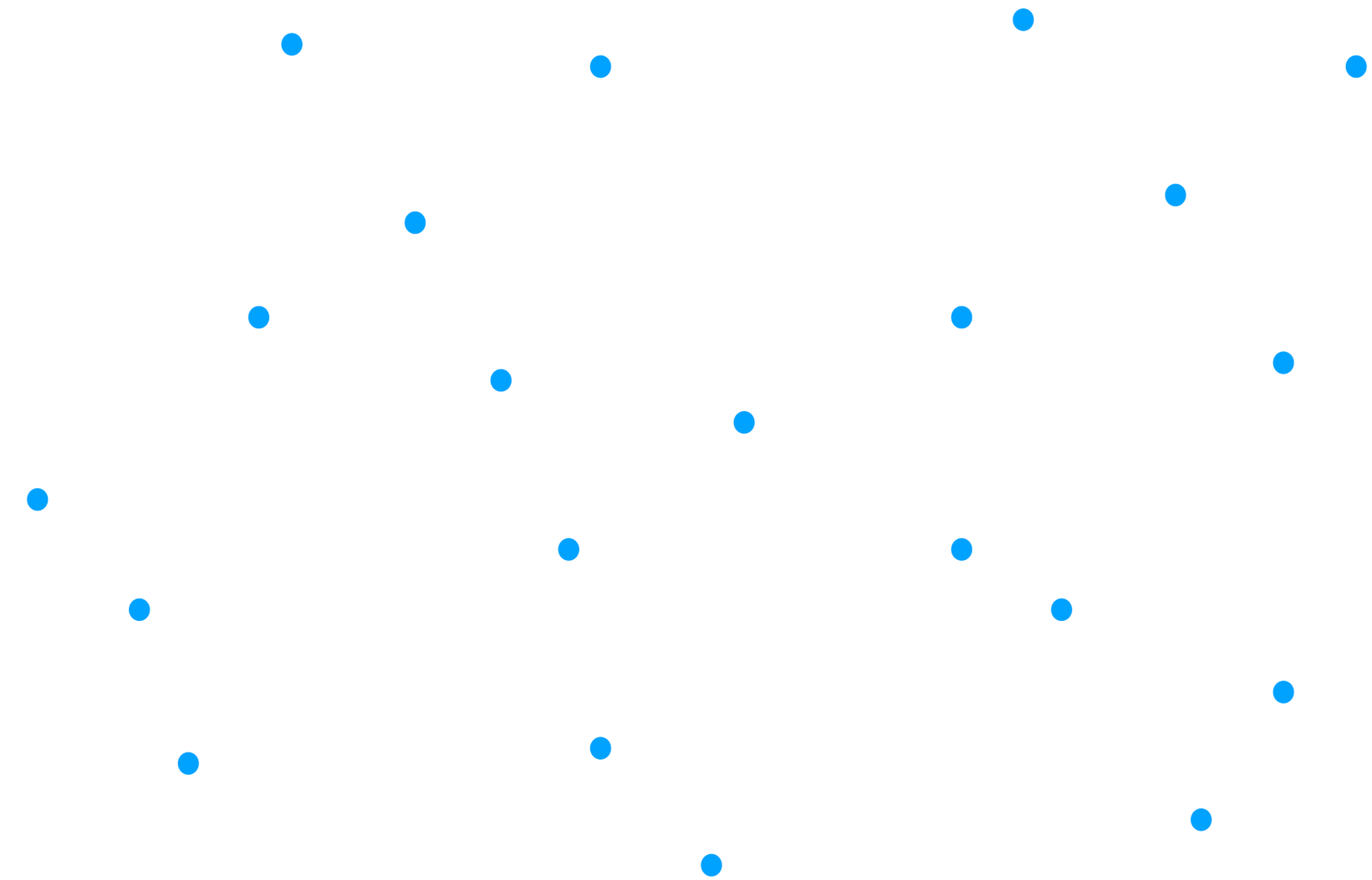
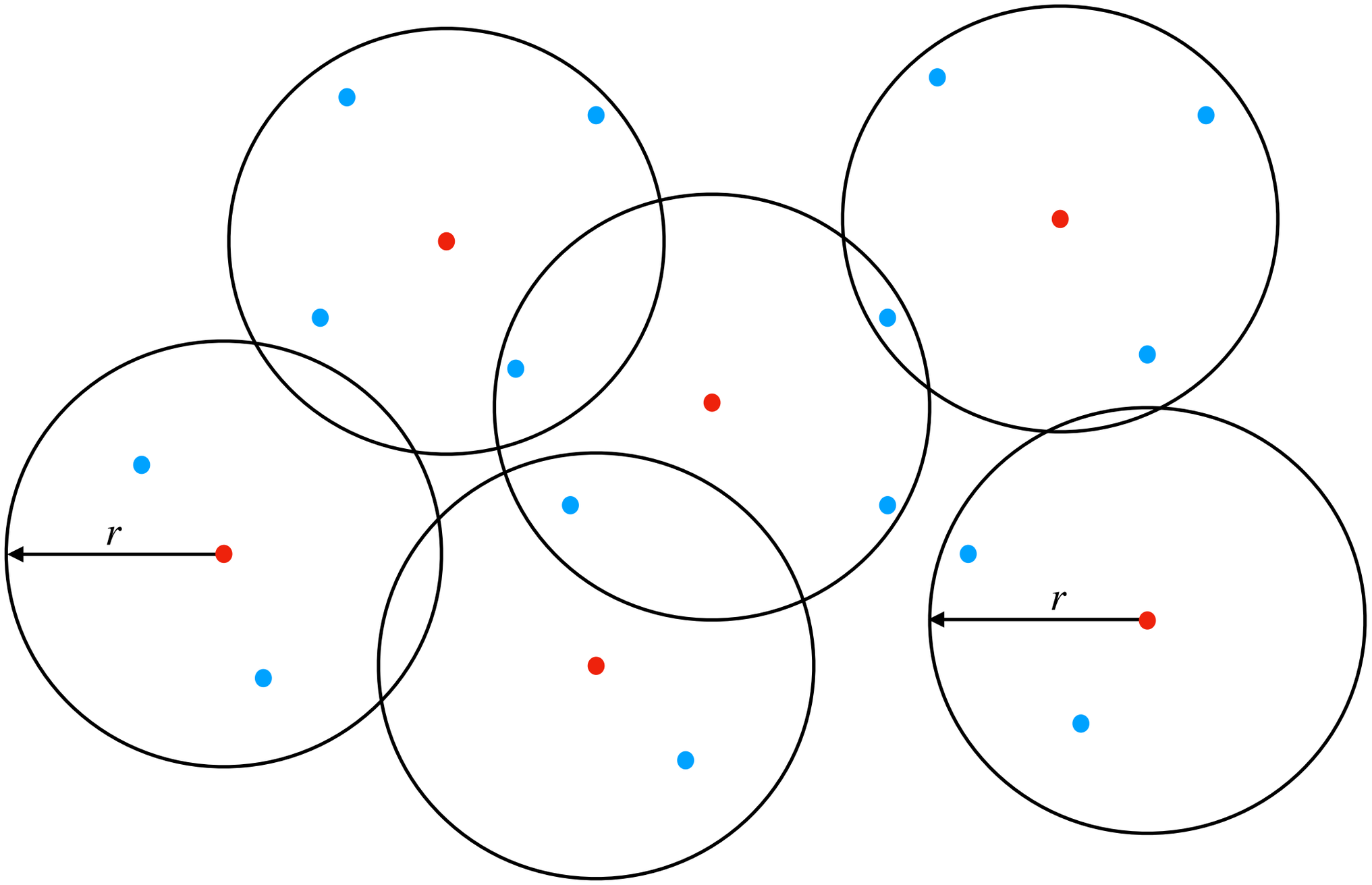
High Dimensional Clustering and Applications

