Dimensionality Reduction Techniques for Proximity Problems

Piotr Indyk, SODA 2000

Bart Adams

Talk Summary

Core algorithm: dimensionality reduction using hashing

Applied to:

- c-nearest neighbor search algorithm (c-NNS)
- c-furthest neighbor search algorithm (c-FNS)

Talk Overview

Introduction

c-Nearest Neighbor Search

c-Furthest Neighbor Search

Conclusion

Talk Overview

Introduction

- Problem Statement
- Hamming Metric
- Dimensionality Reduction

c-Nearest Neighbor Search

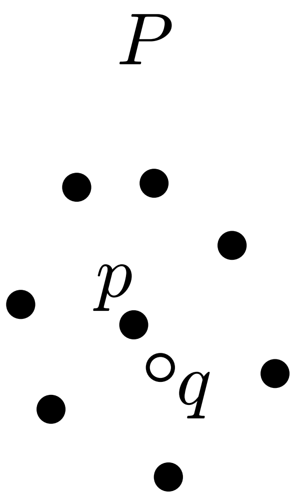
c-Furthest Neighbor Search

Conclusion

Problem Statement

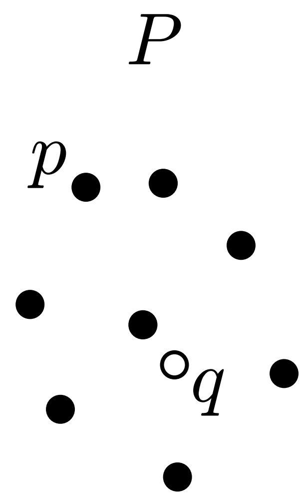
We are dealing with proximity problems

(n points, dimension d)



nearest neighbor search

(NNS)



furthest neighbor search

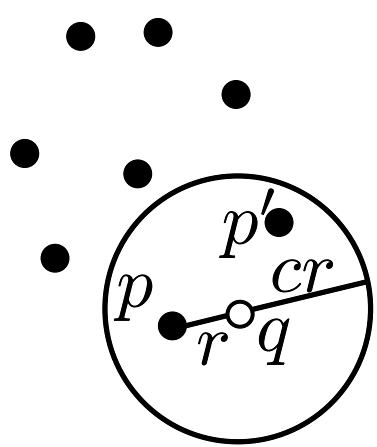
(FNS)

Problem Statement

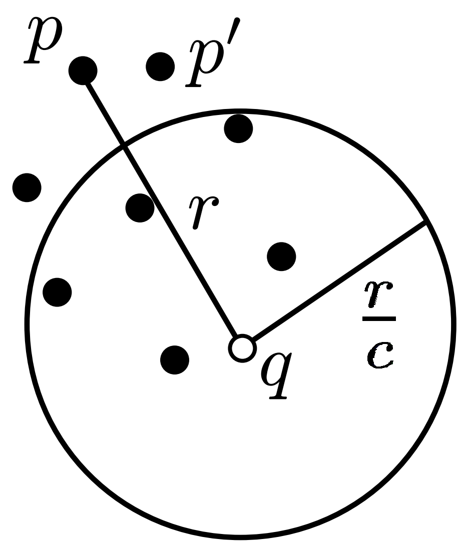
High dimensions: curse of dimensionality

- time and/or space exponential in d

Use approximate algorithms



c-NNS



c-FNS

Problem Statement

Problems with (most) existing work in high d

- randomized Monte Carlo
  - incorrect answers possible

Randomized algorithms in low d

- Las Vegas
  - always correct answer

→ can't we have Las Vegas algorithms for high d?

