Embedding Spatial Information into Nearest Neighbor Joins

The 2nd Tameus Workshop,
Andrej Taliun

Outline

- Spatial Nearest Neighbor Join in Swiss Feed Database;
- A Brute Force Approach for the SNN-Join;
- SNN-Join with R-Tree;
- Target Issues.

Spatial Nearest Neighbor Join in Swiss Feed Database

Embedding Spatial Information into Nearest Neighbor Joins

- for each geographical object (point, polygon, ...) find another one which is the closest and belongs to the same input set.

origin	$Zn_{\{DM\}}$	*DM*	*N.N.*
Worb	27.48	924	*Allmid*
Juch	32.67	915	*Allmid*
Allmid		921	*Hubel*
Hubel	31.25	928	*Allmid*

"Hey samples are collected from Worb, Juch, Allmid and Hubel."

"Nutrients $Zn_{\{DM\}}$ (zinc) and DM (dry matter) are measured."

Spatial Nearest Neighbor Join in Swiss Feed Database

Embedding Spatial Information into Nearest Neighbor Joins

- We use Spatial Nearest Neighbor Join to compute two-dimensional Kernel regression of a selected nutrient:
  - Kernel regression is a statistical approach to estimate a continuous function that best fit the data, i.e.,
    - it is possible to evaluate the containment of a nutrient at any spatial point.
  - differently from linear or polynomial regression, Kernel regression does not assume any underlying distribution, i.e.,
    - can be applied to any data distribution.

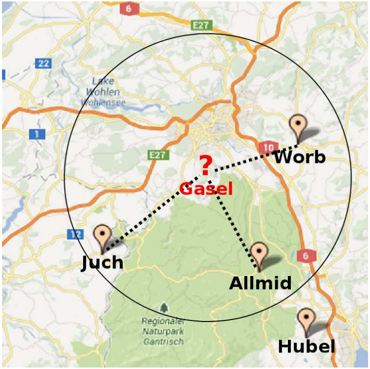
Spatial Nearest Neighbor Join in Swiss Feed Database

Embedding Spatial Information into Nearest Neighbor Joins

- "How much zinc, $Zn_{\{FS\}}$, contains hey near Gasel?"

- $Zn_{\{FS\}} = Zn_{\{DM\}} * DM/1000$
- there is no hey sample from Gasel, therefore,
  - evaluate Kernel regressions for nutrient $Zn_{\{DM\}}$,
  - evaluate Kernel regressions for nutrient DM,
  - apply the above formula

- "How much zinc, $Zn_{\{FS\}}$, contains hey near Gasel?"

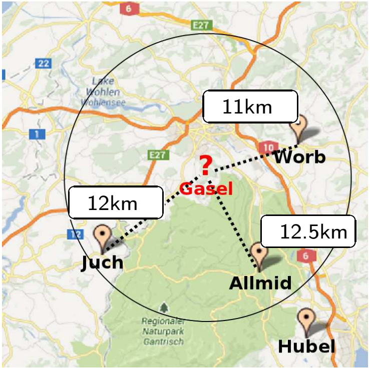


1. consider all hey samples which are within radius of 30km from Gasel:
   - the radius is computed from the variance of the spatial points;

Spatial Nearest Neighbor Join in Swiss Feed Database

Embedding Spatial Information into Nearest Neighbor Joins

- "How much zinc, $Zn_{\{FS\}}$, contains hey near Gasel?"

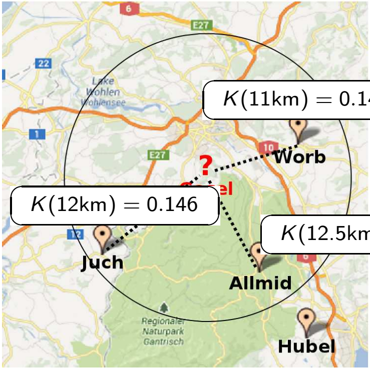


1. consider all hey samples which are within radius of 30km from Gasel:
   - the radius is computed from the variance of the spatial points;
2. for each sample compute the distance to Gasel:
   - from Worb to Gasel: 11 km
   - from Juch to Gasel: 12 km
   - from Allmid to Gasel: 12.5 km

Spatial Nearest Neighbor Join in Swiss Feed Database

Embedding Spatial Information into Nearest Neighbor Joins

- "How much zinc, $Zn_{\{FS\}}$, contains hey near Gasel?"



