# Approximate Proximity Problems in High Dimensions via Locality-Sensitive Hashing

Piotr Indyk

Helsinki, May 2007

# Recap

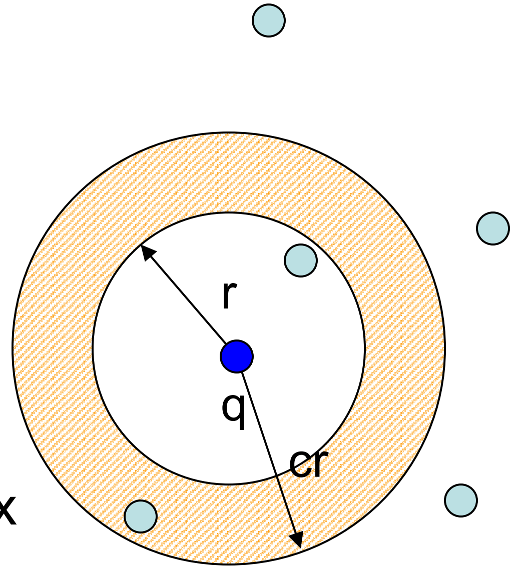
- Recap:
- Nearest Neighbor in $R^d$
  - Motivation: learning, retrieval, compression,..
- Exact: curse of dimensionality
  - Either $O(dn)$ query time, or $n^{O(d)}$ space
- Approximate (factor $c=1+\varepsilon$)
  - Kd-trees: optimal space, $O(1/\varepsilon)^d \log n$ query time

# Today

- Algorithms with polynomial dependence on d
  - Locality-Sensitive Hashing
- Experiments etc

# Approximate Near Neighbor

- c-Approximate r-Near Neighbor: build data structure which, for any query q:
  - If there is a point $p \in P$, $||p-q|| \leq r$
  - it returns $p' \in P$, $||p-q|| \leq cr$




- Reductions:
  - c-Approx r-Close Pair
  - c-Approx Nearest Neighbor reduces to c-Approx Near Neighbor

    (log overhead)
  - One can enumerate all approx near neighbors

    → can solve exact near neighbor problem
  - Other apps: c-approximate Minimum Spanning Tree, clustering, etc.

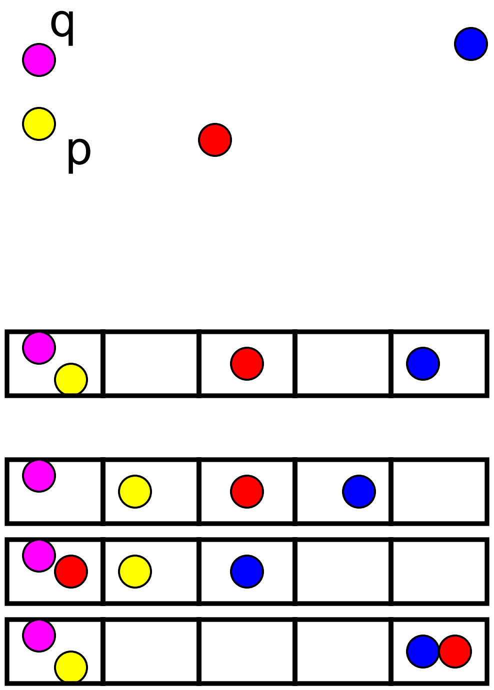
# Approximate algorithms

- Space/time exponential in d [Arya-Mount-et al], [Kleinberg'97], [Har-Peled'02], [Arya-Mount-…]
- Space/time polynomial in d [Kushilevitz-Ostrovsky-Rabani'98], [Indyk-Motwani'98], [Indyk'98], [Gionis-Indyk-Motwani'99], [Charikar'02], [Datar-Immorlica-Indyk-Mirrokni'04], [Chakrabarti-Regev'04], [Panigrahy'06], [Ailon-Chazelle'06]…