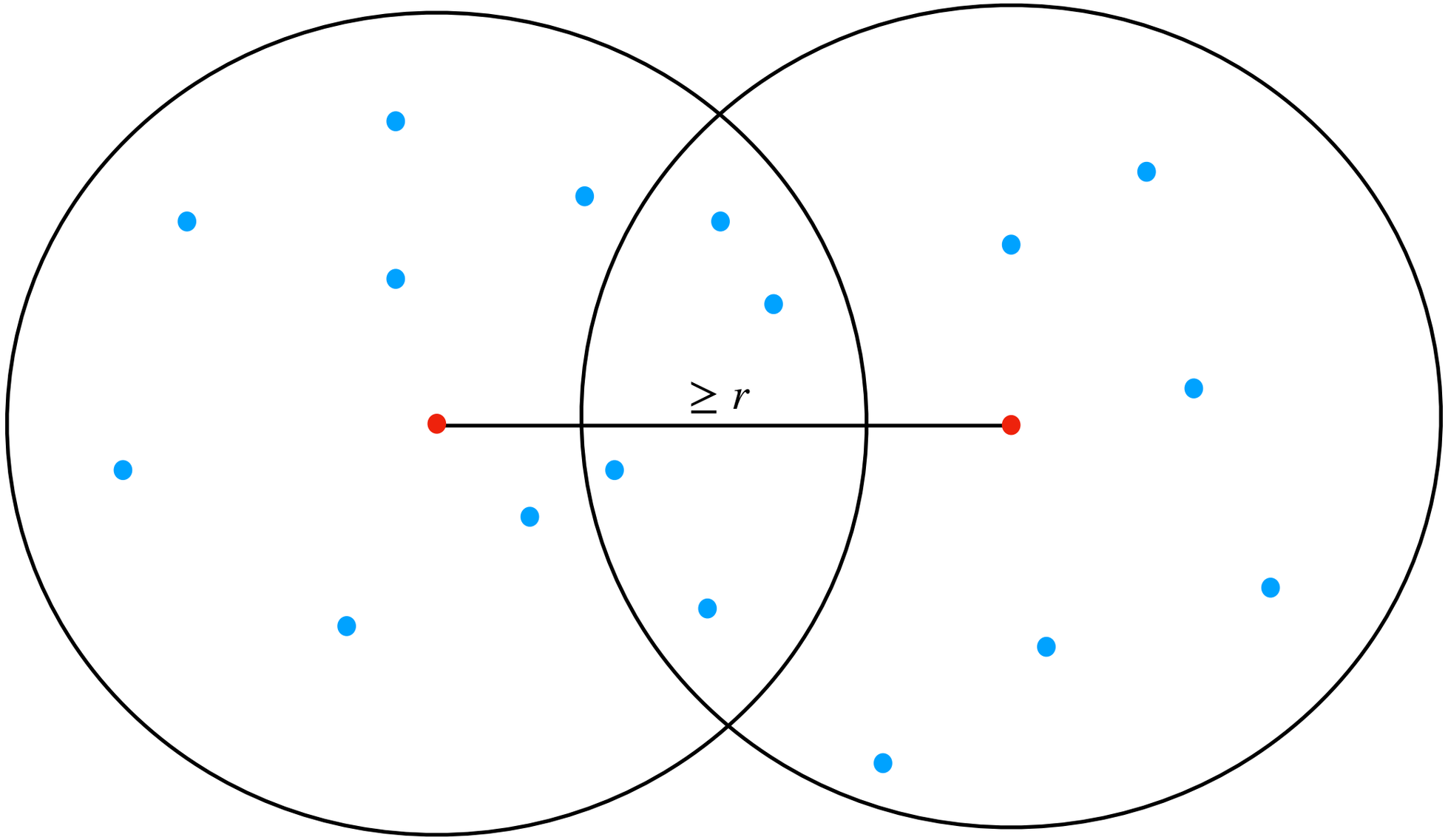
R-nets



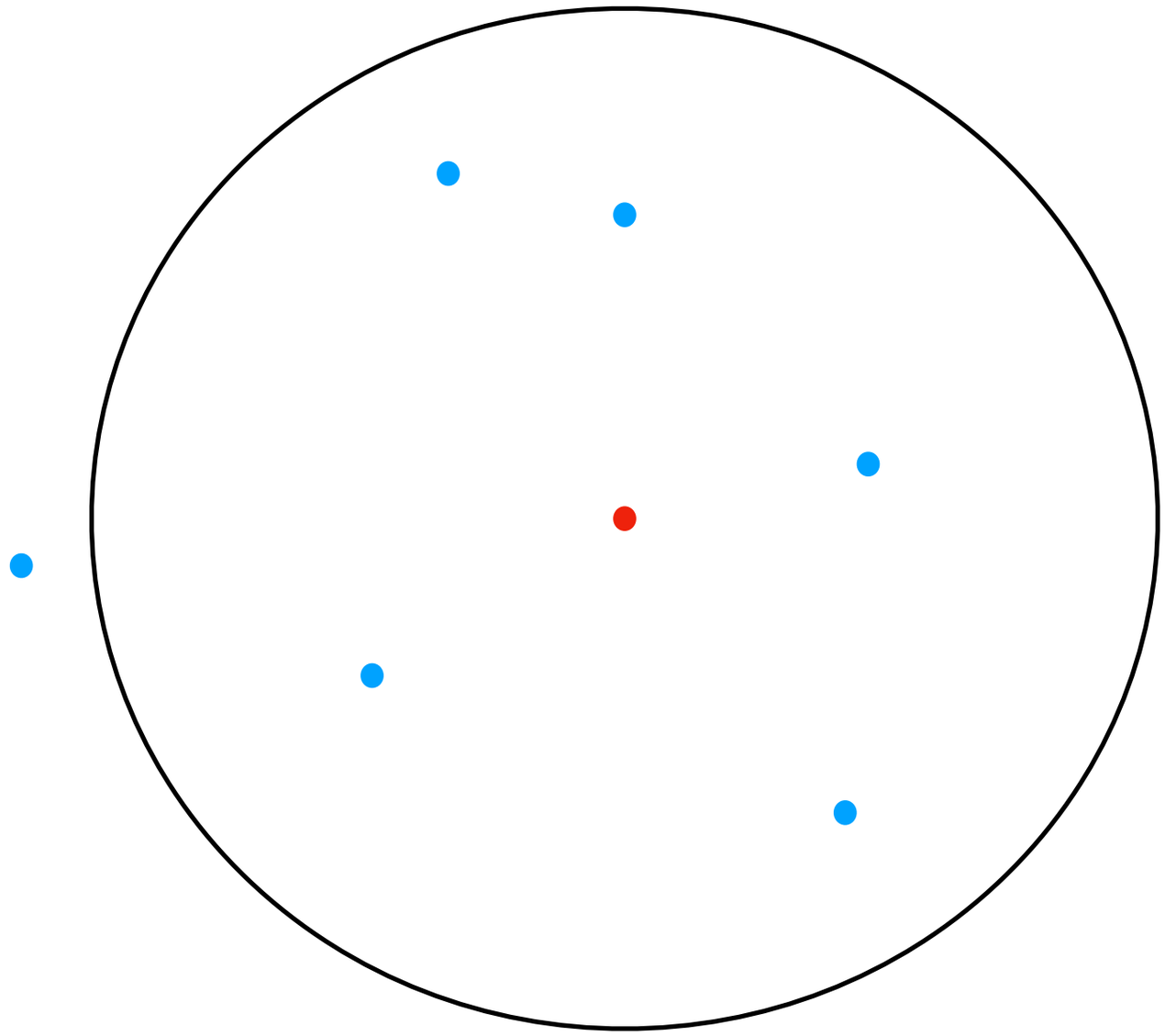
Covering



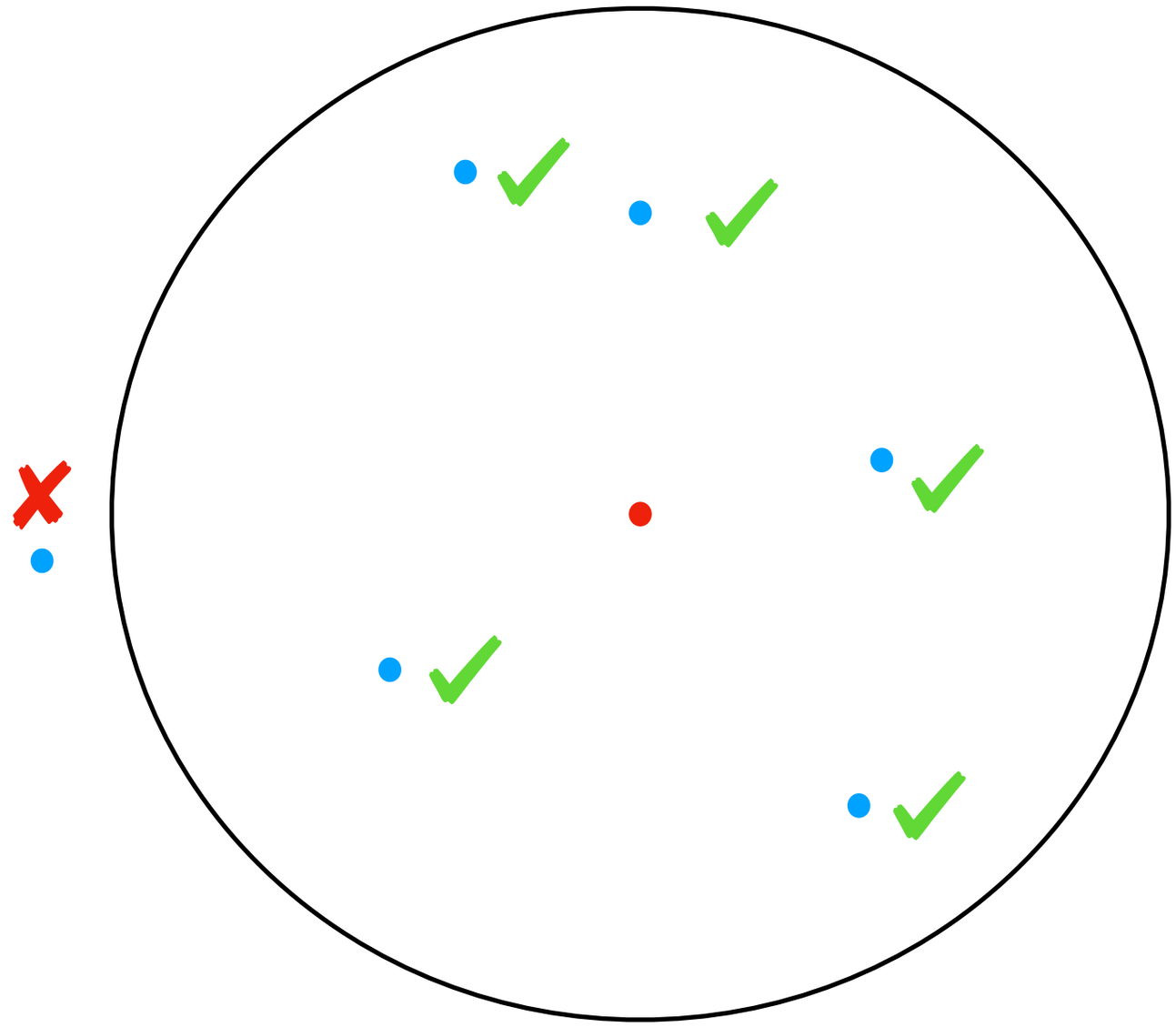
Packing



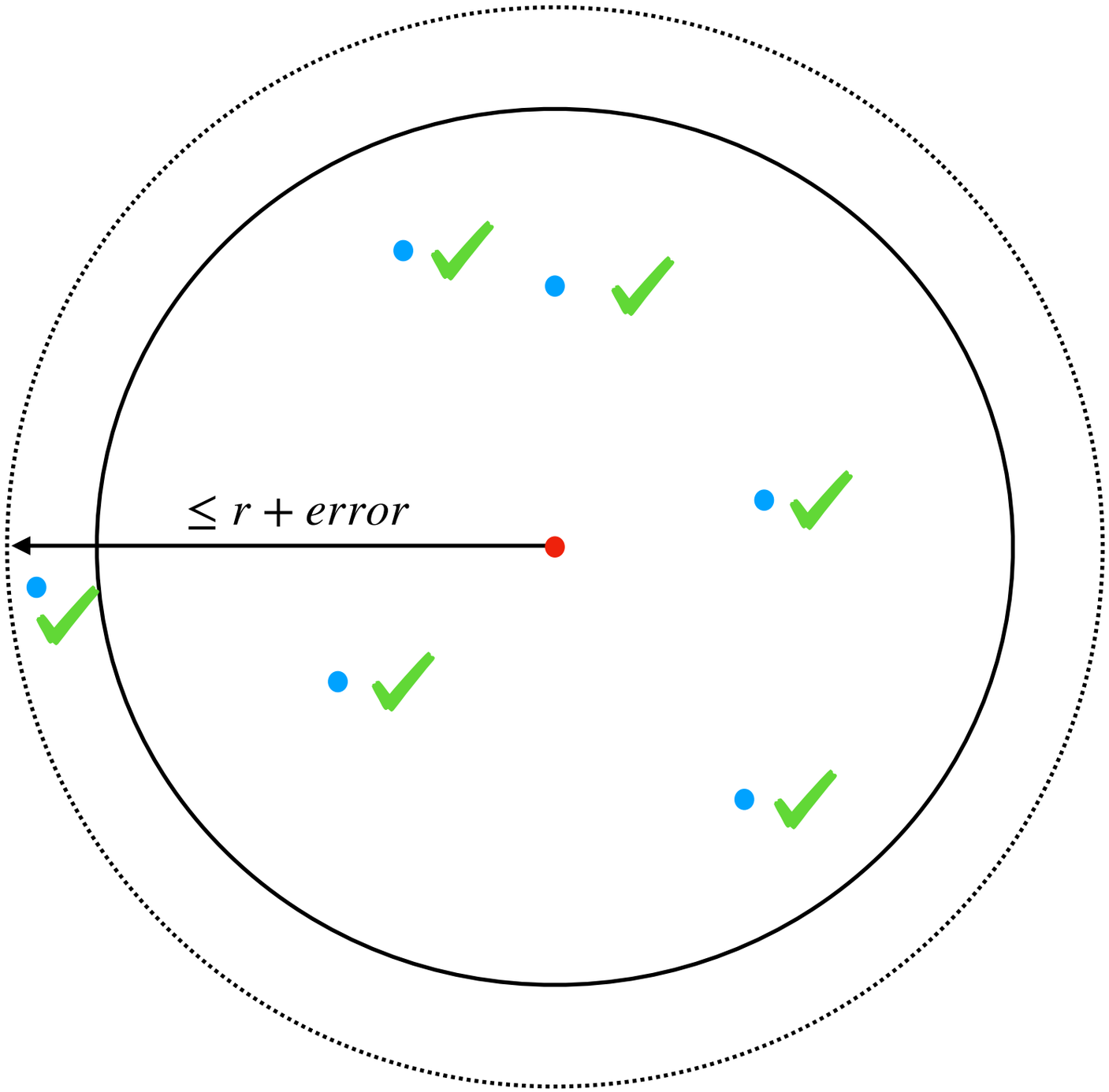
Approximate r-net



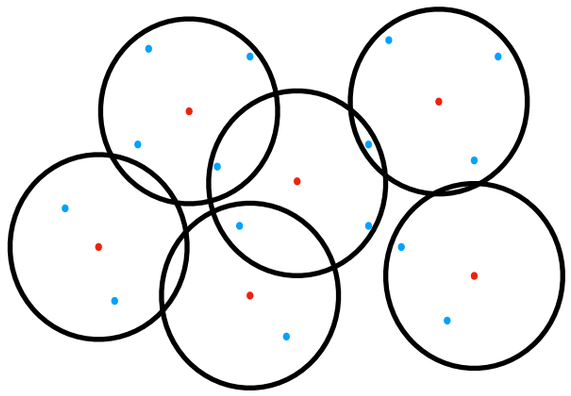
Approximate r-net



Lift covering



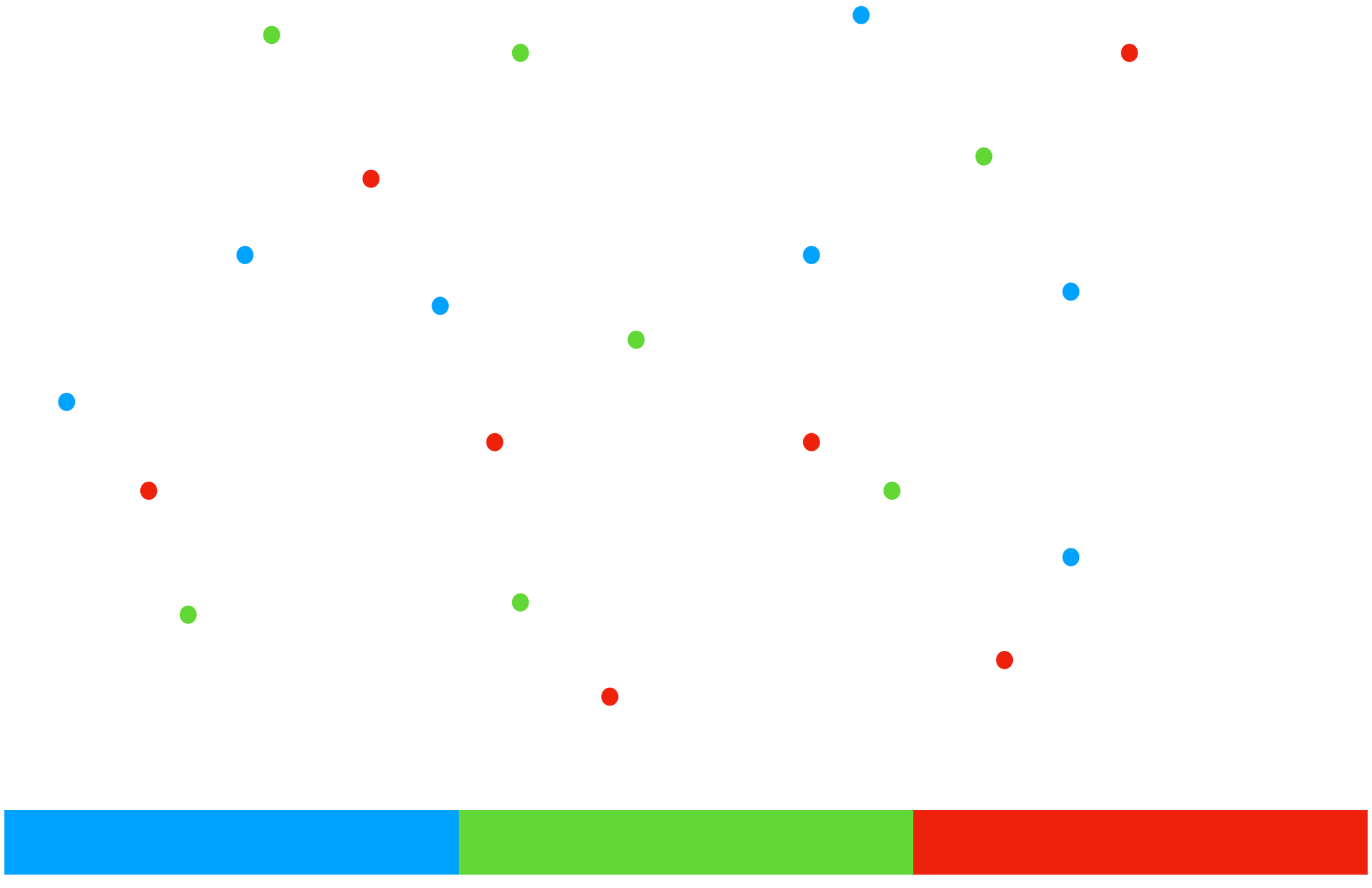
R-nets



Distance matrix

	p_1	p_2		p_n
	yes	no	...	yes
	no	no	...	yes
	no	yes	...	no

Grouping points

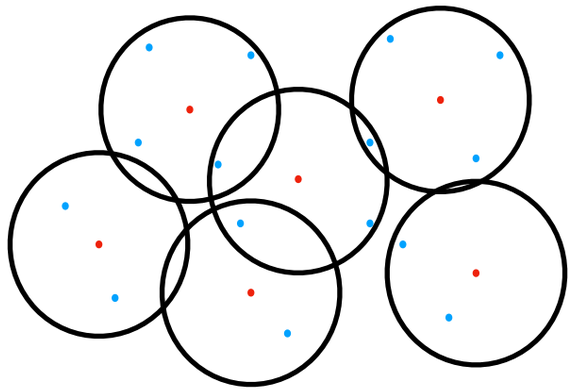


Distance matrix

	p_1	p_2	...	p_n
	yes	no	...	yes
	no	no	...	yes
	no	yes	...	no

How do we efficiently build such a matrix?

