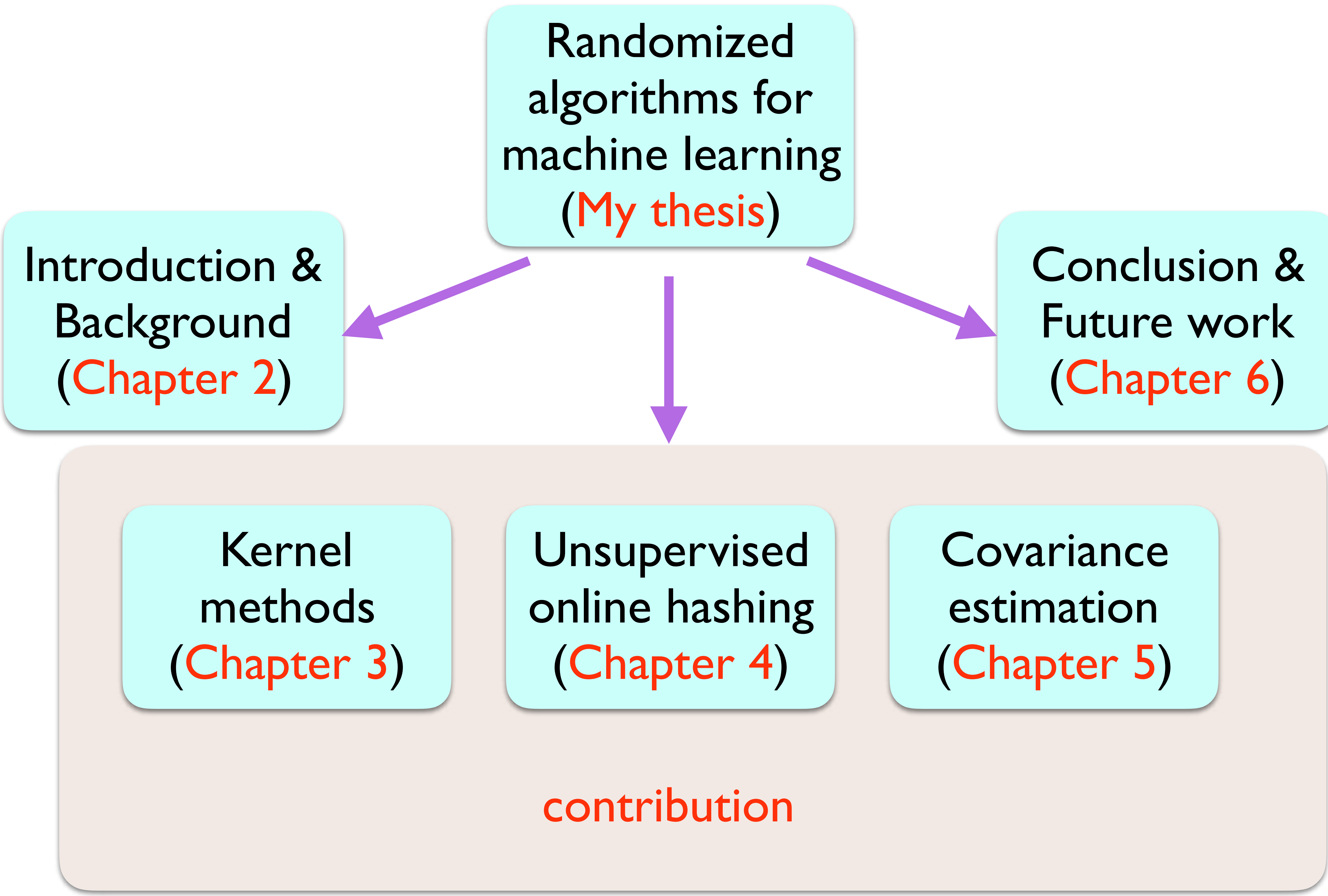
Randomized Algorithms for Machine Learning

