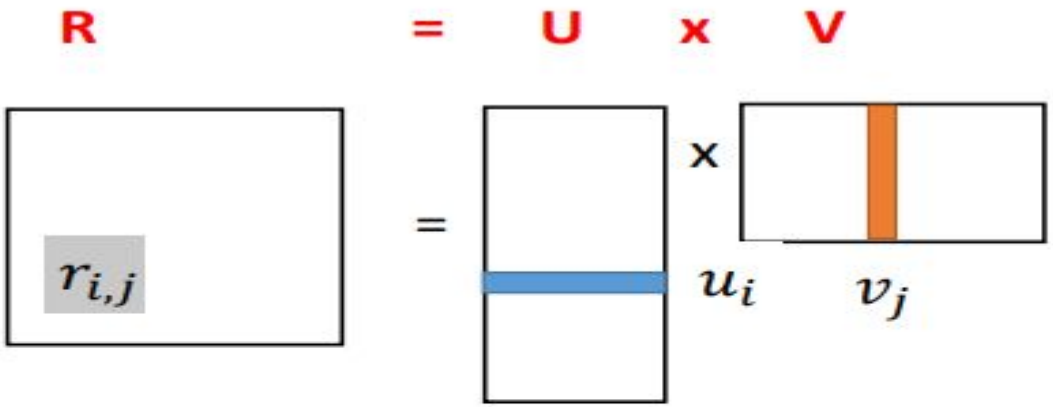
What is MIPS(Maximum Inner Product Search)?

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.

Scenario I - (User Recommendation System)



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

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

Scenario II - (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:

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

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

Goal

Solve the MIPS problem efficiently in sub-linear time.

Why is this necessary?

Assume x images are to be classified by multi-class svm which itself has 10^5 classes. Also the weight vector has dimension (d).

Time complexity = O((No. of queries(x)) X 10^5 X d)

If x is even around 10^5, The computation becomes very large.

Concepts to be discussed:

1. LSH
2. ALSH
3. Related Works

Locality Sensitive Hashing (LSH)

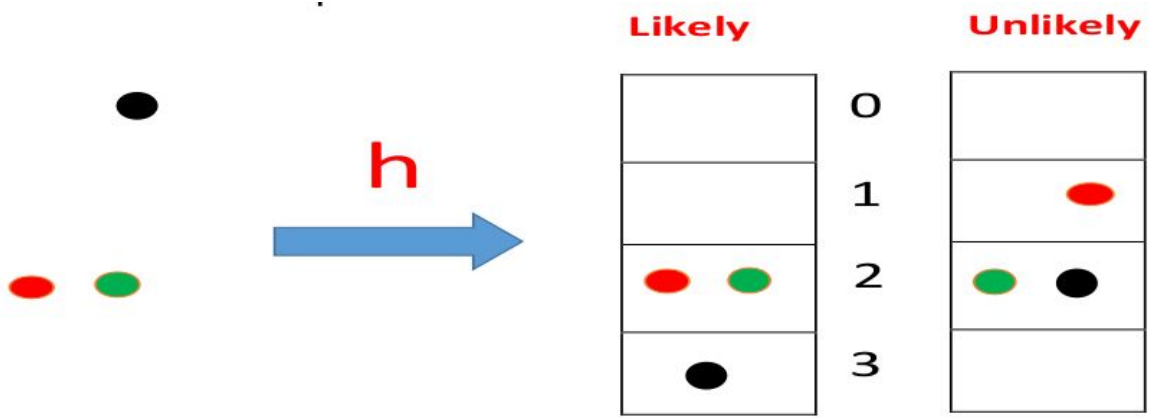
Hashing: Function (randomized) *h* that maps a given data vector $x \in R^D$ to an integer key $\boldsymbol{h} : R^D \Rightarrow [0, 1, 2, ..., N]$

Locality Sensitive: Additional property

$$Pr[h(x) = h(y)] = f(sim(x, y)),$$

where f is monotonically increasing.

