Fast Nearest Neighbour Classification

Gordon Lesti

July 17, 2015

Structure

Nearest-Neighbour Searching

Input

- Set $\mathbb{U}$
- Distance function d on $\mathbb{U}$, with $d : \mathbb{U} \times \mathbb{U} \to \mathbb{R}$
- Set $S \subset \mathbb{U}$ of size n
- Query item $q \in \mathbb{U}$

Output

- Item $a \in S$, with $d(q, a) \leq d(q, x)$ for all $x \in S$

Use

- Pattern recognition
- Statistical classification
- Image editing
- Coding theory
- Data compression
- Recommender system
- ...

Full Search

- Calculate $d(q, x)$ for all $x \in S$
- Return $a \in S$, with $d(q, a) \leq d(q, x)$ for all $x \in S$

Full Search

Example

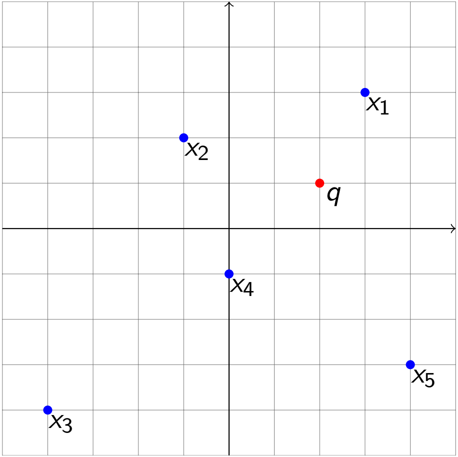
$\mathbb{U} = \mathbb{R}^2$

Items

$x_1 = (3, 3)$
$x_2 = (-1, 2)$
$x_3 = (-4, -4)$
$x_4 = (0, -1)$
$x_5 = (4, -3)$

Query item

$q = (2, 1)$

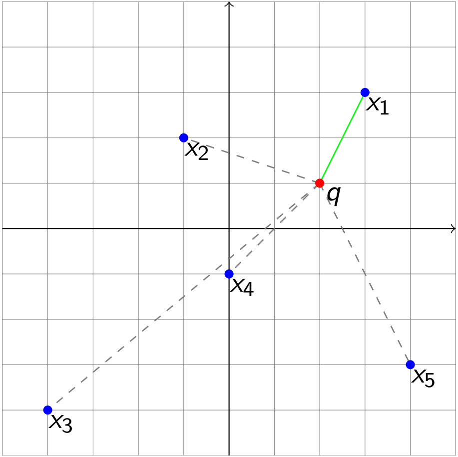


Full Search

Example

Result

$d(q, x_1) \approx 2.236$
$d(q, x_2) \approx 3.162$
$d(q, x_3) \approx 7.810$
$d(q, x_4) \approx 2.828$
$d(q, x_5) \approx 4.472$



Full Search

Advantages and disadvantages

Advantages

- Easy implementation
- Works in none metric spaces

Disadvantages

- Large runtime on big data sets and in higher multidimensional spaces

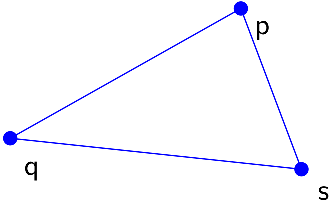
Metric

Given a set X. A *Metric* on X is a function
$d : X \times X \to \mathbb{R}$, $(x, y) \mapsto d(x, y)$
with:

1. $d(x, y) = 0$ exactly when $x = y$.
2. Symmetry: For all $x, y \in X$ is true $d(x, y) = d(y, x)$.
3. Triangle inequality: For all $x, y, z \in X$ is true
   $d(x, z) \leq d(x, y) + d(y, z)$

[Forster, 2006]

Triangle inequality



Lemma For any $q, s, p \in \mathbb{U}$, $r \in \mathbb{R}$ and $P \subset \mathbb{U}$ is true:

1. $|d(p,q) - d(p,s)| \leq d(q,s) \leq d(p,q) + d(p,s)$
2. $d(q,s) \geq d_P(q,s) := \max_{p \in P} |d(p,q) - d(p,s)|$
3. $d(p,s) > d(p,q) + r \vee d(p,s) < d(p,q) - r \Rightarrow d(q,s) > r$
4. $d(p,s) \geq 2 \cdot d(p,q) \Rightarrow d(q,s) \geq d(q,p)$