| Space | Time | Comment | Norm | Ref |
|---|---|---|---|---|
| $dn+n^{4/\varepsilon^2}$ | $d * \log n / \varepsilon^2$ or 1 | $c=1+\varepsilon$ | Hamm, $l_2$ | [KOR'98, IM'98] |
| $n^{\Omega(1/\varepsilon^2)}$ | O(1) | | | [AIP'06] |
| → $dn+n^{1+\rho(c)}$ | $dn^{\rho(c)}$ | $\rho(c)=1/c$ | Hamm, $l_2$ | [IM'98], [GIM'98],[Cha'02] |
| → | | $\rho(c)<1/c$ | $l_2$ | [DIIM'04] |
| $dn * \log s$ | $dn^{\sigma(c)}$ | $\sigma(c)=O(\log c/c)$ | Hamm, $l_2$ | [Ind'01] |
| → $dn+n^{1+\rho(c)}$ | $dn^{\rho(c)}$ | $\rho(c)=1/c^2 + o(1)$ | $l_2$ | [AI'06] |
| | | $\sigma(c)=O(1/c)$ | $l_2$ | [Pan'06] |

# Locality-Sensitive Hashing

- Idea: construct hash functions $g: R^d \rightarrow U$ such that for any points $p,q$:
  - If $||p-q|| \leq r$, then $Pr[g(p)=g(q)]$ is ~~“high”~~ “not-so-small”
  - If $||p-q|| > cr$, then $Pr[g(p)=g(q)]$ is “small”
- Then we can solve the problem by hashing



# LSH [Indyk-Motwani'98]

- A family $H$ of functions $h: R^d \rightarrow U$ is called $(P_1,P_2,r,cr)$-sensitive, if for any $p,q$:
  - if $||p-q|| < r$ then $Pr[\ h(p)=h(q)\ ] > P_1$
  - if $||p-q|| > cr$ then $Pr[\ h(p)=h(q)\ ] < P_2$
- Example: Hamming distance
  - LSH functions: $h(p)=p_i$, i.e., the $i$-th bit of $p$
  - Probabilities: $Pr[\ h(p)=h(q)\ ] = 1-D(p,q)/d$

p=10010010
q=11010110

# Algorithm

- We use functions of the form

$$g(p)=\langle h_1(p),h_2(p),\ldots,h_k(p)\rangle$$

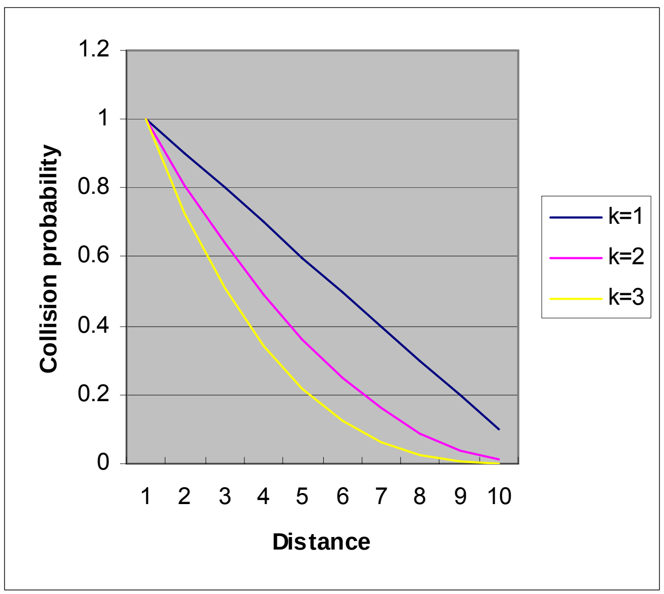
- Preprocessing:
  - Select $g_1 \ldots g_L$
  - For all $p \in P$, hash $p$ to buckets $g_1(p) \ldots g_L(p)$
- Query:
  - Retrieve the points from buckets $g_1(q)$, $g_2(q)$, … , until
    - Either the points from all $L$ buckets have been retrieved, or
    - Total number of points retrieved exceeds $3L$
  - Answer the query based on the retrieved points
  - Total time: $O(dL)$

