Constant Memory Sampling

- Several orders of magnitude speedup compared to naive sampling
- But what if H does not fit into memory?

Need a constant-memory sequential sampling algorithm!

constant-memory sequential sampling algorithm

Algorithm Layered Constant-Memory Sequential Sampling. Input: n (number of desired hypotheses), ε and δ (approximation and confidence parameters), $b > n$ size of hypothesis buffer. **Output:** n approximately best hypotheses (with confidence $1 - \delta$).

1. **Let** d_{max} the smallest number such that $cap(d_{max}, b, n) \geq |H|$
2. **Let** $C^{(d)} := \emptyset$ for all $d \in \{1, \ldots, d_{max}\}$
3. **Let** $FreeMemory := b$
4. **While** $H \neq \emptyset$
   (a) **While** $FreeMemory \geq 0$
      i. **Let** $B := GEN(FreeMemory, H)$.
      ii. **Let** $C^{(1)} := C^{(1)} \cup GSS(n, \frac{\varepsilon}{d_{max}}, \delta(B), B)$.
      iii. **Let** $FreeMemory := FreeMemory - min(|B|, n)$
   (b) **Let** $d := 1$
   (c) **While** $FreeMemory = 0$
      i. **If** $C^{(d)} \neq \emptyset$ **Then**
         A. **Let** $C^{(d+1)} := C^{(d+1)} \cup GSS(n, \frac{\varepsilon}{d_{max}}, \delta(C^{(d)}), C^{(d)})$.
         B. **Let** $C^{(d)} := \emptyset$
         C. **Let** $FreeMemory := FreeMemory + |C^{(d)}| - n$
      ii. **Let** $d := d + 1$
5. **Let** $d := 1$
6. **While** $\exists d' > d : C^{(d')} \neq \emptyset$
   (a) **If** $C^{(d)} \neq \emptyset$ **Then**
      i. **Let** $C^{(d+1)} := C^{(d+1)} \cup GSS(n, \frac{\varepsilon}{d_{max}}, \delta(C^{(d)}), C^{(d)})$.
   (b) **Let** $d := d + 1$
7. **Return** $GSS(n, \varepsilon_{i+1}, \delta_{i+1}, C^{(d)})$.

Constant Memory Sampling Algorithm

Build layered cache.

Number of layers ~ log(hyp size / cache size).

While cache is not exhausted

Fill cache(0) with hypotheses, apply sequential

Sampling, load solutions into cache(1).

layer=1; While cache is exhausted

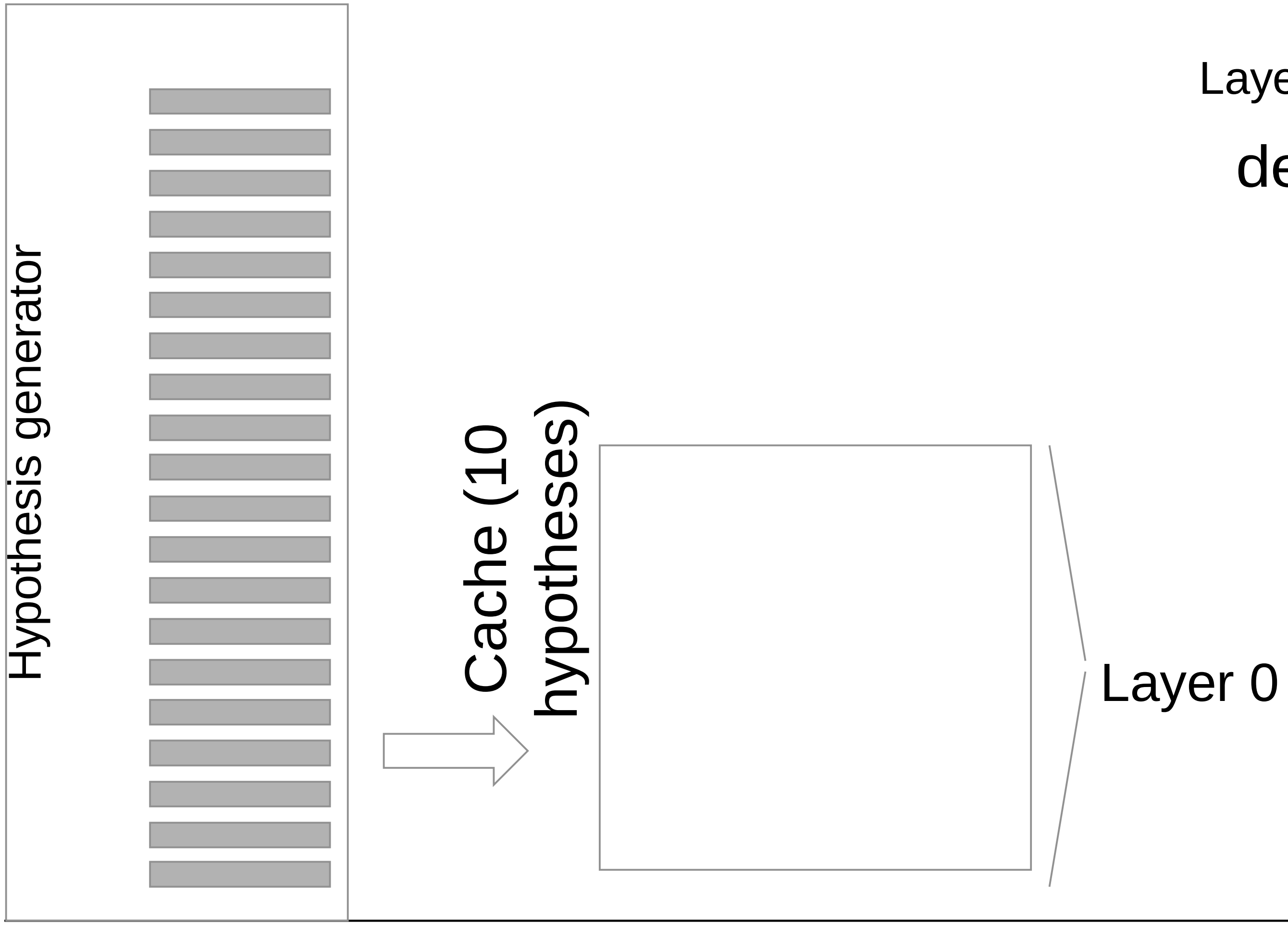
Apply sequential sampling to layer (i), load

Solutions to next layer, increment layer.

Constant Memory Sampling LCM-GSS

Layered Cache:

