Metodi Matematici per l'Infomatica

(e per la Medicina)

Simone Parisotto

Università di Verona
Dipartimento di Scienze Matematiche Fisiche e Naturali
-
Laurea Magistrale in Matematica

18 Marzo 2013

Parte I: Immagini in Medicina

Upsampling di una immagine

Definizione del problema

Data una immagine `A` di dimensione $N \times N$ si vuole ottenere l'immagine migliorata `B` di dimensione $M \cdot N \times M \cdot N$, con $M > N$ (tipicamente $M = 2, 3, \ldots$).

Risoluzione

Indica il grado di qualità di un'immagine. Il numero dei pixel è un indicatore del più piccolo dettaglio dell'immagine e determina la nitidezza dell'immagine.

La funzione `imresize` di Matlab implementa vari metodi per fare upsampling o downsampling di una immagine: questi metodi sono basati su interpolazione o convoluzione con kernels specifici.

Esempio

Immagine presa dal Dataset online `http://morguefile.com`.

(A) 256 × 256

(B) 512 × 512

Il workflow seguito usa tutta l'informazione dell'immagine grayscale (A) per ottenere (B). Il risultato viene poi confrontato con l'immagine di riferimento nel dataset.

Stato dell'Arte

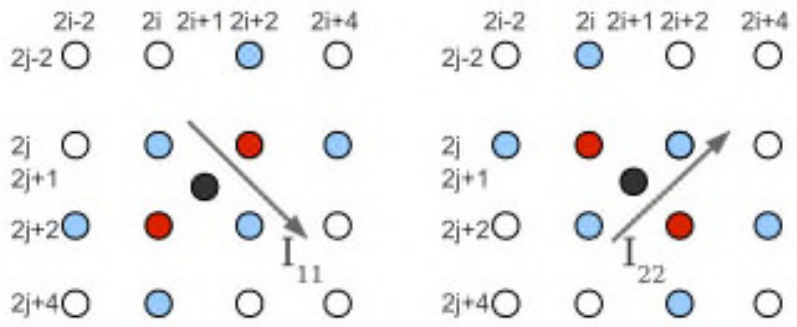
La sintassi di Matlab per ottenere l'immagine `B` a partire da `A` tramite un resize di fattore `factor` e metodo `method` è

```
B = imresize(A,factor,method);
```

dove a `method` si può sostituire:

- `'nearest'`: al pixel di output è assegnato il valore del pixel del punto vicino;
- `'bilinear'`: al pixel di output è assegnata la media pesata dei 2x2 pixel vicini;
- `'bicubic'`: al pixel di output è assegnata la media pesata dei 4x4 pixel vicini;
- `'lanczos2'`: fa uso di una funzione `sinc` finestrata come nucleo di convoluzione per ricampionare un'immagine;

- `ICBI`: Iterative Curvature Based Interpolation (A. Giachetti and N. Asuni), si basa sulla combinazione di due metodi:
  - un algoritmo adattativo è applicato per interpolare i valori dei pixel localmente lungo la direzione dove la derivata seconda è più bassa;



  - i valori interpolati vengono poi modificati usando un raffinamento iterativo allo scopo di minimizzare l'energia globale.

Descrittori statistici per confronto immagini

- `RMSE`: Root Mean Square Error, misura la varianza interna;
- `PSNR`: Peak Signal-to-Noise Ratio, misura la qualità della ricostruzione dell'immagine compressa (lossy)

$$\mathrm{PSNR}(A, B) = 20 \log_{10} \frac{255}{\mathrm{RMSE}(A, B)};$$

- `MSSIM`: Mean Structural Similarity Index Method, misura la similarità tra immagini

$$\mathrm{SSIM}(A, B) = \frac{(2\mu_A\mu_B + c_1)(2\sigma_{AB} + c_2)}{(\mu_A^2 + \mu_B^2 + c_1)(\sigma_A^2 + \sigma_B^2 + c_2)}$$

$$\mathrm{MSSIM}(A, B) = \frac{1}{M} \sum_{j=1}^{M} \mathrm{SSIM}(A_j, B_j);$$

- `PEE`: Percentage Edge Error, misura quanto i dettagli dell'immagine interpolata sono vicini all'originale.

$$\mathrm{PEE}(A, B) = \frac{\mathrm{ES}(A) - \mathrm{ES}(B)}{\mathrm{ES}(A)} \times 100.$$

Interpolazione con Radial Basis Function

$\phi : \mathbb{R}^s \to \mathbb{R}$ è detta radiale se esiste una funzione univariata $\varphi : [0, +\infty) \to \mathbb{R}$ t.c.

$$\phi(\mathbf{x}) = \varphi(r), \text{ dove } r = ||\mathbf{x}||, \ \mathbf{x} \in \mathbb{R}^s$$

con $||\cdot||$ una certa norma su $\mathbb{R}^s$.

Il problema dell'interpolazione

Si vuole trovare l'interpolante $\mathcal{P}_f(\mathbf{x})$ di una funzione $f(\mathbf{x})$, sui punti `eval`, i cui valori di f sono noti sui punti detti `dsites` (=`ctrs` nel nostro caso):

$$\mathcal{P}_f(\mathbf{x}) = \sum_{k=1}^{N} c_k \varphi(||\mathbf{x} - \mathbf{x}_k||), \ \mathbf{x} \in \mathbb{R}^s.$$

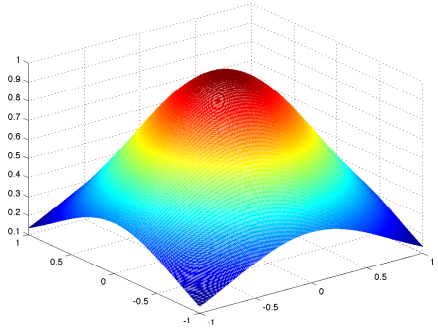
Per trovare i coefficienti c_k si risolve un unico sistema lineare, da cui l'interpolante:

```
P = EM*(IM\rhs);
```

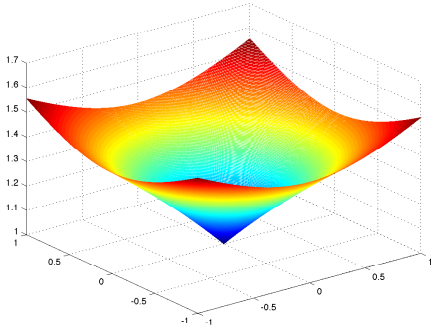
Tipi di Funzioni di Base

Oltre alla norma euclidea, che in $\mathbb{R}^2$ è $\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2)}$, possiamo definire altre norme (ma la definizione di funzioni radiali non cambia).
Alcune funzioni radiali (con norma euclidea):

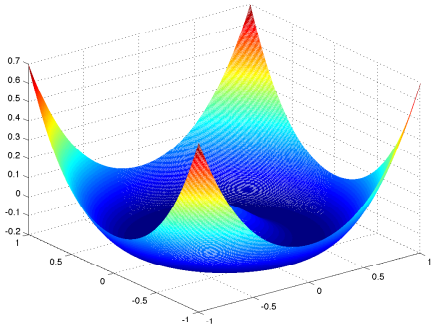
- a supporto globale:



Gaussiane
$\exp(-r^2)$

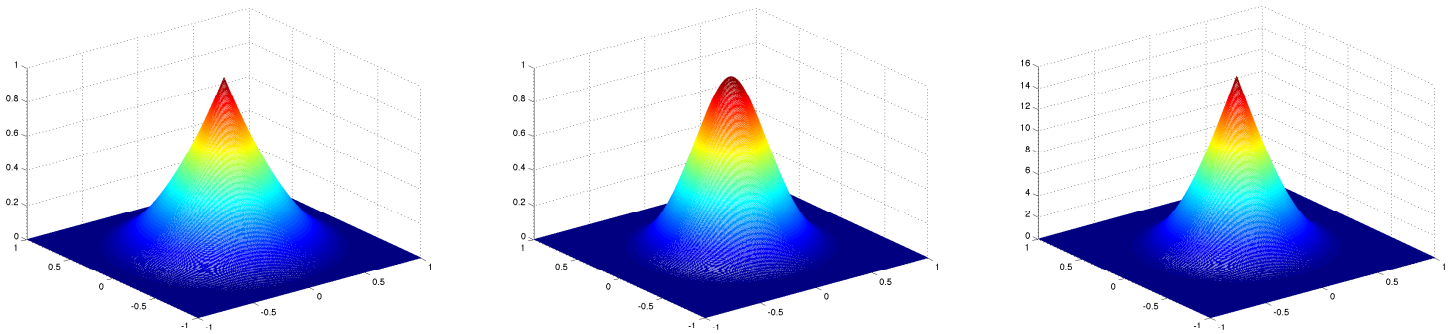


Multiquadriche Inverse
$1/(1 + r^2)$



TPS
$r^2 \log(r)$

- a supporto compatto (la matrice delle distanze diventa sparsa):



a) Wendland C^0 b) Wendland C^2 c) Wu C^0

a) $\varphi_{3,0}(r) = (1-r)_+^2$,

b) $\varphi_{3,2}(r) = (1-r)_+^2(35r^2 + 18r + 3)$,

c) $\psi_{3,3}(r) = (1-r)_+^4(16 + 29r + 20r^2 + 5r^3)$.

Le funzioni sopra citate verranno usate nella versione `shifted form`:

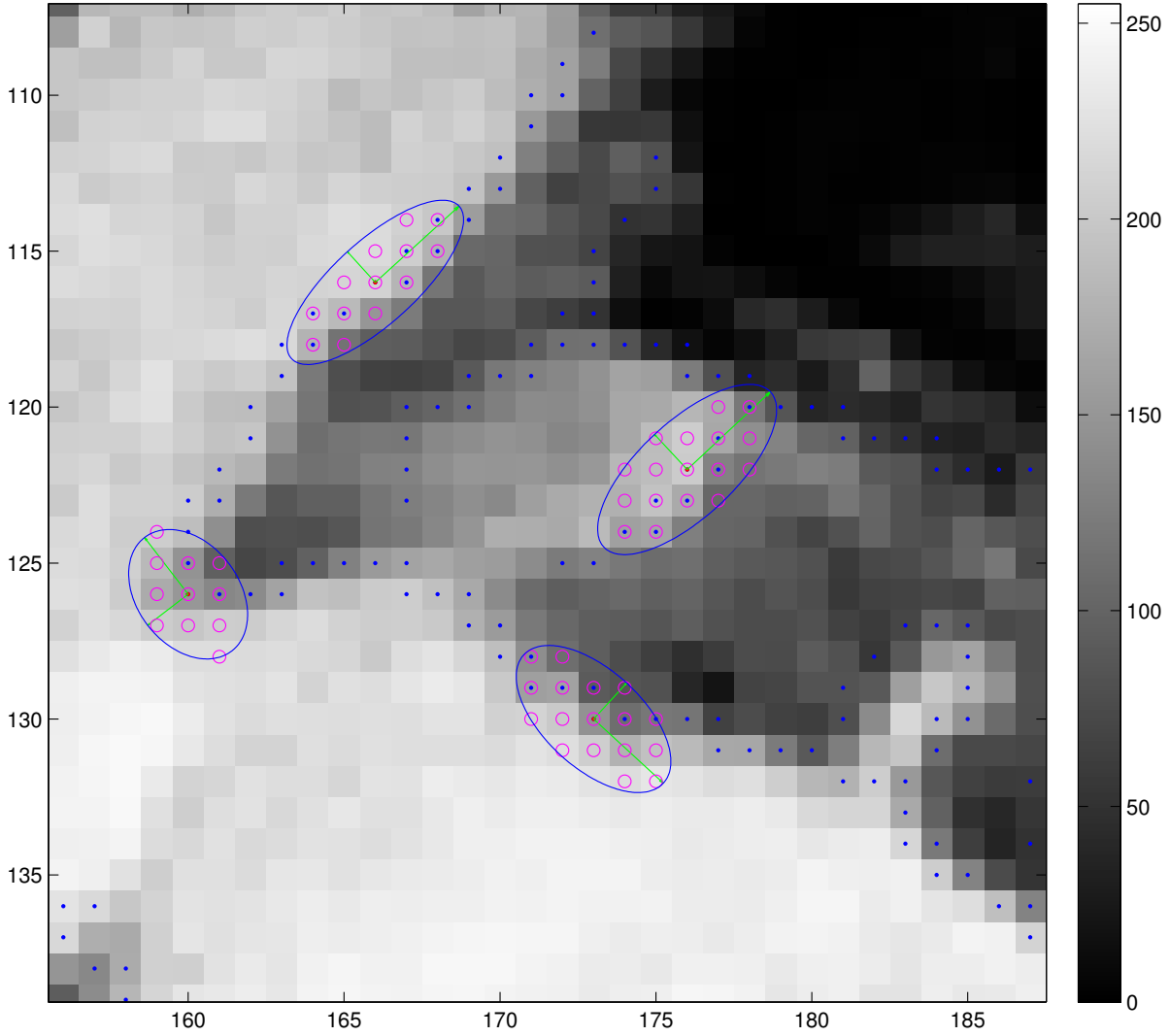
$$r_{\text{new}} = 1 - r;$$

in modo da preservare la struttura sparsa della matrice:

$$\psi_{3,3}(r) = r_+^4(70 - 84r + 35r^2 - 5r^3).$$

L'idea

Usare norme che producano supporti anisotropi (norma di Mahalanobis):



Rangequery e kd-Tree

Scopo: trovare i punti appartenenti ad una regione nello spazio (problema tipico). Con l'approccio naive potrebbe essere necessario testare tutti i punti. L'algoritmo `kd-Tree` suddivide la regione in insiemi più piccoli per facilitare la ricerca.

	naive	`kd-Tree`
costruzione griglia	$\mathcal{O}(dN)$ tempo, spazio	$\mathcal{O}(dN \log N)$ tempo, $\mathcal{O}(dN)$ spazio
range query	$\mathcal{O}(N)$ tempo	$\mathcal{O}(\log N)$ tempo[1]

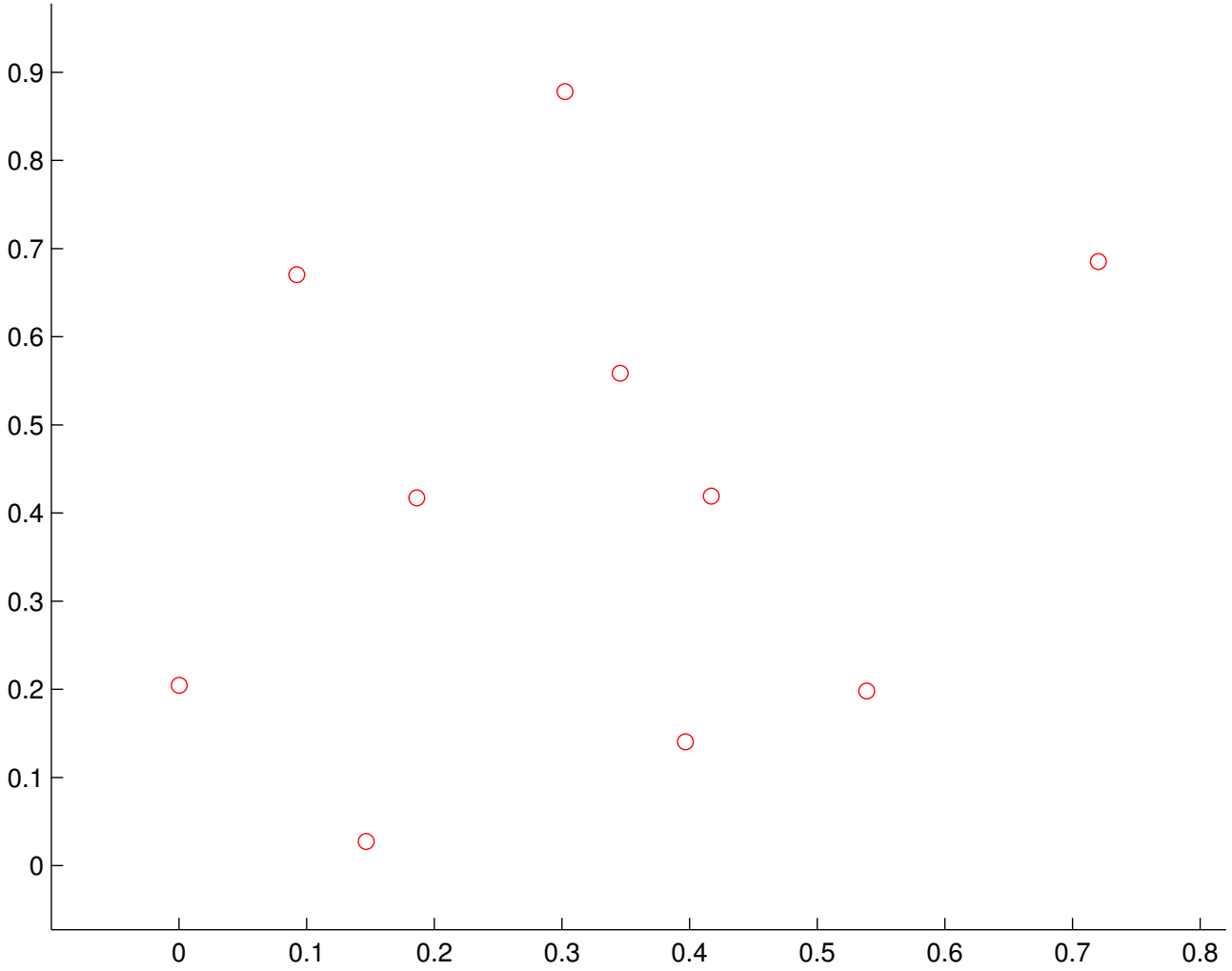
Dove usiamo il `kd-Tree`:

- non usato: costruzione Distance Matrix per punti edge perché `Matlab` non definisce `kd-Tree` per alcune metriche (tra cui Mahalanobis);
- parzialmente usato: costruzione matrice di covarianza locale (informazioni sul gradiente locale inserendo e togliendo punti di volta in volta);
- usato: costruzione Distance Matrix per punti non edge.