Xixian Chen

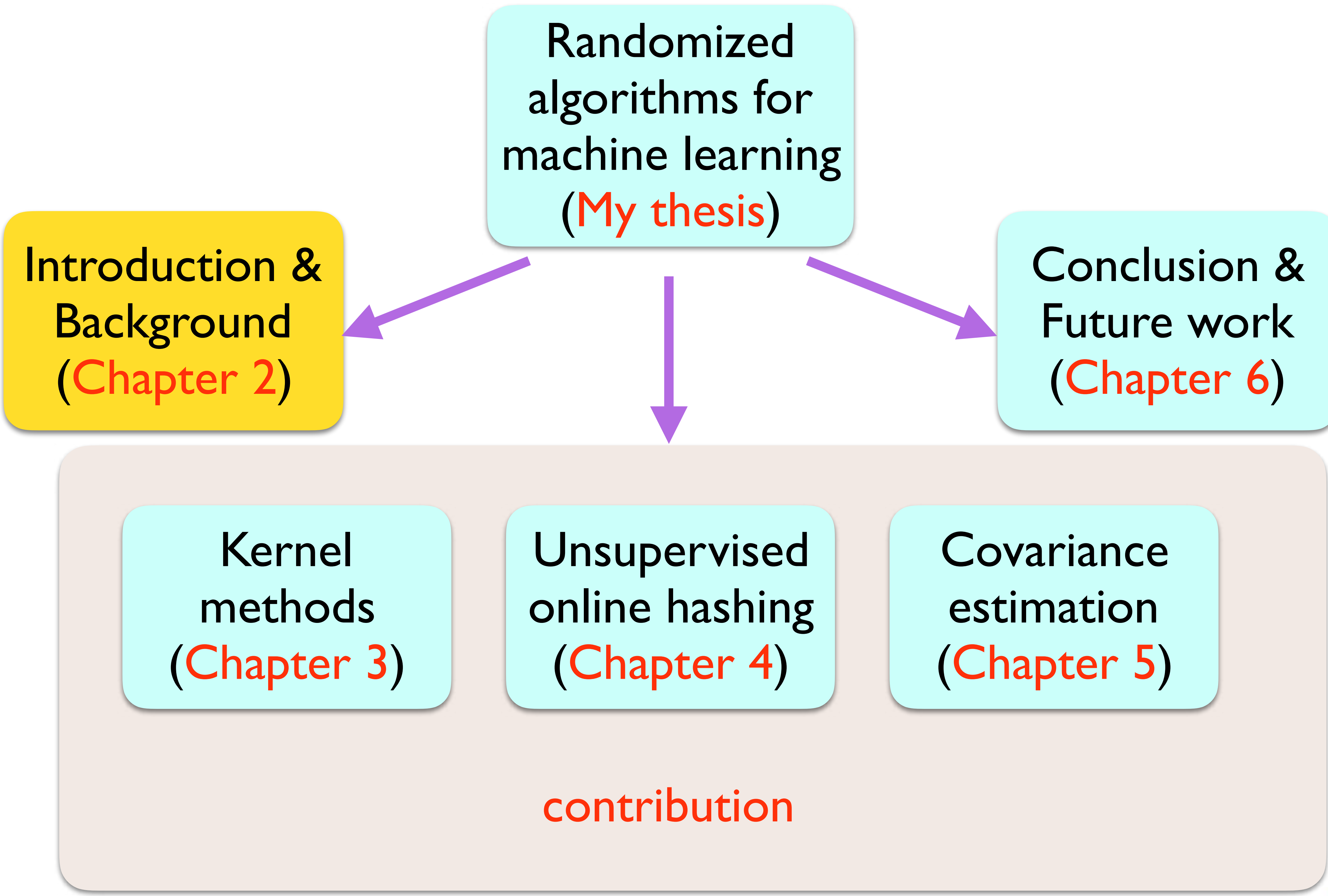
Department of Computer Science and Engineering
The Chinese University of Hong Kong