$$\text{depth} \approx \log\left(\frac{|H|}{|\text{Cache}| - n}\right)$$

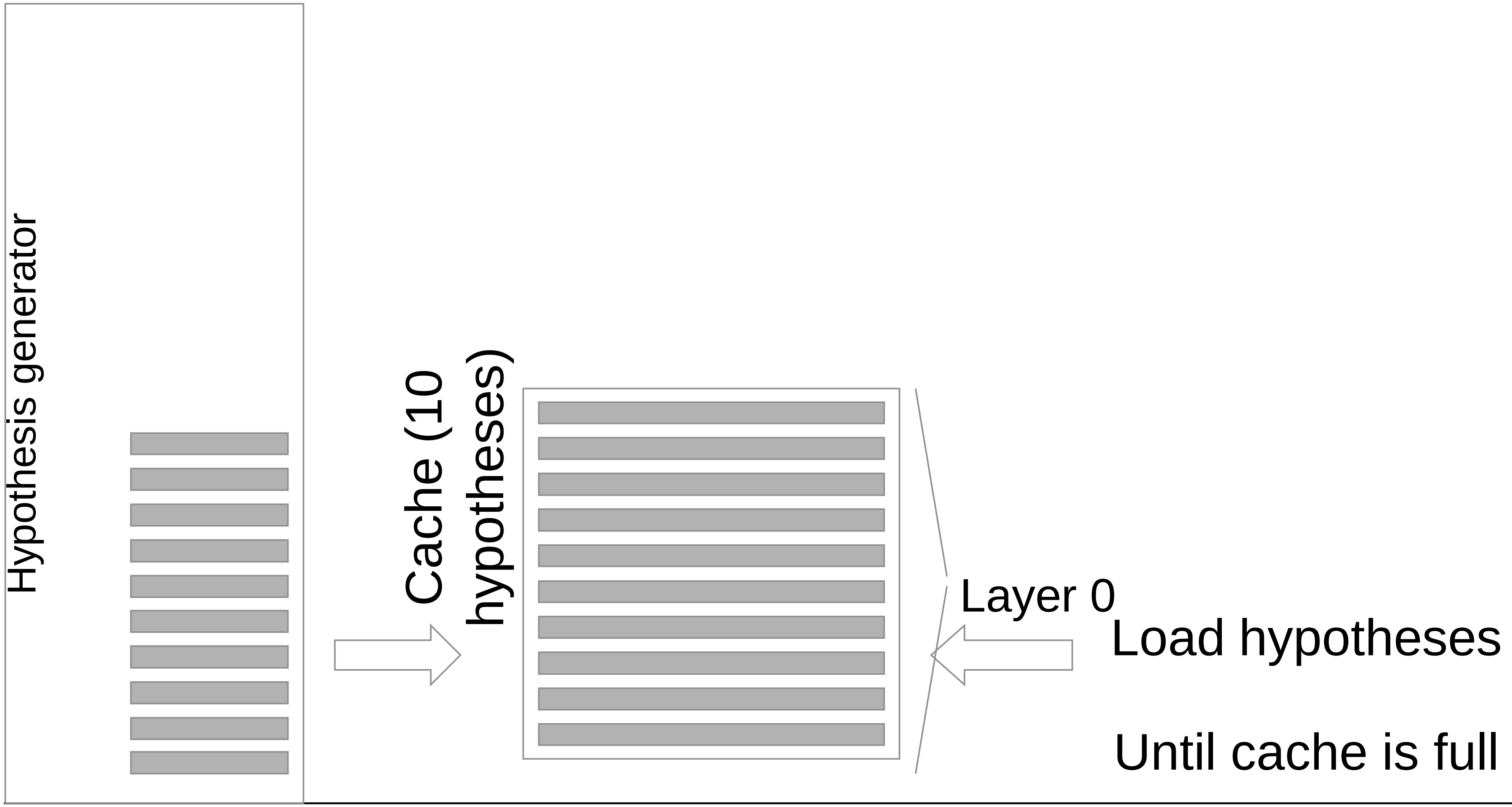


universität bonn

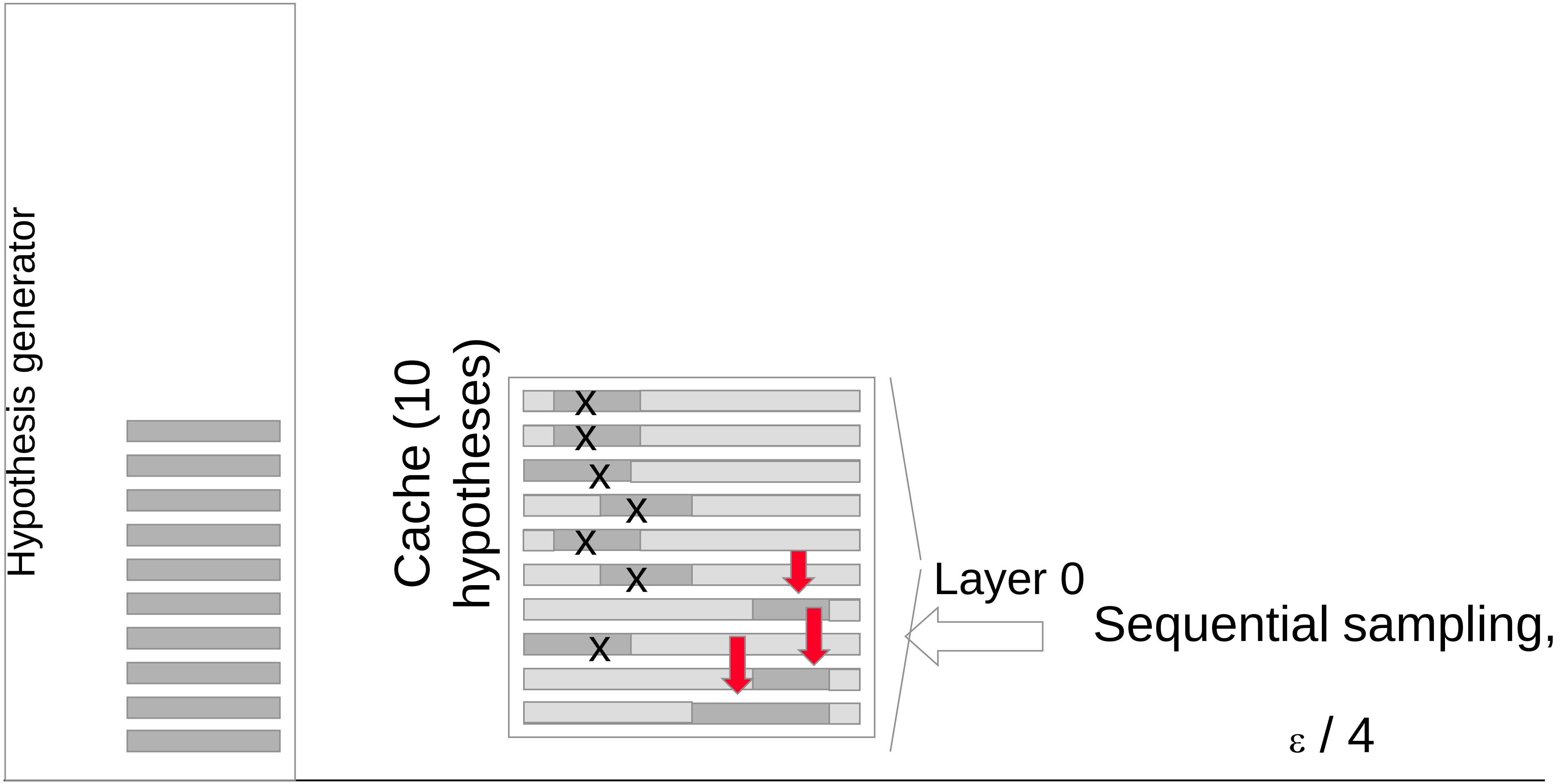
RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT

Fraunhofer IAIS

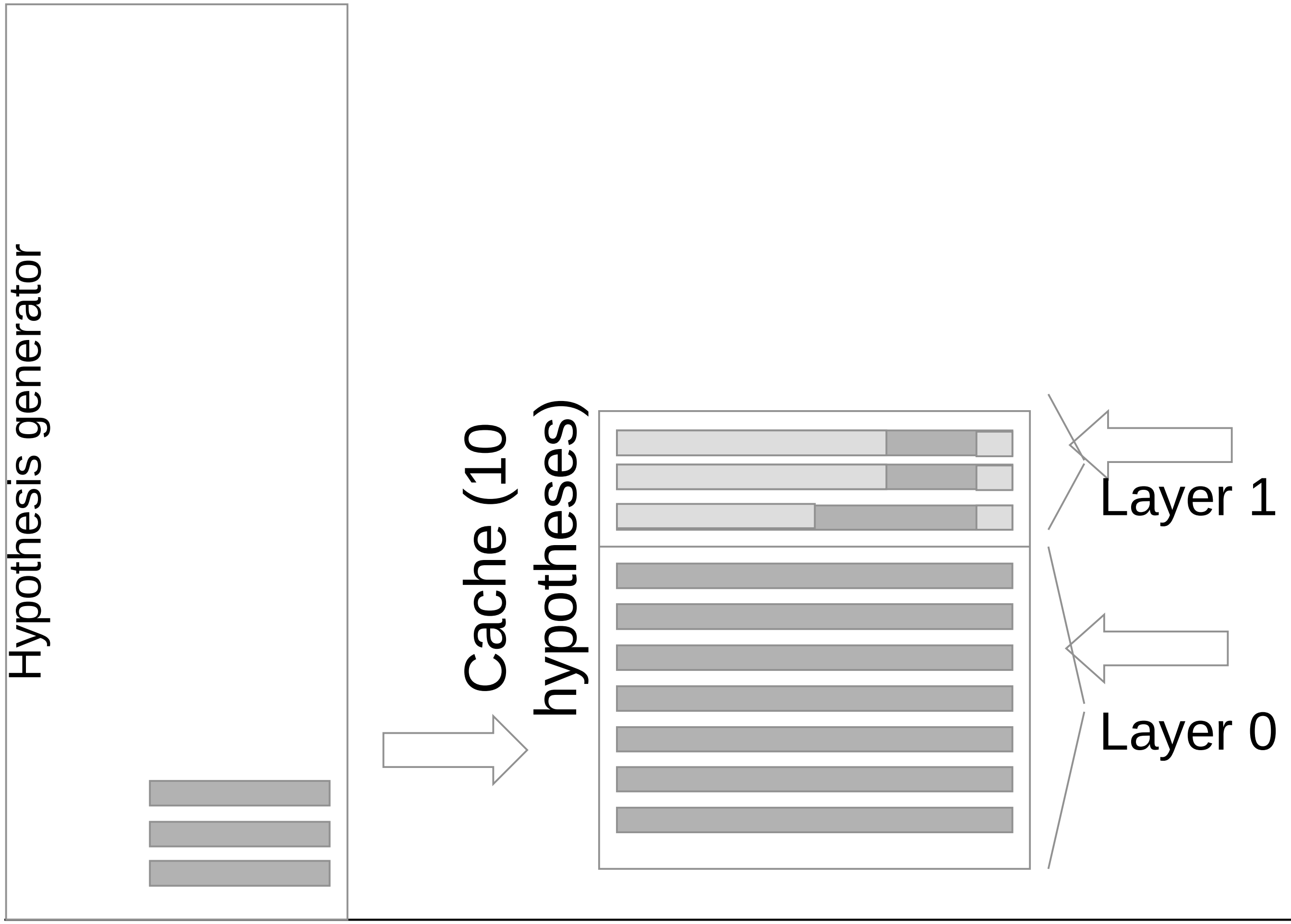
Constant Memory Sampling LCM-GSS



Constant Memory Sampling LCM-GSS



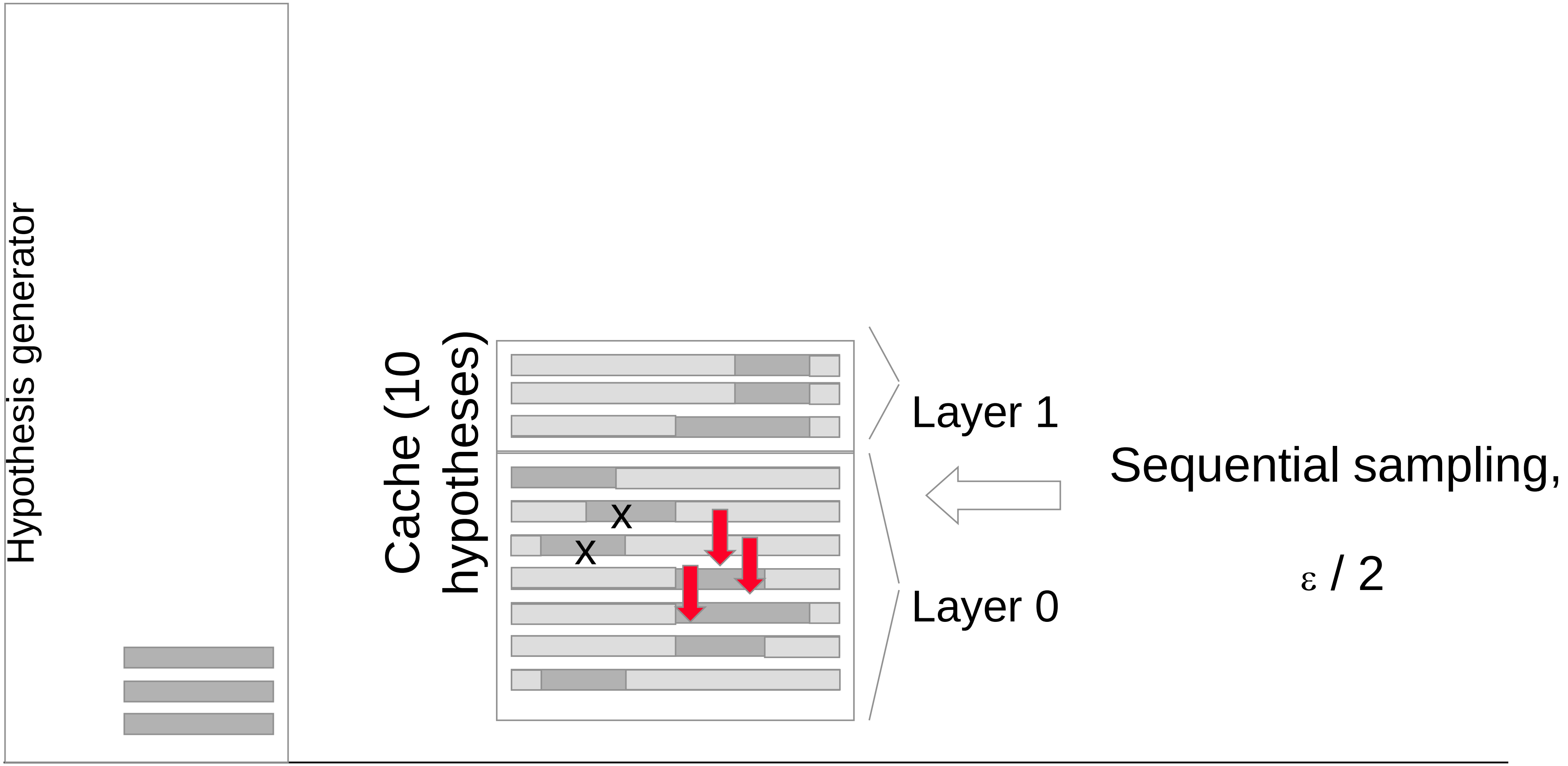
Constant Memory Sampling LCM-GSS



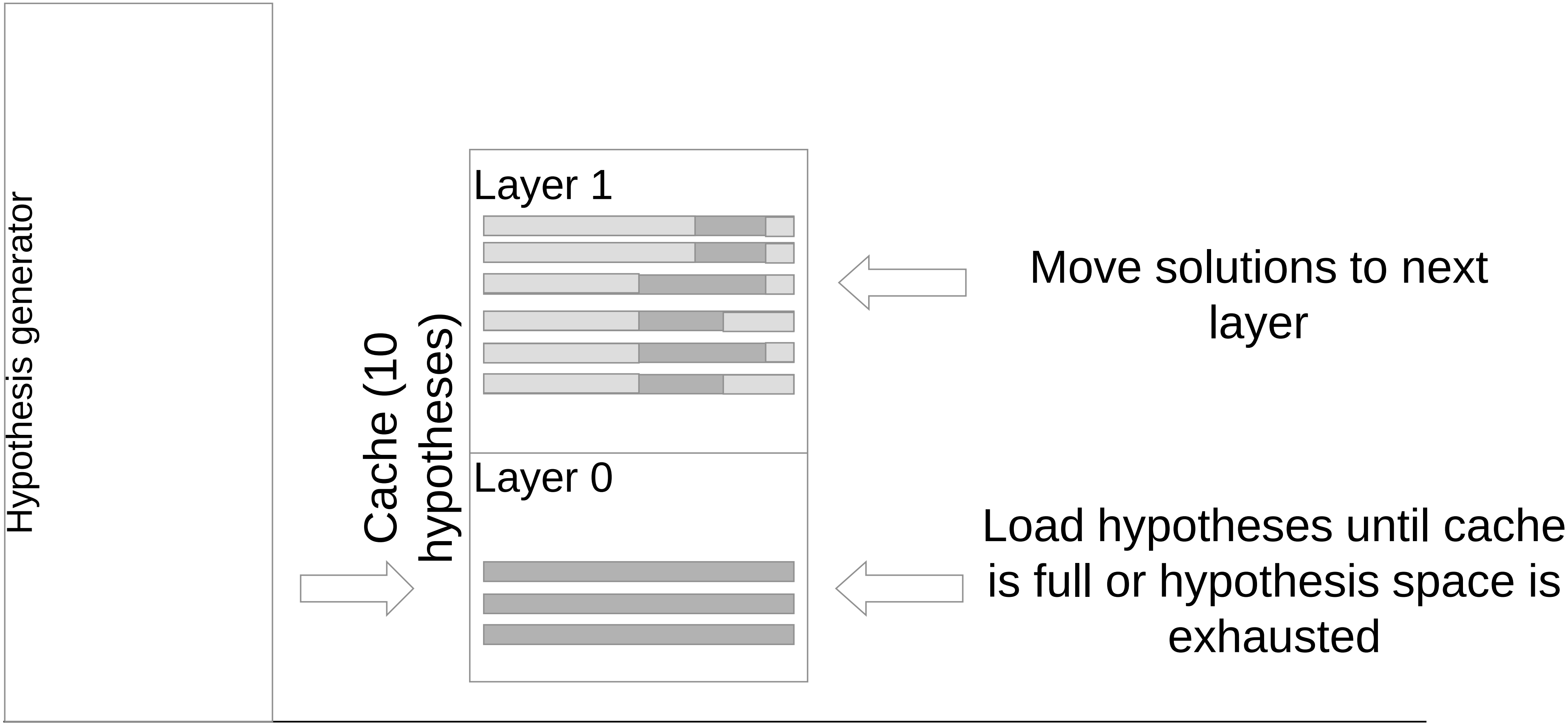
Move solutions to Next layer

Load hypotheses until cache is full

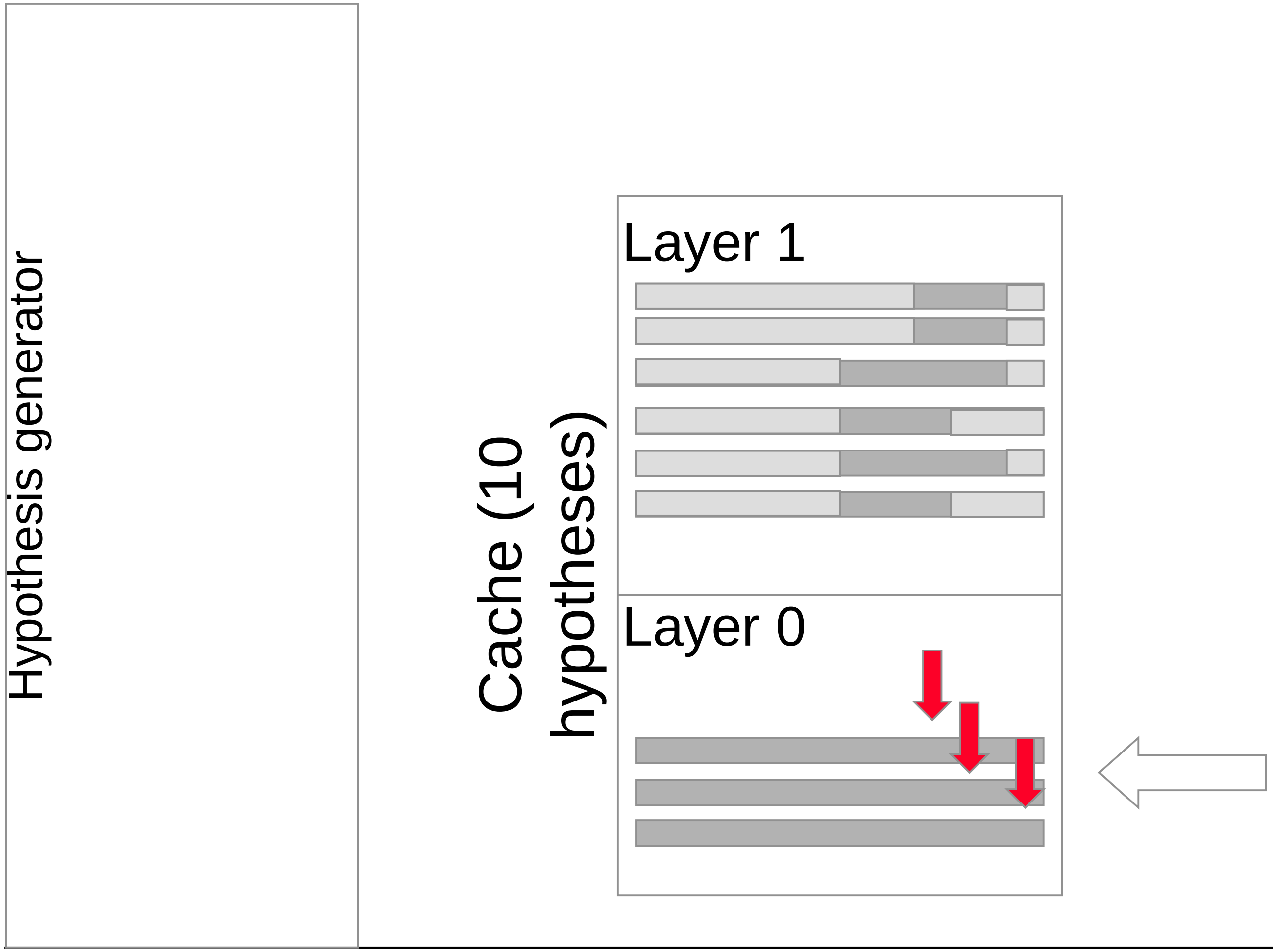
Constant Memory Sampling LCM-GSS



Constant Memory Sampling LCM-GSS

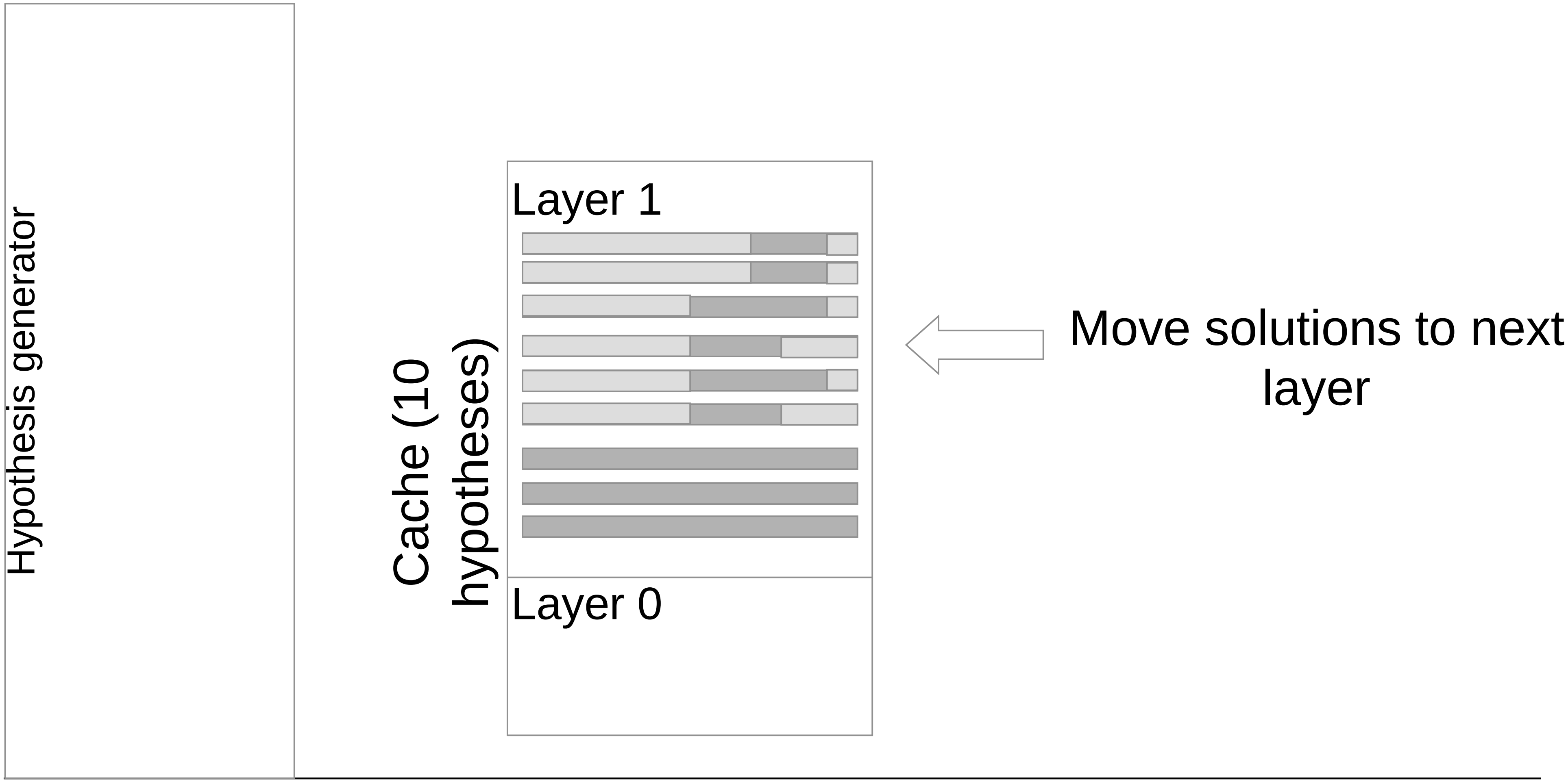


Constant Memory Sampling LCM-GSS



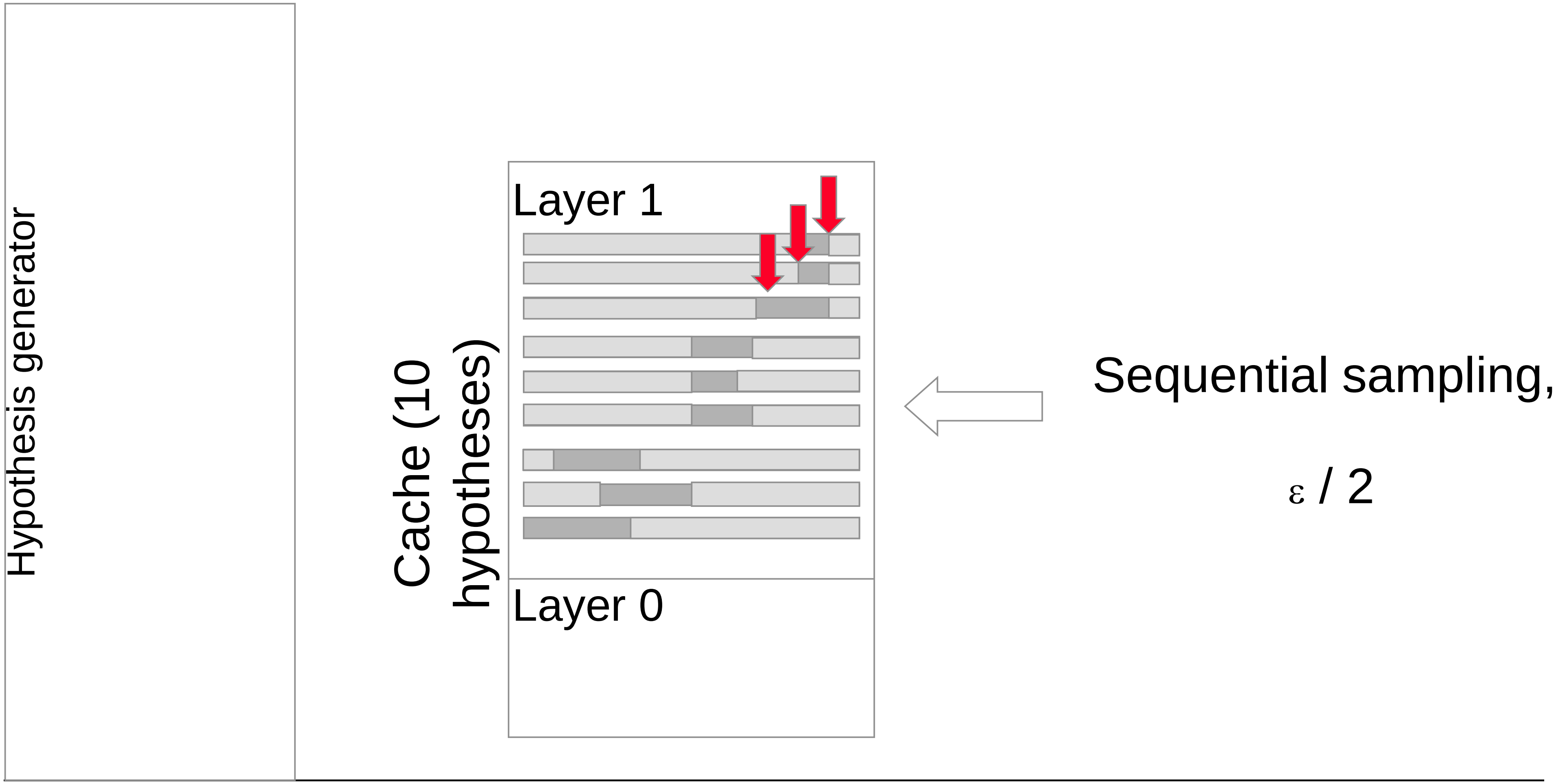
Only 3 hypotheses left, sequential sampling terminates immediately