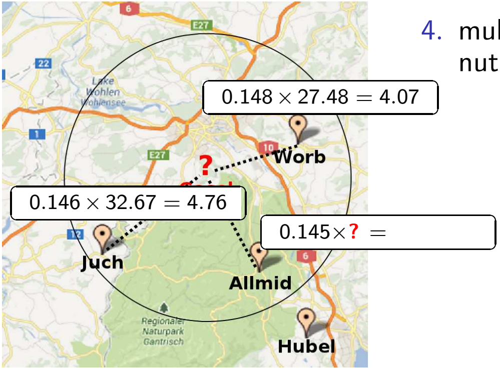
3. for each distance evaluate the kernel function:
   - kernel function assigns higher weights for smaller distances;
   - a typical choice is Gaussian kernel function:

$$K(\text{dist}) = \frac{1}{2\pi} e^{-\frac{\textbf{dist}^2}{2\times\textbf{radius}^2}}$$

Spatial Nearest Neighbor Join in Swiss Feed Database

Embedding Spatial Information into Nearest Neighbor Joins

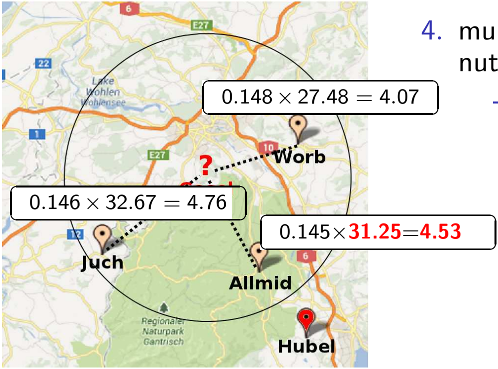
- "How much zinc, $Zn_{\{FS\}}$, contains hey near Gasel?"



4. multiply the kernel function by the nutritive value of the target sample:

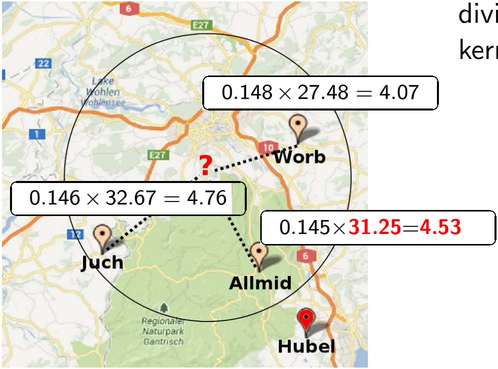
Spatial Nearest Neighbor Join in Swiss Feed Database
Embedding Spatial Information into Nearest Neighbor Joins

- "How much zinc, $Zn_{\{FS\}}$, contains hey near Gasel?"



4. multiply the kernel function by the nutritive value of the target sample:
   - if the nutritive value is not present, then, pick it up from the nearest neighbor;

- "How much zinc, $Zn_{\{FS\}}$, contains hey near Gasel?"



5. sum up the computed values and divide the result by the sum of the kernel functions:

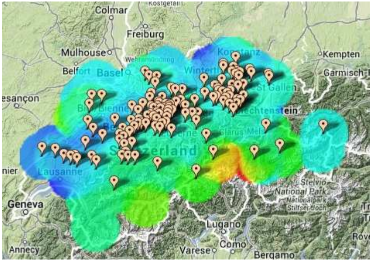
$$\frac{4.07 + 4.76 + 4.53}{0.148 + 0.146 + 0.145} = \frac{13.361}{0.439} = 30.43$$

- Kernel regression is sensitive to the number of distinct location:
  - according to [*Scott, D.W. (1992). Multivariate Density Estimation*] for the relative mean square error of 1% the number of distinct locations must be greater than 1000;
  - currently, there are 1050 distinct locations in the Swiss Feed Database.
- Spatial Nearest Neighbor Joins increases the number of locations and, therefore, improves the quality of the Kernel regression.

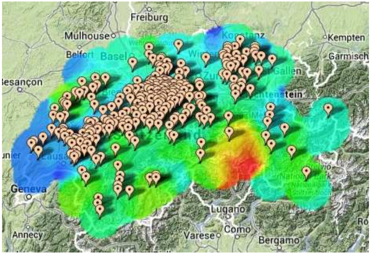
Spatial Nearest Neighbor Join in Swiss Feed Database

Embedding Spatial Information into Nearest Neighbor Joins

- "How much zinc contains hey across the Switzerland?"
  - the hey is collected from 876 distinct locations
  - hey samples from only 356 locations contain measurements of all required nutrients



- without spatial nearest neighbor join



- with spatial nearest neighbor join

- Before computing the Kernel Regression we complete the data by substituting missing measures with a help of spatial nearest neighbor join:
  - **Step One**: compute two views:
    - *stored_measures* - collects measures which are stored in the fact table;
    - *absent_measures* - for each feed sample collects nutrients without a measure in the fact table;
  - **Step Two**: compute the spatial nearest neighbor join between *missing_measures* and *absent_measures*;

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

- All nutrient measurements are stored in a vertically partitioned fact table, i.e., one nutritive measure per row:

sample_id	*origin*	*nutrient*	*quantity*
121-1	Worb	$Zn_{\{DM\}}$	27.48
121-1	Worb	DM	924
300-4	Juch	$Zn_{\{DM\}}$	32.67
300-4	Juch	DM	915
103-0	Allmid	DM	921
555-5	Hubel	$Zn_{\{DM\}}$	31.25
555-5	Hubel	DM	928
787-9	Toffen	P	3.07
784-3	Ami	Mg	2.85
253-0	Ball	P	3.11

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

- consider $Zn_{\{FS\}} = Zn_{\{DM\}} * DM/1000$;
- *stored_measures* is computed with a simple SQL statement.

