Algorithms and Learning for Protein Science

Lecture 6: Nearest neighbors in high dimensional spaces: algorithms and significance

Frederic.Cazals@inria.fr

Overview

- Theory/algorithms
  - Data structures to report nearest neighbors: kd-trees, random project trees, metric trees
  - Performance analysis
  - (Distance) concentration phenomena
- Structural bioinformatics
  - Lecture 3: Exploring conformational spaces
  - Lecture 4: Structural comparisons

Algorithms

PART 1: Nearest neighbors: data structures
PART 2: Nearest neighbors: analysis
PART 3: Concentration phenomena: introduction

Algorithms

Introduction

kd-trees and basic search algorithms

kd-trees and random projection trees: improved search algorithms

Metric trees and variants

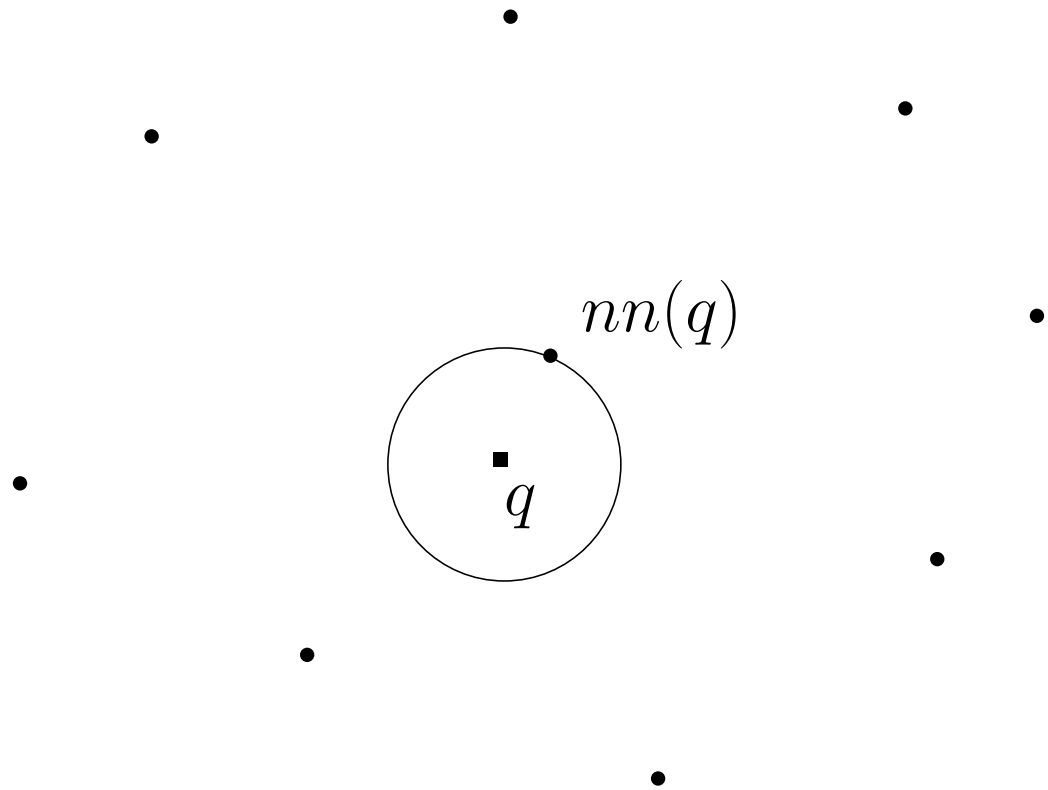
Applications

- A core problem in the following applications:
  - clustering, k-means algorithms
  - information retrieval in databases
  - information theory : vector quantization encoding
  - classification in learning theory
  - ...

Nearest Neighbors: Getting Started

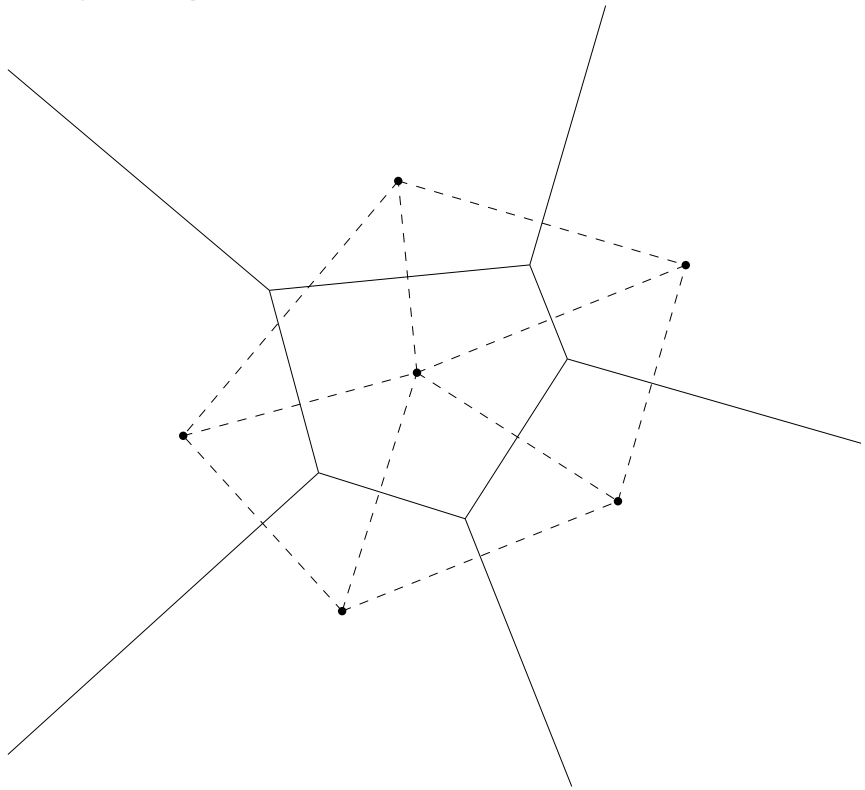
- Input: a set of points (aka sites) P in $\mathbb{R}^d$, a query point q
- Output: $nn(q, P)$, the point of P nearest to q

$$d(q, P) = d\left(q, nn(q, P)\right). \tag{1}$$



The Euclidean Voronoi Diagram and its Dual the Delaunay Triangulation

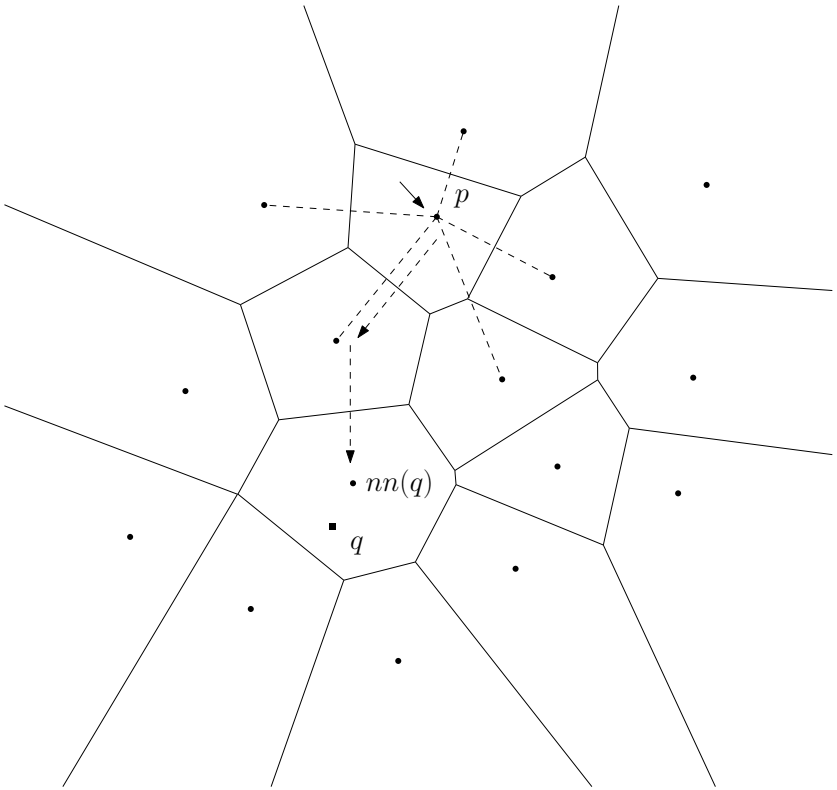
▷ Voronoi and Delaunay diagrams



▷ Key properties:

- Voronoi cells of all dimensions
- Voronoi - Delaunay via the nerve construction
- Duality : cells of dim. $d - k$ vs cells of dimension k
- The empty ball property

Nearest Neighbors Using Voronoi Diagrams



▷ Nearest neighbor by walking

- start from any point $p \in P$
- while $\exists$ a neighbor $n(p)$ of p in Vor(P)
  closer to q than p,
  step to it: $p = n(p)$
- done $nn(q) = p$

▷ Argument: the Delaunay neighborhood of a point is complete

Vor(p, P)= cell of p in Vor(P)

$N(p)$ = set of neighbors of p in Vor(P)

$N'(p) = \{p\} \bigcup N(p)$

$$\text{Vor}(p, N'(p)) = \text{Vor}(p, P)$$

▷ Exercise: specify the algorithm using DT

The Nearest Neighbors Problem: Overview

▷ Strategy: prepocess point set P of n points in $\mathbb{R}^d$ into a data structure (DS) for fast nearest neighbor queries answer.

▷ Ideal wish list:

- The DS should have linear size
- A query should have sub-linear complexity i.e. $o(n)$
  - When $d = 1$: balanced binary search trees yield $O(\log n)$

▷ Core difficulties:

- Curse of dimensionality in $\mathbb{R}^d$: for high d, it is difficult to outperform the linear scan
- Interpretation: meaningfull-ness of distances in high dimensional spaces – distance concentration phenomena.

The Nearest Neighbors Problem: Elementary Options

▷ The trivial solution :
$O(dn)$ space, $O(dn)$ query time

▷ Voronoi diagram

$d = 2$, $O(n)$ space $O(\log n)$ query time
$d > 2$, $O\left(n^{\lceil \frac{d}{2} \rceil}\right)$ space

$\rightarrow$ Under locally uniform condition on point distribution
the 1-skeleton Delaunay hierarchy achieves :
$O(n)$ space, $O(c^d \log n)$ expected query time.

▷ Spatial partitions based on trees

The Nearest Neighbors Problem: Variants

- Variants:
  - k-nearest neighbors: find the k points in P that are nearest to q
  - given $r > 0$, find the points in P at distance less than r from q
  - Various metrics
    - L_2, L_p, L_∞
    - String: Hamming distance
    - Images, graphs: distance based on optimal transportation
    - Point sets: distances via optimal alignment
  - Non metric spaces – cf metric trees
- Main contenders in metric spaces:
  - Tree like data structures:
    - quad-trees – and its variant ANN
    - (randomized) kd-trees
    - k-means trees – partition derived from k-means with k=2
  - Locally Sensitive Hashing

Comparison and appetizer: setup

- Contenders: various hierarchical methods for approximate NN
  - randomized kd-trees: hierarchical partition with split direction chosen at random
  - k-means trees: hierarchical partition with split direction derived from k-means
  - Approximate Nearest Neighbors (ANN)
  - Locally Sensitive Hashing (LSH)

- Assessment for the accuracy of the approximation: precision i.e. fraction of queries for which the correct NN is found

- Two main questions addressed:
  - Question 1: for a fixed database, which algorithm is best?
  - Question 2: are the performances stable when the size of the DB changes?

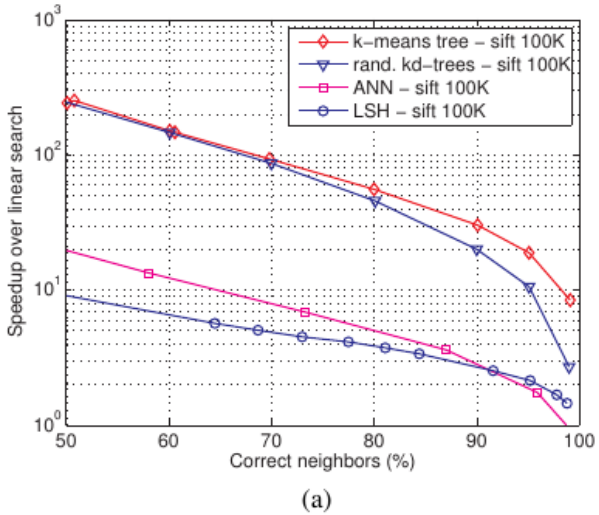
Ref: Muja and Lowe, VISAPP 2009

Ref: O'Hara and Draper, Applications of Computer Vision (WACV), 2013

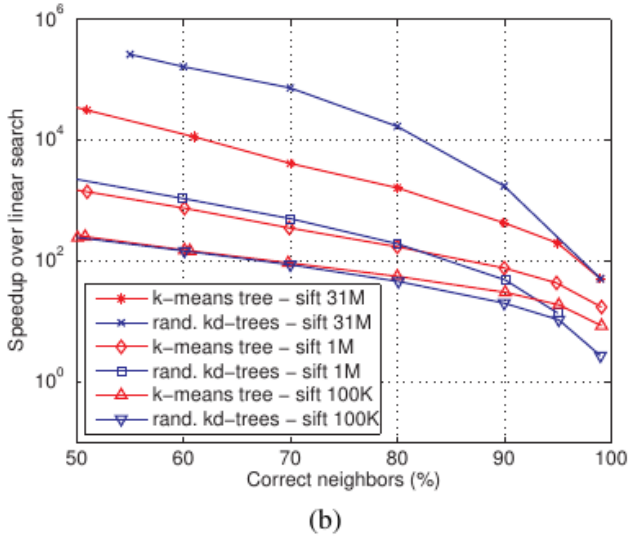
Main Contenders: Typical Results for Approximate NN

