Nearest Neighbors Algorithms in Euclidean and Metric Spaces

Frederic.Cazals@inria.fr
Introduction

kd-trees and basic search algorithms

kd-trees and random projection trees: improved search algorithms

kd-trees and random projection trees: diameter reduction

Metric trees and variants

Distance concentration phenomena: an introduction

A metric from optimal transportation: the earth mover distance
Introduction

kd-trees and basic search algorithms

kd-trees and random projection trees: improved search algorithms

kd-trees and random projection trees: diameter reduction

Metric trees and variants

Distance concentration phenomena: an introduction

A metric from optimal transportation: the earth mover distance
Nearest Neighbors Algorithms in Euclidean and Metric Spaces

Introduction

kd-trees and basic search algorithms

kd-trees and random projection trees: improved search algorithms

kd-trees and random projection trees: diameter reduction

Metric trees and variants

Distance concentration phenomena: an introduction

A metric from optimal transportation: the earth mover distance
Applications

A core problem in the following applications:

- clustering, $k$-means algorithms
- information retrieval in data base
- 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:** \( \text{nn}(q, P) \), the point of \( P \) nearest to \( q \)

\[
d(q, P) = d(q, \text{nn}(q, P)).
\]
The Euclidean Voronoi Diagram and its Dual the Delaunay Triangulation

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 \( \text{Vor}(P) \)
  closer to \( q \) than \( p \),
  step to it: \( p = n(p) \)
- done \( \text{nn}(q) = p \)

▷ Argument: the Delaunay neighborhood of a point is complete
\( \text{Vor}(p, P) = \) cell of \( p \) in \( \text{Vor}(P) \)
\( N(p) = \) set of neighbors of \( p \) in \( \text{Vor}(P) \)
\( N'(p) = \{p\} \cup N(p) \)
\[ \text{Vor}(p, N'(p)) = \text{Vor}(p, P) \]
The Nearest Neighbors Problem: Overview

- **Strategy**: preprocess 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*: typically space \( \mathbb{R}^d \) has a high \( d \) dimension and \( n \gg d \).
  - Interpretation (meaningfull-ness) of distances in high dimensional spaces.
The Nearest Neighbors Problem: Elementary Options

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

▷ Voronoi diagram

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

→ 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
  ▶ $L_2$, $L_p$, $L_\infty$
  ▶ String: Hamming distance
  ▶ Images, graphs: optimal transport
  ▶ Point sets: distances via optimal alignment

▶ Main contenders:

▶ Tree like data structures:
  ▶ $kd$-trees
  ▶ quad-trees
  ▶ metric trees
▶ Locally Sensitive Hashing
▶ Hierarchical $k$-means
Main Contenders: Typical Results

▷ Four main contenders

▷ Winners: size effect

Randomized kd-trees and forest: data structures which are simple, effective, versatile, controlled (in terms of quality performances).

Ref: Muja and Lowe, VISAPP 2009
Ref: O’Hara and Draper, Applications of Computer Vision (WACV), 2013
Nearest Neighbors Algorithms in Euclidean and Metric Spaces

Introduction

kd-trees and basic search algorithms

kd-trees and random projection trees: improved search algorithms

kd-trees and random projection trees: diameter reduction

Metric trees and variants

Distance concentration phenomena: an introduction

A metric from optimal transportation: the earth mover distance
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
kd-tree for a collection of points $P$

▷ Algorithm build_kdTree($S$)

$n \leftarrow newNode$

if $|S| \leq n_0$ 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 value
$L \leftarrow$ point from $S \{v\}$ whose $dir$th coord is $< m$
$R \leftarrow$ point from $S \{v\}$ whose $dir$th coord is $\geq m$
$n.left \leftarrow$ build_kdTree($L$)
$n.right \leftarrow$ build_kdTree($R$)

return $n$
kd-tree: search

Three strategies:

- the defeatist search: simple, but may fail (Nb: see later, distance concentration phenomena)
- the descending search: always succeeds, but may take time
- 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
▷ Algorithm defeatist_search_kdTree: the defeatist search in a kd tree.

**Require:** Maintains $nn(q)$ of $q$, and $\tau = d(q, nn(q))$

$n \leftarrow \text{root}; \tau \leftarrow d(q, n.\text{sample})$

while $n \neq \text{NIL}$ do

Possibly update $nn(q)$ using $n.\text{sample}$, and $\tau$

if $q \in \text{Domain of } L$ then

defeatist_search_kdTree($n.\text{left}$)

if $q \in \text{Domain of } R$ then

defeatist_search_kdTree($n.\text{right}$)

**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 descending search

- Key idea: visit one or two subtree, depending on the distance $d(q, nn(q))$ computed
- Algorithm descending_search_kdTree: the descending search in a kd tree.