[1] se i punti sono quasi uniformi; $\mathcal{O}(N^{1-1/d} + k)$ altrimenti

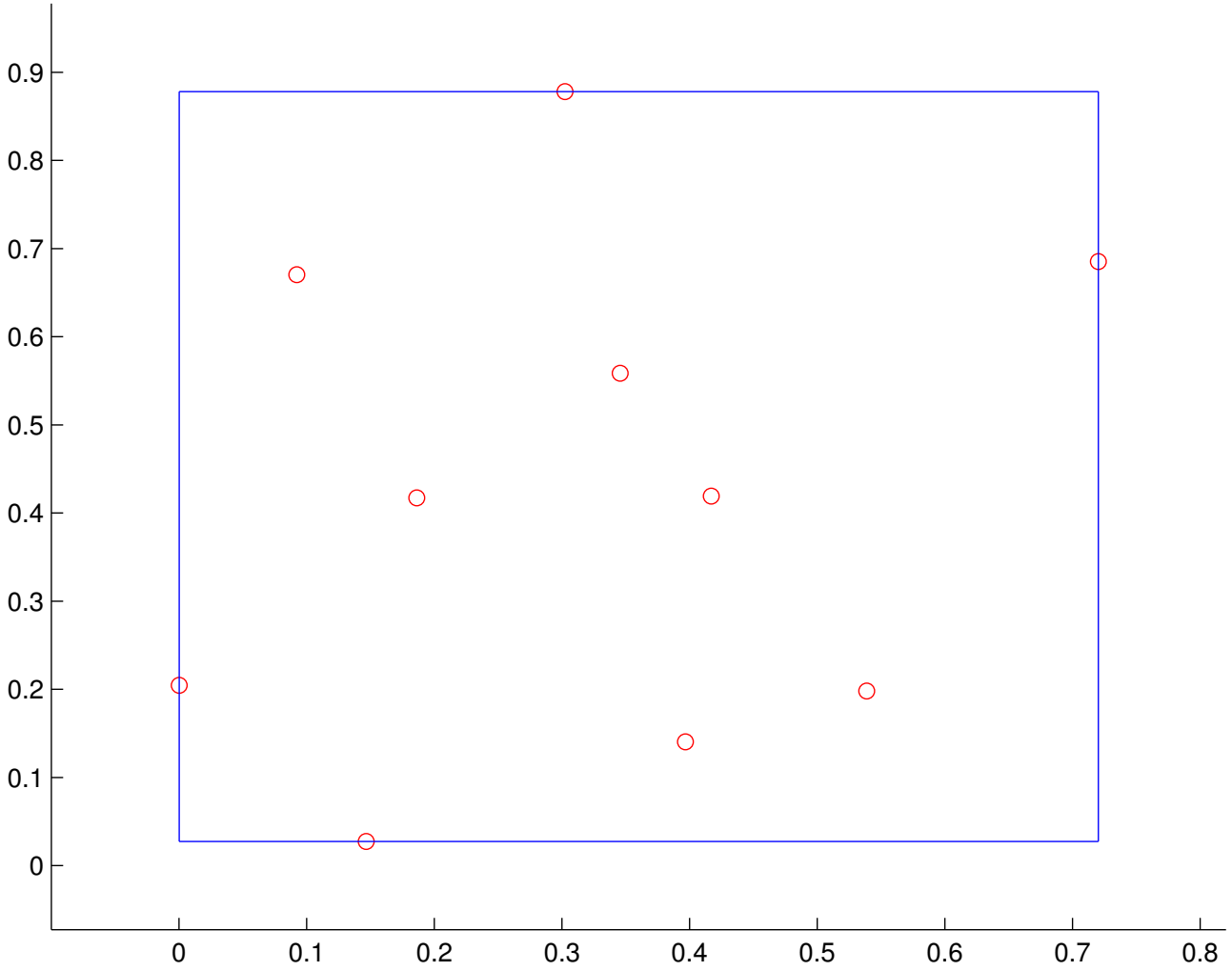
kd-Tree: Un esempio

Si basa su una divisione ricorsiva dello spazio in regioni rettangolari chiamate box.



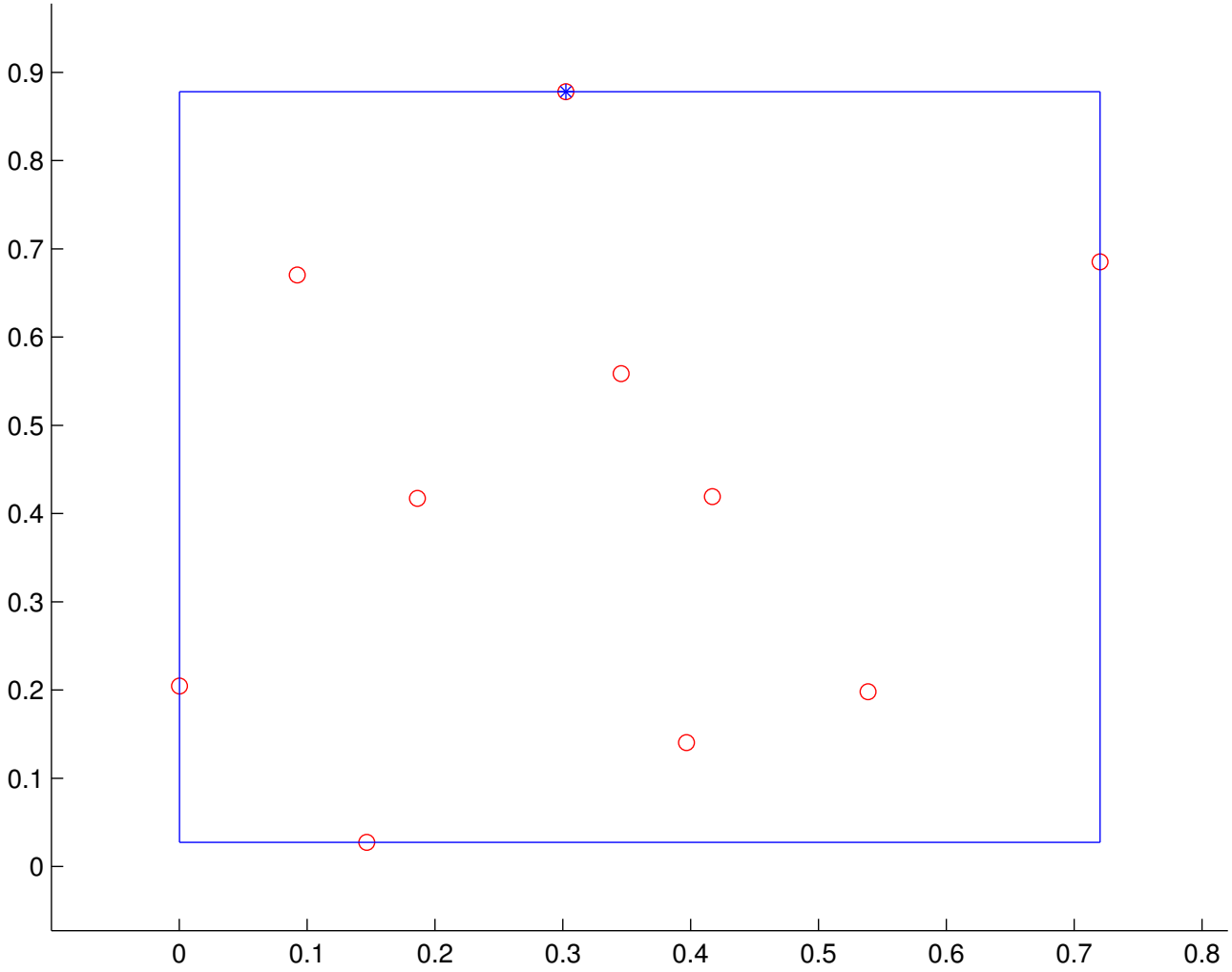
kd-Tree: Un esempio

Si prende la bounding box dei punti.



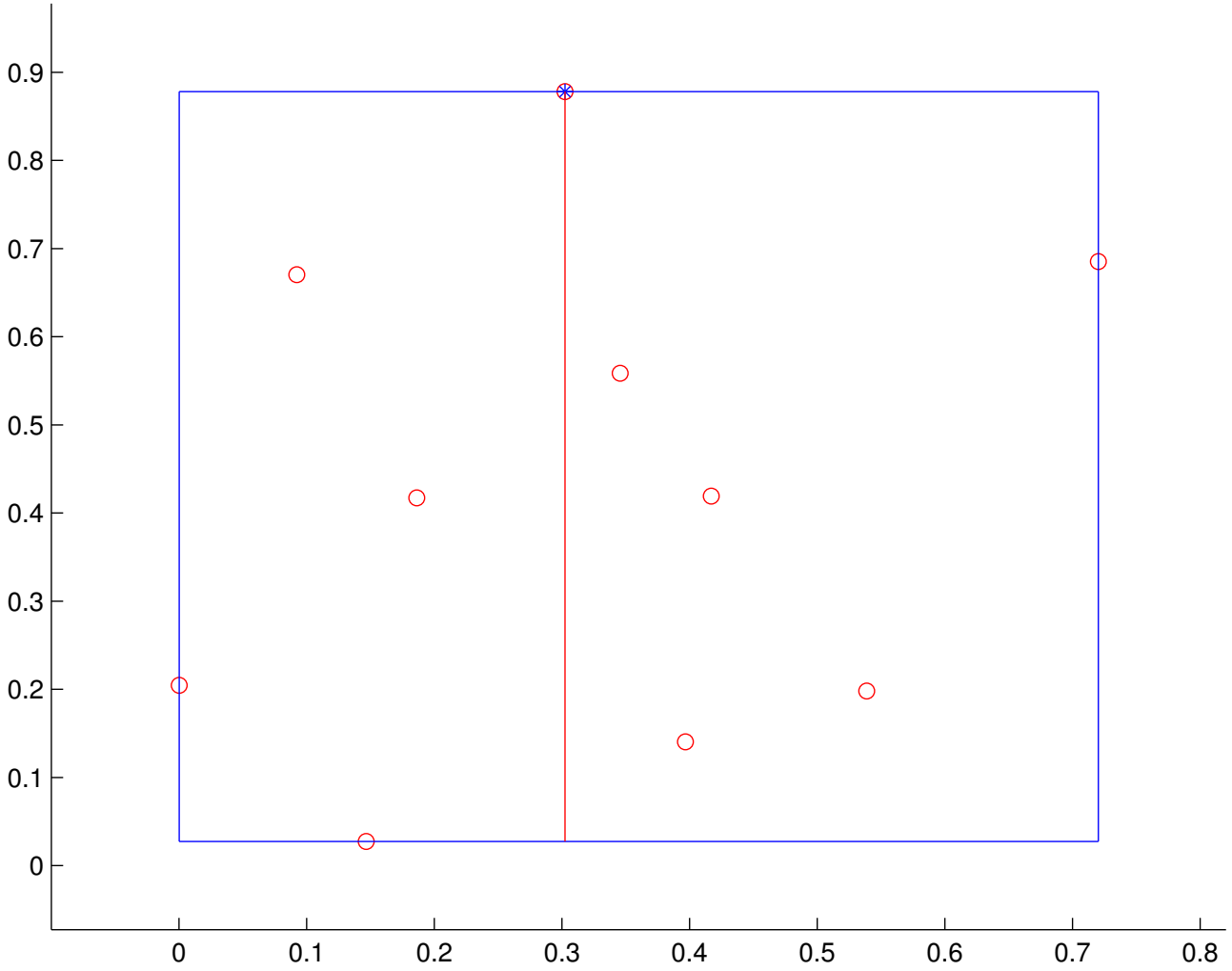
kd-Tree: Un esempio

Si sceglie un punto come radice a cui è associato il bounding box dei punti.



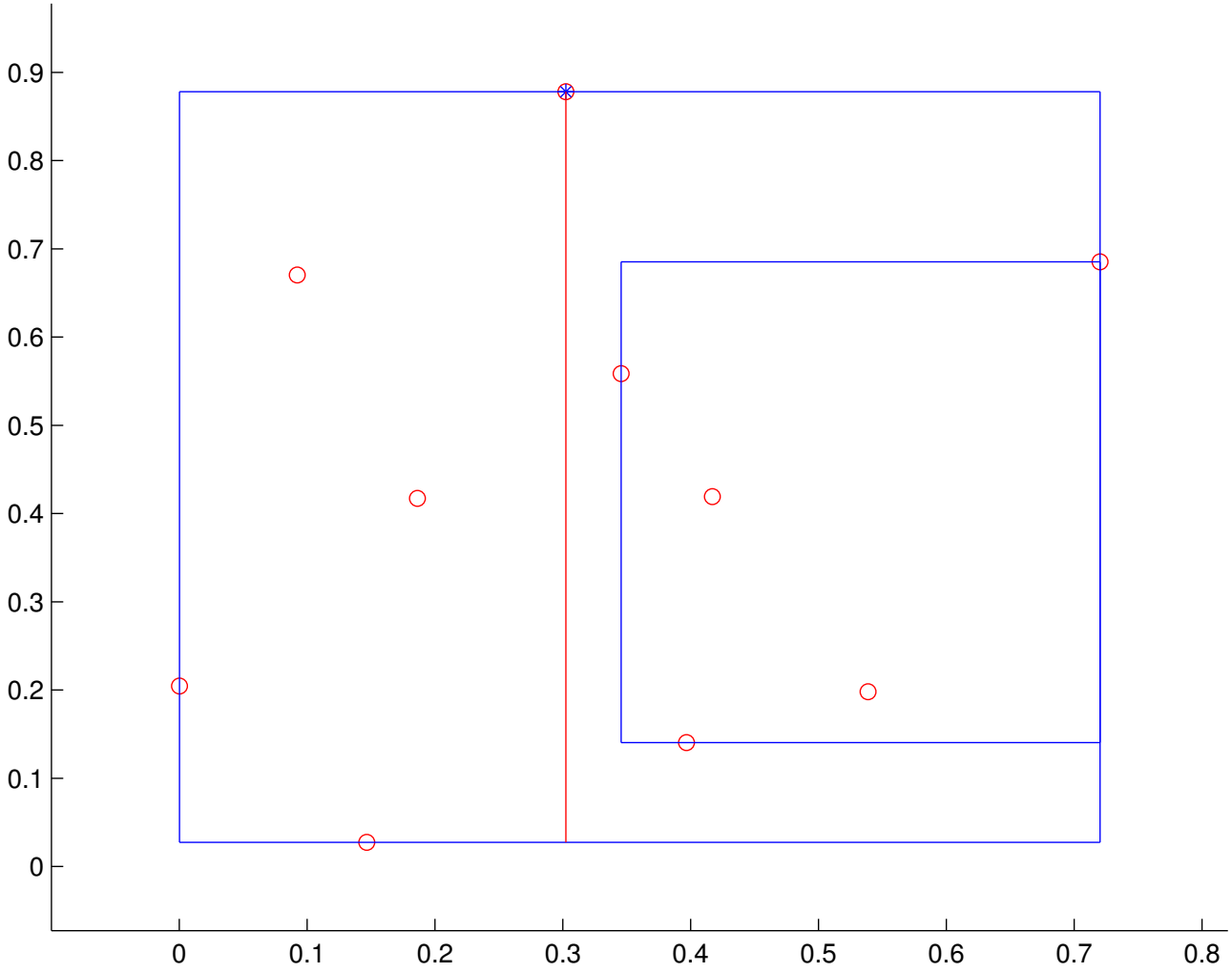
kd-Tree: Un esempio

Se i punti nella box superano la bucket size, la box è divisa in due: splitting rule.



