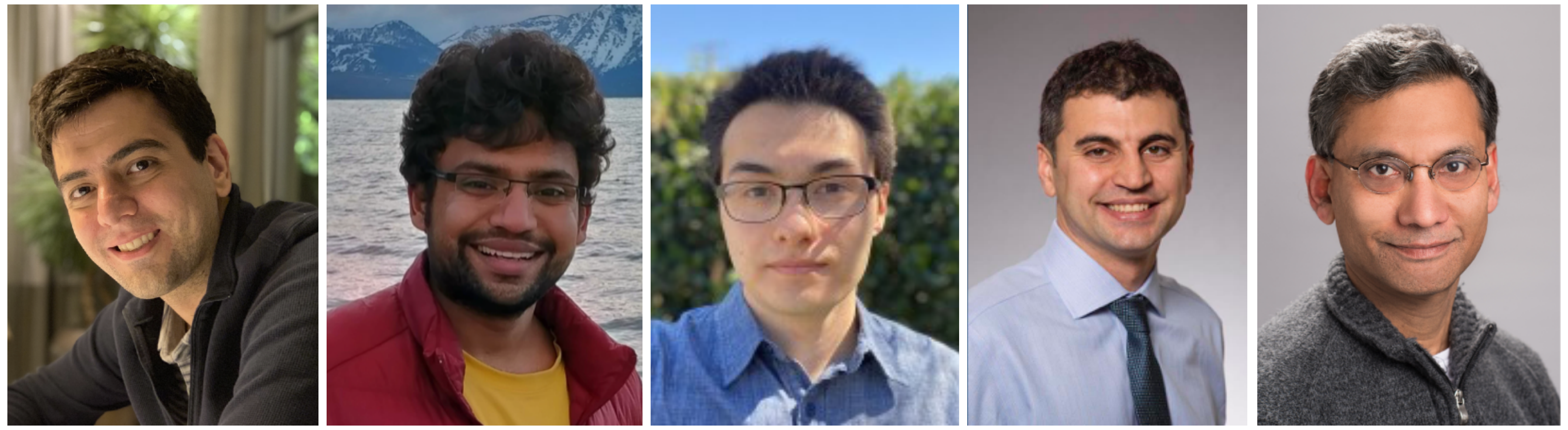
BEAR: sketching BFGS algorithm for ultra-high dimensional feature selection in sublinear memory

Amirali Aghazadeh
Vipul Gupta
Alex DeWeese
Ozan Koyluoglu
Kannan Ramchandran



Aug 16 - Aug 19th

Berkeley Laboratory for Information and System Sciences

big and high dimensional data in everyday life

- web services
- language processing
- networking
- genomics/proteomics
- health-care

- critical need for **scalable algorithms** to extract **important features** from the data
- limited computing **resource**

problem setup

>10^{15} !

- n data points $(\boldsymbol{\theta}_i)_{i=1}^n = (\mathbf{x}_i, y_i)_{i=1}^n$
  living in ultra-high dimensional feature space $\mathbf{x}_i \in \mathbb{R}^p \;\; (p \gg n)$
  goal: find a **small subset of features** best explains the output

- k-**sparse** feature vector $\boldsymbol{\beta}^* \in \mathbb{R}^p$
  loss function $f(\boldsymbol{\beta}, \boldsymbol{\theta}) : \mathbb{R}^p \to \mathbb{R}$

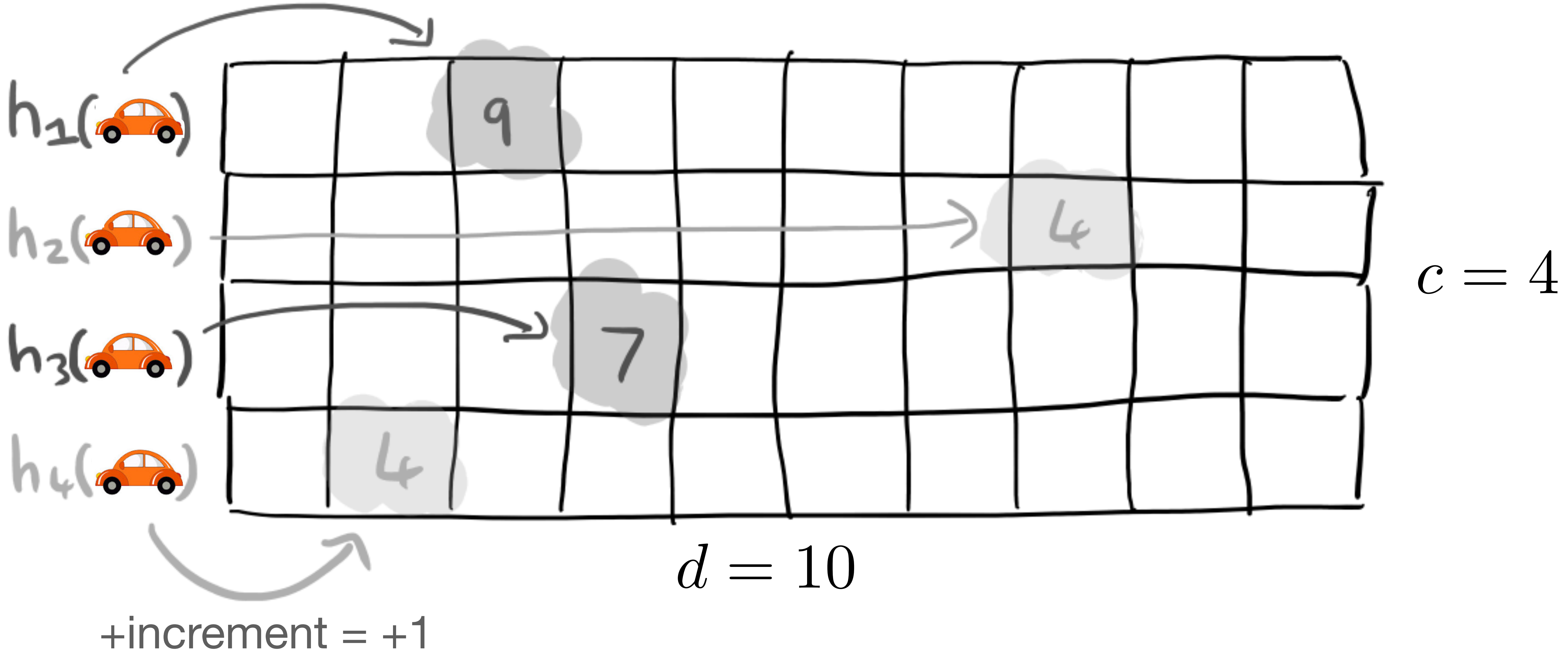
- optimization problem $\min_{\boldsymbol{\beta}} \sum_{i=1}^{n} f(\boldsymbol{\beta}; \boldsymbol{\theta}_i)$

- stochastic gradient descent (SGD) $\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \mathbf{g}(\boldsymbol{\beta}_t; \boldsymbol{\Theta}_t)$
  minibatch $\boldsymbol{\Theta}_t = \{\boldsymbol{\theta}_{t1}, \boldsymbol{\theta}_{2t}, \ldots, \boldsymbol{\theta}_{tb}\}$
  with the SGD term defined as $\mathbf{g}(\boldsymbol{\beta}_t; \boldsymbol{\Theta}_t) = \sum_{i=1}^{b} \nabla_{\boldsymbol{\beta}_t} f(\boldsymbol{\beta}_t; \boldsymbol{\theta}_{ti})$

challenge: not enough **memory** to store the **intermediately dense** feature vector $\boldsymbol{\beta}$ (sublinear alg.)

Count Sketch (CS)

random hash function $h_j : \{1, 2, \ldots, p\} \to \{1, 2, \ldots, d\}$



- data structure to **compressively** store the **number of occurrences** of many number of streaming items

- fast operations
- ADD (item, increment)
- QUERY (item)

$\#$  $\approx \text{median}(\{4, 4, 7, 9\})$

$\#\ \text{items}\ (p)$	# all colors
$m = d \times c$	memory of CS
$\#\ \text{frequent items}\ (k)$	# top colors

...

Count Sketch (CS)

random hash function $h_j : \{1, 2, \ldots, p\} \rightarrow \{1, 2, \ldots, d\}$

- data structure to **compressively** store the **number of occurrences**

Theorem 1 *Charikar et al. (2002) Count Sketch finds top-k items z_i with $\pm\varepsilon\|\mathbf{z}\|_2$ error, with probability at least $1-\delta$, in space $\mathcal{O}(\log(\frac{p}{\delta})(k + \frac{\|\mathbf{z}^{tail}\|_2^2}{(\varepsilon\zeta)^2}))$, where $\|\mathbf{z}^{tail}\|_2^2 = \sum_{i \notin top-k} z_i^2$ is the energy of the non-top-k items and ζ is the k^{th} largest value in $\mathbf{z}$.*

- ADD (item, increment)
- QUERY (item)