Require: Maintains $nn(q)$ of $q$, and $\tau = d(q, nn(q))$

Require: Uses the domain of a node $n$ (an intersection of half-spaces)

$n \leftarrow \text{root}$

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

while $n \neq \text{NIL}$ do
  Possibly update $nn(q)$ using $n\.sample$
  if $\text{Sphere}(q, \tau) \cap \text{Domain of } L$ then
    descending_search_kdTree($n\.left$)
  if $\text{Sphere}(q, \tau) \cap \text{Domain of } R$ then
    descending_search_kdTree($n\.right$)

The value of $\tau$ ensures that the top cell will be visited.
kd-tree search: the priority search (idea)

- 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 $q$

**Ensure:** Maintains $nn(q)$ of $q$, and $\tau = d(q, nn(q))$

1. $nn(q) \leftarrow root.sample$
2. $Q.insert(root)$
3. **while** True **do**
4.   **if** $Q.empty()$ **then**
5.      **return**
6.      \{Node with highest priority\}
7.   \[r \leftarrow Q.pop()\]
8.   \{The nearest box is too far wrt $nn(q)$\}
9.   **if** $d(bbox(r), q) > \tau$ **then**
10.      **return**
11.      \{Descend into box nearest to $q$, \} \{and possibly enqueue the second node\}
12. **for** Nodes $n$ on the path from $r$ to the box nearest to $q$ **do**
13.   \{Possibly update $nn(q)$ and $\tau$\}
14.   \[d \leftarrow d(q, r.sample)\]
15.   **if** $d < \tau$ **then**
16.      $nn(q) \leftarrow r.sample; \tau \leftarrow d$
17.      \{Possibly enqueue the second subtree\}
18.      $f \leftarrow$ farther subtree of $r$ w.r.t $q$
19.   **if** $d(bbox(f), q) \leq \tau$ **then**
20.      \{Insert with priority inverse to distance to $q$\}
21.      $Q.insert(f, 1/d)$
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

- Improvements:
  - Stopping the recursion once a fraction of nodes has been visited
  - Backing up defeatist search with overlapping cells
  - Combining multiple randomized kd-trees
References


Nearest Neighbors Algorithms in Euclidean and Metric Spaces

Introduction

d-trees and basic search algorithms

d-trees and random projection trees: improved search algorithms

d-trees and random projection trees: diameter reduction

Metric trees and variants

Distance concentration phenomena: an introduction

A metric from optimal transportation: the earth mover distance
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
Improvements aiming at fixing the defeatist search

- **Defeatist search**: (early) choice of one side is risky

- **Simple improvements**:
  - Allow overlap between cells in a node
    selected points stored twice: spill trees
  - Use randomization to obtain different partitions:
    several spatial partitions may rescue the defeatist search:
  - Use several trees, and pick the best neighbor(s)
Random projection trees: generic algorithm

Below: version where one also jitters the median

Algorithm build_RPTree($S$)

Ensure: Build the RPTree of a point set $S$

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 $\beta$ uniformly at random from $[1/4, 3/4]$

Let $v$ be the $\beta$-fractile point on the projection of $S$ onto $U$

$Rule(x) = \text{left if } < x, U > < v, \text{ otherwise right}$

$left\_tree \leftarrow \text{build\_RPTree}\{x \in S : Rule(x) = \text{left}\}$

$right\_tree \leftarrow \text{build\_RPTree}\{x \in S : Rule(x) = \text{right}\}$

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

\[
\frac{1}{2} \quad 1/2 \quad \beta \quad 1 - \beta \quad 1/2 + \alpha
\]

- **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**

<table>
<thead>
<tr>
<th></th>
<th>Routing data</th>
<th>Routing queries (defeatist style)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RP tree</td>
<td>Perturbed split</td>
<td>Perturbed split</td>
</tr>
<tr>
<td>Regular spill tree</td>
<td>Overlapping split</td>
<td>Median split</td>
</tr>
<tr>
<td>Virtual spill tree</td>
<td>Median split</td>
<td>Overlapping split</td>
</tr>
</tbody>
</table>
Regular spill trees: size

▷ Tree depth: assume that
   ▷ the number of nodes transmitted to a son decreases by a factor at least \( \beta = 1/2 + \alpha \),
   ▷ a leaf accommodates up to \( n_0 \) points

Then: the tree depth \( l \) satisfies \( \beta^l n \leq n_0 \) i.e. \( l = O(\log_{1/\beta} \frac{n}{n_0}) \).

▷ Tree size:

\[
n_02^l = n_02^{\log_{1/\beta} \frac{n}{n_0}}.
\]

▷ Examples:
   ▷ \( \alpha = 0.05 \): \( O(n^{1.159}) \).
   ▷ \( \alpha = 0.1 \): \( O(n^{1.357}) \).
Spill trees: compromising Storage vs NN searches

- Spill trees: overlapping splits yield superlinear storage
- Yet, search mail fail too:

  \[ p_1 = \text{nn}(q) \in \text{Left} \quad q \in \text{right} \]

- \( p_1 = \text{nn}(q) \): routed in left subtree only
- query point \( q \): routed in right subtree only
Failure of the defeatist search

- **Goal:** probability that a defeatist search does not return the exact nearest neighbor(s)?

- The event to be analyzed, denoted \( \text{Err} \):
  - \( k = 1 \): 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 $q$: $p(1), \ldots, p(n)$

$$\Phi(q, P) = \frac{1}{n} \sum_{i=2}^{n} \frac{\|q - p(1)\|}{\|q - p(i)\|}.$$  \hspace{1cm} (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 $\Phi$. 

