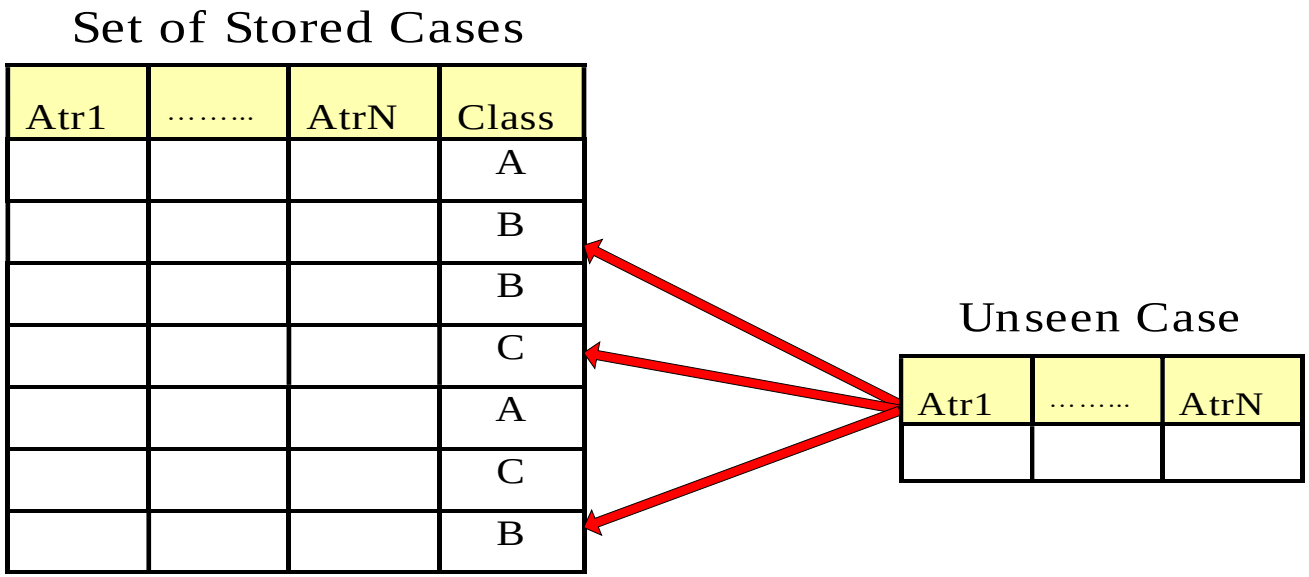
Classificatori K-NN

Prof. Matteo Golfarelli
Alma Mater Studiorum - Università di Bologna

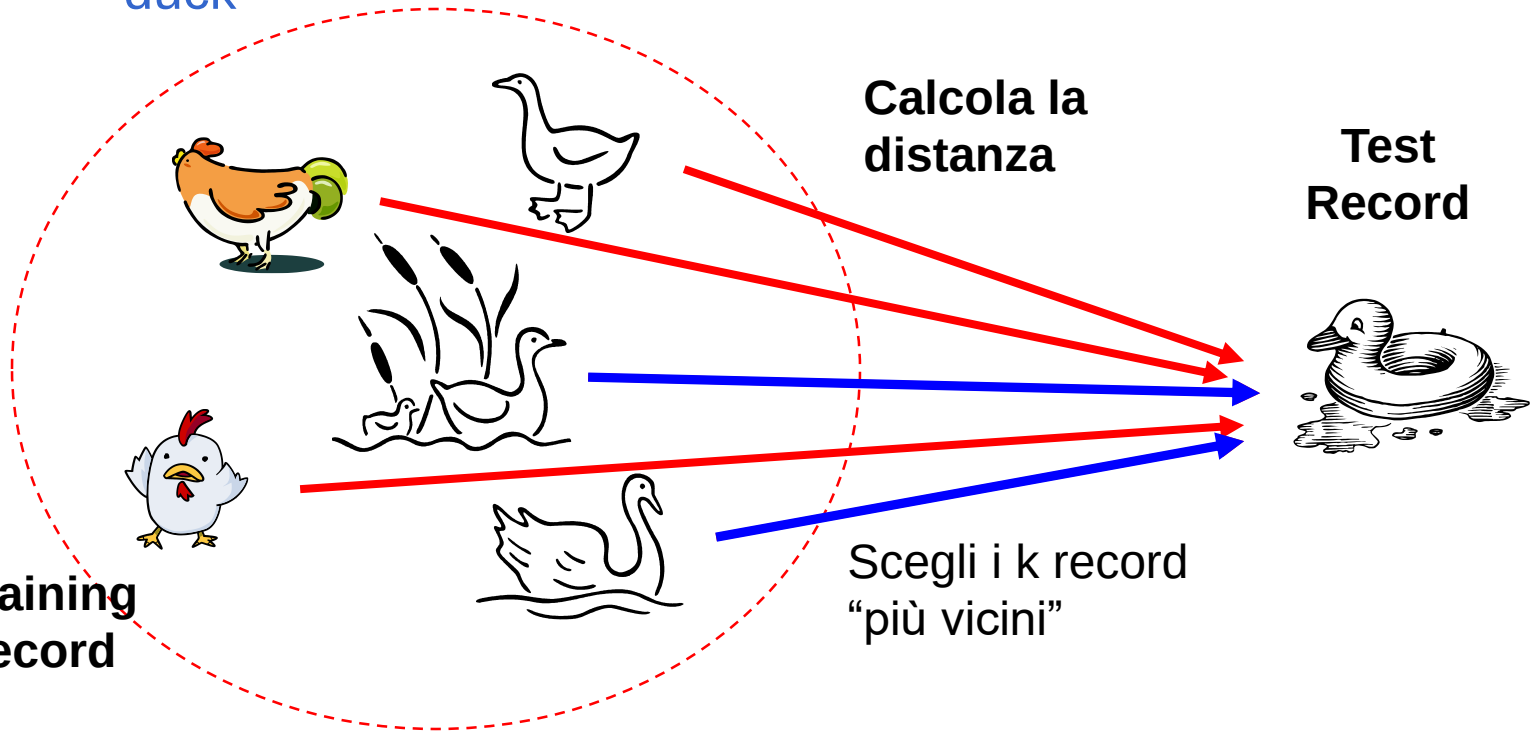
Classificatori Instance-Based

- Non costruiscono modelli ma classificano i nuovi record sulla base della loro somiglianza rispetto agli esempi nel training set
- Sono per questo detti lazy -pigri- learners in contrapposizione agli eager –diligenti, impazienti- learners (rule based, alberi decisionali, reti neurali, ecc.)
  - ✓ Rote-learner: classifica un record solo se coincide con uno del training set
  - ✓ Nearest-Neighbor: classifica il record in base ai più simili del training set