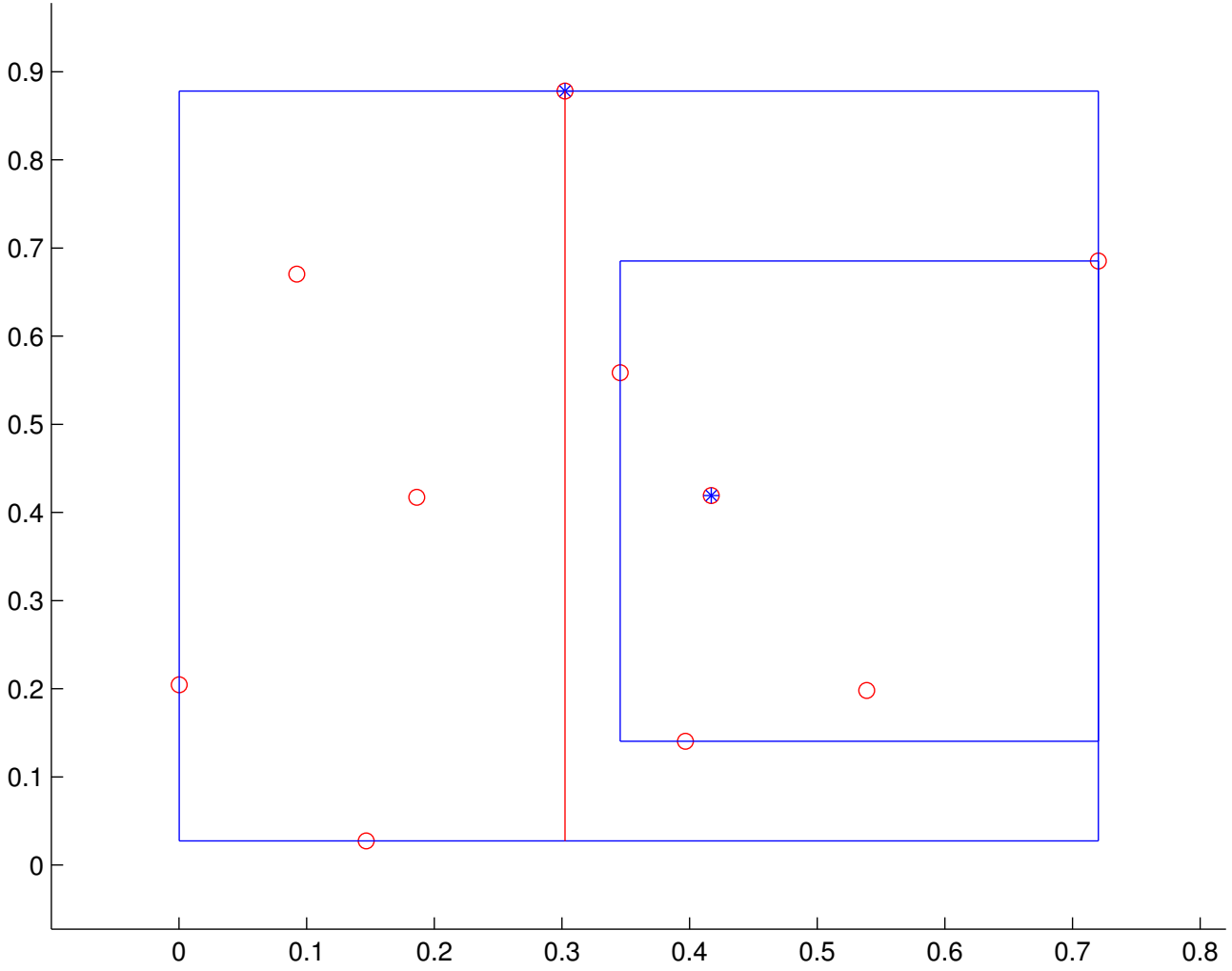
kd-Tree: Un esempio

I punti sono quindi divisi tra i due figli del nodo in base a quale parte si trovano.



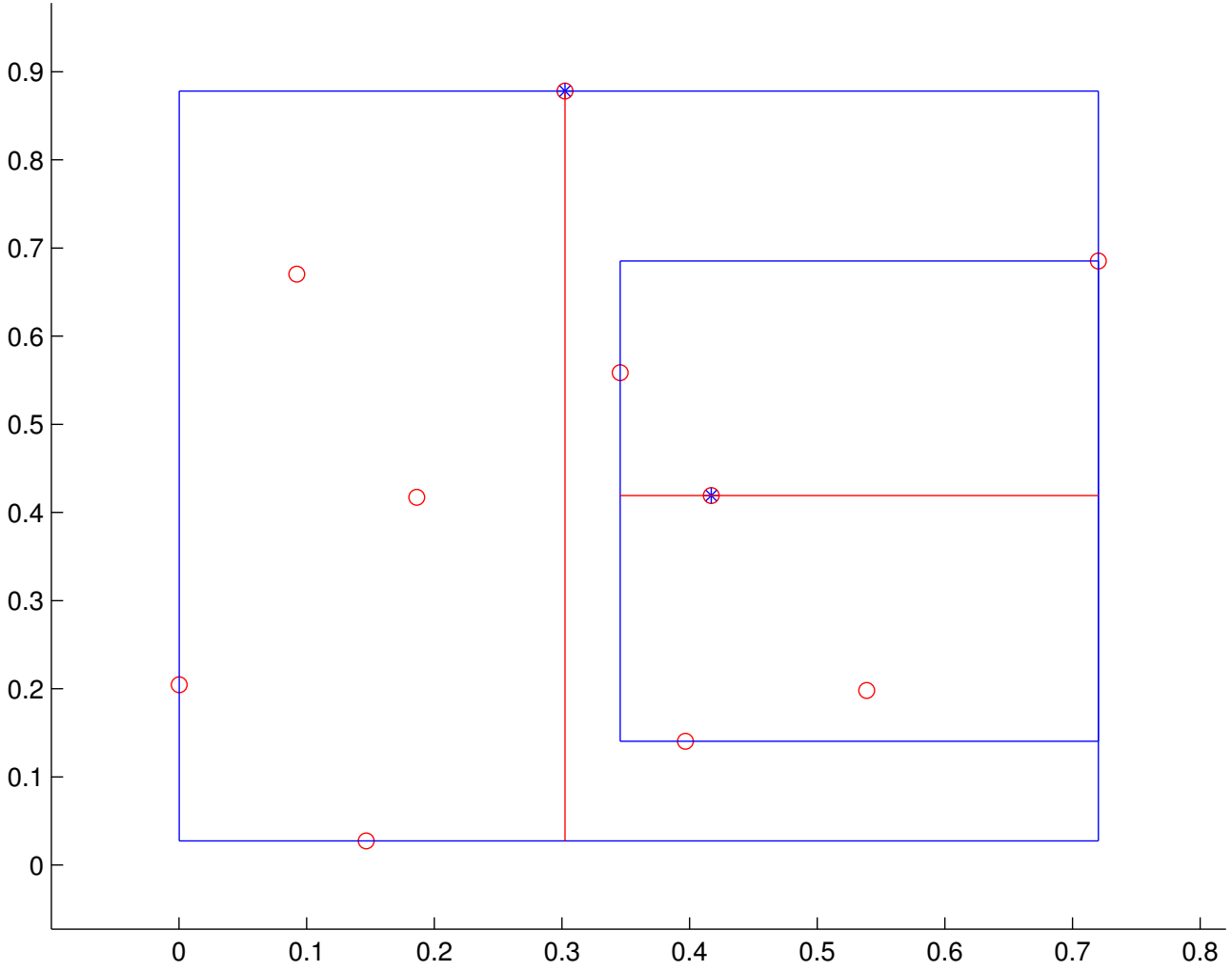
kd-Tree: Un esempio

Prendo un punto.



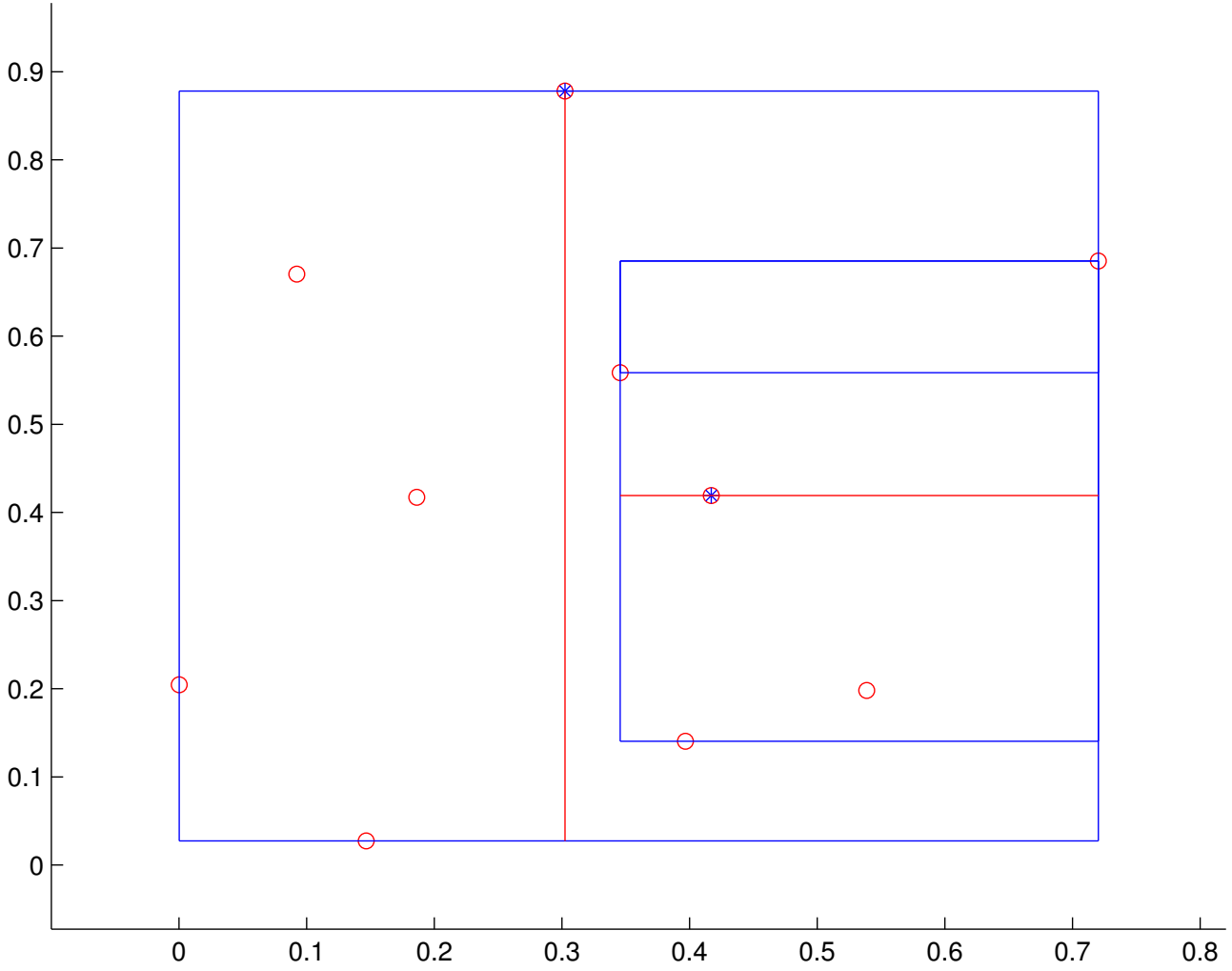
kd-Tree: Un esempio

Trovo l'iperpiano.



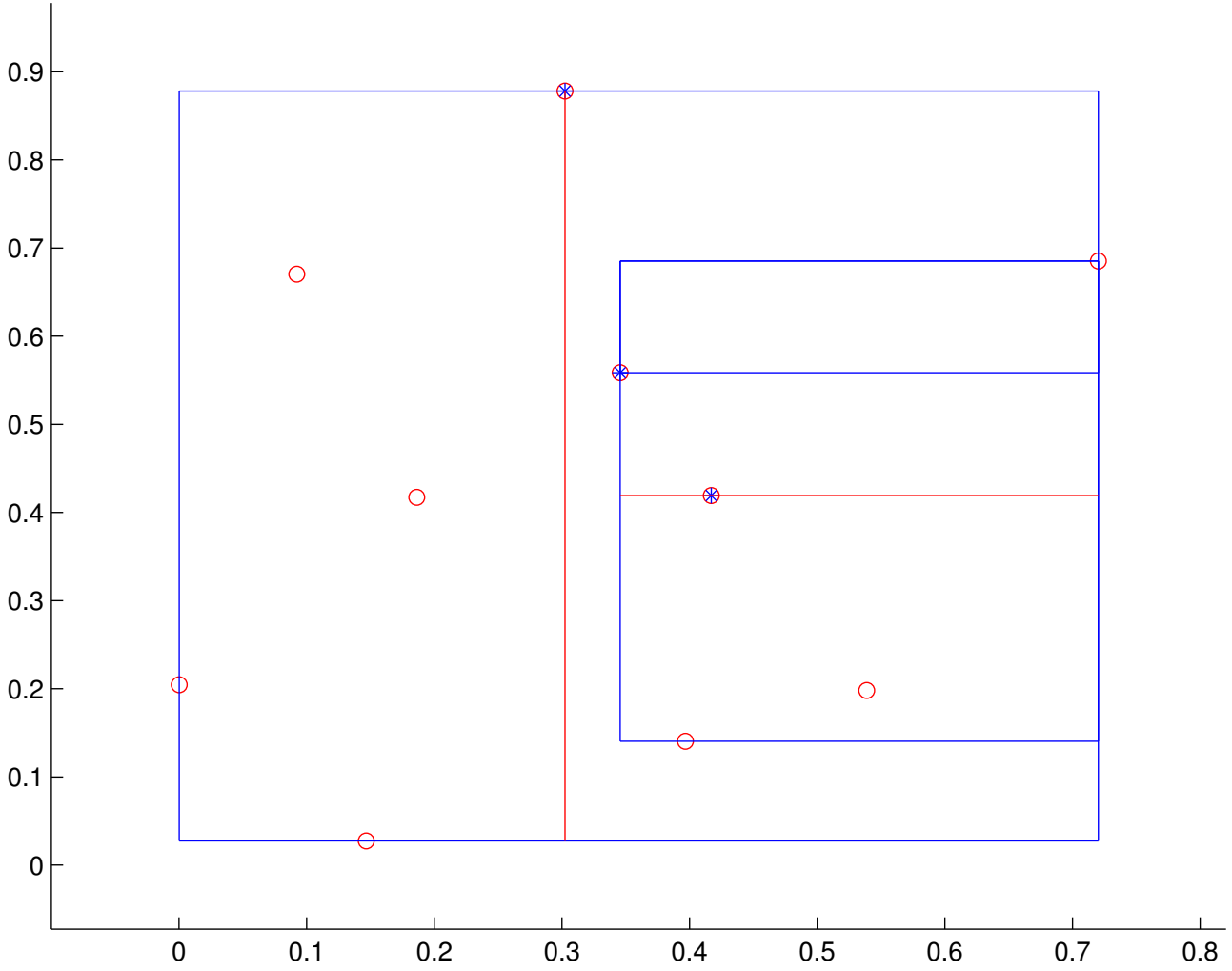
kd-Tree: Un esempio

Itero il procedimento.