$
\begin{array}{c}
\end{array}$
Generalizations of the function $\Phi$

▷ **Rationale:** function $\Phi$ shall be used for nodes containing a subset of the database

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

$$\Phi_m(q, P) = \frac{1}{m} \sum_{i=2}^{m} \frac{\|q - p(i)\|}{\|q - p(1)\|}. \quad (3)$$

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

$$\Phi_{k,m}(q, P) = \frac{1}{m} \sum_{i=k+1}^{m} \frac{\|q - p(1)\| + \cdots + \|q - p(k)\|}{\|q - p(i)\|}. \quad (4)$$
Random projections: relative position of three points

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

- Colinearity index $q, x, y$:

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

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

Lemma 1. Consider $q, x, y \in \mathbb{R}^d$ and $\|q - x\| \leq \|q - y\|$. The probability over random directions $U$, of $E$, satisfies:

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

Corollary 2.

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

Compulsory tools for geometers

- **DrGeo**: [http://www.drgeo.eu/](http://www.drgeo.eu/)
Random projections: separation of neighbors

**Theorem 3.** 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 2, $\mathbb{P}[Z_i] \leq (1/2) \frac{\|q - p_{(1)}\|}{\|q - p(i)\|}$. Then, apply the linearity of expectation.

**Theorem 4.** Let $S \subset P$ with $p_{(1)} \in S$. If $U$ is chosen uniformly at random, then $0 < \alpha < 1$, the probability that an $\alpha$ fraction of the projected points in $S$ fall between $q$ and $p_{(1)}$ is bounded by

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

**Proof.** Previous Thm + Markov’s inequality.
Regular spill trees

▶ Recap:
   ▶ Storage: point possibly stored twice using overlapping split with parameter $\alpha$; depth is $O(\log n/n_0)$
   ▶ Query routing: routing to a single leaf

**Theorem 5.** Let $\beta = 1/2 + \alpha$. The error probability is:

$$
\mathbb{P}[\text{Err}] \leq \frac{1}{2\alpha} \sum_{i=0,\ldots,l} \Phi_{\beta^i n}(q, P)
$$

▶ Proof, steps:
   ▶ Internal node at depth $i$ contains $\beta^i n$ points
   ▶ For such a node: proba to have $q$ separated from $p_{(1)}$
      $p_{(1)}$ transmitted to one side of the split $\Rightarrow$ a fraction $\alpha$ of the points of the cell fall between $q$ and $p_{(1)}$: this occurs with proba $(1/2\alpha)\Phi_{\beta^i n}(q, P)$
   ▶ To conclude: union-bound over all levels $i$
Virtual spill trees

Recap:

- Storage: each point stored in a single leaf with median splits; depth is $O(\log n/n_0)$
- Query routing: with overlapping splits of parameter $\alpha$

Theorem 6. Let $\beta = 1/2$. The error probability is:

$$P[\text{Err}] \leq \frac{1}{2\alpha} \sum_{i=0,\ldots,l} \Phi_{\beta^n}(q, P)$$  \hspace{1cm} (9)

Proof, mutatis mutandis:

- Consider the path root - leaf of $p(1)$
- For a level, bound the proba. to have $q$ routed to one side only
- Add up for all levels
Random projection trees

Recap:
- Pick a random direction and project points onto it
- Split at the $\beta$ 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 7. Consider an RP tree for $P$. Define $\beta = 3/4$, and
\[ l = \log_{1/\beta}(n/n_0). \]
One has:
\[
\mathbb{P} \left[ \text{NN query does not return } \text{p}(1) \right] \leq \sum_{i=0,\ldots,l} \Phi_{\beta^i n} \ln \frac{2e}{\Phi_{\beta^i n}} \tag{10}
\]

Proof, key steps:
- $F$: fraction of points separating $q$ and $\text{p}(1)$ in projection
- Since split chosen at random in interval of mass 1/2: it separates $q$ and $\text{p}(1)$ with proba. $F/(1/2)$
- Integrating yields the result for one level; then, union bound.
Theorem 8. (Spill trees) Consider a spill tree of depth \( l = \log \frac{1}{\beta} \left( \frac{n}{n_0} \right) \), with

- \( \beta = \frac{1}{2} + \alpha \) for regular spill trees,
- and \( \beta = \frac{1}{2} \) for virtual spill trees.

If this tree is used to answer a query \( q \), then:

\[
\Pr[\text{Err}] \leq \frac{1}{2\alpha} \sum_{i=0,\ldots,l} \Phi_{\beta^n}(q, P)
\]  

Nb: \( \beta^n \): number of data points found in an internal node at depth \( i \)
Randomization is critical for separation

Separation property fails in using coordinate axis (kd-trees)

Consider the following point set:

- $x_1$: the all-ones vector
- 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 $(0, 1)$

This examples rules out kd-trees: kd-trees separate $q$ and $p_{(1)}$, even though function $\Phi$ is arbitrarily small

- The NN of $q$ (=origin) is $x_1$
- Upon picking a random direction (a coord. axis), the fraction of points falling in-between $q$ and $x_1$ is arbitrarily large:

\[
\frac{1}{n} \left(n - \frac{n}{d}\right) = 1 - \frac{1}{d}
\]

- But by growing $M$, function $\Phi$ gets close to 0.
Doubling dimension (Assouad dimension) and doubling measures

- Def.: A metric space $X$ with metric is called doubling is any ball $B(x, r)$ with $x \in X$ and $r > 0$ is contained in at most $M$ balls of radius $r/2$. The doubling dimension is $\log_2 M$.