Constant Memory Sampling LCM-GSS



universitätbonn
RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT
Fraunhofer
IAIS

Constant Memory Sampling LCM-GSS



Properties

- Approximate optimality guarantee of returned solution holds whenever cache size is > n.
- Hypothesis space can be arbitrarily large.
- Database can be infinitely large.
- Memory usage is limited to cache size.
- Number of layers: $$\text{depth} \approx \log\left(\frac{|H|}{|\text{Cache}| - n}\right)$$
- Required sample increases with $\frac{1}{layers^2}$

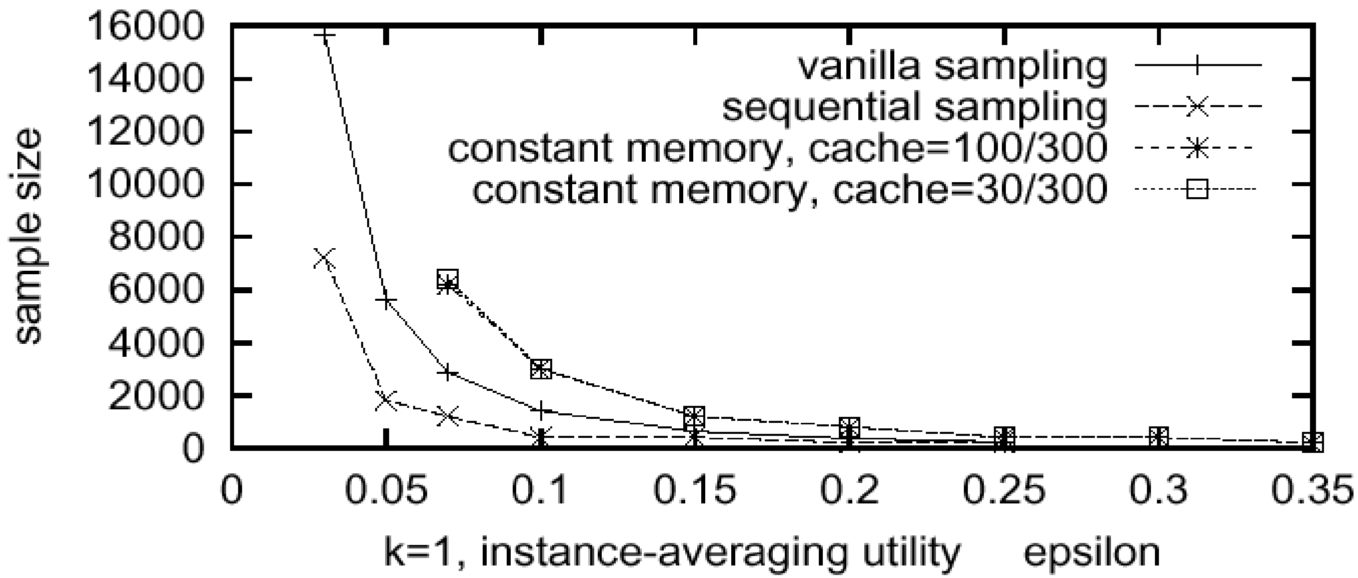
universitätbonn
RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT

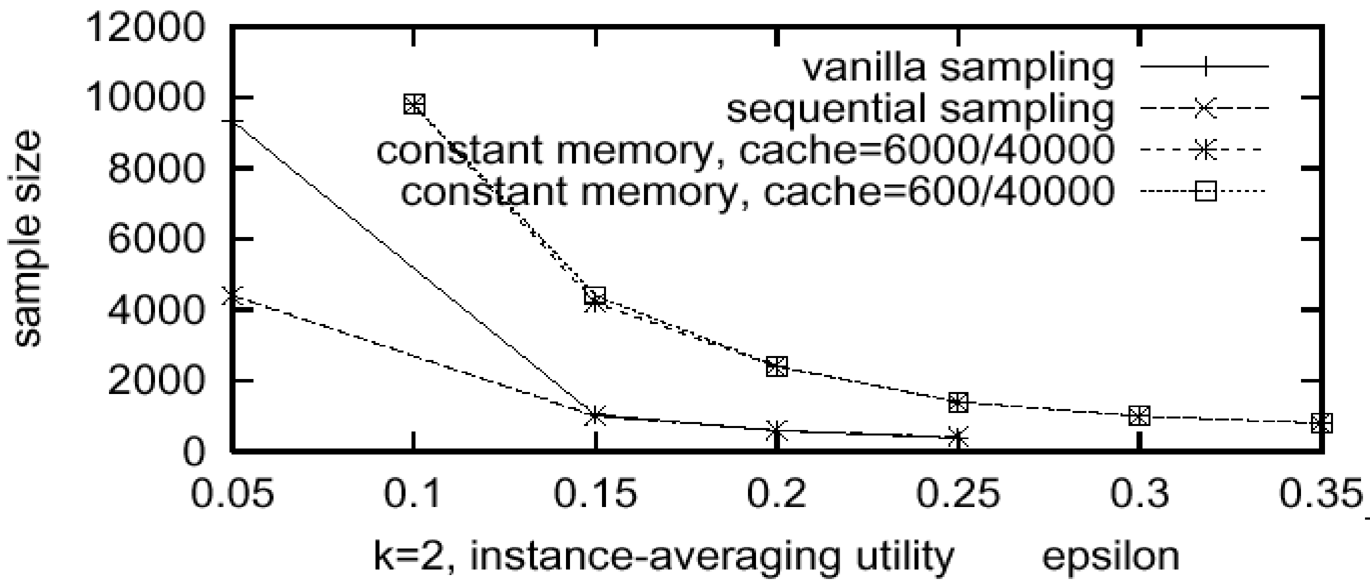
Fraunhofer
IAIS

Empirical Results

- Database with 14,000 juice purchase transactions.
- Simple subgroup discovery algorithm.
- Determine which groups of customers with unusual habits of buying non-recyclable or recyclable bottles exist.
- Comparison of
  - Naive sampling: data-independent sample bound.
  - Sequential sampling: data-dependent bound.
  - Constant memory sampling.
- Equal quality guarantees for all three algorithms.

Empirical Results





Conclusion

- Practical sampling algorithms handle very large databases.
- Stochastic optimality guarantees.
- Data-dependent, sequential sampling requires considerably smaller sample sizes.
- Constant-memory sampling handles large hypothesis spaces.
- Required sample increases with $1/(\log(\text{hypothesis space}/\text{cache}))^2$
- Algorithms are practical, applied to shopping transaction and KDD cup 98 database.

The main learning tasks

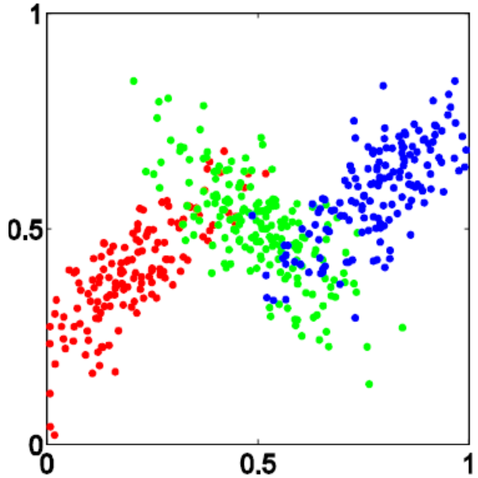
- Classification/prediction based on examples
  - decision trees
  - statistical regression
  - neural networks
  - k-nearest neighbor
  - support vector machines (SVMs)
  - (Naïve) Bayes
  - rule induction
  - ...
- Reinforcement learning
- dependency discovery: association rules, frequent pat
- deviation detection and summarization: subgroup disco
- Clustering
  - k-means/EM clustering
  - Bayesian clustering
  - HAC
  - ...

Orthogonal questions:
- Scalability
- Human interaction

N.B. specialized techniques to perform these tasks for text, spatial, audio, video data!

universitätbonn
RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT
Fraunhofer IAIS

Clustering



Prof. Dr. Stefan Wrobel

University of Bonn and Fraunhofer IAIS

universität bonn

RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT

Fraunhofer IAIS

Sample Application: Clustering

Customer segmentation:

- A company wants to increase sales by offering bundled “packages”
  - e. g. stereo systems
- each bundled package should attract a group of customers, something for everyone
- not too many packages

➢ group customers into clusters of people with similar profile

Applications of Clustering

Business: e. g.

- customer segmentation
- micromarket identification

Science: e. g.

- biological taxonomies
- clustering of star observations

Engineering:

- grouping of process states

Generally: compression of data, preprocessing

Task Definition

Given:

- a set of instances $I = \{x_1, ..., x_n\}$ from an instance space X
- a quality measure q on sets of sets of instances

$$q : 2^{2^X} \rightarrow \Re$$

Find:

- a set $C = \{C_1, C_2, ..., C_k\}$ of *clusters* where $C_i \subseteq X$ for all $i \in \{1, ..., k\}$ and $\bigcup_{i=1...k} C_i \supseteq I$
- q(C) maximal

N. B. Often, we require C to be a partition:

$$C_i \cap C_j = \varnothing \qquad \forall i \neq j \notin \{1 \ldots k\}$$

Quality measures

"Maximize intra-cluster similarity,

minimize inter-cluster similarity"

Simple instantiation

- assume distance measure dist: $X \times X \rightarrow \Re^+$
- often: Euclidean (geometric) distance if $X = \Re^n$
- then define, for $C_1 \in C$, $C_2 \in C$:

  $\text{dist}\ (C_1) := f\ (\{\text{dist}\ (x_1, x_2) | x_1 \neq x_2 \in C_1\})$

  and f is e.g. min, max, average;

  $\text{dist}\ (C_1, C_2) := f\ (\{\text{dist}\ (x_1, x_2) | x_1 \in C_1, x_2 \in C_2\})$

  and f is e.g. min, max, average.

Partitional Clustering: *k*-means

Task:

- assume X is $\Re^d$ (d-dimensional metric space)
- distance is Euclidean
  - This means that centers/mean points can be computed
- quality criterion square-error
- assume size of clustering |C| = k given
- clusters may not overlap: partitioning

universitätbonn

RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT

Fraunhofer IAIS

Square-Error Cluster Quality

Euclidean distance:

$$\operatorname{dist}(x,y) = \sqrt{(x-y)^2} = \sqrt{\sum_{j=1}^{d} (x[j]-y[j])^2}$$

Cluster mean (center) of cluster C

$$m(C) := \frac{1}{|C|} * \sum_{x \in C} x = \left(\frac{\sum_{x \in C} x[1]}{|C|}, \ldots, \frac{\sum_{x \in C} x[d]}{|C|} \right)$$

Cluster square error (within-cluster variation):

$$e^2(C) := \sum_{x \in C} \operatorname{dist}(x, m(C))^2 = \sum_{x \in C} (x - m(C))^2$$

Total square error of clustering **C**

$$E^2(\mathbf{C}) := \sum_{C \in \mathbf{C}} e^2(C)$$

Searching for good clusterings

Brute-force?

- 10 objects in 4 clusters
  - 34.105 possibilities

- 19 objects in 4 clusters
  - 11.259.660.00 possibilities!

➢ Must use “heuristic” search

- hill-climbing
- multiple hill-climbing

K-means algorithm

universitätbonn RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT Fraunhofer IAIS

General k-means type algorithm

- Select k cluster centers
- REPEAT
  - assign each instance to closest center (“E-Step”)
  - compute centers of the thus-formed clusters (“M-Step”)

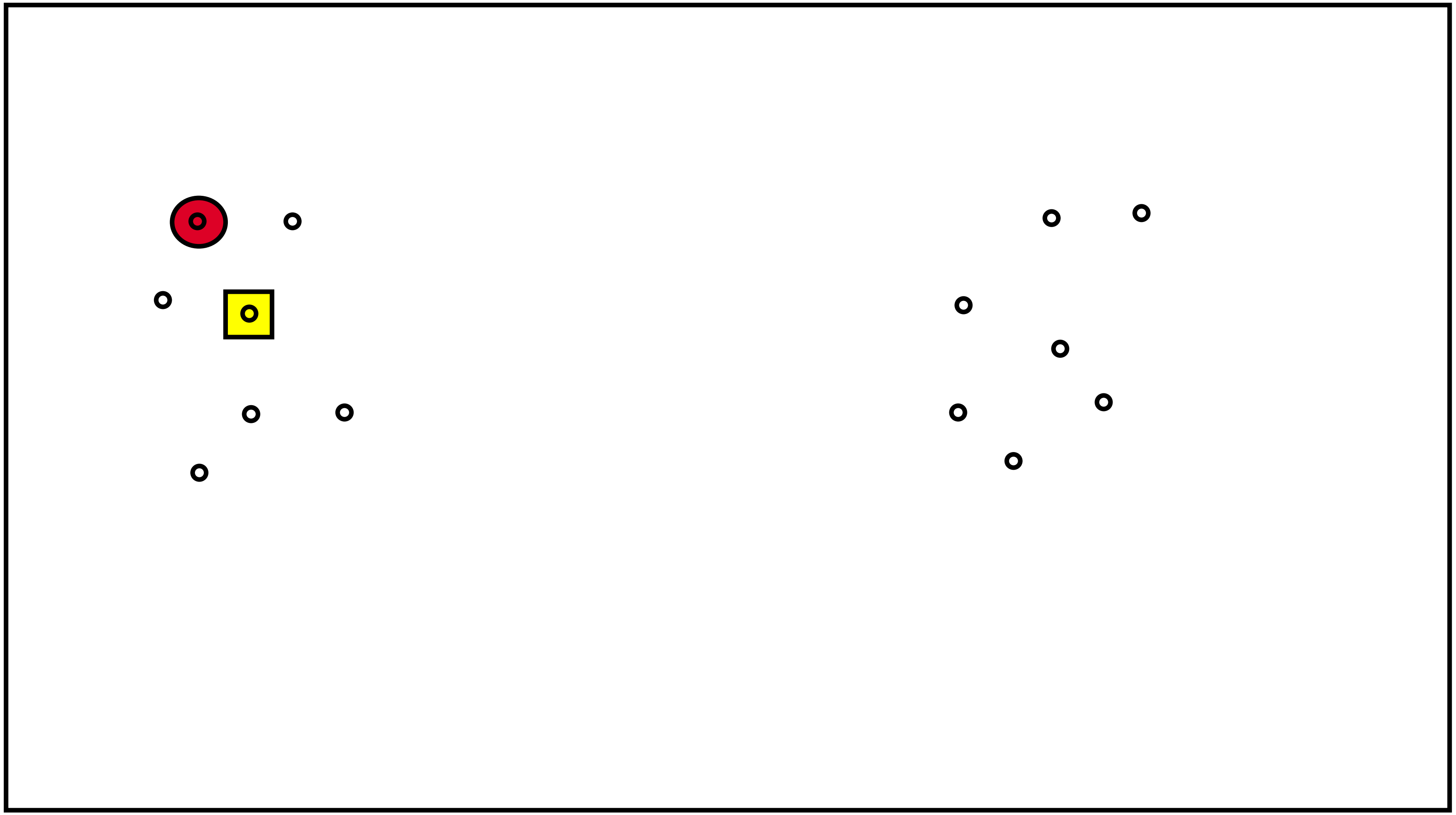
  UNTIL quality does not improve any more