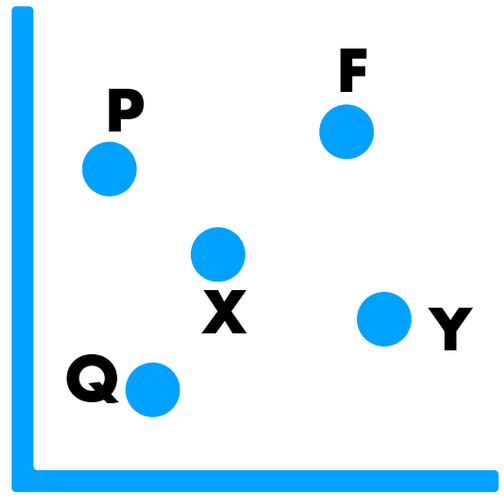
R-nets



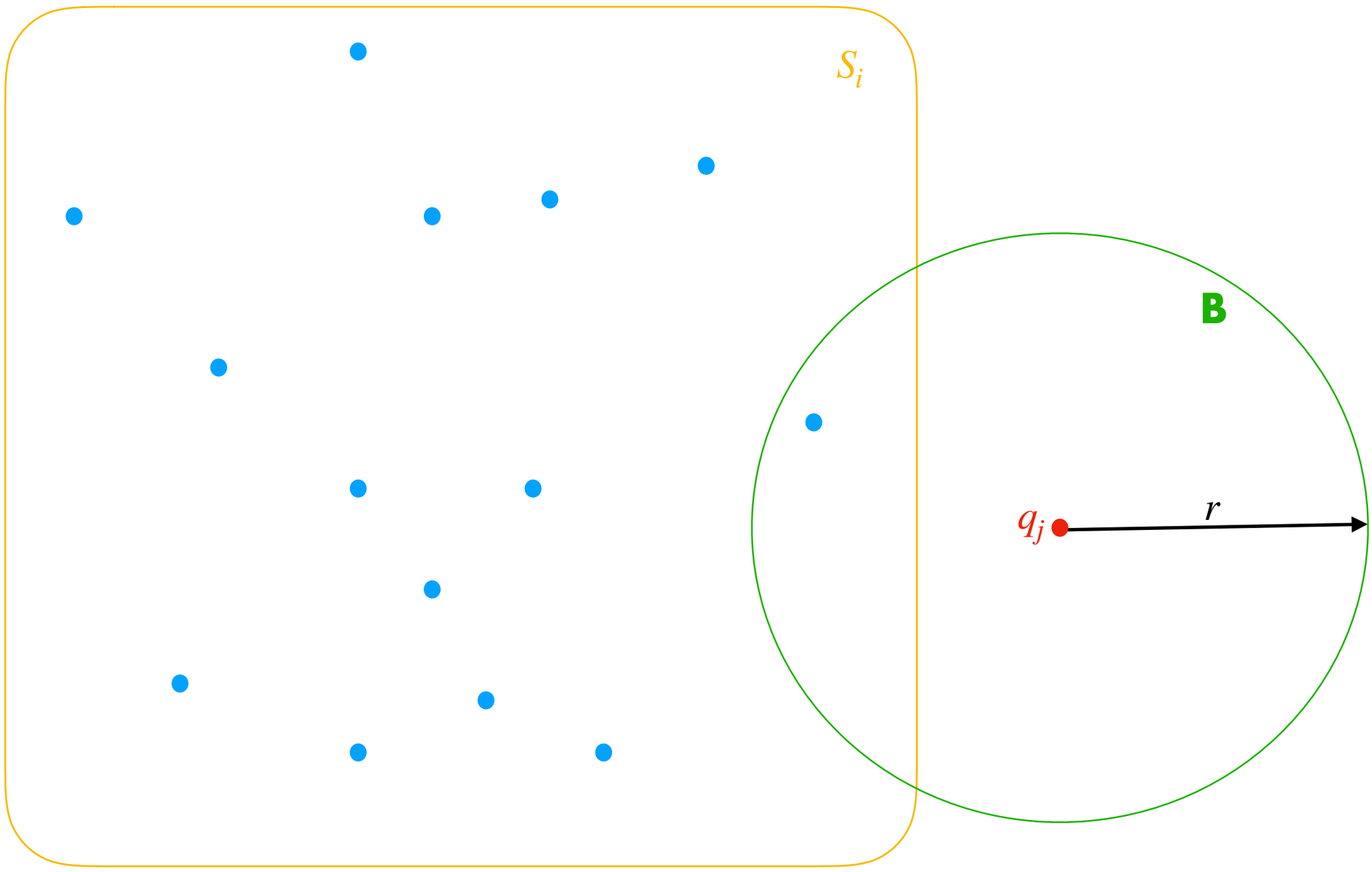
Distance matrix

	p_1	p_2		p_n
	yes	no	...	yes
	no	no	...	yes
	no	yes	...	no

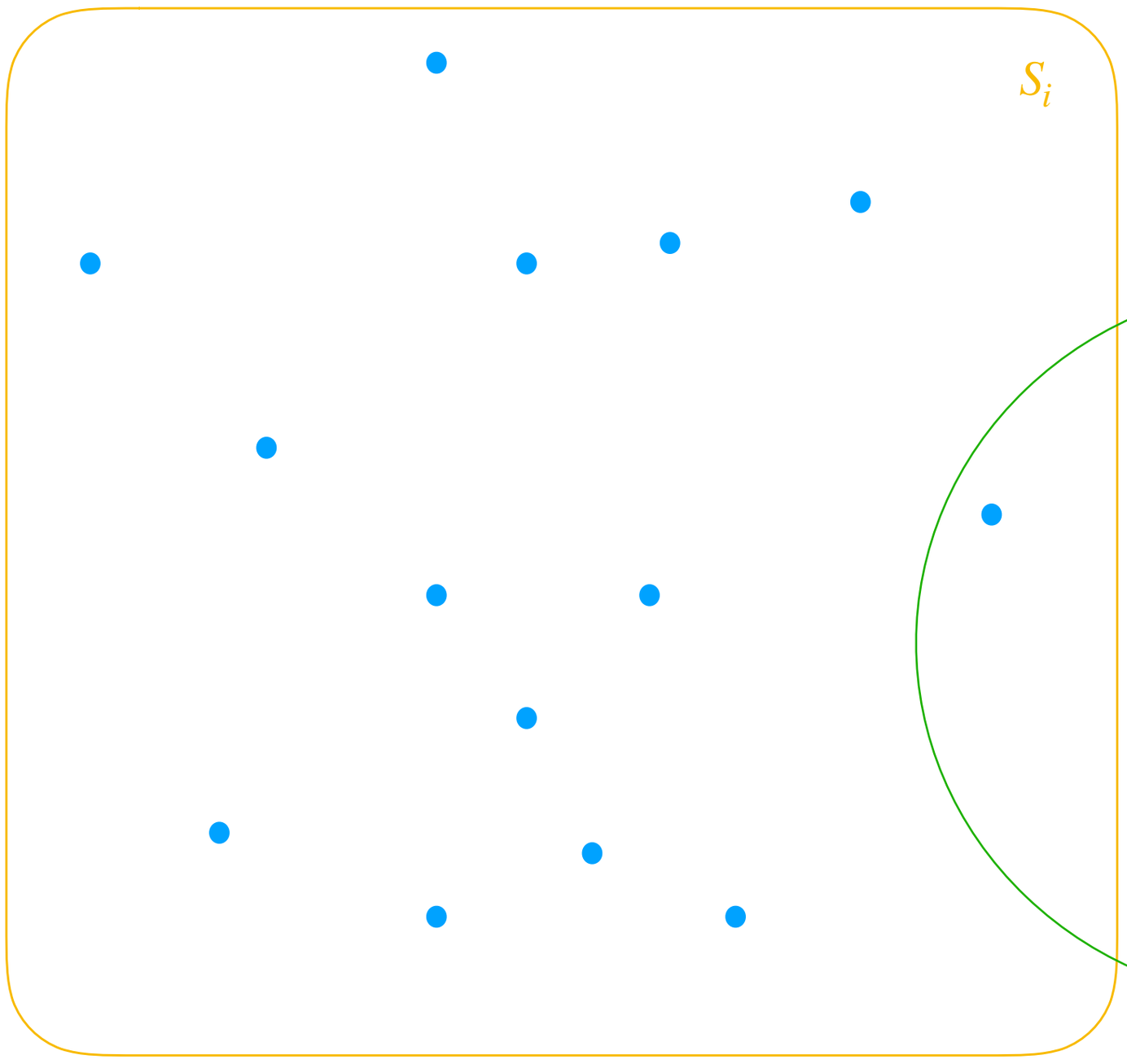
PTF



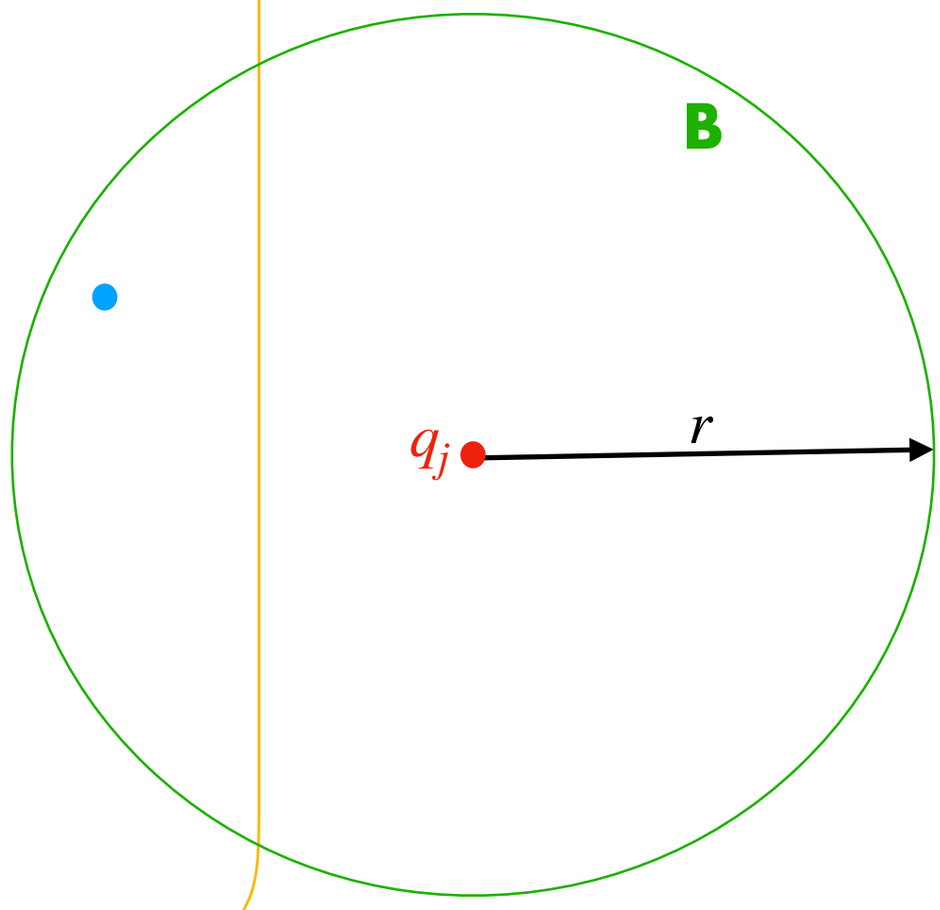
Imagine one entry



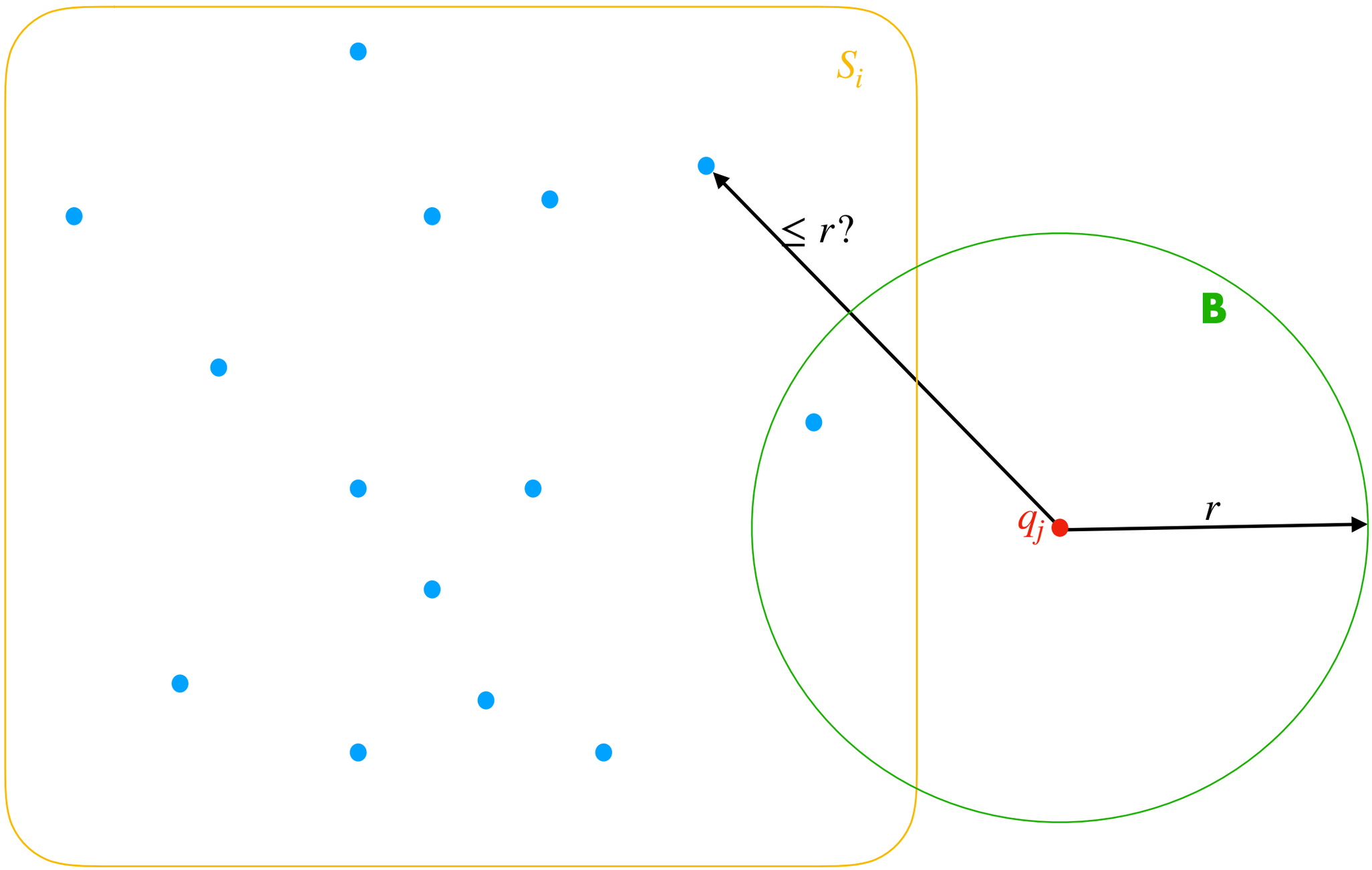
Naive search algorithm



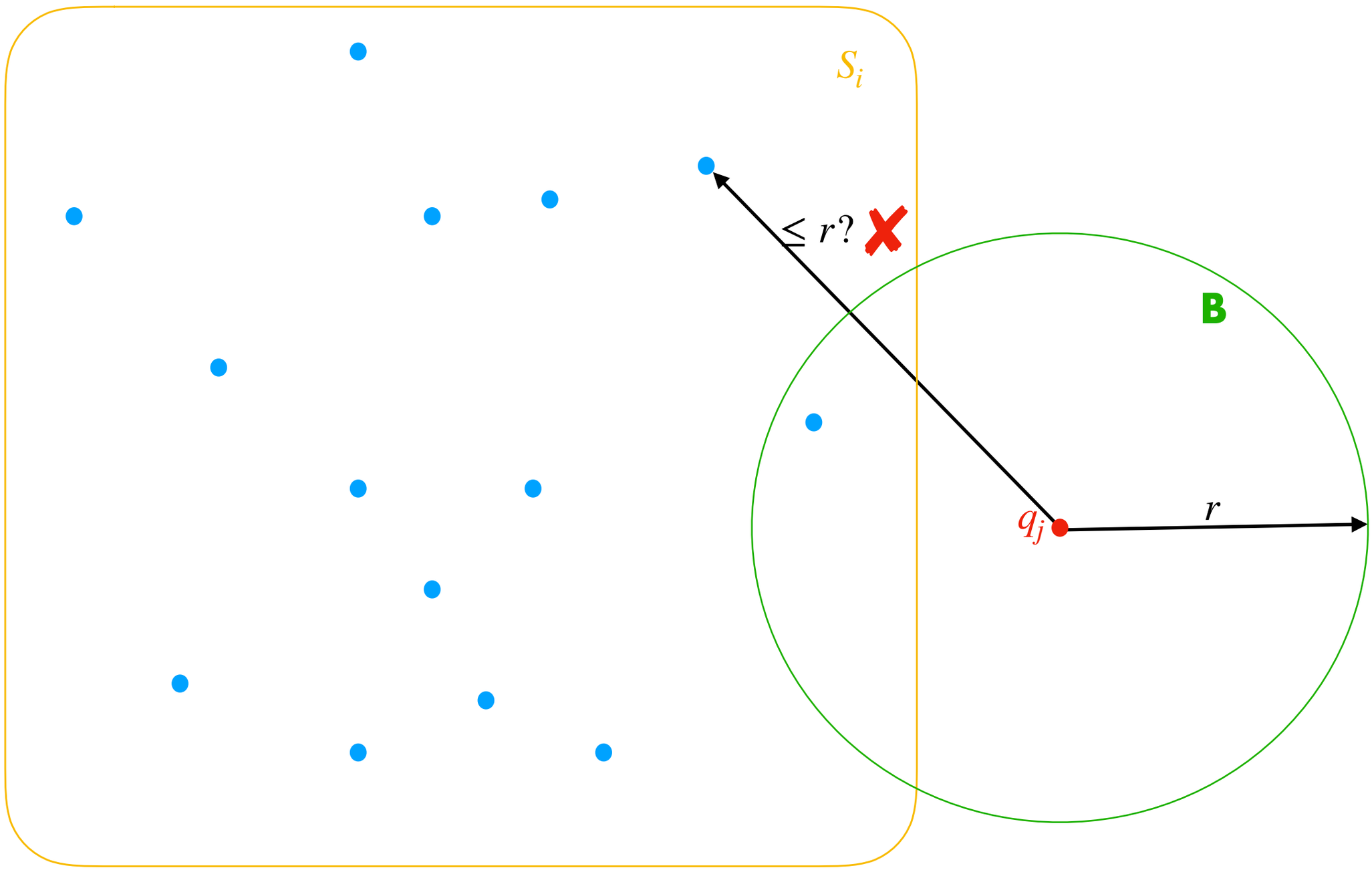
$$S_i \cap B = \emptyset ?$$



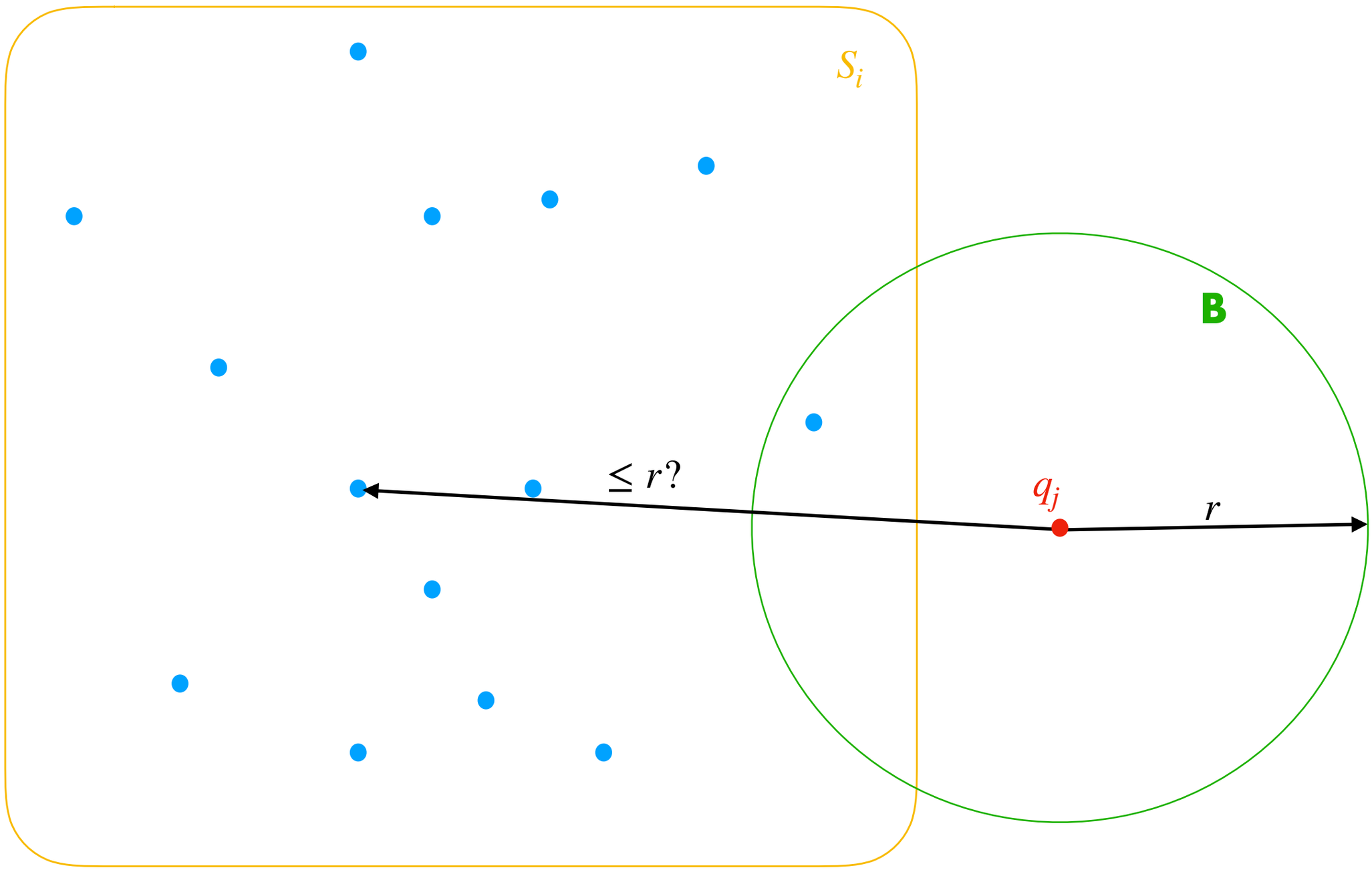
Naive search algorithm



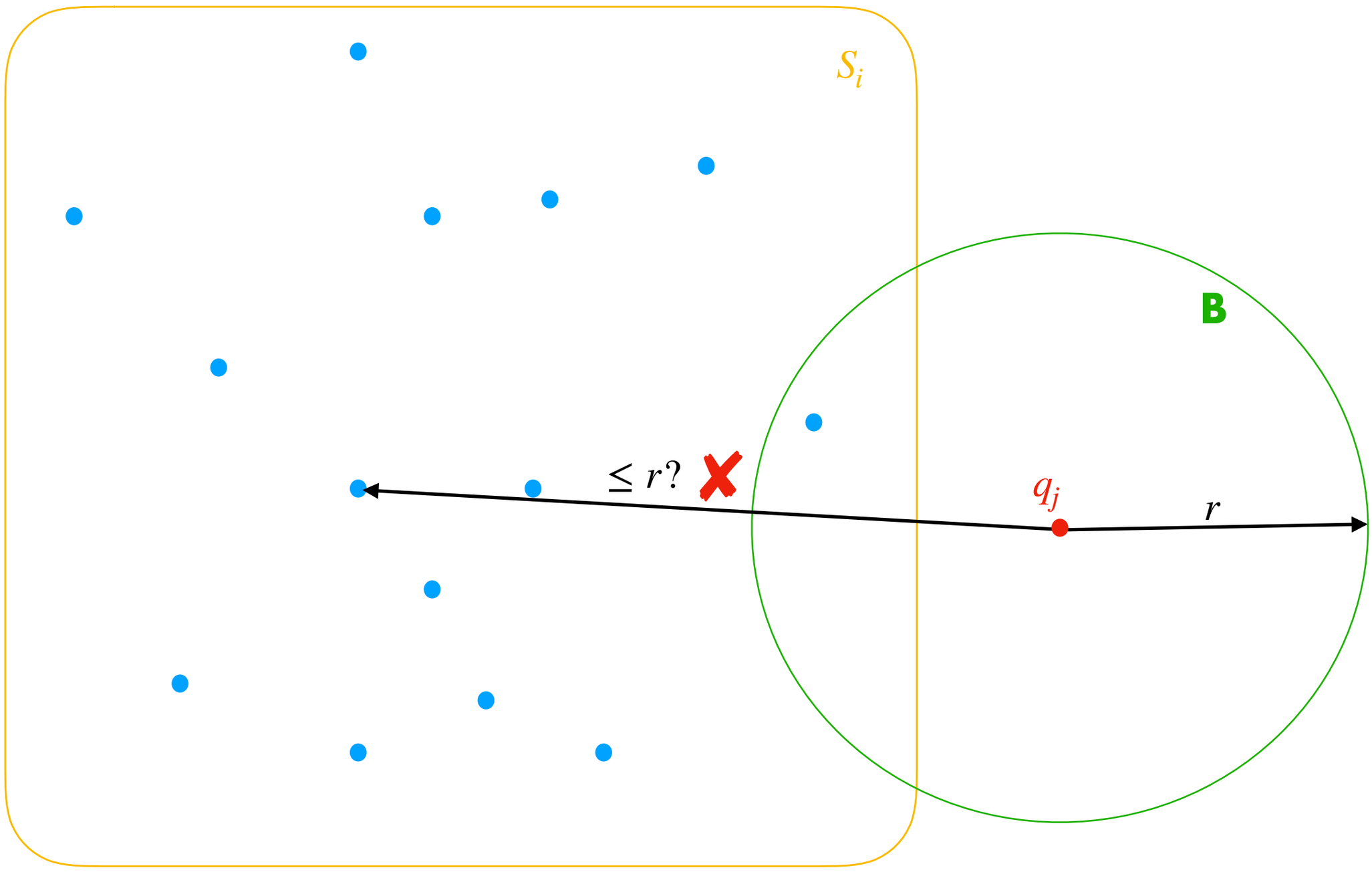
Naive search algorithm



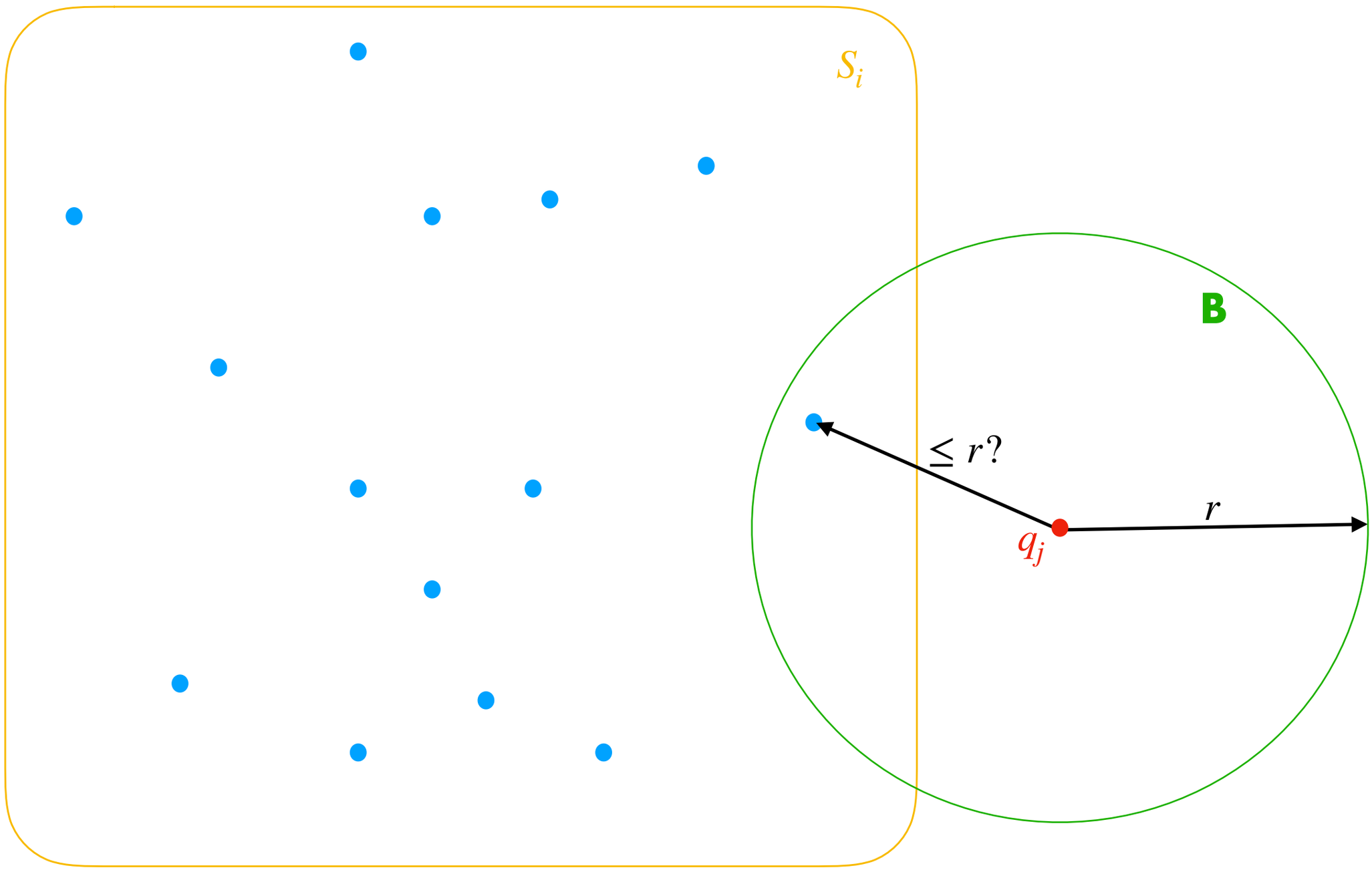
Naive search algorithm



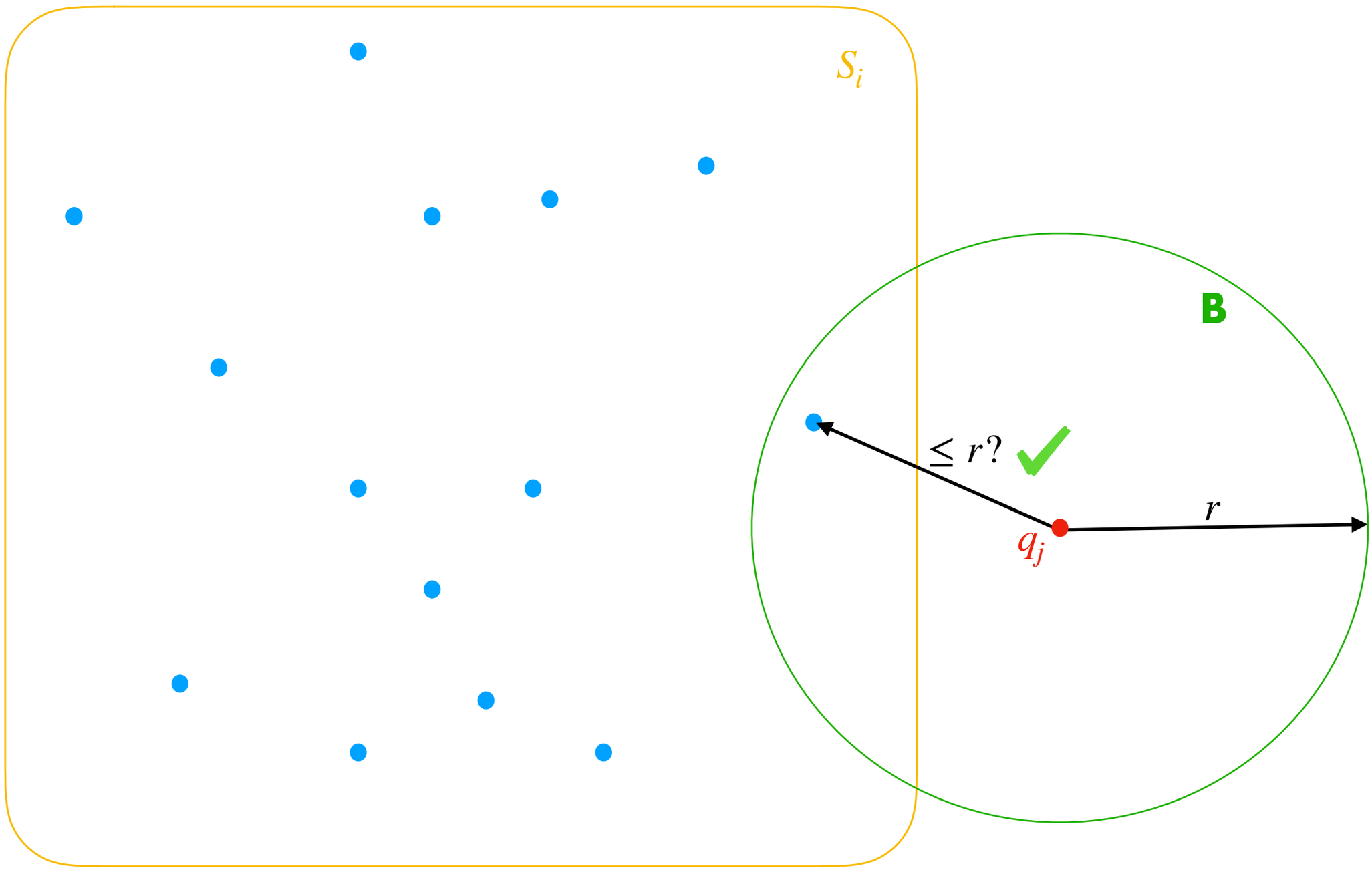
Naive search algorithm



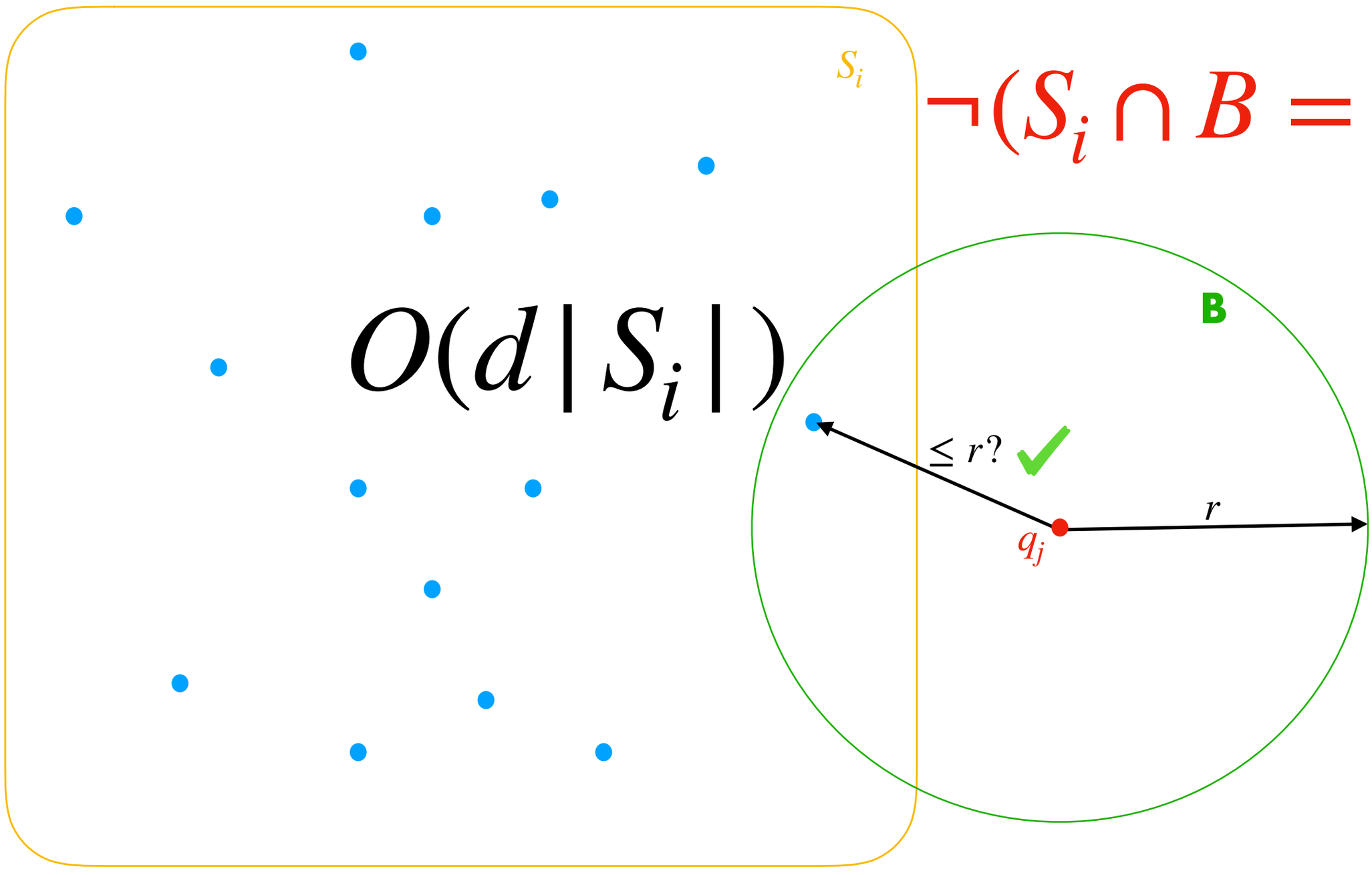
Naive search algorithm



Naive search algorithm

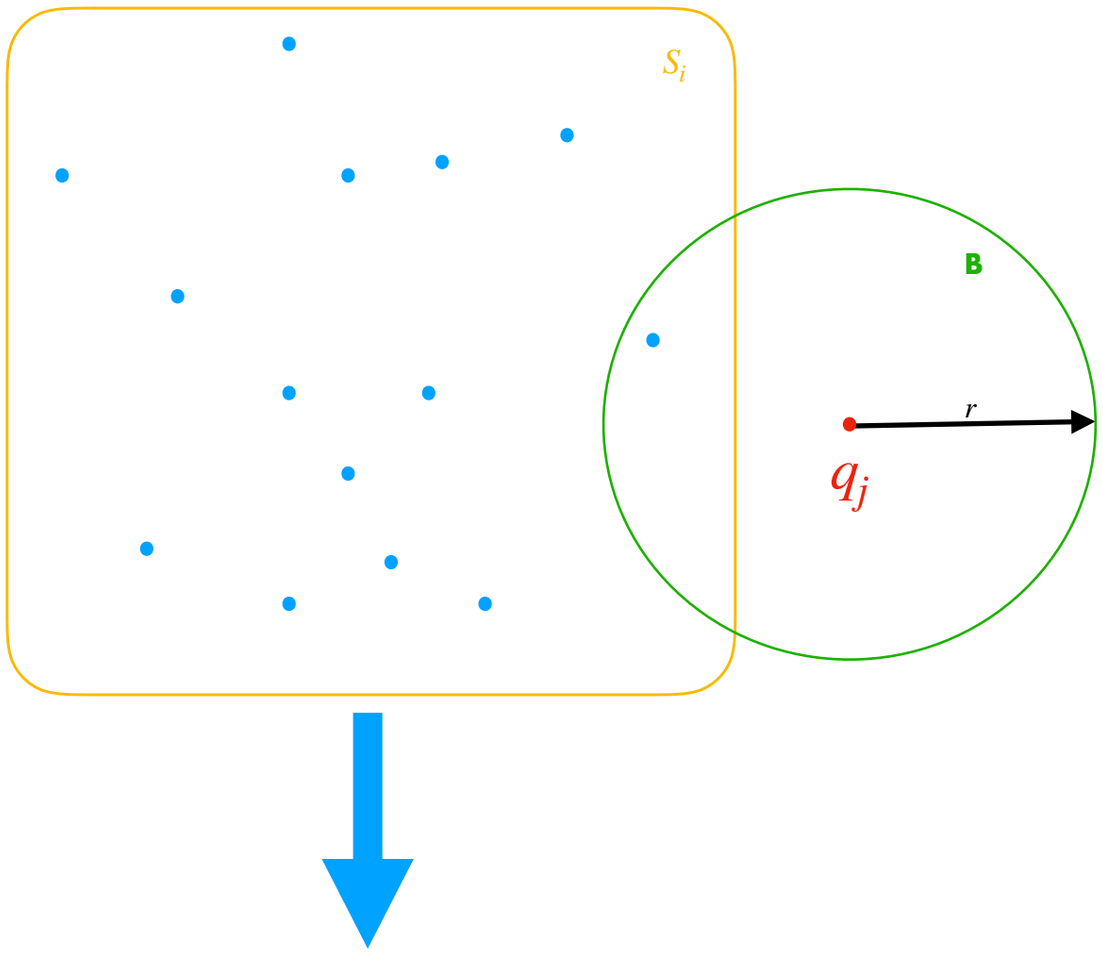


Naive search algorithm



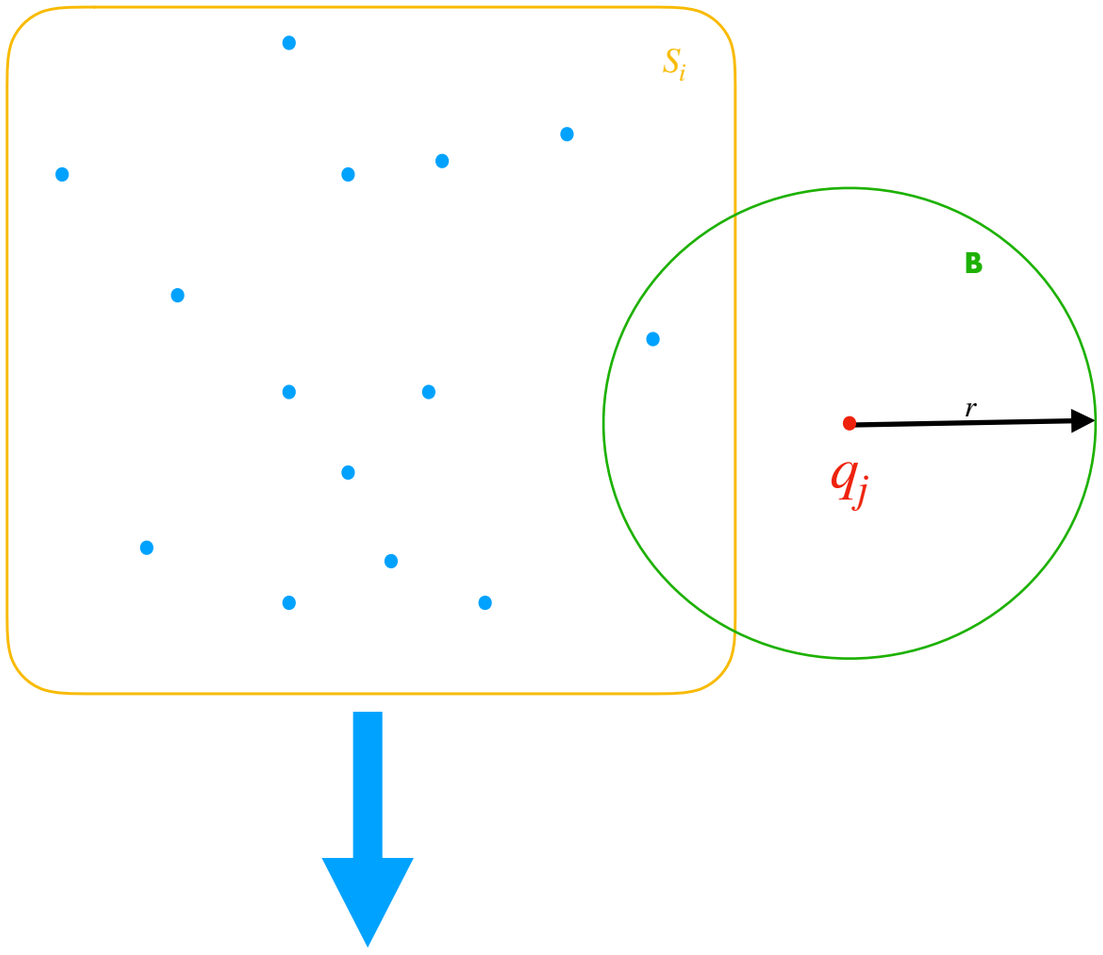
Building the matrix costs as much as a naive algorithm for building the r-net!