▷ DB used : Scale-Invariant Feature Transform (SIFT) for images: $\{(x_i, y_i, \sigma_i)\}$

▷ Question 1: best algorithm



▷ Question 2 – for winners only
i.e. for rand. kd-trees and k-means trees



▷ Take-home messages:

- Randomized kd-trees and k-means trees win
  - splits must exploit the variance in the dataset
- Speed-ups consistent when DB size increases

▷Ref: Muja and Lowe, VISAPP 2009
▷Ref: O'Hara and Draper, Applications of Computer Vision (WACV), 2013

Algorithms

Introduction

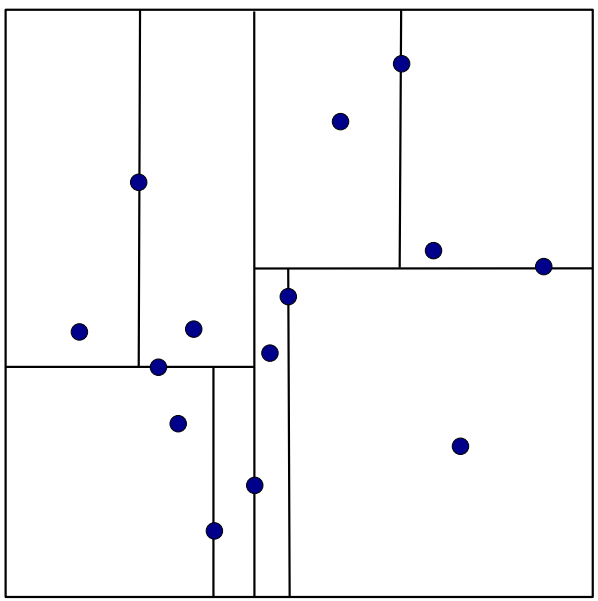
kd-trees and basic search algorithms

kd-trees and random projection trees: improved search algorithms

Metric trees and variants

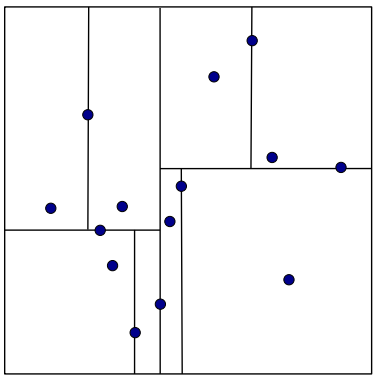
kd-tree for a collection of points (sites) P

- Definition:
  - A binary tree
  - Any internal node implements a spatial partition induced by a hyperplane H, splitting the point cloud into two equal subsets
    - right subtree: points p on one side of H
    - left subtree: remaining points
  - The process halts when a node contains $\leq n_0$ points



Nb: the point realizing the median is stored in the node performing the split

kd-tree for a collection of points P



procedure `build_kdTree`$((S))$

- $n \leftarrow newNode$
- **if** $\mid S \mid \leq n_0$ **then**
  - Store the point of S into a container of n
  - **return** n
- **else**
  - $dir = depth \mod d$
  - Project the points of S along direction dir
  - Compute the median m ▷ *Split into two equal subsets*
  - $n.sample \leftarrow$ sample v realizing the median
  - $L \leftarrow$ point from $S \backslash \{v\}$ whose dirth coord is $< m$
  - $R \leftarrow$ point from $S \backslash \{v\}$ whose dirth coord is $\geq m$
  - $n.left \leftarrow$ build_kdTree(L)
  - $n.right \leftarrow$ build_kdTree(R)
  - **return** n

kd-tree: search

- Main considerations:
  - Exact versus approximate NN
  - No free lunch: complexity matters
- Three main search strategies:
  - (Approx.) the defeatist search: simple, but may fail
    (Nb: see later, distance concentration phenomema)
  - (Exact) the descending search: always succeeds, but may take time
  - (Exact) the priority search: strikes a compromise between the defeatist and descending strategies

kd-tree search: the defeatist search

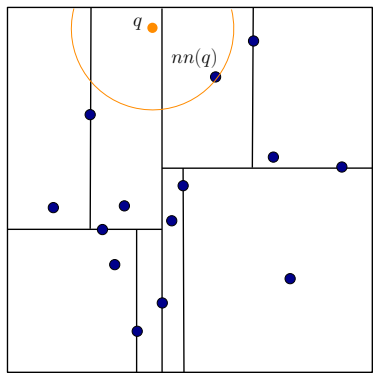
▷ Key idea: recursively visit the subtree containing the query point

procedure DEFEATIEST_SEARCH(q)
- ▷ *Maintains $nn(q)$ of q, and $\tau = d(q, nn(q))$* ◁
- $n \leftarrow root$; $\tau \leftarrow d(q, n.sample)$
- **while** $n \neq NIL$ **do**
  - Possibly update $nn(q)$ using $n.sample$, and τ
  - ▷ *Left subtree only* ◁
  - **if** $q \in$ Domain of L **then**
    - defeatist_search_kdTree($n.left$)
  - ▷ *Right subtree only* ◁
  - **if** $q \in$ Domain of R **then**
    - defeatist_search_kdTree($n.right$)

▷ Complexity: assuming leaves of size n_0 – depth satisfies $2^h n_0 = n$

- search cost: $O(n_0 + \log(n/n_0))$

▷ Caveat: failure



kd-tree search: the exhaustive descending search

▷ Key idea: visit one or two subtree, depending on the distance $d(q, nn(q))$

procedure DESCENDING_SEARCH_KDTREE(q)

▷ *Maintains $nn(q)$ of q, and $\tau = d(q, nn(q))$* ◁

▷ *Uses the domain of a node n* ◁

$n \leftarrow root$

$\tau \leftarrow d(q, n.sample)$

while $n \neq NIL$ **do**

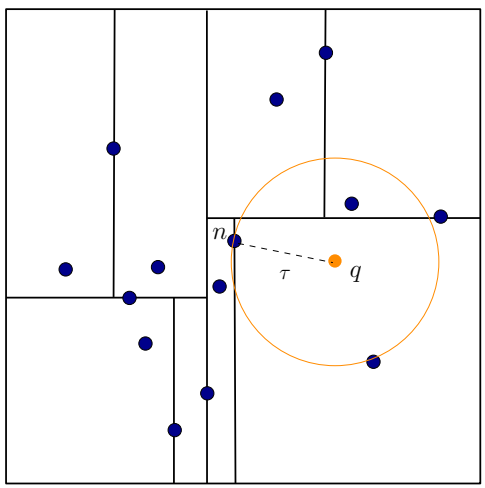
Possibly update $nn(q)$ using $n.sample$

if $Sphere(q, \tau) \cap$ Domain of L **then**

descending_search_kdTree($n.left$)

if $Sphere(q, \tau) \cap$ Domain of R **then**

descending_search_kdTree($n.right$)



▷ Complexity: ?

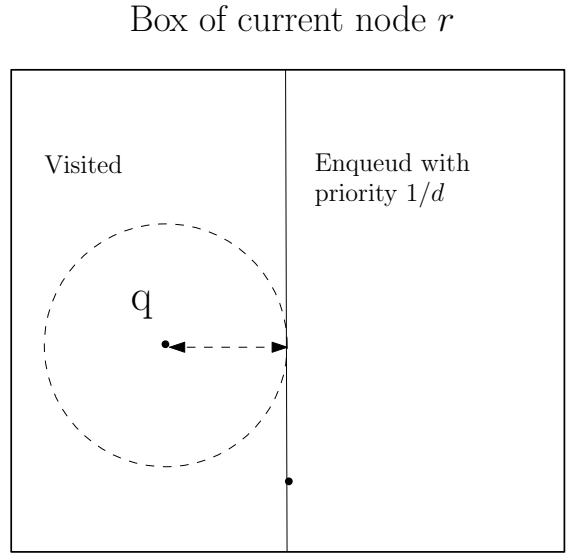
kd-tree search: the priority search

- Challenge: report the exact NN, while visiting as few nodes as possible.
- Priority search, key ideas:
  - Uses a priority queue to store nodes (regions), with a priority inversely proportional to the distance to q.
  - Upon popping a node, the corresponding subtree is descended to visit the node closest to q. Upon descending, $nn(q)$ is updated.
  - While descending, the child not visited is possibly enqueued,

kd-tree search: priority search (algorithm)

- Uses a priority queue Q to enumerate nodes by increasing distance to query q

```
procedure PRIORIRY_SEARCH(q)
    ▷ Maintains nn(q) of q, and τ = d(q, nn(q)) ◁
    nn(q) ← root.sample
    Q.insert(root)
    while True do
        if Q.empty() then
            return
        r ← Q.pop()                ▷ Node with highest priority
        ▷ The nearest box is too far wrt nn(q) ◁
        if d(bbox(r), q) > τ then
            return
        ▷ Descend into box nearest to q (or containing q) and possibly enqueue the 2nd node ◁
        for Nodes n on the path from r to the box nearest to q (or containing q) do
            ▷ One subtree: rocessed for sure ◁
            d ← d(q, n.sample)
            if d < τ then
                nn(q) ← n.sample; τ ← d
            ▷ Second subtree: possibly enqueued ◁
            f ← brother of n
            if d(bbox(f), q) ≤ τ then
                Q.insert(f, 1/d)        ▷ Priority: 1/d
```



NB: Enquing criterion can be adapted to report an $(1+\varepsilon)$ approx. of the exact NN

kd-tree search: priority search (analysis)

- Pros and cons:
  - + nn always found
  - + linear storage
  - – nn often found at an early stage ... then time spent in useless recursion
  - – In the worst-case, all nodes are visited.
  - – Maintaining the priority queue Q has a cost
- Variants and improvements:
  - Initially the Q with all nodes from root to leaf containing the query
  - Stopping the recursion once a fraction of nodes has been visited
  - Backing up defeatist search with overlapping cells
  - Combining multiple randomized kd-trees

References

Sam06 H. Samet. Foundations of multidimensional and metric data structures. Morgan Kaufmann, 2006.

SDE05 G. Shakhnarovich, T. Darrell, and P. Indyk (Eds). Nearest-Neighbors Methods in Learning and Vision. Theory and Practice. MIT press, 2005.

Algorithms

Introduction

kd-trees and basic search algorithms

kd-trees and random projection trees: improved search algorithms

Metric trees and variants

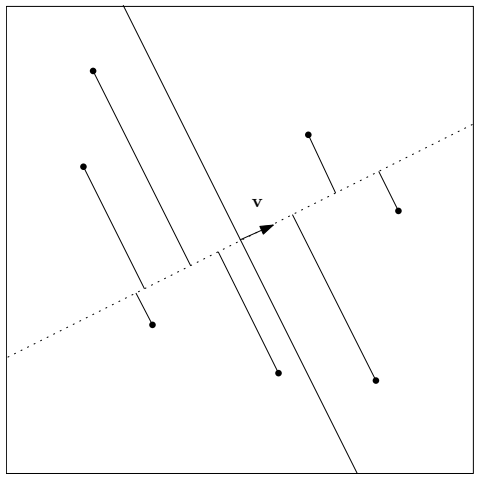
Improvements aiming at fixing the defeatist search

- Defeatist search: (early) choice of one side is risky
- Simple improvements:
  - Use several trees, and pick the best neighbor(s)
  - Allow overlap between cells in a node: selected points stored twice $\rightarrow$ spill trees
  - Use randomization to obtain different partitions rescuing the defeatist search
    - different permutations of coordinate axis
    - directions aiming at maximizing the variance
  - Next: randomization captures information on directions carrying variance

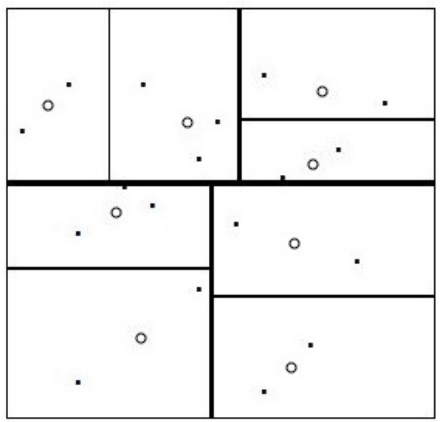
Random projection trees (RPTrees)

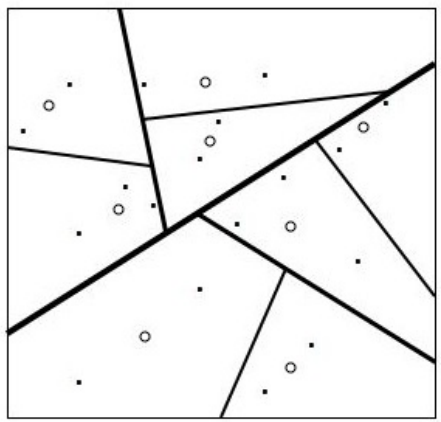
Aka Random partition trees (RPTrees!)