[Clarkson, 2005]

Orchards Algorithm

- Create a list for every item $p \in S$ with all items $x \in S$, ordered ascending to the distance
- Choose random item $c \in S$ as initial candidate
- Calculate $d(c, q)$
- Go along the list of c
- If the current item has smaller distance to q as c, choose current item as c
- Abort, if
  - at the end of the current list or
  - $d(c, s) > 2 \cdot d(c, q)$ for the current item of the list (Triangle inequality 4)
- Else c is nearest neighbour

Orchards Algorithm

Example

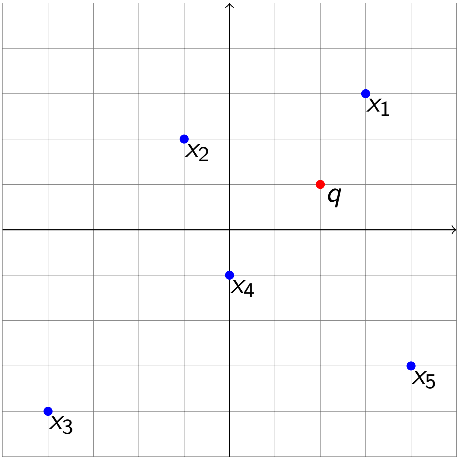
$\mathbb{U} = \mathbb{R}^2$

Items

$x_1 = (3, 3)$
$x_2 = (-1, 2)$
$x_3 = (-4, -4)$
$x_4 = (0, -1)$
$x_5 = (4, -3)$

Query item

$q = (2, 1)$



Orchards Algorithm

Example

Distances

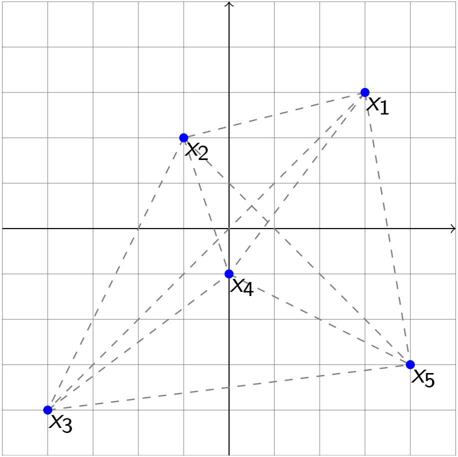
	x_1	x_2	x_3	x_4	x_5
x_1	0	≈ 4.123	≈ 9.899	5	≈ 6.083
x_2	≈ 4.123	0	≈ 6.708	≈ 3.162	≈ 7.071
x_3	≈ 9.899	≈ 6.708	0	5	≈ 8.062
x_4	5	≈ 3.162	5	0	≈ 4.472
x_5	≈ 6.083	≈ 7.071	≈ 8.062	≈ 4.472	0

Orchards Algorithm

Example

Lists

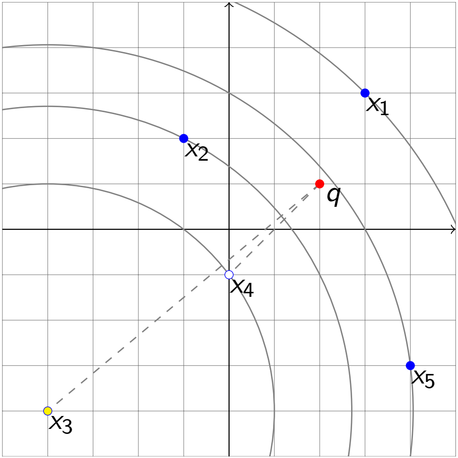
$L(x_1) = \{x_2, x_4, x_5, x_3\}$
$L(x_2) = \{x_4, x_1, x_3, x_5\}$
$L(x_3) = \{x_4, x_2, x_5, x_1\}$
$L(x_4) = \{x_2, x_5, x_1, x_3\}$
$L(x_5) = \{x_4, x_1, x_2, x_3\}$



