Locality Sensitive Hashing Scheme Based on p-Stable Distributions

Mayur Datar (Stanford)

Nicole Immorlica (MIT)

Piotr Indyk (MIT)

Vahab Mirrokni (MIT)

(Streaming) Massive Data Sets $\Rightarrow$ High Dimensional Vectors

- Massive data sets visualized as high dimensional vectors

- E.g. Number of IP-packets sent to address i from IP address j

$$\boldsymbol{v^j} = \{v_1^j, v_2^j, \ldots, v_i^j, \ldots, v_N^j\}$$

  Dimensionality $= 2^{32}$

- E.g. Number of phone calls made from telephone number j to telephone number k

$$\boldsymbol{v^j} = \{v_1^j, v_2^j, \ldots, v_k^j, \ldots, v_{N'}^j\}$$

  Dimensionality $= 10^9$

Update Model

- Vectors constantly updated as per *cash register model*

- Update element (i, a) for vector $\boldsymbol{v}$ changes it as follows:

$$\boldsymbol{v} = \{v_1, v_2, \ldots, (v_i + a), \ldots, v_N\}$$

- Numerous high dimensional vectors
  E.g. one vector per (millions) telephone customers,
  one vector per (millions) IP-address etc.
  Rows of a huge matrix

l_p Norms

- $l_p(\boldsymbol{v}) = (\sum_{i=1}^{N} |v_i|^p)^{1/p}$
  E.g. l_1 norm (Manhattan), l_2 norm (Euclidean)

- l_p norms usually computed over vector differences
  E.g. $l_1(\boldsymbol{v^j} - \boldsymbol{v^k})$, $l_2(\boldsymbol{v^j} - \boldsymbol{v^k})$, $l_{0.005}(\boldsymbol{v^j} - \boldsymbol{v^k})$ etc.

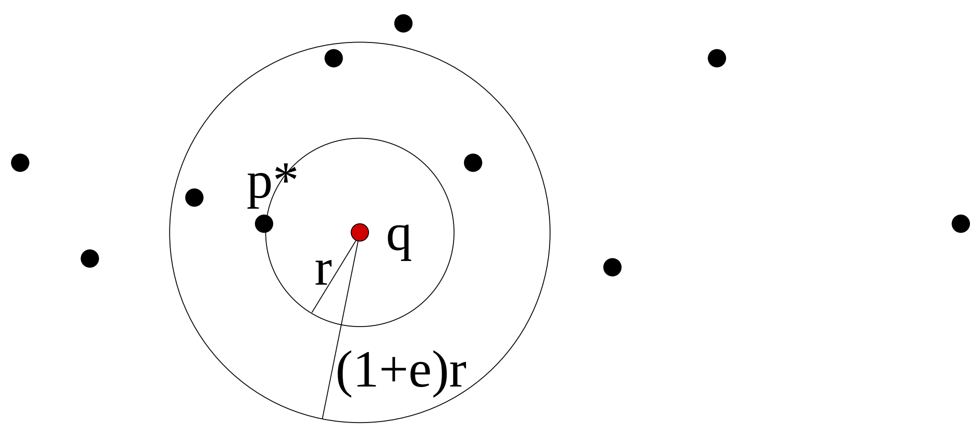
- What do l_p norms capture?
  - l_1 norm applied to telephone vectors: symmetric (multi) set difference between two customers
  - l_p norms for small values of p (0.005): capture Hamming norms, distinct values [CDIM'02]

Proximity Queries

- **Nearest Neighbor**: Given a query $\boldsymbol{q}$ find the closest (smallest l_p norm) point $\boldsymbol{p}$
- **Near Neighbor**: Given a query $\boldsymbol{q}$ and distance R find all (or most) points $\boldsymbol{p}$ s.t. $l_p(\boldsymbol{p} - \boldsymbol{q}) \leq R$
- Applications: Classification, fraud detection etc.
  E.g. find cell phone customers whose calling pattern is similar to that of XYZ (UBL)

Approximate Nearest Neighbor

- Curse of dimensionality
- Error parameter ϵ: Find any point that is within $(1+\epsilon)$ times the distance from true nearest neighbor



Approximate Near Neighbor ((R, ϵ)–PLEB)