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



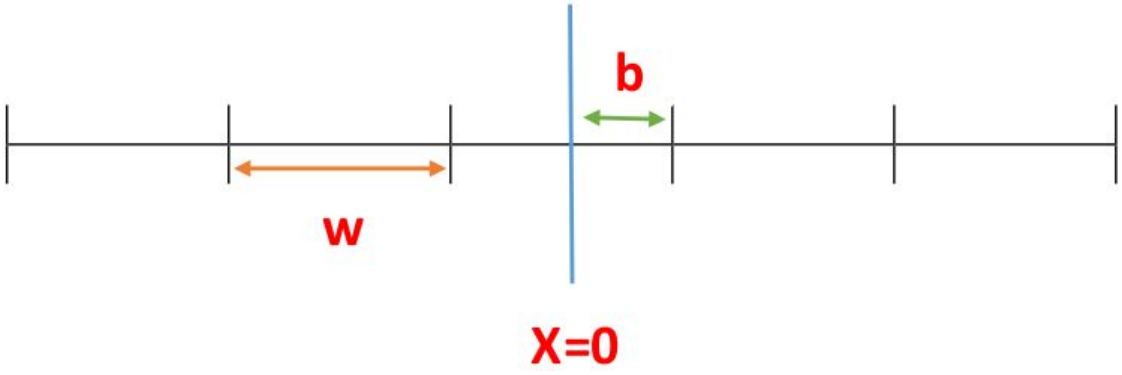
Hashing for L2 Distance

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** $||x - y||_2$



The collision probability under this scheme :

$$Pr(h^{L2}_{a,b}(x) = h^{L2}_{a,b}(y)) = F_r(d); \qquad F_r(d) = 1 - 2\Phi(-r/d) - \frac{2}{\sqrt{2\pi}(r/d)}\left(1 - e^{-(r/d)^2/2}\right)$$

, where **d** = $\|x-y\|_2 = \sqrt{(\|x\|_2^2 + \|y\|_2^2 - 2x^T y)}$

Note that the above euclidean distance won't be monotonic in terms of the inner product $x^T y$ unless the data has a constant norm.

Thus, despite the fact that collision probability is monotonic in **d**, and thus, an LSH for L2 distances, unless the given data has a constant norm, $h^{L2}_{a,b}$ is not suitable for MIPS.

Why LSH cannot solve MIPS

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

Proof :

- Suppose there exists such hash function **h**. For un-normalized inner products the self similarity of a point **x** with itself is $Sim(x, x) = x^T x = \|x\|_2^2$
- Now, there may exist another points **y**, such that :
- $$Sim(x, y) = y^T x > \|x\|_2^2 + C, \text{ for any constant } C.$$
- Under any single randomized hash function h, the collision probability of the event **{h(x) = h(x)}** is always 1. So if h is an LSH for inner product then the event {h(x) = h(y)} should have higher probability compared to the event {h(x) = h(x)}
- This is because we can always choose some **y** with

$$Sim(x, y) = S_0 + \delta > S_0 \text{ and } cS_0 > Sim(x, x) \; \forall S_0 \text{ and } c < 1.$$

- This is definitely not possible because the probability can’t be greater than 1.
- Basic conclusion derived that self similarity is not the highest similarity.
- Therefore, we can’t have Locality sensitive hashing for inner products.
- We need some other method.

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 ≠ Q.

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

- All we need is **Pr (Q(q) = P(x))** to be monotonic in $q^T x$.
- Symmetry in hashing is unnecessary part of LSH definition.

Now we need to construct P and Q.

Construction :

Known : $Pr(h(q) = h(x)) = f(||\mathbf{q}||_2^2 + ||\mathbf{x}||_2^2 - 2\,\mathbf{q}^T\mathbf{x})$ (L2 LSH)

Idea : Construct P and Q such that $||\mathbf{Q(q)}||_2^2 + ||\mathbf{P(x)}||_2^2 - 2\,\mathbf{Q(q)}^T\mathbf{P(x)}$ is monotonic in $\mathbf{q}^T\mathbf{x}$.