Probabilistic polynomial thresholding functions (PTF)

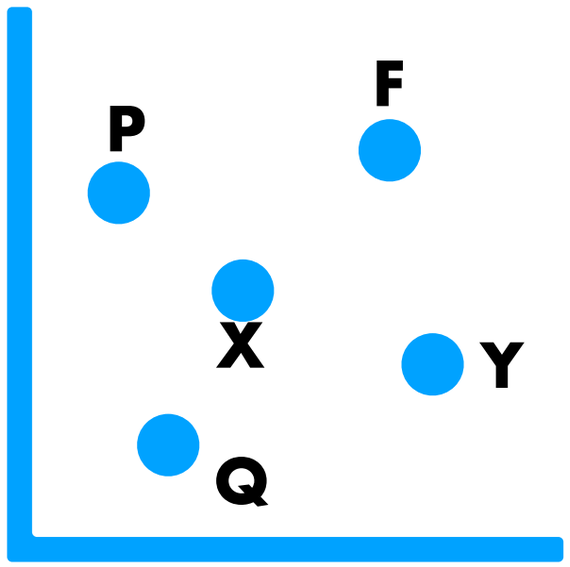


$$Z := \min_{p \in S_i}(dist(p, q_j)) \leq r$$

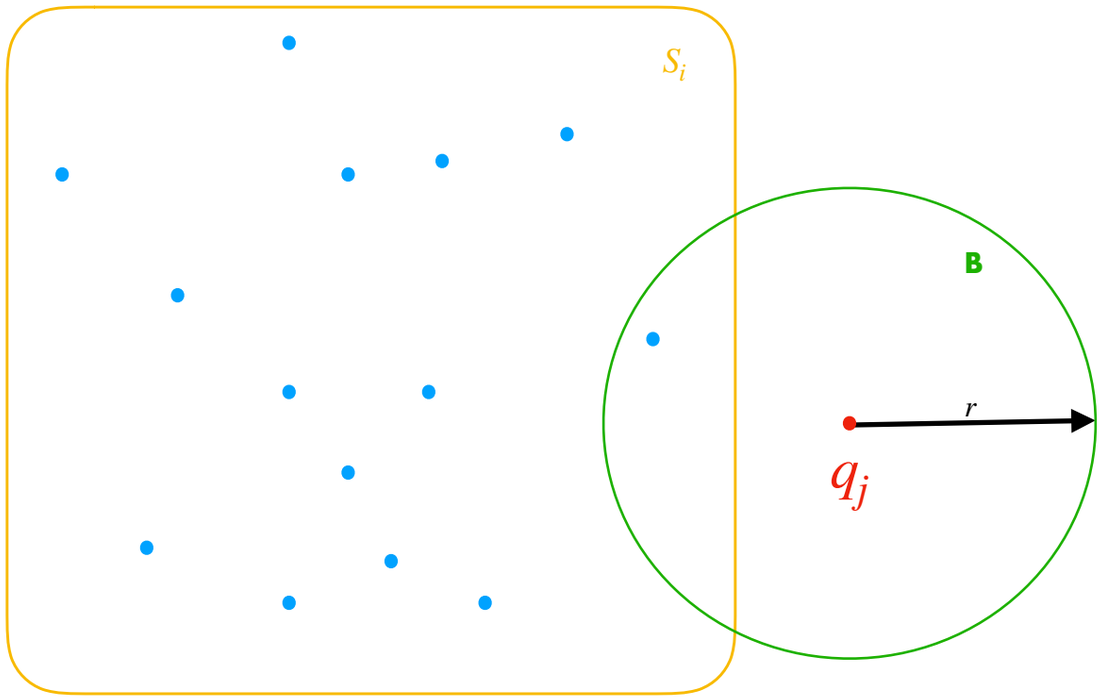
Probabilistic polynomial thresholding functions (PTF)



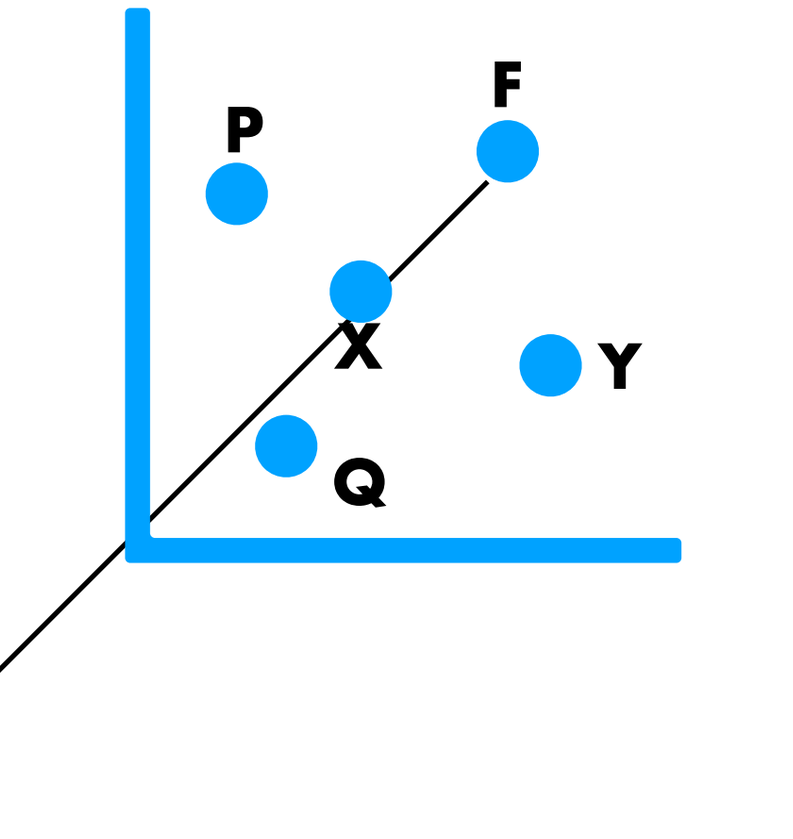
$$Z := \min_{p \in S_i}(dist(p, q_j)) \leq r$$



Probabilistic polynomial thresholding functions (PTF)



$$Z := \min_{p \in S_i}(dist(p, q_j)) \leq r$$



if $Z \leq r$ **then** $F(p_1, p_2, \ldots, p_n, q_j) \geq 2|S_i|$ **with some probability**

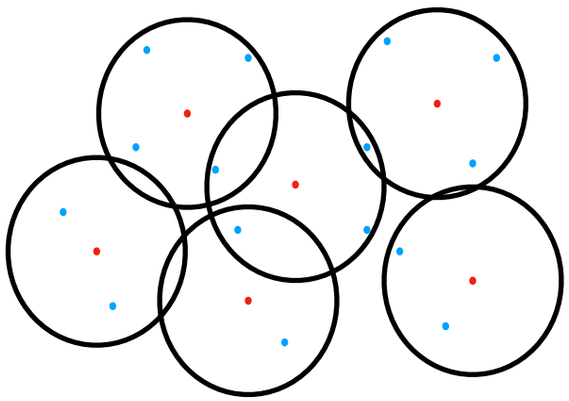
if $Z > r + \epsilon d$ **then** $F(p_1, p_2, \ldots, p_n, q_j) \leq |S_i|$ **with some probability**

Leads to improvement of runtime for building the distance matrix

	p_1	p_2	...	p_n
	yes	yes	...	yes
	yes	yes	...	yes
	yes	yes	...	yes

What now?