SELECT * FROM fact_table WHERE nutrient IN ('$Zn_{\{DM\}}$','DM')

---

sample_id	*origin*	*nutrient*	*quantity*
121-1	Worb	$Zn_{\{DM\}}$	27.48
300-4	Juch	$Zn_{\{DM\}}$	32.67
555-5	Hubel	$Zn_{\{DM\}}$	31.25

Table: *stored_measures*

	n	*nutrient*	*quantity*
	b	$Zn_{\{DM\}}$	27.48
	b	*DM*	924
		$Zn_{\{DM\}}$	32.67
300-4	Juch	*DM*	915
103-0	Allmid	*DM*	921
555-5	Hubel	$Zn_{\{DM\}}$	31.25
555-5	Hubel	*DM*	928
787-9	Toffen	P	3.07
784-3	Ami	Mg	2.85
253-0	Ball	P	3.11

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

sample_id	*origin*	*nutrient*	*quantity*
121-1	Worb	$Zn_{\{DM\}}$	27.48
300-4	Juch	$Zn_{\{DM\}}$	32.67
555-5	Hubel	$Zn_{\{DM\}}$	31.25
121-1	Worb	*DM*	924
300-4	Juch	*DM*	915
103-0	Allmid	*DM*	921
555-5	Hubel	*DM*	928

Table: *stored_measures*

	n	*nutrient*	*quantity*
	b	$Zn_{\{DM\}}$	27.48
	b	*DM*	924
	h	$Zn_{\{DM\}}$	32.67
	h	*DM*	915
	id	*DM*	921
	el	$Zn_{\{DM\}}$	31.25
555-5	Hubel	*DM*	928
787-9	Toffen	P	3.07
784-3	Ami	Mg	2.85
253-0	Ball	P	3.11

- consider $Zn_{\{FS\}} = Zn_{\{DM\}} * DM/1000$;
- *absent_measures* is computed in two steps:
  - compute all possible combinations between distinct sample ids and required nutrient names;

SELECT * FROM (select distinct sample_id from fact_table) *samples*, unnest($Zn_{\{DM\}}, DM$) *nutrient*

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

- consider $Zn_{\{FS\}} = Zn_{\{DM\}} * DM/1000$;
- *absent_measures* is computed in two steps:
  - compute all possible combinations between distinct sample ids and required nutrient names;
  - remove those combination which are present in *stored_measures*.

SELECT * FROM (select distinct sample_id from fact_table) *samples*,
unnest($Zn_{\{DM\}}, DM$) *nutrient*)

except

(SELECT sample_id, nutrient FROM stored_measures)

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

sample_id	*origin*	*nutrient*	*quantity*
121-1	Worb	$Zn_{\{DM\}}$	27.48
300-4	Juch	$Zn_{\{DM\}}$	32.67
555-5	Hubel	$Zn_{\{DM\}}$	31.25
121-1	Worb	DM	924
300-4	Juch	DM	915
103-0	Allmid	DM	921
555-5	Hubel	DM	928

Table: *stored_measures*

555-5	Hubel
787-9	Toffen
784-3	Ami
253-0	Ball

sample_id	*origin*	*nutrient*	*quantity*
121-1	Worb	$Zn_{\{DM\}}$	?
121-1	Worb	DM	?
300-4	Juch	$Zn_{\{DM\}}$	?
300-4	Juch	DM	?
103-0	Allmid	$Zn_{\{DM\}}$	?
103-0	Allmid	DM	?
555-5	Hubel	$Zn_{\{DM\}}$	?
555-5	Hubel	DM	?
787-9	Toffen	$Zn_{\{DM\}}$	?
787-9	Toffen	DM	?
784-3	Ami	$Zn_{\{DM\}}$	?
784-3	Ami	DM	?
253-0	Ball	$Zn_{\{DM\}}$	?
253-0	Ball	DM	?