Supervisors: Prof. Irwin King & Prof. Michael R. Lyu

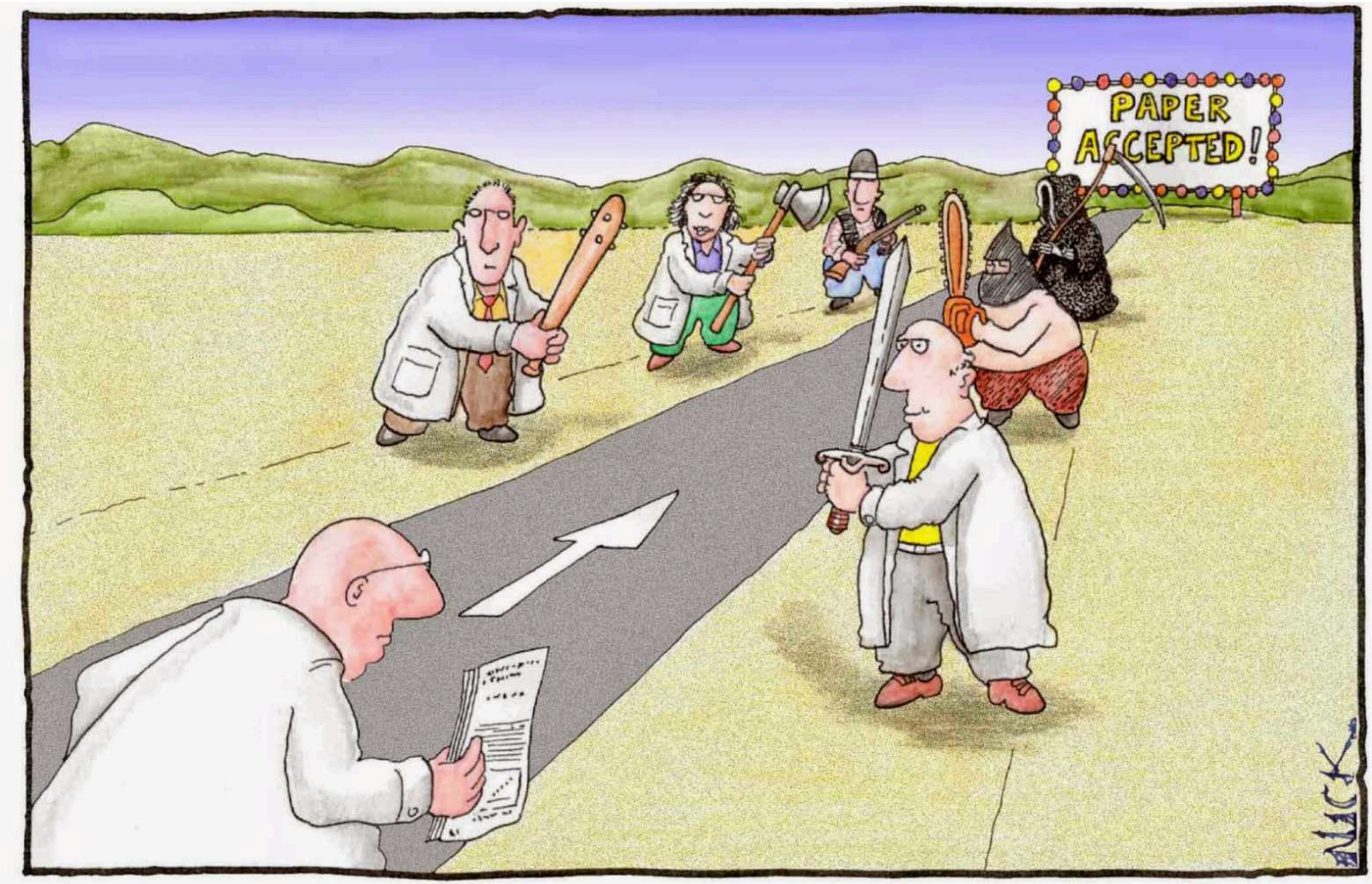
Outline


Randomized algorithms for machine learning (My thesis)
Introduction & Background (Chapter 2)
Conclusion & Future work (Chapter 6)
Kernel methods (Chapter 3)
Unsupervised online hashing (Chapter 4)
Covariance estimation (Chapter 5)
contribution