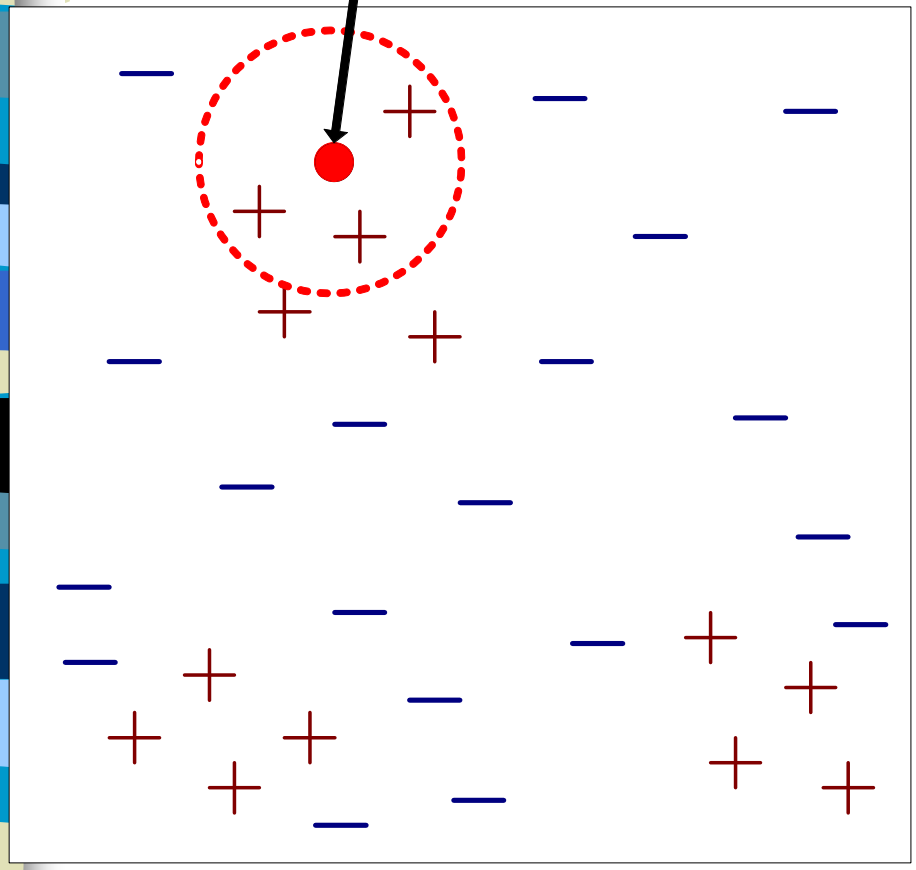


Classificatori Nearest Neighbor

- Utilizzano i k punti “più vicini” (nearest neighbors) per effettuare la classificazione
- Idea di base:
  - ✓ If it walks like a duck, quacks like a duck, then it’s probably a duck