- ▷ kd-tree: axis parallel splits
- ▷ Splitting along a random direction $U \in S^{d-1}$: project onto U and split at the (perturbed) median



- ▷ Resulting spatial partition





Random projection trees: generic algorithm with jitter

▷ Below: version where one also jitters the median defining the split

procedure BUILD_RPTREE(S)
  if If $|S| \leq n_0$ **then**
    $n \leftarrow newNode$
    Store S into n
    return n
  Pick U uniformly at random from the unit sphere
  Pick β uniformly at random from $[1/4, 3/4]$
  Let v be the β-fractile point on the projection of S onto U
  $Rule(x) =$ (left if $\langle x, U \rangle < v$, otherwise right)
  $left\_tree \leftarrow$ build_RPTree($\{x \in S : Rule(x) = left\}$)
  $right\_tree \leftarrow$ build_RPTree($\{x \in S : Rule(x) = right\}$)
  return $(Rule(\cdot), left\_tree, right\_tree)$

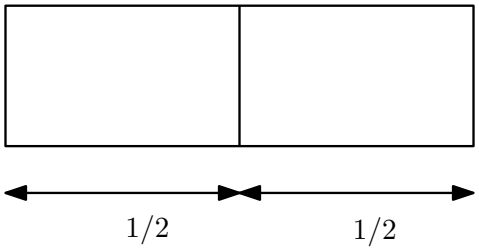
▷ Remark: RP trees have the following property – more later: diameter of the cells decrease down the tree at a rate depending on the intrinsic dimension of the data.

RPTrees: varying splits and their applications

▷ Various types of splits possible

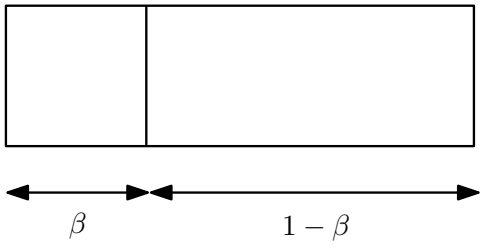
Randomized partition tree:

- exact split



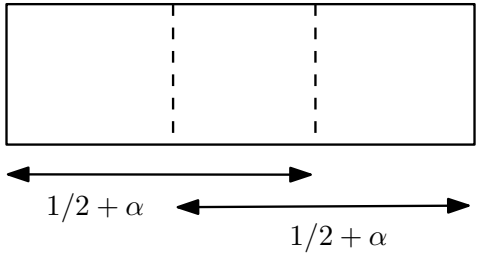
Randomized partition tree:

- perturbed split



Spill tree with overlapping split:

- regular spill tree
- virtual spill tree



▷ NB: splits monitor the tree structure and the search route

▷ Spill trees:

– Regular spill trees:
overlapping cells yield redundant storage of points

– Virtual spill trees:
median splits used – no redundant storage
query routed in multiple leaves using overlapping splits

▷ Summary: tree creation versus search

	Routing data	Routing queries (defeatist style)
RP tree	Perturbed split	Perturbed split
Regular spill tree	Overlapping split	Median split
Virtual spill tree	Median split	Overlapping split

Failure of the defeatist search

▷ Goal: probability that a defeatist seach does not return the exact nearest neighbor(s)?

▷ The event to be analyzed, denoted **Err**:

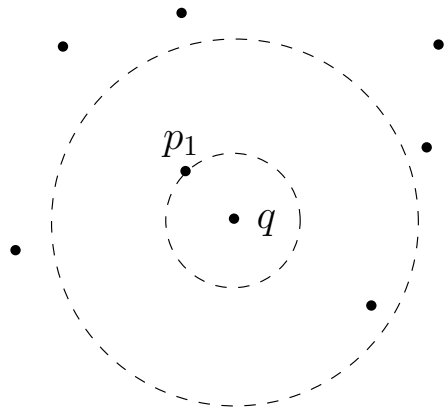
- $k = 1$:the NN query does not return $p_{(1)}$
- $k > 1$: the NN query does not return $p_{(1)}, \ldots, p_{(k)}$

Qualifying the hardness of nearest neighbor queries

▷ Notations:

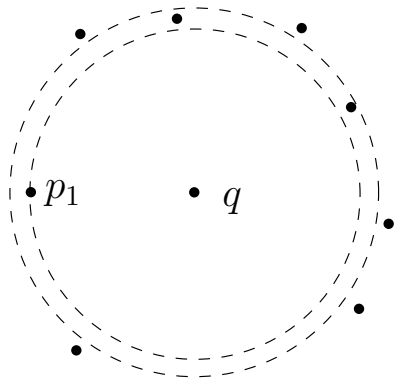
- Dataset $P = p_1, \ldots, p_n$
- Sorted dataset wrt q: $p_{(1)}, \ldots, p_{(n)}$

$$\Phi(q, P) = \frac{1}{n} \sum_{i=2}^{n} \frac{\left\| q - p_{(1)} \right\|}{\left\| q - p_{(i)} \right\|}. \qquad (2)$$



▷ Extreme cases:

- $\Phi \sim 0$: p_1 isolated, finding it should be easy
- $\Phi \sim 1$: points equidistant from q; finding $p_{(1)}$ should be hard



▷ Rationale: in using RPT and spill trees with the defeatist search, the probability of success should depend upon Φ.

Generalizations of the function Φ

▷ Rationale: function Φ shall be used for nodes containing a subset of the database

▷ For a cell containing m points – evaluate the remaining points in that cell:

$$\Phi_m(q, P) = \frac{1}{m} \sum_{i=2}^{m} \frac{\left\| q - p_{(1)} \right\|}{\left\| q - p_{(i)} \right\|}. \tag{3}$$

▷ If one is interested in the k nearest neighbors – evaluate the remaining points too:

$$\Phi_{k,m}(q, P) = \frac{1}{m} \sum_{i=k+1}^{m} \frac{\left\| q - p_{(1)} \right\| + \cdots + \left\| q - p_{(k)} \right\|}{\left\| q - p_{(i)} \right\|}. \tag{4}$$

Theoretical results on the performances

- Analysis to come next:
  - RPTrees: success/failure probability to report NN
  - Random projections and adaptation to intrinsic dimension
  - NN, distances and concentration phenomena

Algorithms

Introduction

kd-trees and basic search algorithms

kd-trees and random projection trees: improved search algorithms

Metric trees and variants

Metric spaces

Definition 1. A metric space is a pair (M, d), with $d : M \times M \to \mathbb{R}^+$, such that:

- (1) Positivity: $d(x, y) \geq 0$
- (1a) Self-distance: $d(x, x) = 0$
- (1b) Isolation: $x \neq y \Rightarrow d(x, y) > 0$
- (2) Symmetry: $d(x, y) = d(y, x)$
- (3) Triangle inequality: $d(x, y) \leq d(x, z) + d(y, z)$

▷ Product metric. Assume that for some $k > 1$:

$$M = M_1 \times \cdots \times M_k. \tag{5}$$

and that each (M_i, d_i) is a metric space. For $p \geq 1$, the product metric is:

$$d(x, y) = (\sum_{k=1}^{k} d_i(x_i, y_i)^p)^{1/p} \tag{6}$$

Some particular cases are:

- $(M_i = \mathbb{R}, d_i = |\cdot|)$: L_p metrics.
- $p = 1, d_i =$ uniform metric: Hamming distance.

Using the triangle inequality

Lemma 2. For any three points $p, q, s \in M$, for any $r > 0$, and for any point set $P \subset M$, one has:

$$| d(q,p) - d(p,s) | \leq d(q,s) \leq d(q,p) + d(p,s) \tag{7}$$

$$d(q,s) \geq d_P(q,s) := \max_{p \in P} | d(q,p) - d(p,s) | \tag{8}$$

$$\begin{cases} d(p,s) > d(p,q) + r \Rightarrow d(q,s) > r \\ d(p,s) < d(p,q) - r \Rightarrow d(q,s) > r. \end{cases} \tag{9}$$

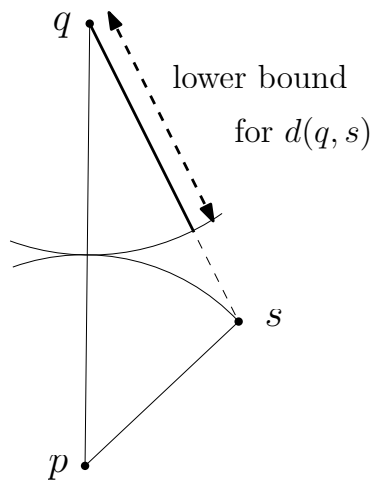
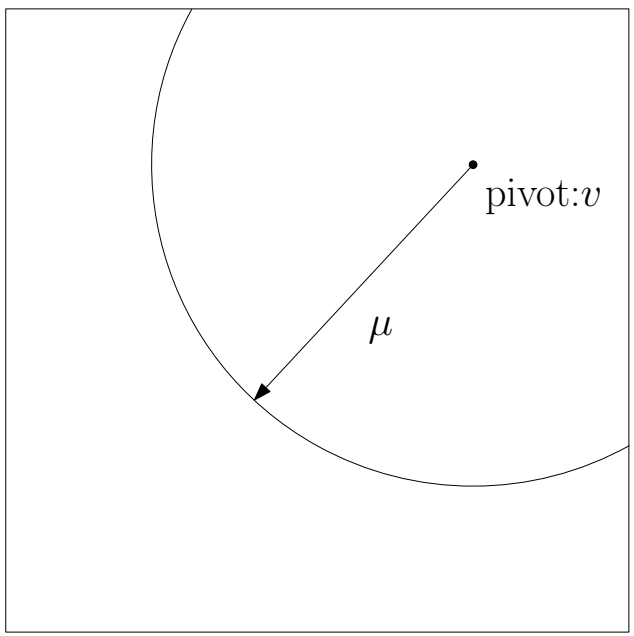


Figure: **Lower bound from the triangle inequality, see Lemma 2**

Metric tree: definition

▷ Definition:

- A binary tree
- Any internal node implements a spherical cut defined by the distance μ to a pivot v
  - right subtree: points p such that $d(pivot, p) \geq \mu$
  - left subtree: points p such that $d(pivot, p) < \mu$



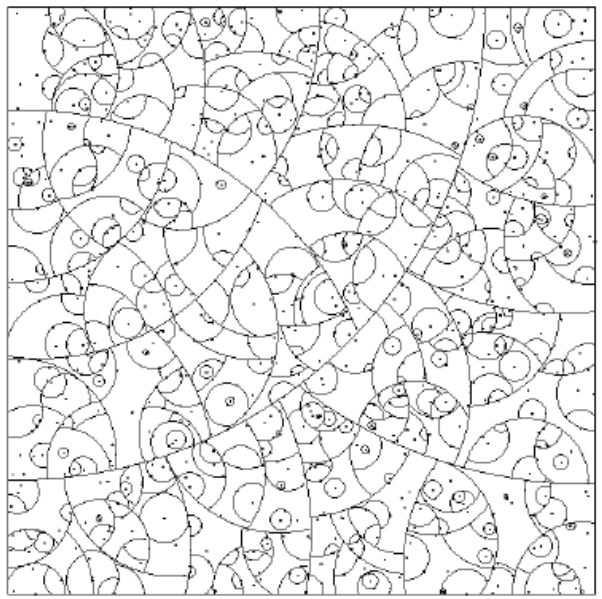


Figure: **Metric tree for a square domain (A)** One step **(B)** Full tree

Metric tree: construction

- Recursively construction:
  - Choose a pivot, ideally inducing a partition into subsets of the same size
  - Assign points to subtrees and recurse
  - Complexity under the balanced subtrees assumption: $O(n \log n)$.
- Algorithm build_MetricTree(S)

procedure `build_MetricTree`(S)
  if $S = \emptyset$ **then**
    return NIL
  $n \leftarrow newNode$
  Draw at random $Q \subset S$ and $v \in Q$
  $n.pivot \leftarrow v$
  $\mu \leftarrow \text{median}(\{d(v,p), p \in Q \backslash \{v\}\})$
  ▷ *The pivot splits points into two subsets* ◁
  $L \leftarrow \{s \in S \backslash \{p\} | d(s,v) < \mu\}$
  $R \leftarrow \{s \in S \backslash \{p\} | d(s,v) \geq \mu\}$
  ▷ *For each subtree: min/max distances to points in that subtree* ◁
  $n.(d_1, d_2) \leftarrow$ (min,max) of distances $d(v,p), p \in L$
  $n.(d_3, d_4) \leftarrow$ (min,max) of distances $d(v,p), p \in R$
  ▷ *Recursion* ◁
  $n.L \leftarrow$ build_MetricTree(L)
  $n.R \leftarrow$ build_MetricTree(R)

Searching a metric tree: algorithm

procedure SEARCH_METRICTREE((T, q))

▷ *Node of T is denoted n* ◁

$nn(q) \leftarrow \emptyset$

$\tau \leftarrow \infty$

if $n = NIL$ **then**

  return

▷ *Check whether the pivot is the nn*

$l \leftarrow d(q, n.pivot)$

if $l < \tau$ **then**

  $nn(q) \leftarrow n.pivot$

  $\tau \leftarrow l$

▷ *Dilate the distance intervals for left and right subtrees*

$I_l \leftarrow [n.d_1 - \tau, n.d_2 + \tau]$

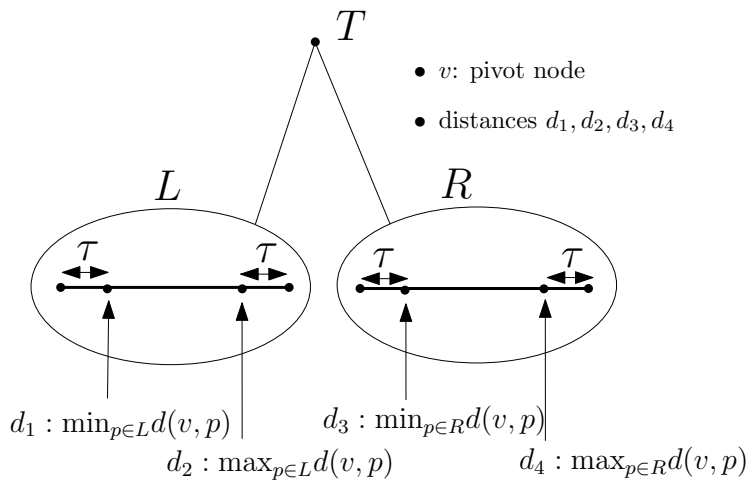
$I_r \leftarrow [n.d_3 - \tau, n.d_4 + \tau]$

if $l \in I_l$ **then**

  search_MetricTree$(n.L, q)$

if $l \in I_r$ **then**

  search_MetricTree$(n.R, q)$



Searching a metric tree: correctness – pruning lemma

Lemma 3. Consider the intervals associated with a node, as defined in Algorithm **??**, that is $I_l \leftarrow [n.d_1 - \tau, n.d_2 + \tau]$ $I_r \leftarrow [n.d_3 - \tau, n.d_4 + \tau]$. Then:
(1) If $l \notin I_l$, the left subtree can be pruned.
(2) If $l \notin I_r$, the left subtree can be pruned.