Hamming Metric

Hamming Space of dimension d

- points are bit-vectors $\{0,1\}^d$

  $$d = 3 : 000, 001, 010, 011, 100, 101, 110, 111$$

- hamming distance $d(x, y)$
  - # positions where x and y differ

Remarks

- simplest high-dimensional setting
- generalizes to larger alphabets Σ

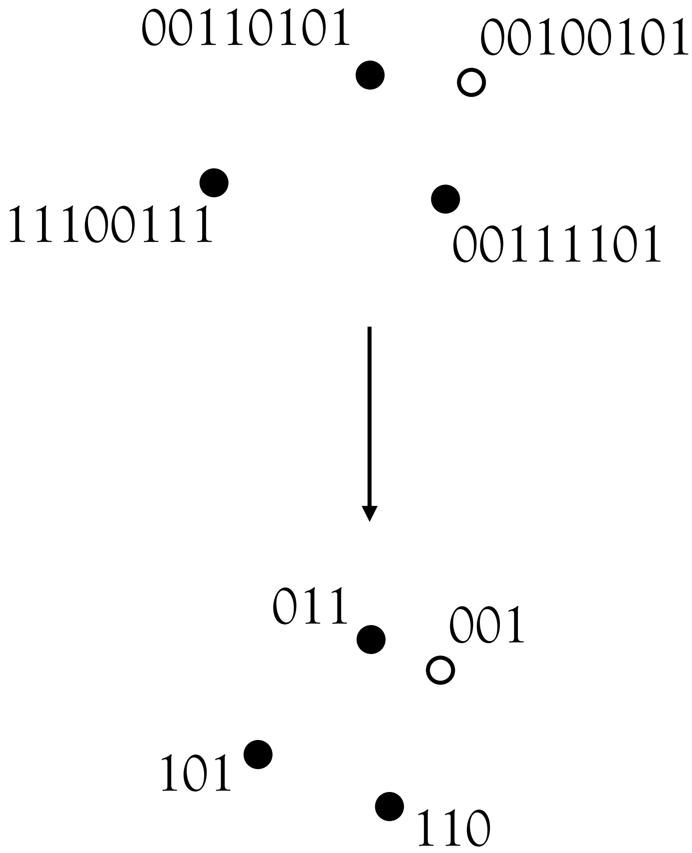
  $$\Sigma = \{\alpha, \beta, \gamma, \delta, \ldots\}$$

Dimensionality Reduction

Main idea

- map from high to low dimension
- preserve distances
- solve problem in low dimension space

→ improved performance at the cost of approximation error



Talk Overview

Introduction

c-Nearest Neighbor Search

c-Furthest Neighbor Search

Conclusion

Las Vegas 1+ε-NNS

Probabilistic NNS

- for Hamming metric
- approximation error 1+ε
- always returns correct answer

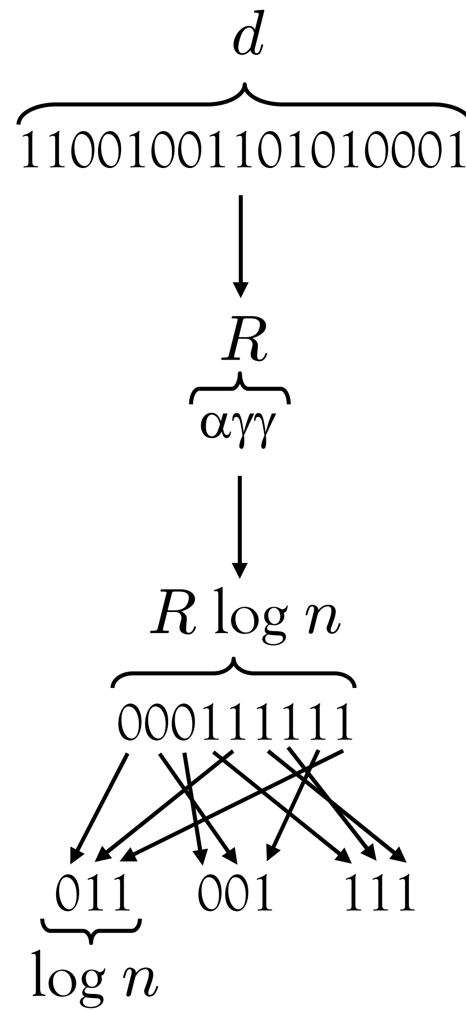
Recall: c-NNS can be reduced to (r, R)-PLEB