Orchards Algorithm

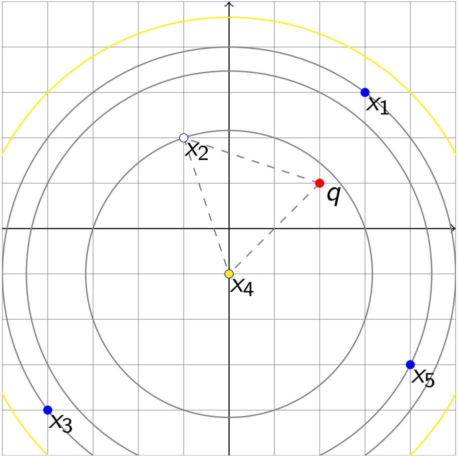
Example

- Set $c := x_3$ and $s := x_4$
- As $7.810 \approx d(c, q) > d(s, q) \approx 2.828$, set $c := s$



Orchards Algorithm
Example

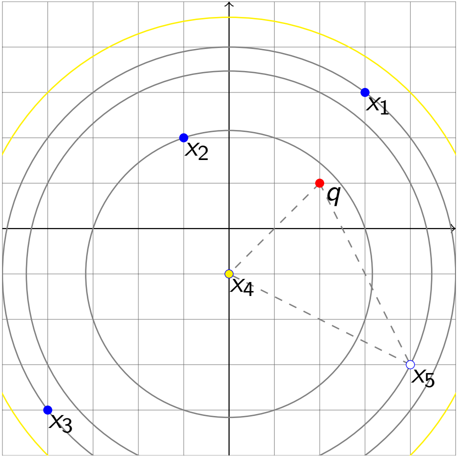
- Set $c := x_4$ and $s := x_2$
- As $2.828 \approx d(c, q) < d(s, q) \approx 3.162$, no new c
- As $3.162 \approx d(c, s) < 2 \cdot d(c, q) \approx 5.656$, no abort



Orchards Algorithm

Example

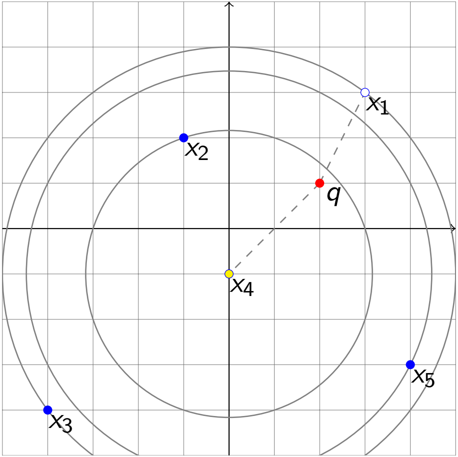
- Set $s := x_5$
- As $2.828 \approx d(c,q) < d(s,q) \approx 4.472$, no new c
- As $4.472 \approx d(c,s) < 2 \cdot d(c,q) \approx 5.656$, no abort



Orchards Algorithm

Example

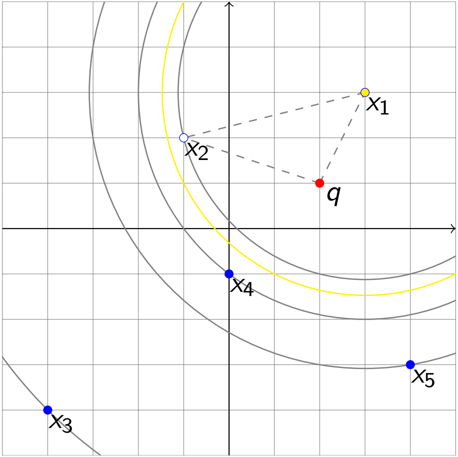
- Set $s := x_1$
- As $2.828 \approx d(c, q) > d(s, q) \approx 2.236$, set $c := s$



Orchards Algorithm

Example

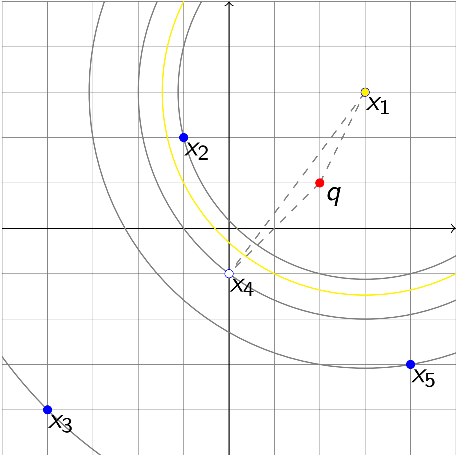
- Set $c := x_1$ and $s := x_2$
- As $2.236 \approx d(c, q) < d(s, q) \approx 3.162$, no new c
- As $4.123 \approx d(c, s) < 2 \cdot d(c, q) \approx 4.472$, no abort