Proof.
We prove (1), as condition (2) is equivalent. Let us denote $I_L = [d_1, d_2]$. Since $l = d(v, q) \notin I_l$, we have $d(v, q) < d_1 - \tau$ and $d(v, q) > d_2 + \tau$. We analyze these two conditions in turn.

▷ Condition on the right hand side. By definition of d_2, with v the pivot, we have:

$$\forall p \in L : d(v, q) > d(v, p) + \tau.$$

Using the triangle inequality for $d(v, q)$ yields

$$d(v, p) + d(p, q) \geq d(v, q) > d(v, p) + \tau \Rightarrow d(q, p) > \tau.$$

▷ Mutatis mutandis. □

Metric tree: choosing the pivot

▷ By the pruning lemma: for small τ and if q is picked uniformly at random, the measure of the boundary of the spheres of radius $d_1, \ldots, d_4$ determines the probability that no pruning takes place.
$\Rightarrow$ pick the pivot so as to minimize this measure.

▷ Example in 2D: 3 choices for the pivot, so as to split the unit square (mass: 1) into two regions of equal size (mass: $1/2$)

▷ Choice of pivots (illustrated using μ (rather than the d_is):

- Best pivot: p_c
- Worst pivot: p_m

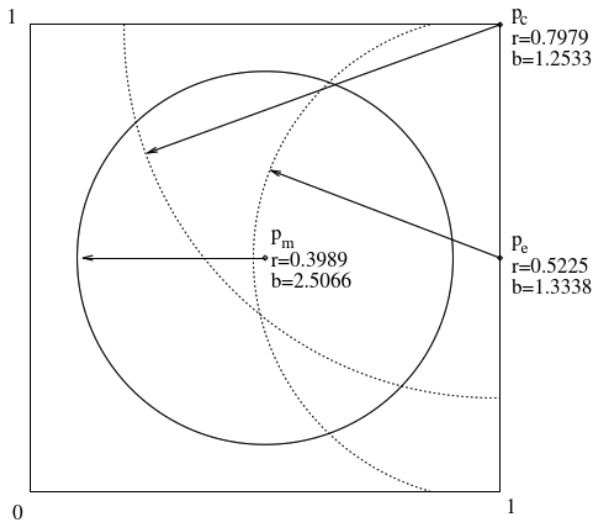


Figure: **Metric trees: minimizing the measure of boundaries.**

From metric trees to metric forests

- Search options:
  - (I) The exact search, based on the pruning lemma.
  - (II)The defeatist style search: visit one subtree only
- Compromising speed versus accurary
  - (I) Exact, but possibly costly if little/no pruning occurs. Worst-case: linear time.
  - (II) Faster, but error prone.
  - Compromise: using a forest of trees rescues erroneous branching decisions in the course of the defeatist search.

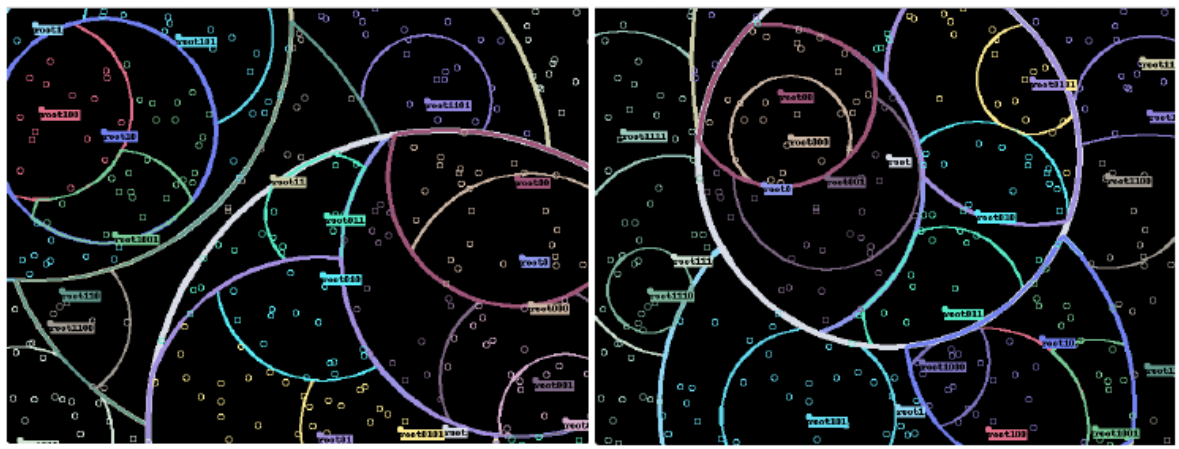


Figure: **Metric forest**

References

AMN+98 S. Arya et al. An optimal algorithm for approximate nearest neighbor searching fixed dimensions. Journal of the ACM (JACM), 45(6):891–923, 1998.

ML09 Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In VISAPP (1), pages 331–340, 2009.

MSMO03 Francisco Moreno-Seco, Luisa Mico, and Jose Oncina. A modification of the laesa algorithm for approximated k-nn classification. Pattern Recognition Letters, 24(1):47–53, 2003.

OD13 S. O'Hara and B.A. Draper. Are you using the right approximate nearest neighbor algorithm? In Applications of Computer Vision (WACV), 2013 IEEE Workshop on, pages 9–14. IEEE, 2013.

Yia93 Peter N Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In SODA, volume 93, pages 311-321, 1993.

Algorithms

PART 1: Nearest neighbors: data structures
PART 2: Nearest neighbors: analysis
PART 3: Concentration phenomena: introduction

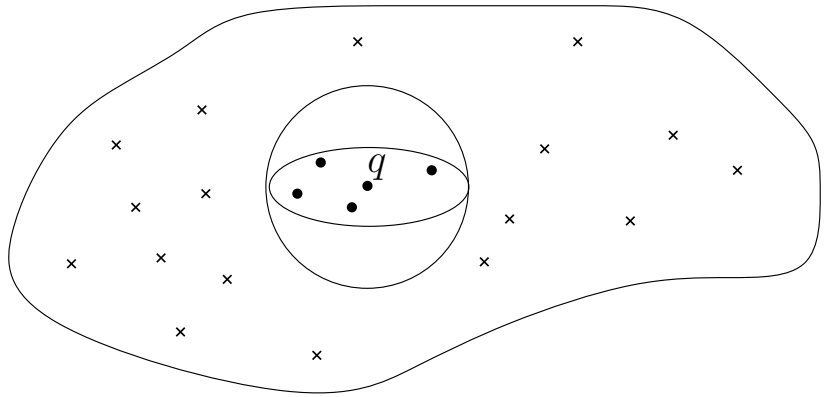
Algorithms

Intrinsic dimension?

Selected experiments on NN, regression, dimension estimation

RPTrees: search performance analysis

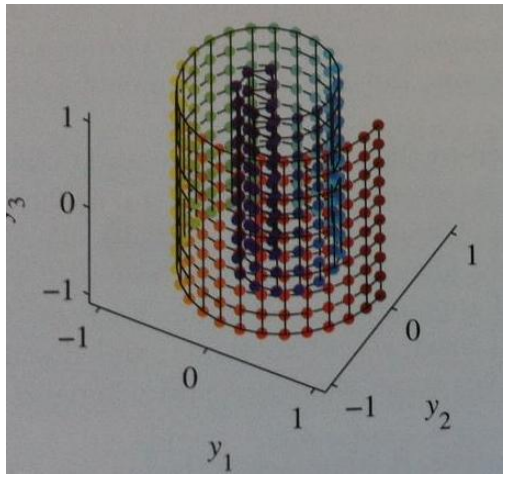
Nearest neighbors: on the importance of locality



- Typical settings:
  - Regression – estimating a response variable from neighbors
  - Supervised classification using neighbors
  - Manifold / shape learning: learning a mathematical model for the data (e.g. simplicial complex)
- Samples used at a given location q:
  - nearest neighbors
  - points in a cell of a spatial partition e.g. a RPTree

Intermezzo: data and their intrinsic dimension (I)

▷ Intrinsic dimension: in many real world problems, features may be correlated, redundant, causing data to have low intrinsic dimension, i.e., data lies close to a low-dimensional manifold



▷ Example: binary ie B&W image

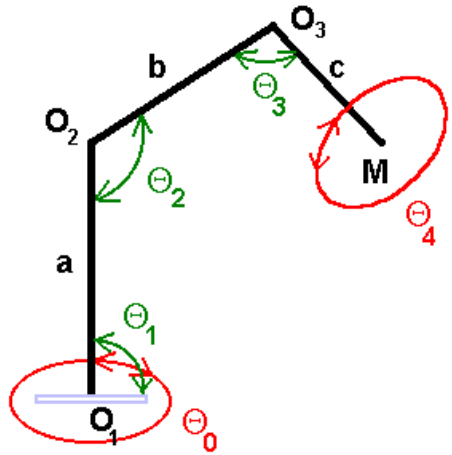
- Consider an $n \times n$ binary image: image $\sim$ point on the hypercube of dimension n^2

▷ Example: rotating an image

- Consider an $n \times n$ pixel image, with each pixel encode in the RGB channels: 1 image $\sim$ on point in dimension $d = 3n^2$.
- Consider N rotated versions of this image: N point in $\mathbb{R}^{3n^2}$
- But these points intrinsically have one degree of freedom (that of the rotation)

Intermezzo: data and their intrinsic dimension (II)

- Example: 2D robotic arm with 3 d.o.f.



- Example: human body motion capture
  - N markers attached to body (typically N=100).
  - each marker measures position in 3 dimensions, 3N dimensional feature space.
  - But motion is constrained by a dozen-or-so joints and angles in the human body.

▷Ref: Verma et al. Which spatial partitions are adaptive to intrinsic dimension? UAI 2009

Formal notions of intrinsic dimension

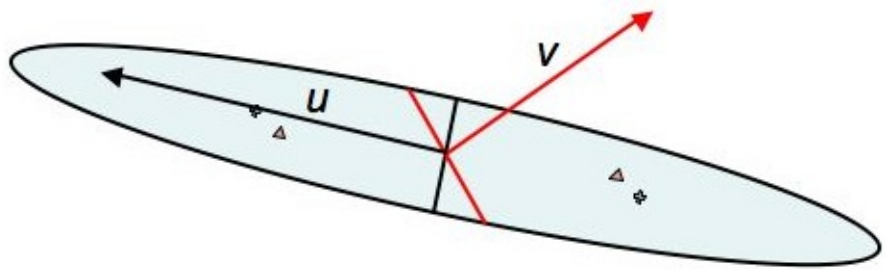
- Natural ones:
  - Affine dimension
  - Manifold dimension
- Requiring (elaborate) calculations:
  - (Local) covariance dimension
  - Assouad - doubling dimension

Local covariance dimension and its multi-scale estimation

▷ Def.: a set $T \subset \mathbb{R}^D$ has covariance dimension (d, ϵ) if the largest d eigenvalues of its covariance matrix satisfy

$$\sigma_1^2 + \cdots + \sigma_d^2 \geq (1 - \epsilon) \cdot (\sigma_1^2 + \cdots + \sigma_D^2).$$

▷ Def.: Local covariance dimension with parameters (d, ϵ, r): the previous must hold when restricting T to balls of radius r.

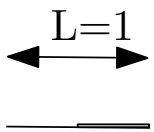


▷ Multi-scale estimation from a point cloud P:

For each datapoint p and each scale r
  Collect samples in $B(x, r)$
  Compute covariance matrix
  Check how many eigenvalues are required: yields the dimension

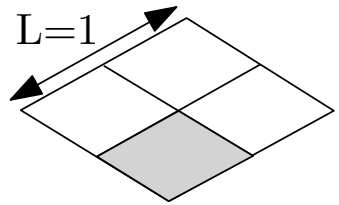
Assouad / doubling dimension: intuition

▷ Pick a cube of side length L: count how many cubes of side length $L/2$ are needed to cover it



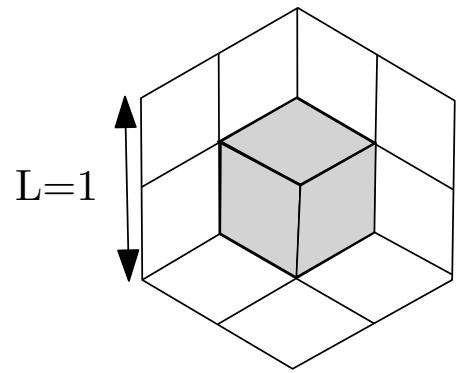
D=1

$N = 2^1$



D=2

$N = 2^2$



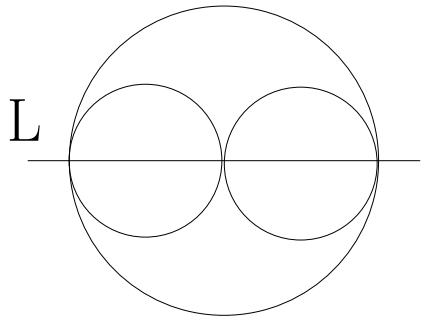
D=3

$N = 2^3$

$\Rightarrow$ take the log of the number of cubes

Assouad dimension

▷ Def: Set $S \subset \mathbb{R}^D$ has Assouad dimension $\leq d$: for any ball B, subset $S \cap B$ can be covered by 2^d balls of half the radius. Also called doubling dimension.