- so we will solve this problem

Las Vegas 1+ε-NNS

Main outline

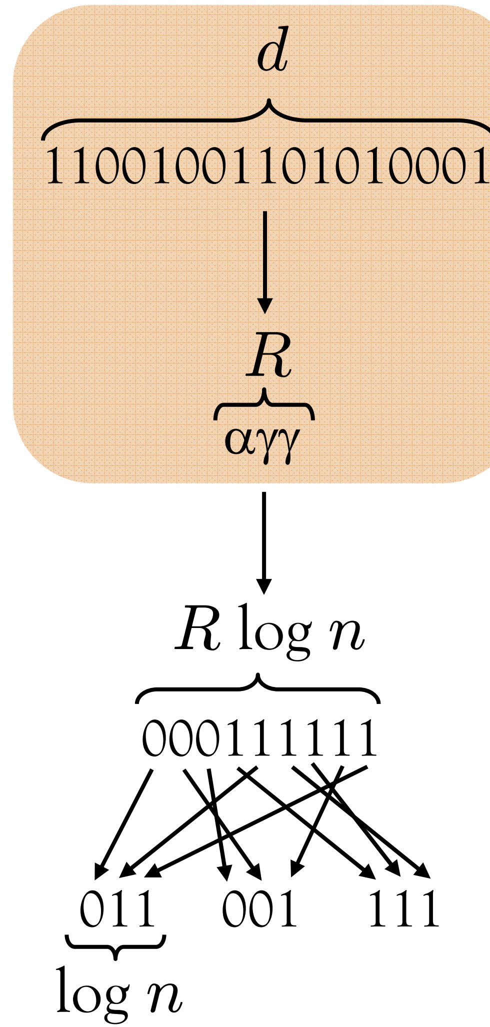
1. hash $\{0,1\}^d$ into $\{\alpha,\beta,\gamma,\delta,\ldots\}^{O(R)}$
   - dimension $O(R)$
2. encode symbols $\alpha,\beta,\gamma,\delta,\ldots$ as binary codes of length $O(\log n)$
   - dimension $O(R \log n)$
3. divide and conquer
   - divide into sets of size $O(\log n)$
   - solve each subproblem
   - take best found solution



Las Vegas 1+ε-NNS

Main outline

1. hash $\{0,1\}^d$ into $\{\alpha,\beta,\gamma,\delta,\ldots\}^{O(R)}$
   - dimension $O(R)$
2. encode symbols α,β,γ,δ,... as binary codes of length O(log n)
   - dimension $O(R \log n)$
3. divide and conquer
   - divide into sets of size $O(\log n)$
   - solve each subproblem
   - take best found solution



Hashing

Find a mapping $f : \{0,1\}^d \to \Sigma^D$

- f is non-expansive

$$d(f(x), f(y)) \leq Sd(x, y)$$

- f is (ε,R)-contractive (almost non-contractive)

