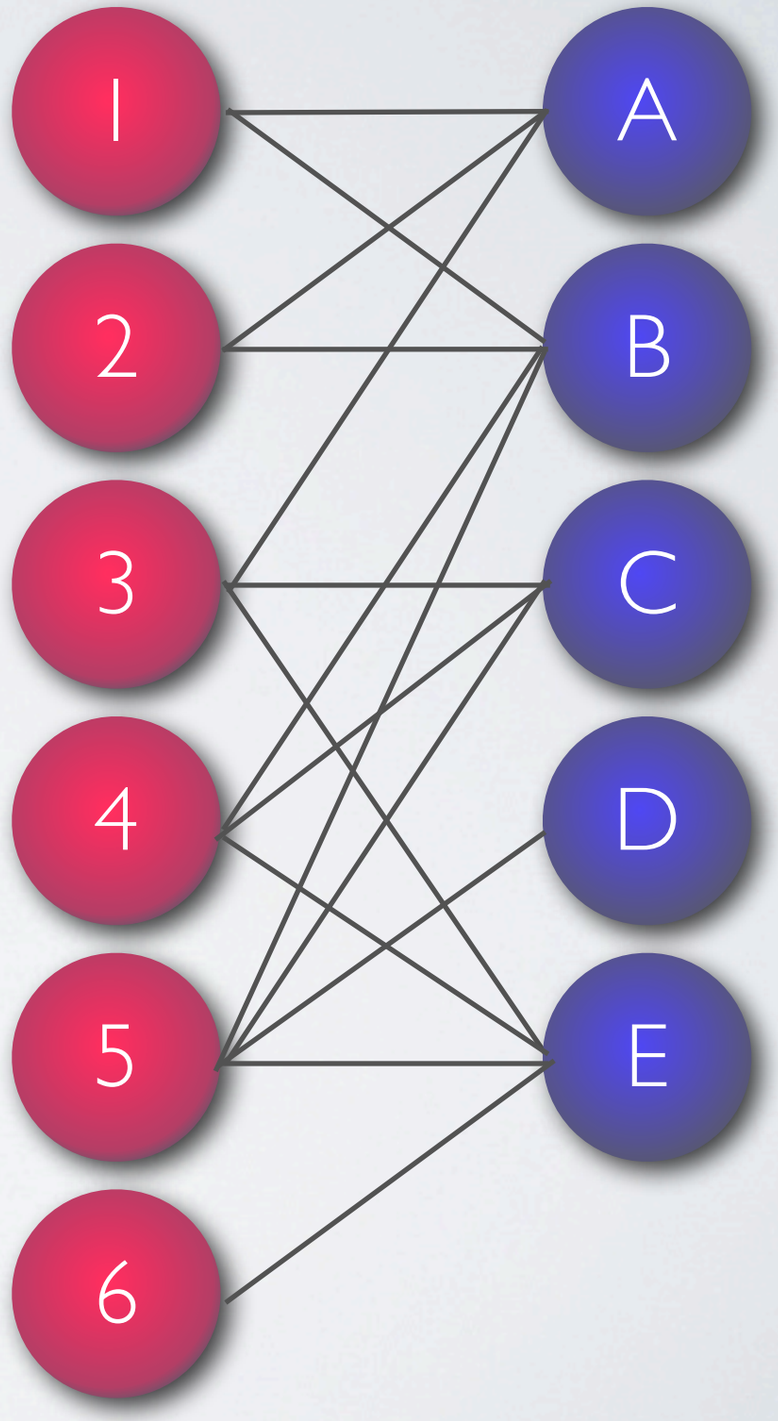
DYNAMIC BOOLEAN MATRIX FACTORIZATIONS

Pauli Miettinen

13 December 2012

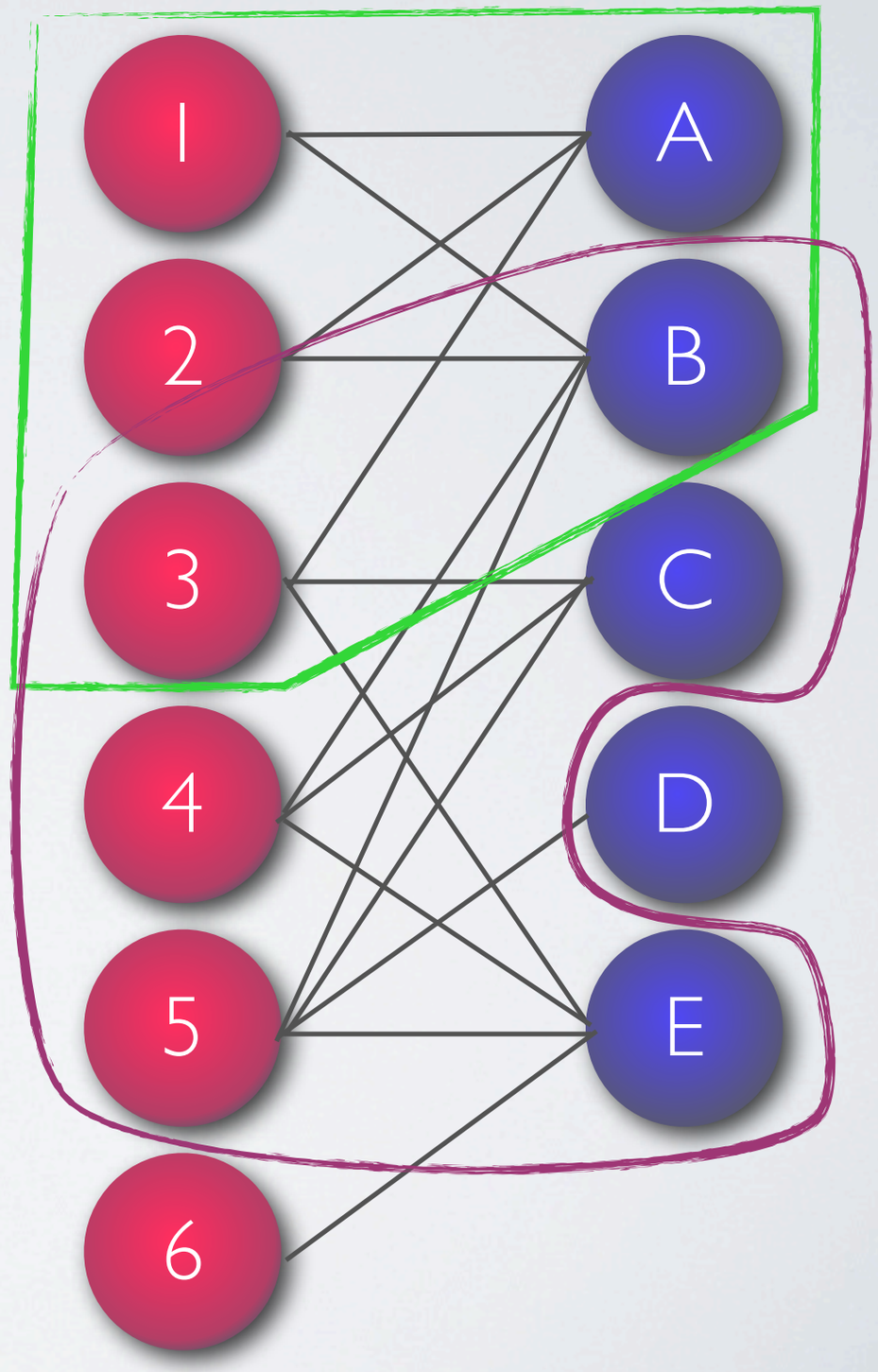
USERS AND WEBPAGES

- Users "like" webpages
  - A bipartite graph
- We want to know (approximate) bicliques of users who like similar webpages