▷ Examples:

- S = line: Assouad dimension = 1
- S = k-dimensional affine subspace: Assouad dimension = O(k)
- Union of D intervals $[-1, 1]$ in $\mathbb{R}^D$; dim is $\log 2D$
- S = k-dim submanifold of $\mathbb{R}^D$ with finite condition number: Assouad dimension = O(k) in small enough neighborhoods
- S = set of N points: Assouad dimension $\leq logN$

▷ Hardness: computing doubling dimensions and constants is generally hard: related to packing problems.

Generalization: doubling dimension and doubling measures

▷ Def.: A metric space X with metric is called doubling if there exists $M(X) \in \mathbb{N}$ so that any closed ball $B(x, r)$ can be covered by at most M balls of radius $r/2$. The doubling dimension is $\log_2 M$.

▷ Def.: A measure μ on a metric space X is called doubling if $\exists C > 0$ such that $\forall x \in X$ and $r > 0$

$$\mu(B(x, 2r) \leq C\mu(B(x, r)).$$

The dimension of the doubling measure satisfies $d_0 = \log_2 C$.

▷ Remarks:

- A metric space supporting a doubling measure is necessarily a doubling metric space, with dimension depending on C.
- Conversely, any complete doubling metric space supports a doubling measure.

Algorithms

Intrinsic dimension?

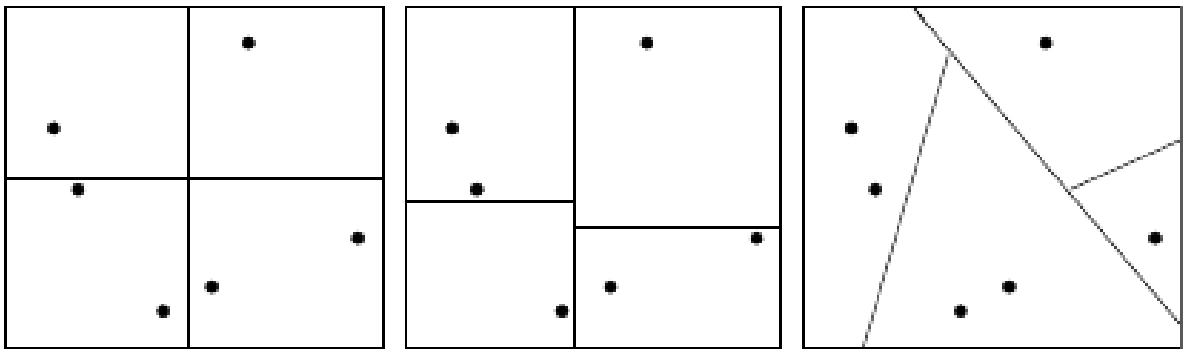
Selected experiments on NN, regression, dimension estimation

RPTrees: search performance analysis

Empirical results: contenders

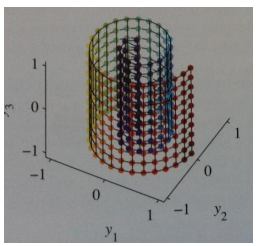
- Contenders / algorithms:
  - dyadic trees aka tries: pick a direction and split at the midpoint; cycle through coordinates.
  - kd-tree: split at median along direction with largest spread.
  - random projection trees: split at the median along a random direction.
  - PD / PCA trees: split at the median along the principal eigenvector of the covariance matrix.
  - two means trees: solve the 2-means; pick the direction spanned by the centroids, and split the data as per cluster assignment.

- dyadic trees, kd-trees, RP trees

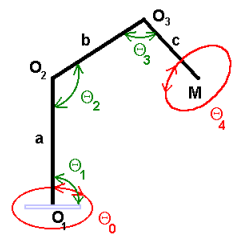


Real word datasets

- Datasets:
  - Swiss roll
  - Teapot dataset: rotated images of a teapot (1 B&W image: 50x30 pixels); thus, 1D dataset in ambient dimension 1500.
  - Robotic arm: dataset in $\mathbb{R}^{12}$; yet, robotic arm has 2 joints: (noisy) 2D dataset in ambient dimension 12.
  - 1 from the MNIST OCR dataset; 20x20 B&W images, i.e. points in ambient dimension 400.
  - Love cluster from Australian Sign Language time-seris
  - aw phoneme from MFCC TIMIT dataset



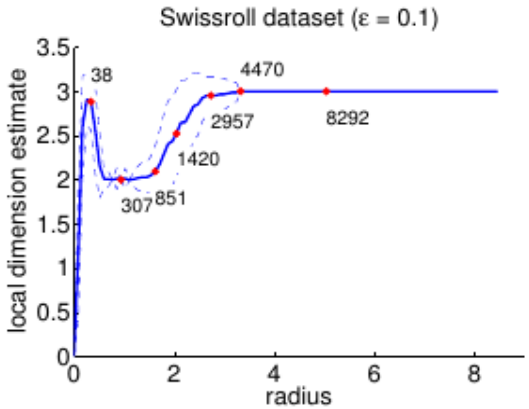


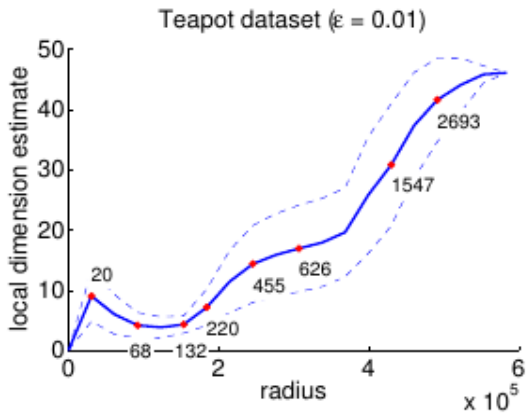


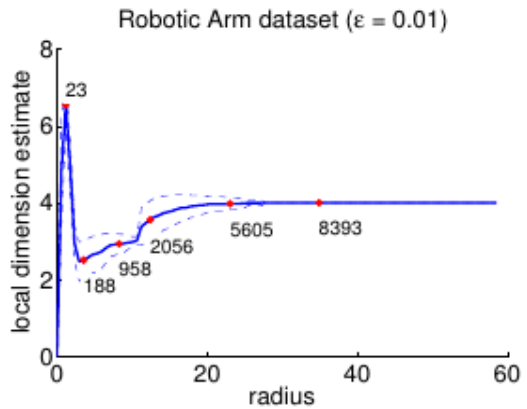
- Ref: Verma, Kpotufe, and Dasgupta, UAI 2009.

Empirical results: local covariance dimension estimation

▷ Conventions: bold lines: estimate $d(r)$; dashed lines: std dev; numbers: ave. over samples in balls of the given radius







▷ Observations:

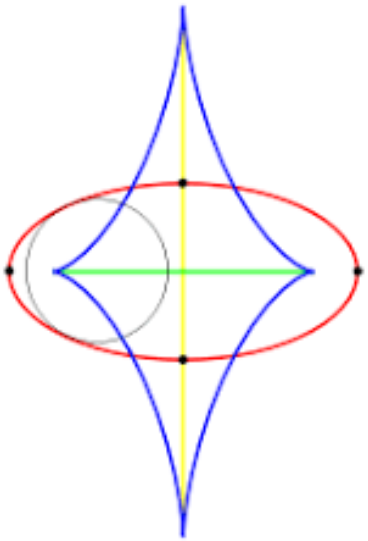
- Swiss roll (ambient space dim is 3): failure at small (noise dominates) and large scales (sheets get blended).
- Teapot: clear small dimensional structure at low scale, but rather 3-4 than 1.
- Robotic arm: tiny spot (r values) to get the correct dimension. . . noise.

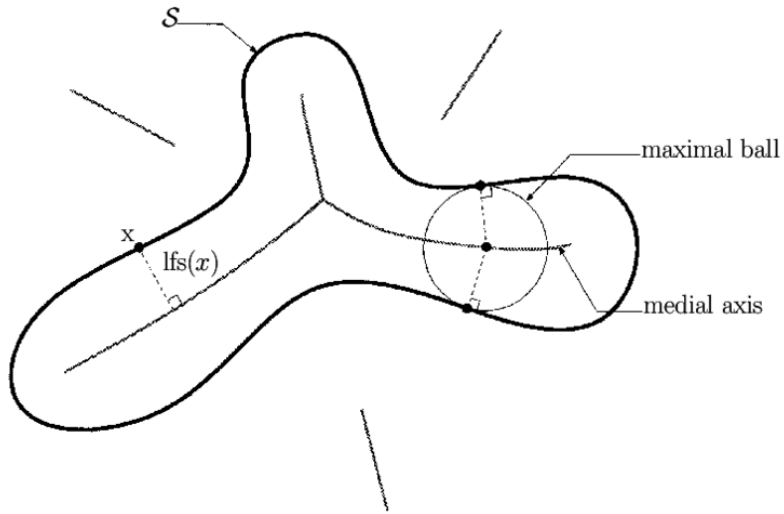
▷Ref: Verma, Kpotufe, and Dasgupta, UAI, 2009

Intermezzo: medial axis of an open set

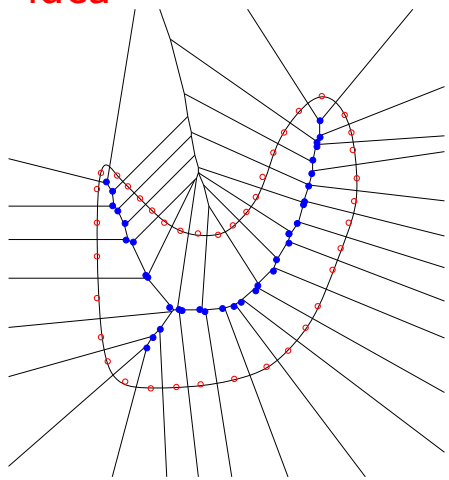
Using local neighborhoods / topological disks

- Def.:



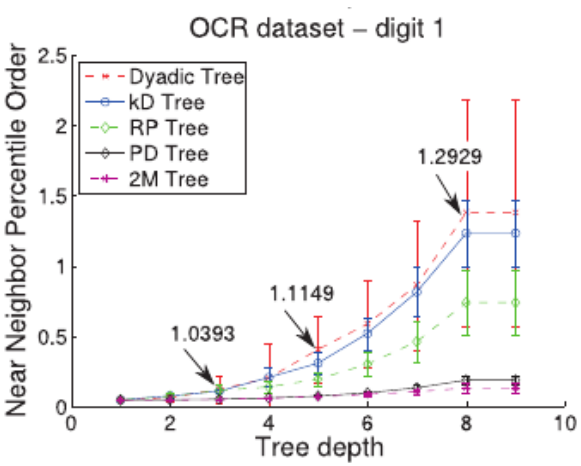


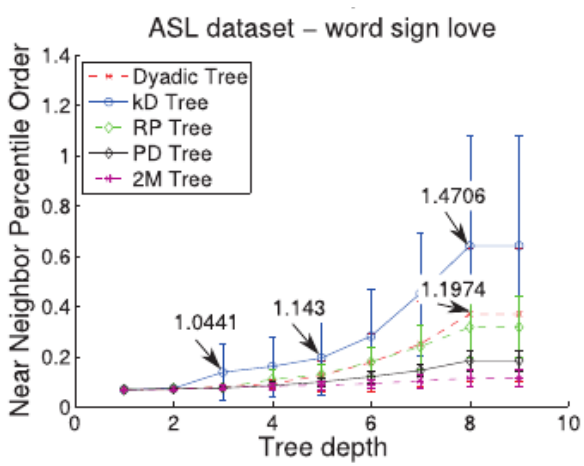
- Construction from Voronoi: idea

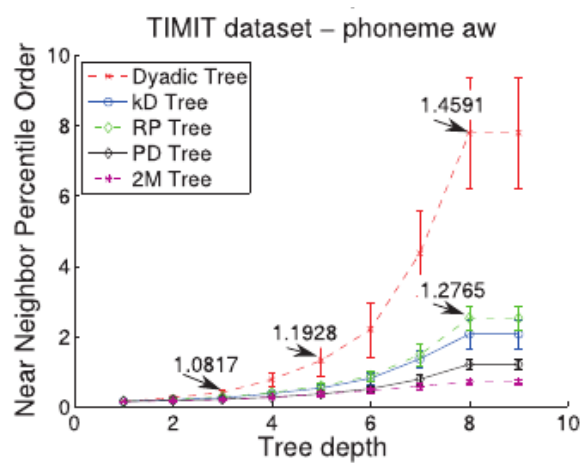


Empirical results: performance for NN searches

- Searching $p_{(1)}$: performance is the order of the NN found / dataset size
  - percentile order: order of NN found / dataset size (the smaller the better; max is 100%)
  - tree depth: NN sought at each level in the tree
  - decorating numbers: distance ratio $\|q - nn(q)\| / \left\|q - p_{(1)}\right\|$







- Observations:
  - percentile order deteriorates with depth – separation does occur
  - yet, the distance ratio remains small even at high percentile orders
  - 2M and PD (i.e. PCA trees) consistently yield better nearest neighbors: better adaptation to the intrinsic dimension

- Ref: Verma, Kpotufe, and Dasgupta, UAI, 2009

References

- Dasgupta, Sanjoy, and Yoav Freund. Random projection trees and low dimensional manifolds. Proceedings of the fortieth annual ACM symposium on Theory of computing. ACM, 2008.
- Verma, Nakul, Samory Kpotufe, and Sanjoy Dasgupta. Which spatial partition trees are adaptive to intrinsic dimension?. Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence. AUAI Press, 2009.
- J. Heinonen, Lectures on analysis on metric spaces, Springer, 2001.