Orchards Algorithm

Example

- Set $s := x_4$
- As $2.236 \approx d(c, q) < d(s, q) \approx 2.828$, no new c
- As $5 \approx d(c, s) > 2 \cdot d(c, q) \approx 4.472$, abort



Orchards Algorithm

Advantages and disadvantages

Advantages

- Faster as Full Search

Disadvantages

- Preprocessing needs large memory and runtime

Improvement

- Use MarkBits to ensure that no distance is calculated twice

Annulus Method

- Create a list for a random item $p^* \in S$ with all items $x \in S$, ordered ascending to the distance
- Choose random item $c \in S$
- Walk alternating away from p^* and back to it in the list
- If current item s has smaller distance to q as c, set $c := s$
- Current item s is under c:
  - If $d(p^*, s) < d(p^*, q) - d(c, q)$, ignore all items under s (Triangle inequality 3)
- Current item s is above c:
  - If $d(p^*, s) > d(p*, q) + d(c, q)$, ignore all items above s (Triangle inequality 3)
- c is the nearest neighbour if the entire list is traversed

Annulus Method

Example

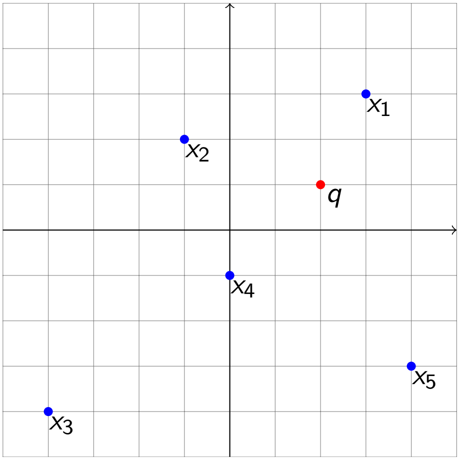
$\mathbb{U} = \mathbb{R}^2$

Items

$x_1 = (3, 3)$
$x_2 = (-1, 2)$
$x_3 = (-4, -4)$
$x_4 = (0, -1)$
$x_5 = (4, -3)$

Query item

$q = (2, 1)$



Annulus Method

Example

Distances

	x_1	x_2	x_3	x_4	x_5
x_5	≈ 6.083	≈ 7.071	≈ 8.062	≈ 4.472	0

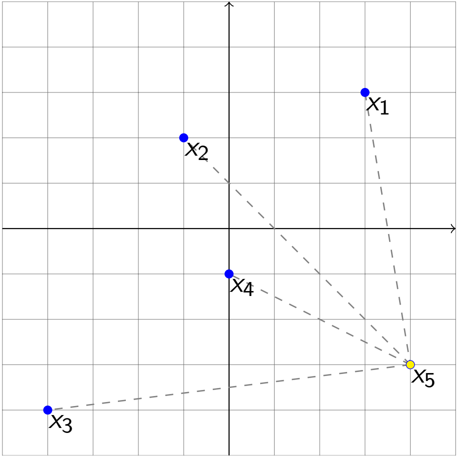
Annulus Method

Example

$p^* := x_5$ with
$d(p^*, q) \approx 4.472$

List

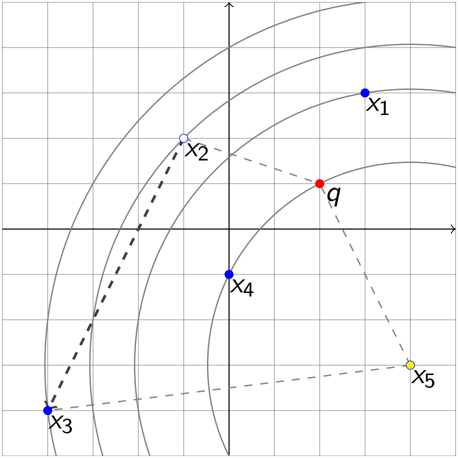
$L(x_5) =$
$\{x_5, x_4, x_1, x_2, x_3\}$