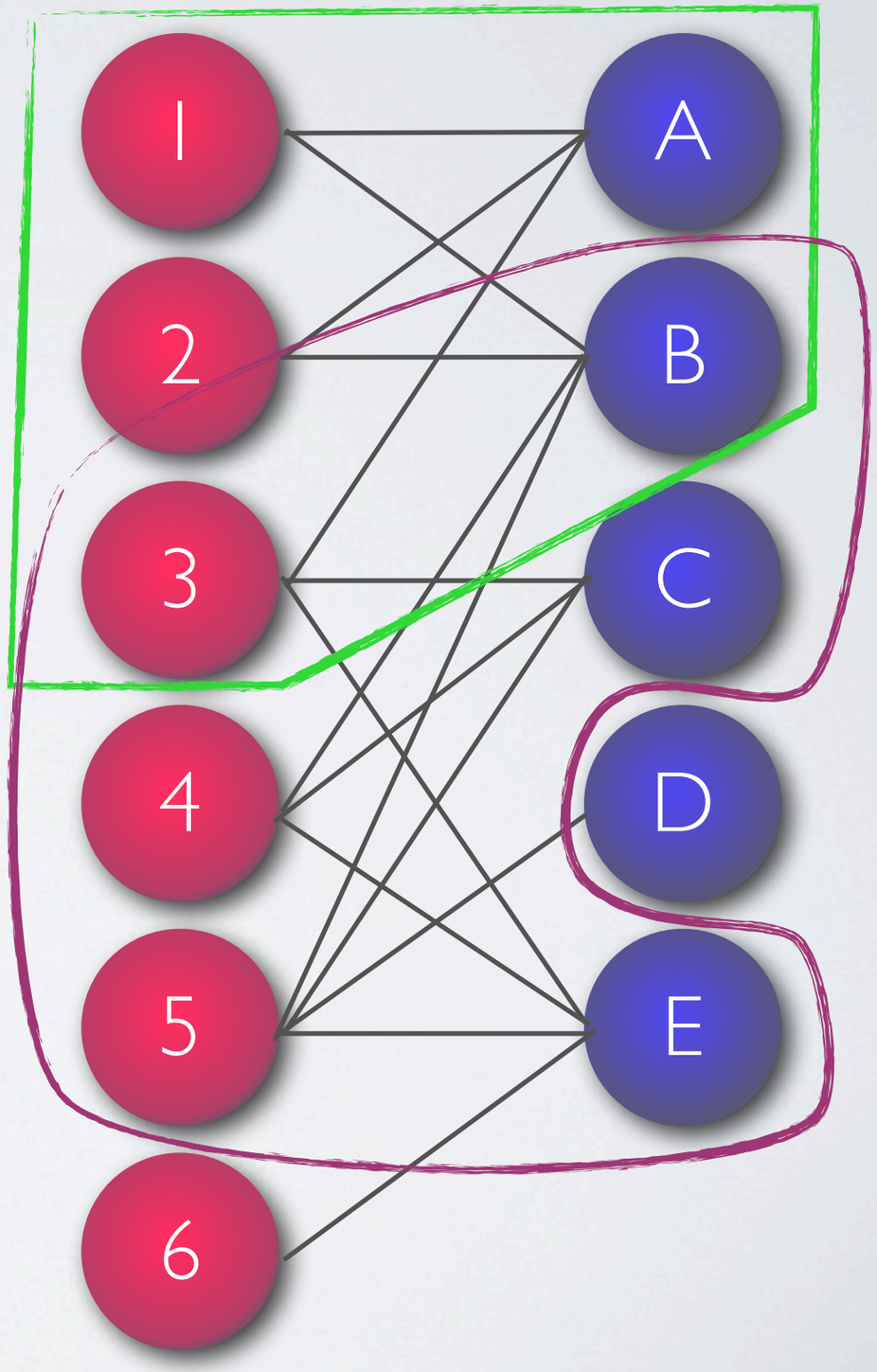
USERS AND WEBPAGES

- Users "like" webpages
  - A bipartite graph
- We want to know (approximate) bicliques of users who like similar webpages