Optional:

- adjust cluster number and restart at REPEAT

RETURN last clustering

Example (Initialization)



k=2

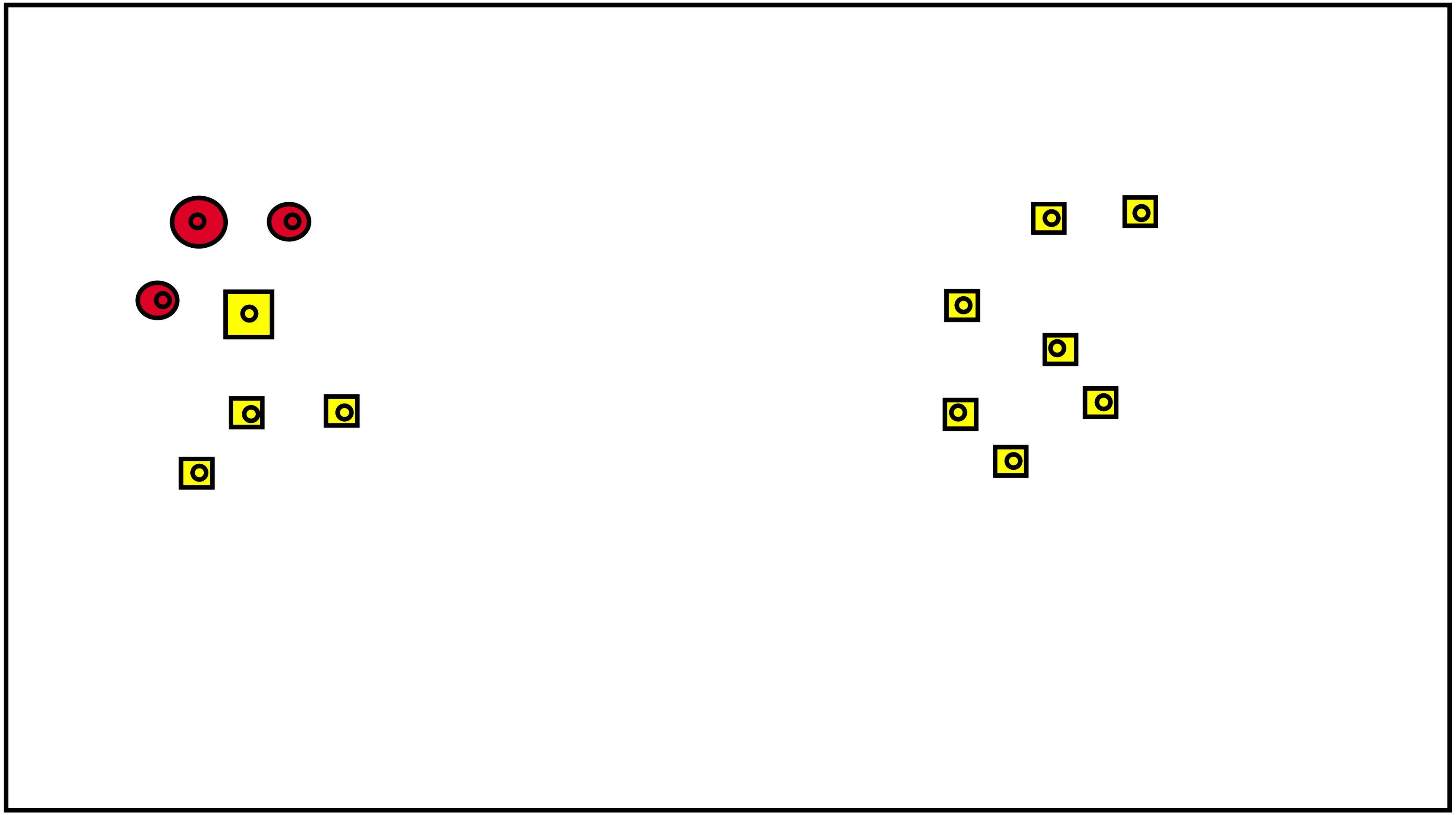
universitätbonn

RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT

Fraunhofer

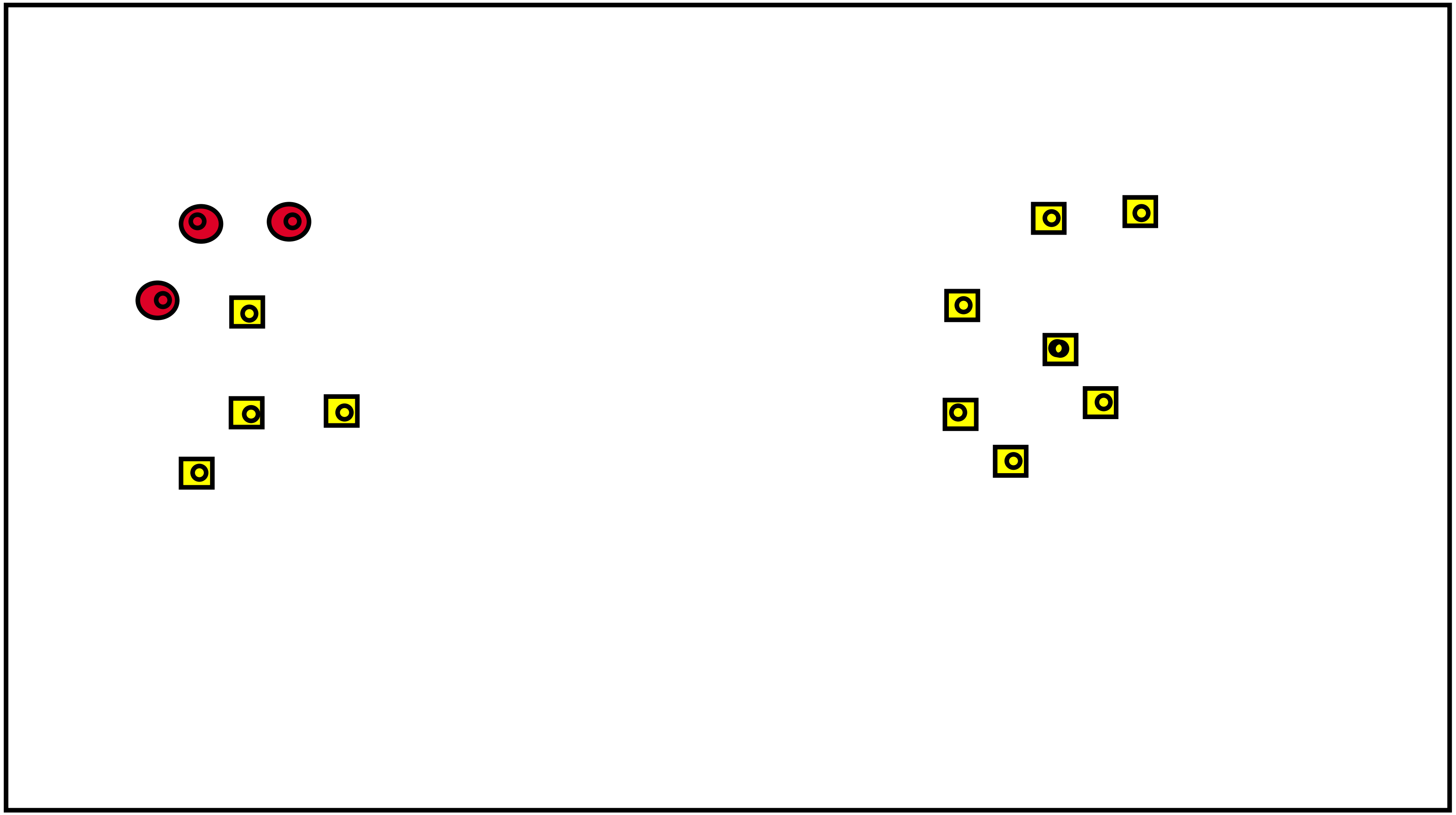
IAIS

Example (1. Iteration “E”)