Pre-Processing : Scale data x, such that $||\mathbf{x}_i|| < 1 \;\forall\, \mathbf{x}_i \in \mathbf{C}$

$$P(xi) - [\mathbf{x}_i;\ ||\mathbf{x}_i||_2^2;\ ||\mathbf{x}_i||_2^4;\ \ldots\ldots;\ ||\mathbf{x}_i||_2^{2\hat{}m}]$$

Querying : Q(q) = [q; ½; ½; …… ; ½]

Construction (cond) :

$$||\mathbf{Q(q)}||_2^{\,2} - ||\mathbf{P(x)}||_2^{\,2} = (||q||^2 + m/4) - 2\,\mathbf{q}^T\,\mathbf{x_i} + ||\mathbf{x_i}||_2^{\,2\hat{}m+1}$$

$||\mathbf{x_i}||_2^{\,2\hat{}m+1} \longrightarrow 0$, and m is constant. Hence we will have,

$$\mathbf{arg\ max\ q^Tx \simeq arg\ min\ ||Q(p) - P(x)||_2}$$

The Final Algorithm

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

- Scale $x \in C$ to have norm ≤ 0.83
- Append $||\mathbf{x}_i||^2$, $||\mathbf{x}_i||^4$, and $||\mathbf{x}_i||^8$ to vector $\mathbf{x}_i$.
- Use standard L2 hash to create hash tables.

Querying:

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

That's all!

Conclusion

- MIPS finds data vectors from the repository which are most similar to the query in terms of (un-normalized) inner product.

- Paper develops ALSH, which generalizes the existing LSH framework by applying (appropriately chosen) asymmetric transformations to the input query vector and the data vectors in the repository.

- ALSH implemented using a novel transformation which converts the original inner products into L2 distances in the transformed space.

- Theoretical proof and empirical demonstration for showing that this implementation of ALSH provides a provably efficient solution to MIPS.

Related Works (Improved ALSH for MIPS)

- The quantizations used in L2-LSH is suboptimal for MIPS compared to SRP (signed random projections hashing scheme for cosine similarity or correlations).

- It provides different asymmetric transformations which convert the problem of approximate MIPS into the problem amenable to SRP instead of L2-LSH.

- Advantage: Can obtain LSH type space partitioning which is not possible with the existing scheme.

Related Works (Improved ALSH for MIPS)

- Theoretical analysis shows that the new scheme is significantly better than the original scheme for MIPS.

- The new transformations with Sign-ALSH can be adapted to generate LSH like random data partitions which is very useful for large scale clustering. Such an adaptation is not possible with existing L2-ALSH.

- Paper also establishes experimentally that hashing based algorithms are superior to tree based space partitioning methods for MIPS.

Related works 2(On Symmetric and Asymmetric LSHs for Inner Product Search)

- Shows a simple symmetric LSH that enjoys stronger guarantees and better empirical performance than the ALSH our paper suggests.

- Also shows a variant of the settings where asymmetry is in-fact needed, but there a different asymmetric LSH is required.

- Shows that normalized queries and bounded database vectors are not advantageous to asymmetry.

Related works 2 (On Symmetric and Asymmetric LSHs for Inner Product Search)

- Paper provides a different asymmetric hash for a setting where asymmetric hash is not needed, the hashes suggested in our paper are not ALSH.

- In the MIPS setting when queries are normalized , the asymmetric hashes suggested in our paper are not universal and require tuning parameters specific to S, c, in contrast to SIMPLE-LSH which is symmetric, parameter-free and universal.

- To use a symmetric hash, one must normalize the queries but not the database vectors, which can legitimately be viewed as an asymmetric operation which is part of the hash (though then the hash would not be, strictly speaking, an ALSH).