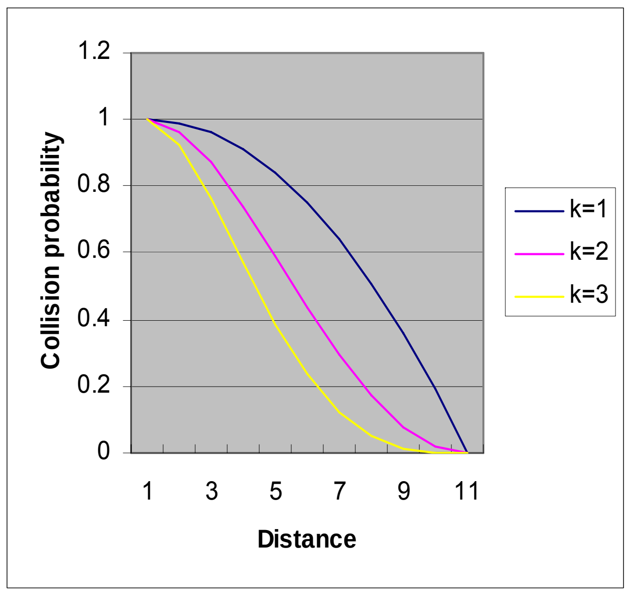
# Analysis [IM'98, Gionis-Indyk-Motwani'99]

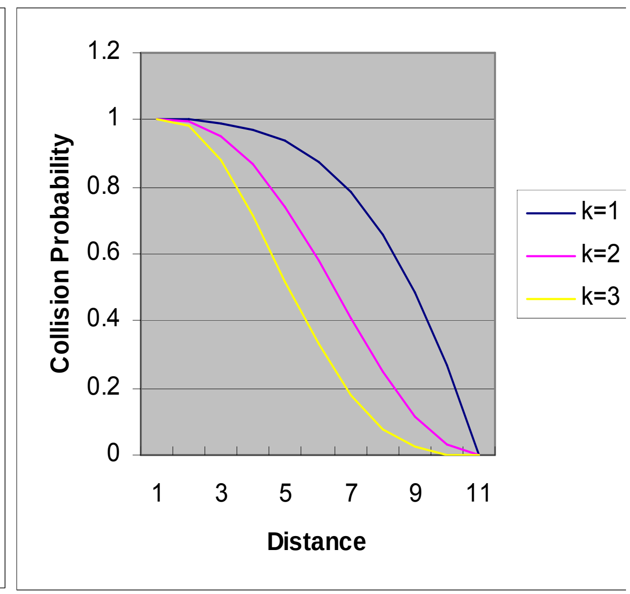
- Lemma1: the algorithm solves c-approximate NN with:
  - Number of hash fun: $L=n^{\rho}$, $\rho=\log(1/P1)/\log(1/P2)$
  - Constant success probability per query q
- Lemma 2: for Hamming LSH functions, we have $\rho=1/c$

# Proof of Lemma 1 by picture

- Points in $\{0,1\}^d$
- Collision prob. for k=1..3, L=1..3 (recall: L=#indices, k=#h's )
- Distance ranges from 0 to d=10



# Proof

- Define:
  - $p$: a point such that $||p-q|| \le r$
  - $FAR(q)=\{ p' \in P: ||p'-q|| > c\, r \}$
  - $B_i(q)=\{ p' \in P: g_i(p')=g_i(q) \}$
- Will show that both events occur with $>0$ probability:
  - $E_1$: $g_i(p)=g_i(q)$ for some $i=1 \ldots L$
  - $E_2$: $\Sigma_i |B_i(q) \cap FAR(q)| < 3L$

# Proof ctd.

- Set $k = \log_{1/P2} n$
- For $p' \in FAR(q)$ ,

  $$\Pr[g_i(p')=g_i(q)] \le P_2^k = 1/n$$
- $E[\ |B_i(q) \cap FAR(q)|\ ] \le 1$
- $E[\Sigma_i\ |B_i(q) \cap FAR(q)|\ ] \le L$
- $\Pr[\Sigma_i\ |B_i(q) \cap FAR(q)| \ge 3L\ ] \le 1/3$

# Proof, ctd.

- $\Pr[\, g_i(p)=g_i(q)\,] \geq 1/P_1^{\,k} = 1/n^{\rho} = 1/L$
- $\Pr[\, g_i(p) \neq g_i(q),\ i=1..L] \leq (1-1/L)^L \leq 1/e$

# Proof, end

- $\Pr[E_1$ not true$]+\Pr[E_2$ not true$]$
  $\leq 1/3+1/e = 0.7012$.
- $\Pr[E_1 \cap E_2] \geq 1-(1/3+1/e) \approx 0.3$