- $B(\boldsymbol{c}, R)$ denotes a *ball* of radius R centered at $\boldsymbol{c}$
- Given: radius R, error parameter ϵ and query point $\boldsymbol{q}$:
  - if there exists data point $\boldsymbol{p}$ s.t. $\boldsymbol{q} \in B(\boldsymbol{p}, R)$, return YES and a point (or all points) p' s.t. $\boldsymbol{q} \in B(\boldsymbol{p'}, (1+\epsilon)R)$,
  - if $\boldsymbol{q} \notin B(\boldsymbol{p}, R)$ for all data points p, return NO,
  - if closest data point to q is at distance between R and $R(1+\epsilon)$ then return YES or NO

Approximate Near Neighbor

- Useful problem formulation in itself
- Approximate nearest neighbor can be reduced to approximate near neighbor (binary search on R)
- Henceforth, we will concentrate on solving approximate near neighbor

Our contribution

- Data structure for the approximate near neighbor problem ((R, ϵ)–PLEB)
- Small query time, **update time** and easy to implement
- works for l_p norms, for $0 < p \leq 2$. In particular $0 < p < 1$
- Earlier result ([IM'98]) worked for l_1, l_2 and Hamming norm.
- Our technique improves the query time for l_2 norm

Locality Sensitive Hashing (LSH)([IM'98])

- Intuition: if two points are close (less than dist r_1) they hash to same bucket with prob at least p_1. Else, if they are far (more than dist $r_2 > r_1$) they hash to same bucket with prob no more than $p_2 < p_1$

- Formally: A family $\mathcal{H} = \{h : S \rightarrow U\}$ is called (r_1, r_2, p_1, p_2)-*sensitive* for distance function D if for any $\boldsymbol{v}, \boldsymbol{q} \in S$
  - if $\boldsymbol{v} \in B(\boldsymbol{q}, r_1)$ then $\Pr_{\mathcal{H}}[h(\boldsymbol{q}) = h(\boldsymbol{v})] \geq p_1$,
  - if $\boldsymbol{v} \notin B(\boldsymbol{q}, r_2)$ then $\Pr_{\mathcal{H}}[h(\boldsymbol{q}) = h(\boldsymbol{v})] \leq p_2$.
  - $r_1 < r_2$, $p_1 > p_2$

Using LSH to solve (R, ϵ)–PLEB ([IM'98])

- Let $c = 1 + \epsilon$

Theorem. *Suppose there is a* (R, cR, p_1, p_2)*-sensitive family* $\mathcal{H}$ *for a distance measure* D*. Then there exists an algorithm for* (R, c)*-PLEB under measure* D *which uses* $O(dn + n^{1+\rho})$ *space, with query time dominated by* $O(n^\rho)$ *distance computations, and* $O(n^\rho \log_{1/p_2} n)$ *evaluations of hash functions from* $\mathcal{H}$*, where* $\rho = \frac{\ln 1/p_1}{\ln 1/p_2}$

- Bottom-line: Design LSH scheme with small ρ for l_p norms

Recap

- Proximity problems reduced to designing LSH schemes
- Design LSH schemes for l_p norms with small ρ, update time etc.
- A family $\mathcal{H} = \{h : S \rightarrow U\}$ is called (r_1, r_2, p_1, p_2)-*sensitive* for distance function D if for any $\boldsymbol{v}, \boldsymbol{q} \in S$
  - if $\boldsymbol{v} \in B(\boldsymbol{q}, r_1)$ then $\Pr_{\mathcal{H}}[h(\boldsymbol{q}) = h(\boldsymbol{v})] \geq p_1$,
  - if $\boldsymbol{v} \notin B(\boldsymbol{q}, r_2)$ then $\Pr_{\mathcal{H}}[h(\boldsymbol{q}) = h(\boldsymbol{v})] \leq p_2$
- $r_1 = R = 1$, $r_2 = R(1 + \epsilon) = 1 + \epsilon = c$

p–Stable distributions

- p**–stable distribution** $(p \geq 0)$: A distribution $\mathcal{D}$ over $\Re$ s.t
  - n real numbers $v_1 \ldots v_n$,
  - i.i.d. variables $X_1 \ldots X_n$ with distribution $\mathcal{D}$,
  - r.v. $\sum_i v_i X_i$ has the same distribution as the variable $(\sum_i |v_i|^p)^{1/p} X = l_p(\boldsymbol{v})X$, where X is a r.v. with distribution $\mathcal{D}$

- E.g. p–Stable distr for $p = 1$ is Cauchy distr, for $p = 2$ is Gaussian distr

- for $0 < p < 2$ there is a way to sample from a p–stable distribution given two uniform r.v.'s over $[0, 1]$ [Nol]

How are p–Stable distributions useful?