Table: *absent_measures*

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

sample_id	*origin*	*nutrient*	*quantity*
121-1	Worb	$Zn_{\{DM\}}$	27.48
300-4	Juch	$Zn_{\{DM\}}$	32.67
555-5	Hubel	$Zn_{\{DM\}}$	31.25
121-1	Worb	*DM*	924
300-4	Juch	*DM*	915
103-0	Allmid	*DM*	921
555-5	Hubel	*DM*	928

Table: *stored_measures*

555-5	Hubel
787-9	Toffen
784-3	Ami
253-0	Ball

sample_id	*origin*	*nutrient*	*quantity*
121-1	Worb	$Zn_{\{DM\}}$	?
121-1	Worb	*DM*	?
300-4	Juch	$Zn_{\{DM\}}$	?
300-4	Juch	*DM*	?
103-0	Allmid	$Zn_{\{DM\}}$	?
103-0	Allmid	*DM*	?
555-5	Hubel	$Zn_{\{DM\}}$	?
555-5	Hubel	*DM*	?
787-9	Toffen	$Zn_{\{DM\}}$	?
787-9	Toffen	*DM*	?
784-3	Ami	$Zn_{\{DM\}}$	?
784-3	Ami	*DM*	?
253-0	Ball	$Zn_{\{DM\}}$	?
253-0	Ball	*DM*	?

Table: *absent_measures*

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

sample_id	*origin*	*nutrient*	*quantity*
121-1	Worb	$Zn_{\{DM\}}$	27.48
300-4	Juch	$Zn_{\{DM\}}$	32.67
555-5	Hubel	$Zn_{\{DM\}}$	31.25
121-1	Worb	*DM*	924
300-4	Juch	*DM*	915
103-0	Allmid	*DM*	921
555-5	Hubel	*DM*	928

Table: *stored_measures*

sample_id	*origin*	*nutrient*	*quantity*
103-0	Allmid	$Zn_{\{DM\}}$	?
787-9	Toffen	$Zn_{\{DM\}}$	?
787-9	Toffen	*DM*	?
784-3	Ami	$Zn_{\{DM\}}$	?
784-3	Ami	*DM*	?
253-0	Ball	$Zn_{\{DM\}}$	?
253-0	Ball	*DM*	?

Table: *absent_measures*

555-5	Hubel	*DM*	928
787-9	Toffen	P	3.07
784-3	Ami	Mg	2.85
253-0	Ball	P	3.11

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

- A naive approach is to do an exhaustive search of the whole space comparing distances and sorting them:
  - for each entry in *absent_measures* compute the distances to all samples in *stored_measures* with the same nutrient;
  - sort samples in *stored_measures* in ascending order of the computed distances;
  - select the first sample of the sorted *stored_measures*.

SELECT A.sample_id, A.nutrient, (select B.quantity FROM *stored_measures* B WHERE A.nutrient =B.nutrient ORDER BY st_distance(A.origin, B.origin) LIMIT 1)

FROM *absent_measures* A

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

- The brute force approach is to do an exhaustive search of the whole space comparing distances and sorting them:
  - for each entry in *absent_measures* compute the distances to all samples in *stored_measures* with the same nutrient;;
  - sort samples in *stored_measures* in ascending order of the computed distances;
  - select the first sample of the sorted *stored_measures*.

SELECT DISTINCT ON (A.sample_id), A.nutrient, B.quantity FROM

FROM *absent_measures* A, *stored_measures* B

WHERE A.nutrient =B.nutrient

ORDER BY A.sample_id, st_distance(A.origin, B.origin)

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

sample_id	*origin*	*nutrient*	*quantity*
103-0	Allmid	$Zn_{\{DM\}}$	?
787-9	Toffen	DM	?
787-9	Toffen	$Zn_{\{DM\}}$	?
784-3	Ami	$Zn_{\{DM\}}$	?
784-3	Ami	DM	?
253-0	Ball	$Zn_{\{DM\}}$	?
253-0	Ball	DM	?

Table: *absent_measures*

sample_id	*origin*	*nutrient*	*quantity*	*dist*
121-1	Worb	$Zn_{\{DM\}}$	27.48	
300-4	Juch	$Zn_{\{DM\}}$	32.67	
555-5	Hubel	$Zn_{\{DM\}}$	31.25	
121-1	Worb	DM	924	
300-4	Juch	DM	915	
103-0	Allmid	DM	921	
555-5	Hubel	DM	928	

Table: *stored_measures*

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

sample_id	*origin*	*nutrient*	*quantity*
103-0	Allmid	$Zn_{\{DM\}}$	?
787-9	Toffen	DM	?
787-9	Toffen	$Zn_{\{DM\}}$	?
784-3	Ami	$Zn_{\{DM\}}$	?
784-3	Ami	DM	?
253-0	Ball	$Zn_{\{DM\}}$	?
253-0	Ball	DM	?

Table: *absent_measures*

sample_id	*origin*	*nutrient*	*quantity*	*dist*
121-1	Worb	$Zn_{\{DM\}}$	27.48	11.1km
300-4	Juch	$Zn_{\{DM\}}$	32.67	12.2km
555-5	Hubel	$Zn_{\{DM\}}$	31.25	8.7km
121-1	Worb	DM	924	
300-4	Juch	DM	915	
103-0	Allmid	DM	921	
555-5	Hubel	DM	928	

Table: *stored_measures*

A Brute Force Approach for the SNN-Join
Embedding Spatial Information into Nearest Neighbor Joins

sample_id	*origin*	*nutrient*	*quantity*
103-0	Allmid	$Zn_{\{DM\}}$	?
787-9	Toffen	*DM*	?
787-9	Toffen	$Zn_{\{DM\}}$	?
784-3	Ami	$Zn_{\{DM\}}$	?
784-3	Ami	*DM*	?
253-0	Ball	$Zn_{\{DM\}}$	?
253-0	Ball	*DM*	?