# Proof of Lemma 2

- Statement: for
  - $P1=1-r/d$
  - $P2=1-cr/d$

  we have $\rho=\log(P1)/\log(P2) \leq 1/c$
- Proof:
  - Need $P1^c \geq P2$
  - But $(1-x)^c \geq (1-cx)$ for any $1>x>0, c>1$

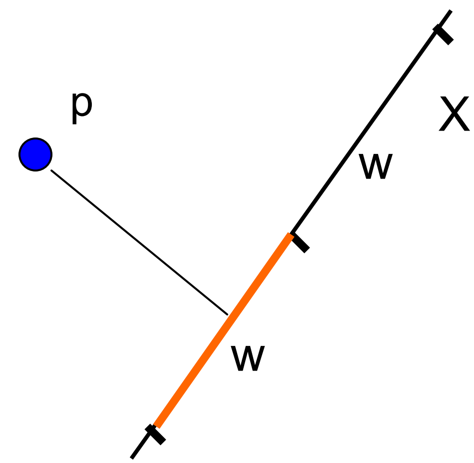
# Recap

- LSH solves c-approximate NN with:
  - Number of hash fun: $L=n^{\rho}$, $\rho=\log(1/P1)/\log(1/P2)$
  - For Hamming distance we have $\rho=1/c$
- Questions:
  - Can we extend this beyond Hamming distance ?
    - Yes:
      - embed $l_2$ into $l_1$ (random projections)
      - $l_1$ into Hamming (discretization)
  - Can we reduce the exponent $\rho$ ?

# Projection-based LSH

[Datar-Immorlica-Indyk-Mirrokni'04]

- Define $h_{X,b}(p)=\lfloor(p*X+b)/w\rfloor$:
  - $w \approx r$
  - $X=(X_1 \ldots X_d)$ , where $X_i$ is chosen from:
    - Gaussian distribution (for $l_2$ norm)*
  - $b$ is a scalar

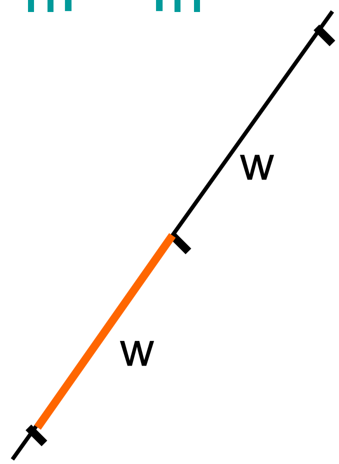



* For $l_s$ norm use "s-stable" distribution, where $p*X$ has same distribution as $||p||_s$ Z, where Z is s-stable