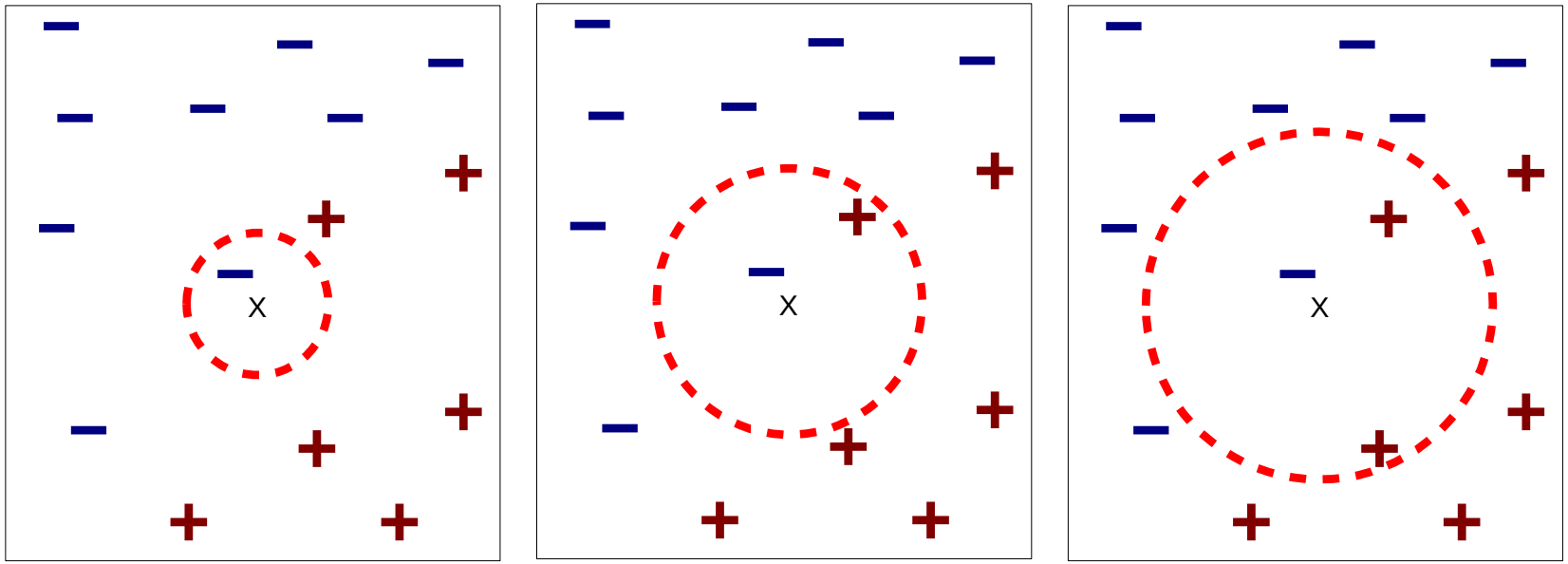


Classificatori Nearest Neighbor



- Richiedono:
  - Un training set
  - Una metrica per calcolare la distanza tra i record
  - Il valore di *k*, ossia il numero di vicini da utilizzare
- Il processo di classificazione:
  - Calcola la distanza rispetto ai record nel traing set
  - Identifica *k* nearest neighbors
  - Utilizza le label delle classi dei nearest neighbor per determinare la classe del record sconosciuto (es. scegliendo quella che compare con maggiore frequenza)

Definizione di Nearest Neighbor



(a) 1-nearest neighbor (b) 2-nearest neighbor (c) 3-nearest neighbor

I *k* nearest neighbors di un record **x** sono i record del training set che hanno le più piccole *k* distance da **x**

K-Nearest Neighbor

ML

- La classificazione di un record **z** è ottenuta con un processo di *majority voting* tra i *k* elementi D_z del training set *D* più vicini (o simili) a **z**

$$\bar{y} = \arg\max_{y \in \mathbf{Y}} \sum_{(\mathbf{x}_i, y_i) \in D_z} I(y_i = y)$$

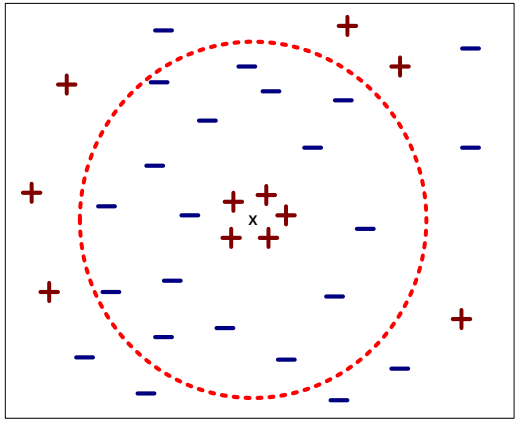
  - ✓ Y è l'insieme delle label di classe
  - ✓ I() restituisce 1 se il suo argomento è TRUE, 0 altrimenti

- Tutti i vicini hanno lo stesso peso
  - ✓ L'algoritmo è molto sensibile al valore di *k*
  - ✓ Questo rischio può essere ridotto pesando il contributo dei vicini in base alla distanza $w = 1/d(\mathbf{z},\mathbf{x}_i)$

$$\bar{y} = \arg\max_{y \in \mathbf{Y}} \sum_{(\mathbf{x}_i, y_i) \in D_z} w \times I(y_i = y)$$

K-Nearest Neighbor

- La scelta di *k* è importante perchè:
  - ✓ Se *k* è troppo piccolo, l'approccio è sensibile al rumore
  - ✓ Se *k* è troppo grande, l'intorno può includere esempi appartenenti ad altre classi