Table: *absent_measures*

sample_id	*origin*	*nutrient*	*quantity*	*dist*
555-5	Hubel	$Zn_{\{DM\}}$	31.25	8.7km
121-1	Worb	$Zn_{\{DM\}}$	27.48	11.1km
300-4	Juch	$Zn_{\{DM\}}$	32.67	12.2km
121-1	Worb	*DM*	924	
300-4	Juch	*DM*	915	
103-0	Allmid	*DM*	921	
555-5	Hubel	*DM*	928	

Table: *stored_measures*

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

sample_id	*origin*	*nutrient*	*quantity*
103-0	Allmid	$Zn_{\{DM\}}$	31.25
787-9	Toffen	*DM*	?
787-9	Toffen	$Zn_{\{DM\}}$	?
784-3	Ami	$Zn_{\{DM\}}$	?
784-3	Ami	*DM*	?
253-0	Ball	$Zn_{\{DM\}}$	?
253-0	Ball	*DM*	?

Table: *absent_measures*

sample_id	*origin*	*nutrient*	*quantity*	*dist*
555-5	Hubel	$Zn_{\{DM\}}$	31.25	8.7km
121-1	Worb	$Zn_{\{DM\}}$	27.48	11.1km
300-4	Juch	$Zn_{\{DM\}}$	32.67	12.2km
121-1	Worb	*DM*	924	
300-4	Juch	*DM*	915	
103-0	Allmid	*DM*	921	
555-5	Hubel	*DM*	928	

Table: *stored_measures*

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

sample_id	*origin*	*nutrient*	*quantity*
103-0	Allmid	$Zn_{\{DM\}}$	31.25
787-9	Toffen	*DM*	?
787-9	Toffen	$Zn_{\{DM\}}$	?
784-3	Ami	$Zn_{\{DM\}}$	?
784-3	Ami	*DM*	?
253-0	Ball	$Zn_{\{DM\}}$	?
253-0	Ball	*DM*	?

Table: *absent_measures*

sample_id	*origin*	*nutrient*	*quantity*	*dist*
121-1	Worb	$Zn_{\{DM\}}$	27.48	
300-4	Juch	$Zn_{\{DM\}}$	32.67	
555-5	Hubel	$Zn_{\{DM\}}$	31.25	
121-1	Worb	*DM*	924	
300-4	Juch	*DM*	915	
103-0	Allmid	*DM*	921	
555-5	Hubel	*DM*	928	

Table: *stored_measures*

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

sample_id	*origin*	*nutrient*	*quantity*
103-0	Allmid	$Zn_{\{DM\}}$	31.25
787-9	Toffen	*DM*	?
787-9	Toffen	$Zn_{\{DM\}}$	?
784-3	Ami	$Zn_{\{DM\}}$	?
784-3	Ami	*DM*	?
253-0	Ball	$Zn_{\{DM\}}$	?
253-0	Ball	*DM*	?

Table: *absent_measures*

sample_id	*origin*	*nutrient*	*quantity*	*dist*
121-1	Worb	$Zn_{\{DM\}}$	27.48	
300-4	Juch	$Zn_{\{DM\}}$	32.67	
555-5	Hubel	$Zn_{\{DM\}}$	31.25	
121-1	Worb	*DM*	924	9.2km
300-4	Juch	*DM*	915	15.4km
103-0	Allmid	*DM*	921	8.4km
555-5	Hubel	*DM*	928	14.1km

Table: *stored_measures*

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

sample_id	*origin*	*nutrient*	*quantity*
103-0	Allmid	$Zn_{\{DM\}}$	31.25
787-9	Toffen	*DM*	?
787-9	Toffen	$Zn_{\{DM\}}$	?
784-3	Ami	$Zn_{\{DM\}}$	?
784-3	Ami	*DM*	?
253-0	Ball	$Zn_{\{DM\}}$	?
253-0	Ball	*DM*	?

Table: *absent_measures*

sample_id	*origin*	*nutrient*	*quantity*	*dist*
121-1	Worb	$Zn_{\{DM\}}$	27.48	
300-4	Juch	$Zn_{\{DM\}}$	32.67	
555-5	Hubel	$Zn_{\{DM\}}$	31.25	
103-0	Allmid	*DM*	921	8.4km
121-1	Worb	*DM*	924	9.2km
555-5	Hubel	*DM*	928	14.1km
300-4	Juch	*DM*	915	15.4km

Table: *stored_measures*

A Brute Force Approach for the SNN-Join
Embedding Spatial Information into Nearest Neighbor Joins

sample_id	*origin*	*nutrient*	*quantity*
103-0	Allmid	$Zn_{\{DM\}}$	31.25
787-9	Toffen	*DM*	921
787-9	Toffen	$Zn_{\{DM\}}$	?
784-3	Ami	$Zn_{\{DM\}}$	?
784-3	Ami	*DM*	?
253-0	Ball	$Zn_{\{DM\}}$	?
253-0	Ball	*DM*	?