# Analysis

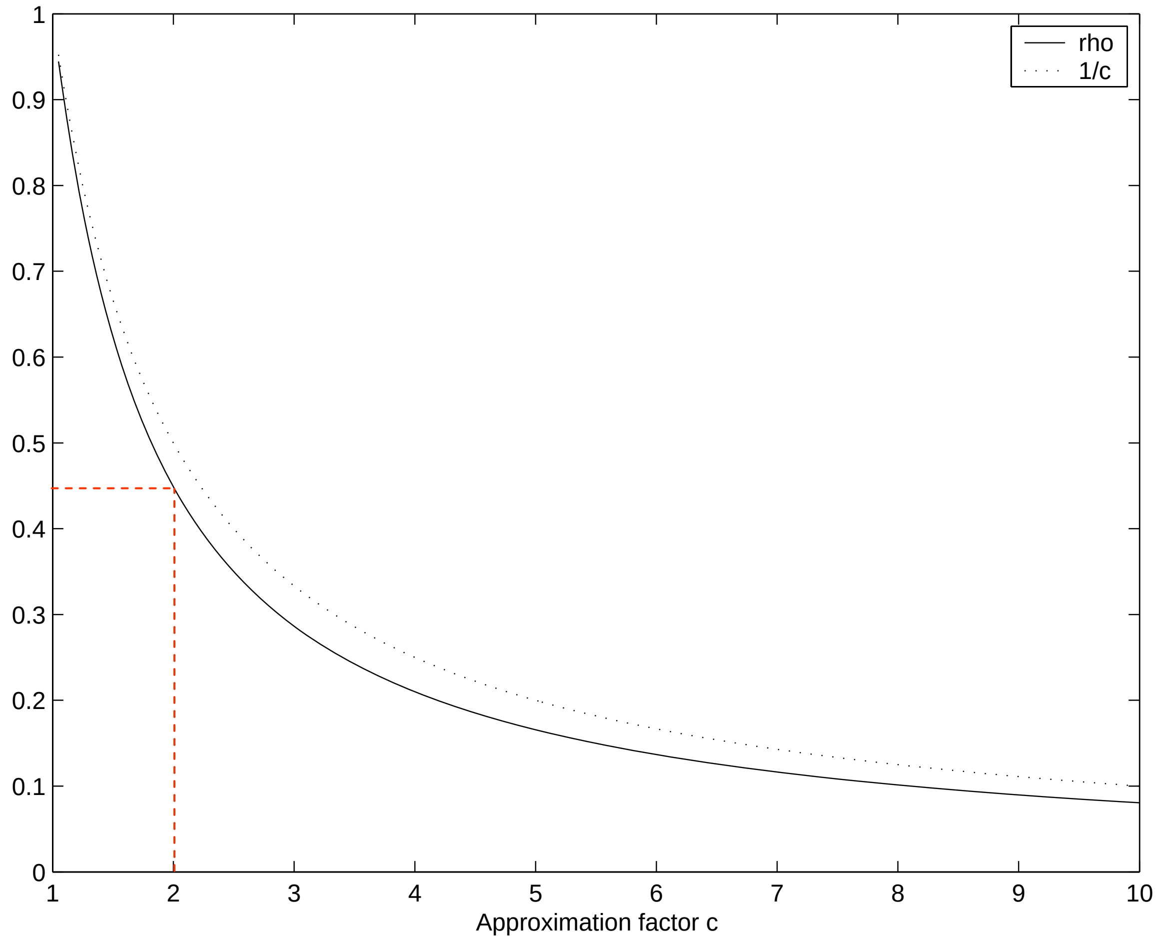
- Need to:
  - Compute Pr[h(p)=h(q)] as a function of ||p-q|| and w; this defines $P_1$ and $P_2$
  - For each c choose w that minimizes

$$\rho=\log_{1/P2}(1/P_1)$$




- Method:
  - For $l_2$: computational
  - For general $l_s$: analytic

# $\rho(c)$ for $l_2$

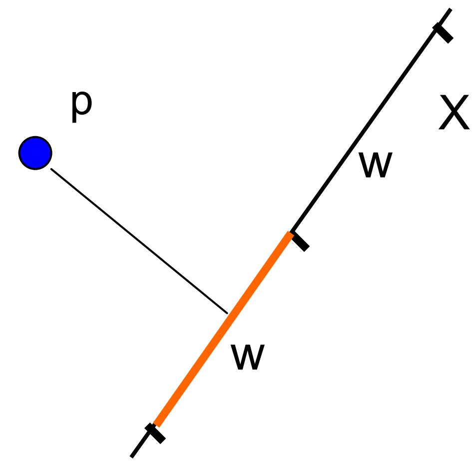


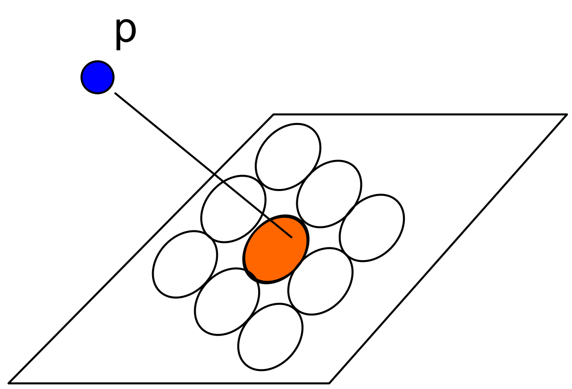

Helsinki, May 2007

# New LSH scheme

[Andoni-Indyk'06]