$\#$ items (p) — # all colors

$m = d \times c$ — memory of CS

$\#$ frequent items (k) — # top colors

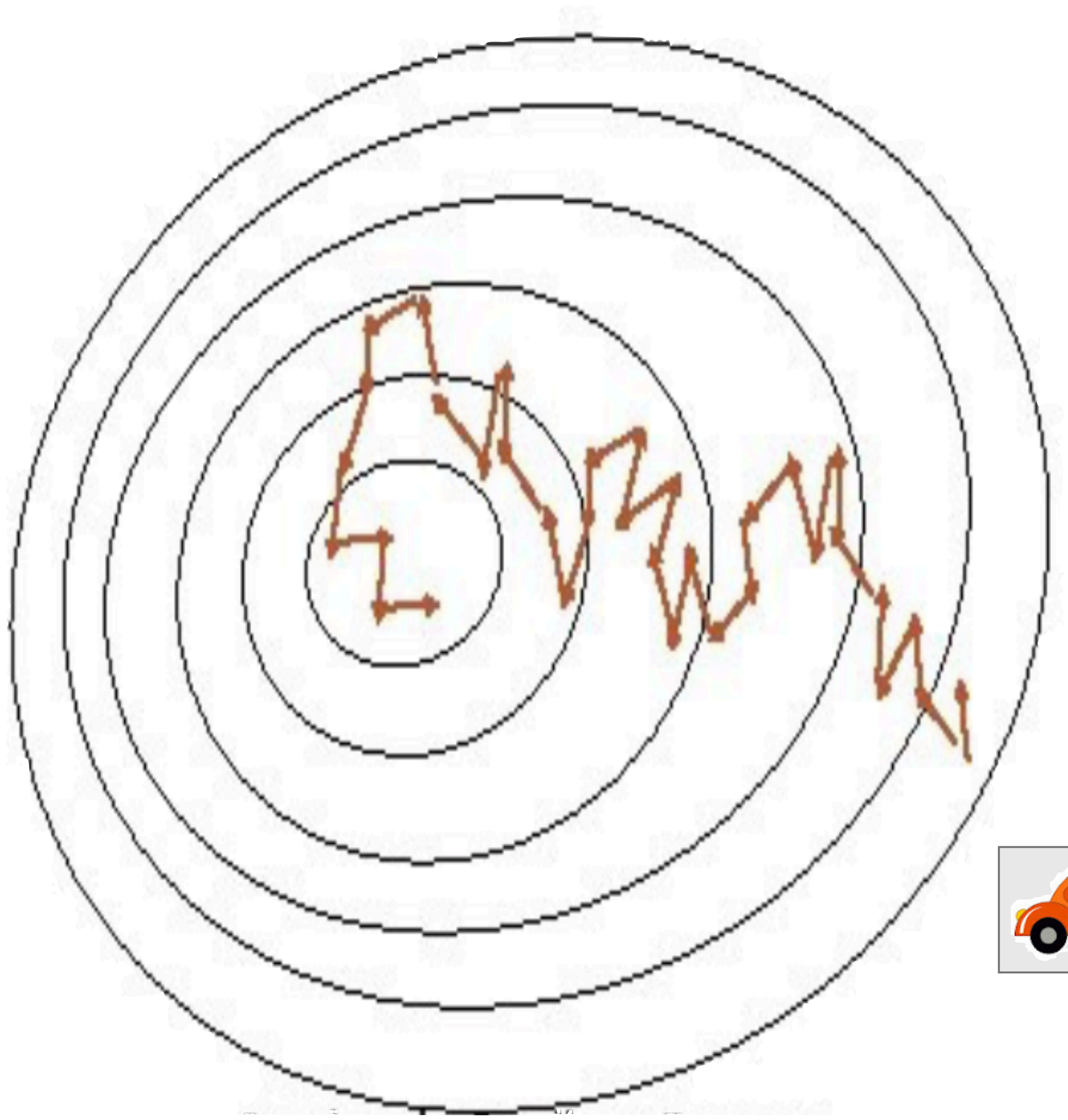
$\# \approx \text{median}(\{4, 4, 7, 9\})$

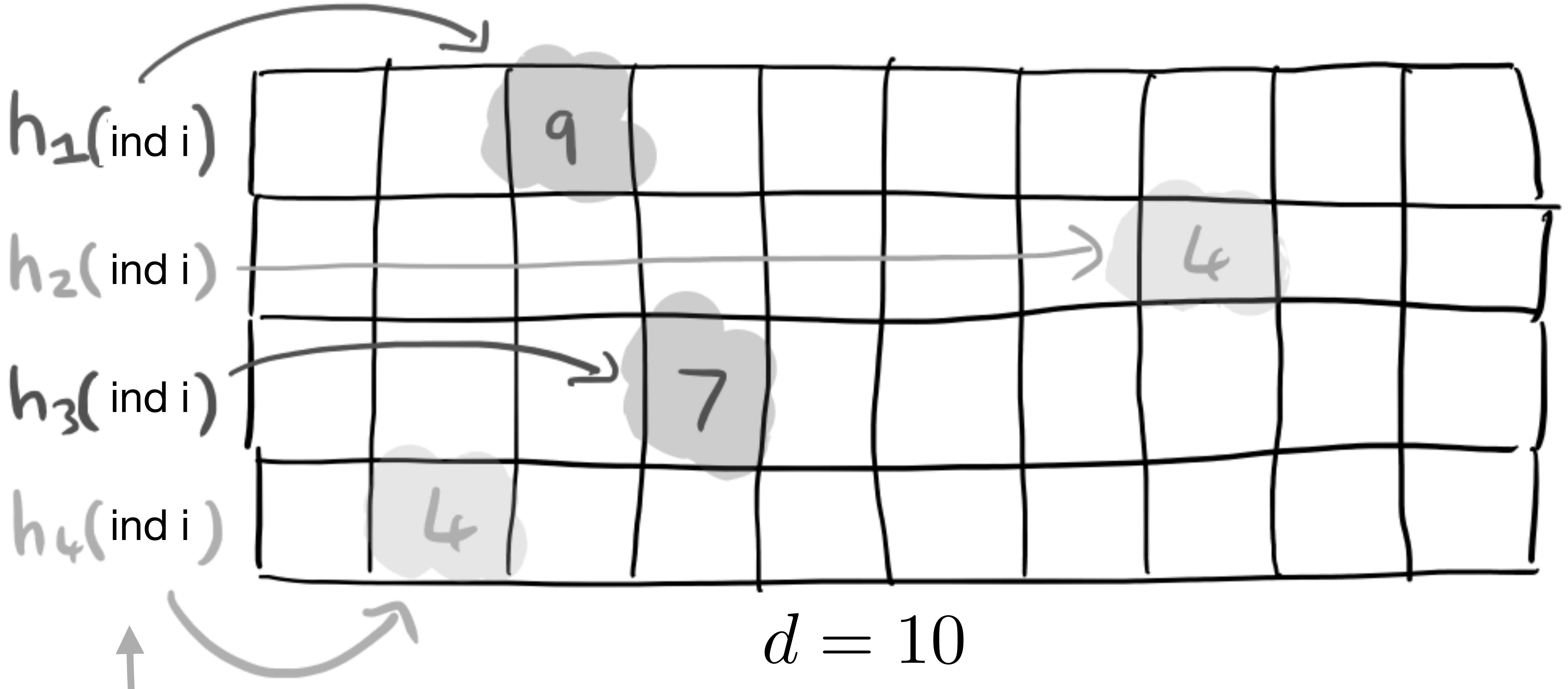
...

feature selection with CS

$$h_j : \{1, 2, \ldots, p\} \rightarrow \{1, 2, \ldots, d\}$$

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \mathbf{g}(\boldsymbol{\beta}_t; \boldsymbol{\Theta}_t)$$





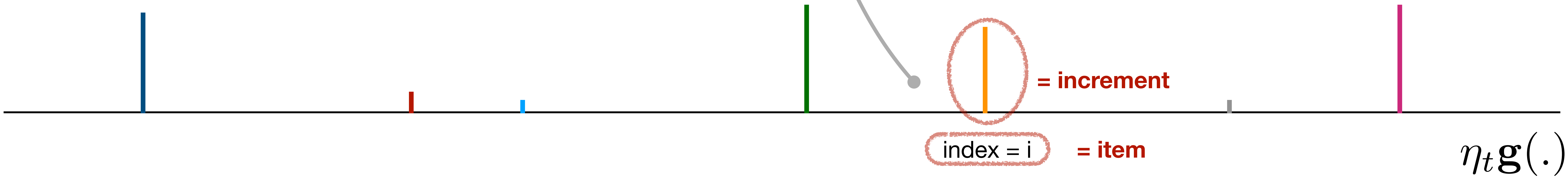
$c = 4$

$d = 10$

+increment $= -\eta_t \mathbf{g}_i(.)$

 $\equiv \eta_t \mathbf{g}_i(.)$

$\#$ items (p)	# all **features**
$m = d \times c$	memory of CS
$\#$ frequent items (k)	# top **features**