- Consider a vector $\boldsymbol{X} = \{X_1, X_2, \ldots, X_N\}$, where each X_i is drawn from a p–Stable distr

- For any pair of vectors $\boldsymbol{a}, \boldsymbol{b}$ $\boldsymbol{a} \cdot \boldsymbol{X} - \boldsymbol{b} \cdot \boldsymbol{X} = (\boldsymbol{a} - \boldsymbol{b}) \cdot \boldsymbol{X}$ (by linearity)

- Thus $\boldsymbol{a} \cdot \boldsymbol{X} - \boldsymbol{b} \cdot \boldsymbol{X}$ is distributed as $(l_p(\boldsymbol{a} - \boldsymbol{b}))X'$ where X' is a p–Stable distr r.v.

- Using multiple independent $\boldsymbol{X}$'s we can use $\boldsymbol{a} \cdot \boldsymbol{X} - \boldsymbol{b} \cdot \boldsymbol{X}$ to estimate $l_p(a - b)$ [Ind'01]

How are p–Stable distributions useful?

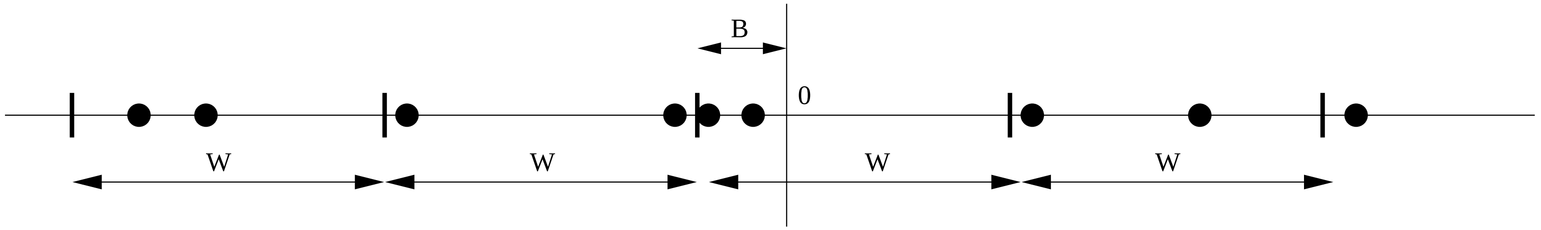
- For a vector $\boldsymbol{a}$, the dot product $\boldsymbol{a} \cdot \boldsymbol{X}$ projects it onto the real line

- For any pair of vectors $\boldsymbol{a}, \boldsymbol{b}$ these projections are "close" (w.h.p.) if $l_p(\boldsymbol{a} - \boldsymbol{b})$ is "small" and "far" otherwise

- Divide the real line into segments of width w

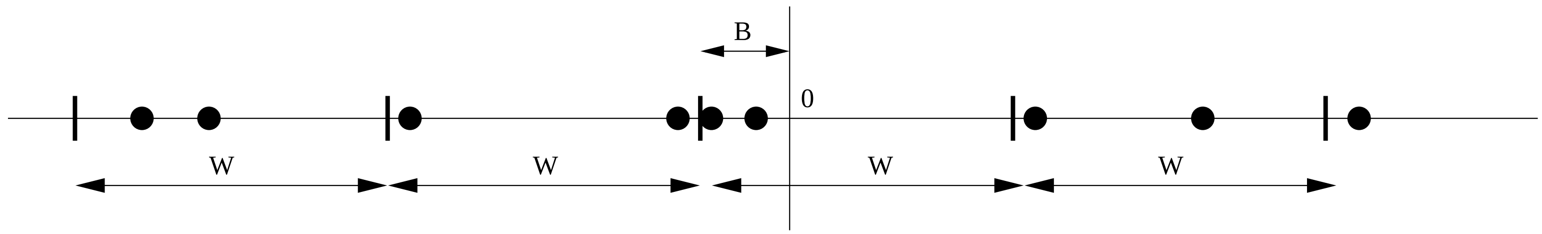
- Each segment defines a hash bucket, i.e. vectors that project onto the same segment belong to the same bucket