- Per operare correttamente gli attributi devono avere la stessa scala di valori e vanno quindi normalizzati in fase di pre-processing
  - ✓ Esempio: su quale attributo una differenza di 0.5 vale di più?
    - l'altezza di un adulto varia 1.5m to 2.1m
    - il peso di un adulto varia da 40kg a 150kg
    - Lo stipendio di una persona varia da 10K€ to 1M€

- Normalizzare la scala può non essere sufficiente in presenza di diverse distribuzioni dei dati
  - ✓ Mahalanobis distance

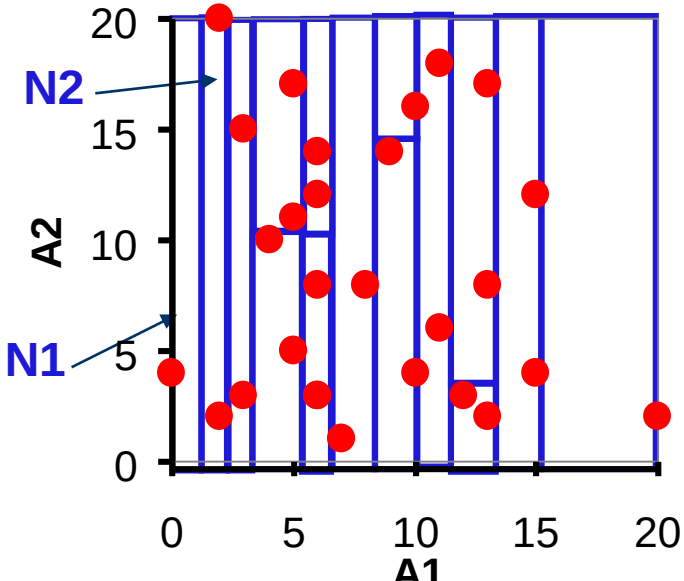
K-Nearest Neighbor: Pro & Contro

- Pro
  - ✓ Non richiedono la costruzione di un modello
  - ✓ Rispetto ai sistemi basati su regole o decision tree permettono di costruire “contorni” delle classi non lineari e sono quindi più flessibili

- Contro
  - ✓ Richiedono una misura di similarità o distanza per valutare la vicinanza
  - ✓ Richiedono una fase di pre-processing per normalizzare il range di variazione degli attributi
  - ✓ La classe è determinata localmente e quindi è suscettibile al rumore dei dati
  - ✓ Sono molto sensibili alla presenza di attributi irrilevanti o correlati che falseranno le distanze tra gli oggetti
  - ✓ Il costo di classificazione può essere elevato e dipende linearmente dalla dimensione del training set in mancanza di opportune strutture ad indice