- Exple: line

- Exple: $k$-affine space: $O(k)$

- Example: smooth $k$ manifold: also $O(k)$

   Nb: proof uses bound on sectional curvatures.

- Def.: A measure $\mu$ on a metric space $X$ is called doubling if

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

Equivalently, $\exists d_0$ such that $\forall \alpha > 0$:

$$\mu(B(x, 2r) \leq \alpha^{d_0} \mu(B(x, r)).$$

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

- Remarks:

  - A measure 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.
Intermezzo: Data and their intrinsic dimension

- **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 1: 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)

- **Example 2: 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
Bounding function $\Phi$ in specific settings

Improving the bound $\Phi \leq 1$

**Theorem 9.** Let $\mu$ be a continuous measure on $\mathbb{R}^d$, a doubling measure of dimension $d_0 \geq 2$. Assume $p_1, \ldots, p_n \sim \mu$. With probability $\geq 1 - 3\delta$, for all $2 \leq m \leq n$:

$$
\Phi_m(q, P) \leq 6\left(\frac{2}{m} \ln \frac{1}{\delta}\right)^{1/d_0}.
$$

**Theorem 10.** Under the same hypothesis:

- For both variants of the spill trees:

$$
P[Err] \leq \frac{c_0 kd_0}{\alpha} \left(\frac{8 \max(k, \ln 1/\delta)}{n_0}\right)^{1/d_0}
$$

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

$$
P[Err] \leq c_0 (d_0 + \ln n_0)\left(\frac{8 \max(k, \ln 1/\delta)}{n_0}\right)^{1/d_0}
$$

▶ Rmk: failure proba. can be made arbitrarily small by taking $n_0$ large enough.
References


**VKD09** N. Verma, S. Kpotufe, S. Dasgupta, Which spatial partitions are adaptive to intrinsic dimension? UAI 2009.
Nearest Neighbors Algorithms in Euclidean and Metric Spaces

Introduction

kd-trees and basic search algorithms

kd-trees and random projection trees: improved search algorithms

kd-trees and random projection trees: diameter reduction

Metric trees and variants

Distance concentration phenomena: an introduction

A metric from optimal transportation: the earth mover distance
Recursive splits: how many splits are required to halve the diameter of a point set?

- Another set defined along axis:
  - Consider $S = \bigcup_{i=1,...,d} \{ t e_i, -1 \leq t \leq 1 \}$.
  - $S \subseteq B(0,1)$ and covered by $2d$ balls $B(\cdot, 1/2)$ (this num. is minimal).
  - Assouad dimension is $\log 2d$

- Observation: kd-trees requires
  - $d$ splits / levels to halve the diameter of $S$
  - this requires in turn $\geq 2^d$ points

- Fact: RPTree will do the job!
Local covariance dimension and its multi-scale estimation

▶ Def.: A set $T \subset \mathbb{R}^D$ has covariance dimension $(d, \epsilon)$ 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 $(d, \epsilon, r)$: 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
Partitioning rules that adapt to intrinsic dimension

- **Principal component analysis**: split the data at the median along the principal direction of covariance.
  - Drawback 1: estimation of principal component requires a significant amount of data and only about \( \frac{1}{2^l} \) fraction of data remains at a cell at level \( l \)
  - Drawback 2: computationally too expensive for some applications

- **2-means i.e. solution of k-means with \( k = 1 \)**: compute the 2-means solution, and split the data as per the cluster assignment
  - Drawback 1: 2-means is an NP-hard optimization problem
  - Drawback 2: the best known \((1 + \epsilon)\)-approximation algorithm for 2-means (A. Kumar, Y. Sabharwal, and S. Sen, 2004) would require a prohibitive running time of \( O(2^{dO(1)} Dn) \), since we need \( \epsilon \approx 1/d \).
  - Approximate solution can be obtained using Loyd iterations.
Random projections and distances

In $\mathbb{R}^D$: distance roughly get shrunk by a factor $1/\sqrt{D}$

Lemma 11. Fix any vector $x \in \mathbb{R}^D$. Pick any random unit vector $U$ on $S^{d-1}$. One has:

\[
\mathbb{P} \left[ |\langle x, U \rangle| \leq \alpha \frac{\|x\|}{\sqrt{D}} \right] \leq \frac{2}{\pi} \alpha \quad (12)
\]

\[
\mathbb{P} \left[ |\langle x, U \rangle| \geq \beta \frac{\|x\|}{\sqrt{D}} \right] \leq \frac{2}{\beta} \exp -\frac{\beta^2}{2} \quad (13)
\]
Random projections and diameter

- Projecting a subset $S \subset \mathbb{R}^D$ along a random direction: how does the diameter of the projection compares to that of $S$?