Table: *absent_measures*

sample_id	*origin*	*nutrient*	*quantity*	*dist*
121-1	Worb	$Zn_{\{DM\}}$	27.48	
300-4	Juch	$Zn_{\{DM\}}$	32.67	
555-5	Hubel	$Zn_{\{DM\}}$	31.25	
103-0	Allmid	*DM*	921	8.4km
121-1	Worb	*DM*	924	9.2km
555-5	Hubel	*DM*	928	14.1km
300-4	Juch	*DM*	915	15.4km

Table: *stored_measures*

A Brute Force Approach for the SNN-Join

Embedding Spatial Information into Nearest Neighbor Joins

- Why the exhaustive search is a problem:
  - it has quadratic runtime complexity, i.e. $O(n^2)$. It takes more than 8 min to compute the spatial nearest neighbor join between 1050 locations.
  - accurate computation of a distance between two spatial points is expensive since the Earth is not flat, i.e., expensive trigonometric functions are involved.

SNN-Join with R-Tree

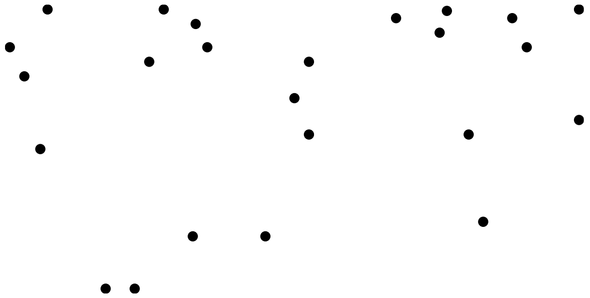
Embedding Spatial Information into Nearest Neighbor Joins

- An alternative approach to the exhaustive search is to use an indexing structure such as R-Tree.

SNN-Join with R-Tree

Embedding Spatial Information into Nearest Neighbor Joins

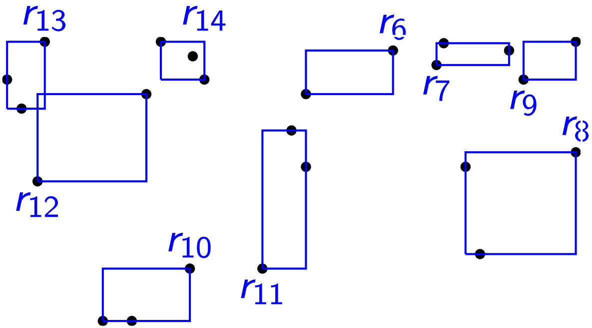
- The R-Tree partitions spatial data with a help of **minimum bounding rectangles**, i.e. MBR's:
  - MBR at the bottom of the R-Tree encloses spatial objects;
  - MBR at a higher level encloses MBR's from the previous level;
  - MBR's can overlap.



SNN-Join with R-Tree

Embedding Spatial Information into Nearest Neighbor Joins

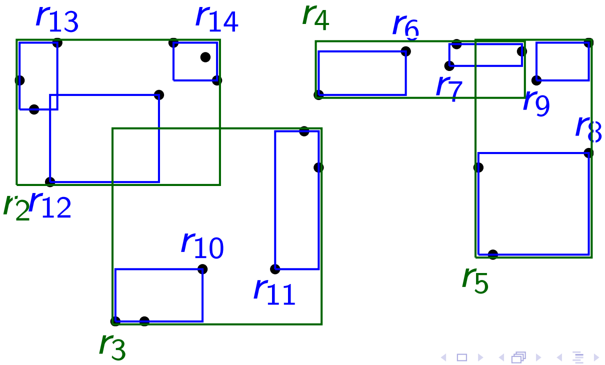
- The R-Tree partitions spatial data with a help of **minimum bounding rectangles**, i.e. MBR's:
  - MBR at the bottom of the R-Tree encloses spatial objects;
  - MBR at a higher level encloses MBR's from the previous level;
  - MBR's can overlap.