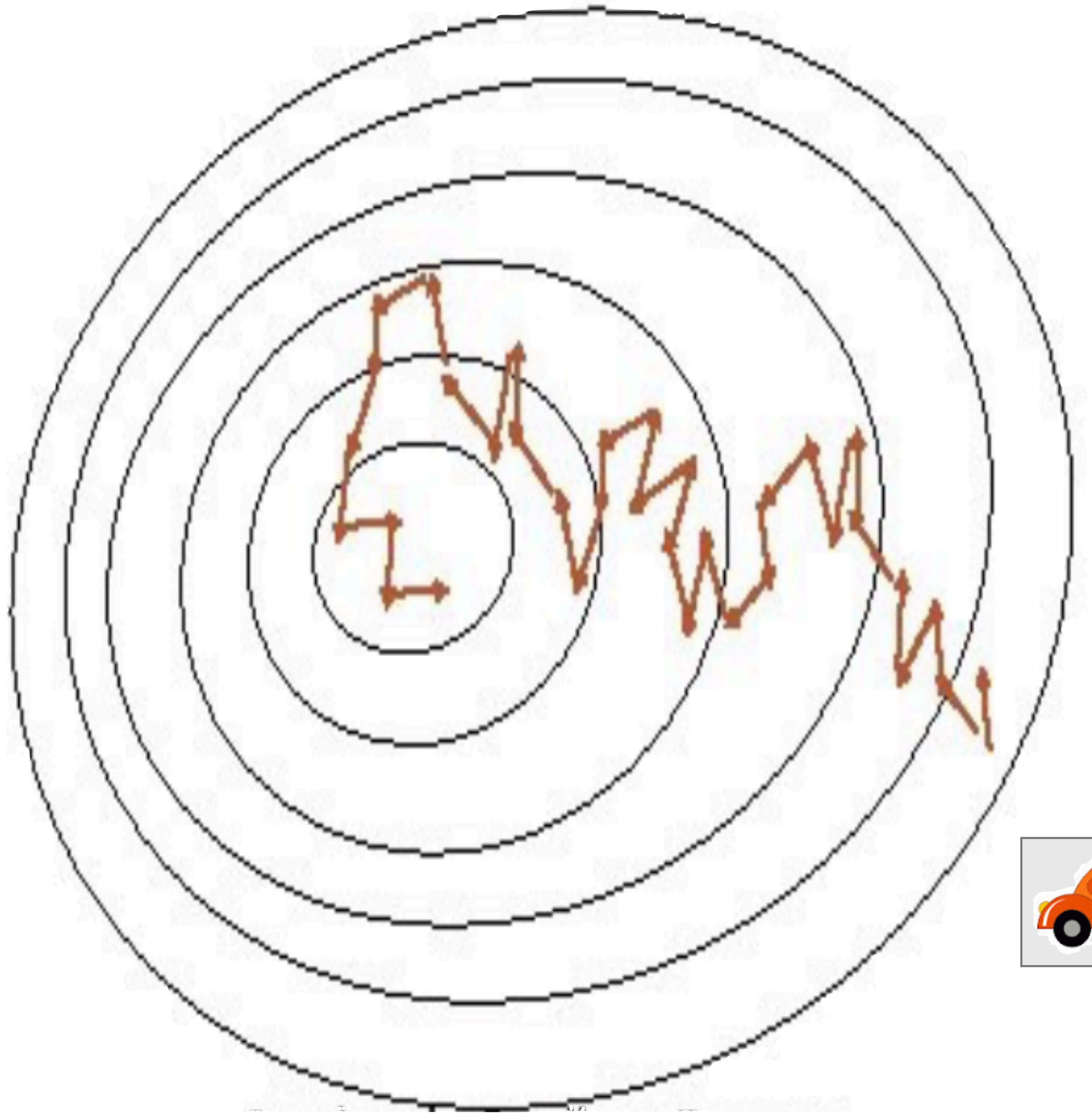


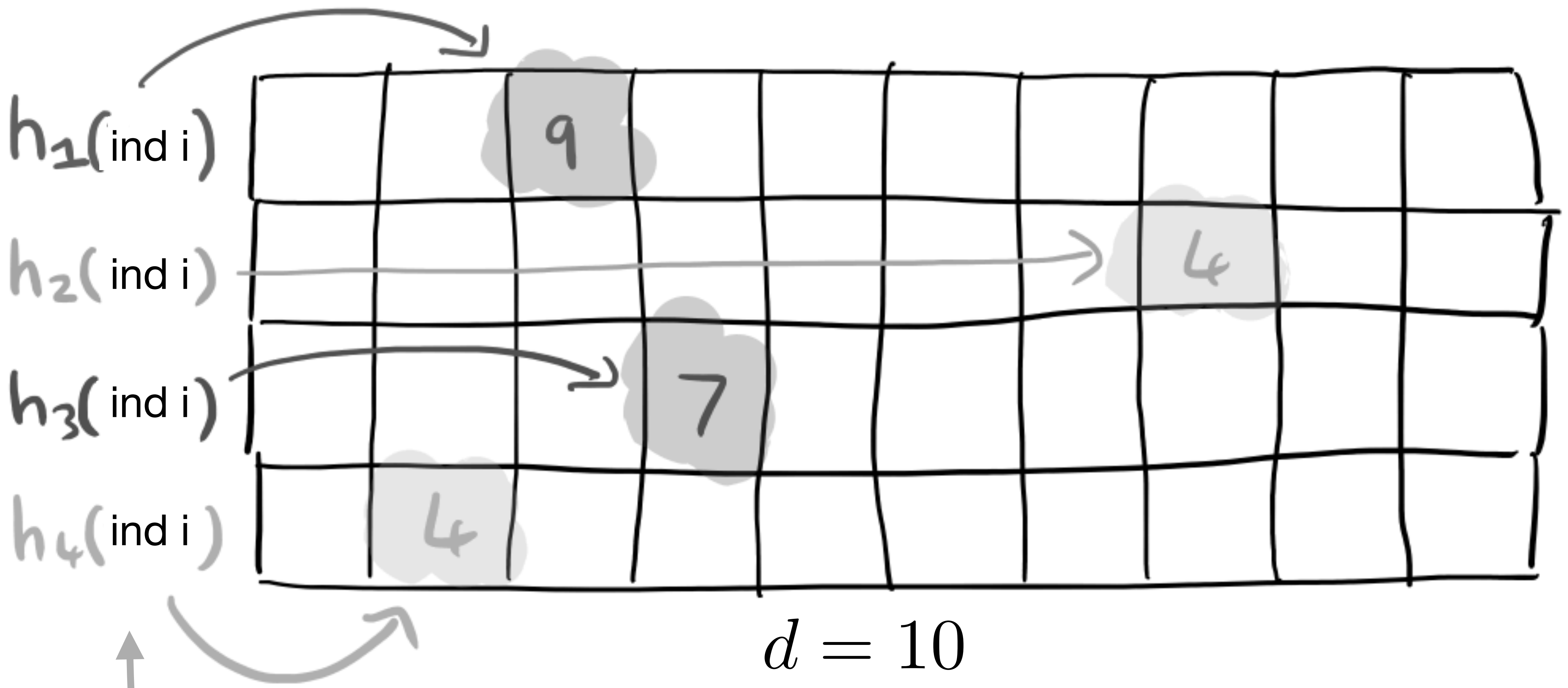
$\eta_t \mathbf{g}(.)$

feature selection with CS

$$h_j : \{1, 2, \ldots, p\} \rightarrow \{1, 2, \ldots, d\}$$

$$\text{MISSION} : \boldsymbol{\beta}^s_{t+1} = \boldsymbol{\beta}^s_t - \eta_t \mathbf{g}^s(\text{Query}_{\text{top}-k}(\boldsymbol{\beta}^s_t); \boldsymbol{\Theta}_t)$$





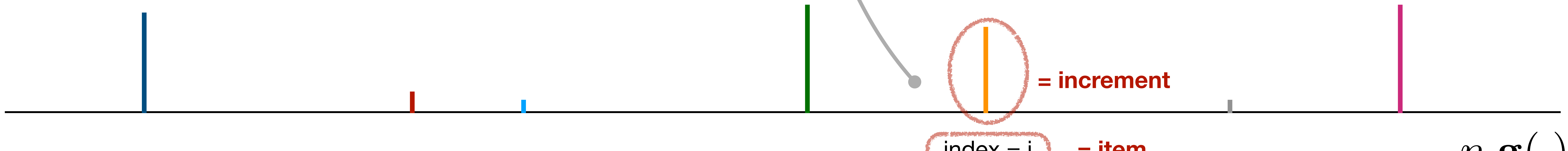
$c = 4$

$d = 10$

+increment = $-\eta_t \mathbf{g}_i(.)$

 $\equiv \eta_t \mathbf{g}_i(.)$

$\#\ \text{items}\ (p)$	# all **features**
$m = d \times c$	memory of CS
$\#\ \text{frequent items}\ (k)$	# top **features**



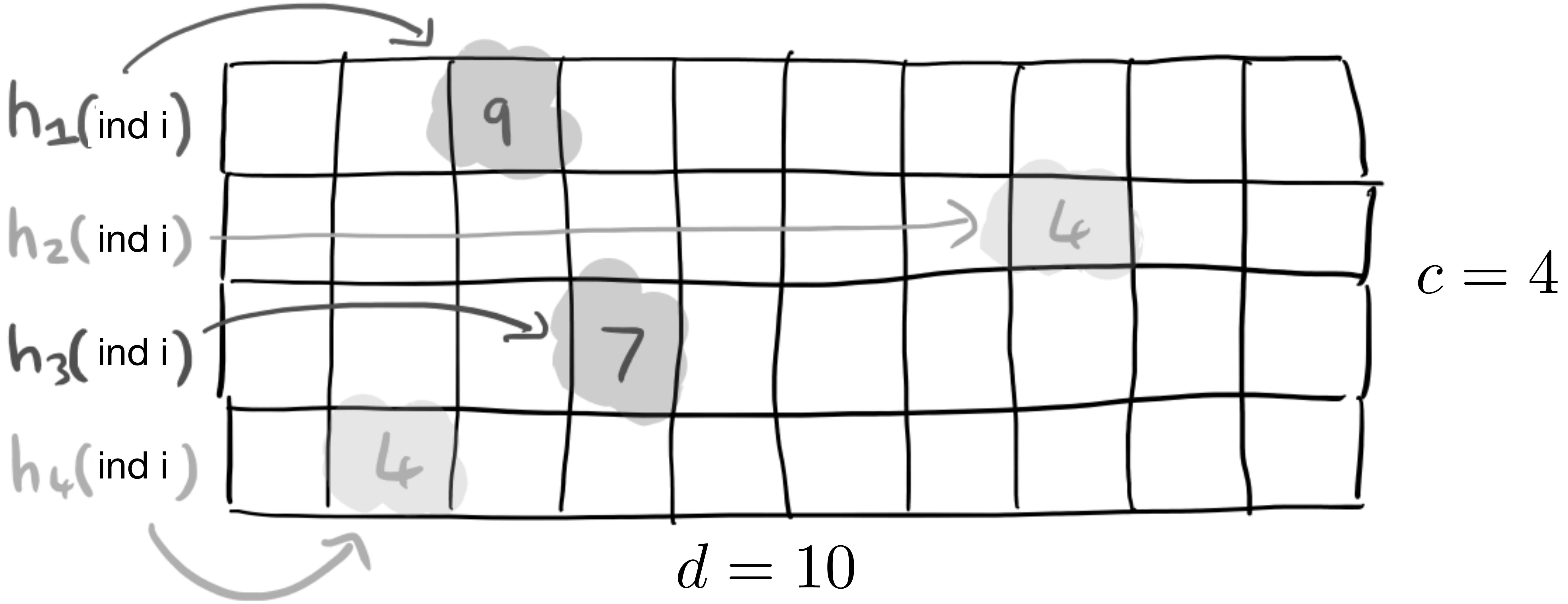
$\eta_t \mathbf{g}(.)$

Aghazadeh, Shrivastava, Baraniuk, et al. *ICML* (2018) PMLR: 80-88.