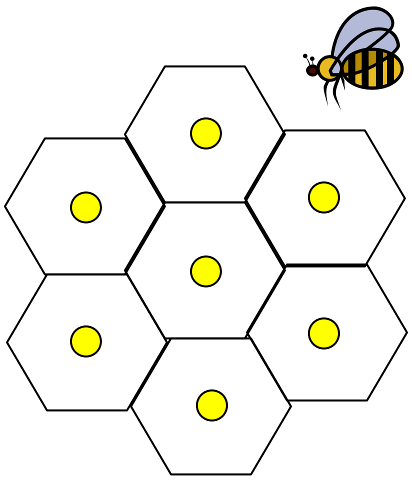
- Instead of projecting onto $R^1$, project onto $R^t$, for constant $t$
- Intervals → lattice of balls
  - Can hit empty space, so hash until a ball is hit
- Analysis:
  - $\rho = 1/c^2 + O(\log t / t^{1/2})$
  - Time to hash is $t^{O(t)}$
  - Total query time: $dn^{1/c^2+o(1)}$
- [Motwani-Naor-Panigrahy'06]: LSH in $l_2$ must have $\rho \geq 0.45/c^2$



# New LSH scheme, ctd.

- How does it work in practice ?
- The time $t^{O(t)}dn^{1/c^2+f(t)}$ is not very practical
  - Need $t \approx 30$ to see some improvement
- Idea: a different decomposition of $R^t$
  - Replace random balls by Voronoi diagram of a lattice
  - For specific lattices, finding a cell containing a point can be very fast →fast hashing



# Leech Lattice LSH

- Use Leech lattice in $R^{24}$ , $t=24$
  - Largest kissing number in 24D: 196560
  - Conjectured largest packing density in 24D
  - 24 is 42 in reverse…
- Very fast (bounded) decoder: about 519 operations [Amrani-Beery'94]
- Performance of that decoder for $c=2$:
  - $1/c^2$ 0.25
  - $1/c$ 0.50
  - Leech LSH, any dimension: $\rho \approx 0.36$
  - Leech LSH, 24D (no projection): $\rho \approx 0.26$

# LSH Zoo

- Hamming metric
- $L_s$ norm, $s \in (0,2]$
- Vector angle [Charikar'02] based on [GW'94]
- Jaccard coefficient [Broder et al'97]

$$J(A,B) = |A \cap B| / |A \cup B|$$

# Experiments

# Experiments (with ’04 version)

- $E^2$LSH: Exact Euclidean LSH (with Alex Andoni)
  - Near Neighbor
  - User sets r and P = probability of NOT reporting a point within distance r (=10%)
  - Program finds parameters k,L,w so that:
    - Probability of failure is at most P
    - Expected query time is minimized
- Nearest neighbor: set radius (radiae) to accommodate 90% queries (results for 98% are similar)
  - 1 radius: 90%
  - 2 radiae: 40%, 90%
  - 3 radiae: 40%, 65%, 90%
  - 4 radiae: 25%, 50%, 75%, 90%

# Data sets

- MNIST OCR data, normalized (LeCun)
  - d=784
  - n=60,000
- Corel_hist
  - d=64
  - n=20,000
- Corel_uci
  - d=64
  - n=68,040
- Aerial data (Manjunath)
  - d=60
  - n=275,476