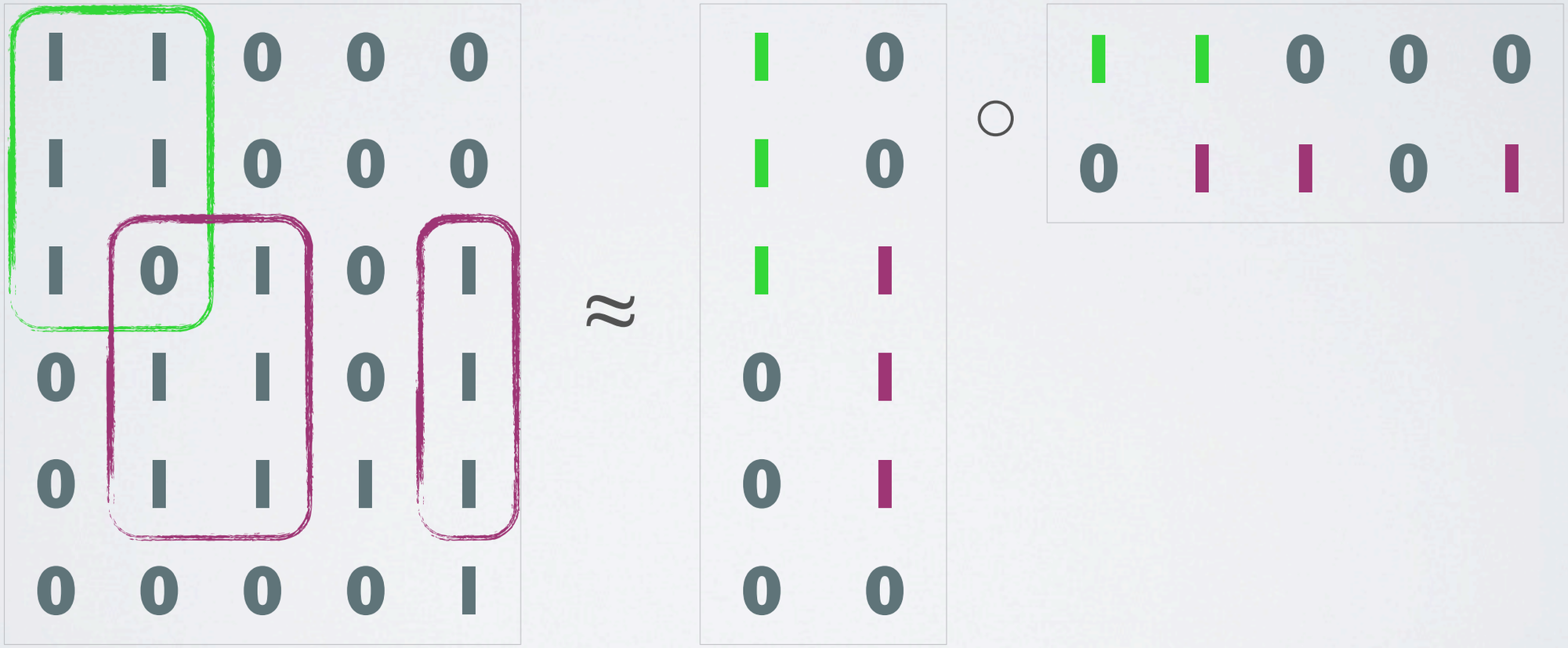


BOOLEAN MATRIX

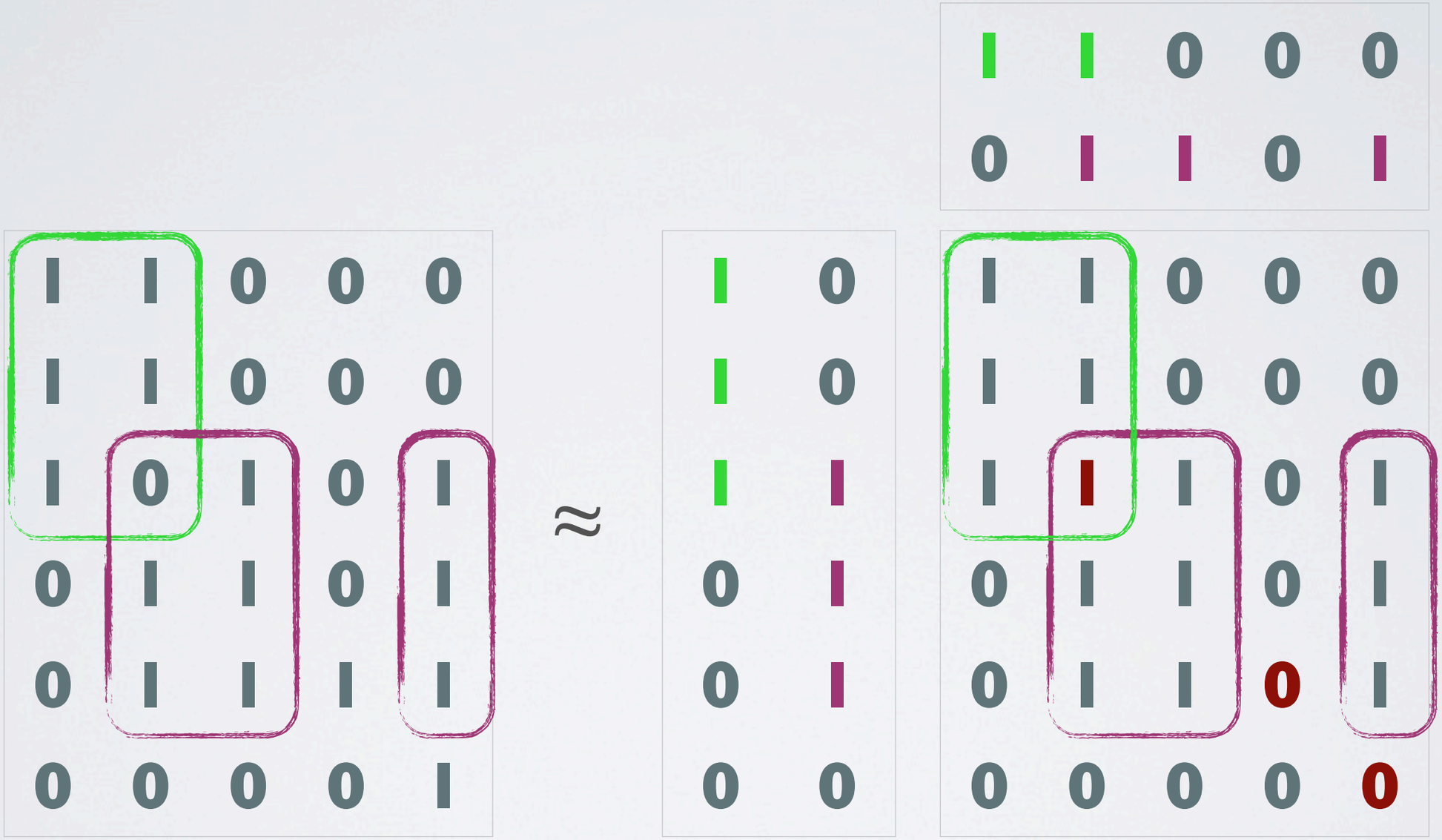
1	1	0	0	0
1	1	0	0	0
1	0	1	0	1
0	1	1	0	1
0	1	1	1	1
0	0	0	0	1



BOOLEAN MATRIX FACTORIZATIONS



BOOLEAN MATRIX FACTORIZATIONS



max planck institut informatik

DYNAMIC FACTORIZATIONS

- Users keep on liking new webpages
  - New users, new webpages, and old users liking old pages
- We want our factorization to adapt to new data
- **Problem.** Given a binary matrix ***A***, its Boolean factorization (***B***, ***C***), and a series of added 1s to ***A***, update ***B*** and ***C*** so that they define a good approximation of ***A*** after any addition

NOTES

- We're only adding 1s to the data
  - You can't dislike a page
- We're **not** doing prediction, we're adapting
  - Being good at predicting helps adapting, though

FIRST ATTEMPT

- We can re-compute the factorization after every addition
- Too slow
- Too much effort given the minimal change

SECOND ATTEMPT

- We can fold-in the new data: if we add a new column to ***A***, we keep ***B*** fixed and add a new column to ***C***
  - Common in IR when new terms/documents appear
- But we're not necessarily adding new rows or columns
  - We could still do alternating updates to ***B*** and ***C***, except that the problem is NP-hard to even approximate well

THIRD ATTEMPT

- An **online** algorithm
  - Will never remove any 1s it has added to factor matrices
- We consider three cases when a new 1 arrives