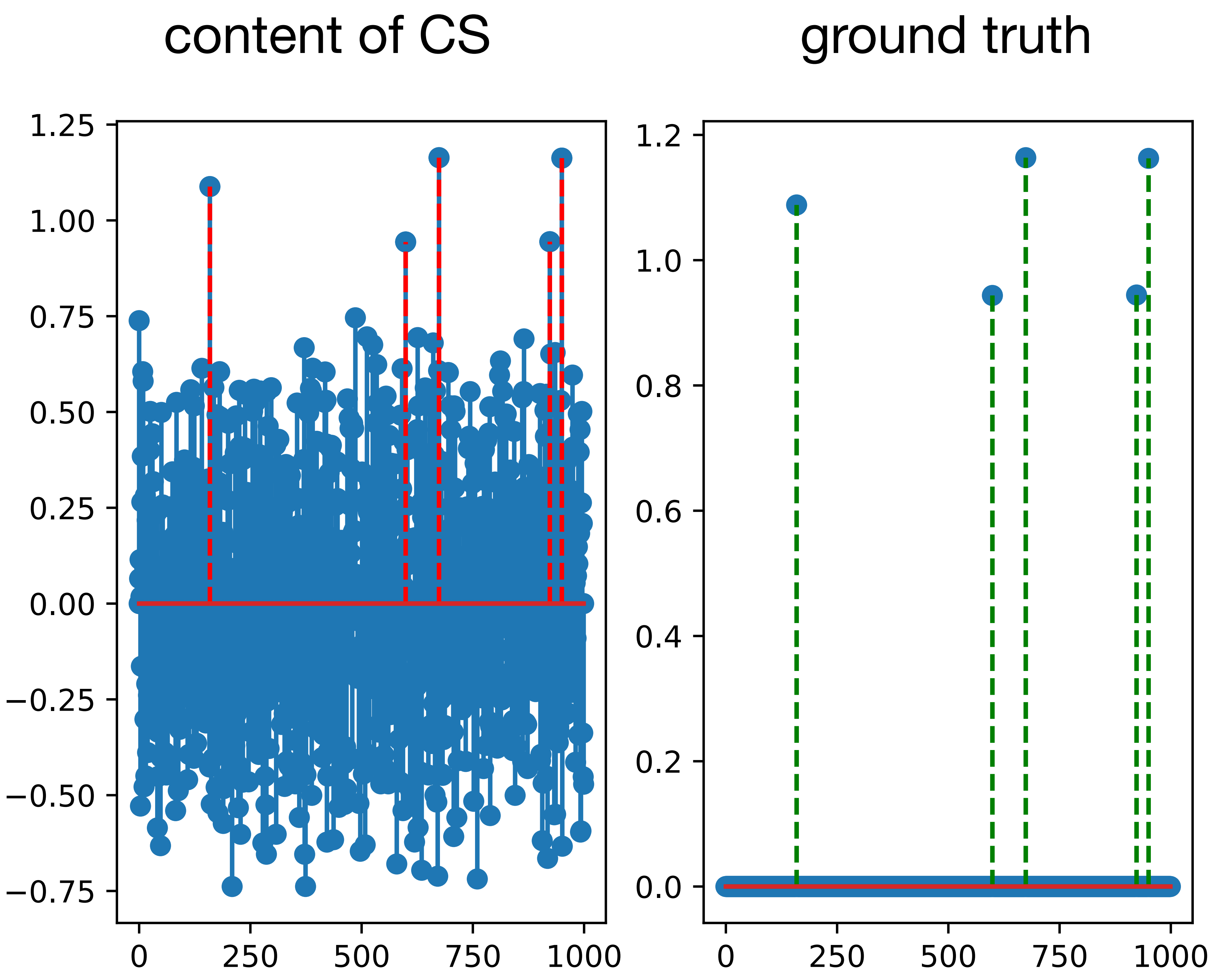
feature selection with CS

$$h_j : \{1, 2, \ldots, p\} \to \{1, 2, \ldots, d\}$$

$$\text{MISSION} : \boldsymbol{\beta}^s_{t+1} = \boldsymbol{\beta}^s_t - \eta_t \mathbf{g}^s(\text{Query}_{\text{top}-k}(\boldsymbol{\beta}^s_t); \boldsymbol{\Theta}_t)$$



after convergence



observation: sketch of noisy component of SGD in CS do not cancel out and results in **memory wasted** to store sketched noise

Theorem 1 *Charikar et al. (2002) Count Sketch finds top-k items z_i with $\pm\varepsilon\|\mathbf{z}\|_2$ error, with probability at least $1-\delta$, in space $\mathcal{O}(log(\frac{p}{\delta})(k + \frac{\|\mathbf{z}^{tail}\|_2^2}{(\varepsilon\zeta)^2}))$, where $\|\mathbf{z}^{tail}\|_2^2 = \sum_{i \notin top-k} z_i^2$ is the energy of the non-top-k items and ζ is the k^{th} largest value in $\mathbf{z}$.*

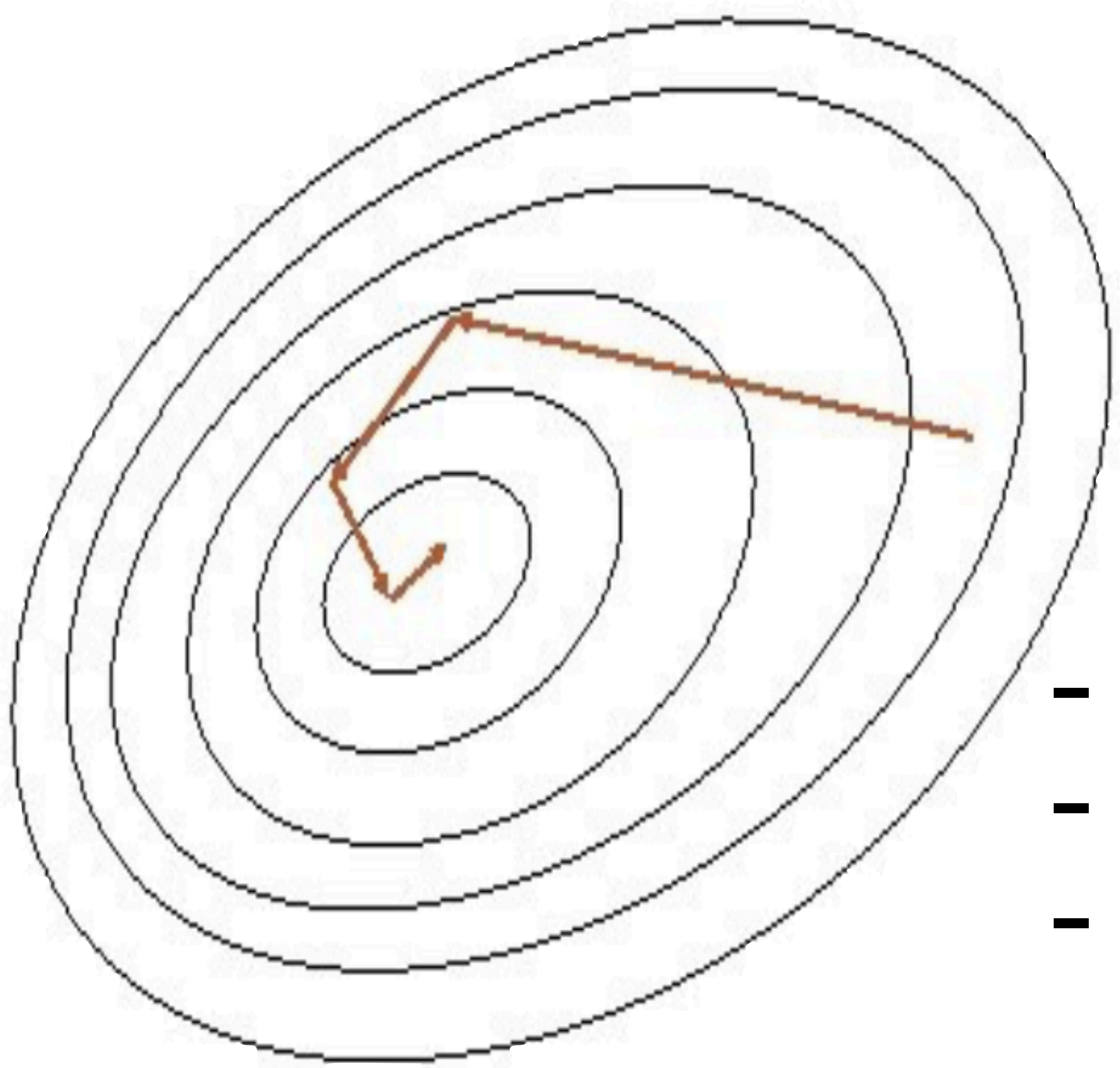
Aghazadeh, Gupta, Ramchandran et al. *MSML* (2021)