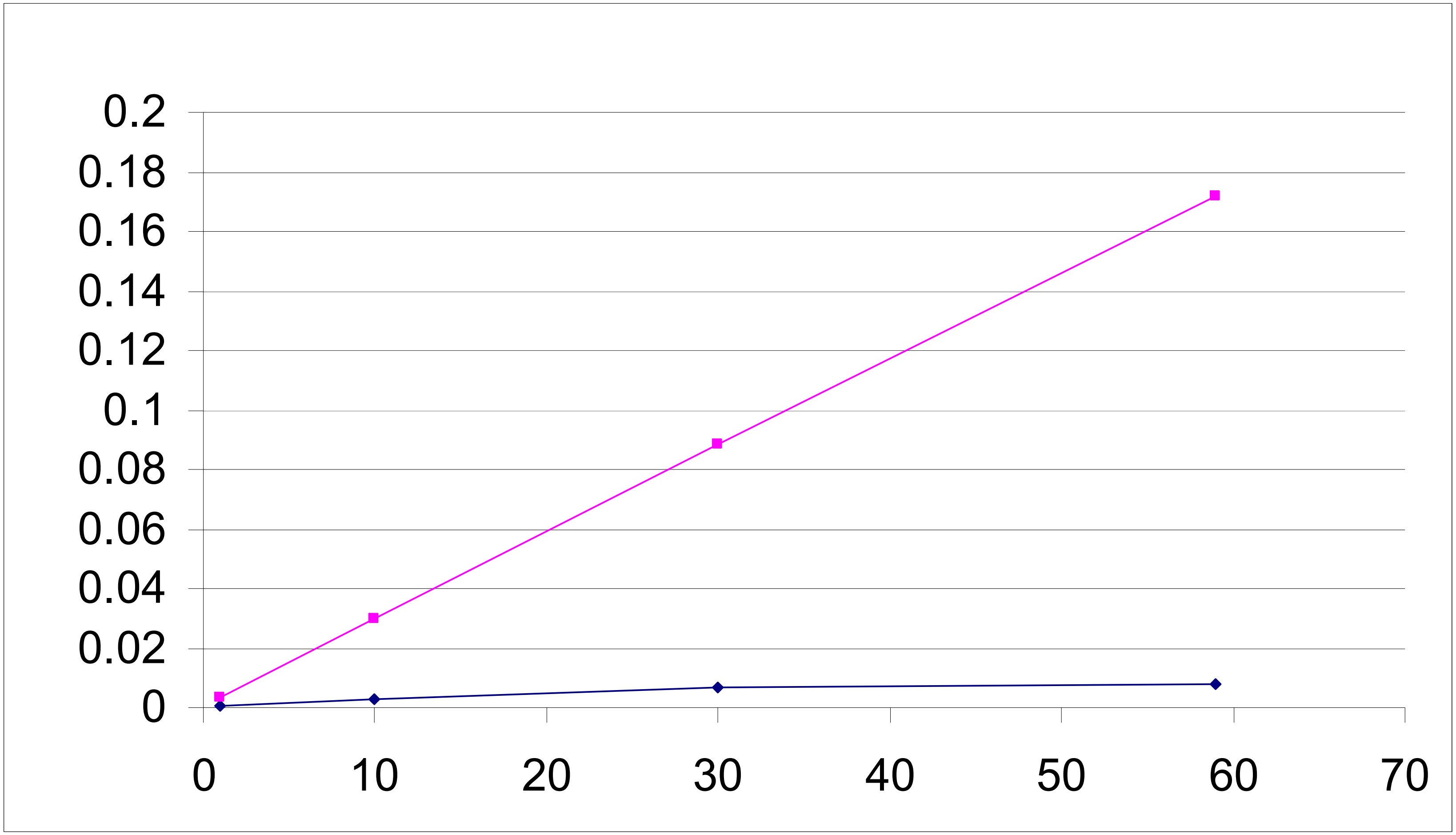
# Other NN packages

- ANN (by Arya & Mount):
  - Based on kd-tree
  - Supports exact and approximate NN
- Metric trees (by Moore et al):
  - Splits along arbitrary directions (not just x,y,..)
  - Further optimizations

# Running times

| | MNIST | Speedup | Corel_hist | Speedup | Corel_uci | Speedup | Aerial | Speedup |
|---|---|---|---|---|---|---|---|---|
| E2LSH-1 | 0.00960 | | | | | | | |
| E2LSH-2 | 0.00851 | | 0.00024 | | 0.00070 | | 0.07400 | |
| E2LSH-3 | | | 0.00018 | | 0.00055 | | 0.00833 | |
| E2LSH-4 | | | | | | | 0.00668 | |
| ANN | 0.25300 | 29.72274 | 0.00018 | 1.011236 | 0.00274 | 4.954792 | 0.00741 | 1.109281 |
| MT | 0.20900 | 24.55357 | 0.00130 | 7.303371 | 0.00650 | 11.75407 | 0.01700 | 2.54491 |

# LSH vs kd-tree (MNIST)



# Caveats

- For ANN (MNIST), setting $\varepsilon$=1000% results in:
  - Query time comparable to LSH
  - Correct NN in about 65% cases, small error otherwise
- However, no guarantees
- LSH eats much more space (for optimal performance):
  - LSH: 1.2 GB
  - Kd-tree: 360 MB

# Conclusions

- Locality-Sensitive Hashing
  - Very good option for near neighbor
  - Worth trying for nearest neighbor
- $E^2$LSH [DIIM'04] available – check my web page for more info

# Refs

- LSH web site (with references):
  http://web.mit.edu/andoni/www/LSH/index.html
- M. Charikar, Similarity estimation techniques from rounding algorithms, STOC'02.
- A. Broder, On the resemblance and containment of documents, SEQUENCES'97.