Hashing (formal) definition



- Consider $h_{\boldsymbol{a},b} \in \mathcal{H}^w$, $h_{\boldsymbol{a},b}(\boldsymbol{v}) : \mathcal{R}^d \rightarrow \mathcal{N}$
- $\boldsymbol{a}$ is a d dimensional random vector whose each entry is drawn from a p-stable distr
- b is a random real number chosen uniformly from $[0, w]$ (random shift)
- $h_{\boldsymbol{a},b}(\boldsymbol{v}) = \lfloor \frac{\boldsymbol{a} \cdot \boldsymbol{v} + b}{w} \rfloor$

Collision probabilities



- Consider two vectors $\boldsymbol{v_1}, \boldsymbol{v_2}$ and let $\ell = l_p(\boldsymbol{v_1}, \boldsymbol{v_2})$

- Let Y denote the distance between their projections onto the random vector $\boldsymbol{a}$ (Y is distributed as ℓX where X is a p-stable distr r.v.)

- if $Y > w$, $\boldsymbol{v_1}, \boldsymbol{v_2}$ will not collide

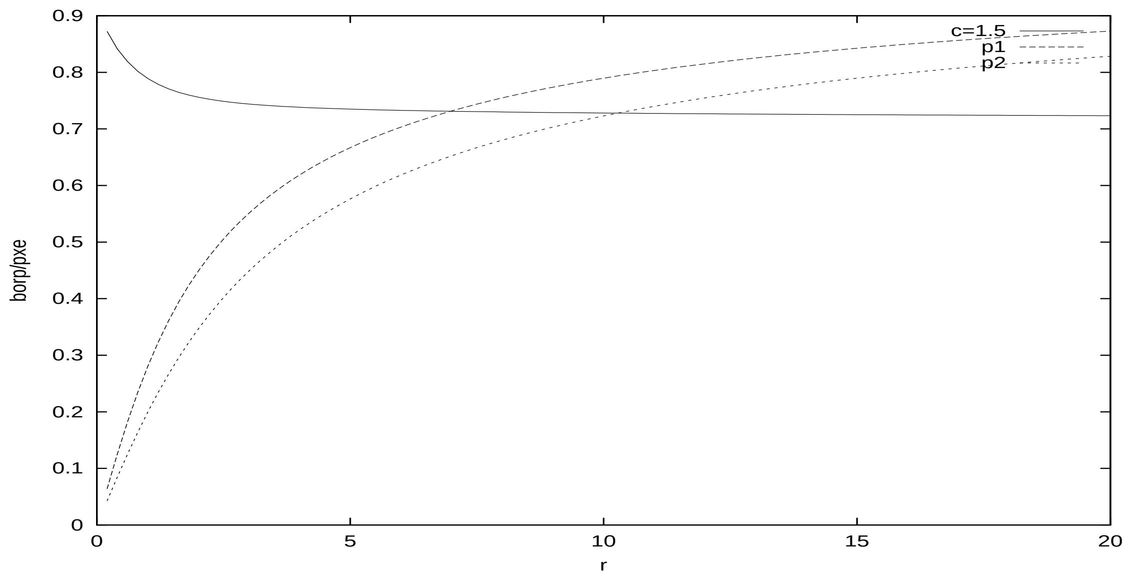
- if $Y \leq w$, $\boldsymbol{v_1}, \boldsymbol{v_2}$ will collide with probability equal to $(1 - (Y/w))$ (random shift b)