EXAMPLE

1	1	0	0	0
0	1	1	0	1

1	1	0	0	0
1	0	1	0	1
0	1	1	0	1
0	1	1	1	1
0	0	0	0	1

≈

1	0
1	1
0	1
0	1
0	0

1	1	0	0	0
1	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	0	0	0	0

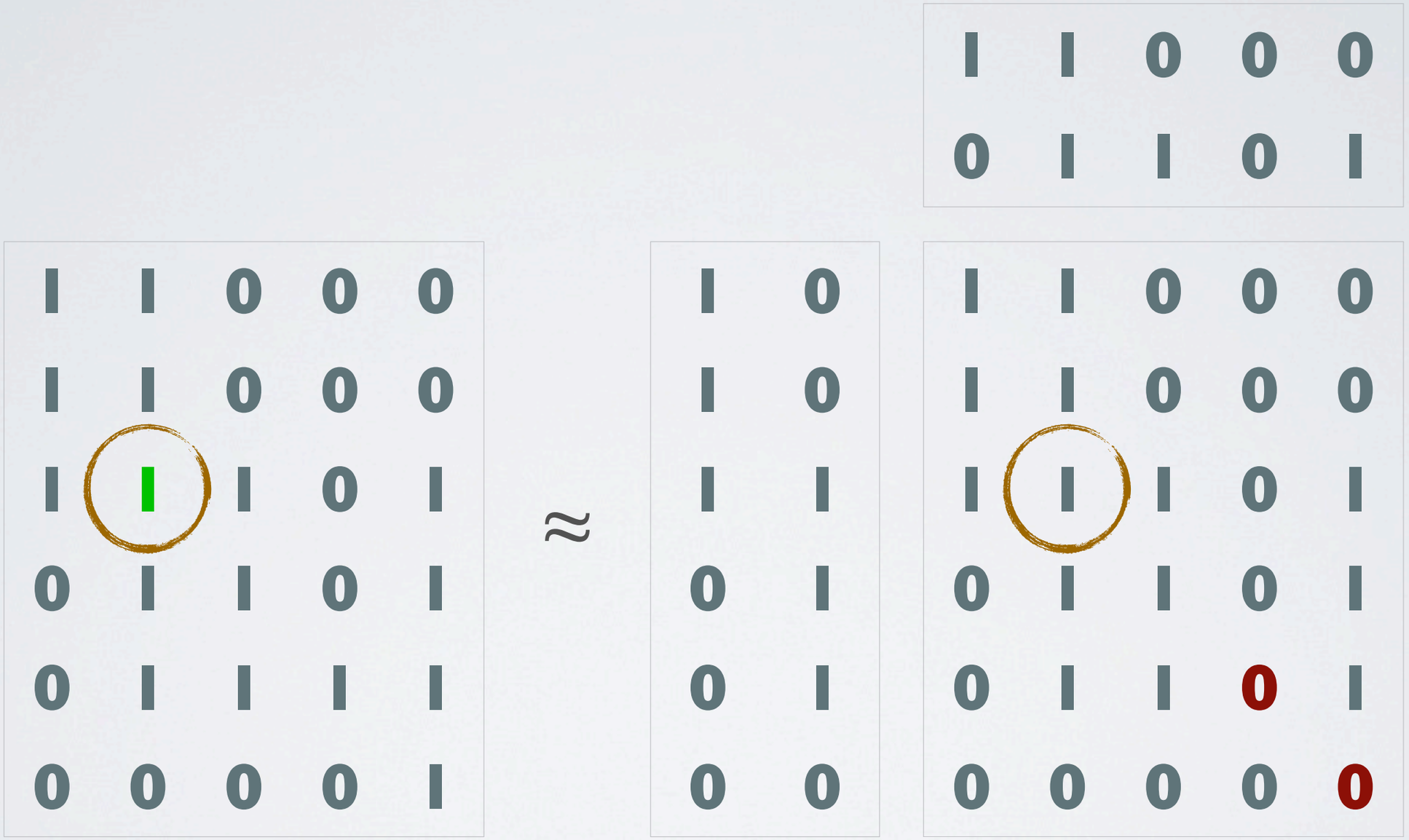
CASE 1: PREDICTED 1

max planck institut informatik

CASE 1: PREDICTED 1

max planck institut informatik

CASE 1: PREDICTED 1



max planck institut informatik

CASE 2: NO PRIOR INFO

max planck institut informatik

CASE 2: NO PRIOR INFO

1	1	0	0	0
0	1	1	0	1

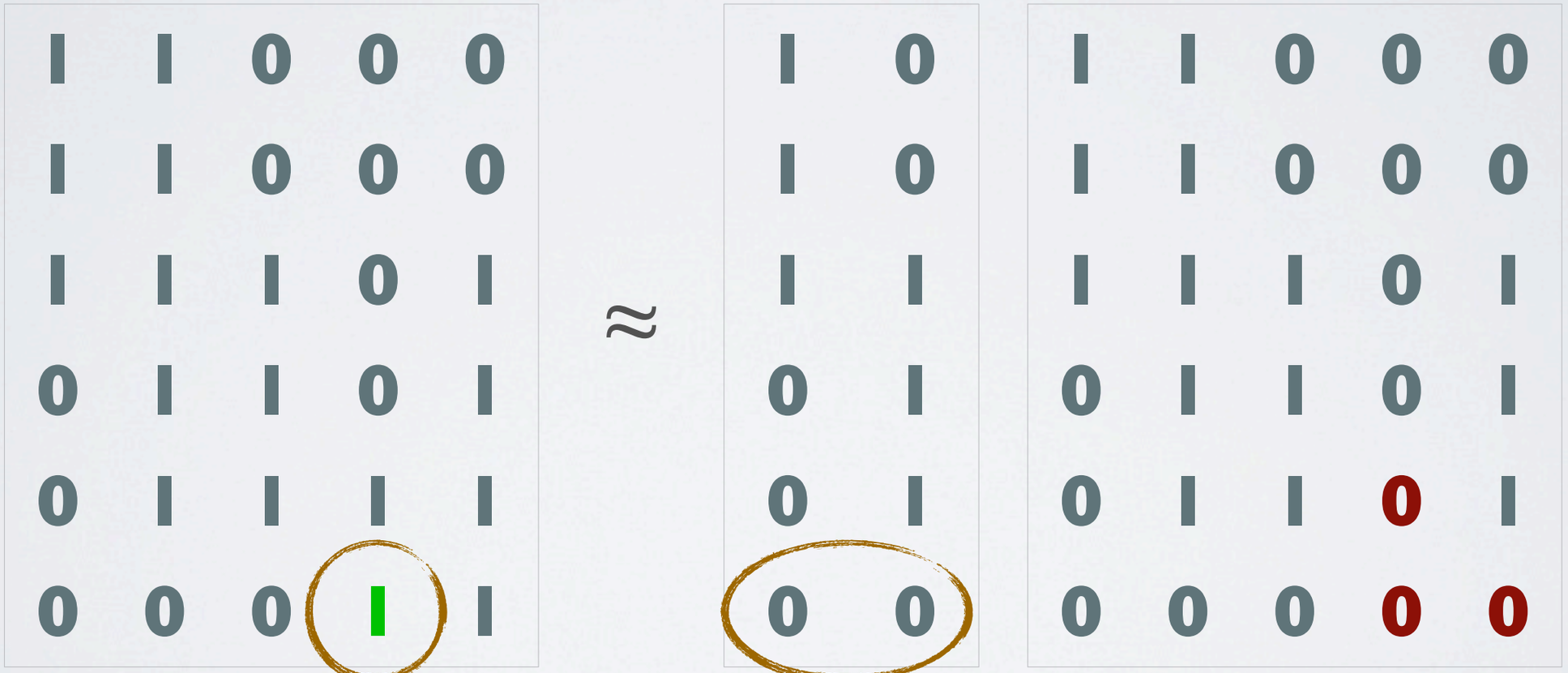
1	1	0	0	0
1	1	0	0	0
1	1	1	0	1
0	1	1	0	1
0	1	1	1	1
0	0	0	1	1