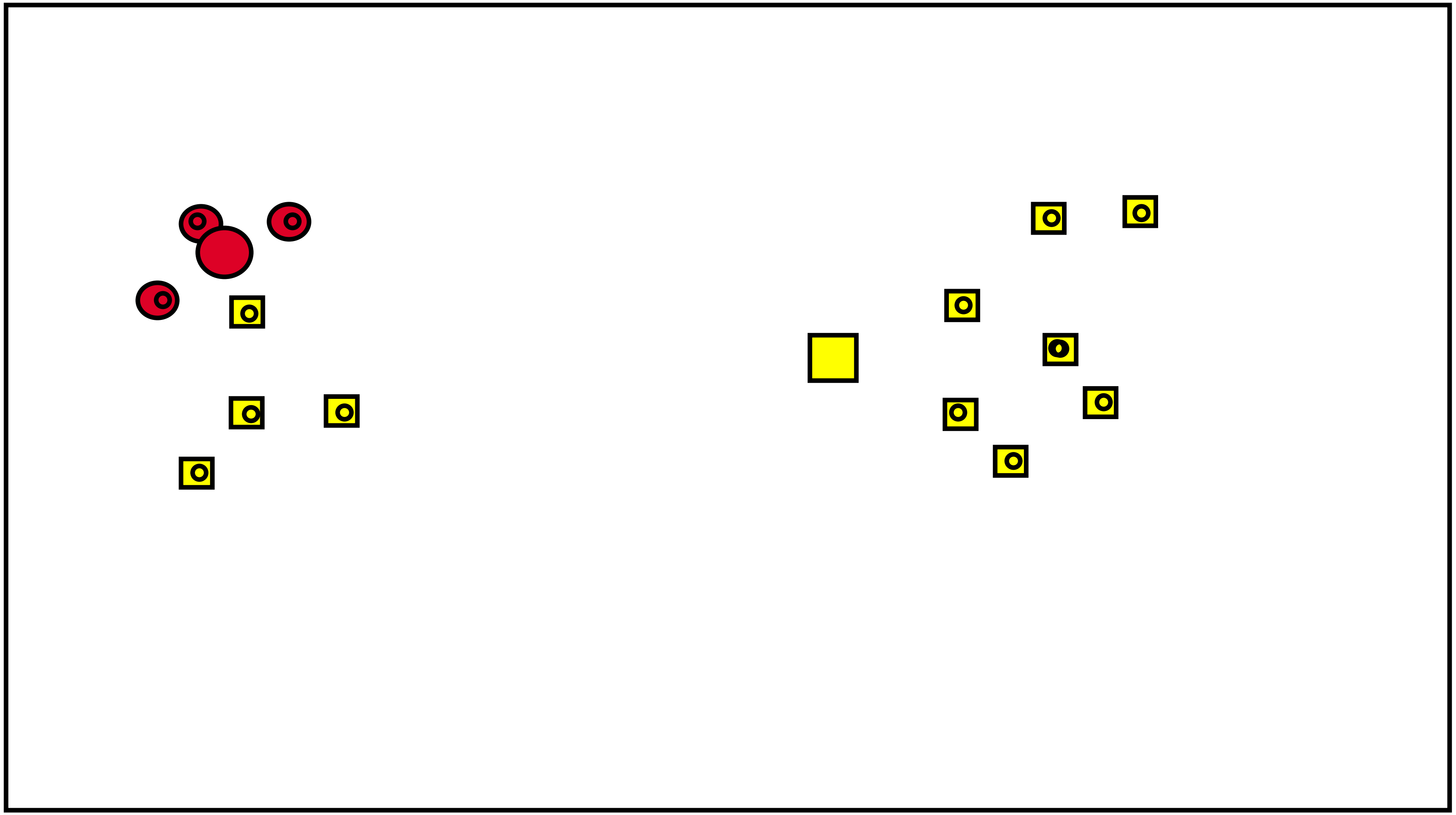
k=2

Example (1. Iteration “M”)



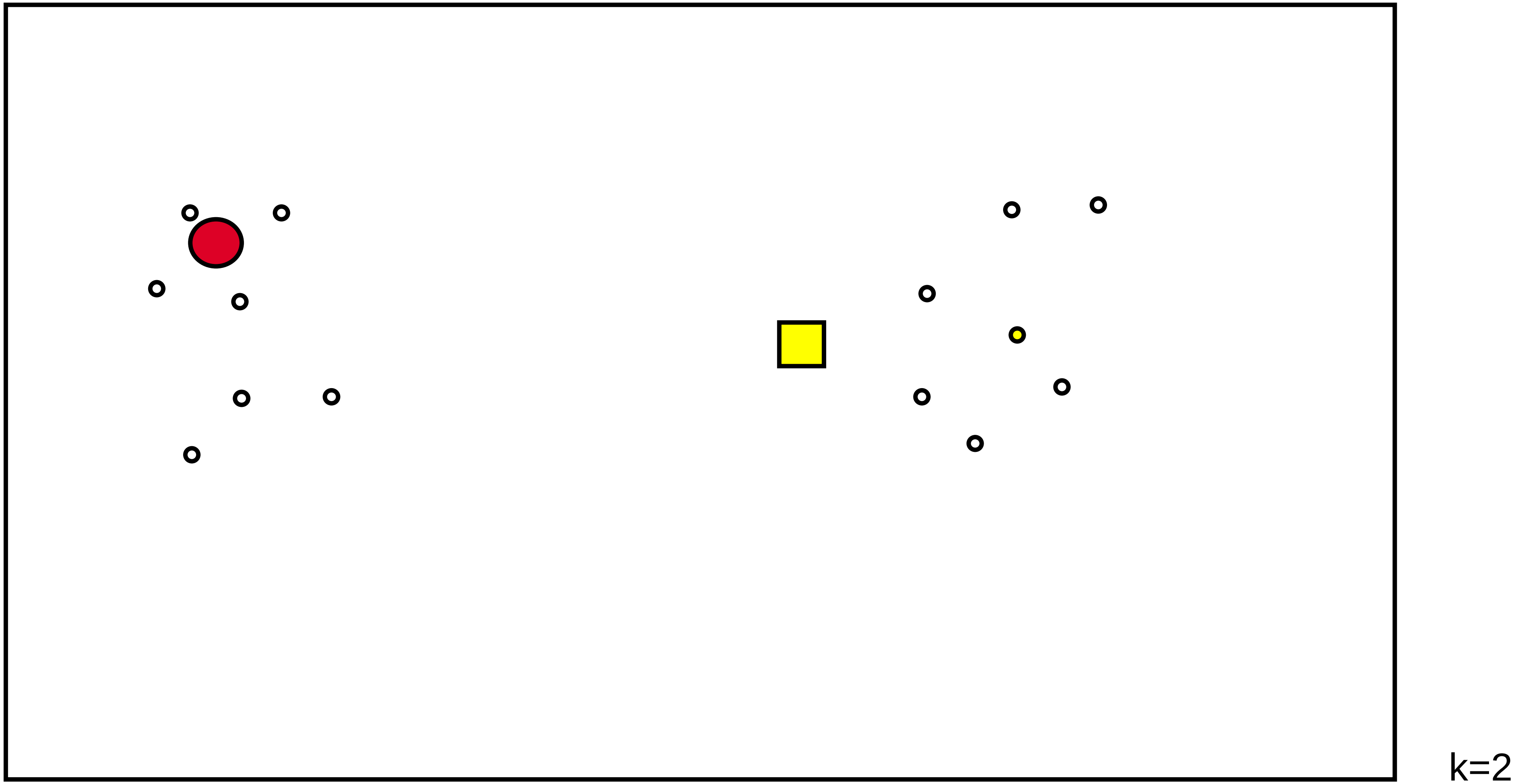
k=2

Example (1. Iteration “M”)

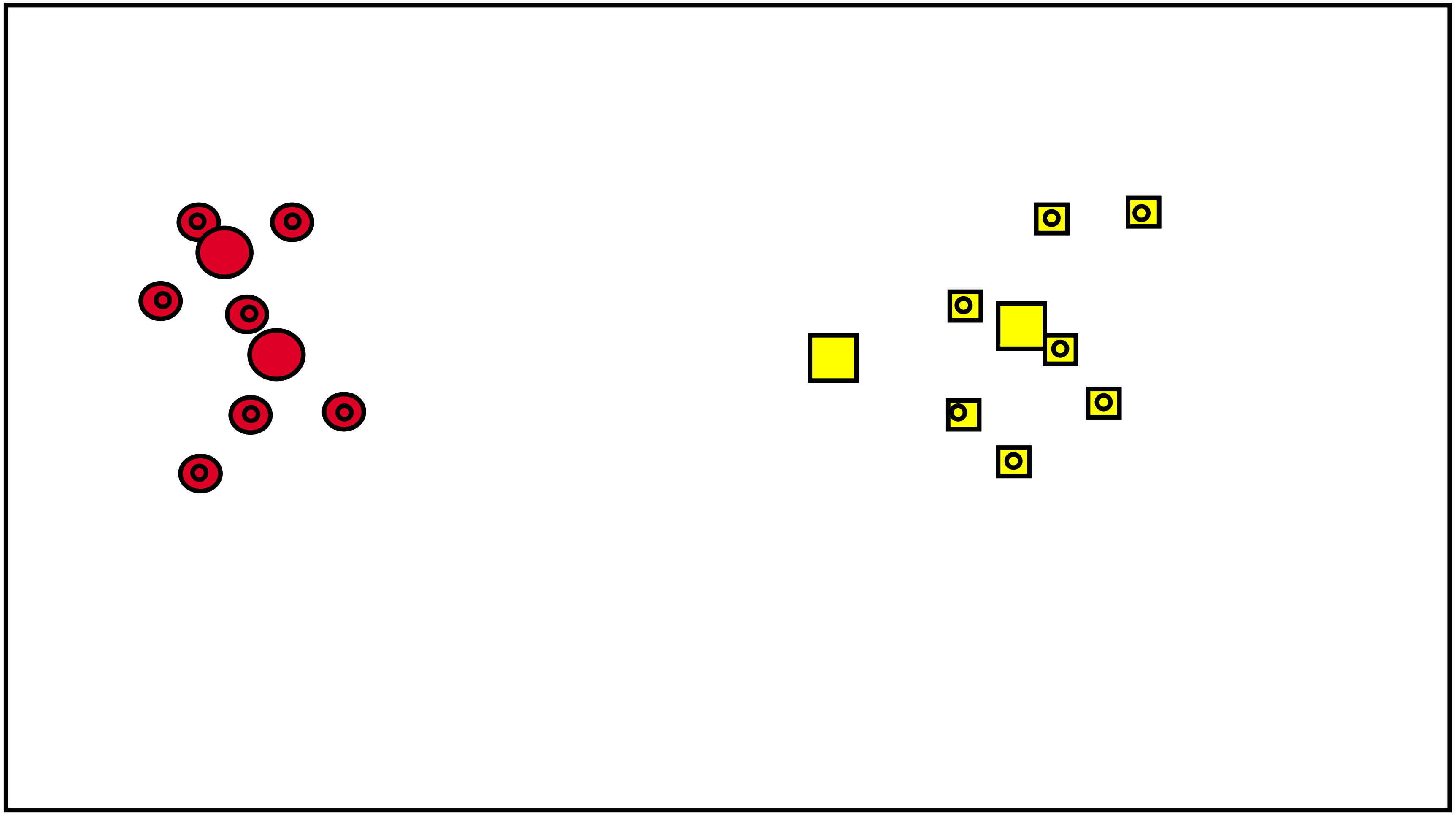


k=2

Example (1. Iteration “M”)