Collision probabilities

- $f_p(t)$: p.d.f. of the absolute value of a p-stable distribution
- $\ell = l_p(\boldsymbol{v_1}, \boldsymbol{v_2})$
- $\ell \leq 1$, $p_1 = Pr[h_{\boldsymbol{a},b}(\boldsymbol{v_1}) = h_{\boldsymbol{a},b}(\boldsymbol{v_2})] \geq \int_0^w f_p(t)(1 - \frac{t}{w})dt$
- $\ell > 1 + \epsilon = c$, $p_2 = Pr[h_{\boldsymbol{a},b}(\boldsymbol{v_1}) = h_{\boldsymbol{a},b}(\boldsymbol{v_2})] \leq \int_0^w \frac{1}{c} f_p(\frac{t}{c})(1 - \frac{t}{w})dt$
- $\mathcal{H}^w$ hash family is (r_1, r_2, p_1, p_2)-sensitive for $r_1 = 1$, $r_2 = c$ and p_1, p_2 given as above

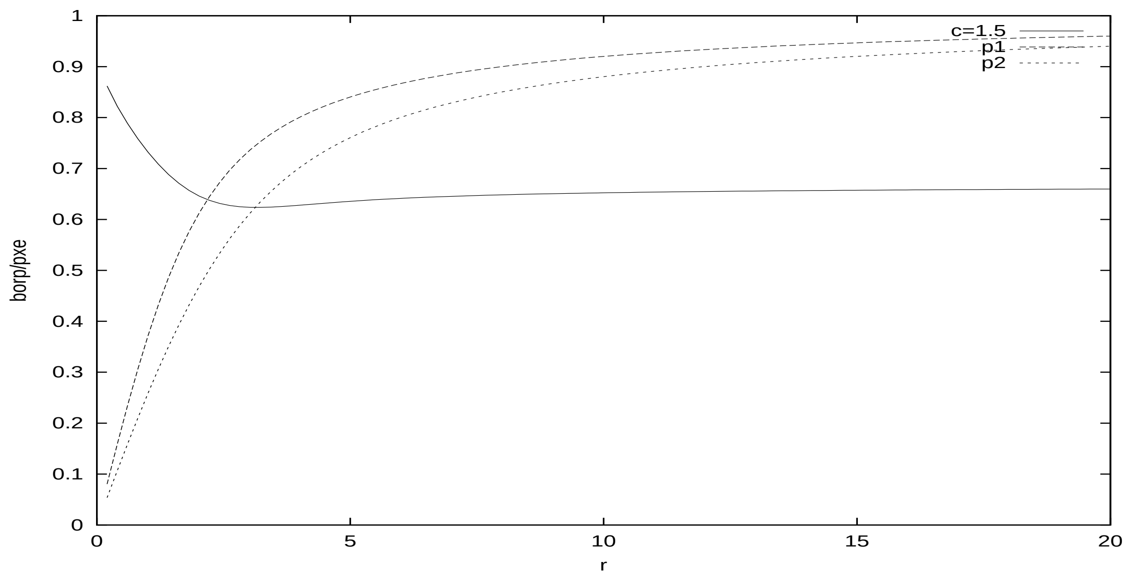
Special cases

- $p = 1$(Cauchy distr): $f_p(t) = \frac{2}{\pi}\frac{1}{1+t^2}$
- $p_2 = 2\frac{tan^{-1}(w/c)}{\pi} - \frac{1}{\pi(w/c)}\ln(1 + (w/c)^2)$
- p_1 obtained by substituting $c = 1$