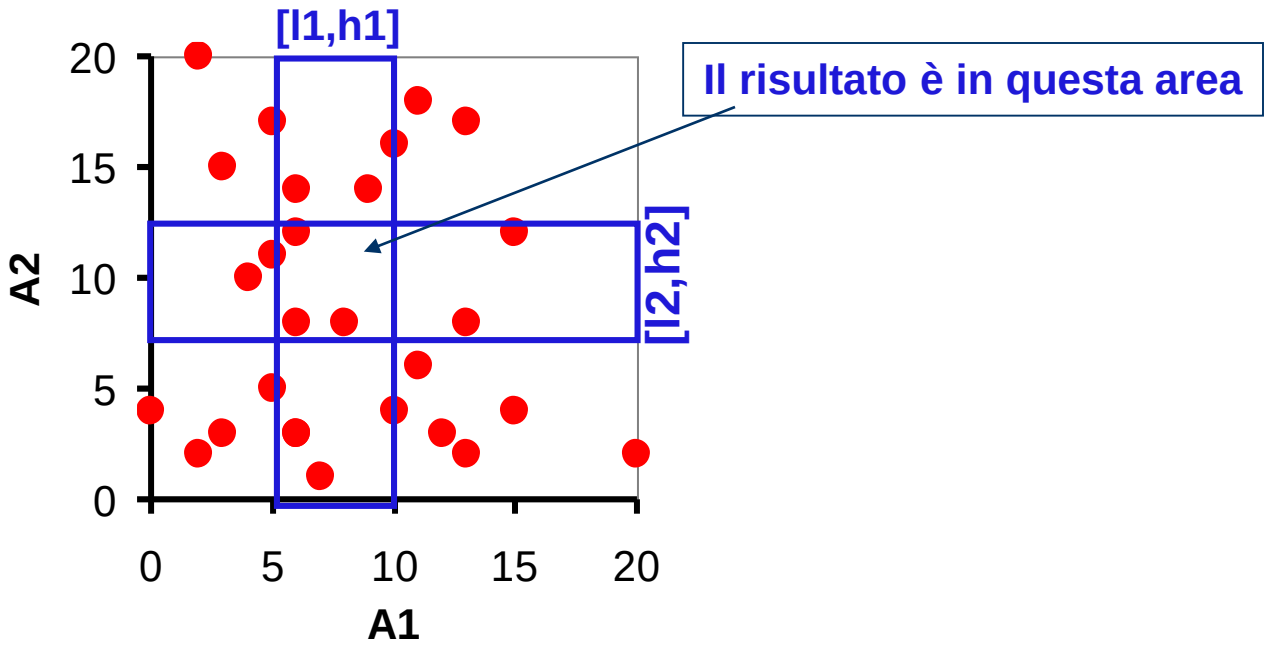
Utilizzo di indici per query spaziali

- Dati n attributi A1,…,An e in assenza di strutture dati specifiche si potrebbe pensare di costruire un B^+-tree multi-attributo, che organizzi (ordini) le tuple in base ai valori di A1,A2,…,An
- Considerando i nodi foglia del B^+-tree: N1→N2→N3 →…
  - ✓ La prima foglia, N1, conterrà le tuple con i più piccoli valori di A1 in base alla capacità della foglia stessa
  - ✓ La seconda foglia inizierà con i valori seguenti, e così via
- Qualsiasi ordinamento si scelga, la ricerca di un k-NN di un punto richiederà l'accesso a molti nodi poichè questa soluzione non cattura il concetto di distanza spaziale



Un'altra soluzione basata sui B$^+$-tree

- Assumiamo di conoscere, per esempio mediante le statisiche del DB, che i k-NN di un punto q stiano nell'(iper) rettangolo con lati [*l1*,*h1*] x [*l2*,*h2*] x…
- Allora possiamo utilizzare *n* query di range indipendenti **`Ai BETWEEN li AND hi`** su *n* indici distinti *A1*,*A2*,…,*An* e intersecare il risultato
  - ✓ Oltre a richiedere di conoscere i valori dei range questa tecnica richiederebbe comunque molto lavoro poichè leggerà una quantità di tuple proporzionale all'unione degli *n* risultati a meno della loro intersezione



Indici multi-dimensionali (spaziali)

