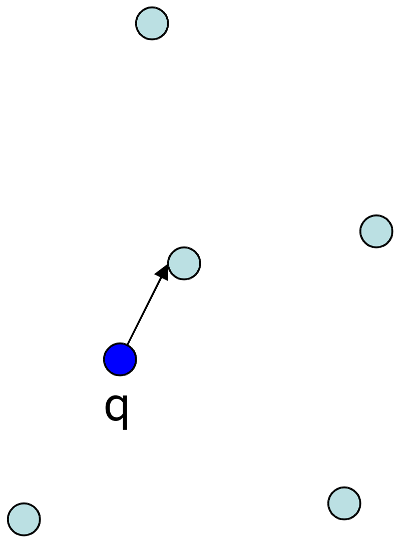
Near(est) Neighbor in High Dimensions

Piotr Indyk

Nearest Neighbor

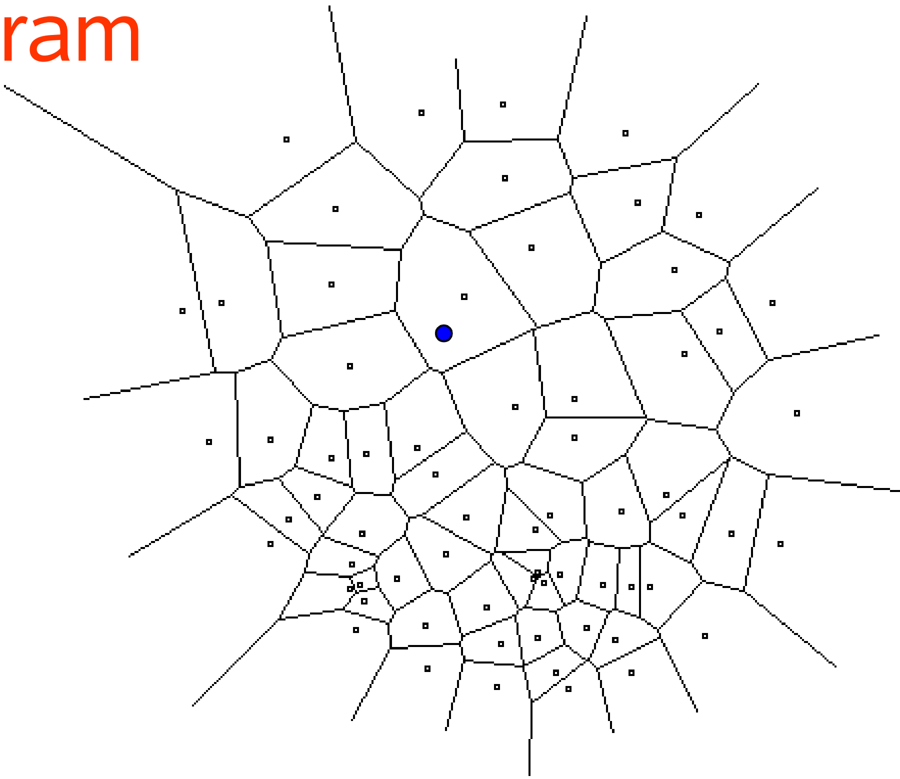
- Given:
  - A set P of points in R^d
  - Goal: build data structure which, for any query q, returns a point $p \in P$ minimizing $||p-q||$



Solution for d=2 (sketch)

- Compute Voronoi diagram
- Given q, perform point location
- Performance:
  - Space: O(n)
  - Query time: O(log n)

