Similarity Search and Locality Sensitive Hashing using TCAMs

Ashish Goel, Rajendra Shinde, Pankaj Gupta

Nearest Neighbor Search (NNS):

• Given a data set S of n points in R^d and a query point q, report p in S which is nearest to q.

• Applications: Web search, Information retrieval etc.

• Exact NNS suffers from the "curse of dimensionality" .

• Space\time required is exponential in d.

c-Approximate Nearest Neighbor Search (c-ANNS):

• Given a data set S of n points in R^d and a query point q, if p is nearest neighbor of q, report p' in S s.t. ||p'-q|| is at most c||p-q||.

• The (l,c)-Near Neighbor problem or (l,c)-NN problem:

• Given a data set S and a query point q, if there exists a point p in S, ||p-q|| ≤ l, report a point p' in S, ||p'-q|| ≤ cl.

• The (l,c)-NN problem is the decision version of the c-ANNS problem.

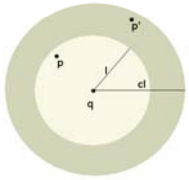

Fig. 1 (l,c)-NN problem.

• Scaling coordinates implies it suffices to solve the (1,c)-NN problem.

• The decision version adds logarithmic overhead in space and time complexity.

Solving the (1,c)-NN problem:

• **Dimension Reduction**:
For e.g. Use of Johnson Lindenstrauss lemma to reduce no. of dimensions to O(log n).

• **Locality Sensitive Hashing (LSH)**: Construct locality sensitive hash functions g: $R^d \longrightarrow \{0,1\}$ such that
if ||p-q|| ≤ l, **P**(g(p)=g(q)) ≥ p_1 and
if ||p-q|| ≥ cl, **P**(g(p)=g(q)) ≤ p_2.

• Space\time tradeoff: Typical LSH based single probe approach requires space $O(dn^{(1+\rho)}\log(n))$ and time $O(dn^{\rho})$ where $\rho = \log(p_1)/\log(p_2)$.

Comparison of existing techniques:

Approach	Space Requirement	Query time
Dimensionality reduction (Kushilevitz et al. , STOC '98, [12])	$d^2n^{O(1/(c_1\ 1)^2)}$ i.e. polynomial in n	$O(d\log d + (c_1\ 1)^{-3}\log^2 n)$
LSH single probe (Andoni et al. , FOCS' 06, [7])	$O(dn^{1+1/c^2}\log n)$ i.e. sub quadratic in n	$O(dn^{1/c^2})$
LSH near linear space (Panigrahy, SODA' 06, [10])	O(n)	$dn^{O(1/c^2)}$
TCAM based on TLSH (our paper)	O(n)	single TCAM lookup, single distance computation

Table 1: Comparison based on Space Requirement and Query Time

TCAMs:

• Capable of storing {0,1,*} in each data bit.

• * denotes a wildcard, which matches both 0 and 1.

• A TCAM takes a search key as query, returns the address of the entry that contains the key.

• TCAMS are Fast Associative Memories with O(1) memory lookup typically used for networking applications.

• They have been recently optimized for power consumption.

• Propose to use TCAMs to solve NNS problem with a query time of O(1) and near linear space requirement.

• Analogous to LSH we define a TLSH family.

Ternary LSH (TLSH):

•Construct ternary locality sensitive hash functions g: $R^d \longrightarrow \{0,1\}$ such that
if ||p-q|| ≤ l, **P**(g(p)=$_T$g(q)) ≥ p_1 and
if ||p-q|| ≥ cl, **P**(g(p)=$_T$g(q)) ≤ p_2.

where =$_T$ denotes the TCAM match operation.

• **Construction** of a ternary hash family G_δ :
Let g be a ternary hash function in this family,
• **a** : random vector in R^d, drawn from $N^d(0,I)$
• b : chosen uniformly from (0,2δ) for some constant δ. Consider a set of parallel hyper planes randomly translated, orthonormal to **a,** and adjacent hyper planes separated by a distance δ, thus partitioning R^d. Then g hashes alternate regions to * and remaining regions to {0,1}.

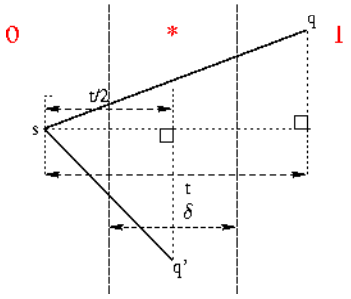

Fig. 2 A TLSH hash function.

TLSH Theorem:

G_δ is a $(1,c,p_1(\delta),p_2(\delta/c))$-TLSH family where
$p1(z) = 1 - \exp(-z^2/2)/(z^3)$, and
$p2(z) = 1 - \exp(-z^2/2)/(5\ z^3)$.

Comparison with LSH:

• Lower Bound: Motwani *et. al.* showed that
$\rho \geq 0.45/c\text{^}2$.

• On the other hand, for a TLSH family we have
$\rho \approx \exp(-\delta^2(c^2-1)/c^2)/c^3$,
which decreases to 0 as δ increases .

Applications to Similarity Search and Nearest Neighbors:

• **The (1,c)-NN problem:**

• **Algorithm**:

• Parameters (w,δ) chosen as a function of n and error probability ε and separation c.

1. **Preprocessing**: Choose w hash functions from a TLSH family and store the images of each point in S in the TCAM of width w.
2. **Runtime**: Given a query q, find its TCAM representation T(q) using the same hash functions and perform a TCAM lookup of T(q). If the point returned p' is at a distance of at most c from q, report "Yes" and that point as output, otherwise report "No".

• In order to decrease the width of the TCAM required, it is possible to choose (w,δ) so that algorithm succeeds with a constant probability say(1/2) and repeating the algorithm $O(\log(1/\varepsilon))$ times, it can be made to succeed with high probability.

Analysis:

• Analysis in terms of false negatives and false positives.
• False negative probability is $1-p_1^w$.
• False positive probability w.r.t. a specific point is p_2^w.
• Increasing δ reduces the probability of false negatives.
• Increasing w increases the chance of false positives.
• Choosing $\delta = O(\sqrt{(\log \log n)})$ and width $w = O(\log n^{(c\text{^}2/(c\text{^}2-1))}(\log \log n^{1.5}))$ ensures that the false negative probability and expected no. of false positives can be kept small.

Granularity:

• If the width is restricted to O(log n), the TCAM can solve the (1,c)-NN problem when $c = O(\sqrt{(\log \log n)})$.
• If LSH family is used along with a RAM of width O(log n), the (1,c)-NN problem can be solved only when $c = \Omega(\sqrt{(\log n)})$.
• This implies an exponential improvement in the granularity by using TLSH method.

Experiment:

• TCAMs are commercially available in sizes of 72, 144, 288 bits.
• Simulating a TCAM of 288 bits, experiments were carried out on Wikipedia data (a snapshot of Wikipedia) obtained from Yahoo! which contained a million data points (web pages).
• A small percentage of false negatives (less than 1%) were observed and no false positives were observed.