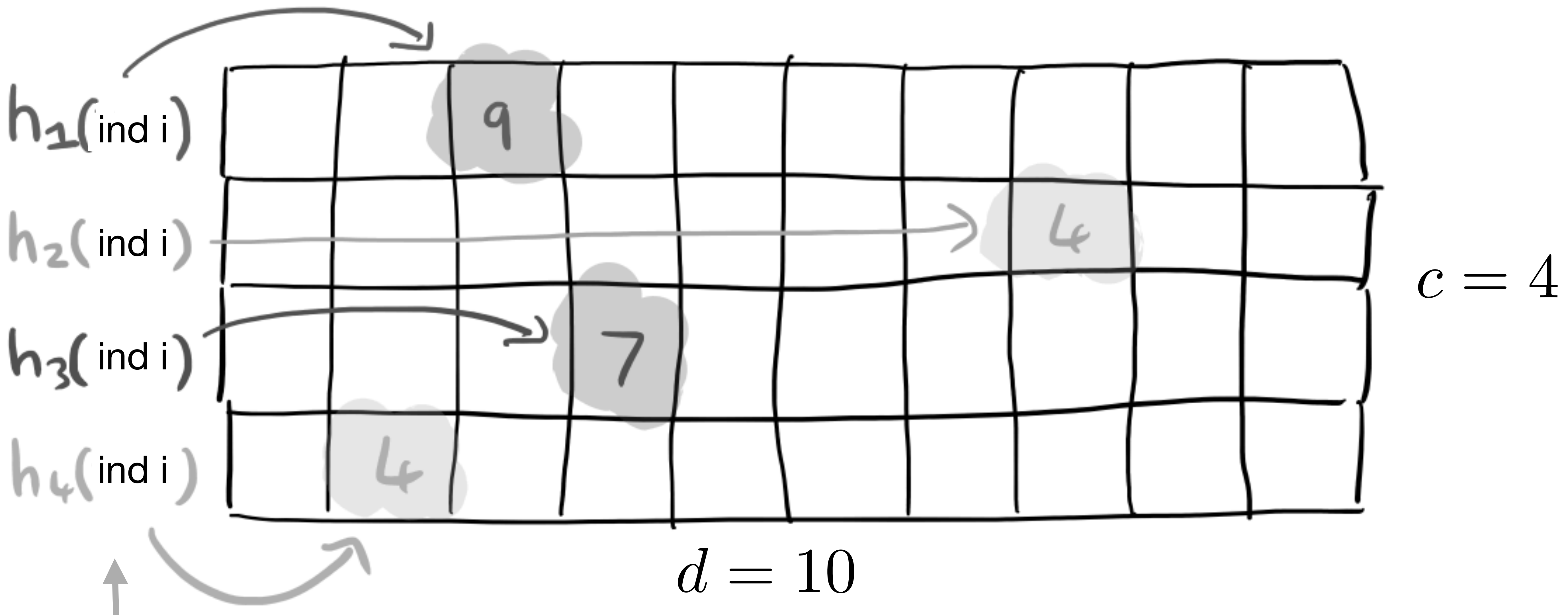
idea: second order sketching

$$h_j : \{1, 2, \ldots, p\} \to \{1, 2, \ldots, d\}$$

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \mathbf{B}_t^{-1} \mathbf{g}(\boldsymbol{\beta}_t, \boldsymbol{\Theta}_t)$$

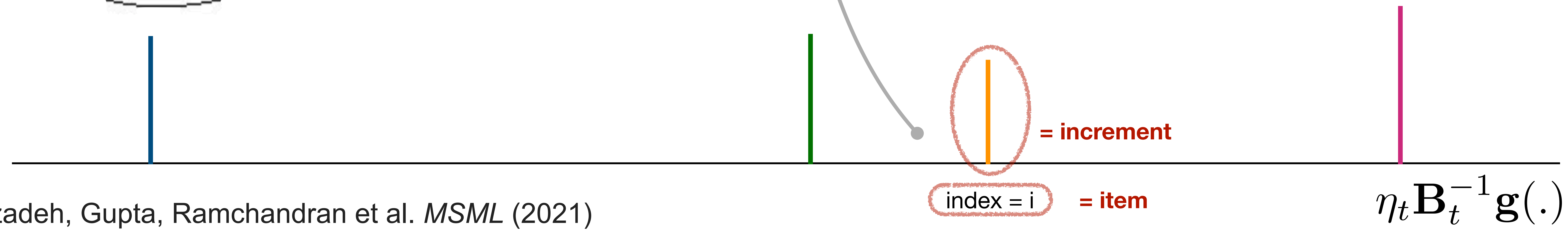
$$\mathbf{B}_t = \nabla^2_{\boldsymbol{\beta}_t} f(\boldsymbol{\beta}_t, \boldsymbol{\Theta}_t) \in \mathbb{R}^{p \times p}$$





- more comp. cost per iteration
- less noisy gradient
- memory-accuracy tradeoff

question: **how to compute/ store the Hessian?**



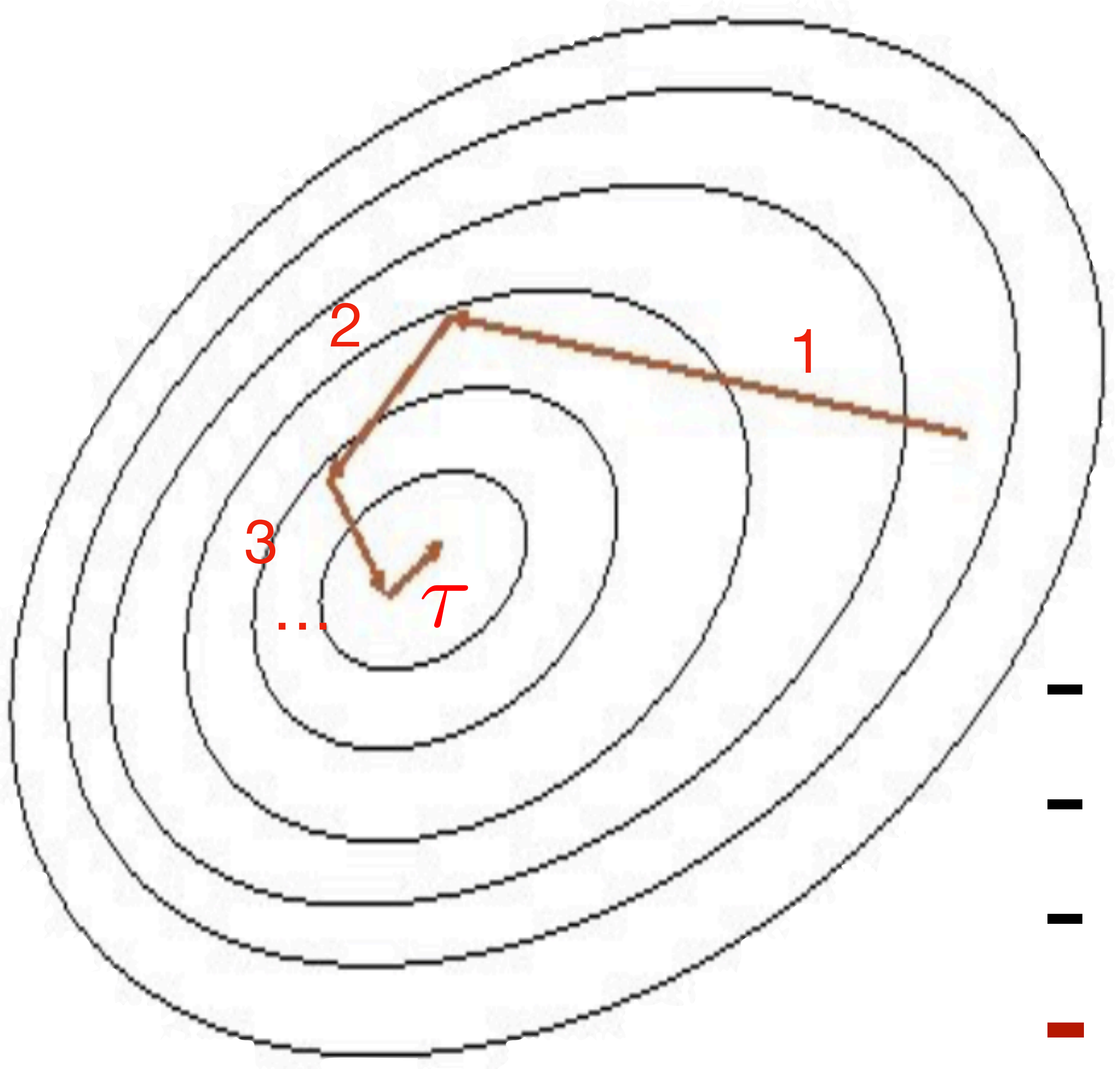
Aghazadeh, Gupta, Ramchandran et al. *MSML* (2021)