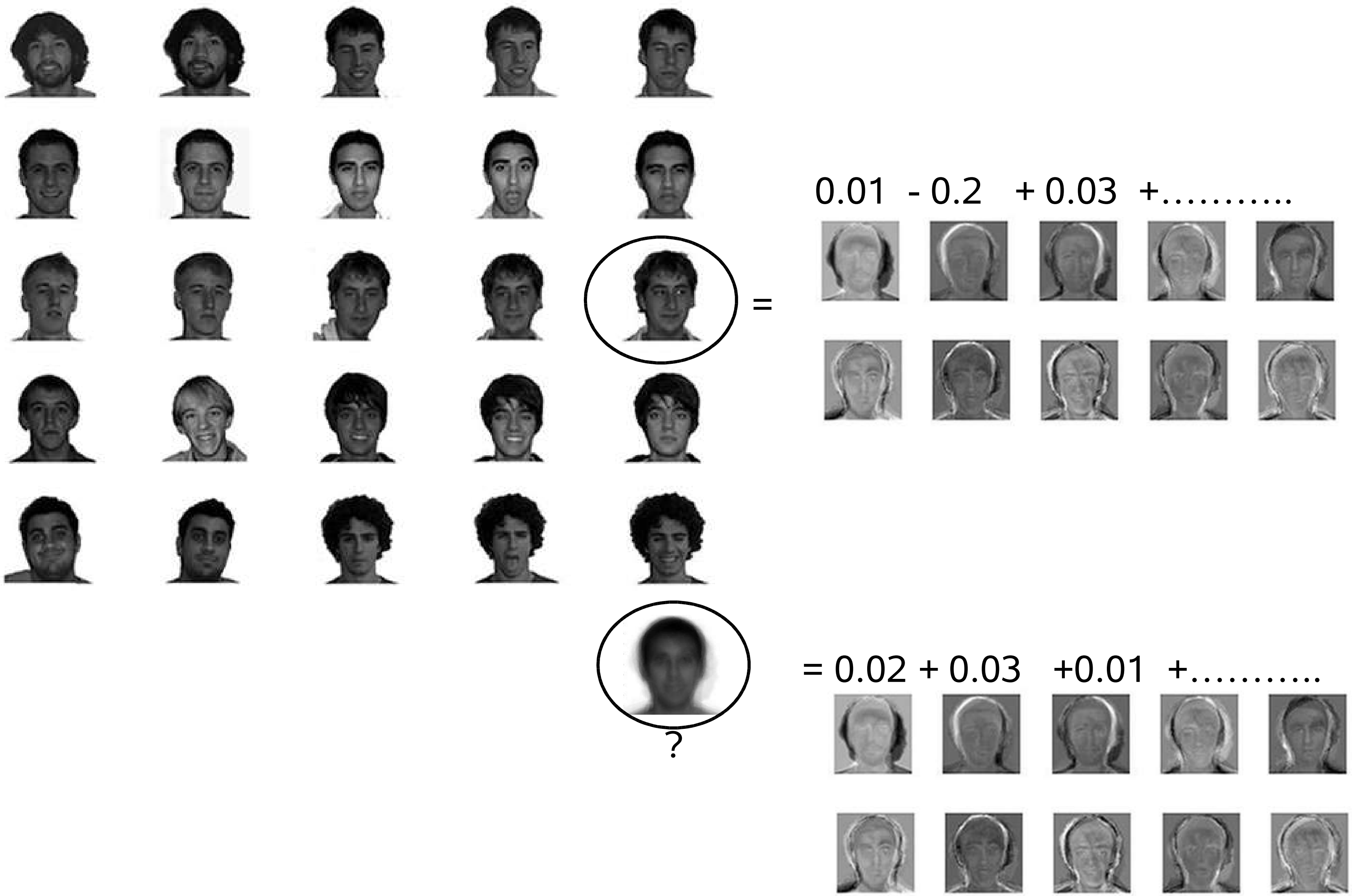
  (see 6.838 for details)