SNN-Join with R-Tree

Embedding Spatial Information into Nearest Neighbor Joins

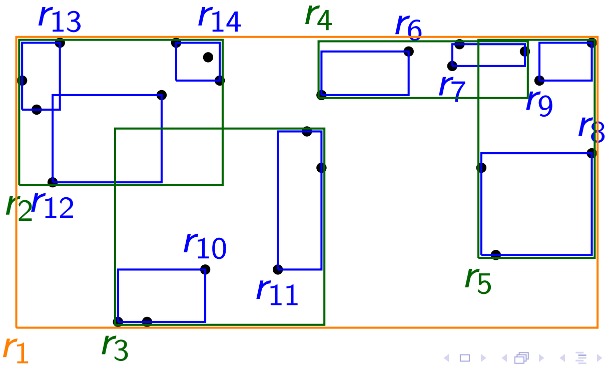
- The R-Tree partitions spatial data with a help of **minimum bounding rectangles**, i.e. MBR's:
  - MBR at the bottom of the R-Tree encloses spatial objects;
  - MBR at a higher level encloses MBR's from the previous level;
  - MBR's can overlap.



SNN-Join with R-Tree

Embedding Spatial Information into Nearest Neighbor Joins

- The R-Tree partitions spatial data with a help of **minimum bounding rectangles**, i.e. MBR's:
  - MBR at the bottom of the R-Tree encloses spatial objects;
  - MBR at a higher level encloses MBR's from the previous level;
  - MBR's can overlap.



SNN-Join with R-Tree

- The idea of the nearest neighbor search with the R-Tree:
  - traverse the tree depth-first starting from the root MBR;
  - visit an MBR only when necessary.

- The approach to find the nearest neighbor with the R-Tree relies on **the priority queue**:
  - **the priority queue** consists of the promising MBRs, i.e., from the MBRs which are likely to contain the nearest neighbor;
  - the MBRs in **the priority queue** are ordered based on the expectation to contain the nearest neighbor.

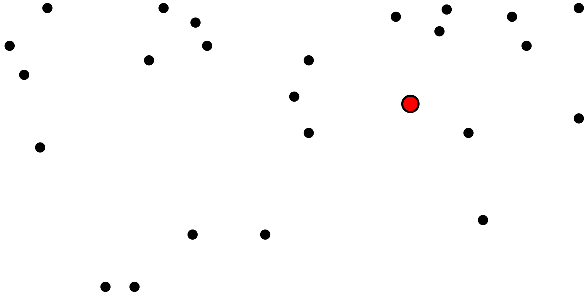
The Algorithm to find the nearest neighbor:

1. compute **the priority queue** from the MBRs which are children of the root node;
2. iterate through the **the priority queue** until is empty:
    2.1 if a child MBR is a leaf node, then, compute the distance and update the best nearest neighbor so far;
    2.2 if a child MBR is not a leaf node, then, initialize another **priority queue** and apply steps 2. – 3.;
3. based on the current nearest neighbor reduce **the priority queue** of the parent MBR ;

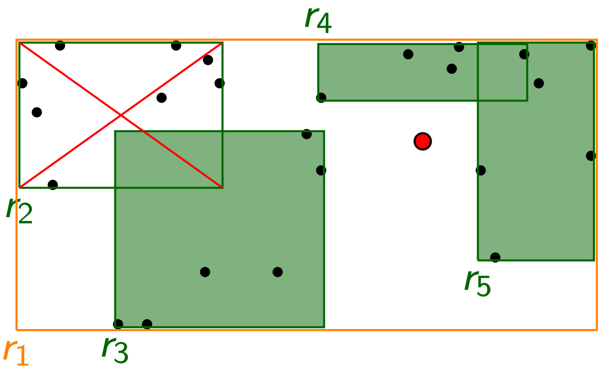
SNN-Join with R-Tree

Embedding Spatial Information into Nearest Neighbor Joins

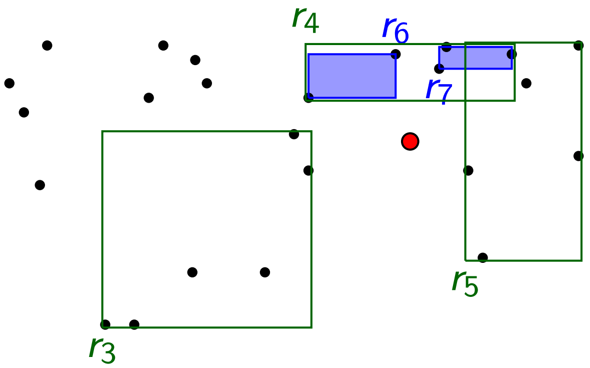
"Which data point is the closest to the red spot?"



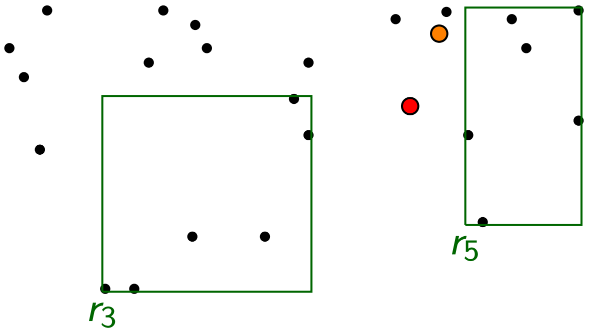
1. Compute **the priority queue** for MBRs of the root node:
   - r_2 is pruned since it is too far away from the red spot;
   - the remaining are ordered as $\{r_4, r_5, r_3\}$.