$$d(x, y) \geq R \Rightarrow d(f(x), f(y)) \geq SR(1 - \epsilon)$$

Hashing

- $f(x)$ is defined as concatenation

$$f = f_{h_1}(x) f_{h_2}(x) \dots f_{h_{|\mathcal{H}|}}(x)$$

- one $f_h(x)$ is defined using a hash function

$$h(x) = ax \ modP, \ P = \frac{R}{\epsilon}, \ a \in [P]$$

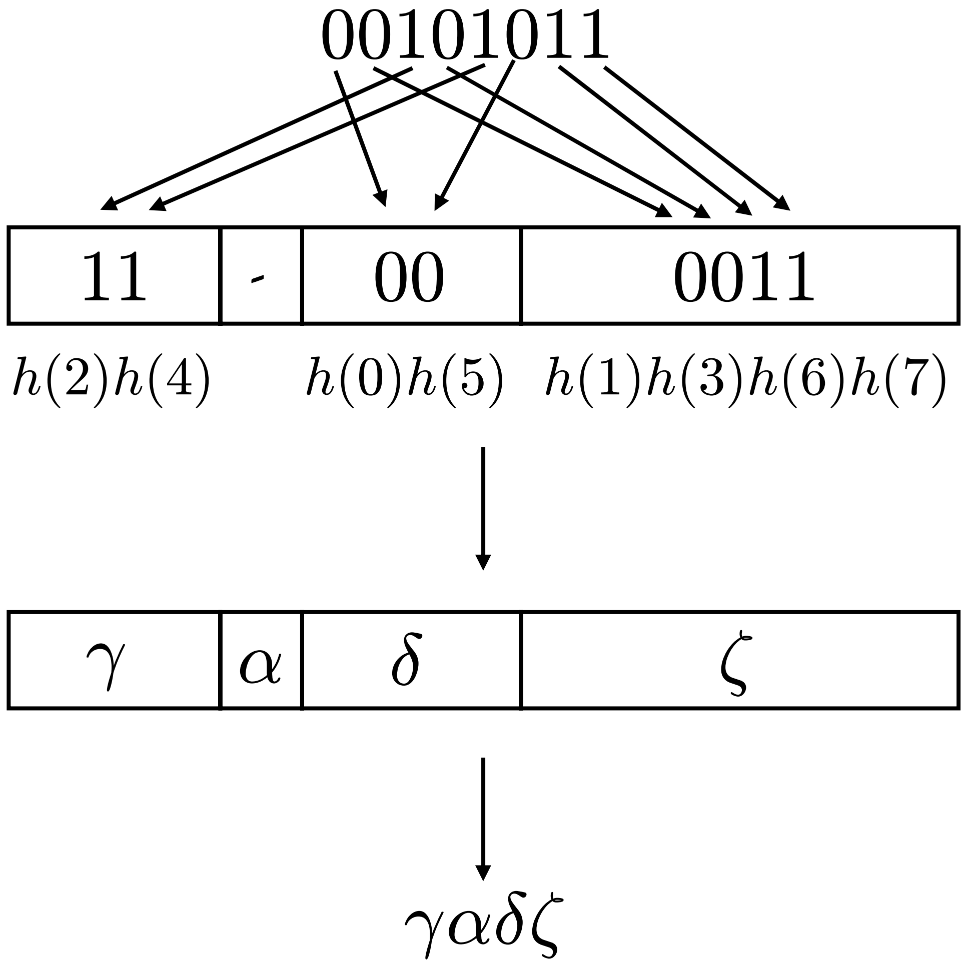
- in total there are P such hash functions, i.e.,