≈

1	0
1	1
0	1
0	1
0	0

1	1	0	0	0
1	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	0	0	0	0

CASE 2: NO PRIOR INFO



max planck institut informatik

CASE 3: PARTIAL INFO

max planck institut informatik

CASE 3: PARTIAL INFO

1	1	0	0	0
0	1	1	0	1

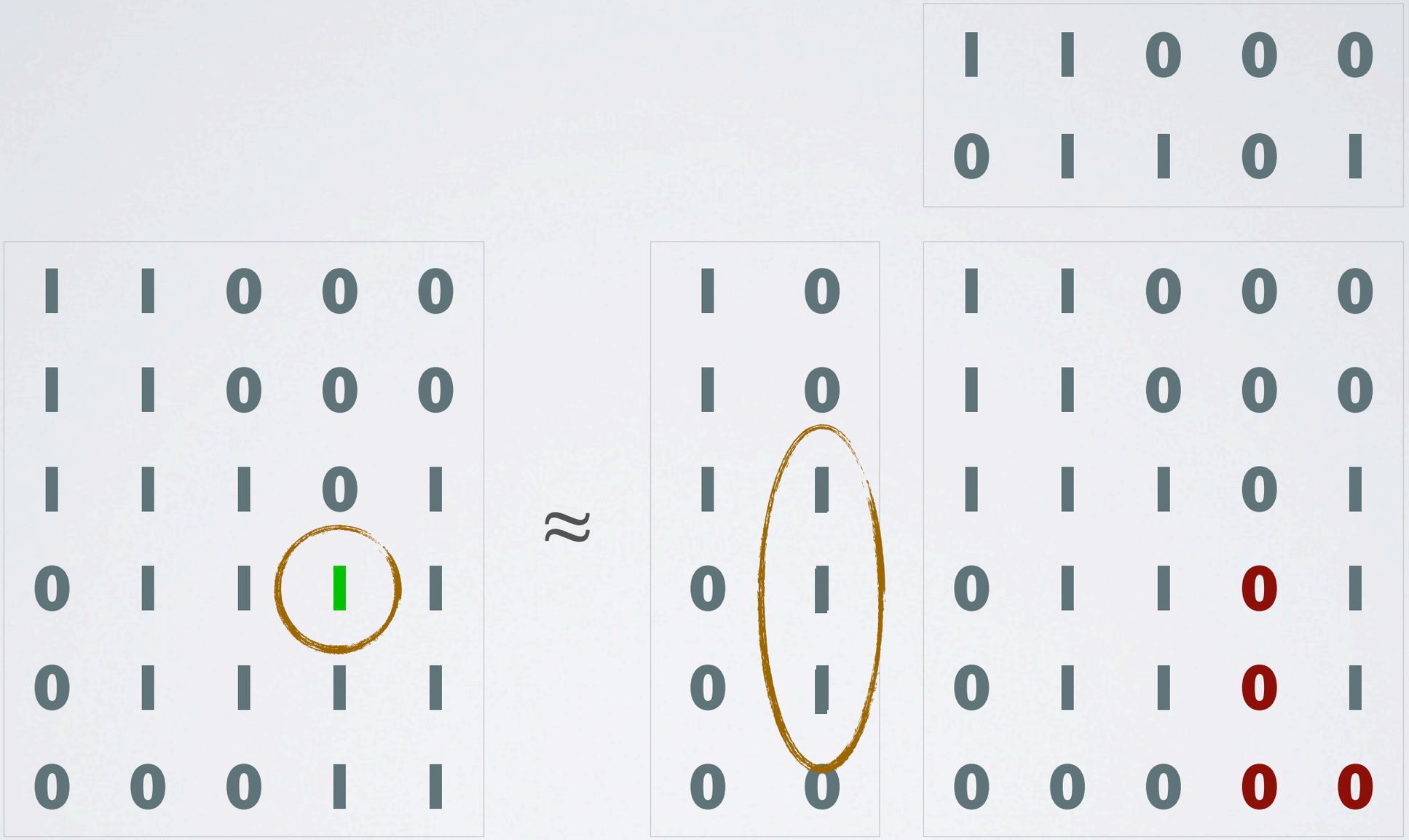
1	1	0	0	0
1	1	0	0	0
1	1	1	0	1
0	1	1	1	1
0	1	1	1	1
0	0	0	1	1

≈

1	0
1	1
0	1
0	1
0	0

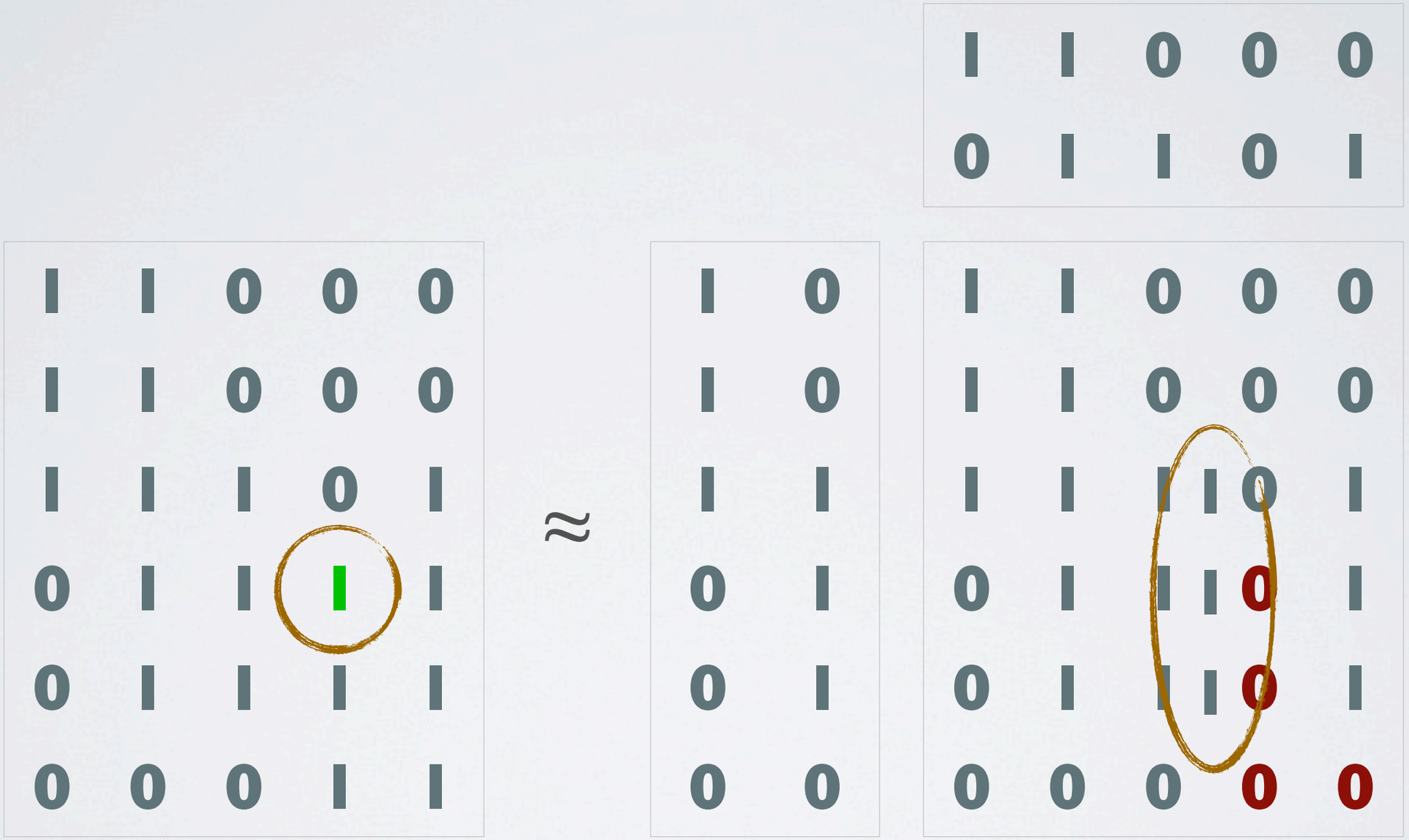
1	1	0	0	0
1	1	1	0	1
0	1	1	0	1
0	1	1	0	1
0	0	0	0	0