- $S$ full dimensional:
  \[
  \text{diam(projection)} \leq \text{diam}(S)
  \]

- $S$ has Assouad dimension $d$:
  (with high probability)
  \[
  \text{diam(projection)} \leq \text{diam}(S)\sqrt{d/D}
  \]

- Rmk:

  Cover $S$ with
  - $2^d$ balls of radius $1/2$
  - $4^d$ balls of radius $1/4$
  - $(1/\varepsilon)^d$ balls of radius $\varepsilon$
Random projection trees algorithm: rationale

- Keep the good properties of PCA at a much lower cost
  - Intuition: splitting along a random direction is not that different since it will have some component in the direction of the principal component
- Generally works, but in some cases fails to reduce diameter
  - Think of a dense spherical cluster around the mean containing most of the data and a concentric shell of points much farther away
  - Characterized by the average interpoint distance $\Delta_A$ within cell being much smaller than its diameter $\Delta$
  - $\Rightarrow$ another split is used, based on distance from the mean
Linear versus spherical cuts

▷ Linear split with jiter:

\{\text{Split by projection: no outlier}\}

\textbf{ChooseRule}(S)

choose a random unit direction \(v\)
pick any \(x \in S\) at random
let \(y \in S\) a point realizing the diameter of \(S\)
choose \(\delta\) at random in \([-1, 1]\) \(\|x - y\| / \sqrt{d}\)

\text{Rule}(x) := x \cdot v \leq (\text{median}_{z \in S}(z \cdot v) + \delta)

▷ Combined split:

\{\text{Split by projection: no outlier}\}

\textbf{ChooseRule}(S)

\textbf{if} \ \Delta^2(S) \leq c \cdot \Delta_A^2(S) \ \textbf{then}

choose a random unit direction \(v\)

\text{Rule}(x) := x \cdot v \leq \text{median}_{z \in S}(z \cdot v)

\textbf{else}

\{\text{Spherical cut: remove outliers}\}

\text{Rule}(x) := \|x - \text{mean}(S)\| \leq \text{median}_{z \in S}(\|z - \text{mean}(S)\|)
Random projection trees algorithm: RPTree-max and RPTree-mean

▷ Algorithm:

\begin{verbatim}
MakeTree(S)
if |S| < MinSize then
    return (Leaf)
else
    Rule ← ChooseRule(S)
    LeftTree ← Maketree(\{x ∈ S : Rule(x) = true\})
    RightTree ← Maketree(\{x ∈ S : Rule(x) = false\})
    return [Rule, LeftTree, RightTree]
\end{verbatim}

▷ Two options

- RPTree-max: linear split with jiter
- RPTree-mean: combined split
Performance guarantee:
amortized (i.e., global) result for RPTree-max

▷ Def.: \textit{radius} of a cell $C$ of a
RPTree: smallest $r > 0$ such that
$S \cap C \subset B(x, r)$ for some $x \in C$.

\textbf{Theorem 12.} (RPTree-max) Consider a RPTree-max built for a dataset
$S \subset \mathbb{R}^D$. Pick any cell $C$ of the tree; assume that $S \cap C$ has Assouad
dimension $\leq d$. There exists a constant $c_1$ such that with proba. $\geq 1/2$, for
every descendant $C'$ more than $c_1 d \log d$ levels below $C$, one has
radius($C'$) $\leq$ radius($C$)/2.
Performance guarantee:
per-level result for RPTree-mean, with adaptation to covariance dimension

**Theorem 13.** (RPTree-mean) There exists constants $0 < c_1, c_2, c_3 < 1$ for which the following holds.

- Consider any cell $C$ such that $S \cap C$ has covariance dimension $(d, \epsilon)$, $\epsilon < c_1$.
- Pick $x \in S \cap C$ at random, and let $C'$ be the cell containing it at the next level down.
- Then, if $C$ is split:
  - by projection: $(\Delta^2(S) \leq c \cdot \Delta_A^2(S))$
    $$\mathbb{E}[\Delta_A^2(S \cap C')] \leq (1 - (c_3/d))\Delta_A^2(S \cap C)$$
  - by distance i.e. spherical cut:
    $$\mathbb{E}[\Delta^2(S \cap C')] \leq c_2 \Delta^2(S \cap C)$$

**NB:** the expectation is over the randomization in splitting $C$ and the choice of $x \in S \cap C$. 
Empirical results: contenders

 Algorithms:

- dyadic trees: 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.
Real world datasets

- Teapot dataset: rotated images of a teapot (1 B&W image: 50×30 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; 20×20 B&W images, i.e. points in ambient dimension 400.
Empirical results: local covariance dimension estimation

▶ Conventions: bold lines: estimate $d(r)$; dashed lines: std dev; numbers: ave. num of 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.

Empirical results: performance for NN searches

Searching $p(1)$: performance is the order of the NN found / dataset size

- tree depth: if one stops the recursion at a given depth
- numbers indicate ratio $\|q - nn(q)\| / \|q - p(1)\|$

Observations:

- quality index deteriorates with depth (separation does occur)
- 2M and PD (i.e. PCA trees) consistently yield better nearest neighbors: better adaptation to the intrinsic dimension

Empirical results: regression

Regression:
- one predicts the rotation angle (response variable)
- performance is $l_2$ error on the response variable
- theory says that best results are expected for data structure adapting to the intrinsic dimension

Observations:
- Best results for 2M trees, PD (i.e., PCA) trees, and RP trees.