Annulus Method

Example

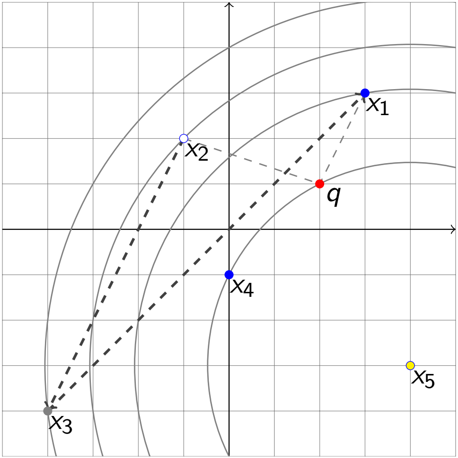
- Set $c := x_2$ with $d(c, q) \approx 3.162$
- Set $s := x_3$ with $d(s, q) \approx 7.810$
- $8.062 \approx d(p^*, s) > d(p^*, q) + d(c, q) \approx 7.634 \Rightarrow$ ignore items above x_3



Annulus Method

Example

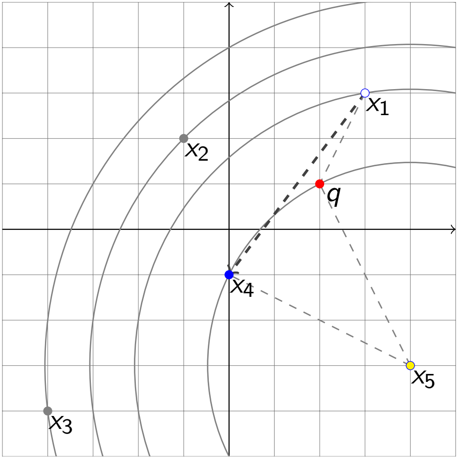
- Set $s := x_1$ with $d(s, q) \approx 2.236$
- As $d(s, q) < d(c, q)$, set $c := s$



Annulus Method

Example

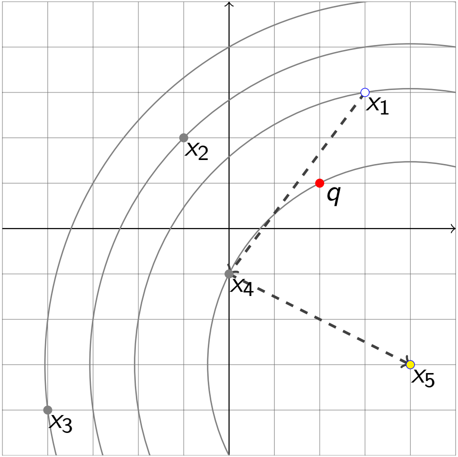
- Set $c := x_1$ with $d(c, q) \approx 2.236$
- Set $s := x_4$ with $d(s, q) \approx 2.828$
- $4.472 \approx d(p^*, s) > d(p^*, q) - d(c, q) \approx 2.236 \Rightarrow$ ignore no items



Annulus Method

Example

- Set $s := x_5$ with $d(s, q) \approx 4.472 < 2.236 \approx d(c, q)$
- End of list, $c = x_1$ nearest neighbour of q



Annulus Method

Advantages and disadvantages

Advantages

- Faster as Full Search
- Less memory usage than Orchards Algorithm

AESA

Approximating and Eliminating Search Algorithm

- Create matrix with all distances $d(x, y)$, with $x, y \in S$
- Every item is always in one status
  - *Known*, $d(x, q)$ is known
  - *Unknown*, only $d_P(x, q)$ is known
  - *Rejected*, $d_P(x, q)$ is bigger as smallest known distance r
- All $x \in S$ are *Unknown* and $d_P(x, q) = -\infty$
- Repeat until all $x \in S$ *Known* oder *Rejected*
  1. Choose *Unknown* item $x \in S$ with smallest $d_P(x, q)$
  2. Calculate $d(x, q)$, so that x gets *Known*
  3. Refresh the smallest known distance r
  4. Set $P := P \cup \{x\}$, refresh $d_P(x', q)$, if x' is *Unknown* mark x' as *Rejected*, if $d_P(x', q) > r$

LAESA

Linear Approximating and Eliminating Search Algorithm

- Works with a set of *pivot* items instead of a matrix
- Works best if *pivot* items are strongly separated

Outlook

- Metric trees
- . . .

Resources

- Otto Forster, 2006, *Analysis 2*, Friedr. Vieweg & Sohn Verlag
- Kenneth L. Clarkson, 2005, *Nearest-Neighbor Searching and Metric Space Dimensions*, http://kenclarkson.org/nn_survey/p.pdf