k-Nearest Neighbor

Grundidee 1951 aus Statistik: nonparametric discriminatory Analysis und Mustererkennung.

Annahme: Punkte im geom Raum bilden Cluster und nahe liegende Punkte bilden Klasse. (Mustererkennung - numerische Verktoren).

Klassifizierung eines Musters durch Ausgabe der Klassifikation des near.neighbor Ähnlichkeitsmaß wird verwendet.

Geom Interpretation von Ähnlichkeit (räumliche Nähe)

effizienteres Retrieval durch Gewichtung oder besser: durch Indexierung (k-d trees)

k-Nearest Neighbor

$x = (a_1(x), \ldots, a_n(x))$

distance $d(x_1, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - a_r(x_j))^2}$

For discrete valued function f

Training

- For each training example $(x, f(x))$ add it to *trainings examples*

bf Classification

- Given query x_q to be classified
  - Let $x_1 \ldots x_k$ be k nearest instances from *trainings examples*
  - Return
    $\overline{f}(x_q) \leftarrow argmax_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$,
    where $\delta(a, b) = 1$ if $a = b$, $\delta(a, b) = 0$ otherwise

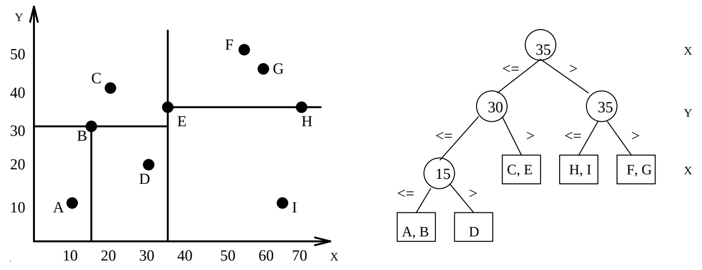
---

For real valued functions f:

...

$\overline{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$

Generierung von kd-Trees



`CreateTree`(CB)

Input:CB, Ähnlichkeitsmaß sim
Output: kd-tree

begin
if Split?(CB,b) **then** return(`MakeTerminalNode`(CB))
else

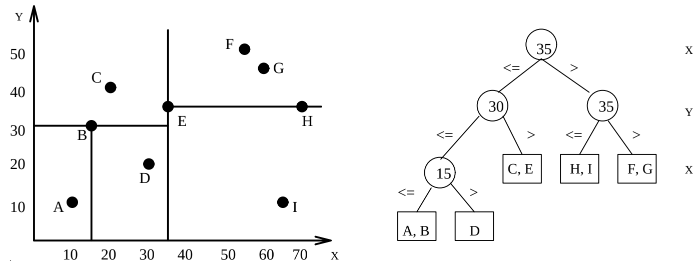
D := `SelectAttribute(CB)`

p_i := `SelectValue`(D,CB)

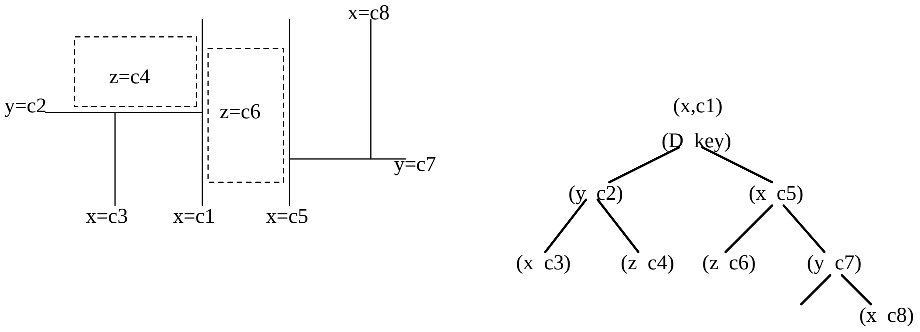
return `MakeInternalNode`(D,p_i,`CreateTree`(LowerPartition), `CreateTree`(UpperPartition))

end

Retrieval mit k-d trees



Generierung von kd-Trees



Effizienter Retrieval

- Retrieval aus Fallbasis, ohne backtracking
- Retrieval mit k-d trees und Nearest Neighbour, mit backtracking

Retrieval mit k-d trees

- Idee: k-dim Intervallschachtelung
- exakter Match eweitert zu NN-match
- Anfrage Q wird als Mittelpunkt einer k-dim Hyperkugel H aufgefasst. Radius wird so gewählt, daß alle aktuell bekannten m NN SQ(i) von Q umfaßt werden und SQ(m) auf Oberfläche der H. H ist eine dynamische Struktur.
- Partitionierungswerte p_i beschreiben Hyperebenen die senkrecht zu i-Achse .
- Bei Retrieval Beschränkung auf solche Elemente p aus CB für die gilt: $d(Q,p) \leq d((Q, SQ(m))$.
- Wenn diese Teilmenge m Elemente hat, terminiere. Sonst suche weiter.

k-d Retrieval Algorithmus

Gehe rekursiv durch k-d tree, um die K nearest neighbours zum Target zu finden.
Input: Target mit Werten von q Attributen, K or min.Ähnlichkeit, k-d tree
Output: die K ähnlichsten Fälle

- wenn Knoten final, dann modifiziere Prioritätenliste danach, wie ähnlich das Target ist
- wenn Knoten nicht final, iteriere über Sohn, der durch gegenwärtigen Attributwert spezifiziert im Target
- Bound-Overlap-Ball Boolean (BOB) Test bestimmt, ob man andere Söhne inspizieren muß. Wenn nein, dann enthält Partition des anderen Sohnes keine k-nearest neighbours. Wenn JA iteriere auch über diesen Knoten.
- Ball-Within-Bounds Boolean (BWB) Test ob alle k nearest neighbours gefunden, terminiere. Wenn nein, dann Suche auch in vorangegangenen Knoten im Baum. (Backtracking)

BILDER BOB BWB