Diameter reduction again: the revenge of kd-trees

▷ Diameter reduction property: holds for kd-trees on randomly rotated data

▷ Rmk: one random ration suffices

References


Nearest Neighbors Algorithms in Euclidean and Metric Spaces

Introduction

kd-trees and basic search algorithms

kd-trees and random projection trees: improved search algorithms

kd-trees and random projection trees: diameter reduction

Metric trees and variants

Distance concentration phenomena: an introduction

A metric from optimal transportation: the earth mover distance
Metric spaces

**Definition 14.** 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 \(k > 1\):

\[ M = M_1 \times \cdots \times M_k. \]  

(14)

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

\[ d(x, y) = \left( \sum_{k=1}^{k} d_i(x_i, y_i)^p \right)^{1/p} \]

(15)

Some particular cases are:

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

- **Hausdorff distance.** Consider a metric space \((M, d)\). The *Hausdorff distance* of two non-empty subsets \(X\) and \(Y\) is defined by

\[
d_H(X, Y) = \max(H(X, Y), H(Y, X)), \text{ with } H(X, Y) = \sup_{x \in X} \inf_{y \in Y} d(x, y). \tag{16}
\]

Note that the one-sided distance is not symmetric, as seen on Fig. 1.

- **Rmk.** For closed set, the min distance is realized: \(\inf\) becomes \(\min\); \(\sup\) becomes \(\max\).

![Figure: The one-sided Hausdorff distance is not symmetric](image_url)
A geometric distance for two ordered point clouds:

the least Root Mean Square Deviation: \( \text{lRMSD} \)

▷ Data: two point sets \( A = \{ a_i \}_{i=1,...,n}, \) \( B = \{ b_i \}_{i=1,...,n}, \) with a 1-1 correspondence \( a_i \leftrightarrow b_i \)

▷ Root Mean Square Deviation:

\[
\text{RMSD}(A, B) = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \| a_i - b_i \|^2}
\] (17)

▷ least Root Mean Square Deviation:

\[
\text{IRM}SD(A, B) = \min_{g \in SE(3)} \text{RMSD}(A, g \cdot B).
\] (18)

▷ Ref: Umeyama, IEEE PAMI 1991
▷ Ref: Steipe, Acta Crystallographica Section A, 2002
Using the triangle inequality

**Lemma 15.** For any three points \( p, q, s \in M \), for any \( r > 0 \), and for any point set \( P \subset M \), one has:

\[
|d(q, p) - d(p, s)| \leq d(q, s) \leq d(q, p) + d(p, s) \tag{19}
\]

\[
d(q, s) \geq d_P(q, s) := \max_{p \in P} |d(q, p) - d(p, s)| \tag{20}
\]

\[
\begin{cases}
   d(p, s) > d(p, q) + r \Rightarrow d(q, s) > r,
   d(p, s) < d(p, q) - r \Rightarrow d(q, s) > r.
\end{cases} \tag{21}
\]

**Figure:** Lower bound from the triangle inequality, see Lemma 15
Metric tree: definition

Definition:

- A binary tree
- Any internal node implements a spherical cut defined by the distance $\mu$ 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 1 Algorithm build_MetricTree($S$)

```
{build_MetricTree(S)}
if $S = \emptyset$ then
  return NIL

$n \leftarrow newNode$

Draw at random $Q \subset S$ and $v \in Q$

$n.pivot \leftarrow v$

$\mu \leftarrow \text{median}\left\{d(v, p), p \in Q\setminus\{p\}\right\}$

{The pivot splits points into two subsets}
$L \leftarrow \{s \in S\setminus\{p\} | d(s, v) < \mu\}$
$R \leftarrow \{s \in S\setminus\{p\} | d(s, v) \geq \mu\}$

{For each subtree: min/max distances to points in that subtree}
$n.(d_1, d_2) \leftarrow (\text{min}, \text{max})$ of distances $d(v, p), p \in L$
$n.(d_3, d_4) \leftarrow (\text{min}, \text{max})$ of distances $d(v, p), p \in R$

{Recursion}
$n.L \leftarrow \text{build}_\text{MetricTree}(L)$
$n.R \leftarrow \text{build}_\text{MetricTree}(R)$
```
Searching a metric tree: algorithm

Algorithm 2 Algorithm
search_MetricTree( T, q)

{Note of T is denoted n}
nn(q) ← ∅
τ ← ∞
if n = NIL then
    return

{Check whether the pivot is the nn}
l ← d(q, n.pivot)
if l < τ then
    nn(q) ← n.pivot
    τ ← l

{Dilate the distance intervals for left and right subtrees}

if l ∈ l_L then
    search_MetricTree(n.L, q)
if l ∈ l_R then
    search_MetricTree(n.R, q)
Lemma 16. Consider the intervals associated with a node, as defined in
Algorithm 1, 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 $d_2$, 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. \[\square\]
Metric tree: choosing the pivot

- By the pruning lemma: for small $\tau$ 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 $\mu$ (rather than the $d_i$s):
  - Best pivot: $p_c$
  - Worst pivot: $p_m$

Figure: Metric trees: minimizing the measure of boundaries.
From metric trees to metric forests

- Compromising speed versus accuracy
  - The exact search may be replaced by the defeatist search: visit one subtree only, instead of using the pruning lemma.
  - Then, using a forest of trees rescues erroneous branching decisions in the course of the search.