$$|\mathcal{H}| = P$$

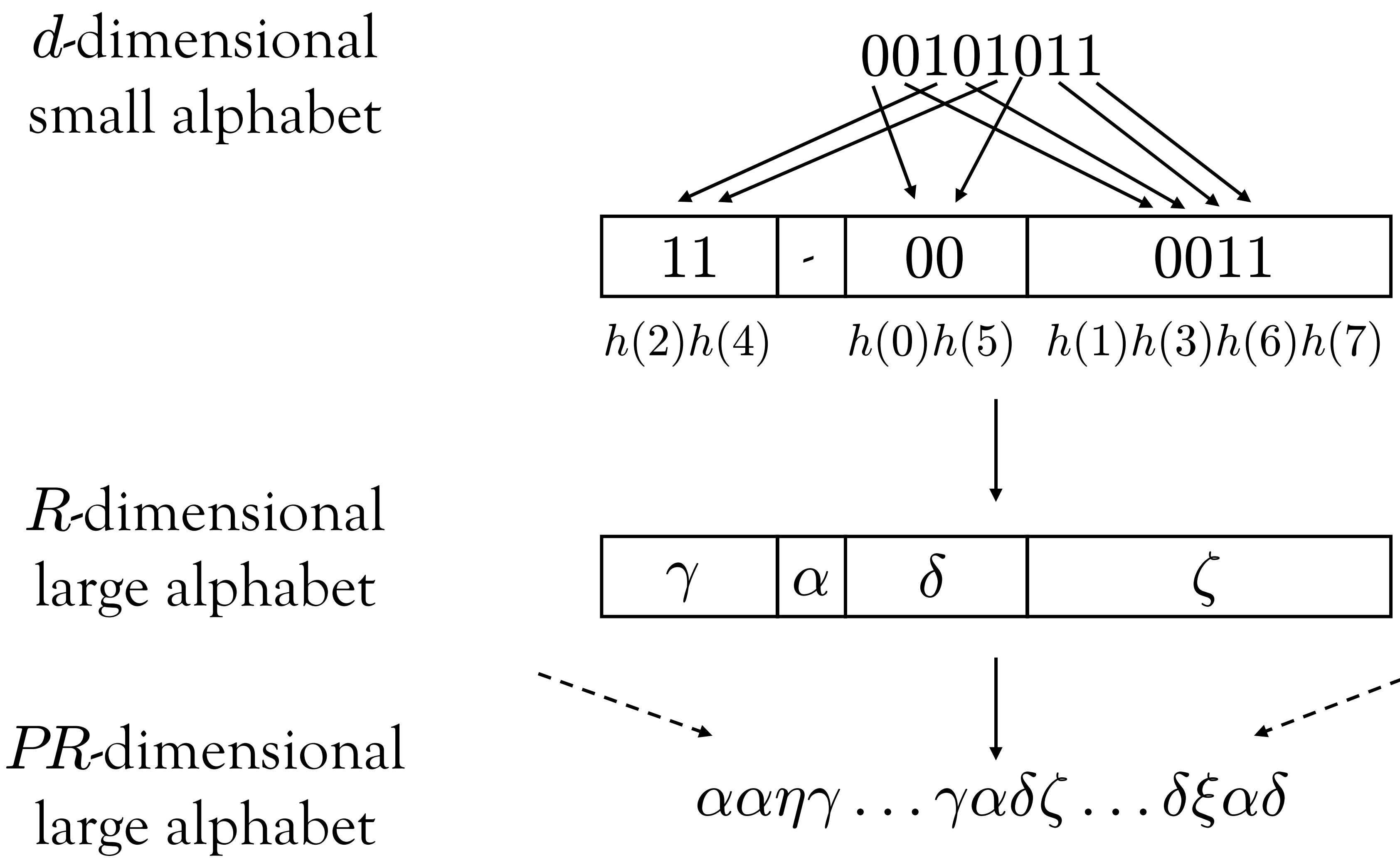
Hashing

Mapping $f_h(x)$

- map each bit x_i into bucket $h(i)$
- sort bits in ascending order of i's
- concatenate all bits within each bucket to one symbol



Hashing



d-dimensional
small alphabet
00101011
11
-
00
0011
h(2)h(4)
h(0)h(5)
h(1)h(3)h(6)h(7)
R-dimensional
large alphabet
γ
α
δ
ζ
PR-dimensional
large alphabet
ααηγ . . . γαδζ . . . δξαδ

Hashing

With $S = |\mathcal{H}|$, one can prove that

- f is non-expansive

$$d(f(x), f(y)) \leq S d(x, y)$$

$\rightarrow$ proof: for each difference bit,
f can generate at most $|\mathcal{H}| = S$
difference symbols.