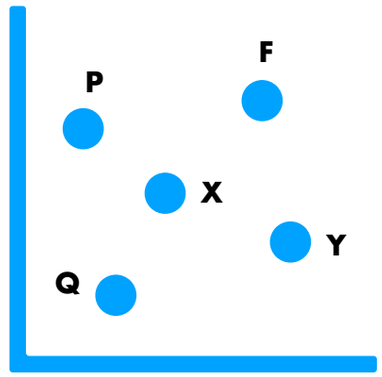
R-nets



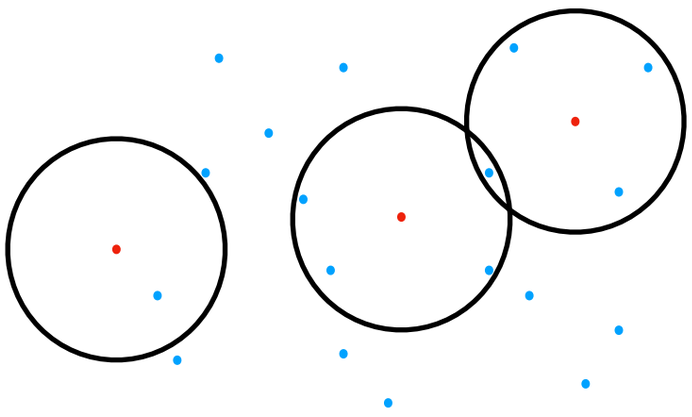
Distance matrix

	p_1	p_2		p_n
	yes	no	...	yes
	no	no	...	yes
	no	yes	...	no

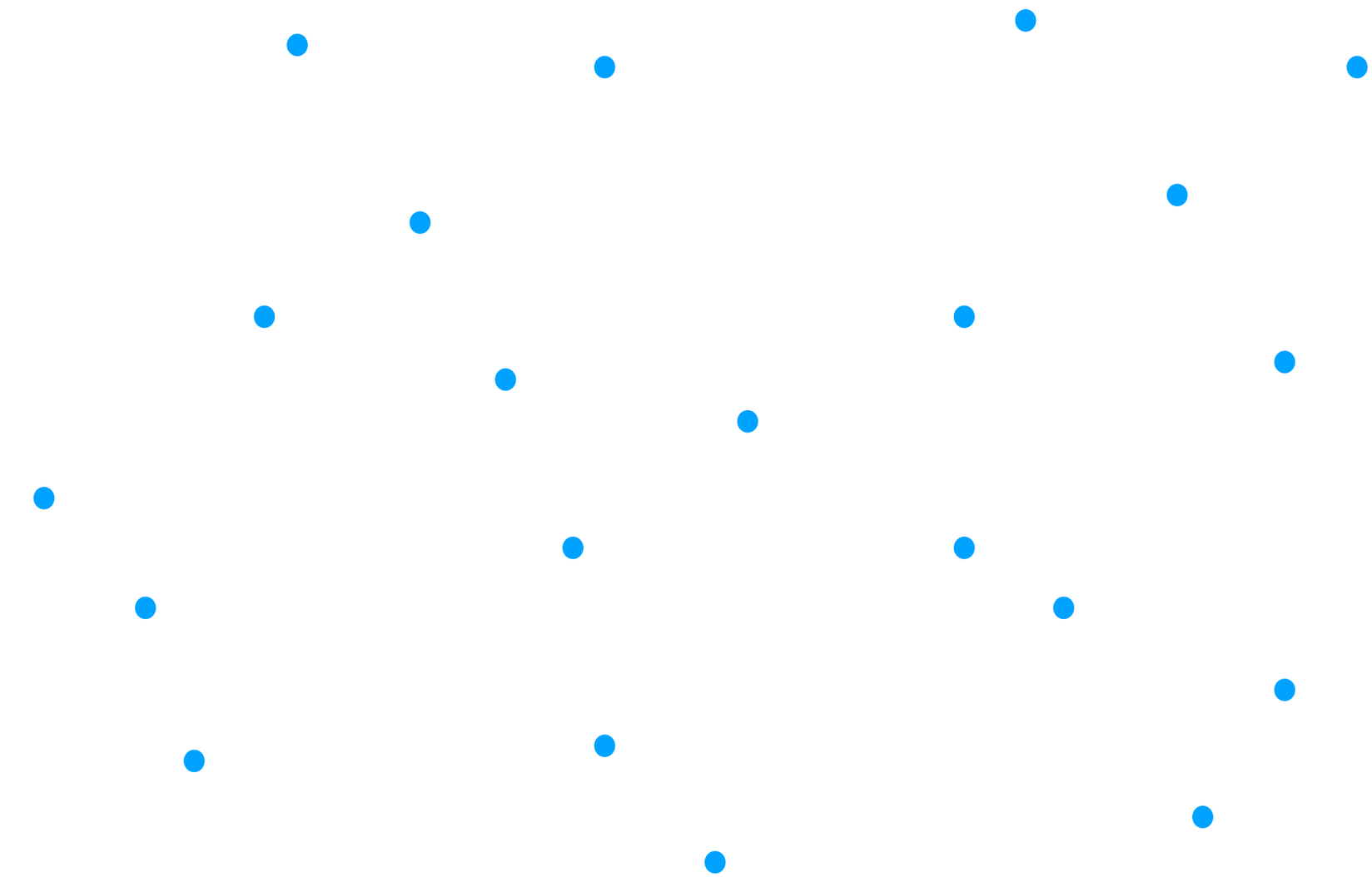
PTF



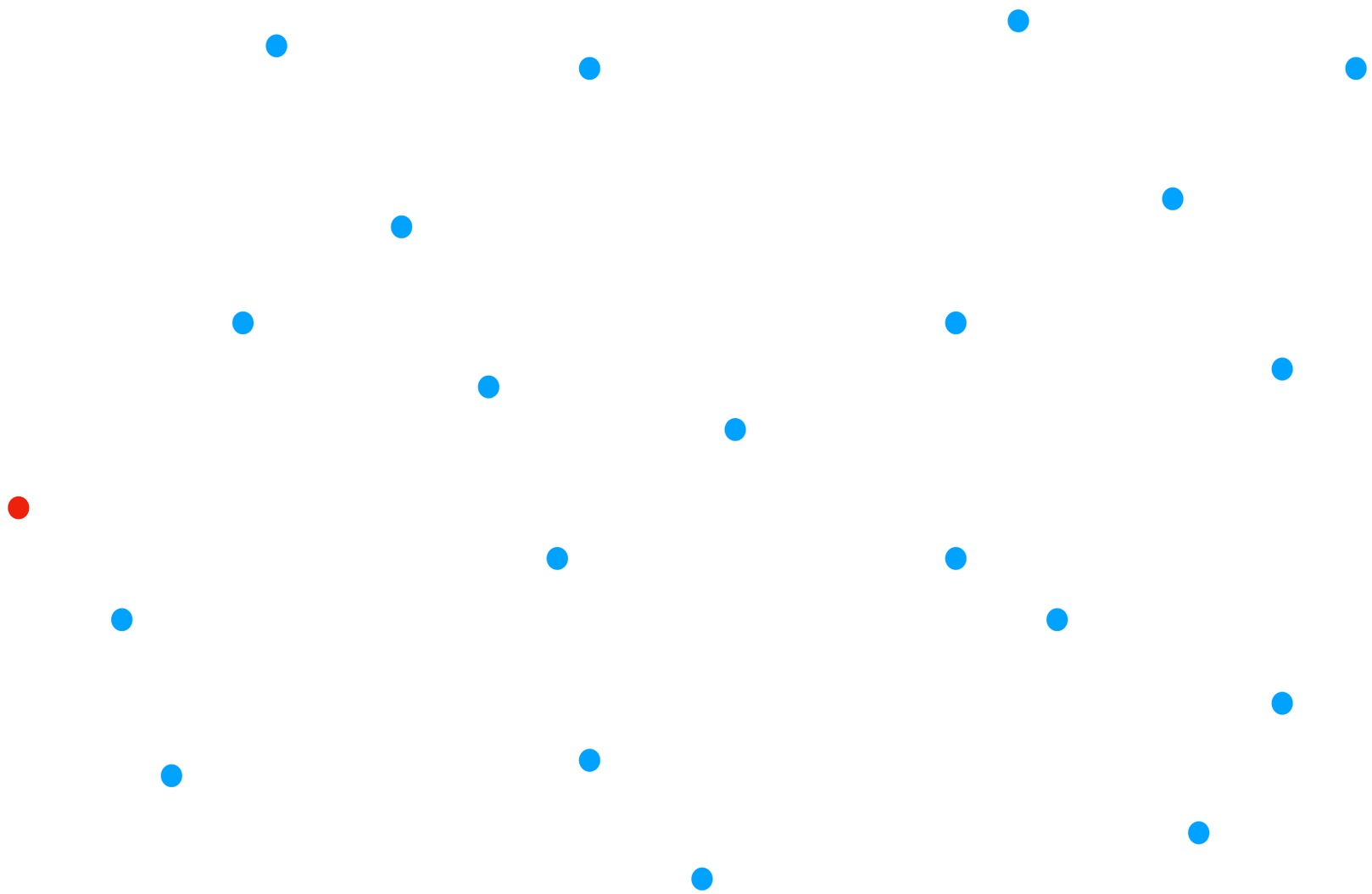
Sparsification



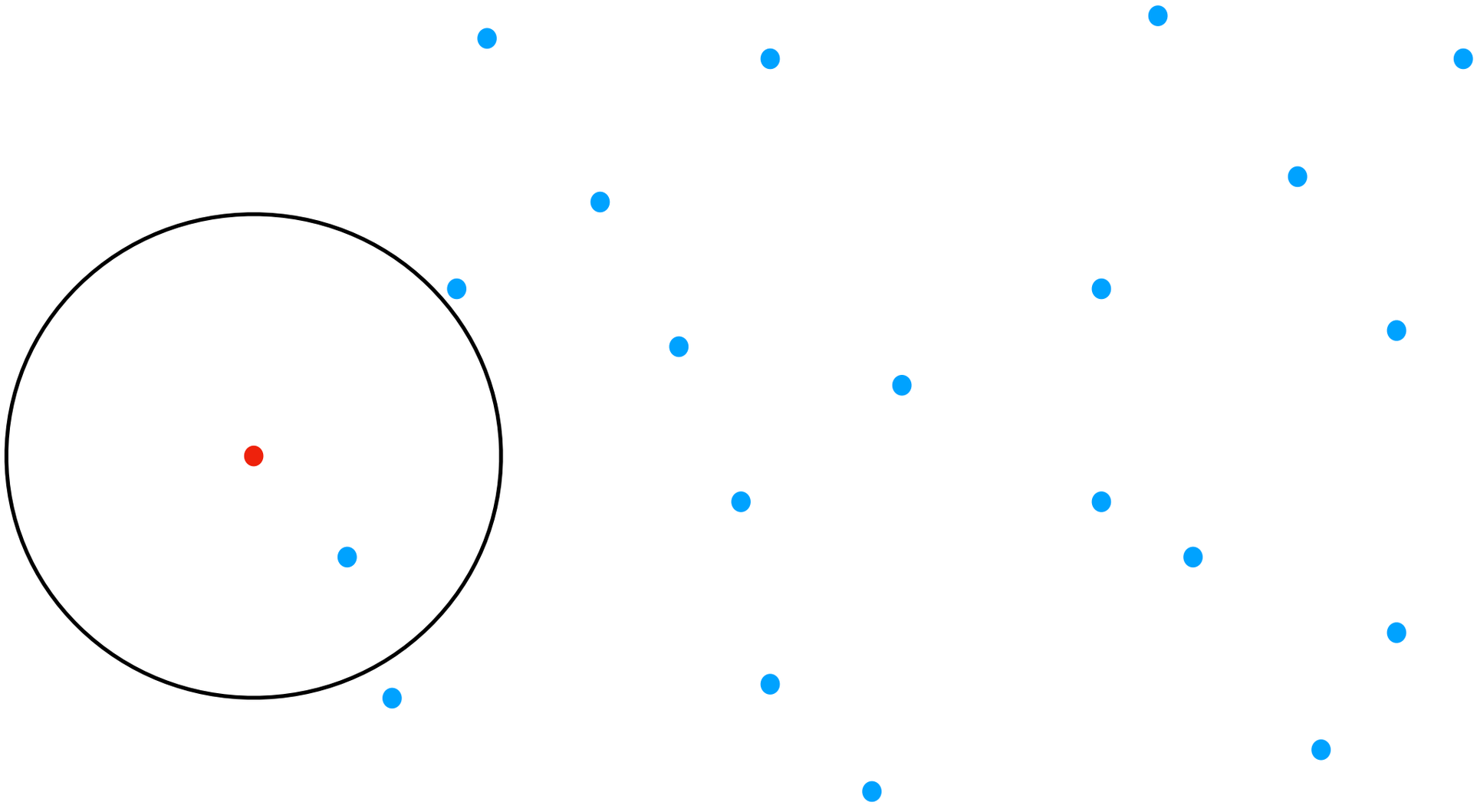
Sparsify



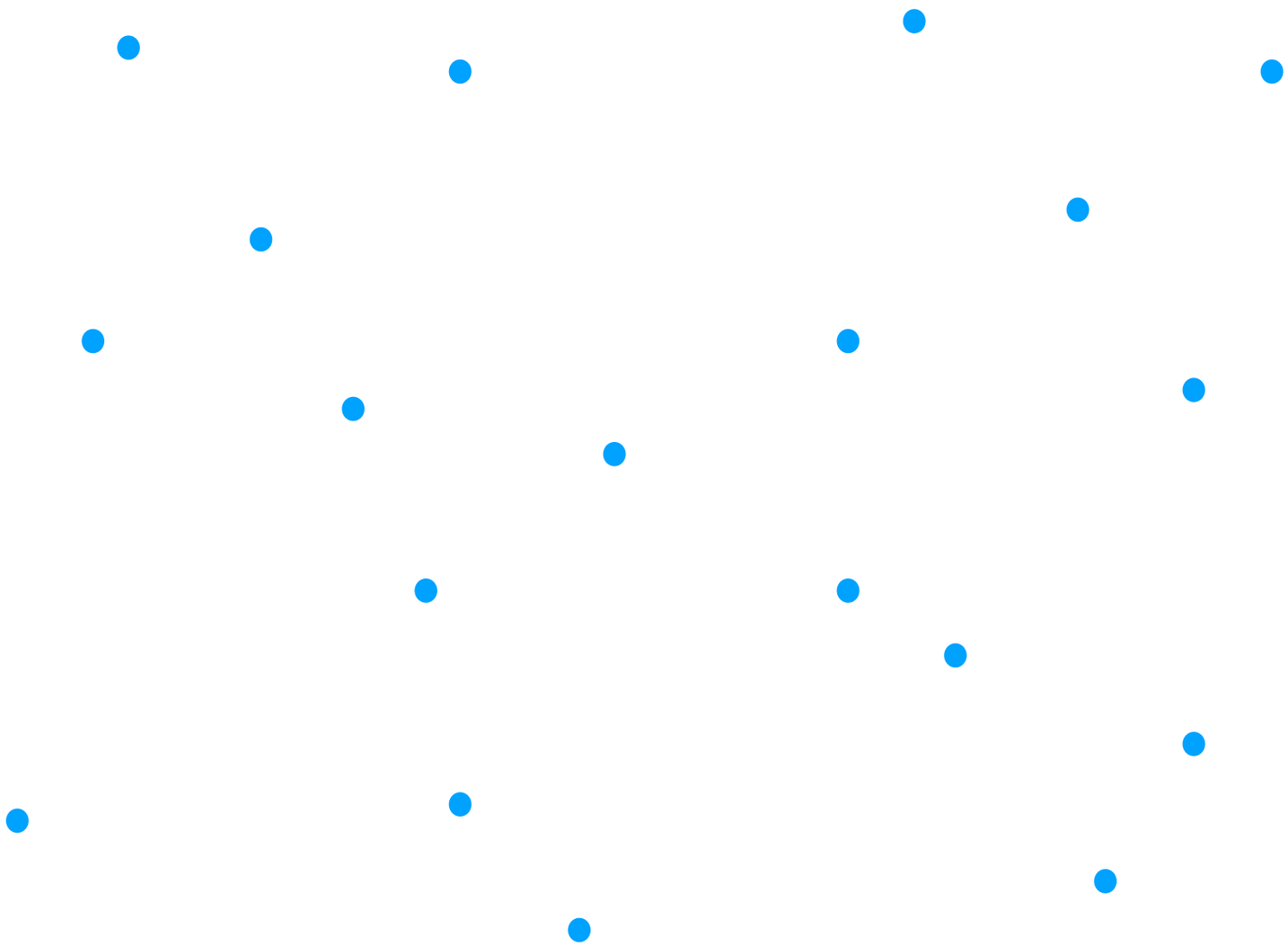
Sparsify



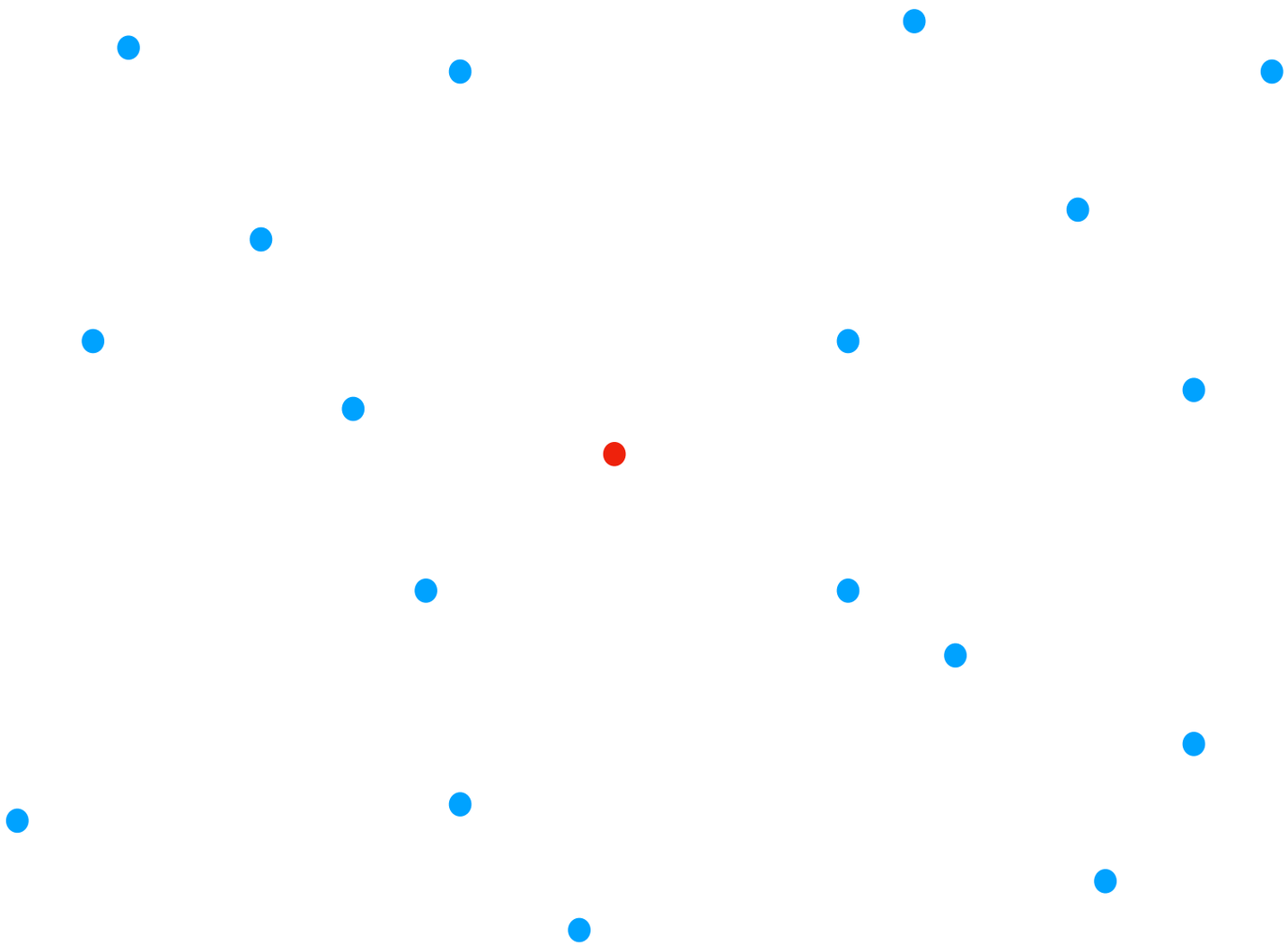
Sparsify



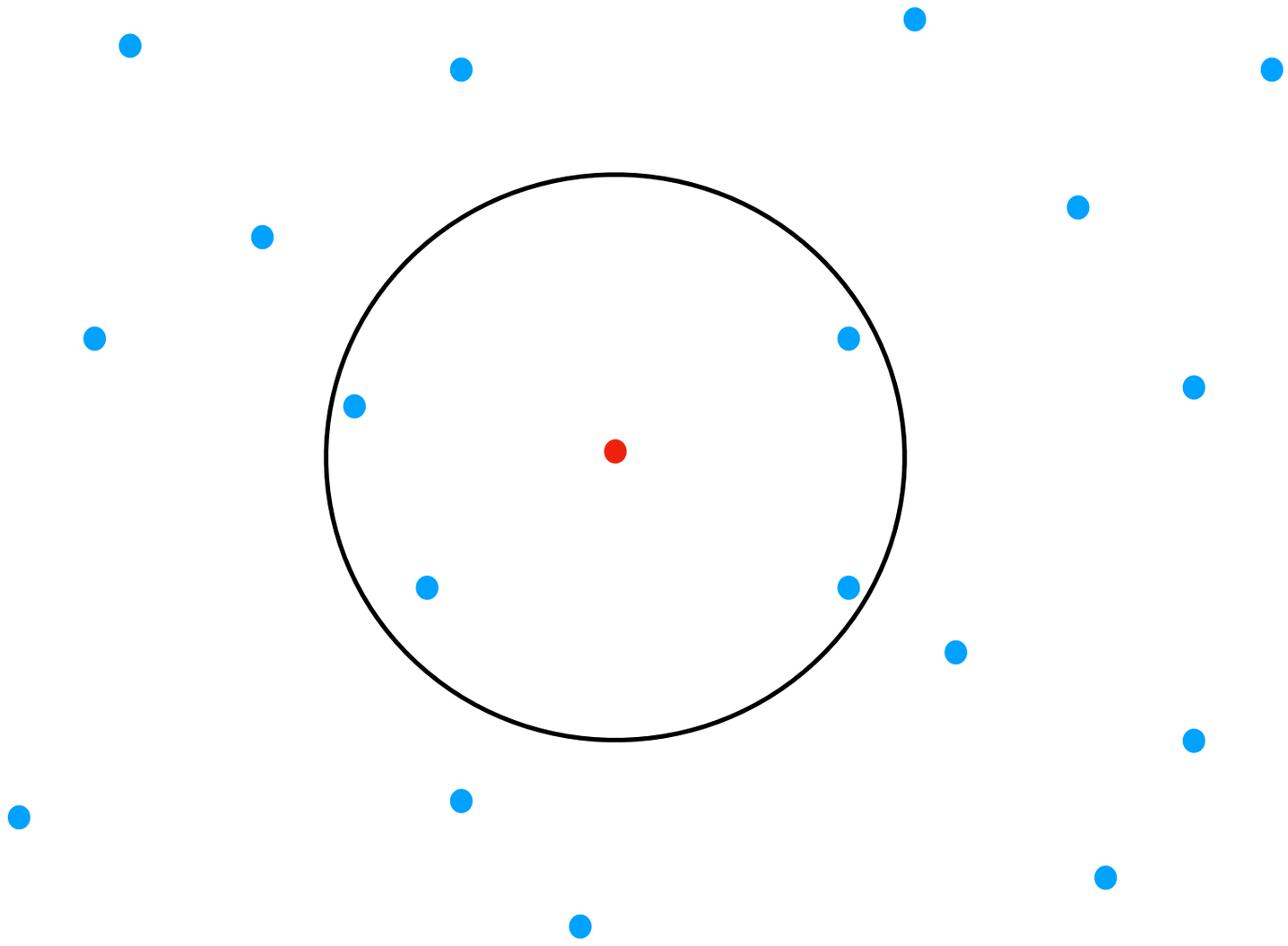
Sparsify



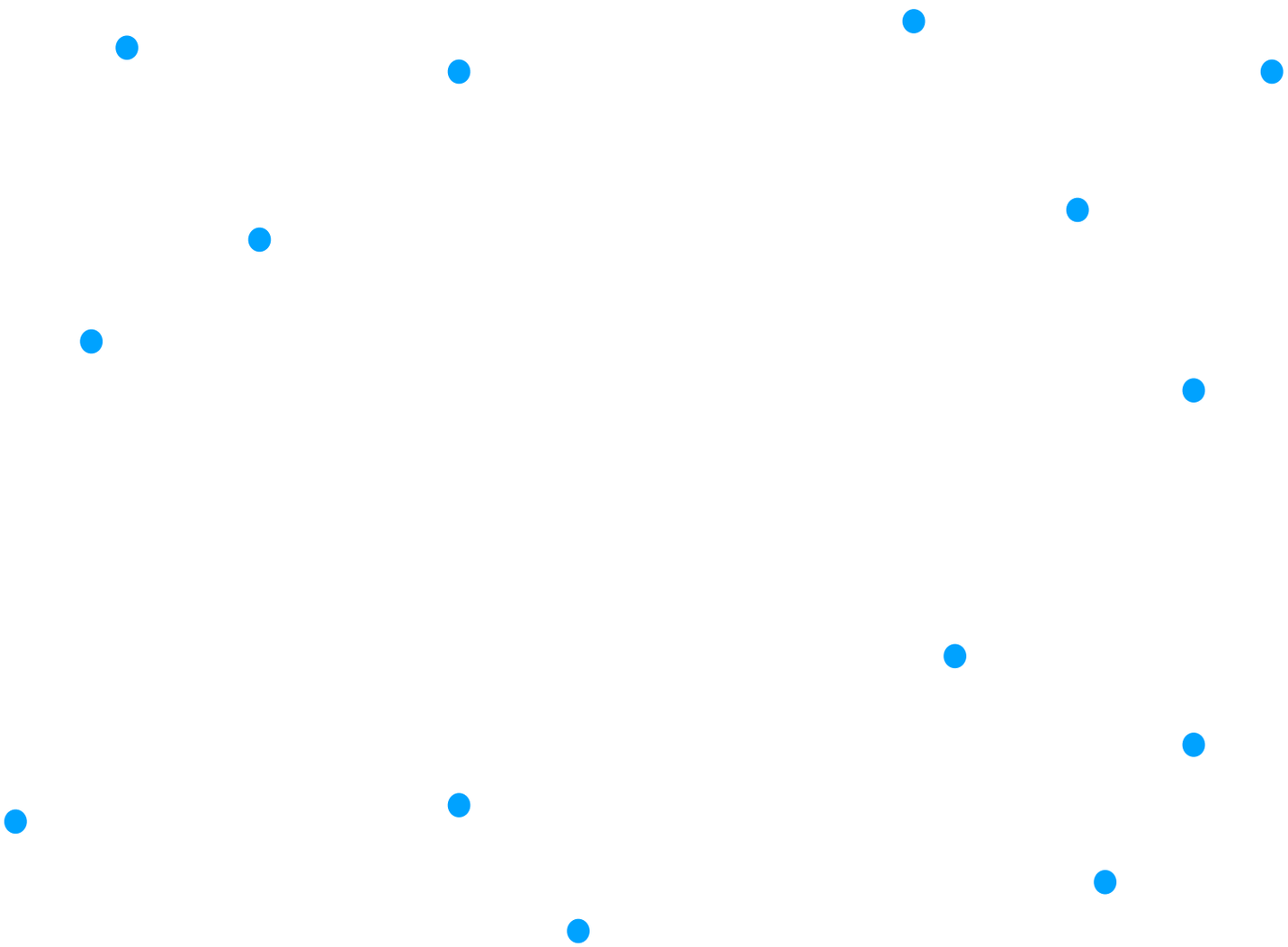
Sparsify



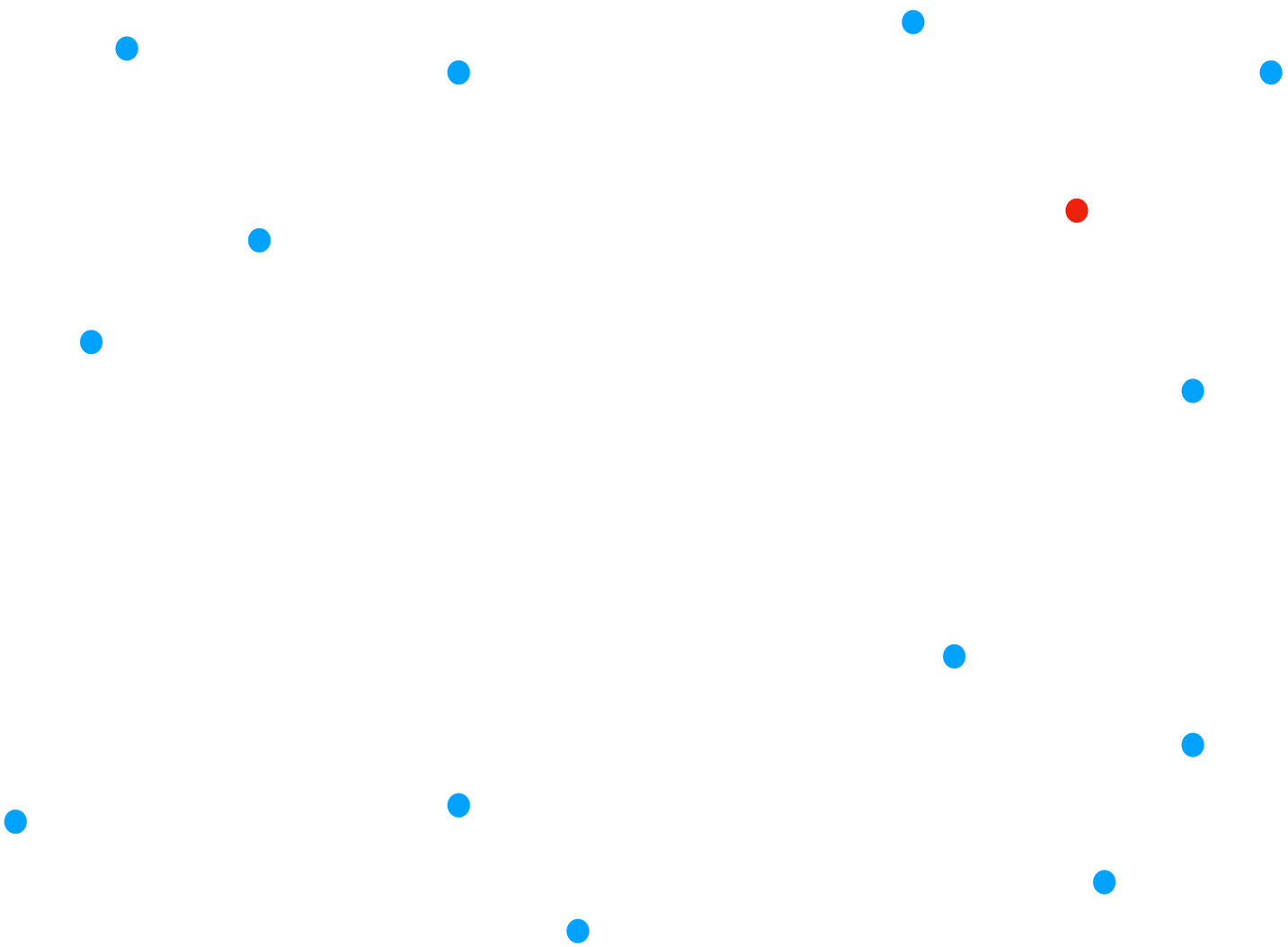
Sparsify



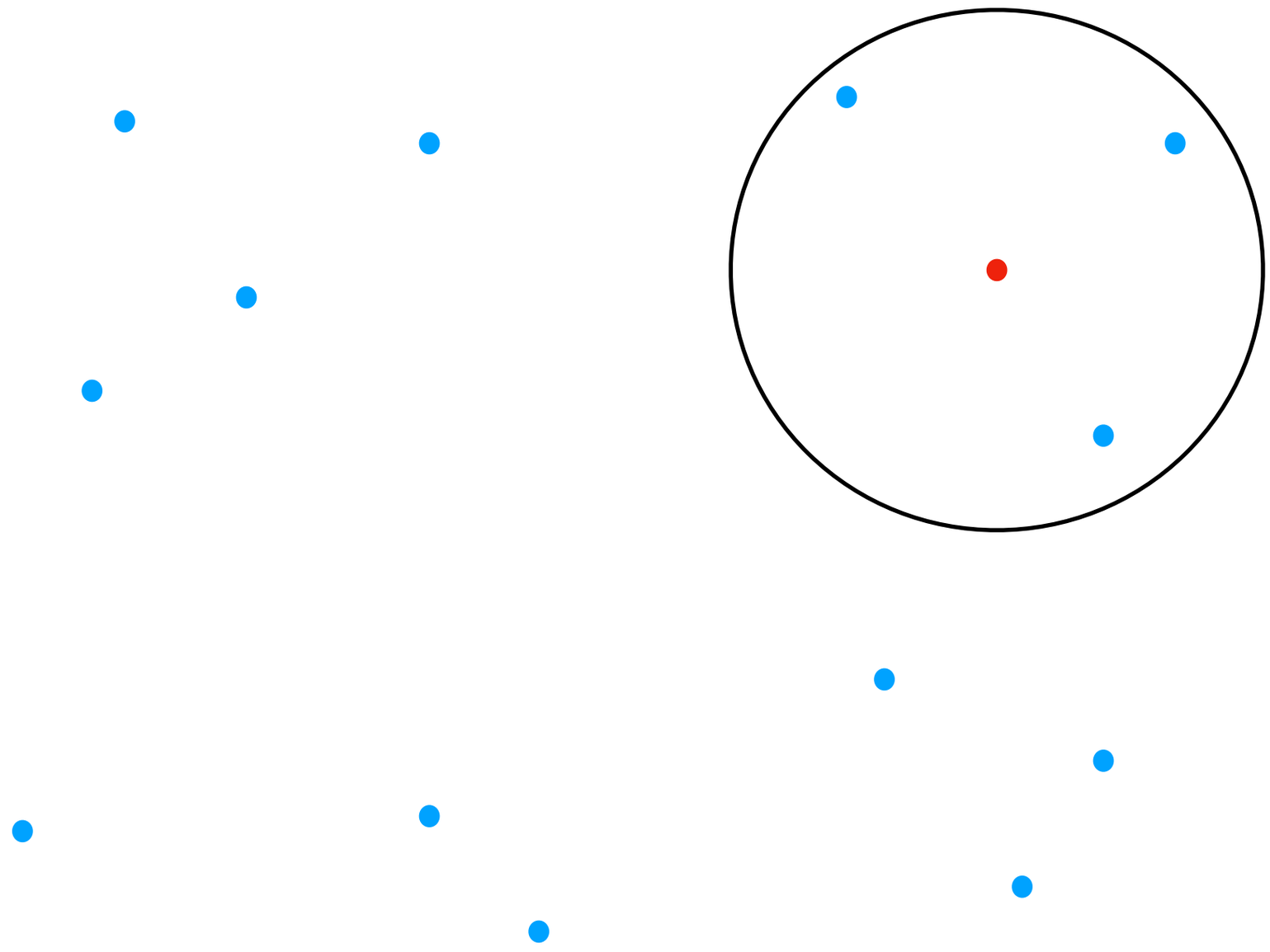
Sparsify



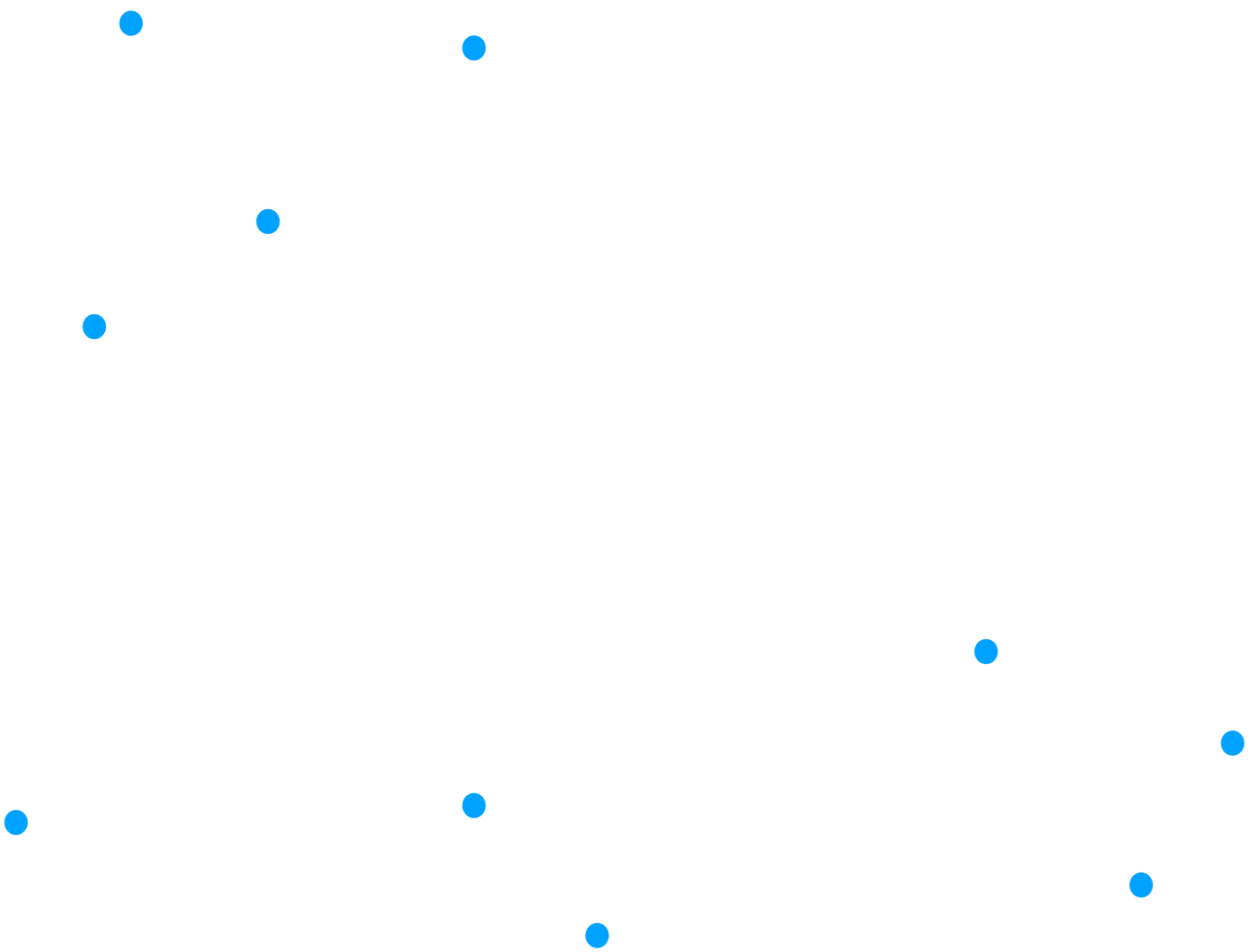
Sparsify



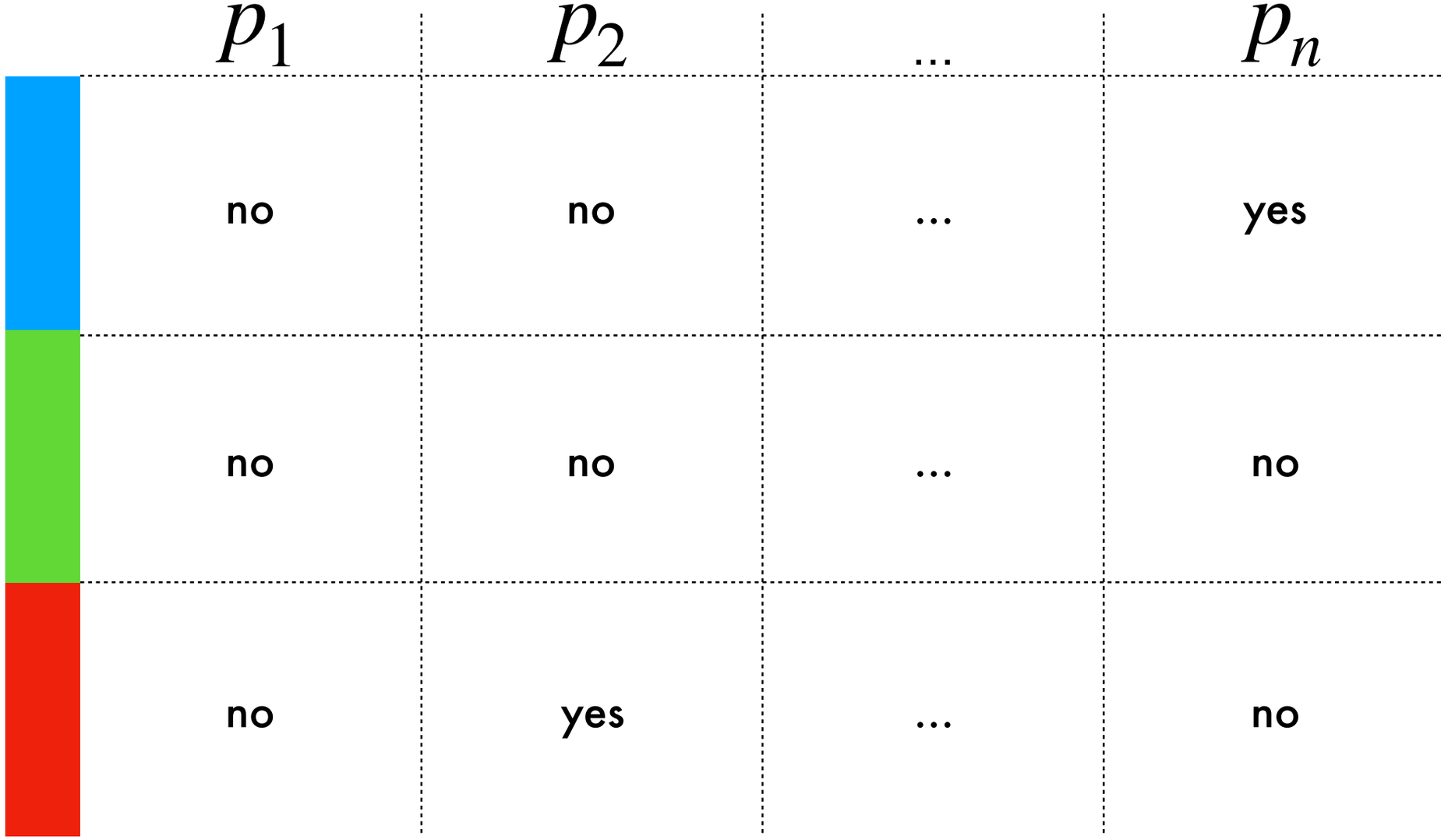
Sparsify



Sparsify

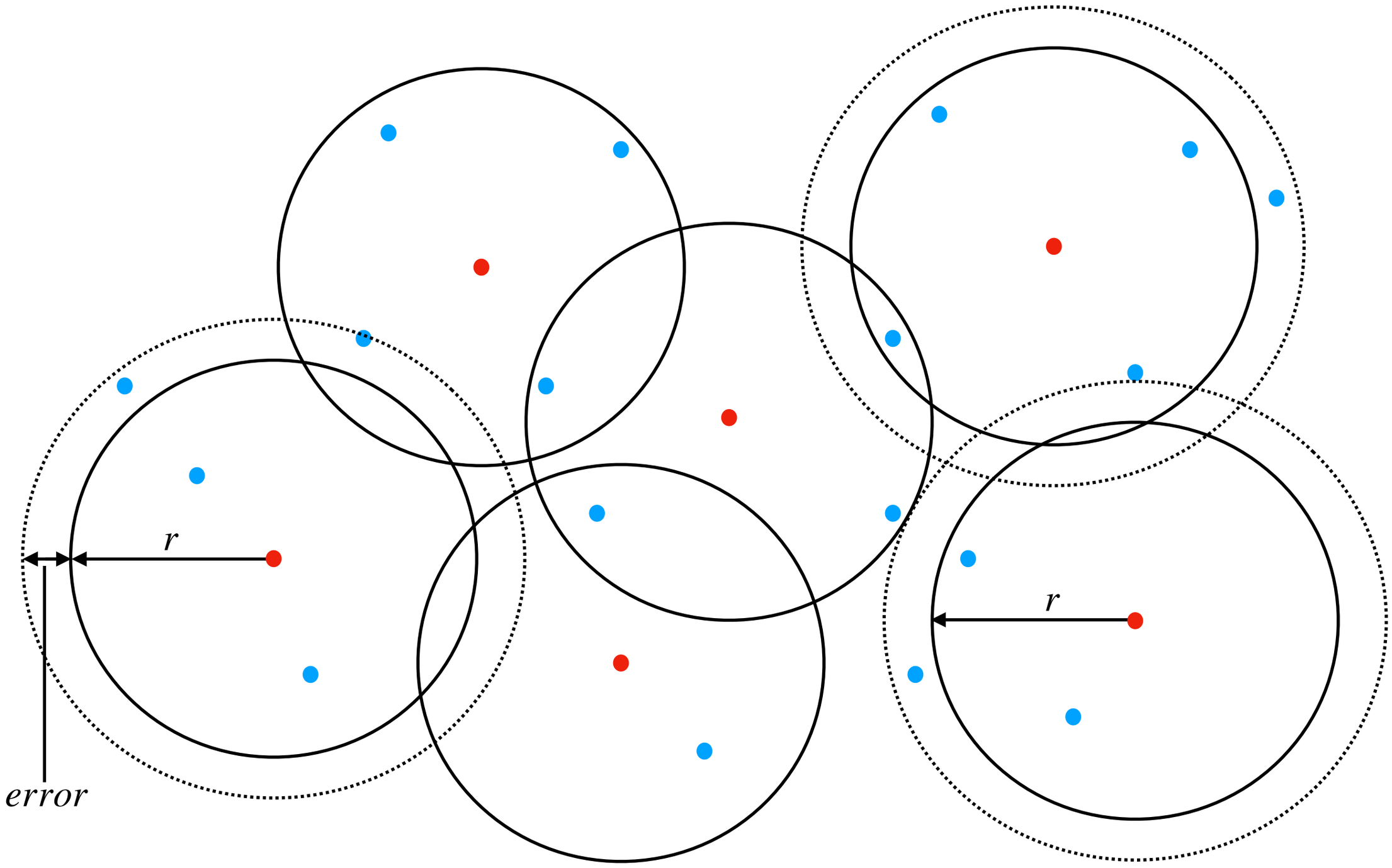


Sparsity with high probability