Figure: Metric forest
References


Nearest Neighbors Algorithms in Euclidean and Metric Spaces

Introduction

kd-trees and basic search algorithms

kd-trees and random projection trees: improved search algorithms

kd-trees and random projection trees: diameter reduction

Metric trees and variants

Distance concentration phenomena: an introduction

A metric from optimal transportation: the earth mover distance
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, \( p > 0 \):

\[
\| \cdot \|_p = \left( \sum_i |X_i|^p \right)^{1/p} \tag{22}
\]

Unit balls: see plots

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.
Concentration of the Euclidean Norm: Observations

- Plotting the variation of the following for random points in \([0, 1]^d\):

\[
\min \|\cdot\|_2, \ E[\|\cdot\|_2] - \sigma[\|\cdot\|_2], \ E[\|\cdot\|_2] + \sigma[\|\cdot\|_2], \ \max \|\cdot\|_2, \ M = \sqrt{d}
\]

(23)

Observation:

- The average value increases with the dimension \(d\)
- The standard deviation seems to be constant
- 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 17.** 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:

\[
\mathbb{E} \left[ \|X\|_2 \right] = \sqrt{ad} - b + O(1/d)
\]

\[
\text{Var} \left[ \|X\|_2 \right] = b + O(1/\sqrt{d}).
\]  

**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.
Contrast and Relative Contrast: Definition

Contrast and relative contrast of $n$ iid random draws from $X$. The annulus centered at the origin and containing the points is characterized by:

$$\text{Contrast}_a := D_{\text{max}} - D_{\text{min}} = \max_j \| p^{(j)} \|_p - \min_j \| p^{(j)} \|_p.$$  \hfill (26)

and the relative contrast is defined by:

$$\text{Contrast}_r = \frac{D_{\text{max}} - D_{\text{min}}}{D_{\text{min}}}.$$ \hfill (27)

Variation of the contrast $|D_{\text{max}} - D_{\text{min}}|$ for various $p$ and increasing $d$:

![Graphs showing variation of contrast](image1)

Fig. 1. $|D_{\text{max}} - D_{\text{min}}|$ depending on $d$ for different metrics (uniform data)
**Theorem 18.** Consider $n$ points which are iid realization of $X$. There exists a constant $C_p$ such that the absolute contrast of a Minkowski norm satisfies:

$$C_p \leq \lim_{d \to \infty} \mathbb{E} \left[ \frac{D_{\text{max}} - D_{\text{min}}}{d^{1/p-1/2}} \right] \leq (n-1)C_p.$$  \hspace{1cm} (28)

**Observations:**

- The contrast grows as $d^{1/p-1/2}$

<table>
<thead>
<tr>
<th>Metric</th>
<th>Contrast $D_{\text{max}} - D_{\text{min}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$L_1$</td>
<td>$C_1 \sqrt{d}$</td>
</tr>
<tr>
<td>$L_2$</td>
<td>$C_2$</td>
</tr>
<tr>
<td>$L_3$</td>
<td>0</td>
</tr>
</tbody>
</table>

- The Manhattan metric: only one for which the contrast grows with $d$.
- For the Euclidean metric, the contrast converge to a constant.
- For $p \geq 3$, the contrast converges to zero: the distance does not discriminate between the notions of close and far.
- NB: the bounds depend on $n$; it makes sense to try to exploit the particular coordinates at hand (cf later).

**NB:** Thm also exist for the relative contract and other $p$-norms.
Practical Implications for Exact NN Queries

- **Curse of dimensionality**: the concentration of distance can be such that a method with expected logarithmic cost performs no better than a linear scan. Recall the non defeatist search strategies in kd-tree and metric trees: if all distances are comparable, a linear number of nodes gets visited.

- **Use less concentrated metrics, with more discriminative power.**

- **Sanity check**: in running a NN query, make sure that distances are meaningful: bi-modality of the distribution of distance is a good sanity check.

- **Example**: selected features yield two modes rather than a continuum of distances:

- ![Histograms](image)

  (a) all rel. Dim.
  
  (b) one non-rel. Dim.
  
  (c) two non-rel. Dim.

*Figure 7: Distance Distribution of Data*
A wise use of distances

- **Distance filtering:**

- **Feature selection:**
References


Nearest Neighbors Algorithms in Euclidean and Metric Spaces

Introduction

kd-trees and basic search algorithms

kd-trees and random projection trees: improved search algorithms

kd-trees and random projection trees: diameter reduction

Metric trees and variants

Distance concentration phenomena: an introduction

A metric from optimal transportation: the earth mover distance
Comparing Histograms

- Bin-to-bin methods:
  \[ d(H, K) = \sum_i |h_k - k_k| \]  
  \[ (29) \]
  → overestimates the distance since neighboring bins are not considered.