Special cases

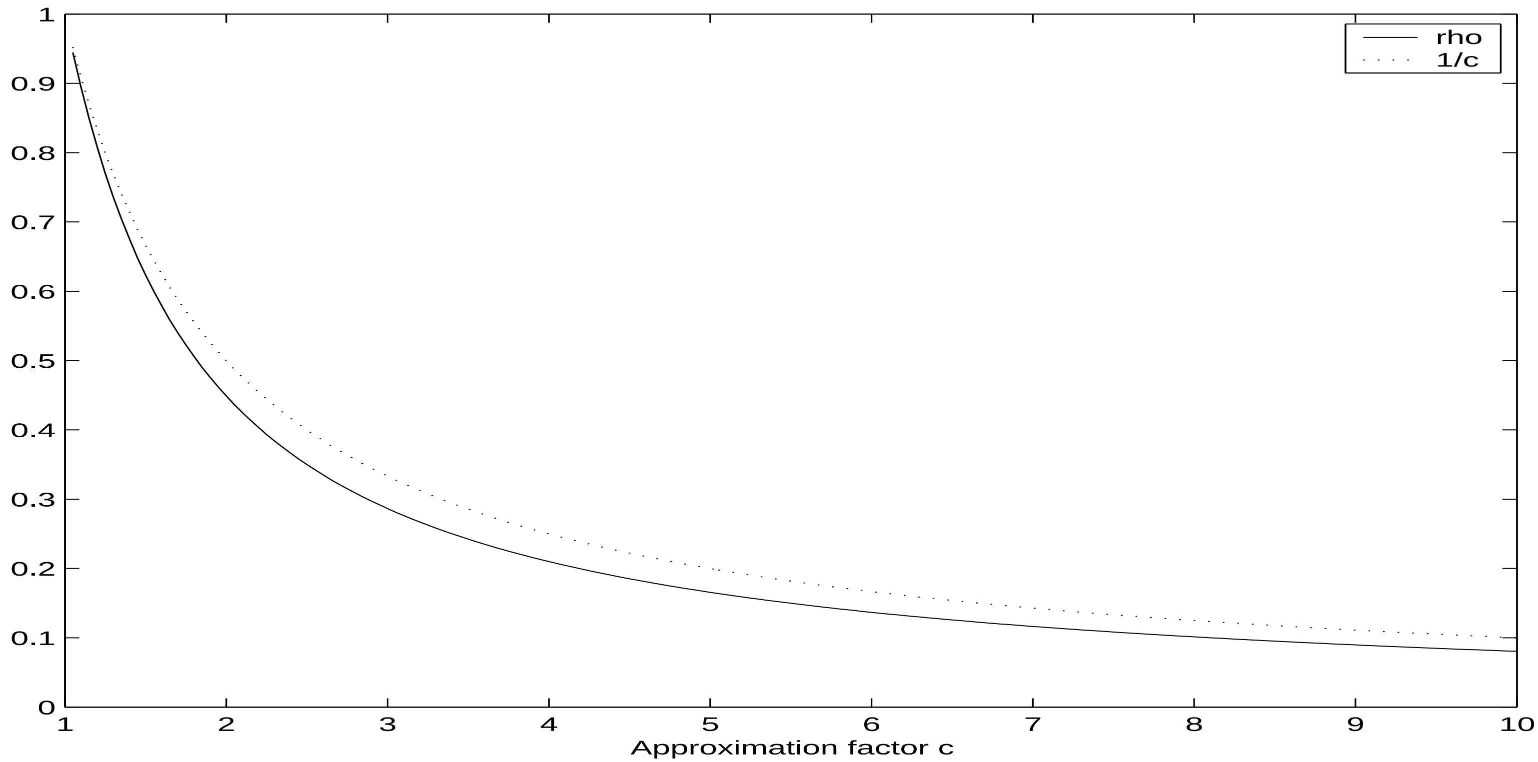
- $p = 2$(Gaussian distr): $f_p(t) = \frac{2}{\sqrt{2\pi}} e^{-t^2/2}$
- $p_2 = 1 - 2norm(-w/c) - \frac{2}{\sqrt{2\pi}w/c}(1 - e^{-(w^2/2c^2)})$
- p_1 obtained by substituting $c = 1$