Algorithms

Intrinsic dimension?

Selected experiments on NN, regression, dimension estimation

RPTrees: search performance analysis

Random projection trees and nearest neighbors

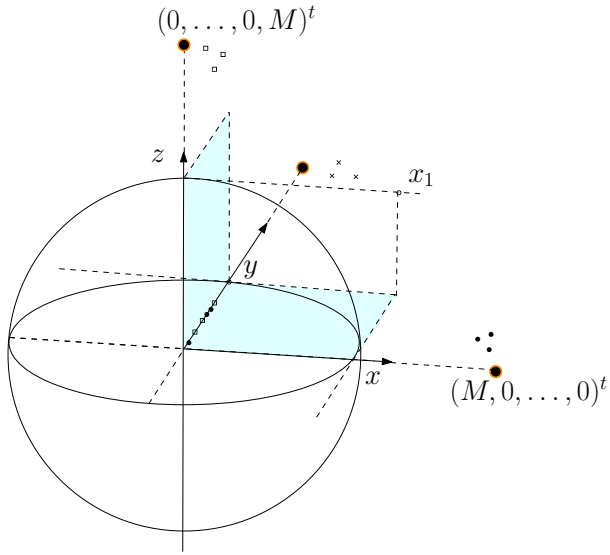
- Recap:
  - Points iteratively projected on random directions
  - Risks jeopardizing the search strategy: points far away (from the NN) squeeze in-between q and $nn(q)$
  - Hardness of the NN search: function Φ

$$\Phi(q, P) = \frac{1}{n} \sum_{i=2}^{n} \frac{\left\| q - x_{(1)} \right\|}{\left\| q - x_{(i)} \right\|}. \tag{10}$$

Projections on random directions is needed

as Projections on coord. axis fails even for cases of $\Phi \sim 0$

▷ Idea: with $q = (0, \ldots, 0)^T$, generate a DB of neighbors such that a kd-tree will always separate q from its NN which is x_1



▷ Consider the following point set $\{x_1, \ldots, x_n\}$:

- $x_1 = (1, \ldots, 1)^T$
- For each $x_i, i > 1$: pick a random coord and set it to a large value M; set the remaining coords to uniform random numbers is $(0, 1)$

▷ Query point q: the origin

▷ kd-trees separate q and x_1, even though function Φ is arbitrarily small:

- The NN of q (=origin) is x_1
- But by growing M, function Φ gets close to 0 $\Rightarrow$ random projections will work well
- However, any coord. projection separates q and x_1: on average, the fraction of points falling in-between q and x_1 is arbitrarily large:

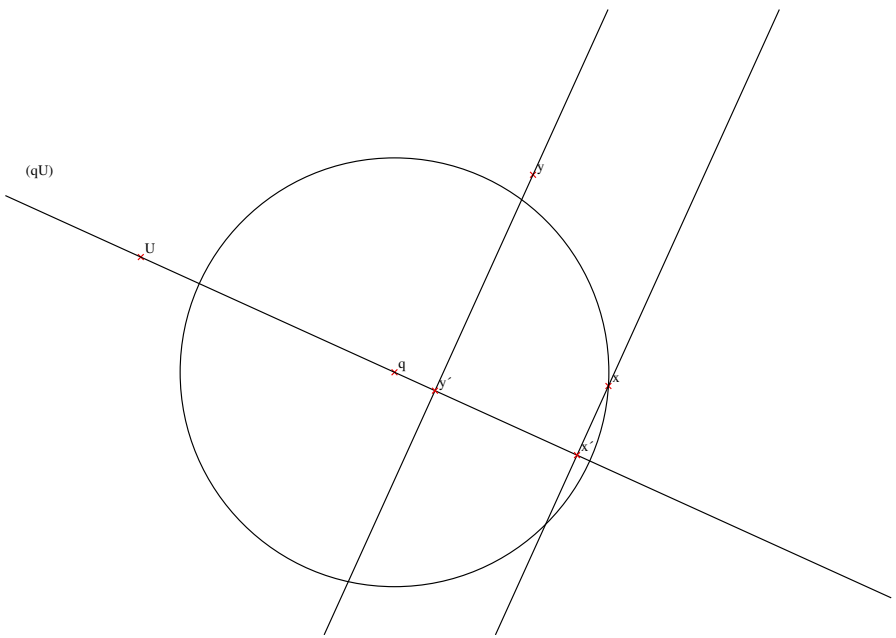
$$\frac{1}{n}(n - \frac{n}{d}) = 1 - \frac{1}{d}$$

▷ Coming next: RPTrees work well in this case; randomness is needed.

Demo with DrGeo

Compulsory tools for geometers

- In the sequel: Consider 3 points q, x, y with $\|q - x\| \leq \|q - y\|$.
- In projection on a random direction U: probability to have the projection of y nearest to q than the projection of x?
- DrGeo: `http://www.drgeo.eu/`



- Event E to avoid: $\langle y, U \rangle$ falls strictly in-between $\langle q, U \rangle$ and $\langle x, U \rangle$

- NB: also of interest: IPE, `http://ipe.otfried.org/`

Random projections: relative position of three points

- In the sequel: q, x, y: 3 points with $\|q - x\| \leq \|q - y\|$
- Colinearity index q, x, y:

$$\text{coll}(q, x, y) = \frac{\langle q - x, y - x \rangle}{\|q - x\| \, \|y - x\|} \tag{11}$$

- Event E: $\langle y, U \rangle$ falls strictly in-between $\langle q, U \rangle$ and $\langle x, U \rangle$

Lemma 4. Consider $q, x, y \in \mathbb{R}^d$ and $\|q - x\| \leq \|q - y\|$. The proba. over random directions U, of E, satisfies:

$$\mathbb{P}[E] = \frac{1}{\pi} \arcsin\left(\frac{\|q - x\|}{\|q - y\|} \sqrt{1 - \text{coll}(q, x, y)^2} \right) \tag{12}$$

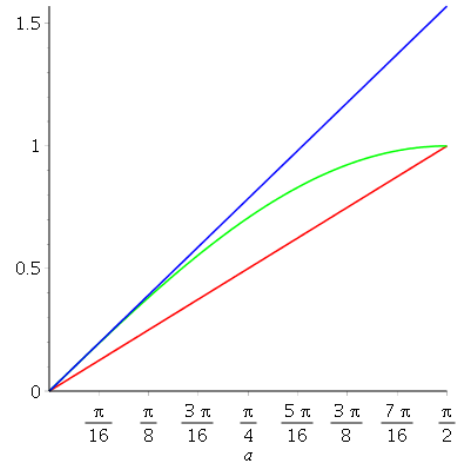
Corollary 5.

$$\frac{1}{\pi} \frac{\|q - x\|}{\|q - y\|} \sqrt{1 - \text{coll}(q, x, y)^2} \leq \mathbb{P}[E] \leq \frac{1}{2} \frac{\|q - x\|}{\|q - y\|} \tag{13}$$

Proof of the corollary

▷ Using the Inequality:

$$\theta \in [0, \pi/2] : \frac{2\theta}{\pi} \leq \sin\theta \leq \theta \quad (14)$$



▷ Lower bound of the corr.: from the upper bound of Eq. (14): $\theta \leq \arcsin\theta$ applied to $\mathbb{P}[E]$

▷ Upper bound of the corr.:

First note that:

$$\frac{\|q-x\|}{\|q-y\|}\sqrt{1-\text{coll}(q,x,y)^2} \leq \frac{\|q-x\|}{\|q-y\|}$$

Then, apply $(2\phi/\pi) \leq \phi$ to $\phi = \arcsin\|q-x\| / \|q-y\|$.

Random projections: separation of neighbors

▷ Recall that for $m \geq 1$

$$\Phi_m(q, P) = \frac{1}{m} \sum_{i=2}^{m} \frac{\|q - p_{(1)}\|}{\|q - p_{(i)}\|}. \tag{15}$$

Theorem 6. Consider $q, p_1, \ldots, p_n \in \mathbb{R}^d$, and a random direction U.
The expected fraction of the projected p_i that fall between q and $p_{(1)}$ is at most

$$\frac{1}{2}\Phi(q, P).$$

▷ Proof. Let Z_i be the event : "$p_{(i)}$ falls between q and $p_{(1)}$ in the projection" . By the corollary 5, $\mathbb{P}[Z_i] \leq (1/2)\|q - p_{(1)}\| / \|q - p_{(i)}\|$. Then, apply the linearity of expectation to $\sum Z_i/n$ (divide by n to get the fraction).

Theorem 7. Let $S \subset P$ with $p_{(1)} \in S$. If U is chosen uniformly at random, then for any $0 < \alpha < 1$, the proba. (over U) that a fraction $\geq \alpha$ of the projected points in S fall between q and $p_{(1)}$ is

$$\leq \frac{1}{2\alpha}\Phi_{|S|}(q, P).$$

▷ Proof. Φ is maximized when S consists of the points closest to q. Then, previous Thm + Markov's inequality.

Random projection trees

▷ Recap:

- Pick a random direction and project points onto it
- Split at the β fractile for $\beta \in (1/4, 3/4)$
- Storage: each point mapped to a single leaf
- Query routing: query point mapped to a single leaf too

Theorem 8. Consider an RP tree for P. Define $\beta = 3/4$, and $l = \log_{1/\beta}(n/n_0)$. One has:

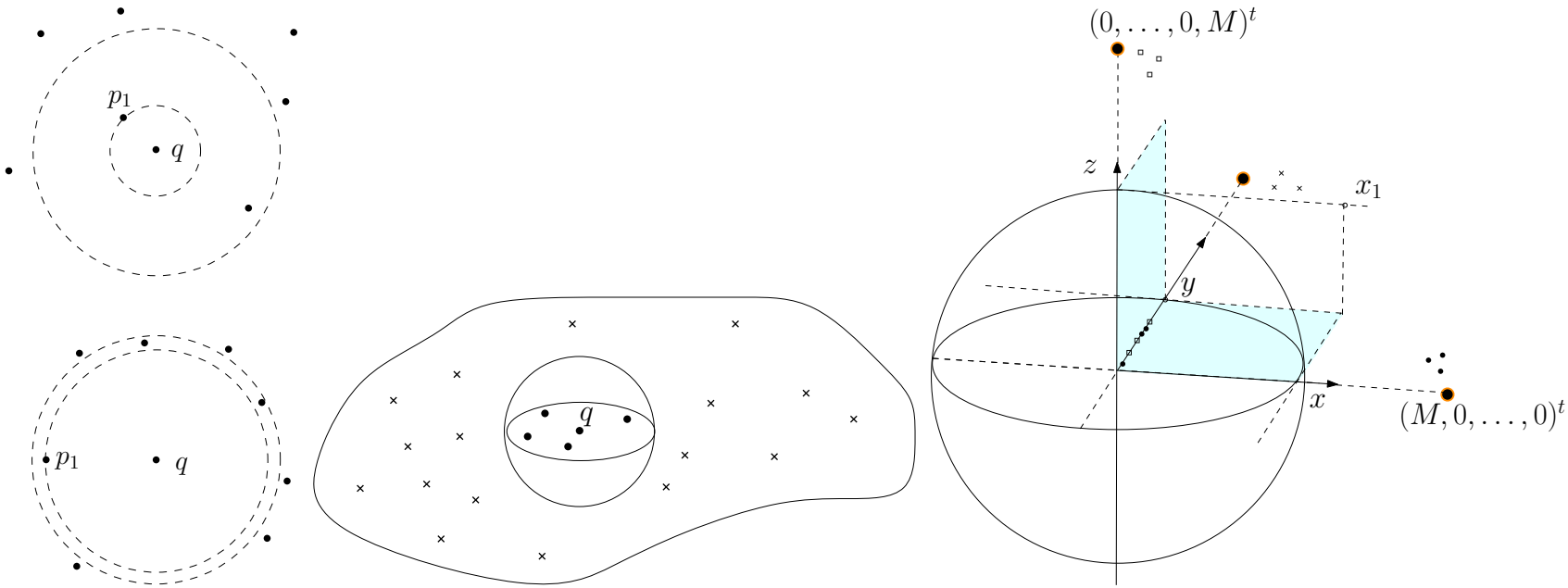
$$\mathbb{P}\left[\text{NN query does not return } p_{(1)}\right] \leq \sum_{i=0,\ldots,l} \Phi_{\beta^i n} \ln \frac{2e}{\Phi_{\beta^i n}} \tag{16}$$

▷ Proof, key steps:

- $F \in \{0, 1/2, \ldots, (m-1)/m\}$: fraction of points falling in-between q and $p_{(1)}$ in projection
- Since split chosen at random in interval of mass $1/2$: it separates q and $p_{(1)}$ is at most $F/(1/2)$. (Indeed: assume any value in the interval of width F is eligible.)
- Summing on values of F yields the result for one level; then, union bound.

Error bound depends on Φ?

- Φ qualifies the hardness of the query situations
- Focus: pathological cases versus settings with some regularity



Bounding function Φ in specific settings

▷ Perspective: assume that $x_1, \ldots, x_n$ are drawn i.i.d. from a doubling measure. Can this regularity be used?