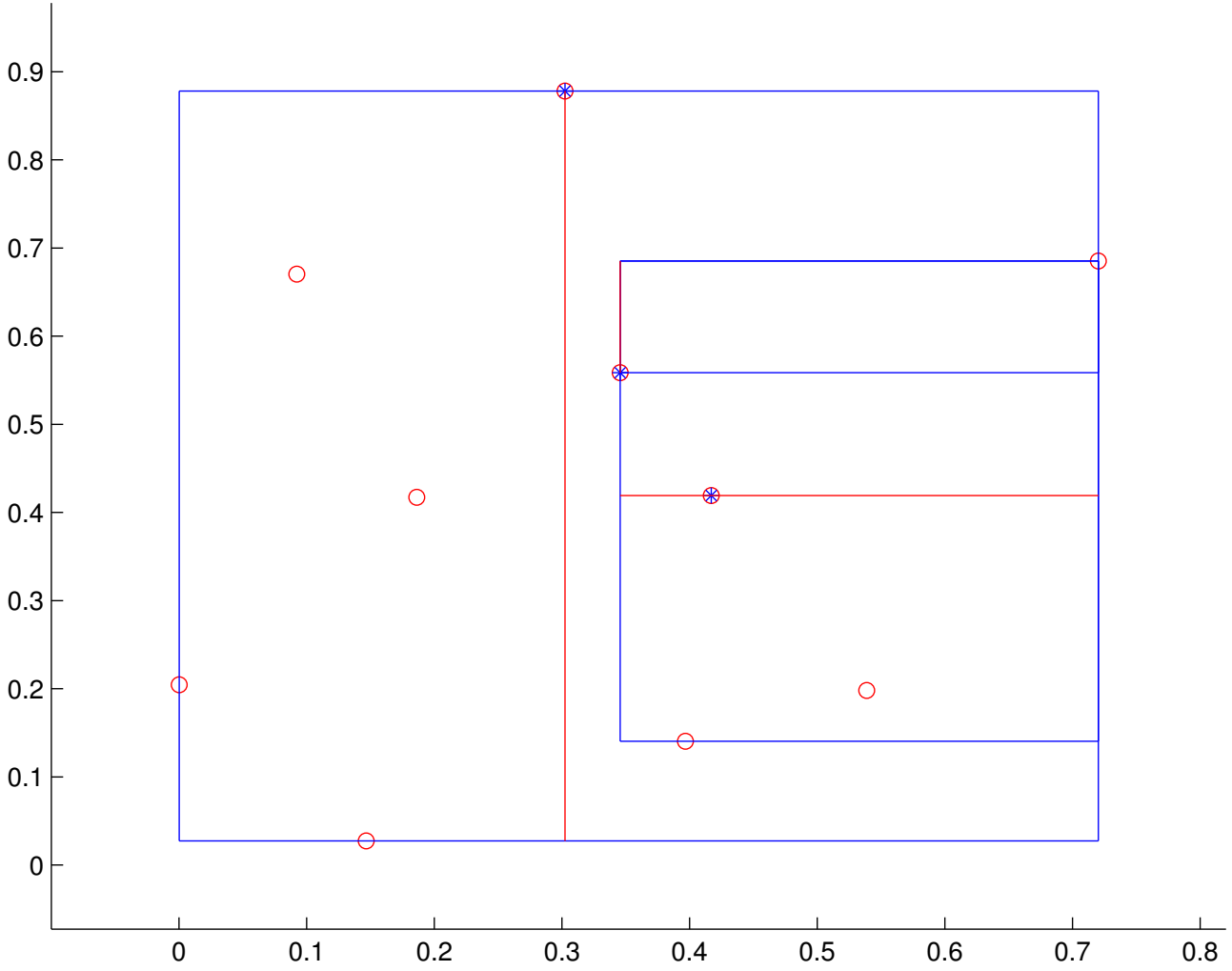
kd-Tree: Un esempio

Itero il procedimento.



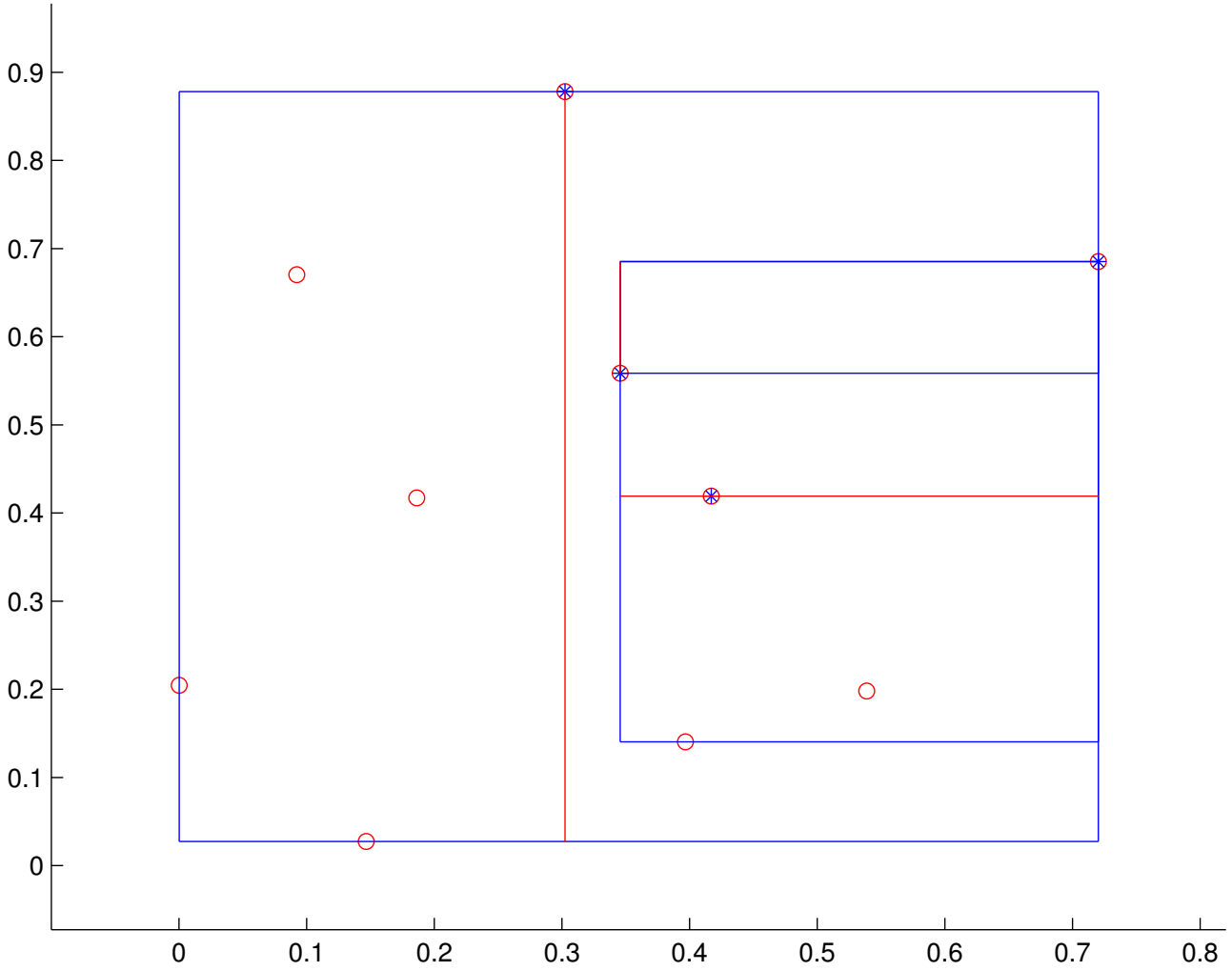
kd-Tree: Un esempio

Itero il procedimento.



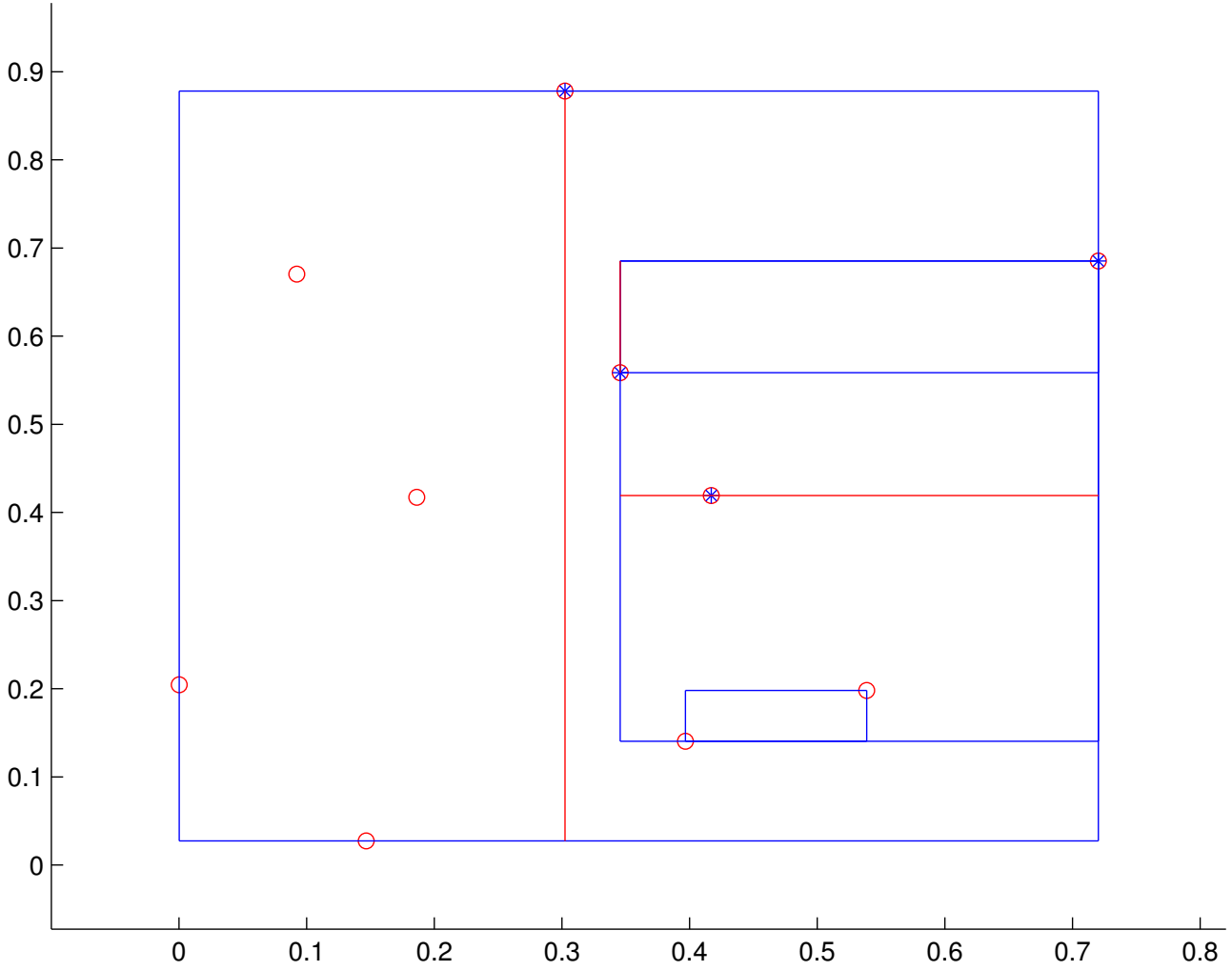
kd-Tree: Un esempio

Foglia: il numero di punti associati ad un nodo è inferiore alla bucket size.



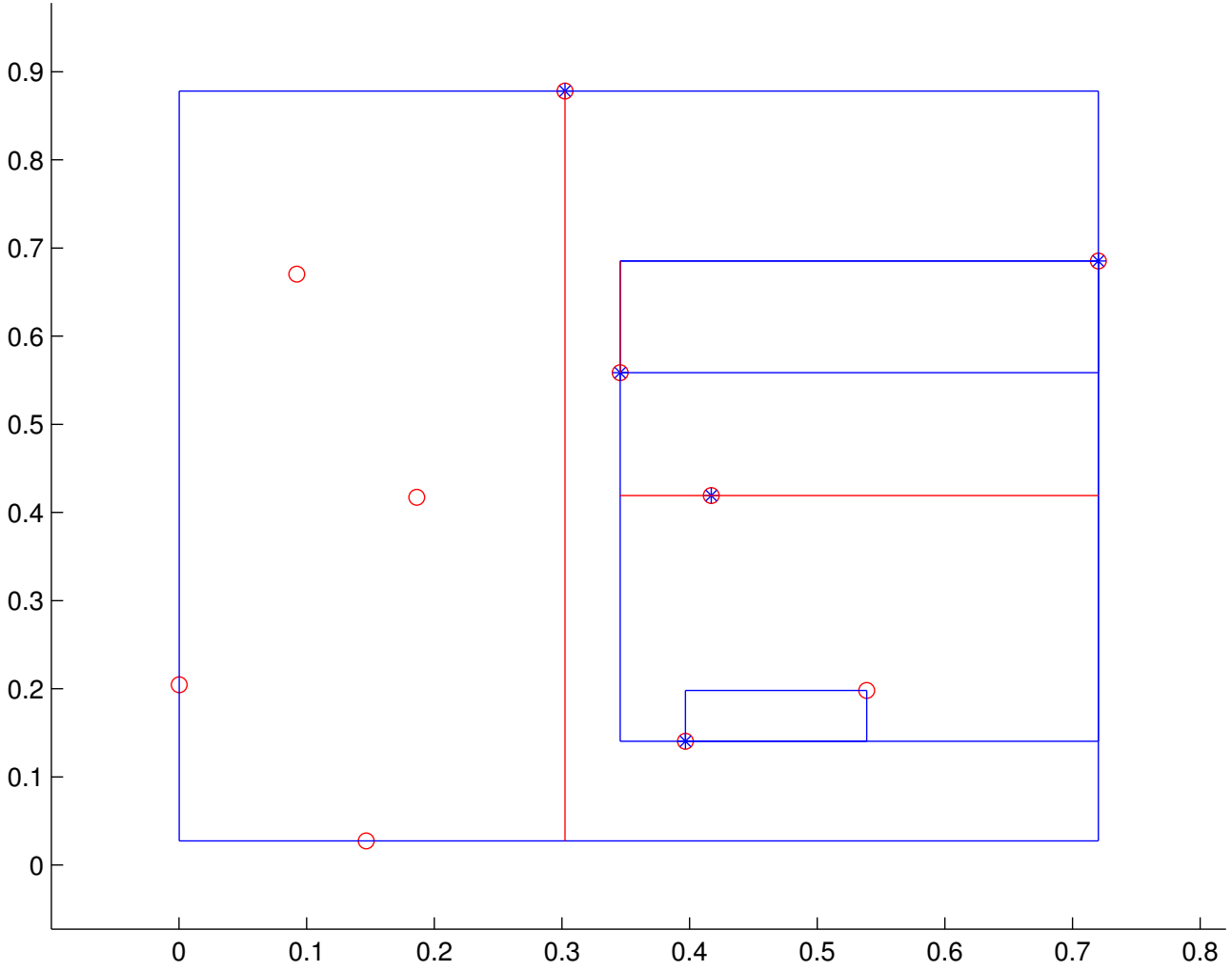
kd-Tree: Un esempio

Itero il procedimento.