Comparison with previous scheme

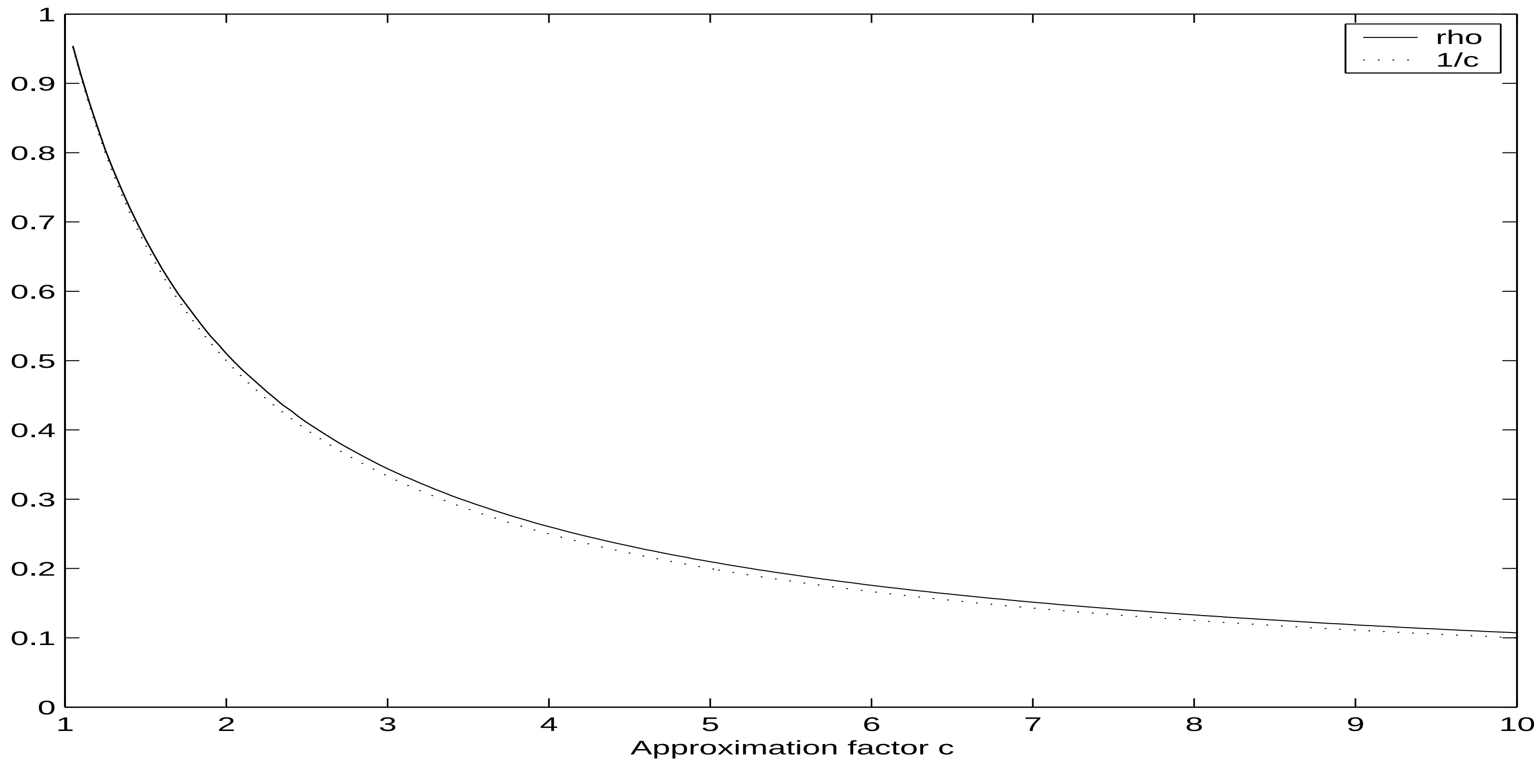
- Previous hashing scheme for $p = 1, 2$ achieved $\rho = 1/c$
- Based on reduction to hamming distance
- New scheme achieves smaller ρ (than $1/c$) for $p = 2$
- Large constants and $\log$ factors for $p = 2$ in query time besides n^ρ
- Achieves $\rho = 1/c$ for $p = 1$

ρ for $p = 2$



rho
1/c
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9
10
Approximation factor c

ρ **for** $p = 1$



rho
1/c
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
1
2
3
4
5
6
7
8
9
10
Approximation factor c

General case

- what about general case, i.e. $p \neq 1, 2$?

Theorem. *For any $p \in (0, 2]$ there is a (r_1, r_2, p_1, p_2)-sensitive family $\mathcal{H}^w$ for l_p^d such that for any $\gamma > 0$,*

$$\rho = \frac{\ln 1/p_1}{\ln 1/p_2} \leq (1 + \gamma) \cdot \max\left(\frac{1}{c^p}, \frac{1}{c}\right).$$

- Achieves $\frac{1}{c^p}$ for $p < 1$

Conclusions

- New LSH scheme for $0 < p \leq 2$. First one for $0 < p < 1$
- Easy to implement (experiments in progress)
- Easy to update hash value in cash register model
- Improves running time for $p = 2$ over previous scheme