Example (2. Iteration)



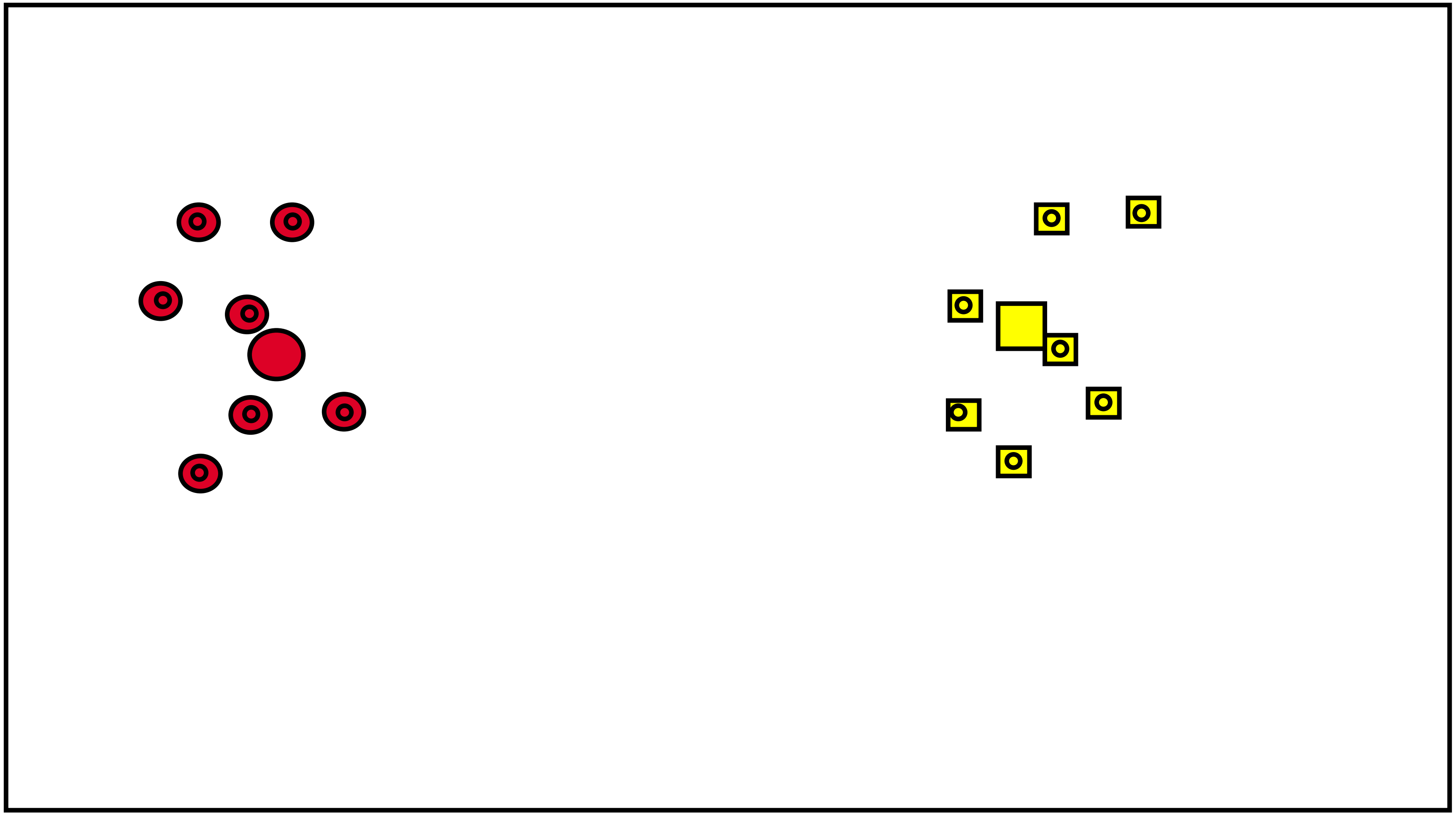
k=2

universitätbonn

RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT

Fraunhofer IAIS

Example (3. Iteration)



k=2

See also …

- http://www.rob.cs.tu-bs.de/content/04-teaching/06-interactive/Kmeans/Kmeans.html
  - Nice coloring, easy to see effect of initialization

- http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

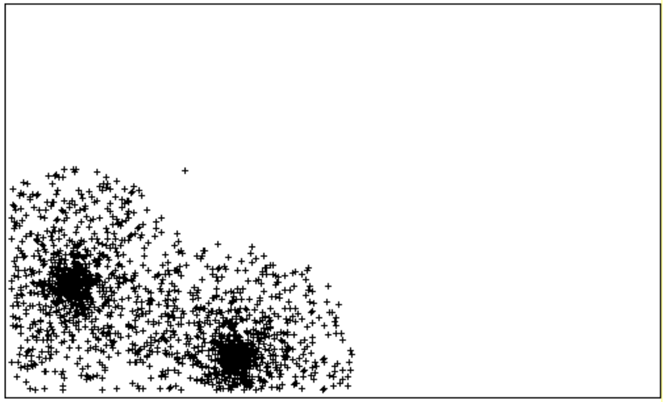
Nice application: image compression

- http://www.leet.it/home/lale/joomla/component/option,com_wrapper/Itemid,50/

Properties

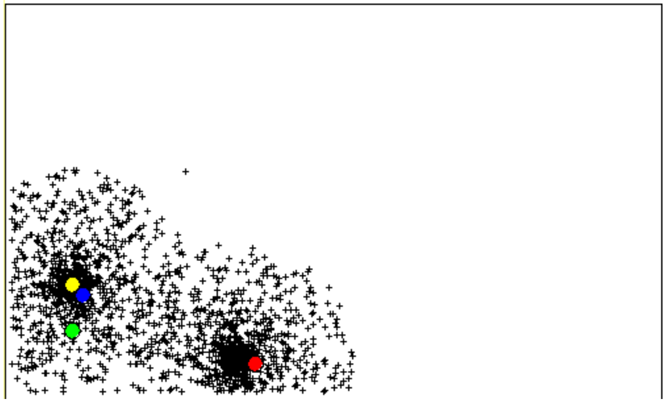
- Convergence - to a local minimum!
  - ➢ use non-local “jumps”
  - ➢ use multiple hill-climbing searches
- Result depends on initial “seeds”
  - ➢ choose systematically: e. g.
    Start with data centroid, add most distant
- distorted by outliers
  - ➢ preprocess or recognize in method

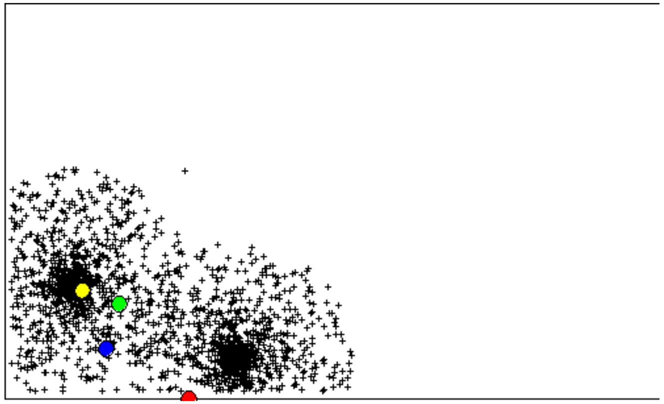
Dependence on starting points, local minima

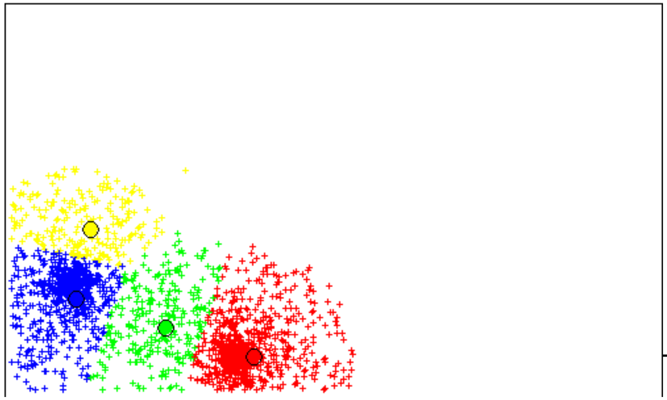


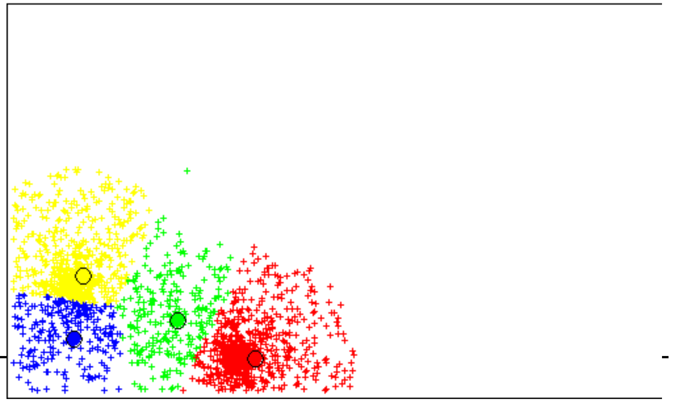
http://home.dei.polimi.it/matteucc/

Clustering/tutorial_html/AppletKM.html









Local optima may be suboptimal globally

- Image by Andrew Moore, CMU

Local optima may be suboptimal globally

- Image by Andrew Moore, CMU

Adjusting k

Search k = 2 ... MAX_k

- expensive!

Split/merge incrementally:

- split cluster if “large” and “high variance” on feature with largest spread
- join if two cluster centers “close”