NN in R^d

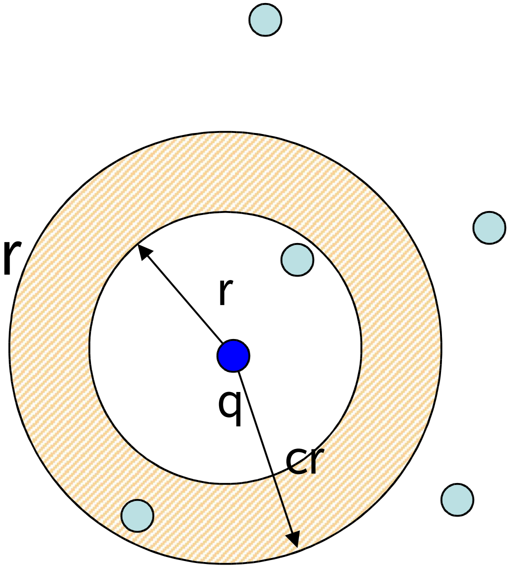
- Exact algorithms use
  - Either $n^{O(d)}$ space,
  - Or $O(dn)$ time
- Approximate algorithms:
  - Space/time exponential in d [Arya-Mount-et al], [Kleinberg'97], [Har-Peled'02]
  - Space/time polynomial in d [Kushilevitz-Ostrovsky-Rabani'98], [Indyk-Motwani'98], [Indyk'98],…

Why high dimensions ? Eigenfaces



(Approximate) Near Neighbor

- Near neighbor:
  - Given:
    - A set P of points in R^d, $r>0$
  - Goal: build data structure which, for any query q, returns a point $p \in P$, $||p-q|| \leq r$ (if it exists)

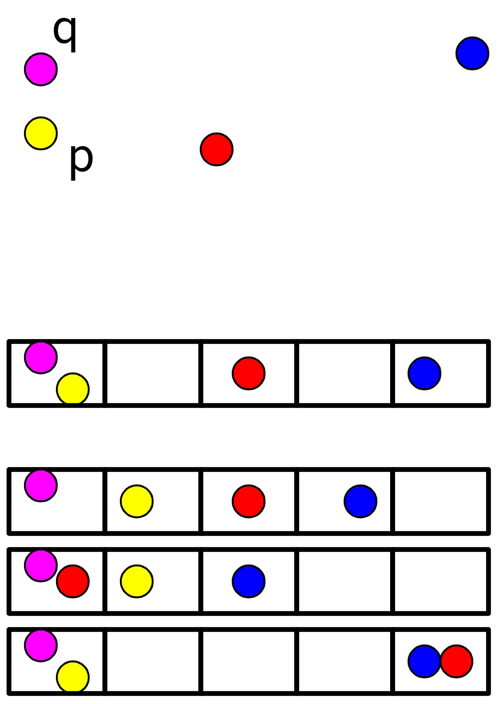


- c-Approximate Near Neighbor:
  - Goal: build data structure which, for any query q:
    - If there is a point $p \in P$, $||p-q|| \leq r$
    - it returns $p' \in P$, $||p-q|| \leq cr$

Locality-Sensitive Hashing

[Indyk-Motwani'98]

- 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

- 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 $Pr[\ h(p)=h(q)\ ] < P_2$
- Algorithm: "essentially" hash using $g(p)=h_1(p).h_2(p)\ldots h_k(p)$
  - Intuition: amplify the probability gap

LSH for Hamming metric [IM'98]

- Hamming metric:
  - p,q are 0-1 vectors of length d
  - $||p-q||$ = # positions i on which $p_i \neq q_i$
- Functions: $h(p)=p_i$, i.e., the i-th bit of p
- We have

$$\Pr[\ h(p)=h(q)\ ] = 1-||p-q||/d$$

Remaining parts

- The details of the algorithm
- Analysis: how many different hash tables do we need
  - Storage
  - Query time
- Extension to non-0-1 case

Technical part

- See slides at
  http://theory.lcs.mit.edu/~indyk/MASS/lec6.pdf
- Notation change: $c=1+\varepsilon$, $||p-q||=D(p,q)$

Other norms

- Can embed l_1^d with coordinates in $\{1 \ldots M\}$ into dM-dimensional Hamming space