CASE 3: PARTIAL INFO



max planck institut informatik

CASE 3: PARTIAL INFO

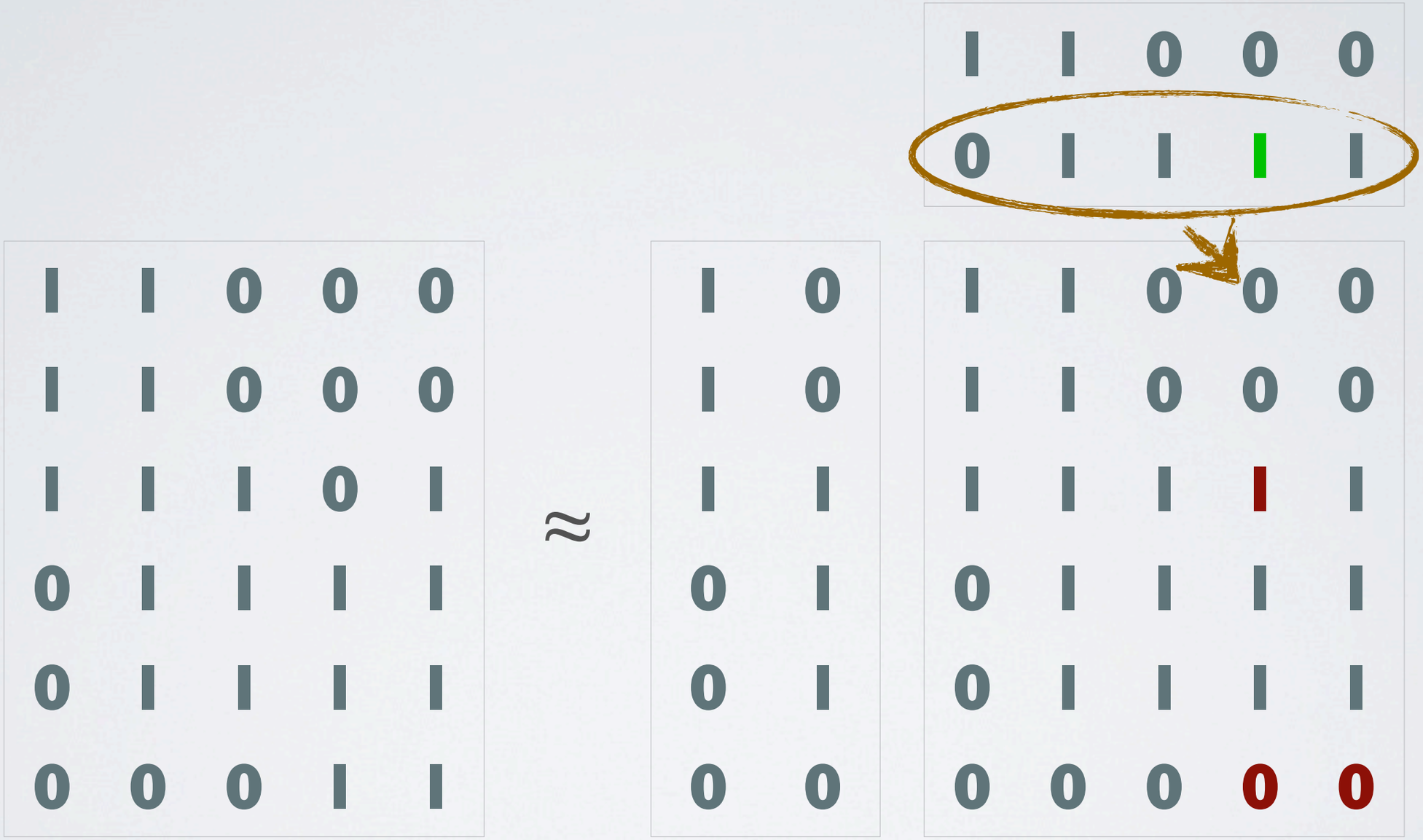


max planck institut informatik

CASE 3: PARTIAL INFO

max planck institut informatik

CASE 3: PARTIAL INFO

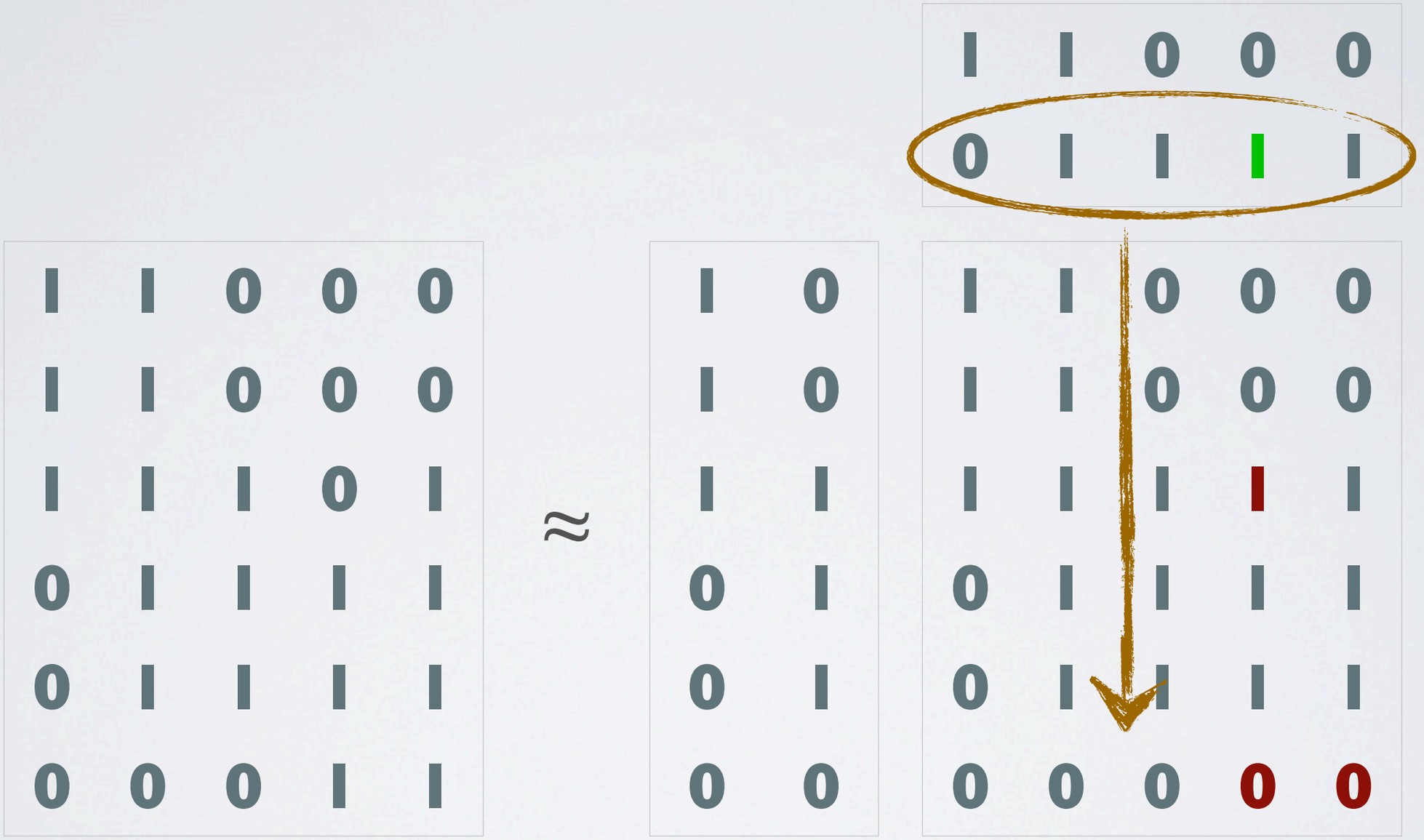


max planck institut informatik

CASE 3: PARTIAL INFO

max planck institut informatik

CASE 3: PARTIAL INFO



CASE 3: PARTIAL INFO

MAKING UPDATES FAST

- Recognizing the case is $O(k)$ with proper index structure