Theorem 9. Let μ be a continuous measure on $\mathbb{R}^d$, a doubling measure of dimension $d_0 \geq 2$. Assume $p_1, \ldots, p_n \sim \mu$. Let $0 < \delta < 1/2$.
With probability $\geq 1 - 3\delta$:

$$\forall m \in [2, n] : \quad \Phi_m(q, P) \leq 6\left(\frac{2}{m} \ln \frac{1}{\delta}\right)^{1/d_0}$$

Theorem 10. Under the same hypothesis, with k the num. of NN sought:
– For both variants of the spill trees:

$$\mathbb{P}\left[Err\right] \leq \frac{c_o k d_o}{\alpha} \left(\frac{8 \max(k, \ln 1/\delta)}{n_0}\right)^{1/d_0}$$

– For random projection trees with $n_0 \geq c_0 (3k)^{d_0} \max(k, \ln 1/\delta)$:

$$\mathbb{P}\left[Err\right] \leq c_o k (d_o + \ln n_0) \left(\frac{8 \max(k, \ln 1/\delta)}{n_0}\right)^{1/d_0}$$

▷ Rmk:

- failure proba. can be made arbitrarily small by increasing the leaf size n_0
- The failure proba increases with d_0

References

DS13 S. Dasgupta and K. Sinha. Randomized partition trees for exact nearest neighbor search. JMLR: Workshop and Conference Proceedings, 30:1–21, 2013.

V12 S. Vempala. Randomly-oriented kd Trees Adapt to Intrinsic Dimension. FSTTCS, 2012.

VKD09 N. Verma, S. Kpotufe, S. Dasgupta, Which spatial partitions are adaptive to intrinsic dimension? UAI 2009.

Algorithms

PART 1: Nearest neighbors: data structures
PART 2: Nearest neighbors: analysis
PART 3: Concentration phenomena: introduction

Algorithms

Concentration phenomena: application to nearest neighbor searches

Concentration phenomena: key properties

p-norms and Unit Balls

- Notations:
  - d: the dimension of the space
  - $\mathcal{F}$: a 1d distribution
  - $X = (X_1, \ldots, X_d)$ a random vector such that $X_i \sim \mathcal{F}$
  - $P = \{p^{(j)}\}$: a collection on n iid realizations of X
- Generalizations of L_p norms, $p > 0$:

$$\|X\|_p = (\sum_i^i \mid X_i \mid^p)^{1/p} \qquad (17)$$

Unit balls: see plots

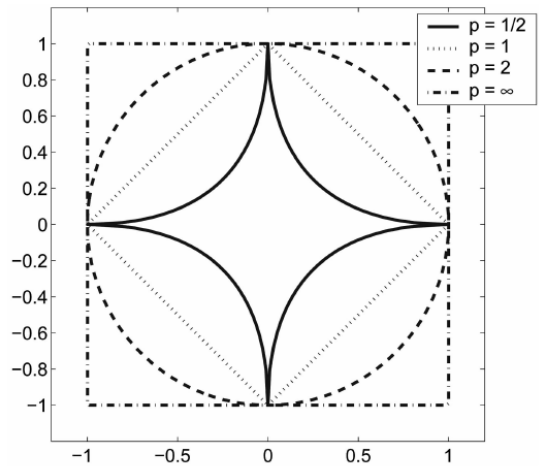


Fig. 2. Two-dimensional unit balls for several values of the parameter of the p-norm.

- Cases of interest in the sequel:
  - Minkowski norms: p, an integer $p \geq 1$:
  - fractional p-norms: $0 < p < 1$. NB: triangle inequality not respected; NB: balls not convex for $p < 1$. sometimes called pre-norms.
- Study the variation of $\|\|\|_p$ as a function of d

Concentration of the Euclidean norm: Observations

▷ Plotting the variation of the following for random points in $[0,1]^d$:

$$\min \|\|^2, \quad \mathbb{E}\left[\|\|^2\right] - \sigma\left[\|\|^2\right], \quad \mathbb{E}\left[\|\|^2\right], \mathbb{E}\left[\|\|^2\right] + \sigma\left[\|\|^2\right], \quad \max \|\|^2, M = \sqrt{d} \tag{18}$$

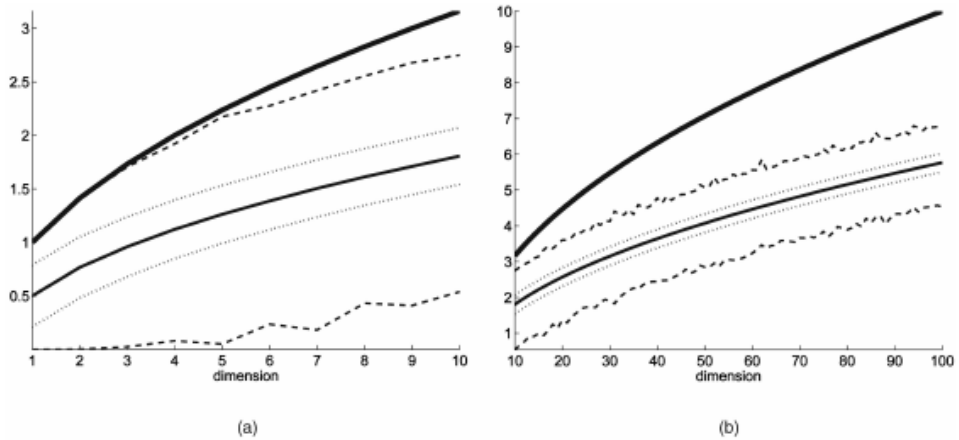


Fig. 1. From bottom to top: minimum observed value, average minus standard deviation, average value, average plus standard deviation, maximum observed value, and maximum possible value of the Euclidean norm of a random vector. The expectation grows, but the variance remains constant. A small subinterval of the domain of the norm is reached in practice.

▷ Observation:

- The average value increases with the dimension d
- The standard deviation seems to be constant; likewise for the min-max values
- For $d \leq 10$ i.e. d small: the min and max values are close to the bounds: lower bound is 0, upper bound is $M = \sqrt{d}$
- For d large say $d \geq 10$, the norm concentrates within a small portion of the domain; the gap wrt the bounds widens when d increases.

Concentration of the Euclidean Norm: Theorem

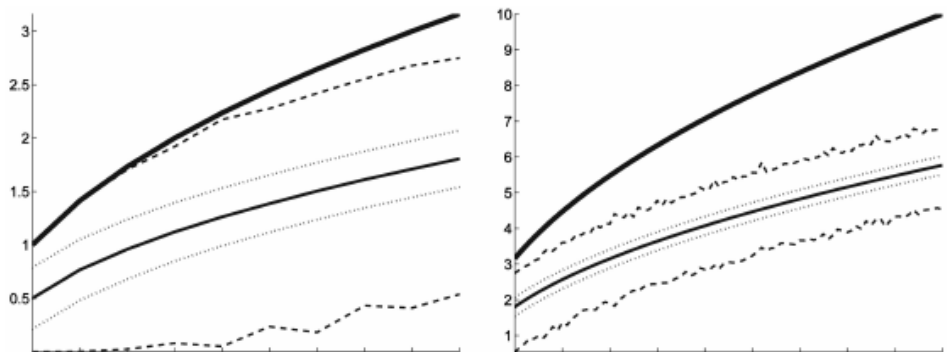
Theorem 11. Let $X \in \mathbb{R}^d$ be a random vector with iid components $X_i \sim \mathcal{F}$. There exist constants a and b that do not depend on the dimension (they depend on $\mathcal{F}$), such that:

$$\mathbb{E}\left[\|X\|^2\right] = \sqrt{ad - b} + O(1/d) \tag{19}$$

$$\mathrm{Var}\left[\|X\|^2\right] = b + O(1/\sqrt{d}). \tag{20}$$

▷ Remarks:

- The variance is small wrt the expectation, see plot
- The error made in using $\mathbb{E}\left[\|X\|^2\right]$ instead of $\|X\|^2$ becomes negligible: it looks like points are on a sphere of radius $\mathbb{E}\left[\|X\|^2\right]$.
- The results generalize even if the X_i are not independent; then, d gets replaced by the number of degrees of freedom.



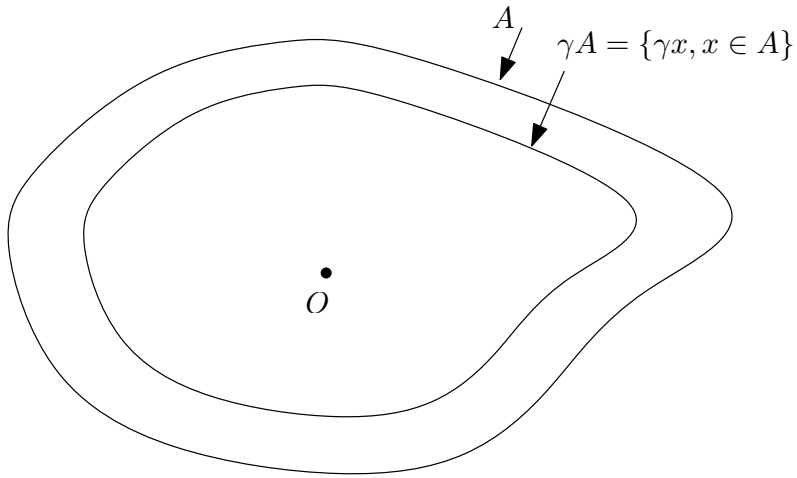
Algorithms

Concentration phenomena: application to nearest neighbor searches

Concentration phenomena: key properties

Geometry in high dimension: scaled bodies and their volume

▷ Scaling a body from $\mathbb{R}^d$:



▷ For $\gamma = 1 - \varepsilon$ [1]:

$$\frac{\texttt{Volume}((1-\varepsilon)A)}{\texttt{Volume}(A)} = (1-\varepsilon)^d \leq e^{-\varepsilon d}. \tag{21}$$

▷ Fix ε and let $d \to \infty$: the ratio tends to zero. That is: nearly all the volume of A belongs to the annulus of width ε.

[1] Use $e^{-x} \geq 1 - x$

Unit sphere: surface area and volume

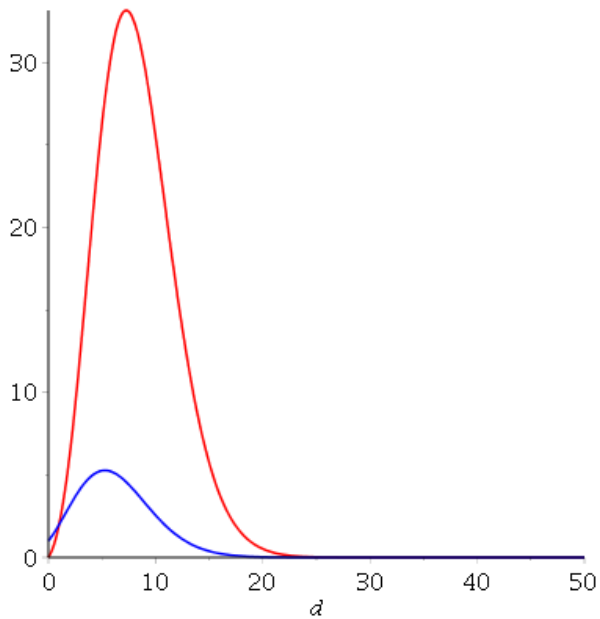
- The Gamma function Γ:

$$\Gamma(x) = \int_0^\infty s^{x-1} e^{-s} ds. \tag{22}$$

NB: for integers $\Gamma(n) = (n-1)!$

- The surface area and volume of the unit sphere S^d are given by:

$$A(d) = \frac{2\pi^{d/2}}{\Gamma(d/2)}, \quad V(d) = \frac{A(d)}{d}. \tag{23}$$

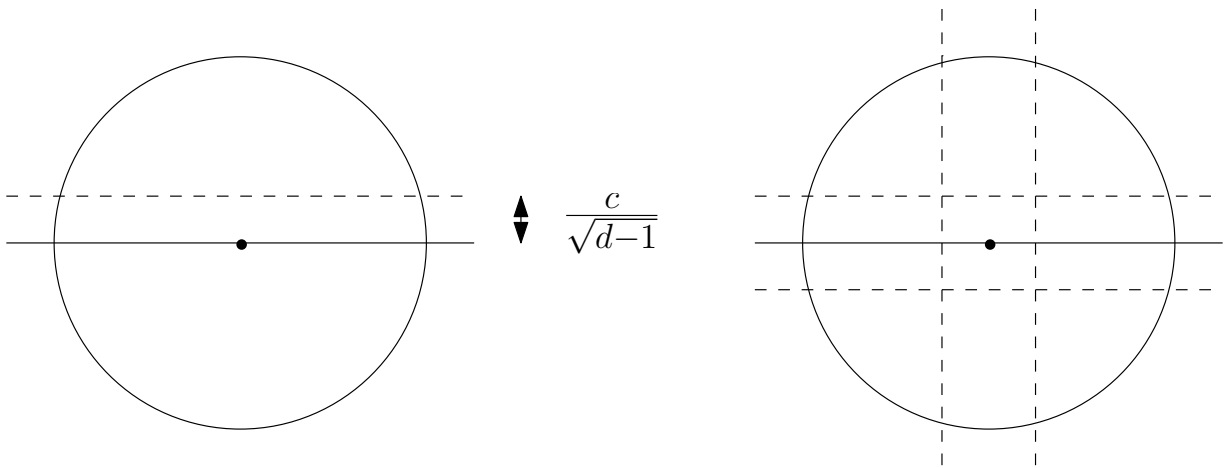


Variation of the surface area (red) and volume (blue) of the unit sphere, as a function of the dimension d

Unit ball: volume concentration near the equator

▷ Thm: (Slab Thm.) For $c \geq 1$ (slab width) and $d \geq 3$, at least a fraction $1 - \frac{2}{c}e^{-c^2/2}$ of the volume of the unit ball satisfies $|\, x_1 \,| \leq \frac{c}{\sqrt{d-1}}$.