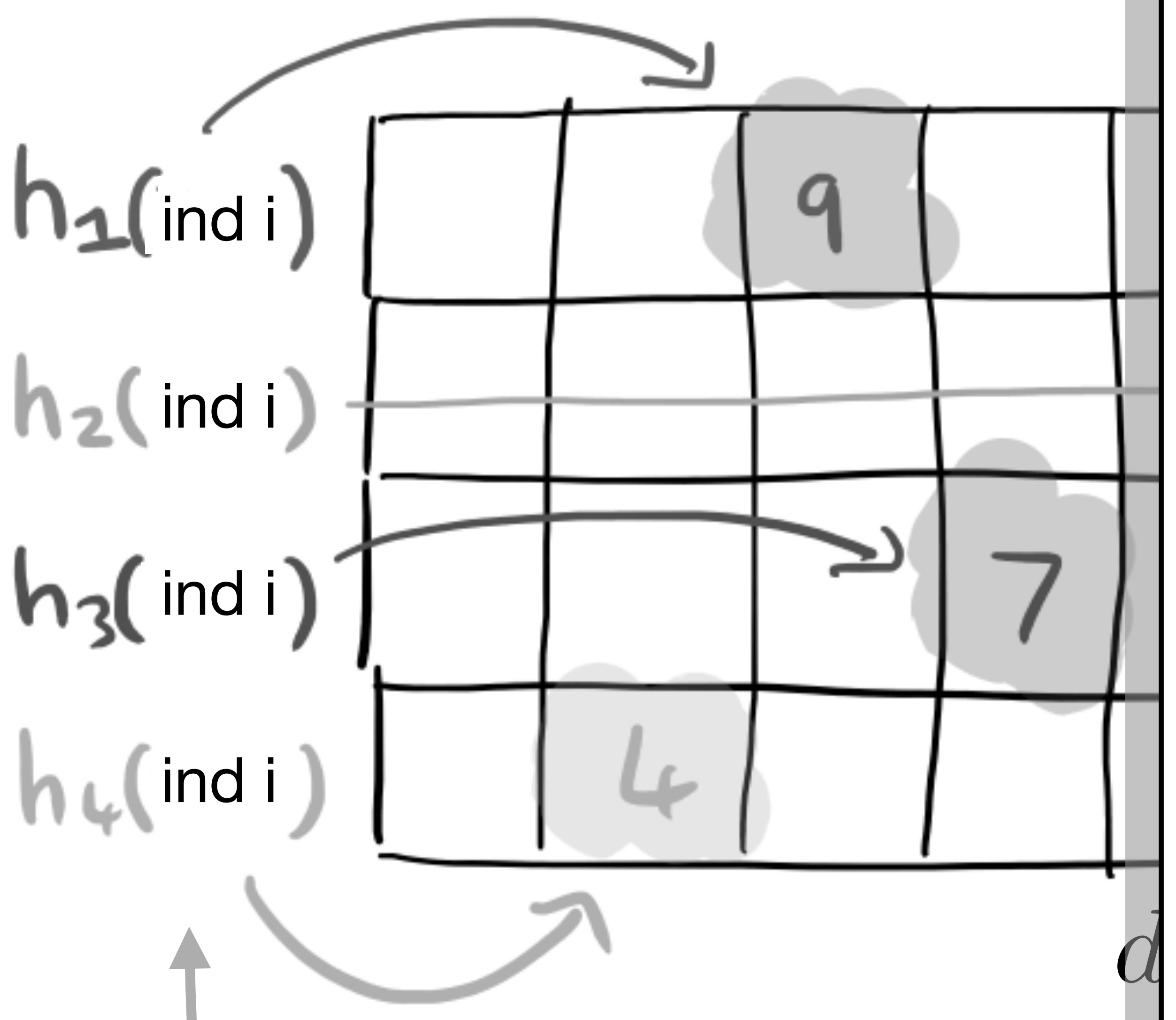
limited-memory BFGS

$$h_j : \{1, 2, \ldots, p\} \rightarrow \{1, 2, \ldots, c\}$$

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t - \eta_t \mathbf{B}_t^{-1} \mathbf{g}(\boldsymbol{\beta}_t, \boldsymbol{\Theta}_t)$$

$$\mathbf{B}_t = \nabla^2_{\boldsymbol{\beta}_t} f(\boldsymbol{\beta}_t, \boldsymbol{\Theta}_t) \in \mathbb{R}^{p \times p}$$





Algorithm 1 Limited-memory BFGS

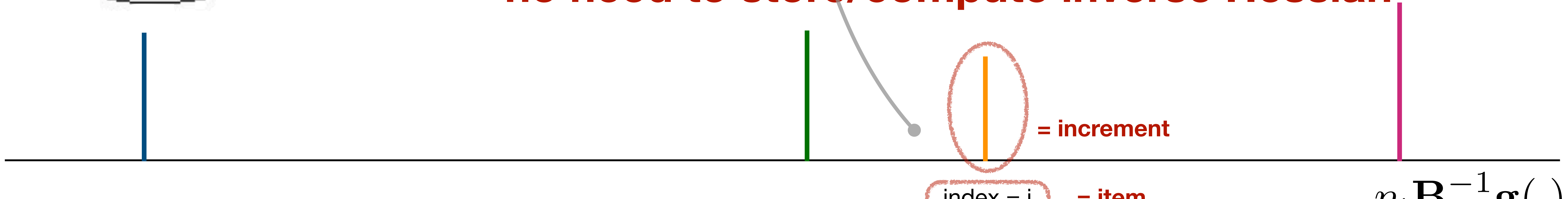
Input: $\mathbf{g}(\hat{\boldsymbol{\beta}}_t, \boldsymbol{\Theta}_t)$ and $\{\mathbf{s}_i, \mathbf{r}_i\}_{i=t-\tau+1}^{t}$

1. $\rho_t = \frac{1}{\mathbf{r}_t^T \mathbf{s}_t}$.
2. $\mathbf{q}_t = \mathbf{g}(\hat{\boldsymbol{\beta}}_t, \boldsymbol{\Theta}_t)$,
   for $i = t$ to $t - \tau + 1$:
   $\alpha_i = \rho_i \mathbf{s}_i^T \mathbf{q}_i$,
   $\mathbf{q}_{i-1} = \mathbf{q}_i - \alpha_i \mathbf{r}_i$.
3. $\mathbf{z}_{t-\tau} = \frac{\mathbf{r}_t^T \mathbf{s}_t}{\mathbf{r}_t^T \mathbf{r}_t} \mathbf{q}_{t-\tau}$,
   for $i = t - \tau + 1$ to t:
   $\gamma_i = \rho_i \mathbf{r}_i^T \mathbf{z}_i$.
   $\mathbf{z}_i = \mathbf{z}_{i-1} + \mathbf{s}_i(\alpha_i - \gamma_i)$.

Return: $\mathbf{z}_t$

approximate $\mathbf{B}_t^{-1}\mathbf{g}(.)$ using gradients from last few τ iterations

- more comp. cost per iteration
- less noisy gradient
- memory-accuracy tradeoff
- **no need to store/compute inverse Hessian**



Aghazadeh, Gupta, Ramchandran et al. *MSML* (2021)

BEAR algorithm: sketch LBFGS gradients using CS

find the descent direction using LBFGS and update CS

Algorithm 2 BEAR

Initialize: $t = 0$, Count Sketch $\boldsymbol{\beta}^s_{t=0} = 0$, top-$k$ heap.

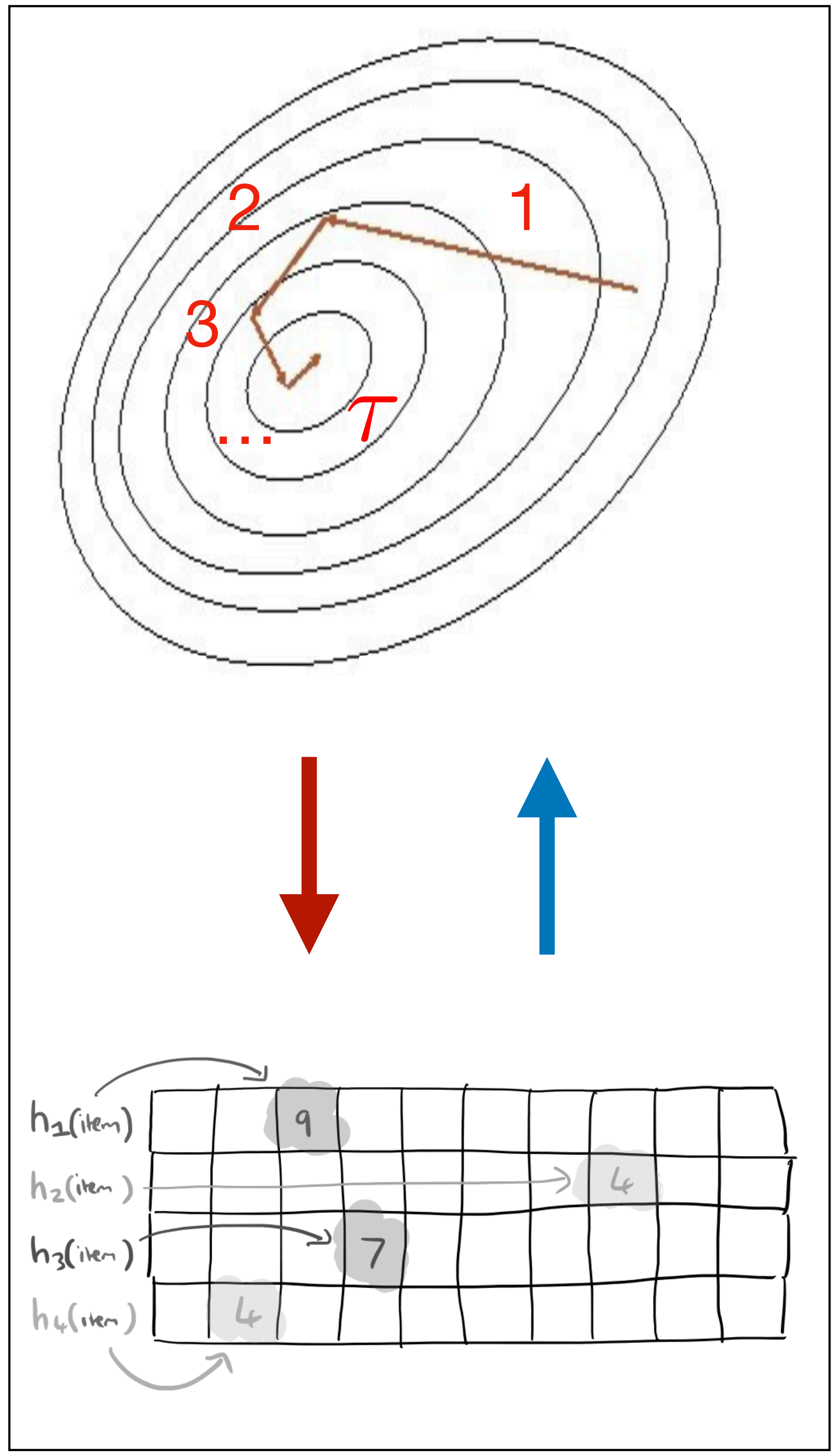
while stopping criteria not satisfied **do**