- Selecting the factor in Case 3 is worst-case $O(|\boldsymbol{B}| + |\boldsymbol{C}|)$
- Extending the factor can be very costly
  - Computing the fit for every row/column
  - We store the historical fits and take an optimistic approach on how much it could have improved
    ⇒We only need to consider those rows/columns where the factor could give a good fit

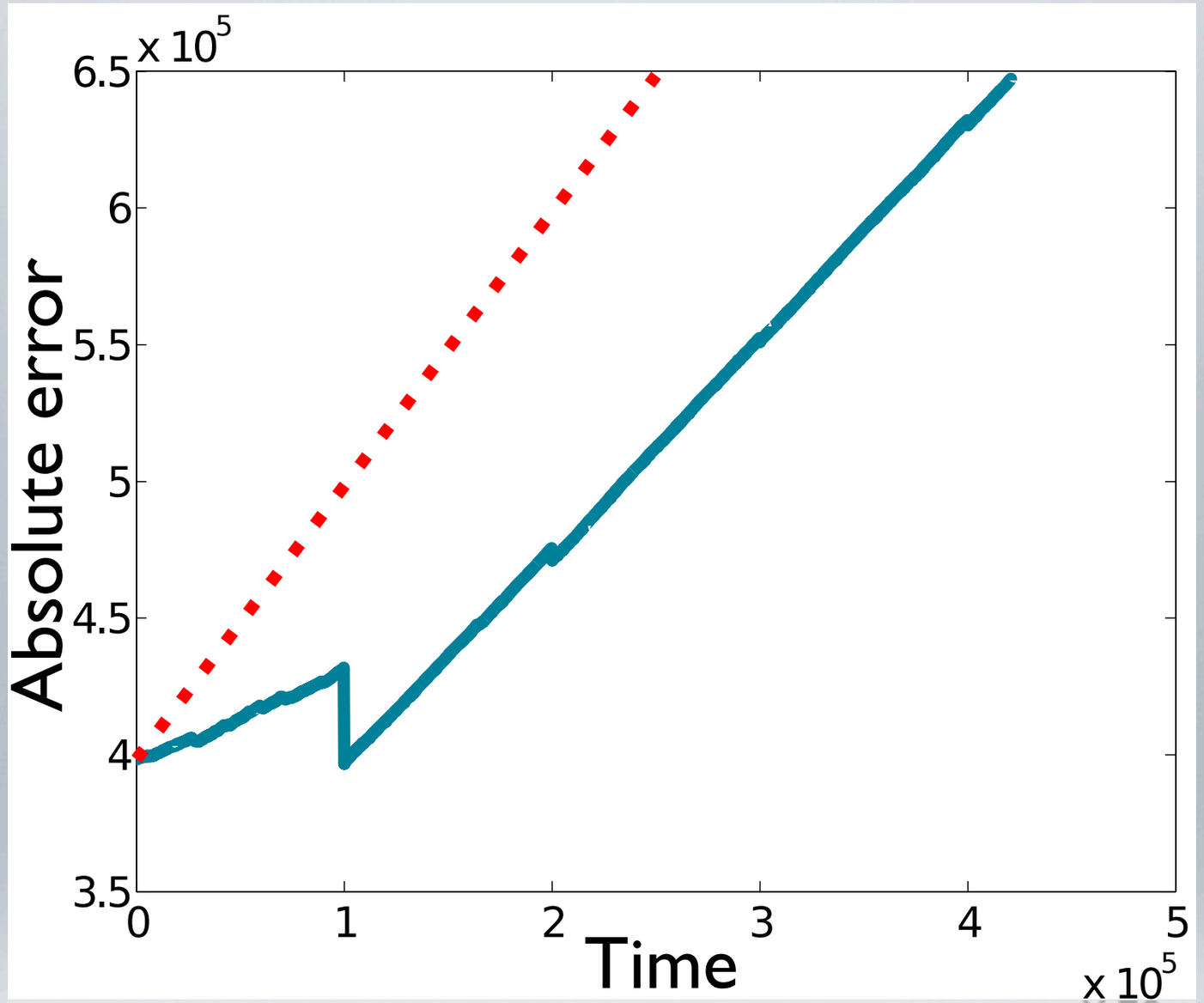
NON-ONLINE ALGORITHM

- Iteratively update ***B*** and ***C*** to remove 1s from them

- Fix ***B***, update ***C***; fix ***C***, update ***B***; etc.