▷ Corr: With $c = 2\sqrt{\ln d}$, a fraction at least $1 - O(\frac{1}{d}) \geq 1/2$ of the volume of the unit ball lies in a cube of half side length $c/\sqrt{d-1} = 2\sqrt{\ln d}/\sqrt{d-1}$.
Since the vol. of this cube $\to 0$, the volume of the unit ball goes to 0 when $d \to \infty$.



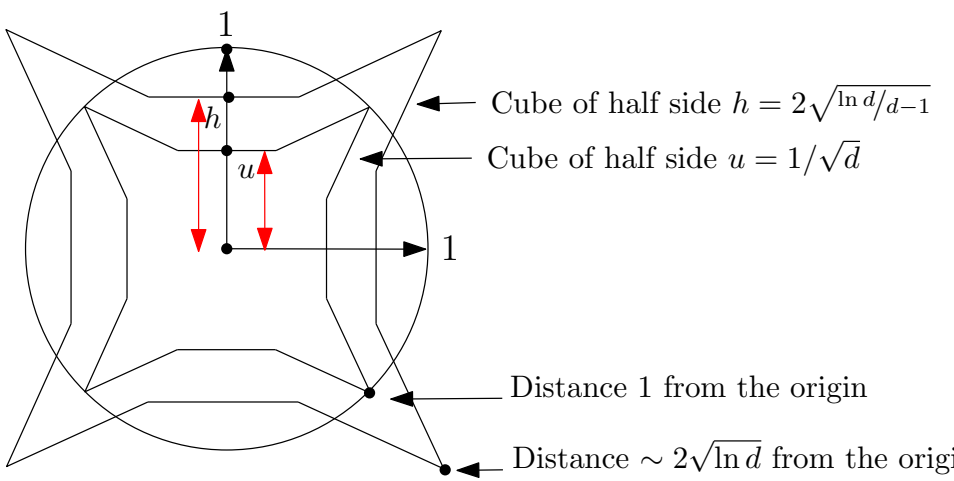
Proof: apply the Thm with $c = 2\sqrt{\ln d}$.

Nb: Vertices of the cube are outside the ball. This does not matter since the Thm integrates slices up to $c/\sqrt{d-1}$.

Unit ball:

are points near the surface of within a small cubic core?

- ▷ Apparent contradiction:
  - Argument from body scaling: mass located near the surface of the unit sphere
  - Previous argument: $\geq 1/2$ of the volume located near the equator, within a cube of side length $4\sqrt{\ln d/d-1}$
- ▷ Explanation:
  - cube whose vertices are on the unit sphere: half side $1/\sqrt{d}$
  - corners of the cube of half side length $h = 2\sqrt{\ln d/d-1}$ are at distance $\sim 2\sqrt{\ln d}$ from the origin. this cube covers a **significant** portion of the unit ball.



The cube of small side length h projects vertices far away from the unit sphere.

Random points are almost orthogonal with high probability

▷ Thm. Consider n points $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n\}$ drawn uniformly at random from the unit ball. The following holds with probability $1 - O(1/n)$:

1. $\mathbb{P}\left[\|\boldsymbol{x}_i\| \geq 1 - \frac{2 \ln n}{d}\right] \geq 1 - O(1/n), \forall i$
2. $\mathbb{P}\left[|\langle \boldsymbol{x}_i, \boldsymbol{x}_j \rangle| \leq \sqrt{\frac{6 \ln n}{d-1}}\right] \geq 1 - O(1/n), \forall i \neq j.$

▷ Discussion:

1. Points near the surface of the ball
2. Vectors associated with a pair of points are nearly orthogonal

Generating random points on S^{d-1} /inside S^{d-1}

- Generate a point $\boldsymbol{x} = (x_1, \ldots, x_d)^t$ whose coordinates are iid Gaussians:
  - Generate $x_1, \ldots, x_d$ iid Gaussian $\mathcal{N}(\mu = 0 \mid \sigma = 1)$
    - distribution is spherically symmetric (on a sphere of given radius).
    - random vector has arbitrary norm
  - The density of X is

$$f_G(\boldsymbol{x}) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{x_1^2 + x_2^2 + \cdots + x_d^2}{2}} = \frac{1}{(2\pi)^{d/2}} e^{-\|\boldsymbol{x}\|^2/2}. \tag{24}$$

  - To obtain a unit vector: $\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$. NB: its coordinates are not independent.
- Inside the unit ball: the point $\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|}$ needs to be scaled by a density $\rho(r) = dr^{d-1}$.

The Gaussian annulus theorem

for an isotropic d dimensional Gaussian

▷ Density of the isotropic Gaussian: Gaussian of zero mean and σ^2 along each dir.:

$$f_G(X) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{x_1^2+x_2^2+\cdots+x_d^2}{2}}. \tag{25}$$

▷ Expectation of $\|X\|^2$:

$$\mathbb{E}\left[\|X\|^2\right] = \mathbb{E}\left[\sum_{i=1,\ldots,d} x_i^2\right] = \sum_{i=1,\ldots,d} \mathbb{E}\left[x_i^2\right] = d\mathbb{E}\left[x_1^2\right] = d. \tag{26}$$

▷ Thm. Consider an isotropic d dimensional Gaussian with $\sigma = 1$ in each direction. For any $\beta \leq \sqrt{d}$, consider the annulus defined by

$$\mathcal{A} = \{X \text{ such that } \sqrt{d} - \beta \leq \|X\| \leq \sqrt{d} + \beta\}. \tag{27}$$

There exists a fixed positive constant $c(\sim 1/100)$ such that

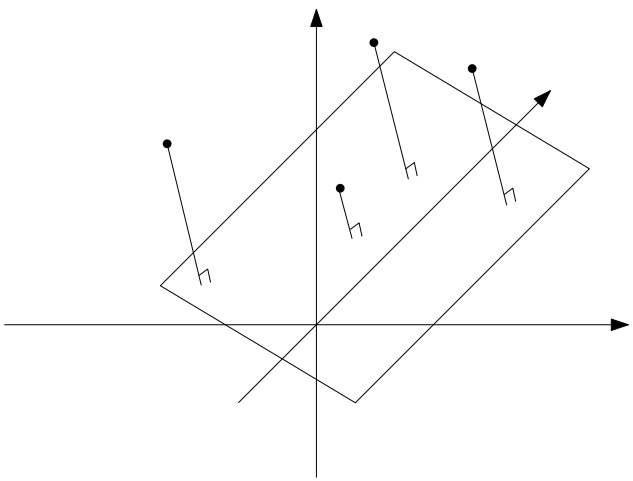
$$\mathbb{P}(\mathcal{A}^c) \leq 3e^{-c\beta^2}. \tag{28}$$

▷ Rmk: how come the mass concentrates around $\sqrt{d}$?

- Concentration thm: the mass concentrates near $\sqrt{\mathbb{E}\left[\|X\|^2\right]} = \sqrt{d}$
- The density f_G is max. at the origin; but integrating over the unit ball ... no mass since the volume of the unit ball tends to 0. (prop. seen earlier.)
- In going well beyond $\sqrt{d}$: the density f_G gets too small.

Projecting onto a (random) affine subspace

- k-dimensional affine subspace: matrix $R : d \times k$ whose vectors define an (orthonormal) basis
- To obtain such an orthonormal matrix R:
  - draw k (unit) random vectors (see above)
  - perform a Gram–Schmidt orthonormalization
    NB: the orthonormalization process complicates things, since entries of the matrix are no longer independent
- To get a randomized dimension-k matrix R – dim is $d \times k$):
  - Draw the $d \times k$ entries at random, using a the normal distribution (Gaussian with 0 mean and unit variance)
  - Then $f(\boldsymbol{v}) = (\boldsymbol{u}_1 \cdot \ \boldsymbol{v}, \boldsymbol{u}_2 \cdot \ \boldsymbol{v}, \ldots, \boldsymbol{u}_k \cdot \ \boldsymbol{v})^{\mathsf{T}}$



Projection $f(\boldsymbol{v})$ of a vector $\boldsymbol{v}$ onto a (random) affine space of dimension k, in matrix form:

$$f(\boldsymbol{v}) = R^{\mathsf{T}} \cdot \boldsymbol{v}. \qquad (29)$$

NB: $f(\boldsymbol{v})$ has dimensions $(k \times d)(d \times 1) = k \times 1$

Projection theorem
onto a random dimension k affine subspace

▷ Goal: we shall prove that in projection $\|f(\boldsymbol{v})\| \sim \sqrt{k}\,\|\boldsymbol{v}\|$

▷ Rmks:

- The distance/norm $\|f\|(\cdot)$ increases since the vectors defining the affine space are not unit length.
- The basis defined by R is not orthonormal.
- BUT: the analysis are much simpler!

▷ Thm. Let v be a vector from $\mathbb{R}^d$. Consider a random affine subspace as defined on the previous slide. Then, for any $\varepsilon > 0$:

$$\mathbb{P}\left[\,|\,\|f(\boldsymbol{v})\| - \sqrt{k}\,\|\boldsymbol{v}\|\,| \geq \varepsilon\sqrt{k}\,\|\boldsymbol{v}\|\right] \leq 3e^{-ck\varepsilon^2}. \tag{30}$$

NB: the constant c comes from the Gaussian annulus them.

▷ Proof: See textbook.

▷ NB: versions where matrix R is orthonormal also exist. See the bibliography.

Application: the Johnson-Lindenstrauss lemma

▷ Rationale: project a point set $P = \{x_1, \ldots, x_n\}$ from $\mathbb{R}^d$ to $\mathbb{R}^k$ while preserving distances / with low distorsion.

▷ Thm / lemma: Johnson-Lindenstrauss For any $\varepsilon \in (0, 1)$, consider

$$k \geq \frac{3}{c\varepsilon^2} \ln n. \tag{31}$$

(NB: c from the Gaussian annulus Thm.) For a random projection onto an affine space of dim. k, define the event:

$$\mathcal{E} : (1-\varepsilon)\sqrt{k} \leq \frac{\|f(x_i) - f(x_j)\|}{\|x_i - x_j\|} \leq (1+\varepsilon)\sqrt{k}, \forall (x_i, x_j). \tag{32}$$

One has:

$$\mathbb{P}[\mathcal{E}] \geq 1 - \frac{3}{2n}. \tag{33}$$

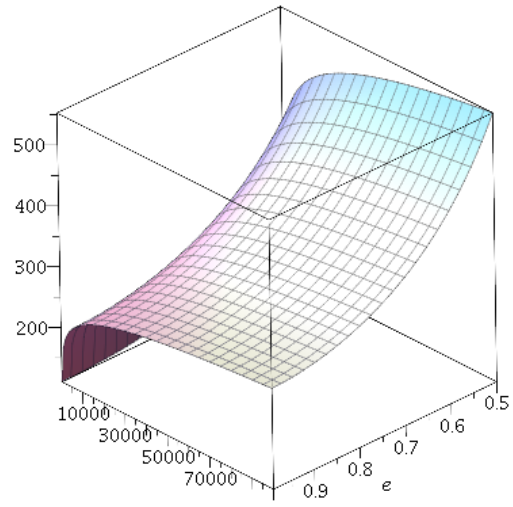
▷ Proof: See textbook.
▷ NB: the only property of data used while defining the projection is the number of samples.

Johnson-Lindenstrauss: lower bound

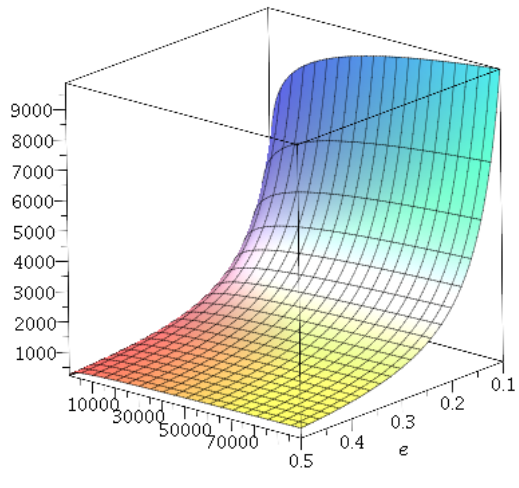
- Embedding dimension k:

$$k = \frac{3}{c\varepsilon^2} \ln n. \qquad (34)$$

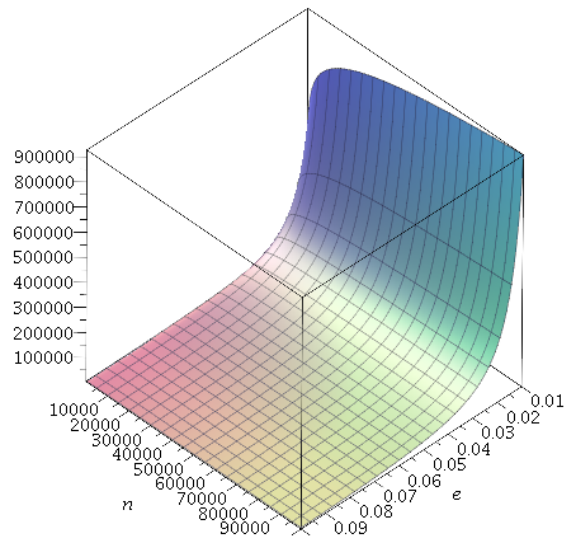
- Large: $\varepsilon \in [0.5 - 0.99]$



- Medium: $\varepsilon \in [0.1 - 5]$



- Small : $\varepsilon \in [0.01 - 0.1]$



Bibliography

- S. Dasgupta and A. Gupta, an elementary proof of a theorem of Johnson and Lindenstrauss, Random structures and algorithms, 2003.
- S. Vempala, The random projection method, AMS, 2005.
- S. Levy, Flavors of geometry, Cambridge, 1997
- A. Blum, J. Hopcroft, R. Kannan, Foundations of Data Science, Cambridge, 2020.