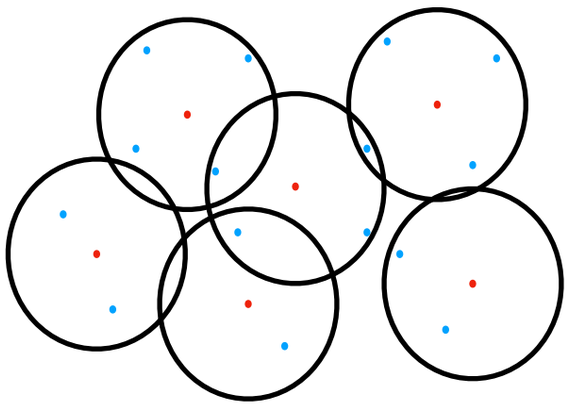


	p_1	p_2	...	p_n
	no	no	...	yes
	no	no	...	no
	no	yes	...	no

Approximate r-net



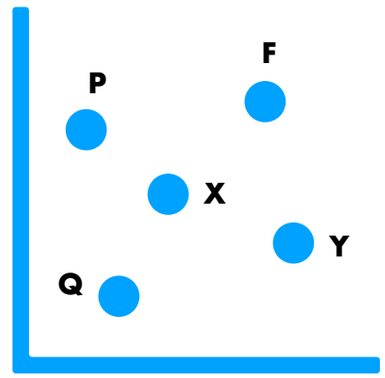
R-nets



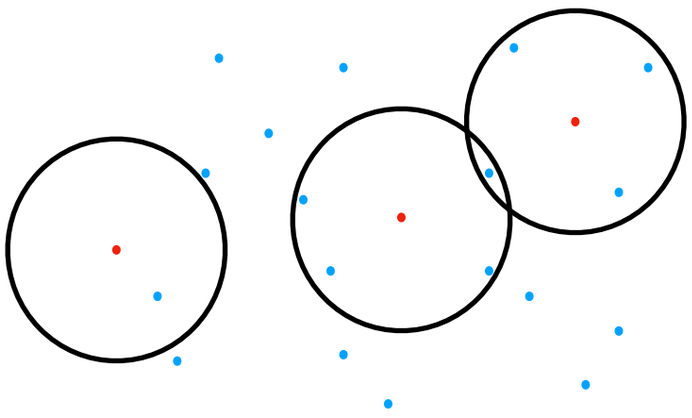
Distance matrix

	p_1	p_2		p_n
	yes	no	...	yes
	no	no	...	yes
	no	yes	...	no

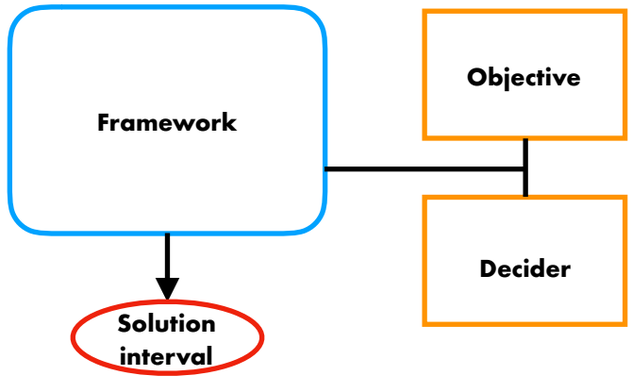
PTF



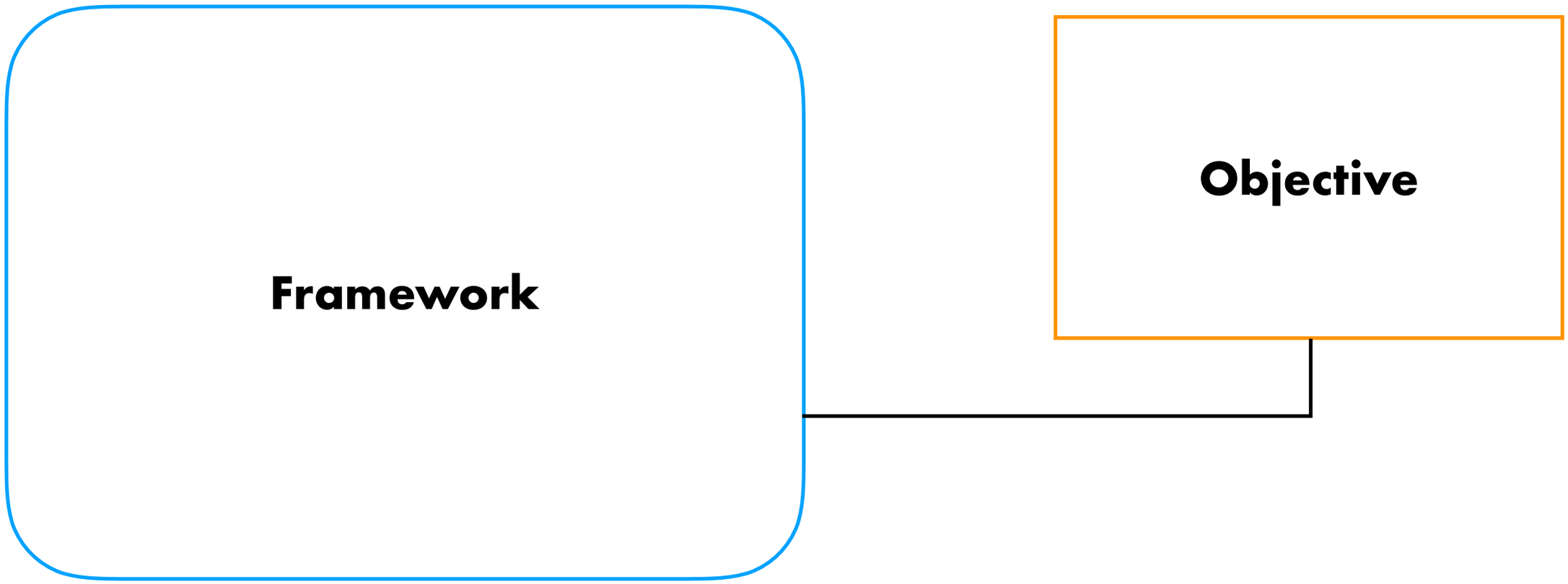
Sparsification



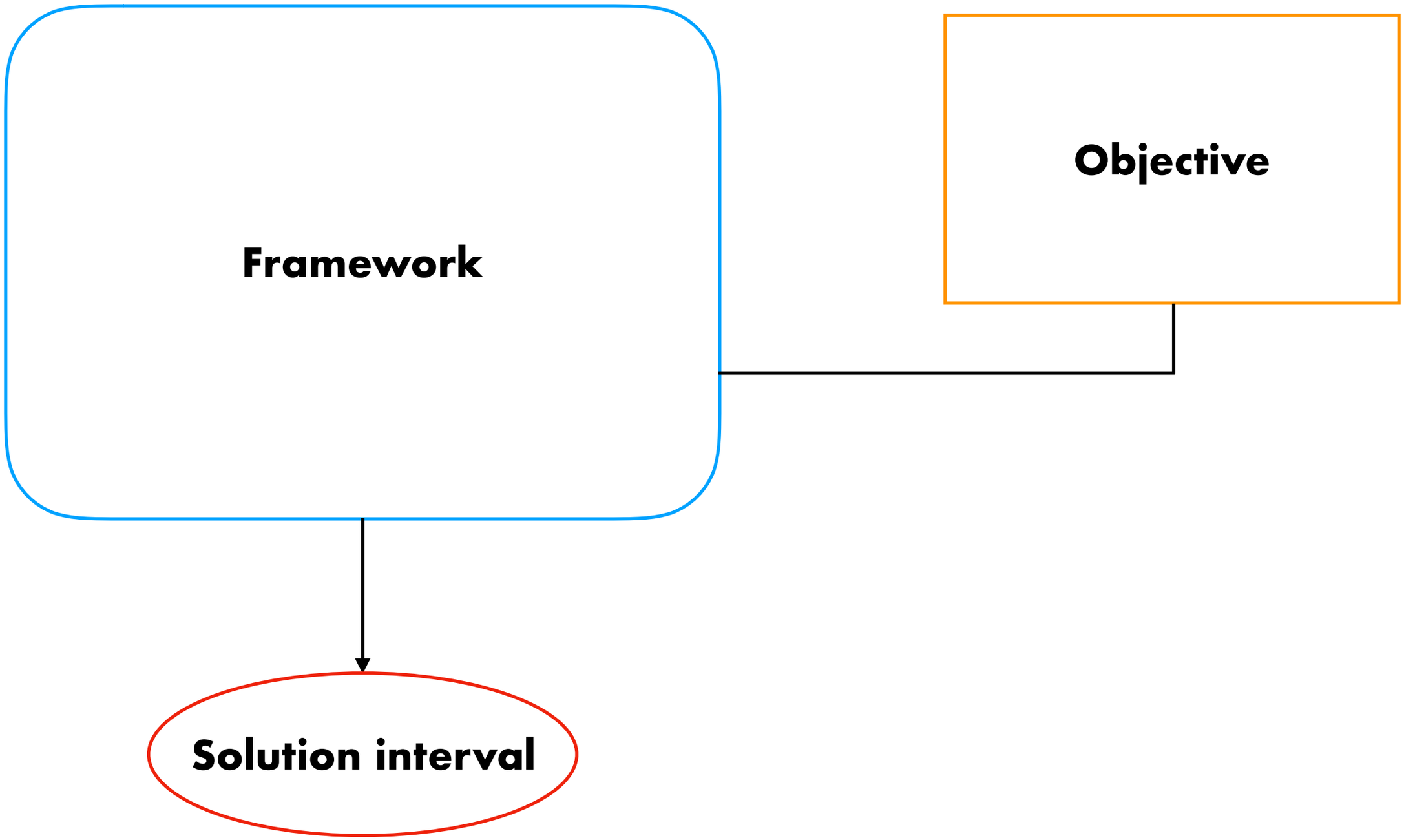
Net & Prune



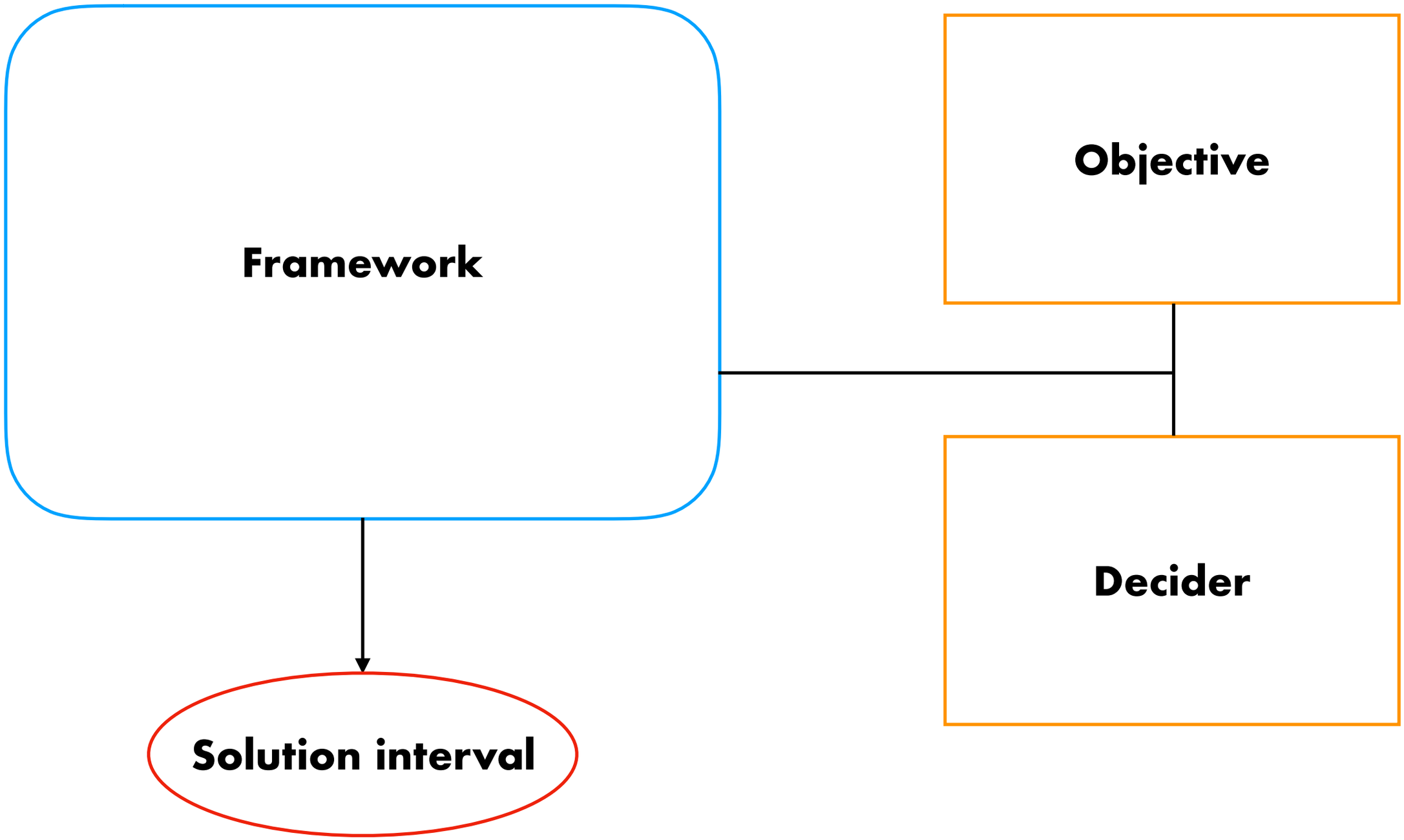
Net & Prune framework



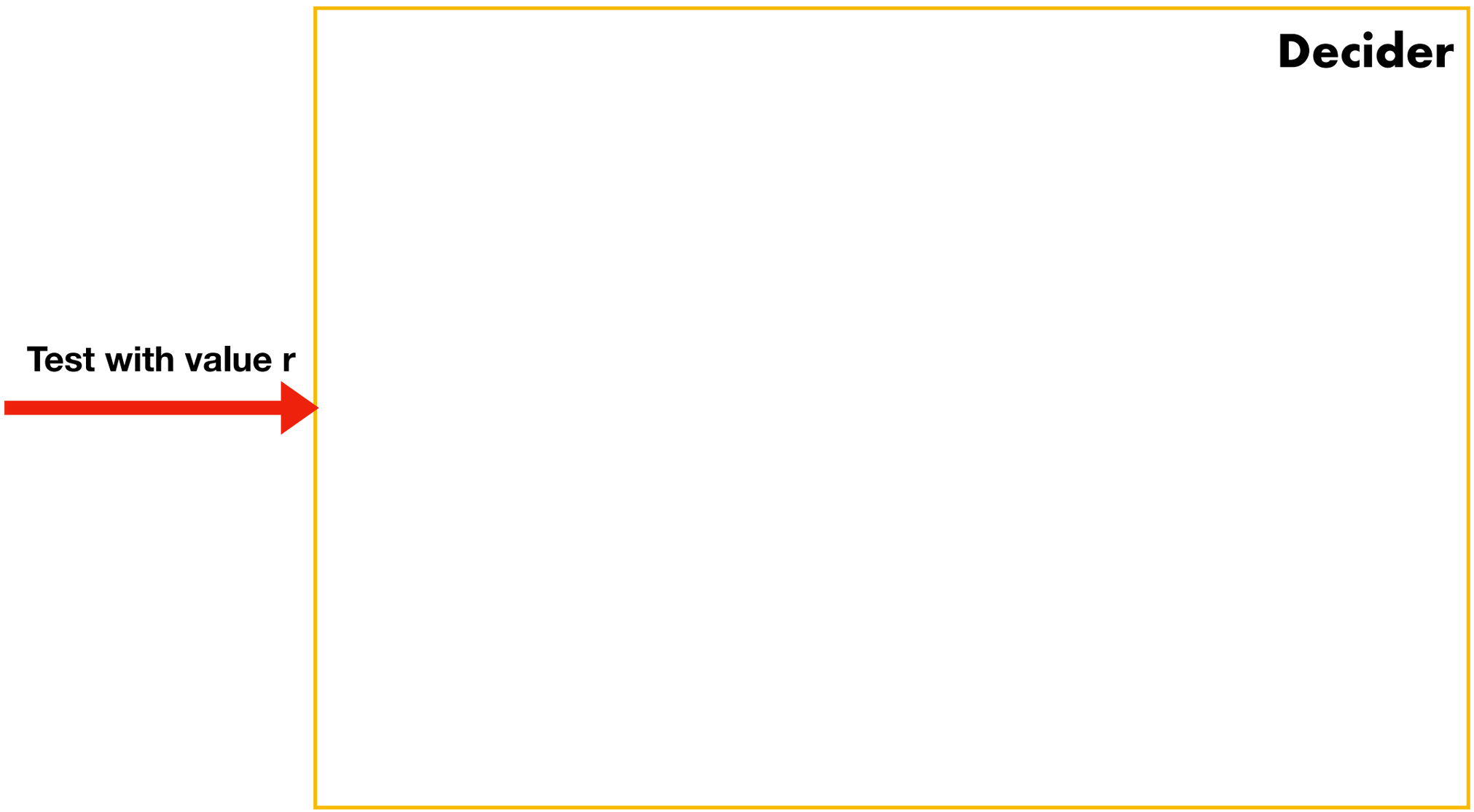
Net & Prune framework



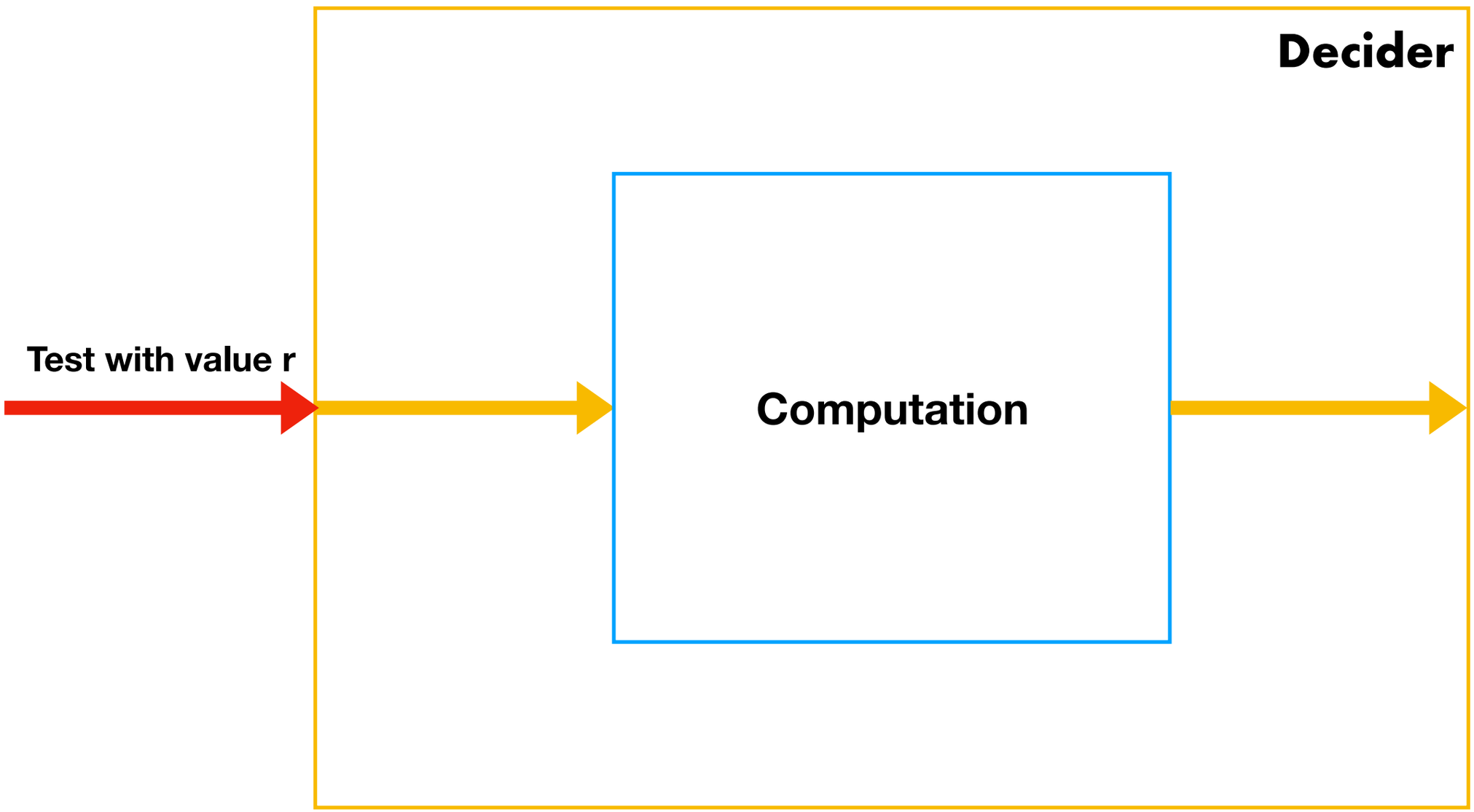
Net & Prune framework



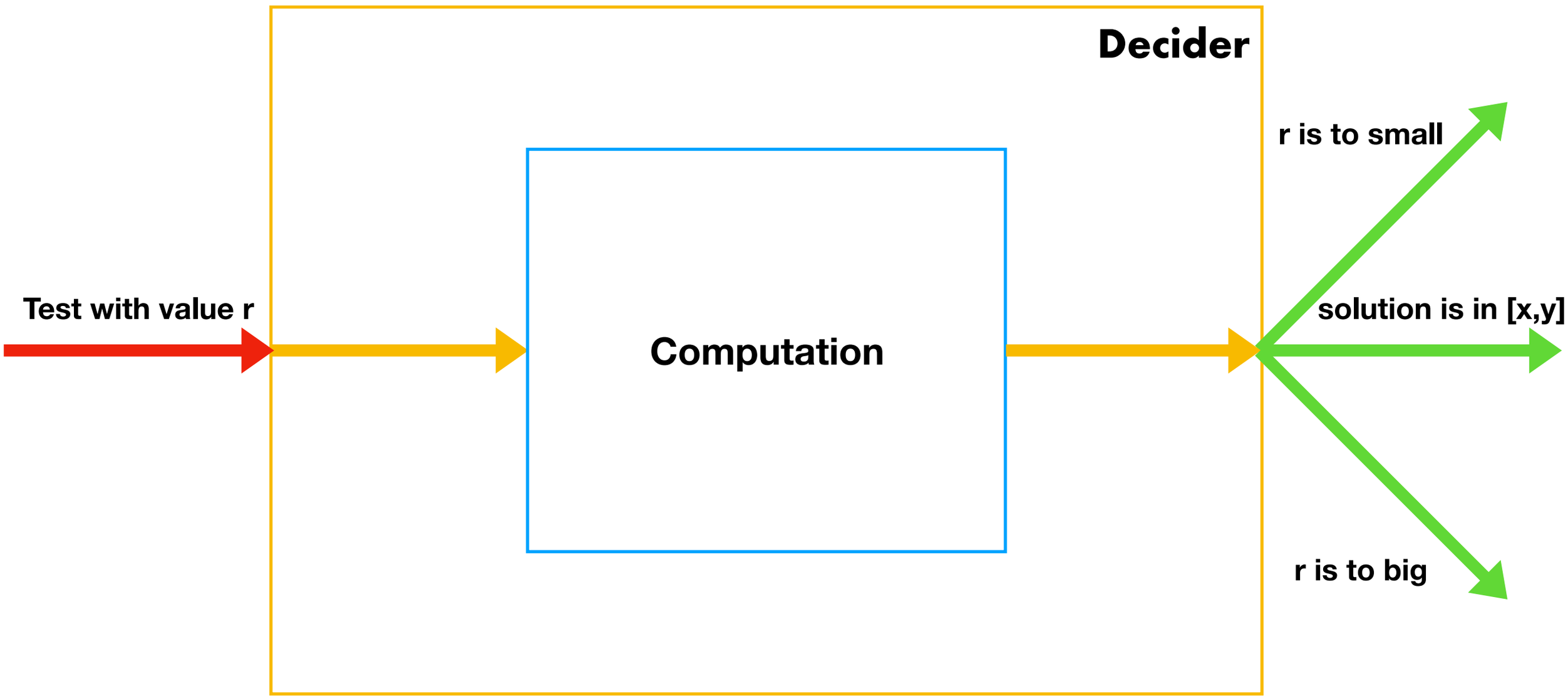
Decider



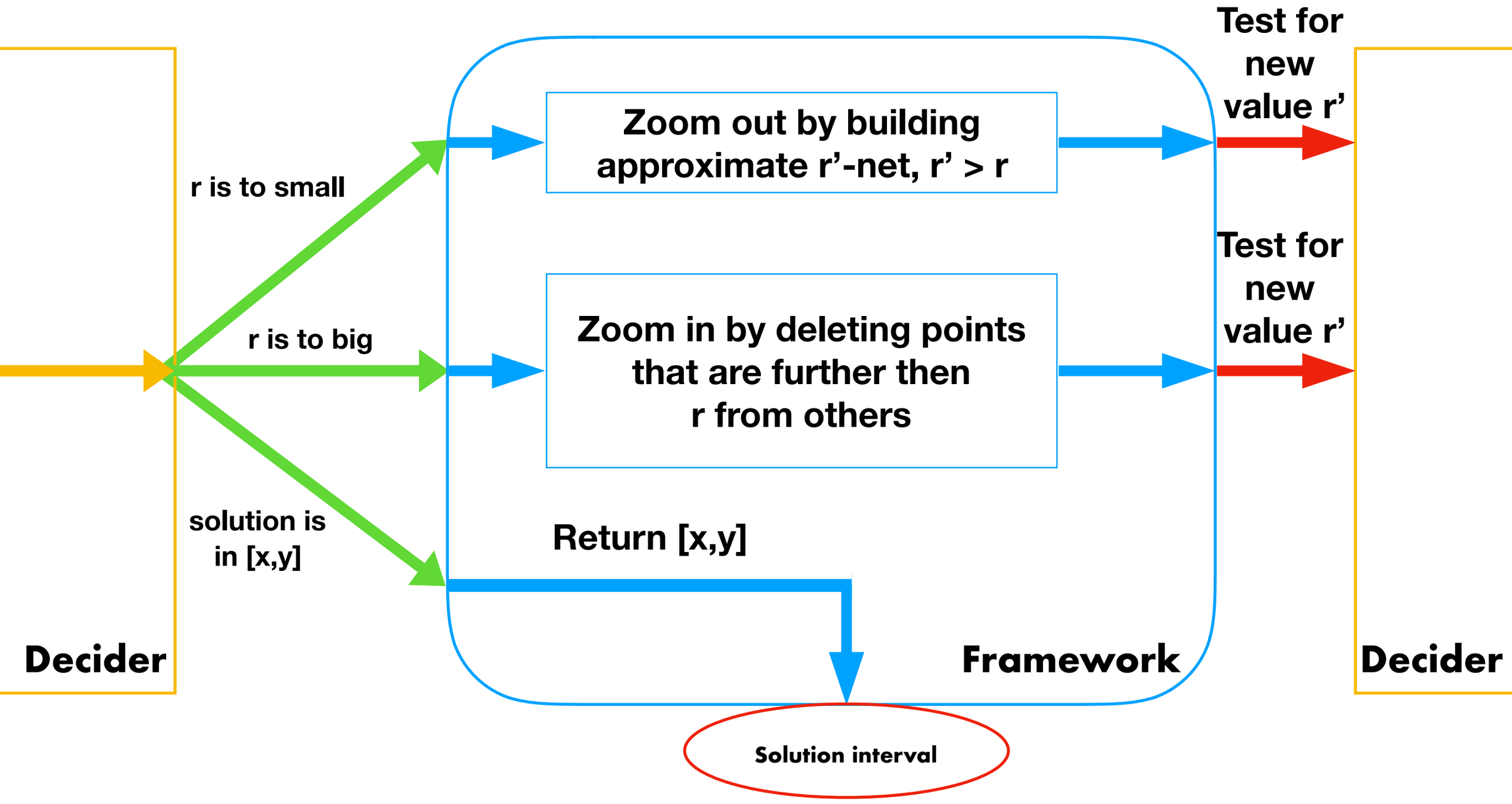
Decider



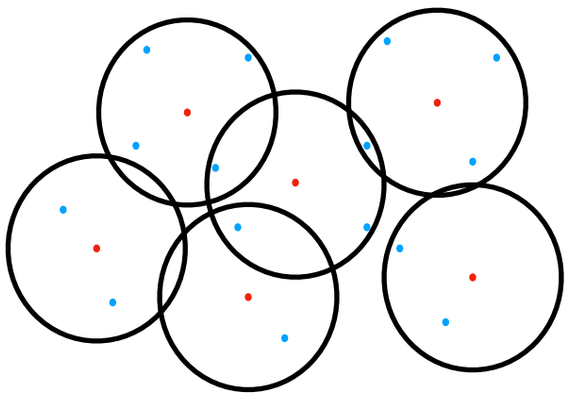
Decider


Decider
Test with value r
Computation