- The problem is (still) NP-hard – we use a heuristic

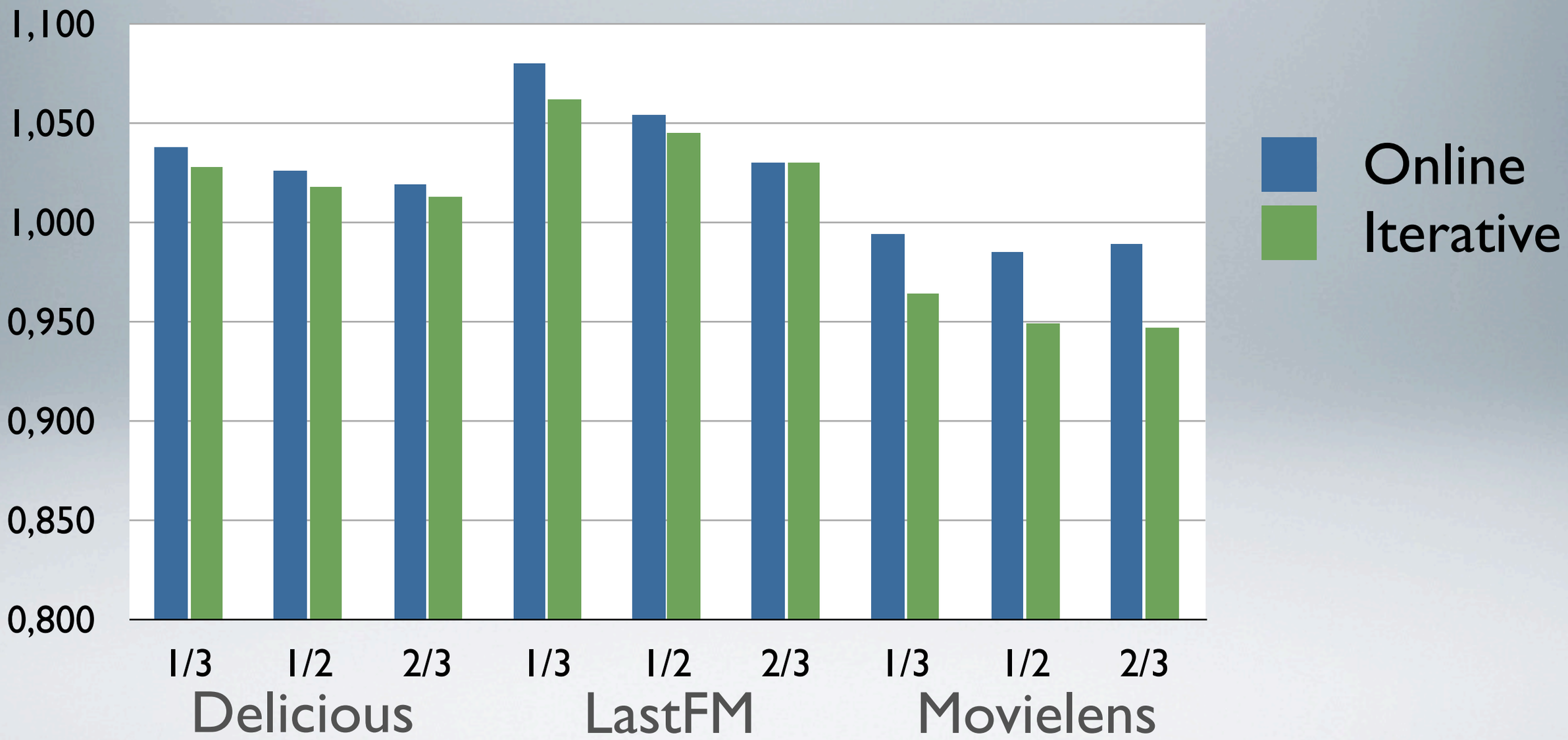
- Computationally expensive



ERROR OVER TIME

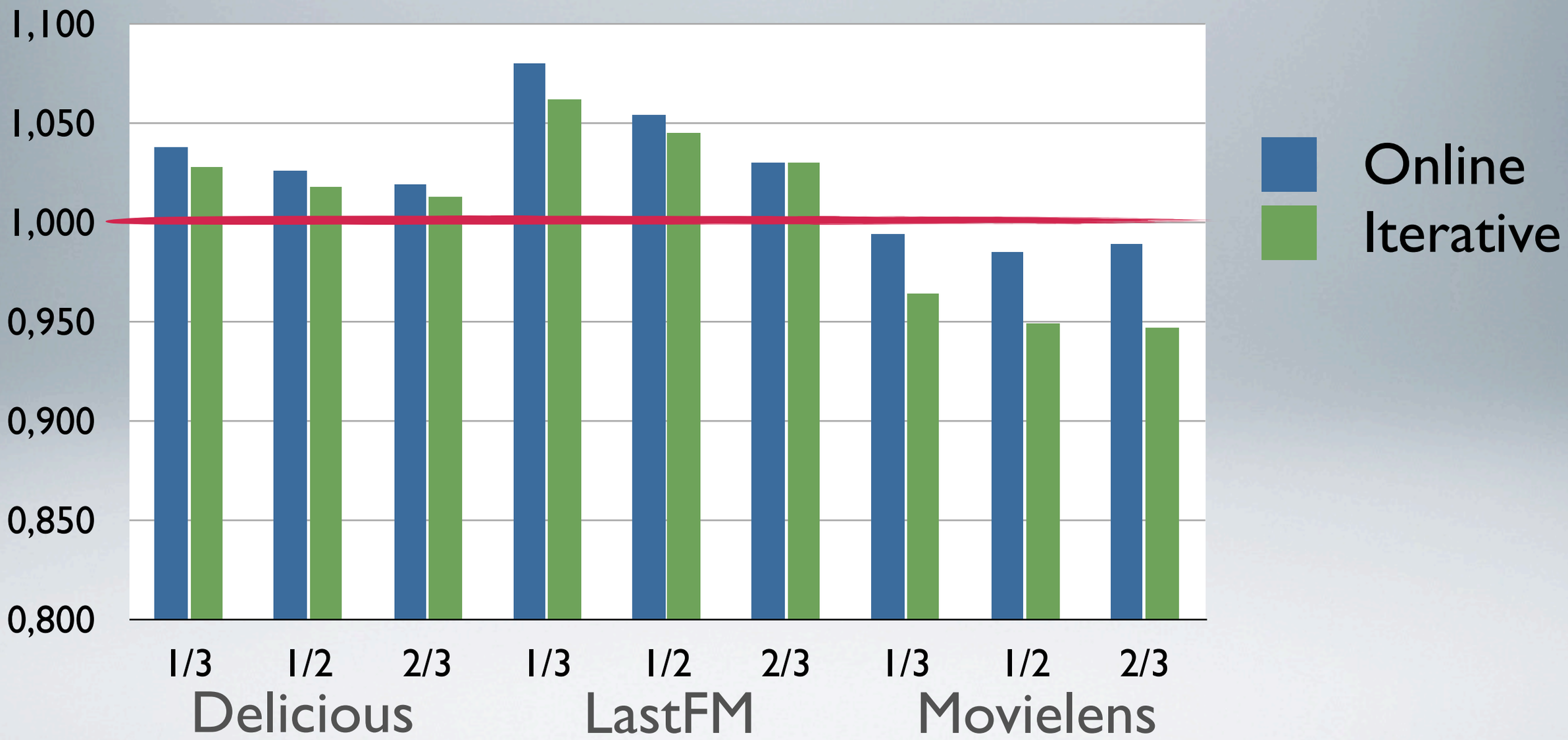
max planck institut informatik

COMPETITIVE FACTOR



max planck institut informatik

COMPETITIVE FACTOR



max planck institut informatik

TIME COMPLEXITY

	Sequence length			
Online?	561K	420K	281K	All
Yes	95	64	33	97
No	1045	1111	1097	

FUTURE WORK

- Adjusting the rank
- Better data structures and analysis
- Paralellization
- Tweaking the base algorithm for better prediction

max planck institut
informatik

CONCLUSIONS

- Dynamically updating a Boolean matrix factorization is possible
- Simple idea performs very well and is reasonably fast
  - Can be better *and* faster than running the off-line algorithm

CONCLUSIONS

- Dynamically updating a Boolean matrix factorization is possible
- Simple idea performs very well and is reasonably fast
- Can be better *and* faster than running the off-line algorithm

Thank You!