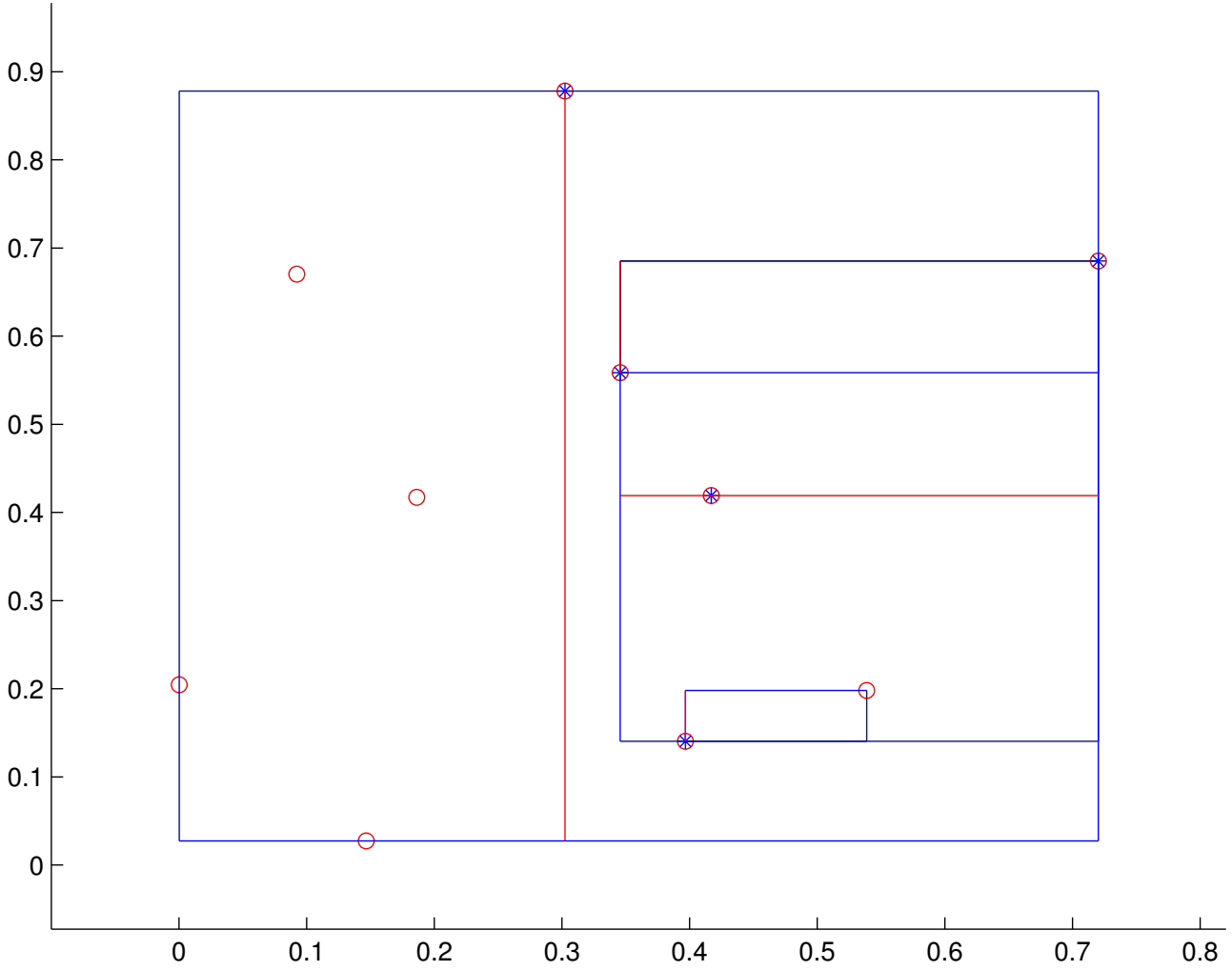
kd-Tree: Un esempio

Itero il procedimento.



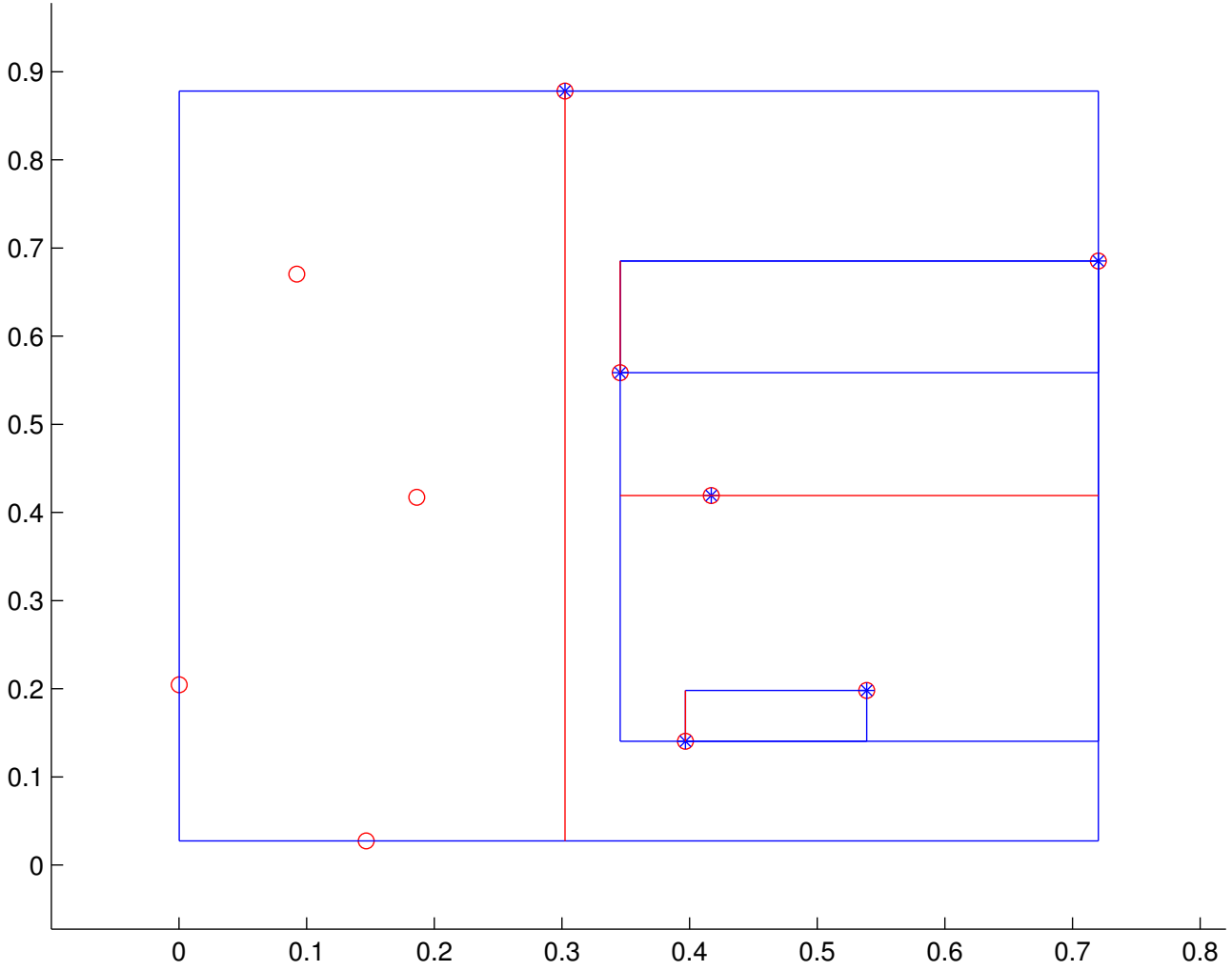
kd-Tree: Un esempio

Itero il procedimento.



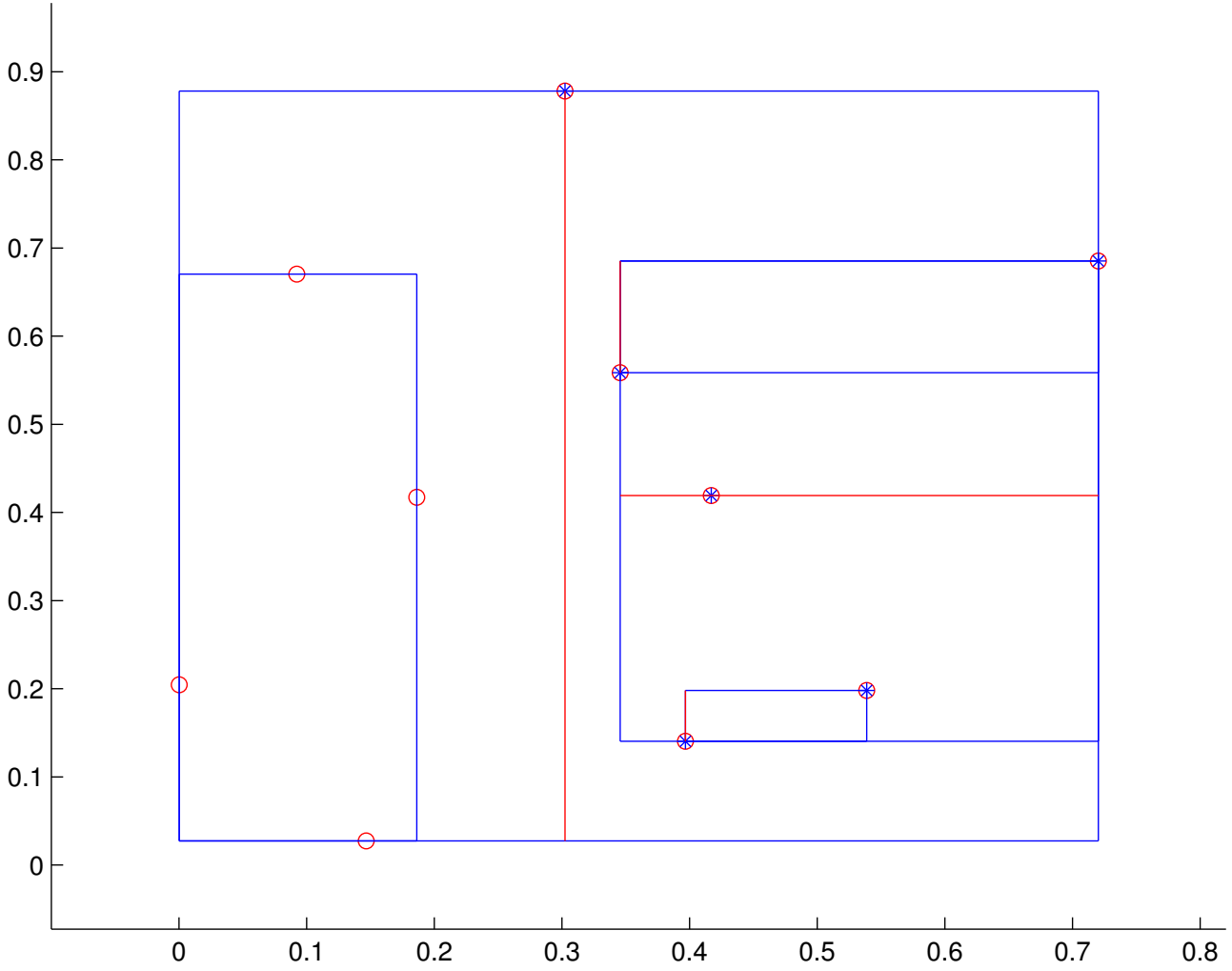
kd-Tree: Un esempio

Itero il procedimento.



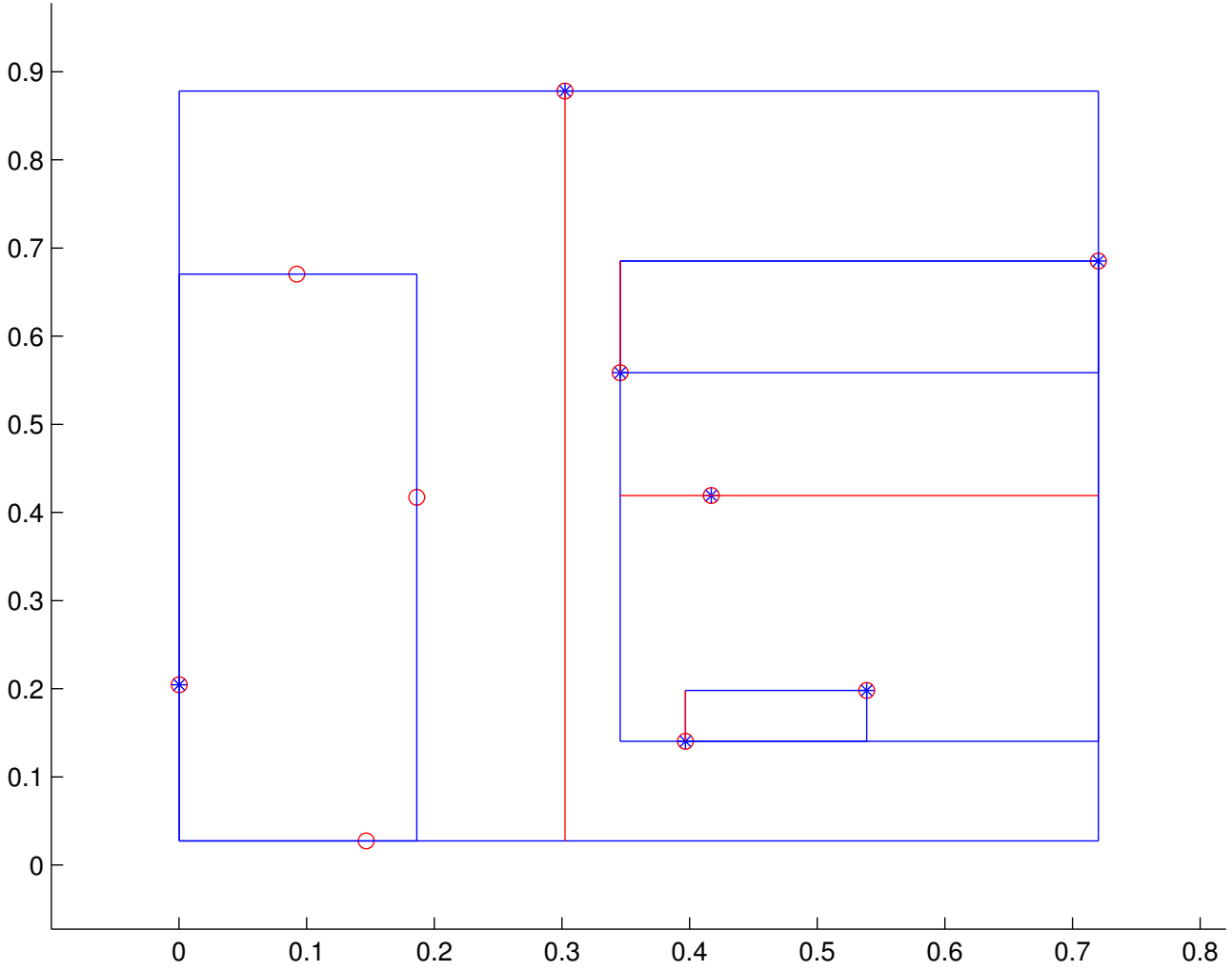
kd-Tree: Un esempio

Itero il procedimento.