r is to small
solution is in [x,y]
r is to big

Decider


Test for new value r'
r is to small
Zoom out by building approximate r'-net, r' > r
Test for new value r'
r is to big
Zoom in by deleting points that are further then r from others
solution is in [x,y]
Return [x,y]
Decider
Framework
Decider
Solution interval

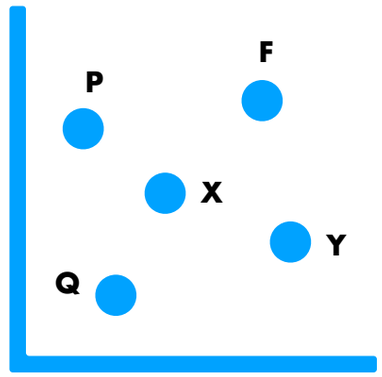
R-nets



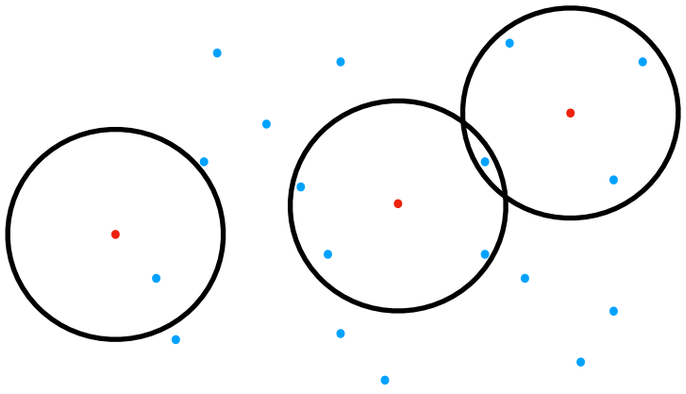
Distance matrix

	p_1	p_2		p_n
	yes	no	...	yes
	no	no	...	yes
	no	yes	...	no

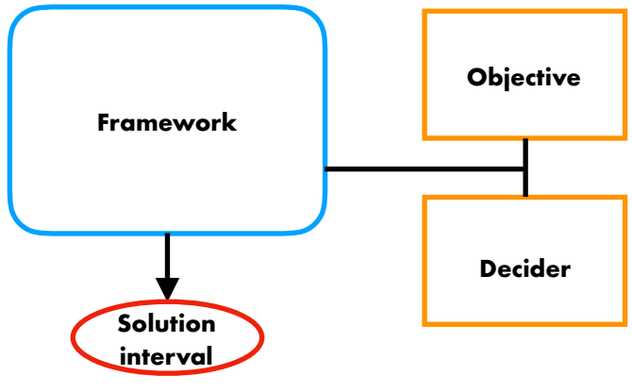
PTF



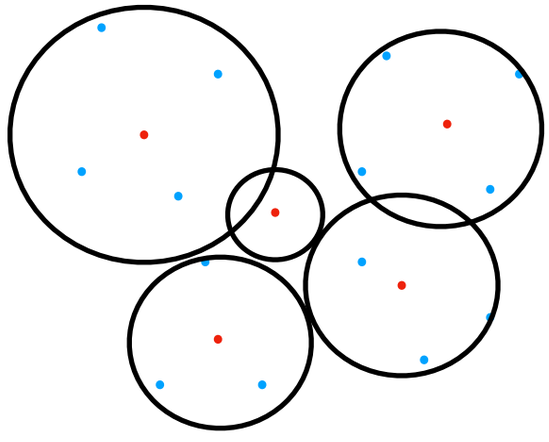
Sparsification



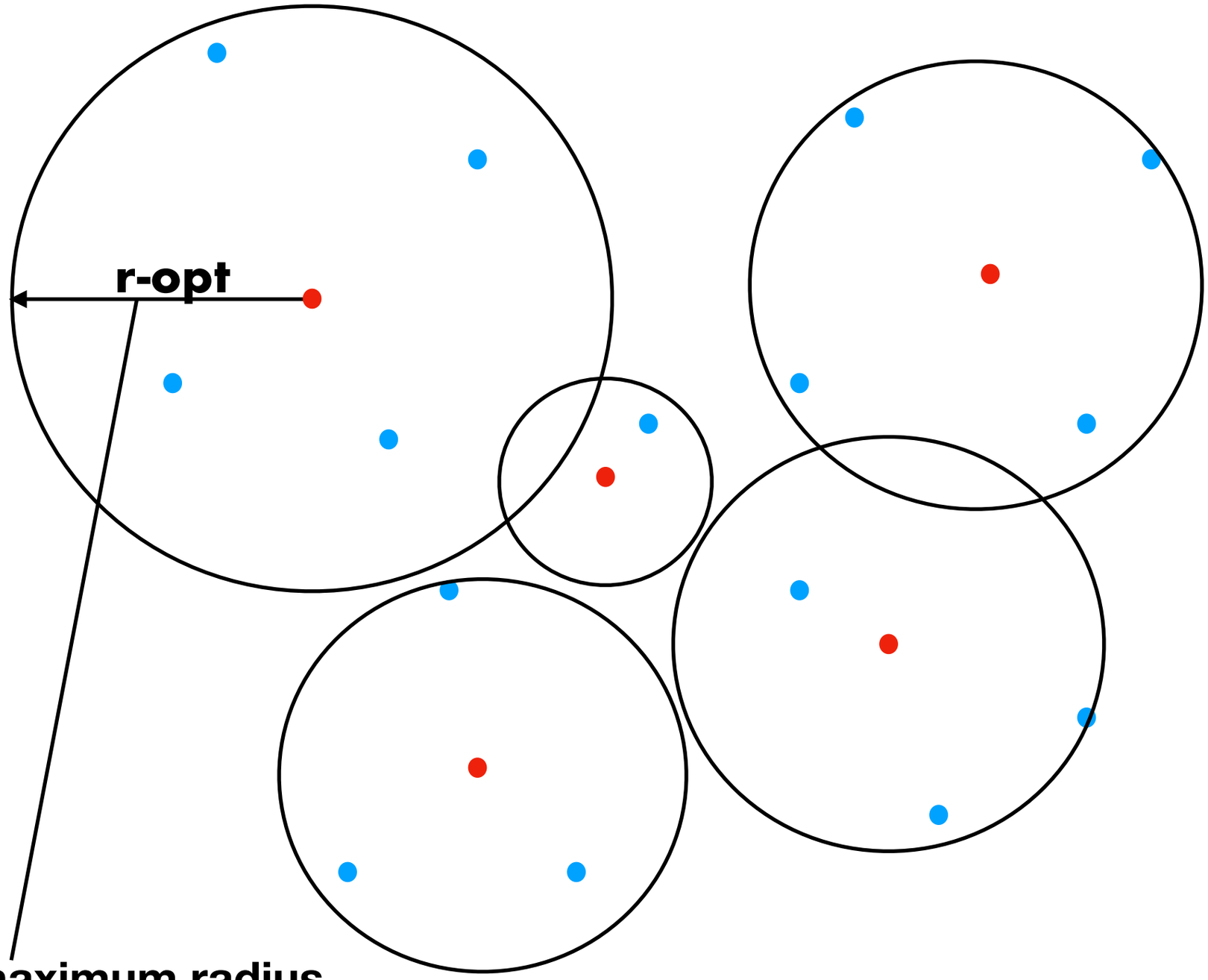
Net & Prune



K-center

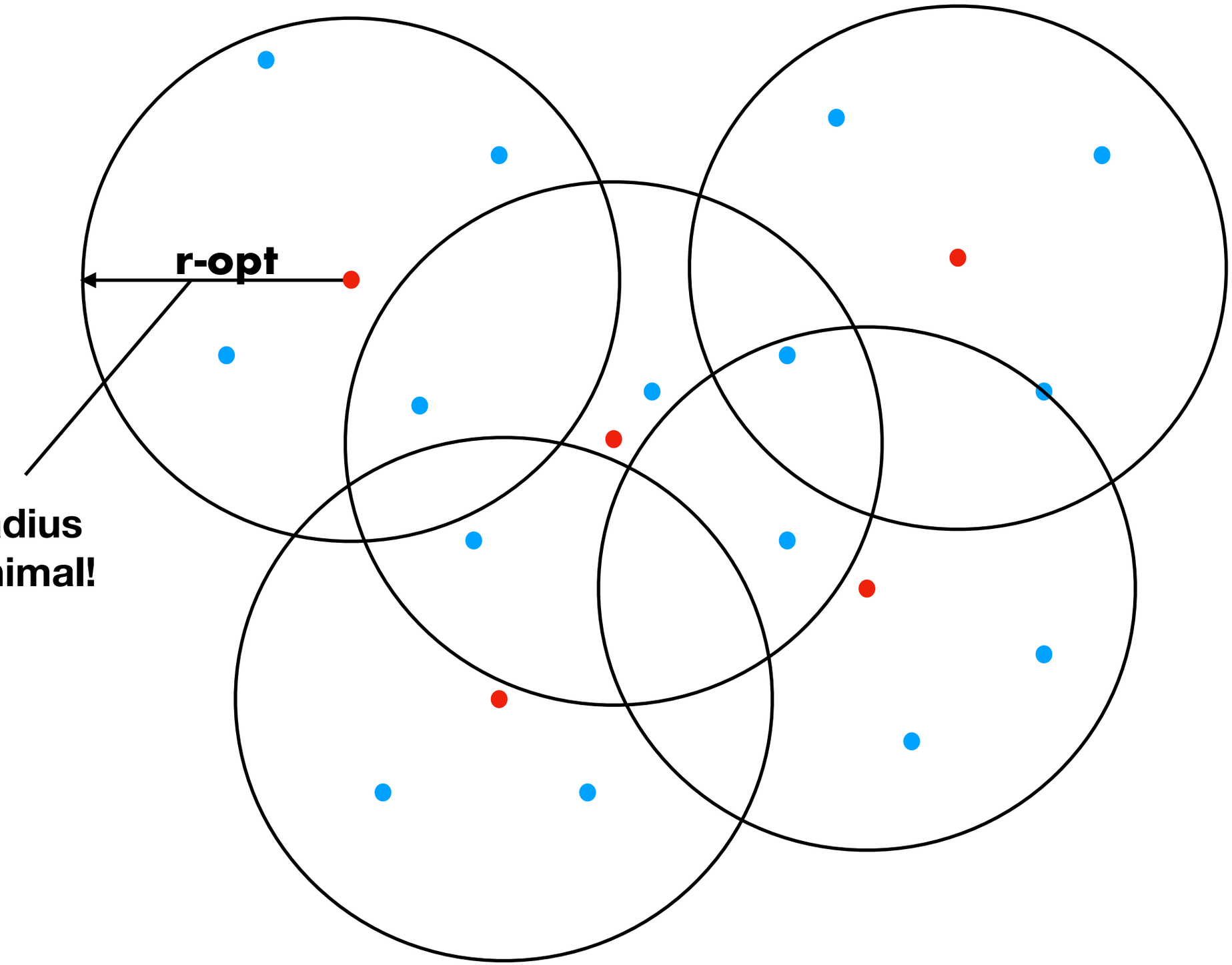


K-Center with k=5



Minimize maximum radius

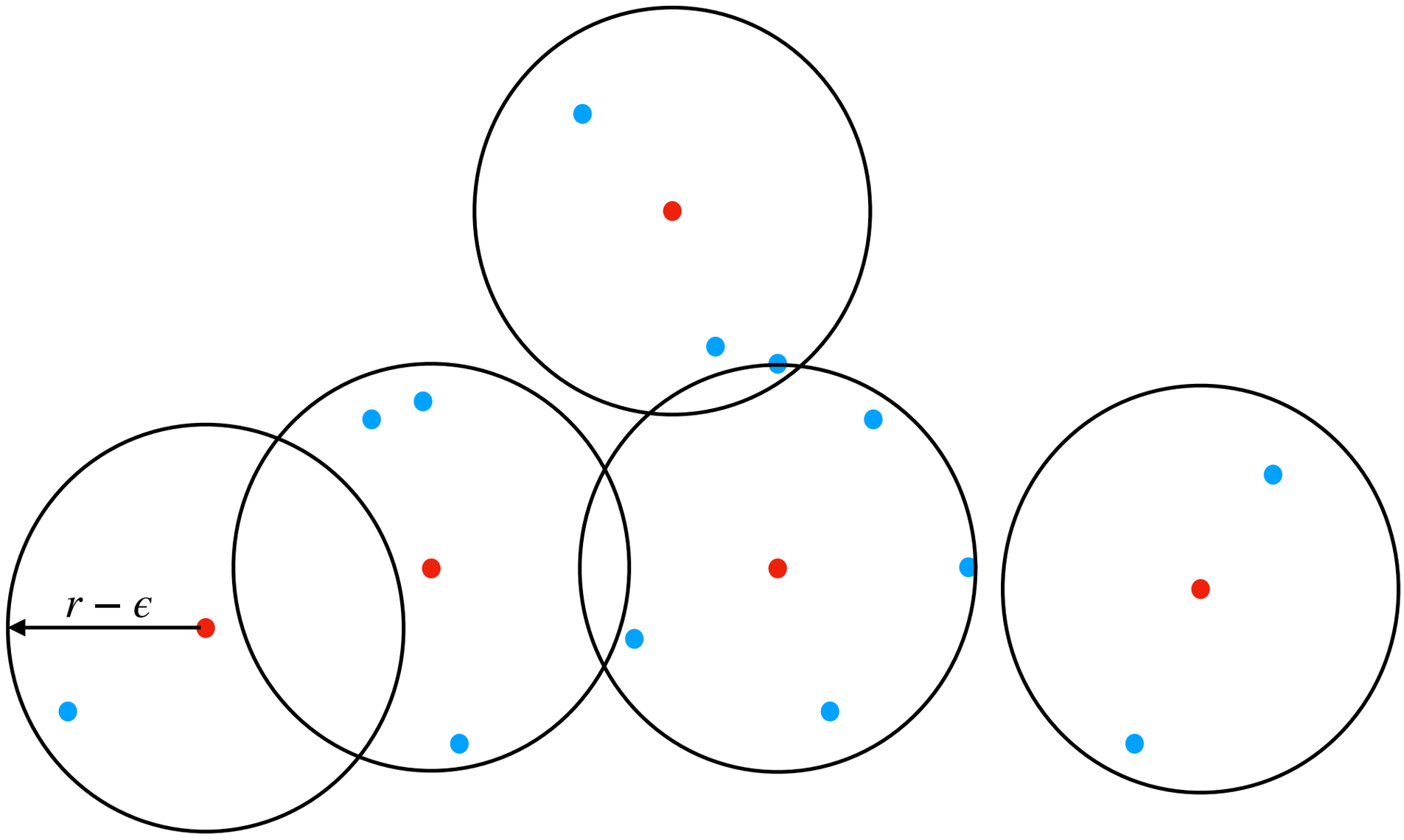
r-opt net = 5-center cluster



Max radius still minimal!