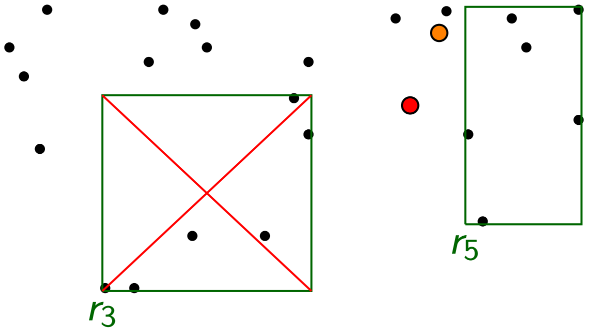
2. Compute **the priority queue** for r_4:
   - $\{r_6, r_7\}$;



2. Compute **the priority queue** for r_4:
   - $\{r_6, r_7\}$;
3. Compute the distances to data points enclosed by r_6 and r_7. Update the nearest neighbor.

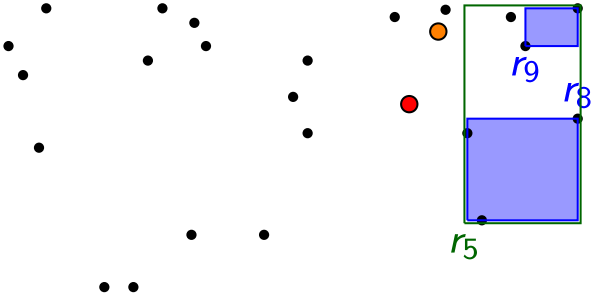


4. Remove r_3 from **the priority queue** of the root note since it is more distant from the red spot than the current nearest neighbor.



SNN-Join with R-Tree
Embedding Spatial Information into Nearest Neighbor Joins

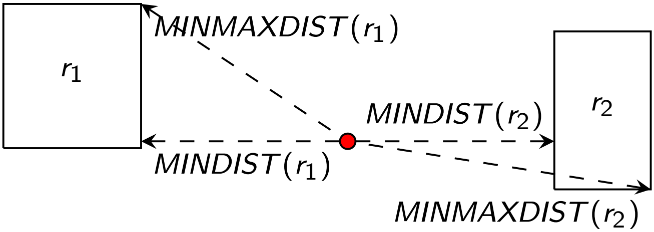
5. Continue with the **the priority queue** of r_5...



SNN-Join with R-Tree

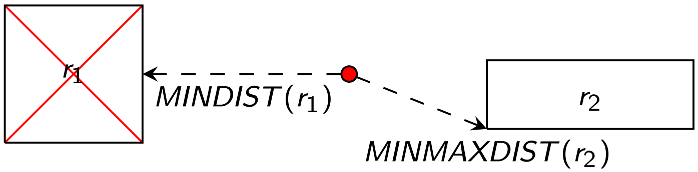
Embedding Spatial Information into Nearest Neighbor Joins

- Computation of **the priority queue** is based on distances **MINDIST** and **MINMAXDIST** between the target point and MBRs.

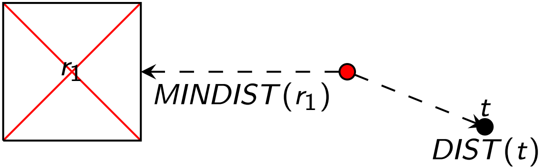


- $MINDIST(r)$ is the smallest possible **lower bound** and $MINMAXDIST(r)$ is the smallest possible **upper bound** for a distance between a data point enclosed by r and the target point;

- **Downward pruning:** r_1 is discarded if there exists another r_2 such, that $MINDIST(r_1) > MINMAXDIST(r_2)$.



- **Upward pruning:** r_1 is discarded if there exists a data point t such, that $MINDIST(r_1) > DIST(t)$.



SNN-Join with R-Tree

Embedding Spatial Information into Nearest Neighbor Joins

- Indexing of spatial data with R-Tree dramatically decreases the computation time of the SNN-Join;

	without R-Tree	with R-Tree
1% of measures are missing	< 1 sec.	< 1 sec.
10% of measures are missing	< 1 min.	< 2 sec.
50% of measures are missing	> 5 min.	10 sec.

- However, the traversing of the R-Tree is not optimal:
  - MBRs inside the priority queue are ordered according to MINDIST or MINMAXDIST;
  - in both cases unnecessary MBRs can be visited.

Target Issues

- Simultaneous computation of the SNN-Join for multiple nutrients;
  - Can we from one spatially nearest sample collect all required nutrients?
- Computation of the Kernel regression with the R-Tree:
  - for each location find all the samples within the given radius and, at the same time, find the nearest neighbor in case of missing measure.

Thank You