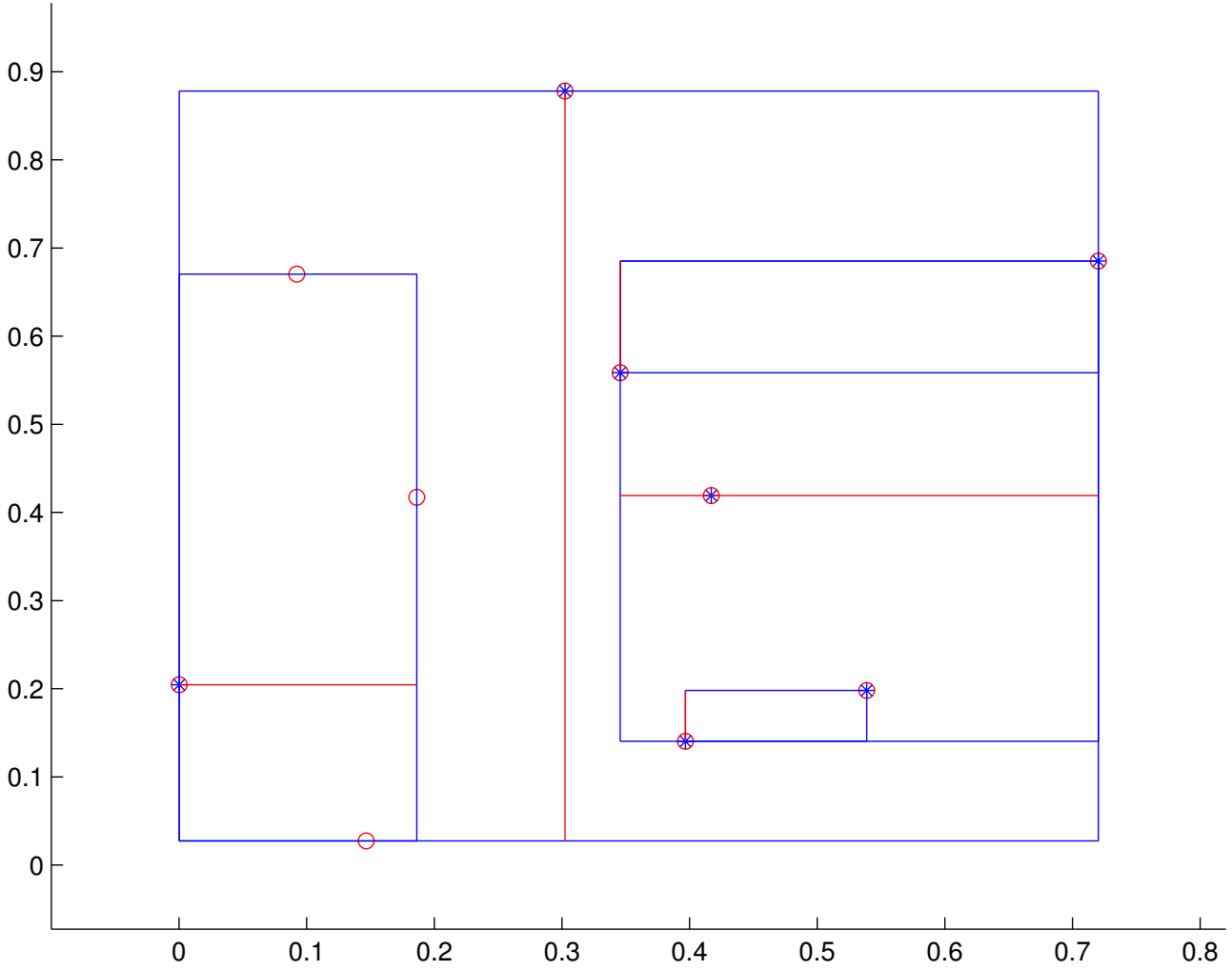
kd-Tree: Un esempio

Itero il procedimento.



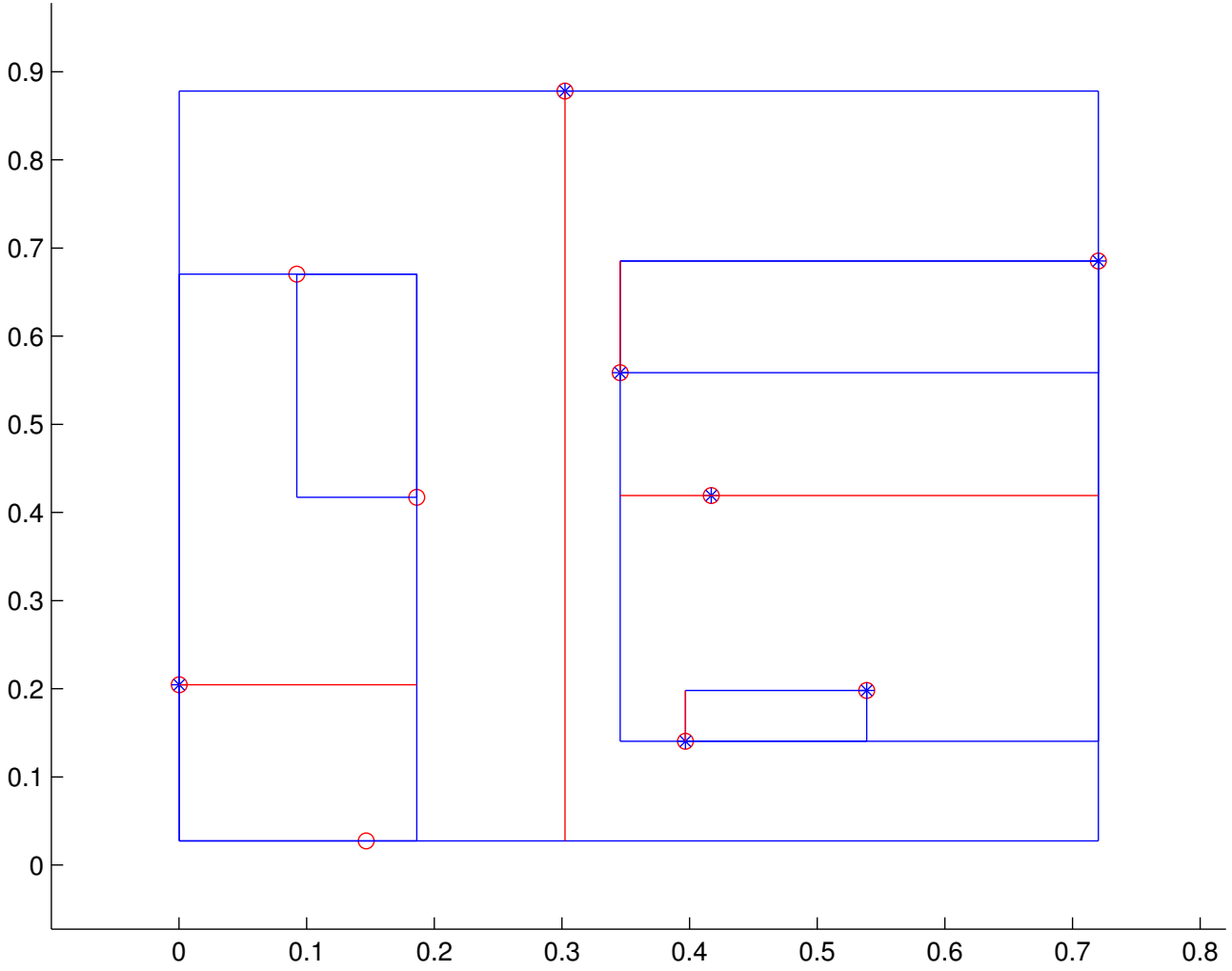
kd-Tree: Un esempio

Itero il procedimento.



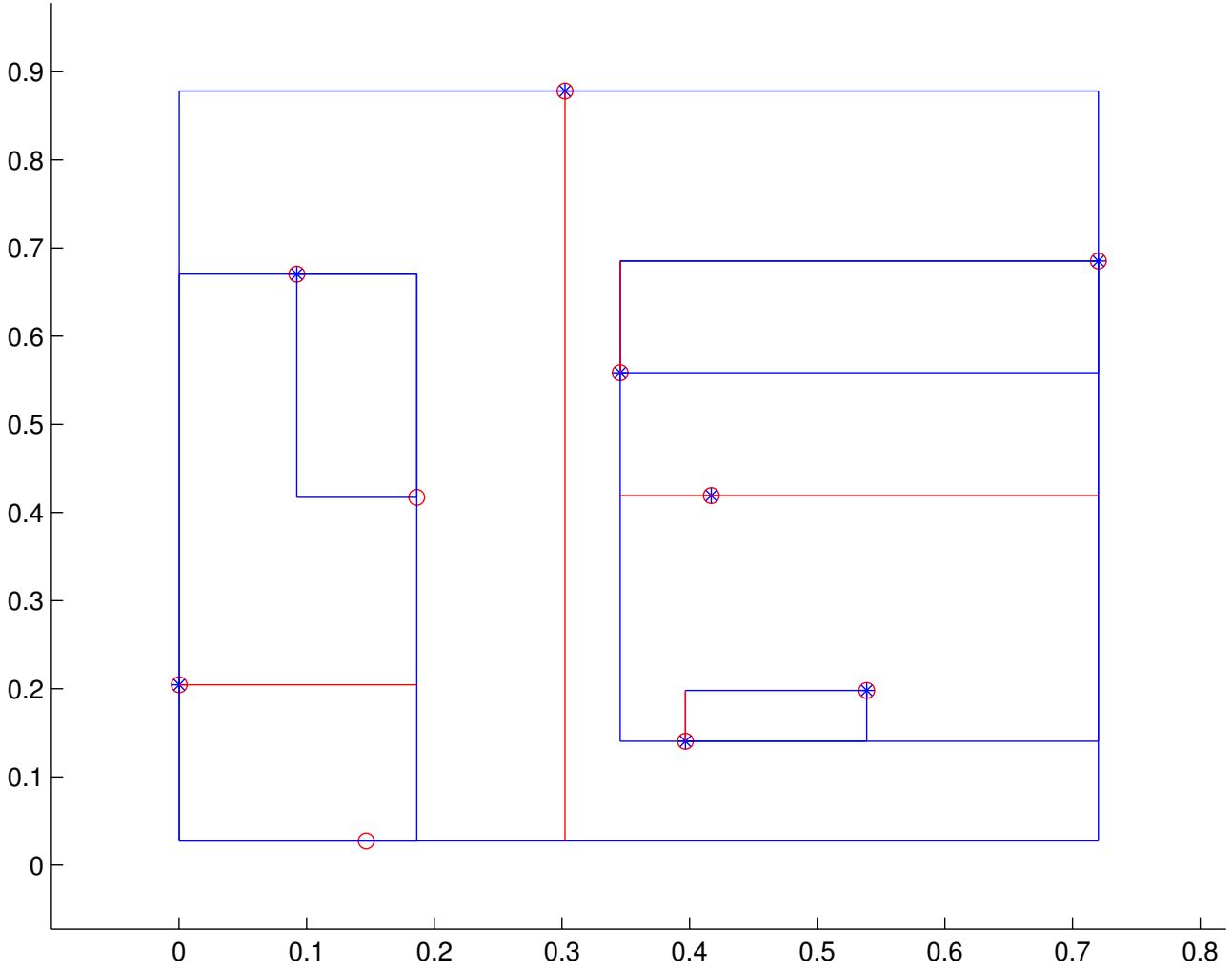
kd-Tree: Un esempio

Itero il procedimento.



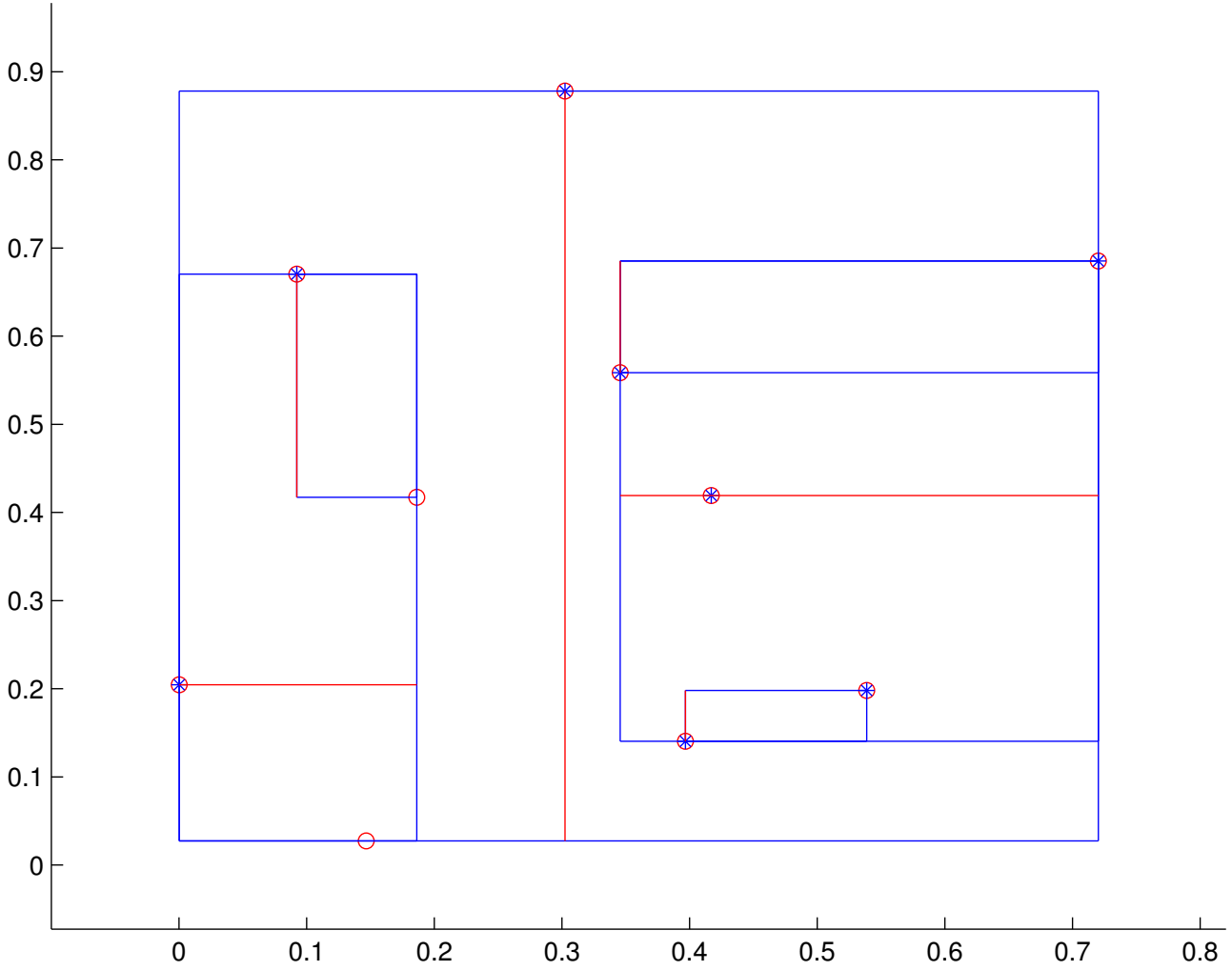
kd-Tree: Un esempio

Itero il procedimento.



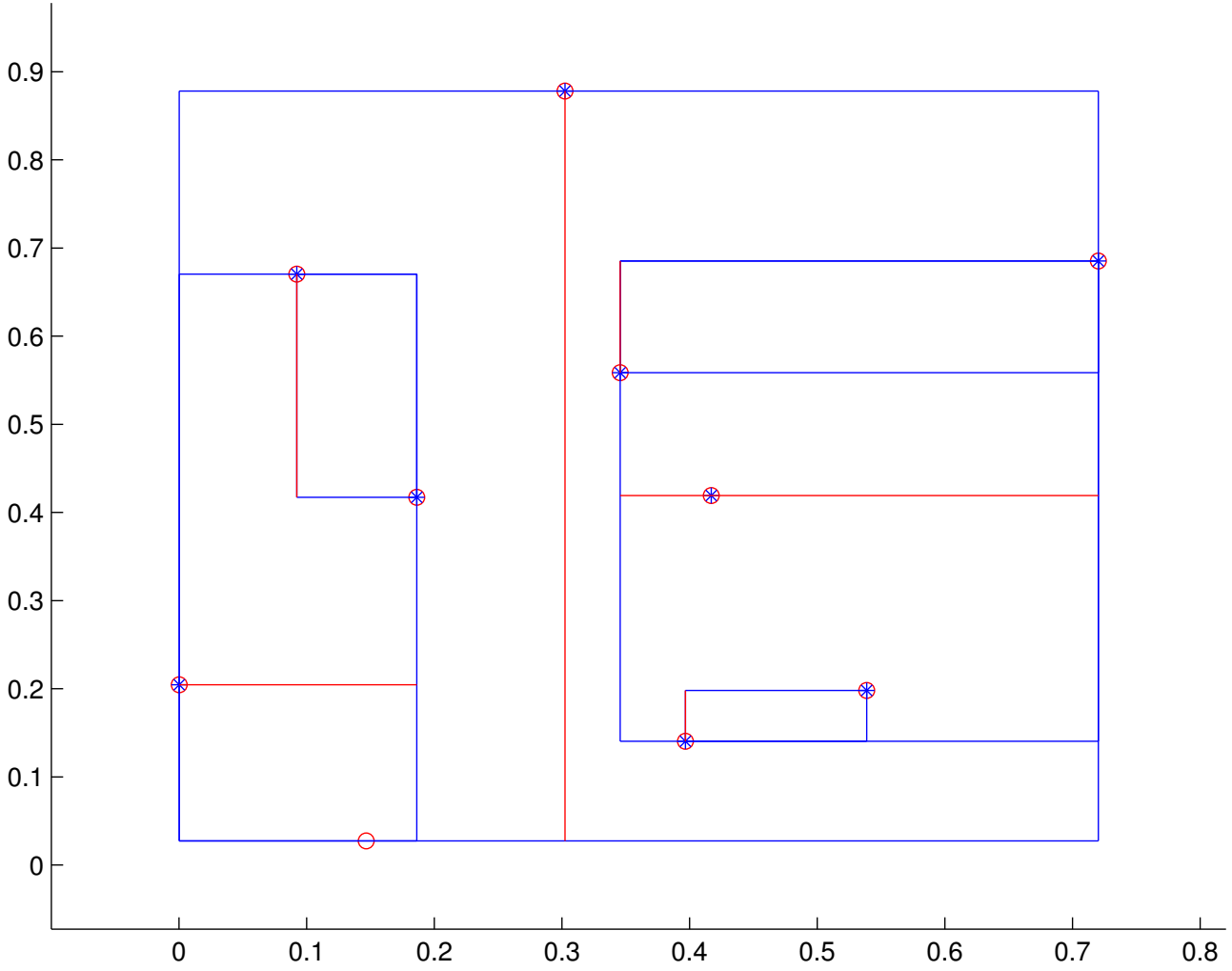
kd-Tree: Un esempio

Itero il procedimento.



kd-Tree: Un esempio

Itero il procedimento.



kd-Tree: Un esempio

Itero il procedimento.