Hashing

With $S = |\mathcal{H}|$, Piotr Indyk states that one can prove that

- f is (ε,R)-contractive

$$d(x, y) \geq R \Rightarrow d(f(x), f(y)) \geq SR(1 - \epsilon)$$

→ however, recall that $h(x) = ax\ modP, P = \frac{R}{\epsilon}$

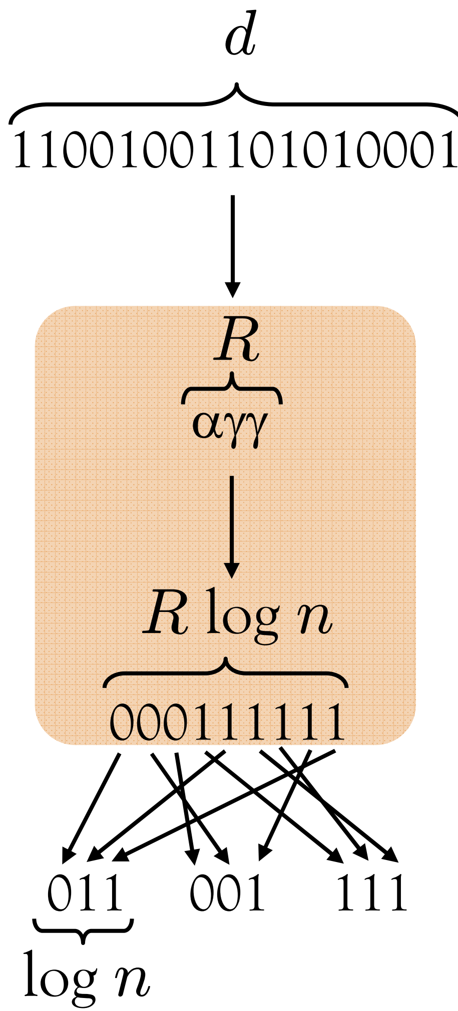
→ it is known that $Pr[h(x) = h(y)] \leq \frac{1}{R/\epsilon}$

→ (ε,R)-contractive only holds with a certain (large) probability (?)

Las Vegas 1+ε-NNS

Main outline

1. hash $\{0,1\}^d$ into $\{\alpha,\beta,\gamma,\delta,\ldots\}^{O(R)}$
   - dimension O(R)
2. encode symbols α,β,γ,δ,... as binary codes of length $O(\log n)$
   - dimension $O(R \log n)$
3. divide and conquer
   - divide into sets of size $O(\log n)$
   - solve each subproblem
   - take best found solution



Coding

Each symbol α from Σ mapped to a binary word $C(\alpha)$ of length l, so that

$$d(C(\alpha), C(\beta)) \in [\tfrac{(1-\epsilon)l}{2}, \tfrac{l}{2}] \qquad l = O(\tfrac{\log|\Sigma|}{\epsilon^2})$$

Example (l=8)

$\alpha \rightarrow C(\alpha)$ = 01000101

$\beta \rightarrow C(\beta)$ = 11011111

Coding

It can be shown, or also seen by intuition, that this mapping is

- non-expansive
- almost non-contractive

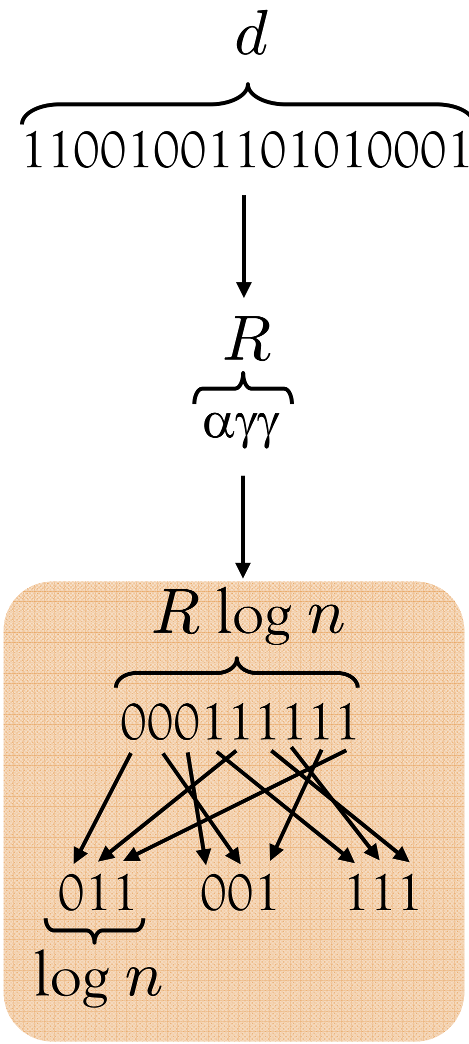
Also, the resulting mapping $g = C \circ f$ (hashing + coding) is

- non-expansive
- almost non-contractive

Las Vegas 1+ε-NNS

Main outline

1. hash $\{0,1\}^d$ into $\{\alpha,\beta,\gamma,\delta,\ldots\}^{O(R)}$
   - dimension O(R)
2. encode symbols α,β,γ,δ,… as binary codes of length O(log n)
   - dimension $O(R \log n)$
3. divide and conquer
   - divide into sets of size $O(\log n)$
   - solve each subproblem
   - take best found solution