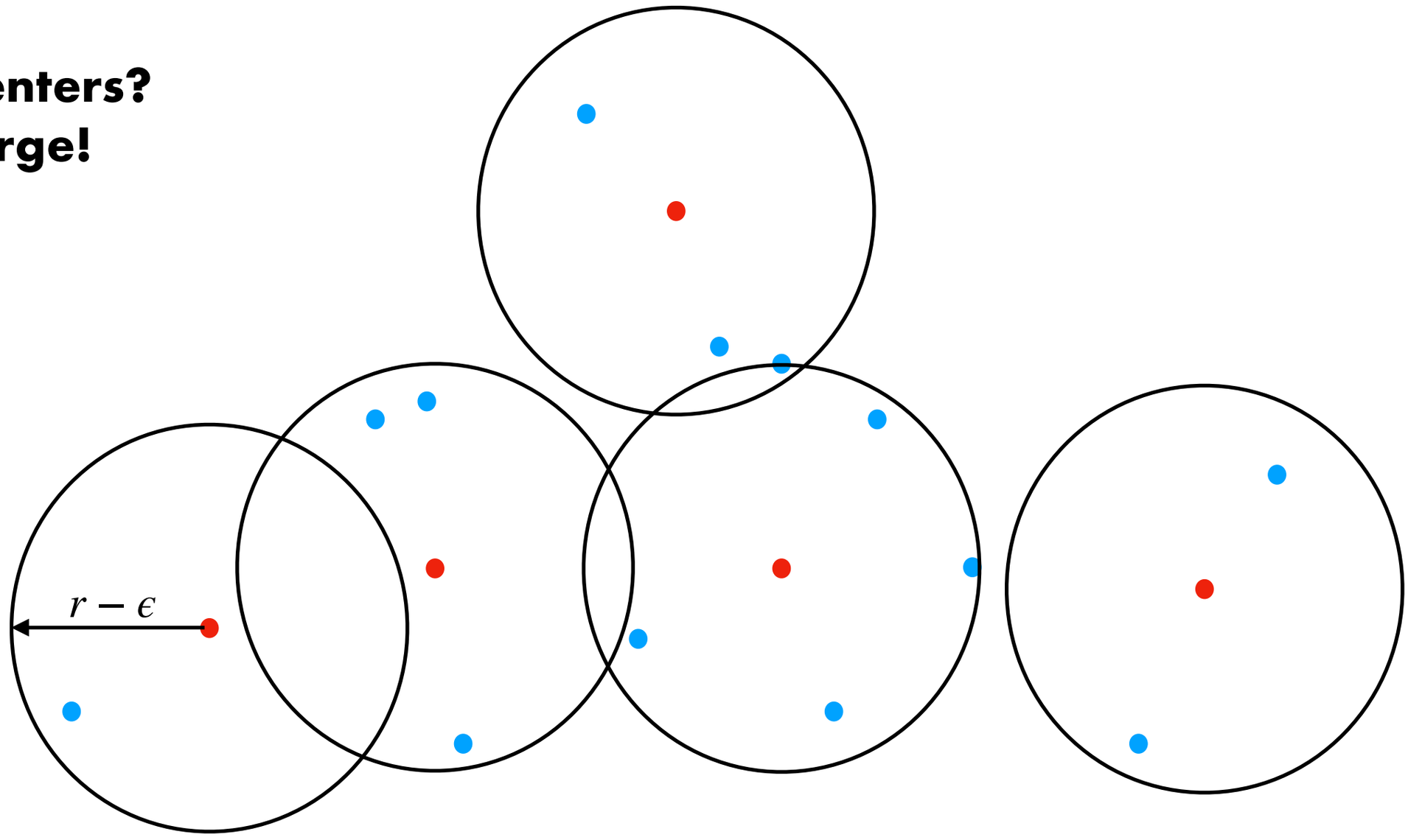
Solve with the framework by building a decider!

Test for value r?

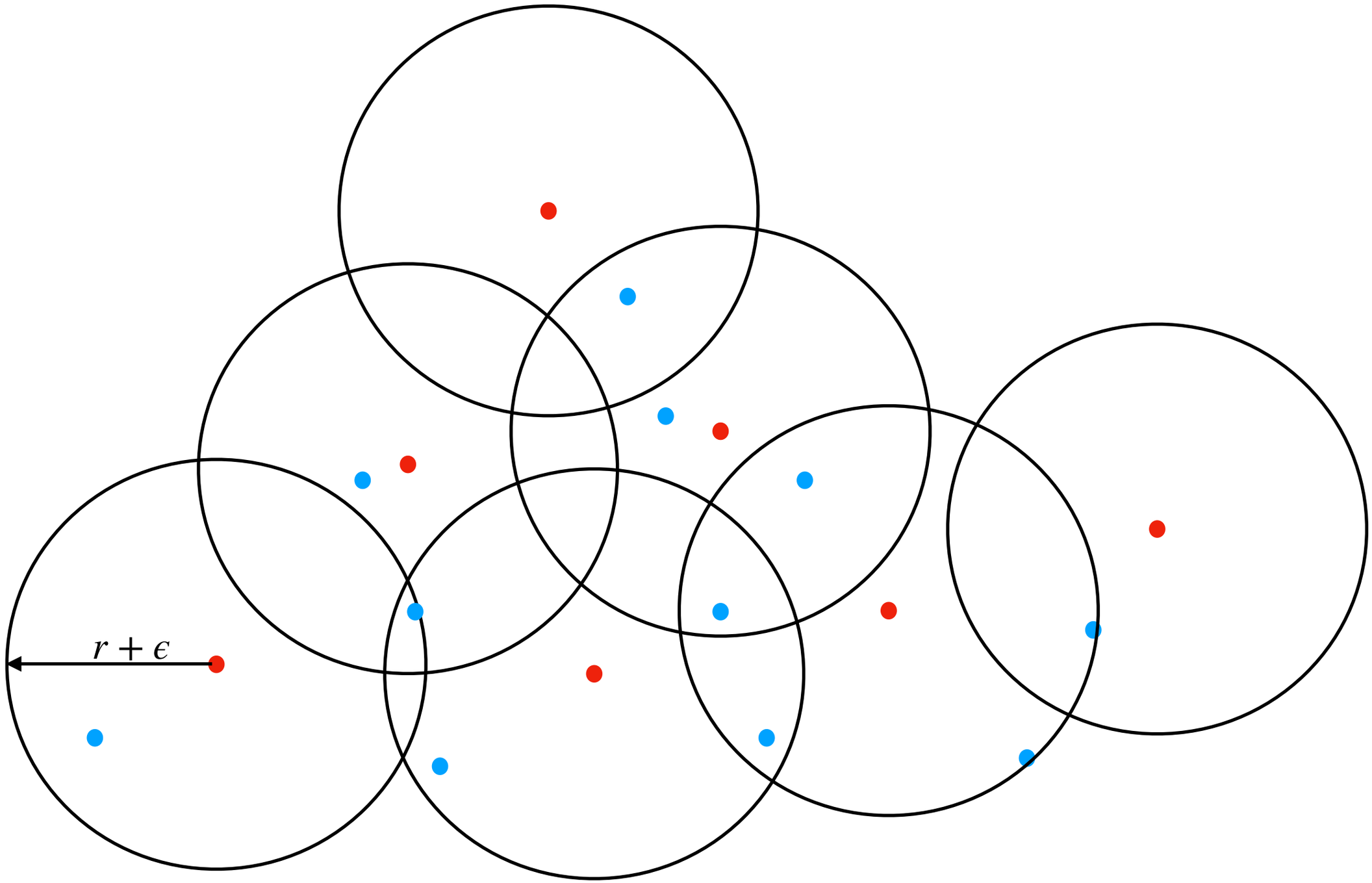


Test for value r?

k or less centers?
r is to large!

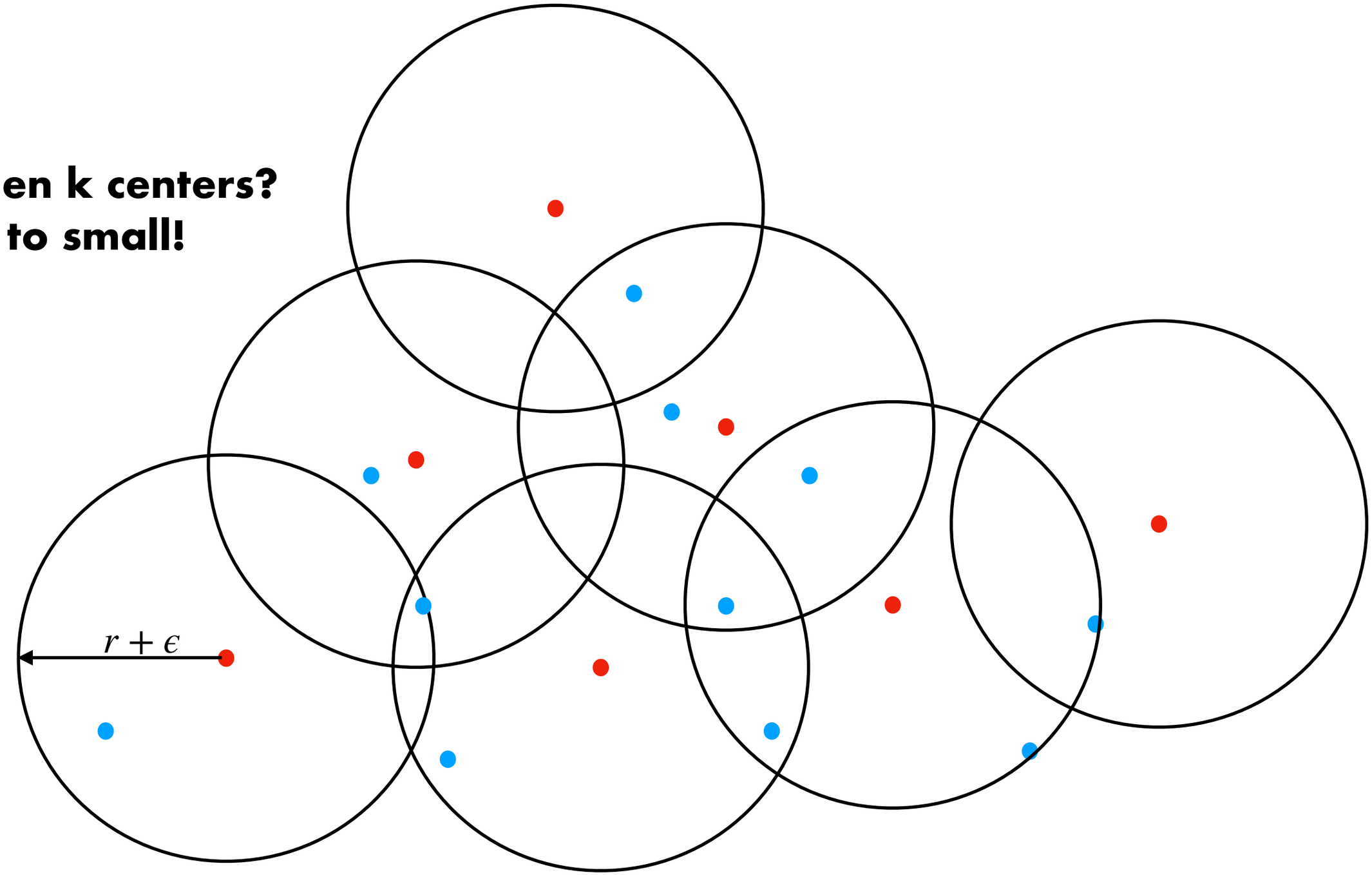


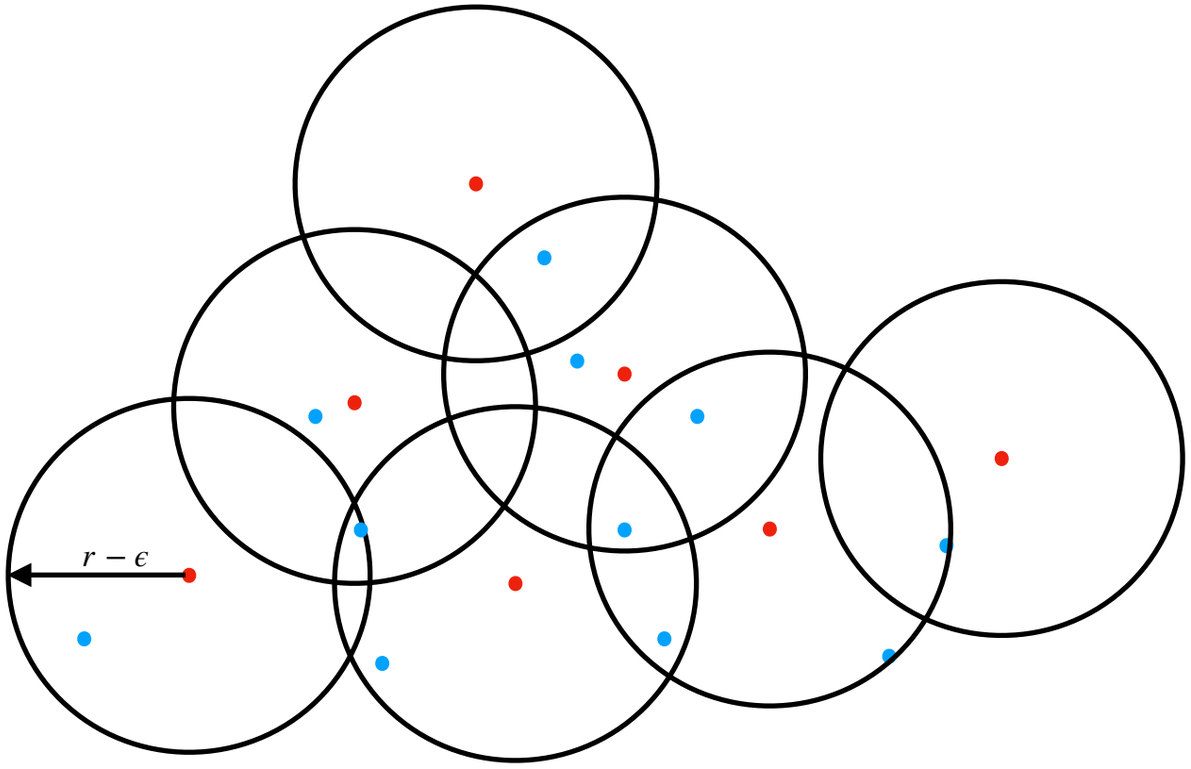
Test for value r?

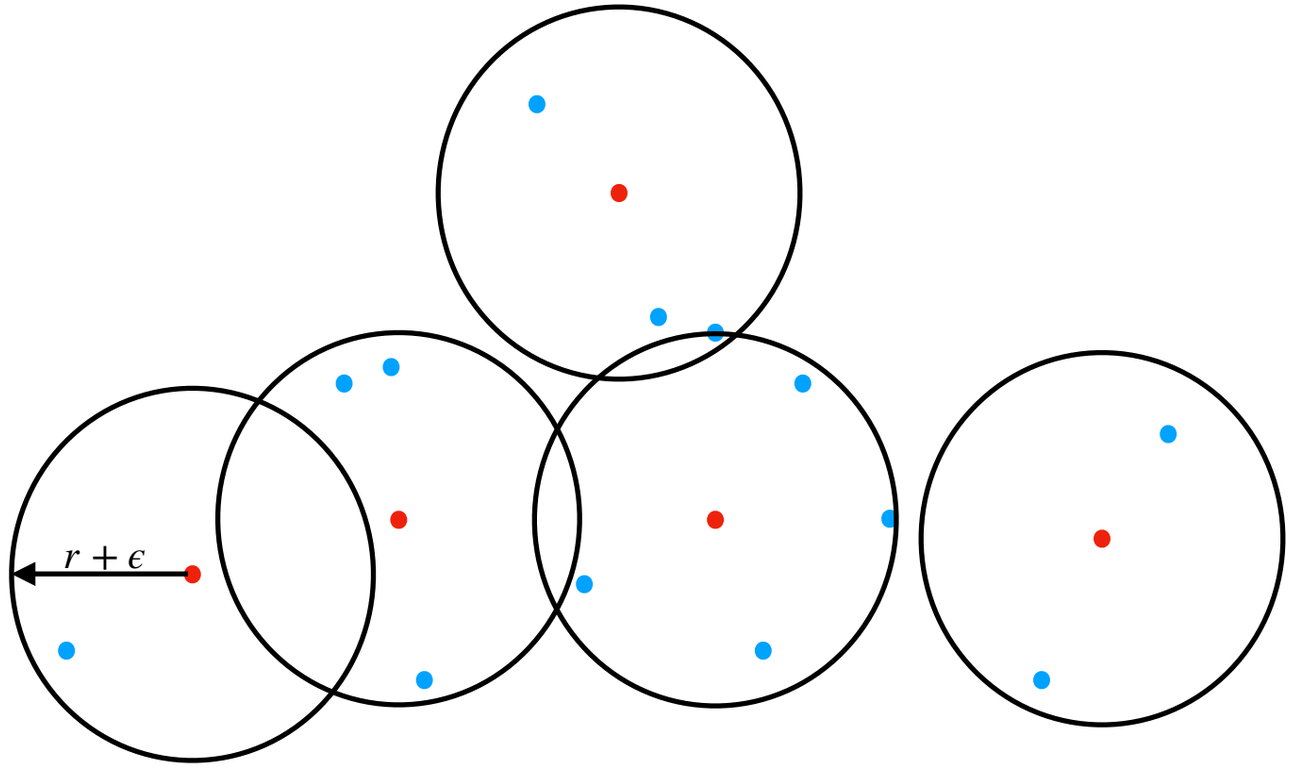


Test for value r?

more then k centers?
r is to small!







More then k centers?

k or less centers?

Optimal solution in [x,y]

Binary search with the decider

Return r where $r_{opt} \in [r, r(4+\epsilon)]$

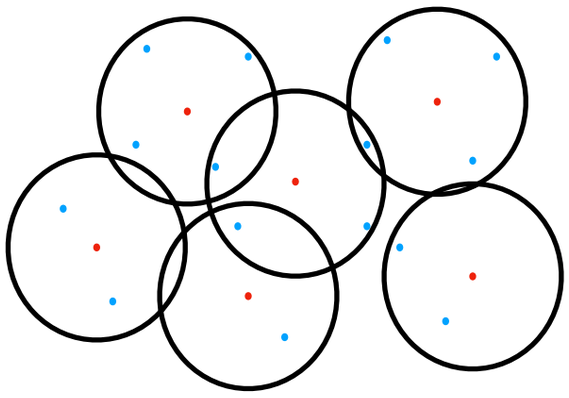
Optimal solution in [x,y]

Slice the interval into pieces

Binary search with the decider

Get a smaller interval [x’,y’], that contains the solution

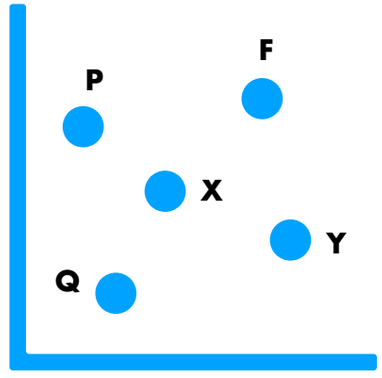
R-nets



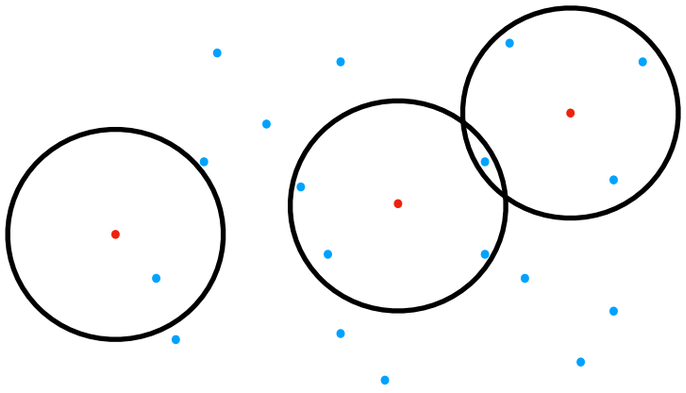
Distance matrix

	p_1	p_2		p_n
	yes	no	...	yes
	no	no	...	yes
	no	yes	...	no

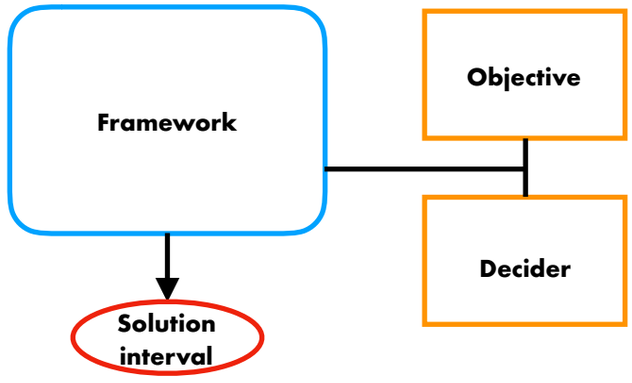
PTF



Sparsification



Net & Prune



K-center