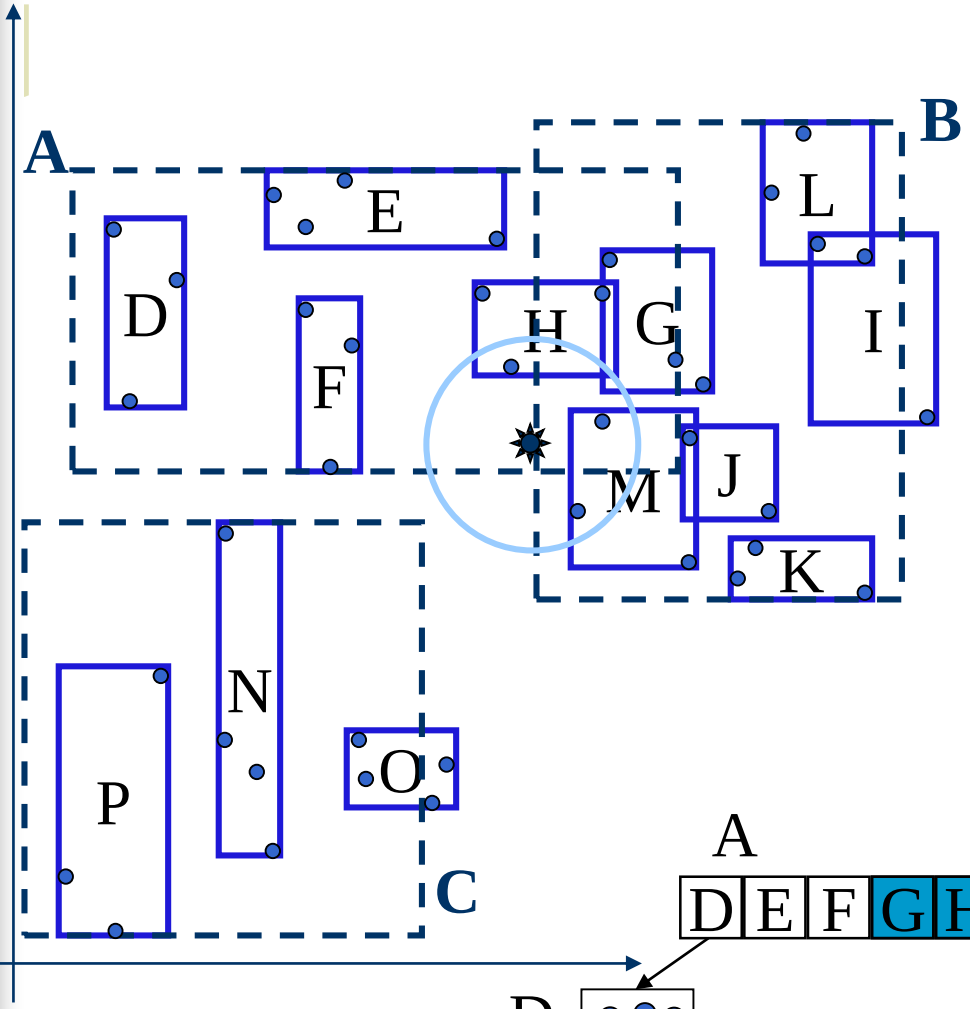
- Il B$^+$-tree multi-attributo mappa punti di $\mathbf{A} \subseteq \Re^D$ in punti di $\Re$
- Questa "linearizzazione" forzatamente favorisce uno degli attributi in base al modo in cui questi sono organizzati nel B$^+$-tree
  - ✓ Un B$^+$-tree su (X,Y) favorisce query su X, e non può essere utilizzato per query che non specifichino il valore di X
- Per questo motivo, dobbiamo utilizzare un modo che al contrario preservi la "prossimità spaziale"
- Il tema dello "spatial indexing" è stato studiato a partire dagli anni '70 a causa della sua importanza (cartografia, GIS, VLSI, CAD)
- Più recentemente (anni '90), è tornato di "moda" grazie a nuove tipologie di applicazioni come quelle multimediali

Gli R-tree (Guttman, 1984)

- Gli R-tree sono estensioni dei B^+-tree a spazi multi-dimensionali
- I B^+-tree organizzano gli oggetti in
  - ✓ Un insieme di intervalli mono-dimensionali non sovrapposti
  - ✓ Applicano questo prinicipio ricorsivamente dalle foglie alla radice
- Gli R-tree organizzano gli oggetti in
  - ✓ Un insieme di intervalli multi-dimensionali sovrapposti (iper-rettangoli)
  - ✓ Applicano questo principio ricorsivamente dalle foglie alla radice

- Gli R-tree sono oggi disponibili in alcuni DBMS commerciali quali Oracle9i

R-tree: intuizione



- Aggregazione ricorsiva di oggetti basata sul MBR
- Le regioni possono sovrapporsi

- Nella slide è mostrata una 2-D *range query* che utilizza L2