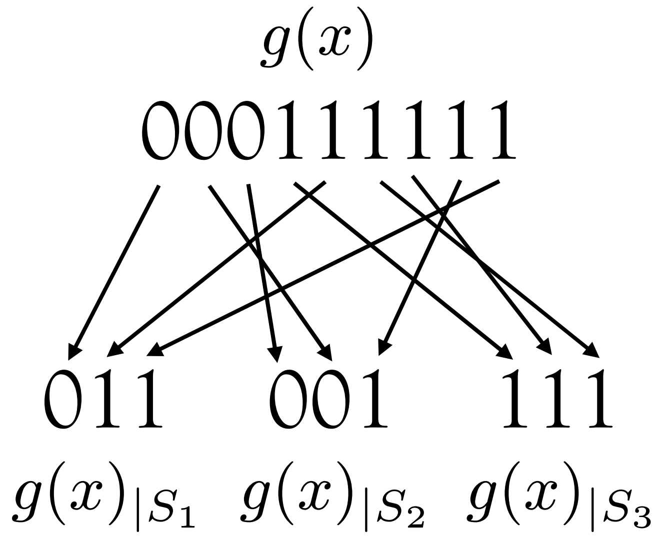
Divide and Conquer

Partition the set of coordinates into random sets $S_1, \ldots, S_k$ of size $s = O(\log n)$

Project g on coordinate sets

One of the projections should be

- non-expansive
- almost non-contractive



Divide and Conquer

Solve NNS problem on each sub-problem $g(x)_{|S_i}$

- dimension $\log n$
- easy problem
- can precompute all solutions with $O(n)$ space

$$O(2^{\log n}) = O(n)$$

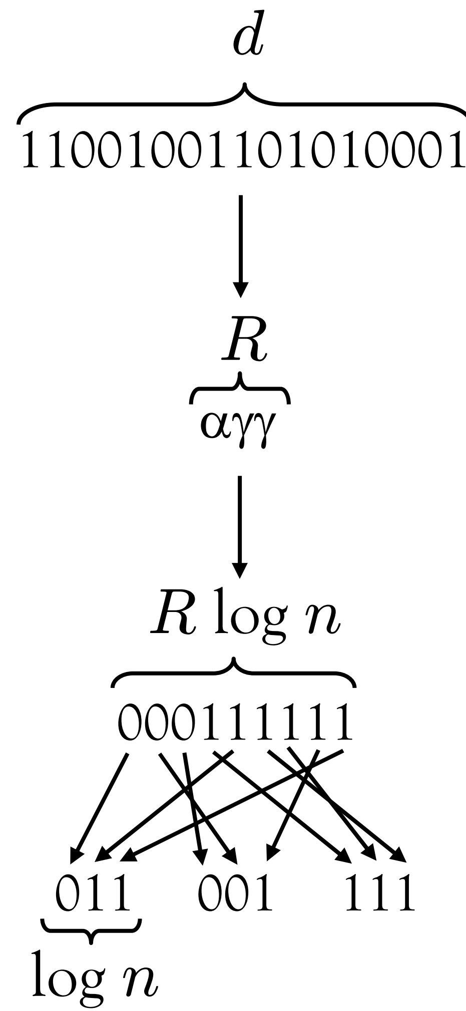
Take best solution as answer

Resulting algorithm is 1+ε approximate
(lots of algebra to prove)

Las Vegas 1+ε-NNS

Main outline

1. hash $\{0,1\}^d$ into $\{\alpha,\beta,\gamma,\delta,\ldots\}^{O(R)}$
   - dimension $O(R)$
2. encode symbols $\alpha,\beta,\gamma,\delta,\ldots$ as binary codes of length $O(\log n)$
   - dimension $O(R \log n)$
3. divide and conquer
   - divide into sets of size $O(\log n)$
   - solve each subproblem
   - take best found solution