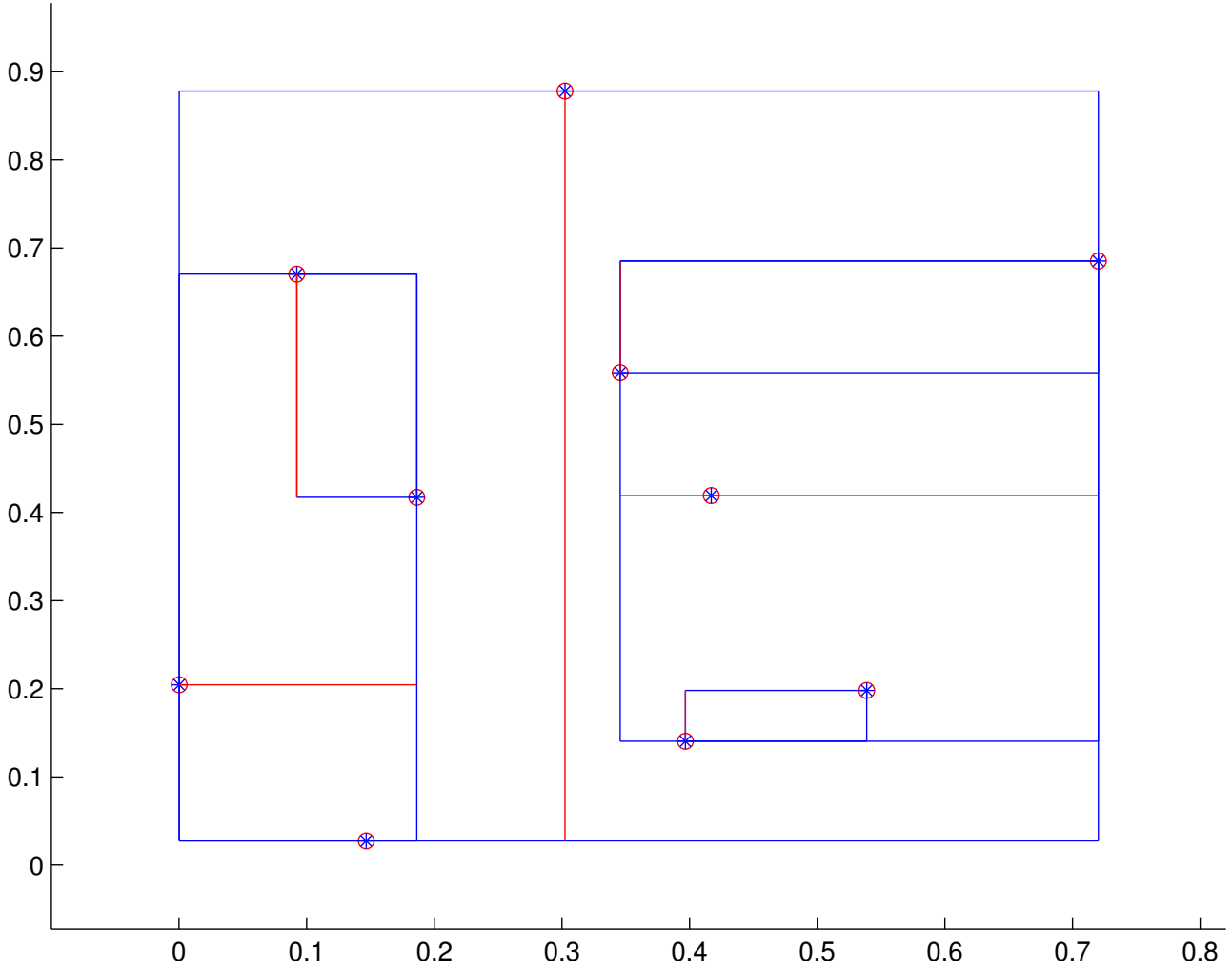


Immagine di partenza

Imresize (bicubic)

Anisotropic CSRBF

ICBI

ICBI
50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500
250
200
150
100
50
0

Expected

Expected
50
100
150
200
250
300
350
400
450
500
50
100
150
200
250
300
350
400
450
500
250
200
150
100
50
0

Risultati numerici e osservazioni

I seguenti risultati sono stati ottenuti usando la funzione di Wu C^0 come CSRBF, `localradii`=1.5, `ellradii`=5 e metodo di `canny` per trovare gli edge.

	RMSE var. interna	PSNR qualità compress.	MSSIM similarità	PEE dettagli
`imresize` (bicubic)	0.0092	+19.98 dB	0.9886	12.9078
Anisotropic CSRBF	0.0105	+18.08 dB	0.9595	**15.3118**
`ICBI`	0.0094	+19.43 dB	0.9771	10.5655

È noto che l'interpolazione con RBF soffre di un errore maggiore ai bordi del dominio per cui è del tutto lecito aspettarsi degli artefatti sul bordo esterno (che comunque in immagini mediche spesso è completamente privo di informazione significativa).

Parte II: Complessità Computazionale

Complessità Computazionale

È una branca della teoria della computabilità.

Studia le risorse minime necessarie (tempo di calcolo e memoria) per la risoluzione di un problema.

Tipico workflow per risolvere un problema:

- viene dato un problema da risolvere;
- Si scrive un programma che lo risolva;
- Si cerca che termini per ogni input;
- Il programma è computazionalmente efficiente?.

Ogni problema viene classificato in differenti classi di complessità, in base all'efficienza del migliore algoritmo noto per risolverlo.

In questo contesto problema, (sotto-)insieme (di $\mathbb{N}$) e linguaggio sono equivalenti.

Formalizzazione del problema

Problema

È posto fornendo una caratterizzazione degli input possibili ed una condizione da verificarsi. Ci saranno input con risposta **YES** ed input con risposta **NO**.

Input

Stringa $\mathbf{x}$, di un alfabeto Σ fissato, (con dimensione $|\mathbf{x}|$).

Linguaggio

$\mathcal{L} = \left\{\mathbf{x} \in \Sigma^* : \text{verifica la condizione del problema } (\mathbf{YES})\right\}$.

- Σ^* insieme di tutte le stringhe su un fissato alfabeto Σ;
- ogni stringa che non è un input finisce nelle istanze **NO**;
- Σ è finito, quindi Σ^* è numerabile: si studiano sottoinsiemi di $\mathbb{N}$;
- **è un problema di appartenenza ad un insieme**.

Modello Computazionale: le Macchine di Turing

Una k-MdT $\mathcal{M}_k = \langle Q, \Sigma, q_0, P \rangle$ con k-nastri semi-infiniti è così definita:

- Q è l'insieme finito degli stati, $q_0 \in Q$ è lo stato iniziale;
- Σ è l'alfabeto dei simboli presenti sui nastri con $\triangleright$ (first iniziale) e \$;
- P è l'insieme delle istruzioni rappresentato da una funzione δ totale

$$\delta : Q \times \Sigma^k \to \left(Q \cup \{\mathbf{h}, \mathbf{YES}, \mathbf{NO}\}\right) \times \left(\Sigma \times \{\mathrm{L}, \mathrm{R}, \mathrm{F}\}\right)^k$$

- **h**, **YES**, **NO** sono stati finali non presenti in Q e δ non è definita su di essi;
- per ogni nastro sono possibili tre movimenti: L (left), R (right), F (fermo);
- $\triangleright$ non può essere cancellato quando viene letto e la testina si posta a destra;
- Configurazione iniziale $(q_0, \underbrace{\varepsilon, \triangleright, \mathbf{x}}_{\textbf{nastro 1}}, \underbrace{\varepsilon, \triangleright, \varepsilon}_{\textbf{nastro 2}}, \ldots, \underbrace{\varepsilon, \triangleright, \varepsilon}_{\textbf{nastro k}})$; in ogni nastro la testina ha tre parametri: la stringa a sinistra, il simbolo letto e la stringa a destra.

La MdT termina solo se raggiunge una configurazione con lo stato **h**, **YES** o **NO**: se solo **h** calcola una funzione; se solo **YES** o **NO** è decisionale.

Classi in tempo deterministico

Le $\mathcal{M}_k$ sono usate per decidere l'appartenenza ad un dato linguaggio $\mathcal{L} \in \Sigma^*$.

Tempo richiesto da $\mathcal{M}_k$ per $\mathbf{x}$: $T_{\mathcal{M}_k}(\mathbf{x})$

Numero di passi di computazione necessari ad $\mathcal{M}_k$ con input $\mathbf{x}$ per terminare.

$\mathcal{M}_k$ *opera* in tempo $f(n)$ (funzione totale) se $\forall \mathbf{x}$, $T_{\mathcal{M}_k}(\mathbf{x}) \leq f(|\mathbf{x}|)$.

Classi in tempo deterministico

Un linguaggio $\mathcal{L} \subseteq \Sigma^*$ è deciso da una k-MdT $\mathcal{M}_k$ se per ogni $\mathbf{x} \in \Sigma^*$ vale:

- se $\mathbf{x} \in \mathcal{L}$ allora $\mathcal{M}_k$ con input $\mathbf{x}$ termina nello stato **YES**;
- se $\mathbf{x} \notin \mathcal{L}$ allora $\mathcal{M}_k$ con input $\mathbf{x}$ termina nello stato **NO**.

Se $\mathcal{L} \subseteq \Sigma^*$ è deciso da una $\mathcal{M}_k$ che opera in tempo $f(n)$, allora $\mathcal{L} \in \mathrm{TIME}(f(n))$.

Tesi di Church Computazionale

Teorema

Se $\mathcal{M}_k$ è una k-MdT che opera in tempo $f(n)$ possiamo costruire una 1-MdT $\mathcal{M}_1$ equivalente e che opera in tempo $\mathcal{O}(nf(n) + f(n)^2)$

Teorema (speed-up)

Sia $\mathcal{L} \in \mathrm{TIME}(f(n))$. Allora $\forall \varepsilon > 0$ si ha che $\mathcal{L} \in \mathrm{TIME}\left(\varepsilon f(n) + n + 2 + \frac{\varepsilon}{6} n\right)$

Conseguenza: potenziare l'HW migliora il tempo di esecuzione al più linearmente.

Tesi di Church Computazionale

Tutti i formalismi di calcolo ragionevoli sono computazionalmente equivalenti a meno di fattori polinomiali.

Riassunto delle classi più comuni

P	$:= \bigcup_{k \geq 0} \mathrm{TIME}(n^k)$	L	$:= \mathrm{SPACE}(\log n)$
EXPTIME	$:= \bigcup_{k \geq 0} \mathrm{TIME}(2^{(n^k)})$	PSPACE	$:= \bigcup_{k \geq 0} \mathrm{SPACE}(n^k)$
NP	$:= \bigcup_{k \geq 0} \mathrm{NTIME}(n^k)$	NL	$:= \mathrm{NSPACE}(\log n)$
NEXPTIME	$:= \bigcup_{k \geq 0} \mathrm{NTIME}(2^{n^k})$	NPSPACE	$:= \bigcup_{k \geq 0} \mathrm{NSPACE}(n^k)$

Il non-determinismo

Diversamente da quanto accade nell'equivalenza dei formalismi DFA e NFA, i risultati qui saranno meno ovvi e alcuni (ancora) sconosciuti.

Macchina di Turing non deterministica

Una 1-ND-MdT $\mathcal{M}_1^{\mathrm{nd}} = \langle Q, \Sigma, q_0, P \rangle$ è non deterministica se P è una relazione:

$$P \subseteq (Q \times \sigma) \times ((Q \cup \{\mathbf{h}, \mathbf{YES}, \mathbf{NO}\}) \times \Sigma \times \{\mathrm{L}, \mathrm{R}, \mathrm{F}\})$$

Differenza con il caso deterministico: i successori di una data configurazione non sono solo 0 (se terminante) o 1 (altrimenti) ma **una configurazione** (q, l, s, r) **ha un numero di successori pari all'insieme delle quintuple in** P **che iniziano con** q, s.

Data una configurazione iniziale $(q_0, \varepsilon, \triangleright, \mathbf{x})$ esiste una famiglia di possibili computazioni non deterministiche.

Un linguaggio $\mathcal{L} \subseteq \Sigma^*$ è deciso da una ND-MdT $\mathcal{M}^{\text{nd}}$ se per ogni $\mathbf{x} \in \Sigma^*$:

- se $\mathbf{x} \in \mathcal{L}$ allora esiste una computazione non deterministrica tale che

$$(q_0, \varepsilon, \triangleright, x) \longrightarrow_* (\mathbf{YES}, u, s, v)$$

- se $\mathbf{x} \notin \mathcal{L}$ allora non esiste una computazione non deterministica tale che

$$(q_0, \varepsilon, \triangleright, x) \longrightarrow_* (\mathbf{YES}, u, s, v)$$

  e quindi tutte le computazioni potrebbero addirittura essere non terminanti.

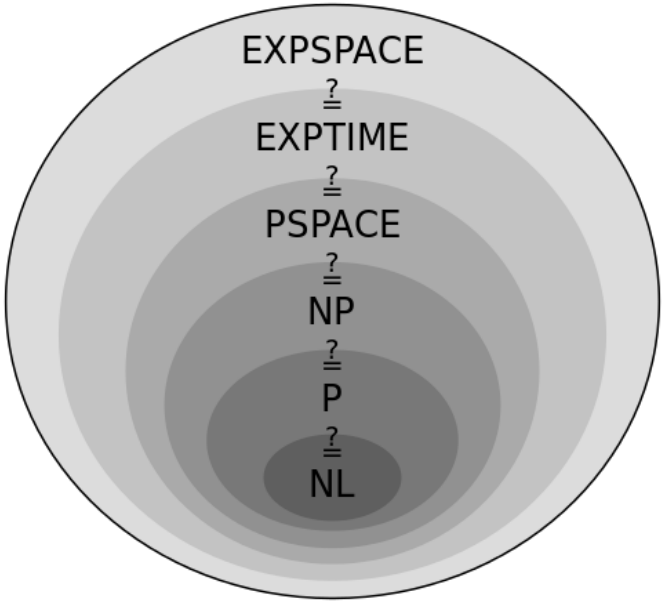
$\mathcal{M}^{\text{nd}}$ opera in tempo $f(n)$ se $\forall \mathbf{x} \in \Sigma^*$, ogni computazione è lunga al più $f(|\mathbf{x}|)$.

Se $\mathcal{L} \in \Sigma^*$ è deciso da $\mathcal{M}^{\text{nd}}$ che opera in tempo $f(n)$, allora $\mathcal{L} \in \text{NTIME}(f(n))$.

Primi risultati di inclusione

L'idea è di partizionare l'insieme dei linguaggi ricorsivi.