- Mixing (e.g. quadratic) methods:
  \[ d^2(H, K) = (h - k)^t A (h - k) \]  
  \[ (30) \]
  → underestimates distances: tends to accentuate the similarity of color distributions without a pronounced mode.

- Illustrations:

  (A) (B) (C) (D)
Transport Plan Between Two Weighted Point Sets

- **Weighted point sets:**
  \[ P = \{(p_1, w_{p_1}), \ldots, (p_m, w_{p_m})\} \text{ and } Q = \{(q_1, w_{q_1}), \ldots, (q_n, w_{q_n})\}. \] (31)

NB: nodes from \( P \) (resp. \( Q \)): production (resp. demand) nodes
Shorthand for the sum of masses: \( W_P = \sum_i w_{p_i}, \; W_Q = \sum_j w_{q_j}. \)

- A metric \( d(\cdot, \cdot) \): distance between two points \( d_{ij} = d(p_i, q_j). \)
- A **transport plan**: is a set of non-negative flows \( f_{ij} \) circulating on the edges of the bipartite graph \( P \times Q. \)

**Figure:** Transport plan between two weighted point sets
The Earth Mover Distance

- **Optimization problem:**

  Minimize: $C_{EMD-LP} = \sum_{ij} f_{ij} d_{ij}$ under the constraints:

  $\begin{cases} 
  (C1) f_{ij} \geq 0 \\
  (C2) \sum_j f_{ij} \leq w_{pi}, \forall i \\
  (C3) \sum_i f_{ij} \leq w_{qj}, \forall j \\
  (C4) \sum_i \sum_j f_{ij} = \min(W_P, W_Q).
  \end{cases}$

  These constraints read as follows:

  - (C1) Flows are positive
  - (C2,C3) A node cannot export (resp. receive) more than its weight.
  - (C4) The total flow neither exceeds the production nor the demand.

- **Earth mover distance:** defined from the cost by

  $$d_{EMD-LP} = \frac{C_{EMD-LP}}{\sum_{ij} f_{ij}} = \frac{C_{EMD-LP}}{\min(W_P, W_Q)}$$

- **Advantages:**

  - Applies to signatures in general, the histograms being a particular case.
  - Embeds the notion of nearness, via the metric in the ground space.
  - Allows for partial matches. See however, the comment in section ??.
  - Easy to compute: linear program.

- **Ref:** Rubner, Tomasi, Guibas, IJCV, 2000
Application to image retrieval

- Image coding, two options:
  - convert image to histogram using a fixed binning of the color space; mass of bin: num. of pixel within it.
  - cluster pixels (say with k-means): mass of cluster is the fraction of pixels assigned to it

- Search on DB of 20,000 images: (a) $L_1$ (d) Quadratic form (e) EMD
Mallow’s Distance: Definition

- Consider: two RV in $\mathbb{R}^d$: $X \sim P$, $Y \sim Q$.
- Mallows distance between $X$ and $Y$: minimum of expected difference between $X$ and $Y$ over all joint distributions $F$ for $(X, Y)$, such that the marginal of $X$ is $P$ and that of $Y$ is $Q$:

$$M_p(X, Y) = \min_{F \in \mathcal{F}} (E_F \|X - Y\|^p)^{1/p} : (X, Y) \sim F, X \sim P, Y \sim Q \}.$$  \hspace{1cm} (35)

- Discrete setting: $P$ and $Q$

$$P = \{(x_1, w_{p_1}), \ldots, (x_m, w_{p_m})\} \hspace{1cm} (36)$$

$$Q = \{(y_1, w_{q_1}), \ldots, (y_n, w_{q_n})\} \hspace{1cm} (37)$$

- Joint distribution is specified by probabilities on all pairs i.e. $F = \{f_{ij}\}$, and the fact that it respects the marginals yields:

$$\sum_j f_{ij} = p_i, \quad \sum_i f_{ij} = q_j, \quad \sum_{ij} f_{ij} = 1.$$  \hspace{1cm} (38)

- Functional to be minimized becomes:

$$E_F \|X - Y\|^p = \sum_{ij} f_{ij} \|x_i - y_j\|^p.$$  \hspace{1cm} (39)
Mallow’s Distance versus EMD

- Mallows’ distance \((W_P = W_Q = 1)\):
  \[
  M_p = \frac{1}{4} \times 0 + \frac{1}{4} \times 1 + \frac{1}{4} \times 1 + \frac{1}{4} \times 0
  \]

- EMD, assuming uniform weights on all points, i.e. \(W_P = 2\) and \(W_Q = 4\): EMD = 0 since a flow of 2 units satisfies all constraints.

Figure: Mallows’s distance
References

**LB01** Elizaveta Levina and Peter Bickel. The earth mover’s distance is the
mallows distance: Some insights from statistics. In Computer Vision,

**RTG00** Y. Rubner, C. Tomasi, and L. J. Guibas. The earth mover’s distance as a
metric for image retrieval. International Journal of Computer Vision,

**ZLZ05** Ding Zhou, Jia Li, and Hongyuan Zha. A new mallows distance based
metric for comparing clusterings. In Proceedings of the 22nd international

**CM14** F. Cazals and D. Mazauric. Mass transportation problems with
connectivity constraints, with applications to energy landscape