Extensions

Basic algorithm can be adapted

- 3+ε-approximate deterministic algorithm
  - make step 3 (divide and conquer) deterministic
- other metrics
  - embed l_1^d into $O(\frac{\Delta d}{\epsilon})$-dimensional Hamming metric (Δ is diameter/closest pair ratio)
  - embed l_2^d into $l_1^{O(d^2)}$

Talk Overview

Introduction

c-Nearest Neighbor Search

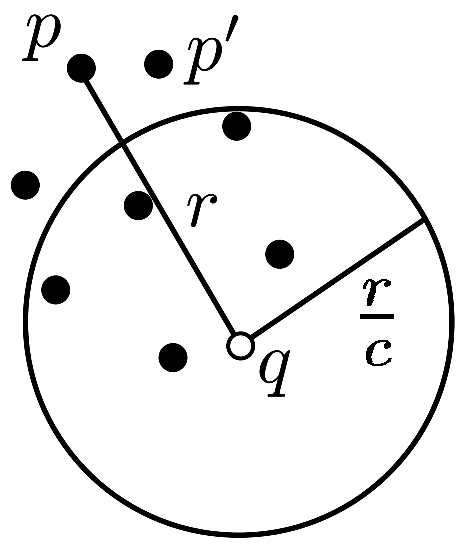
c-Furthest Neighbor Search

Conclusion

FNS to NNS Reduction

Reduce (1+ε)-FNS to (1+ε/6)-NNS

- for $\epsilon \in [0, 2]$
- in Hamming spaces



c-FNS

Basic Idea

For $p, q \in \{0,1\}^d$

$$d(p,q) = d - d(p,\bar{q})$$

$$p = 110011 \qquad p = 110011$$

$$q = 101011 \qquad \bar{q} = 010100$$

$$d(p,q) = 2 = 6 - 4 \qquad d(p,\bar{q}) = 4 = 6 - 2$$

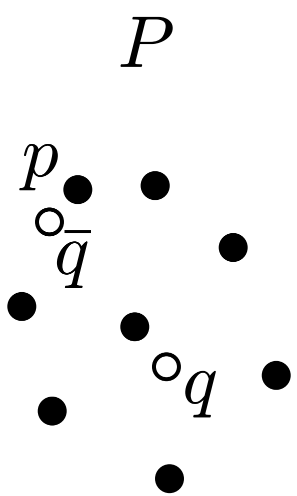
Exact FNS to NNS

Set of points P in $\{0,1\}^d$

p furthest neighbor of q in P

$\Downarrow$

p is nearest neighbor of $\bar{q}$ in P



$\rightarrow$ *exact* versions of NNS and FNS are equivalent

Approximate FNS to NNS

Reduction does *not* preserve approximation

- p FN of q, with $d(q,p) = R$
  - therefore p (exact) NN of $\bar{q}$
- p' c-NN of $\bar{q}$

  $$d(\bar{q}, p') = cd(\bar{q}, p) = c(d - R)$$

- therefore

  $$\frac{d(q,p)}{d(q,p')} = \frac{R}{d-c(d-R)}$$

- so, if we want p' to be c'-FN of q

  $$c' \geq \frac{R}{d-c(d-R)}$$

Approximate FNS to NNS

Reduction does *not* preserve approximation

- so, if we want p' to be c'-FN of q

  $$c' \geq \frac{R}{d-c(d-R)}$$

- or, equivalently,

  $$\frac{1}{c'} \leq \frac{d}{R} + (1 - \frac{d}{R})c$$

- so, the smaller d/R, the better the reduction

→ apply dimensionality reduction
to decrease d/R

Approximate FNS to NNS

With a similar hashing and coding technique, one can reduce d/R and prove:

There is a reduction of

(1+ε)-FNS to (1+ε/6)-NNS

for $\epsilon \in [0, 2]$.

Conclusion

Hashing can be used effectively to overcome the “curse of dimensionality”.

Dimensionality reduction used for two different purposes:

- Las Vegas c-NNS: reduce storage
- FNS $\rightarrow$ NNS: relate approximation factors