1. Sample b independent data points in a minibatch $\boldsymbol{\Theta}_t = \{\boldsymbol{\theta}_{t1}, \ldots, \boldsymbol{\theta}_{tb}\}$.
2. Find the active set $\mathcal{A}_t$.
3. QUERY the feature weights in $\mathcal{A}_t \cap$ top-k from Count Sketch $\boldsymbol{\beta}_t = query(\boldsymbol{\beta}^s_t)$.
4. Compute stochastic gradient $\mathbf{g}(\boldsymbol{\beta}_t, \boldsymbol{\Theta}_t)$.
5. Compute the descent direction with Alg. 1 $\mathbf{z}_t = \text{LBFGS}(\mathbf{g}(\boldsymbol{\beta}_t, \boldsymbol{\Theta}_t)\ ,\ \{\mathbf{s}_i, \mathbf{r}_i\}^t_{i=t-\tau+1})$.
6. ADD the sketch of $\mathbf{z}_t$ at the active set $\hat{\mathbf{z}}_t = {\mathbf{z}_t}^{\mathcal{A}_t}$ to Count Sketch $\boldsymbol{\beta}^s_{t+1} := \boldsymbol{\beta}^s_t - \eta_t \hat{\mathbf{z}}^s_t$.
7. QUERY the features weights in $\mathcal{A}_t \cap$ top-k from Count Sketch $\boldsymbol{\beta}_{t+1} = query(\boldsymbol{\beta}^s_{t+1})$.
8. Compute stochastic gradient $\mathbf{g}(\boldsymbol{\beta}_{t+1}, \boldsymbol{\Theta}_t)$.
9. Set $\mathbf{s}_{t+1} = \boldsymbol{\beta}_{t+1} - \boldsymbol{\beta}_t$, and $\mathbf{r}_{t+1} = \mathbf{g}(\boldsymbol{\beta}_{t+1}, \boldsymbol{\Theta}_t) - \mathbf{g}(\boldsymbol{\beta}_t, \boldsymbol{\Theta}_t)$.
10. Update the top-k heap.
11. $t = t + 1$.

end while

Return: The top-k heavy-hitters in Count Sketch.

query CS and store the gradient and feature difference vectors



convergence

Theorem 2 *Let $f(\cdot)$ and the step sizes η_t satisfy the assumptions above. Let the size of Count Sketch be $m = \theta(\varepsilon^{-2}\log 1/\delta)$ with number of hashes $d = \theta(\varepsilon^{-1}\log 1/\delta)$ for $\varepsilon, \delta > 0$. Then, the Euclidean distance between updates $\boldsymbol{\beta}_t^s$ in the* BEAR *algorithm and the sketch of the solution of problem* (1) *converges to zero with probability $1-\delta$, that is,*

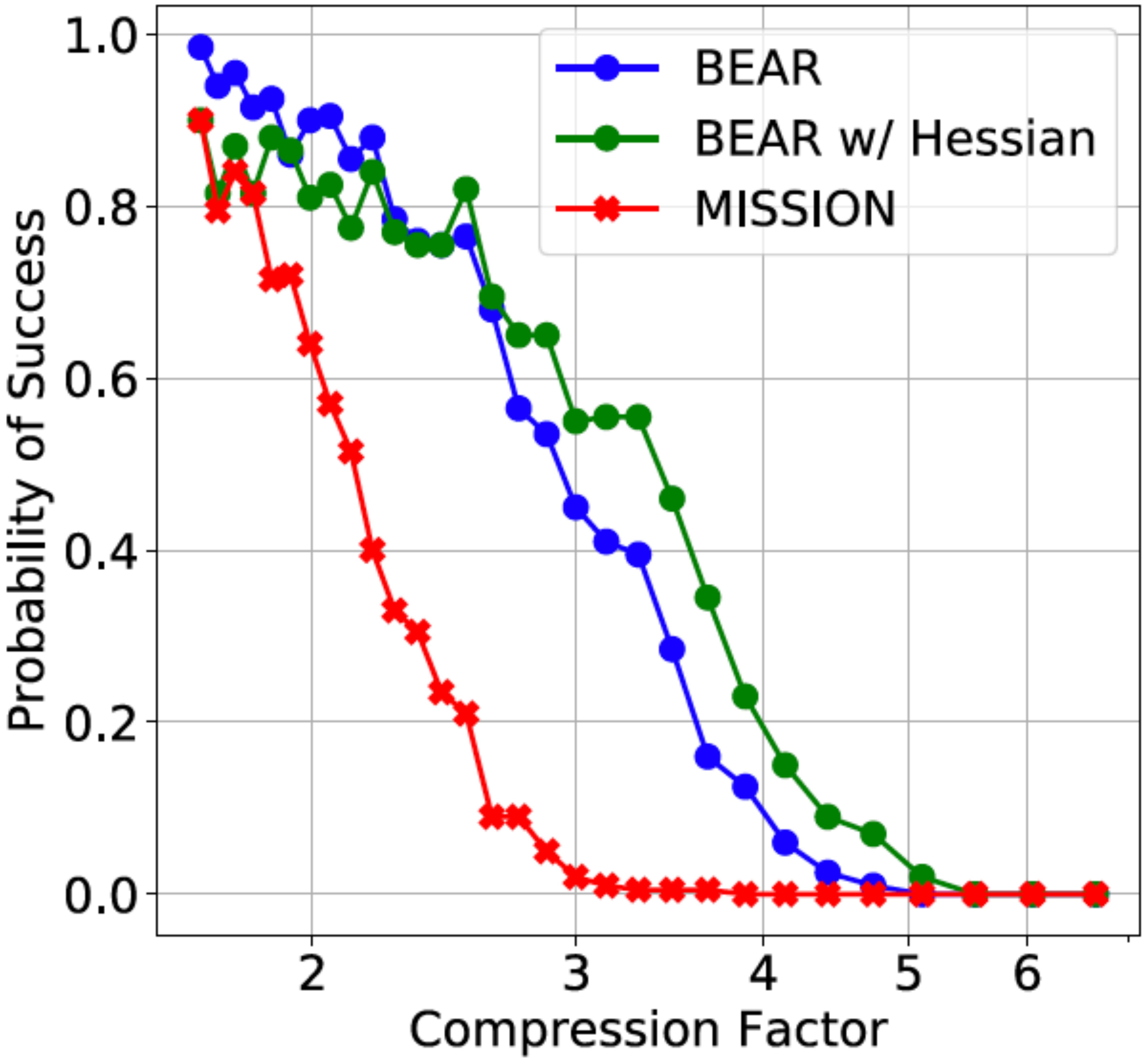
$$\boxed{\mathbb{P}(\lim_{t\to\infty} \|\boldsymbol{\beta}_t^s - \boldsymbol{\beta}^{s*}\|^2 = 0) = 1 - \delta,} \tag{2}$$

where the probability is over the random realizations of random samples $\{\boldsymbol{\Theta}_t\}_{t=0}^{\infty}$. Furthermore, for the specific step size $\eta_t = \eta_0/(t+T_0)$ for some constants η_0 and T_0, the model parameters at iteration t satisfy

$$\boxed{\mathbb{E}_{\boldsymbol{\Theta}}[f(\boldsymbol{\beta}_t^s, \boldsymbol{\Theta}) - \mathbb{E}[f(\boldsymbol{\beta}^{s*}, \boldsymbol{\Theta})] \leq \frac{C_0}{T_0 + t},} \tag{3}$$

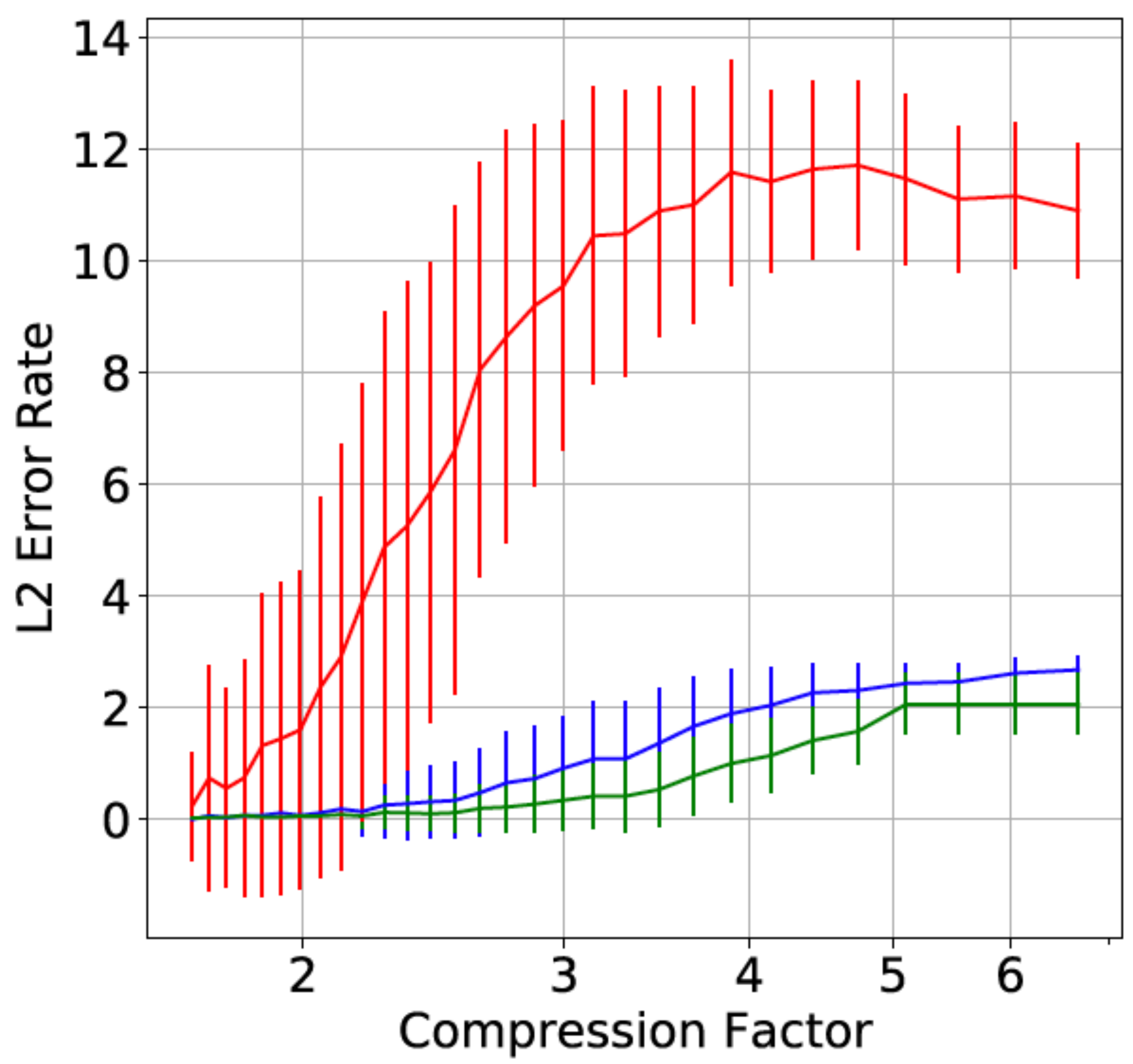
with probability $1-\delta$. Here, C_0 is a constant depending on the parameters of the sketching scheme, the above assumptions, and the objective function.