- $\mathsf{P} \subseteq \mathsf{NP}$
  Sia $\mathcal{L} \in \mathsf{P}$, allora $\mathcal{L}$ è deciso da una k-MdT deterministica $\mathcal{M}$ in tempo $\mathcal{O}(n^h)$, $h \in \mathbb{N}$. Quindi esiste 1-MdT $\mathcal{M}'$ equivalente ad $\mathcal{M}$ che opera in $\mathcal{O}(n^{2h})$. $\mathcal{M}'$ è non deterministica (nel suo insieme di istruzioni vi è una quintupla in P per ogni q, s).
- $\mathsf{NTIME}(f(n)) \subseteq \bigcup_{c \geq 1} \mathsf{TIME}(c^{f(n)})$;
- $\mathsf{NP} \subseteq \mathsf{EXPTIME}$ ecc.



Definizione: *Funzione di complessità propria (f.c.p.)*

Una funzione $f : \mathbb{N} \to \mathbb{N}$ è una funzione propria di complessità se è debolmente crescente ($f(n+1) \geq f(n)$) ed esiste una MdT $\mathcal{M}$ a $k > 1$ nastri tale che:

- non modifica mai il contenuto del primo nastro;
- il k-esimo nastro è a sola scrittura e la testina in esso può solo:
  - stare ferma,
  - scrivere e spostarsi a destra;
- $(q_0, \underbrace{\varepsilon, \triangleright, \mathbf{x}}_{\text{nastro 1}}, \underbrace{\varepsilon, \triangleright, \varepsilon}_{\text{nastro 2}}, \ldots, \underbrace{\varepsilon, \triangleright, \varepsilon}_{\text{nastro k}}) \longrightarrow_t (h, \underbrace{\varepsilon, \triangleright, \mathbf{x}}_{\text{nastro 1}}, \underbrace{\varepsilon, \triangleright, 0^{j_2}}_{\text{nastro 2}}, \ldots, \underbrace{\varepsilon, \triangleright, 0^{j_{k-1}}}_{\text{nastro k-1}}, \underbrace{\triangleright, 1^{f(|x|)}, \$, \varepsilon}_{\text{nastro k}})$
- $t = \mathcal{O}(|x| + f(|x|))$;
- per $i = 2, \ldots, k-1$, $j_i = \mathcal{O}(f(|x|))$.

Una MdT che calcola le funzioni proprie calcola f in unario e non usa spazio inutile.

Abbiamo bisogno di scrivere un interprete per MdT. Assumiamo tale standard:

- $\Sigma = \{1, \ldots, n\}$ per un opportuno $n = |\Sigma|$;
- $Q = \{\underbrace{|\Sigma| + 1}_{q_0}, |\Sigma| + 2, \ldots, |\Sigma| + m\}$, per un opportuno $m = |q|$;
- i numeri $|\Sigma| + |q| + 1, \ldots, |\Sigma| + |q| + 5$ denotano L,R,F,**h**,**YES**,**NO**;
- ognuni numero è rappresentato in esattamente $\lceil \log_2(|\Sigma| + |q| + 6) \rceil$ bits. ρ è la funzione che associa la rappresentazione ad ogni simbolo.

Una k-MdT $\mathcal{M} = \langle Q, \Sigma, q_0, P \rangle$ è rappresentata nel modo seguente:

- $|Q|$ in binario senza zeri inutili davanti, seguito da un `;';
- $|\Sigma|$ in binario senza zeri inutili davanti, seguito da un `;';
- k numero di nastri in binario senza zeri inutili davanti, seguito da un `;';
- per ogni tupla $T = \langle q, s_1, \ldots, s_k, q', s'_1, \mathcal{M}_1, \ldots, s'_k, \mathcal{M}_k \rangle$ di P sia $\rho(T) = \rho(q)\rho(s_1)\ldots\rho(s_k)\rho(q')\rho(s'_1)\rho(\mathcal{M}_1)\ldots\rho(s'_k)\rho(\mathcal{M}_k)$
- se $P = \{T_1, \ldots, T_r\}$ allora la rappresentazione sarà $\rho(T_1)\ldots\rho(T_r)$.

La descrizione della MdT $\mathcal{M}$ occupa pertanto uno spazio

$$\lceil \log_2 |q| \rceil + \lceil \log_2 |\Sigma| \rceil + \lceil log_2 k \rceil + 6\Big(|q||\Sigma|^k \lceil \log_2(|\Sigma| + |Q| + 6) \rceil\Big) + 4.$$

Halting Problem limitato, con input **x** (anche descrizione di una MdT):

$$H_f := \Big\{\mathcal{M}; \mathbf{x} : \mathcal{M} \text{ MdT che accetta } \mathbf{x} \text{ in } \leq f(|x|) + 5|x| + 4 \text{ passi}\Big\}.$$

Teorema

- $H_f \in \mathrm{TIME}(f(n)^3)$; $H_f \notin \mathrm{TIME}\Big(\Big\lfloor \frac{n}{2} \Big\rfloor\Big)$;
- *se* $f(n) \geq n$ *(f.c.p.),* $\mathrm{TIME}(f(n)) \subset \mathrm{TIME}(f(2n+1)^3)$;
- P $\subset$ EXPTIME.

Quindi $\mathrm{P} \subseteq \mathrm{TIME}(2^n) \subset \mathrm{TIME}((2^{2n+1})^3) = \mathrm{TIME}(2^{6n+3}) \subseteq \mathrm{EXPTIME}$ dunque

$$\mathrm{P} \subseteq \mathrm{NP} \subseteq \mathrm{EXPTIME}$$

e almeno una delle due inclusioni è propria ma non sappiamo quale.

Complessità in Spazio

Spesso non occorre tenere solo conto del tempo trascorso ma anche dello spazio utilizzato, sebbene riutilizzabile.

k-MdT con input ed output

Una MdT a k-nastri e con input ed output (I/O-k-MdT) $\mathcal{M}$ è una k-MdT con almeno due nastri. Il primo contiene l'input ed è a sola lettura. L'ultimo contiene l'output ed è a sola scrittura. Su questo la testina può solo:

- stare ferma;
- spostarsi a destra.

Non è un nuovo modello di MdT ma un altro formalismo.

Classi in spazio deterministico

Classi in spazio deterministico

Data una I/O-k-MdT $\mathcal{M}$ e un input $\mathbf{x}$, se

$$(q_0, \underbrace{\varepsilon, \triangleright, \mathbf{x}}_{\textbf{nastro 1}}, \underbrace{\varepsilon, \triangleright, \varepsilon}_{\textbf{nastro 2}}, \ldots, \underbrace{\varepsilon, \triangleright, \varepsilon}_{\textbf{nastro k}}) \rightarrow_x (q, \underbrace{u_1, s_1, v_1}_{\textbf{nastro 1}}, \ldots, \underbrace{u_k, s_k, v_k}_{\textbf{nastro k}})$$

con $q \in \{\mathbf{h}, \mathbf{YES}, \mathbf{NO}\}$ allora diremo che lo spazio richiesto da $\mathcal{M}$ per $\mathbf{x}$ è

$$\sum_{i=2}^{k-1} |u_i| + |v_i| + 1.$$

$\mathcal{M}$ opera in spazio $f(n)$ se per ogni input $\mathbf{x}$ lo spazio richiesto da $\mathcal{M}$ per $\mathbf{x}$ è minore o uguale a $f(|\mathbf{x}|)$.

Se esiste una I/O-k-MdT $\mathcal{M}$ che decide un linguaggio $\mathcal{L} \subseteq \Sigma^*$ e opera in spazio $f(n)$ allora $\mathcal{L} \in \mathrm{SPACE}(f(n))$

Teorema

Le classi deterministiche in spazio e tempo sono collegate da:

$$\mathrm{SPACE}(f(n)) \subseteq \mathrm{TIME}\left(2^{\mathcal{O}(f(n))}\right).$$

Sia $\mathcal{L} \in \mathrm{SPACE}(f(n))$ ed $\mathcal{M}$ la I/O-k-MdT che decide $\mathcal{L}$. Se durante la computazione $\mathcal{M}$ si trovasse nella medesima configurazione $(q, l_1, s_1, r_1, \ldots, l_k, s_k, r_k)$ per due volte, il determinismo garantirebbe che la situazione si ripeterebbe all'infinito e dunque $\mathcal{M}$ non terminerebbe (assurdo). Pertanto il numero delle configurazioni diverse (l'ultimo nastro non conta in quanto non viene letto) è dell'ordine di

$$\underbrace{(Q+3)}_{\textbf{stati}} \underbrace{\Sigma^{\mathcal{O}(|f(x)|)}}_{\textbf{stringhe sui nastri}} \underbrace{(|\mathbf{x}|+1)f(|\mathbf{x}|)^k}_{\textbf{posizione testina}}.$$

Teorema

- $\mathrm{L} \subseteq \mathrm{P}$;
- $\mathrm{PSPACE} \subseteq \mathrm{EXPTIME}$.

Classi in spazio non deterministico

Classi in spazio non deterministico

Una I/O-k-MdT $\mathcal{M}$ decide $\mathcal{L} \in \Sigma^*$ in spazio $f(n)$ se $\mathcal{M}$ decide $\mathcal{L}$ e, per ogni $\mathbf{x} \in \Sigma^*$, se per ogni configurazione non deterministicamente raggiungibile in qualche computazione

$$(q_0, \underbrace{\varepsilon, \triangleright, \mathbf{x}}_{\textbf{nastro 1}}, \underbrace{\varepsilon, \triangleright, \varepsilon}_{\textbf{nastro 2}}, \ldots, \underbrace{\varepsilon, \triangleright, \varepsilon}_{\textbf{nastro k}}) \rightarrow_x (q, \underbrace{u_1, s_1, v_1}_{\textbf{nastro 1}}, \ldots, \underbrace{u_k, s_k, v_k}_{\textbf{nastro k}})$$

vale che

$$\sum_{i=2}^{k-1} |u_i| + |v_i| + 1 \leq f(|\mathbf{x}|).$$

In tal caso si dice che $\mathcal{L} \in \mathrm{NSPACE}(f(n))$.

In base a questa definizione $\mathcal{M}$ potrebbe essere potenzialmente non terminante.

Altri risultati di inclusione

Teorema

- $\mathrm{SPACE}(f(n)) \subseteq \mathrm{NSPACE}(f(n))$;
- $\mathrm{NTIME}(f(n)) \subseteq \mathrm{SPACE}(f(n))$;
- $\mathrm{NSPACE}(f(n)) \subseteq \mathrm{TIME}(2^{\mathcal{O}(f(n)+\log n)})$;
- $\mathrm{L} \subseteq \mathrm{NL} \subseteq \mathrm{P}$;
- $\mathrm{PSPACE} \subseteq \mathrm{NPSPACE}$;
- *se* $f(n) \geq \log n$ *f.c.p. allora* $\mathrm{NSPACE}(f(n)) \subseteq \mathrm{SPACE}(f(n)^2)$;
- $\mathrm{NPSPACE} = \mathrm{PSPACE}$;
- $\mathrm{NP} \subseteq \mathrm{PSPACE}$.

Quindi

$$\mathrm{L} \subseteq \mathrm{NL} \subseteq \underbrace{\mathrm{P} \subseteq \mathrm{NP} \subseteq (\mathrm{PSPACE} = \mathrm{NPSPACE}) \subseteq \mathrm{EXPTIME}}_{\textbf{inclusione stretta}} \subseteq \mathrm{NEXPTIME}.$$

Riduzioni ed NP-Completezza

Lo scopo è ridurre, tramite una funzione ricorsiva e calcolabile in modo semplice, un problema dato ad un altro problema noto: se è possibile, risolvere il primo non può essere più difficile che risolvere il secondo.

Dati due linguaggi $\mathcal{L}_1$ ed $\mathcal{L}_2$, si dice che $\mathcal{L}_1 \subseteq \Sigma_1^*$ è riducibile a $\mathcal{L}_2 \subseteq \Sigma_2^*$ (in breve $\mathcal{L}_1 \prec \mathcal{L}_2$) se esiste una funzione $f : \Sigma_1^* \to \Sigma_2^*$ t.c.

- f è calcolabile da una I/O-k-MdT (deterministica) in spazio $\mathcal{O}(\log n)$ e
- $\forall \mathbf{x} \in \Sigma_1^*$ si ha che $\mathbf{x} \in \mathcal{L}_1$ se e solo se $f(\mathbf{x}) \in \mathcal{L}_2$.

Poichè L $\subseteq$ P, se $\mathcal{L}_2 \in C$ e $\mathcal{L}_1 \prec \mathcal{L}_2$ implica $\mathcal{L}_1 \in C$, allora C è detta chiusa per riduzione.

Tutte le classi introdotte finora, usando $f \in \mathcal{O}(\log n)$ sono chiuse per riduzione.

Problema C-completo

Data una classe C, un problema $\mathcal{P} \in C$ è C-completo se ogni problema della classe può essere ridotto ad $\mathcal{P}$.

Esempio

Problema SAT (SATisfability)

- **Input**: Una formula logica φ data come congiunzione di disgiunzioni (clausole) di letterali, costruita su un insieme di variabili ν, ovvero:

$$\varphi \equiv (l_1^1 \vee \cdots \vee l_{n_1}^1) \wedge \cdots \wedge (l_1^m \vee \cdots \vee l_{n_1}^m)$$

  ove ogni l_j^i è o una variabile $X \in \nu$ o la negazione di una variabile $\neg X$ con $X \in \nu$.

- **Problema**: Esiste un assegnamento per le variabili in ν che renda vera la formula?

Problema HP (Hamiltonian Path)

- **Input**: Un grafo (non diretto) $G = \langle N, E \rangle$.
- **Problema**: Esiste un cammino che visiti ogni nodo esattamente una volta?

Sembrano problemi diversi, ma:

Lemma

È possibile costruire una funzione f che trasformi un'istanza di **HP** (cioè un grafo) in un'istanza di **SAT**:

$$\mathbf{HP} \leq \mathbf{SAT}.$$

Quindi **HP** non può essere più difficile di **SAT** e si può trasformare il problema di appartenenza ad **HP** al problema di appartenenza a **SAT**.

Teorema

SAT *è* NP-*completo.*

Un problema tipo **SAT** presenta una relativa facilità di verifica della correttezza della soluzione ma un'ampiezza esponenziale dello spazio di ricerca della stessa.

Essendo **SAT** NP-completo e poichè la classe NP è chiusa per riduzione, se si dimostra che $L \in$ NP e **SAT** $\leq L$, allora si può dimostrare la NP-completezza di altri problemi.

Alcuni problemi completi per le classi viste

- classe L: **DFA-Acceptation**
  - **Input**: Descrizione di un DFA $\mathcal{M}$ e una stringa **x**.
  - **Problema**: Stabilire se $\mathbf{x} \in L(\mathcal{M})$.
- classe NL: **Reachability**
  - **Input**: Un grafo diretto $G = \langle N, E \rangle$ e due suoi nodi s e t.
  - **Problema**: Stabilire se esiste un cammino da s a t.
- classe P: **Circuit-Value**
  - **Input**: Un circuito logico con porte *and, or* e *not* con una sola uscita e i valori fissati per gli ingressi.
  - **Problema**: Stabilire se l'output è TRUE.
- classe NP: **SAT** (visto prima).

- classe PSPACE: **QSAT**
  - **Input**: Una espressione booleana in forma normale congiuntiva (inf) $\phi = \bigwedge_{i=1}^{n} C_i$ con C_i clausole utilizzanti variabili booleane $x_1, \ldots, \mathbf{x}_n$.
  - **Problema**: Stabilire se $\exists x_1 \forall x_2 \exists x_3 \ldots Q_n X_n \phi$ sia o meno soddisfacibile, ove Q_n è $\exists$ se n è pari, $\forall$ altrimenti.
- classe EXPTIME: **Succint Circuit Value**
  - **Input**: Un circuito booleano dato in rappresentazione implicita, un assegnamento di verità per le variabili in input.
  - **Problema**: Stabilire se tale assegnamento rende vero l'output del circuito booleano.
- classe NEXPTIME: **SB-SAT**
  - **Input**: Una formula della logica del prim'ordine φ costruita con simboli di relazione, di costante e variabili $X_1, \ldots, X_n, Y_1, \ldots, Y_l$ (niente simboli di funzione, niente uguaglianza).
  - **Problema**: Stabilire se $\exists X_1 \ldots \exists X_n \forall Y_1 \ldots \forall Y_l \varphi$ ammette modelli (cioè è soddisfacibile).

Grazie per l'attenzione.