DISTRIBUTION APPROXIMATION – Cont'd.

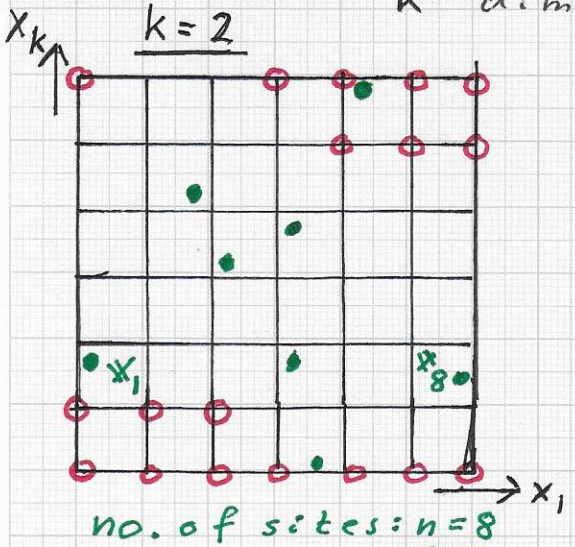
– Efficient Data Structure and Computation

→ See also: Discrete Sibson Interpolation, by Park et al., IEEE TVCG 12(2).

- Given: $\{ \mathbf{x}_i = (x_1^i, \ldots, x_k^i) \mid i = 1, \ldots n \}$
  = set of sites in k-dim. feature space with associated density values ϱ_i, $i = 1, \ldots n$

- Wanted: ρ-function $\rho(\mathbf{x})$ such that $\rho(\mathbf{x}_i) = \varrho_i$, $i = 1, \ldots, n$, and a DISCRETE representation of $\rho(\mathbf{x})$, defined by computing values of the Sibson interpolation function for a Cartesian grid (of k-dim. feature space and resolution N^k)

- Ex.:



no. of sites: $n = 8$

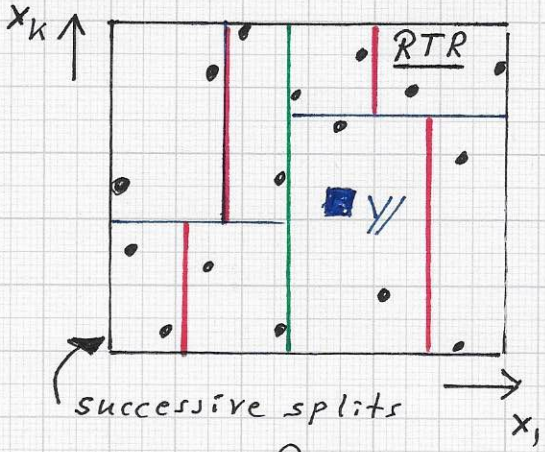
res. of ρ-grid: $7^2 = N^2$

→ $N^k \gg n$

- Approach: DISCRETIZE the region in k-dim. space (where $\rho(\mathbf{x})$ should be evaluated) by N^k grid points (corners of hyper-cube grid cells); DO NOT COMPUTE OR REPRESENT THE VORONOI DIAGRAM FOR $\{\mathbf{x}_i\}$ – USE A "DIRECT" METHOD TO COMPUTE VALUES AT GRID POINTS!

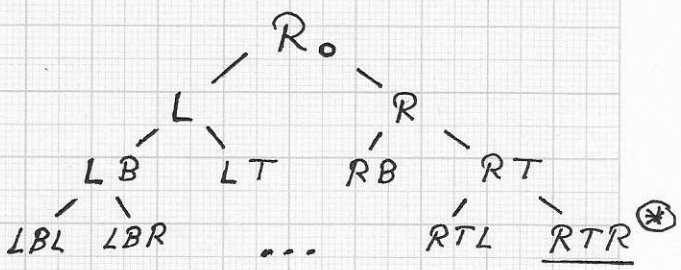
DISTRIBUTION APPROXIMATION – Cont'd

→ Use OPTIMAL data structure to store the sites $*_i$, $i = 1, \ldots, n$:
k-d tree used to order the sites.



successive splits of region R_0 in "left-right" and "bottom-top" sub-regions

Construction of tree:

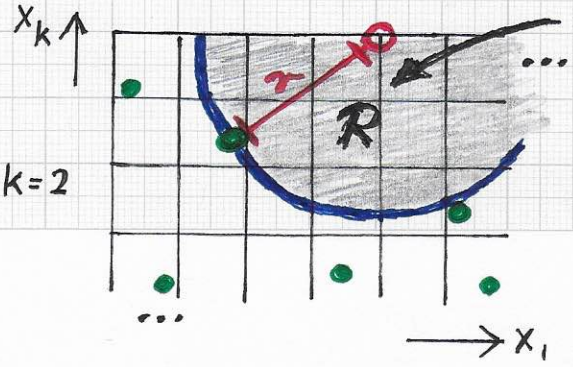


L = left
R = right
B = bottom
T = top
⊛ RTR = right-top-right

⇒ OPTIMALITY:

1) Tree construction complexity: $O(n \log n)$
2) Tree usage = search complexity: $O(\log n)$
(e.g., to find ■ y in example

→ Computing the DISCRETE SIBSON interpolation



region R associated with ○ values
○ specific vertex of p-grid
● original site from $\{x_i, s_i\}$
r minimal distance between ○ and all ●
hyper-sphere with center ○ and radius r; boundary of R

■ DISTRIBUTION APPROXIMATION – Cont'd

Algorithm: Computing p-values at p-grid (evaluation) vertices $V_{\vec{i}}$ – where a vertex V in k-dim. feature space has a k-dim. multi-index $\vec{i}$, referring to a grid point '○'

- **Initialization:** all vertices $V_{\vec{i}}$ have "function value 0" and "counter value 0".
- **Iterative accumulation** of "function values":

DISCRETE SIBSON

■ for each p-grid vertex $V_{\vec{i}}$ do

LOOP

- determine original site x_j (with value q_j) with minimal distance to $V_{\vec{i}}$; this minimal distance is r. (Use k-d tree!)
- determine the set of p-grid vertices inside the region R in k-dim. feature space that, where R is bounded by the hyper-sphere with center point $V_{\vec{i}}$ and radius r.
- add the value q_j to all the "function values" of all p-grid vertices in this set. (Increment "counter values" of all vertices in this set!)

Result: ■ for each $V_{\vec{i}}$ do: $P(V_{\vec{i}})$ = accumulated Value / counter Value

DISTRIBUTION APPROXIMATION - Cont'd.

→ Considerations and Issues:

1) This "DISCRETE SIBSON" algorithm can be viewed as OPTIMALLY EFFICIENT when employing the k-d tree data structure.
2) The model of a specific material type/class is obtained by computing and storing the discrete Sibson interpolation function, defined by all available "training samples" $*_i$ belonging to this class.
3) This model is the discretized representation of the p-function, approximated via the discrete Sibson step-
4) To determine whether a NEW scanned object is of material type/class of a class for which a "model has been trained", one must (i) compute the p-function of the NEW object, $p(*)$, and (ii) compare $p(*)$ with all known model p-functions $p_1(*), \ldots, p_M(*)$, with M being the number of trained models.

CLASSIFICATION

• NOTE: One must decide whether to use an integer-based or floating-point-based approach for parameters $(*_i, w_i, R, \ldots)$