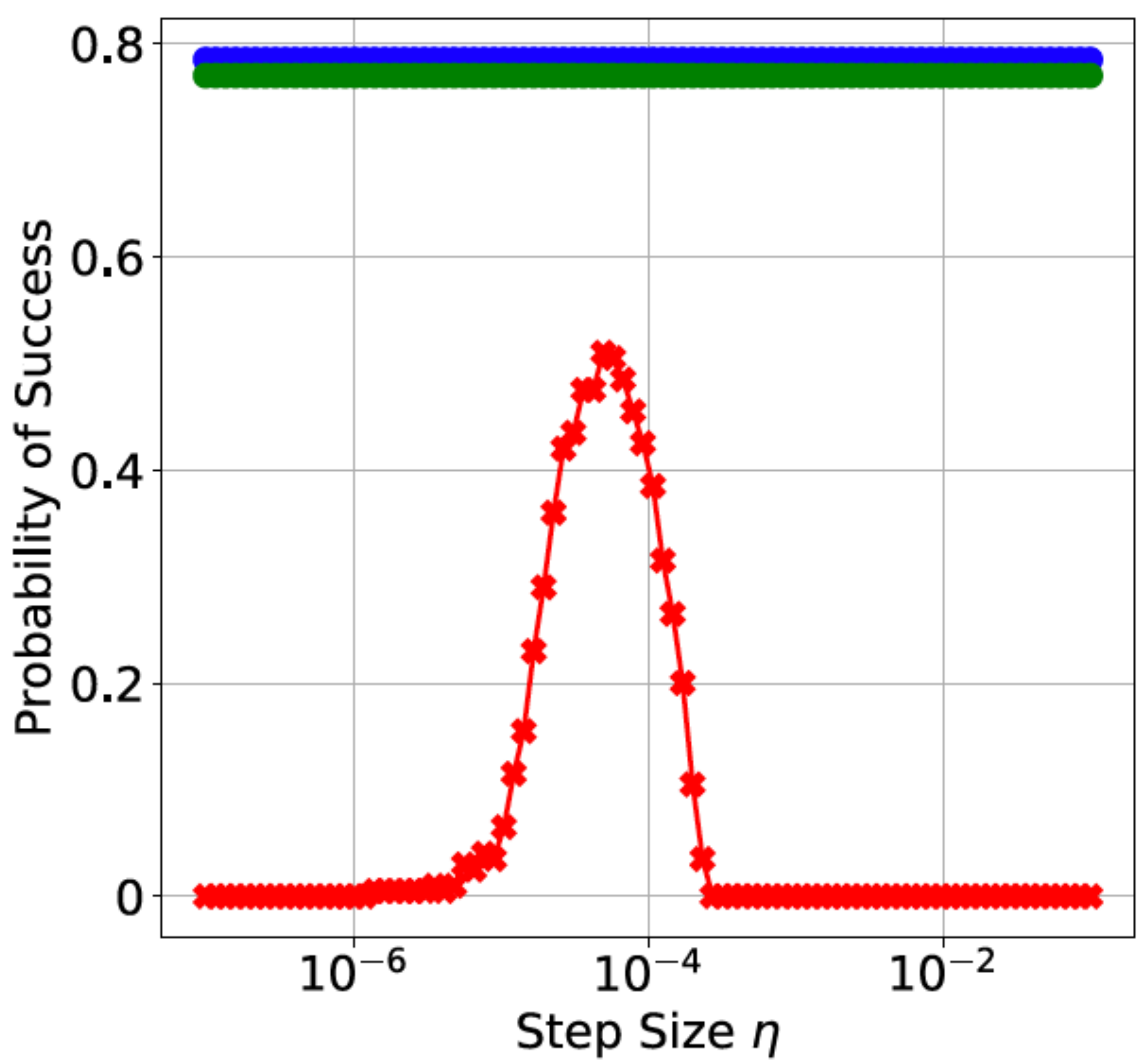
simulations



$$\mathbf{x}_i \sim \mathcal{N}(0,1)$$
$$y_i = \mathbf{x}_i \boldsymbol{\beta}^*$$
$$\boldsymbol{\beta}^* : k - \text{sparse}$$
$$\text{CF} = \frac{p}{\text{size of CS}}$$

$$p = 1000$$
$$n = 900$$
$$k = 8$$

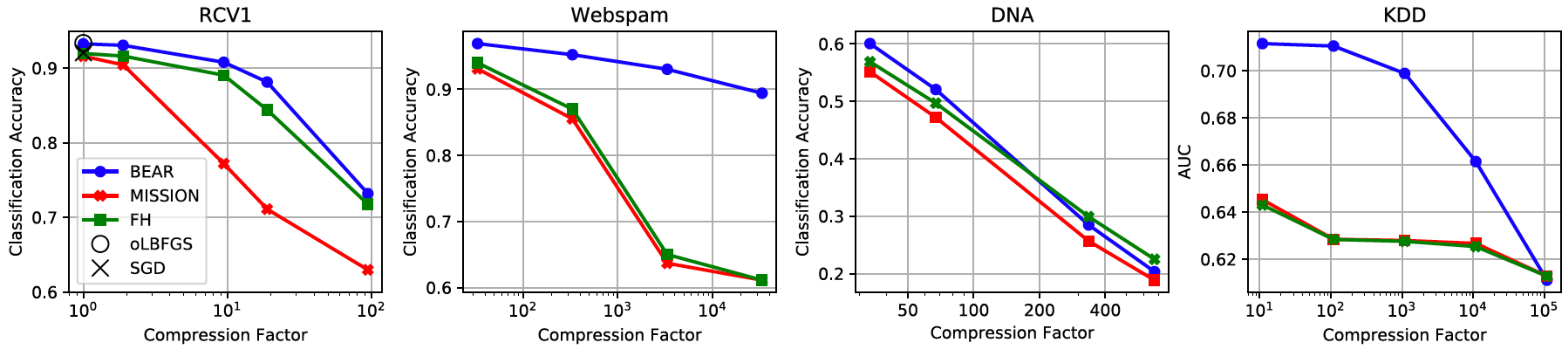




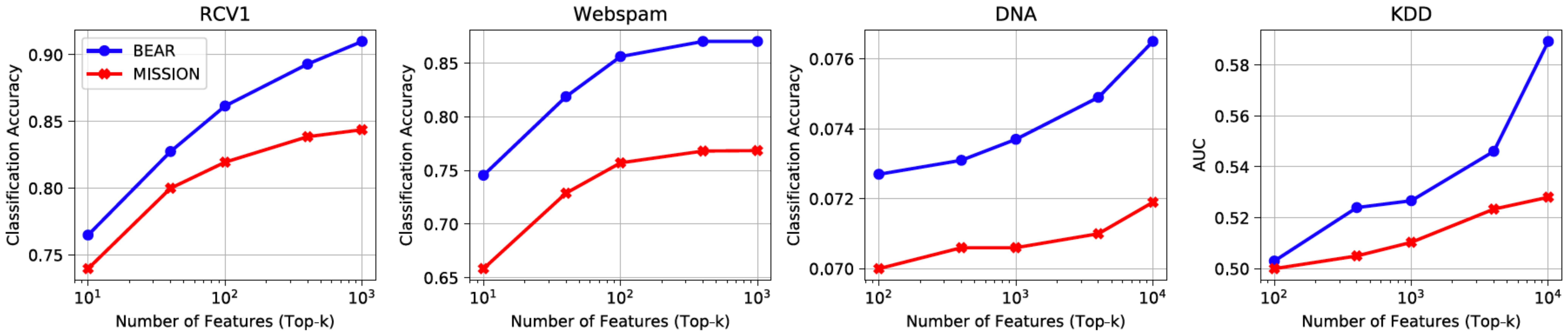
real-world experiments

Data set	Dim (p)	#Train (n)	#Test	Size	#Act.
RCV1	47,236	20,242	677,399	1.2GB	73
Webspam	16,609,143	280,000	70,000	25GB	3730
DNA	16,777,216	600,000	600,000	1.5GB	89
KDD 2012	54,686,452	119,705,032	29,934,073	22GB	12

Classification



Feature selection



summary and future directions

- **adaptively** learn the hashing scheme in the Count Sketch based on the stochastic gradients

- efficient training of massively large **nonlinear models** using LBFGS + sketching (Transforms, etc.)

- **distributed** learning/analysis using LBFGS + sketching explore communication-computation tradeoff

Thanks!

- find the **paper** at https://arxiv.org/abs/2010.13829
- find the **code** at https://github.com/BEAR-algorithm/BEAR