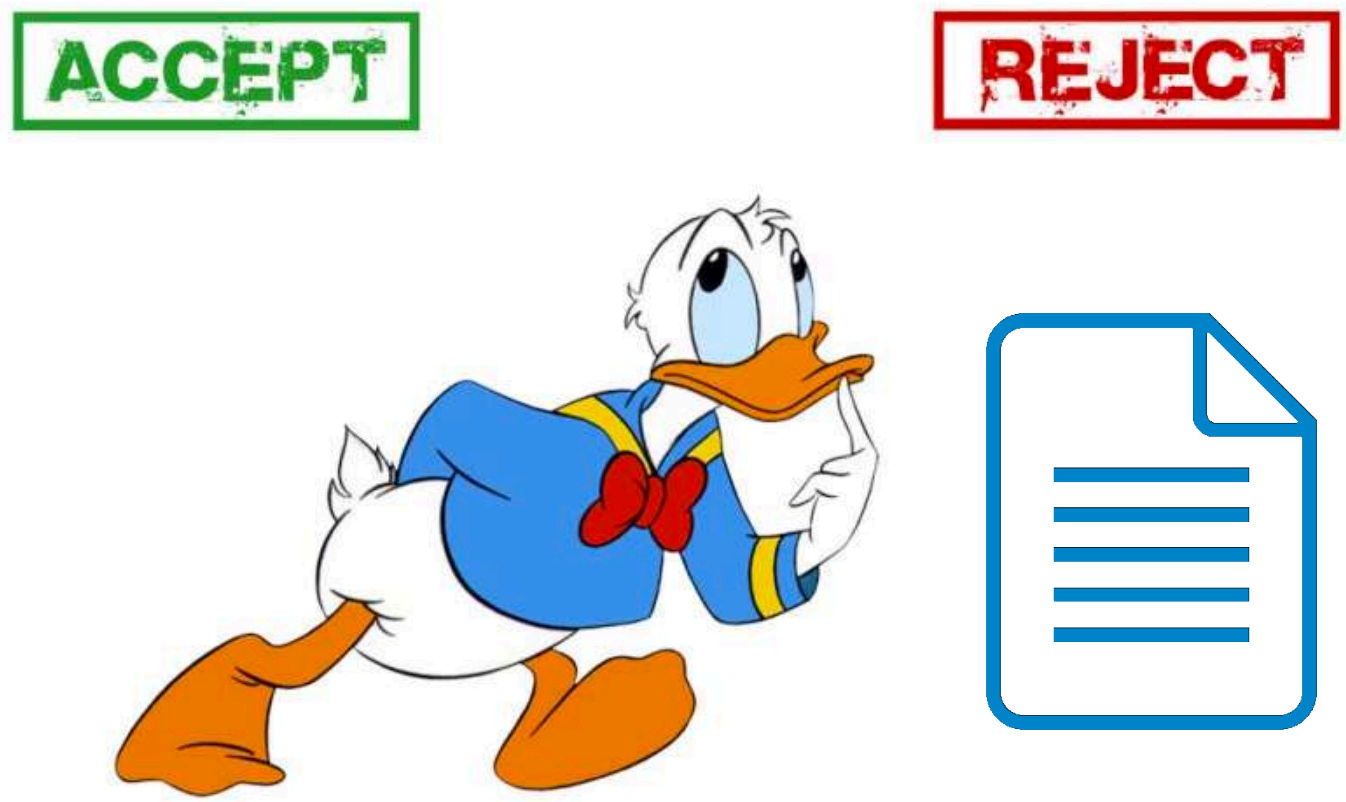
Outline


Randomized algorithms for machine learning (My thesis)
Introduction & Background (Chapter 2)
Conclusion & Future work (Chapter 6)
Kernel methods (Chapter 3)
Unsupervised online hashing (Chapter 4)
Covariance estimation (Chapter 5)
contribution

The Dangerous Path of Publication

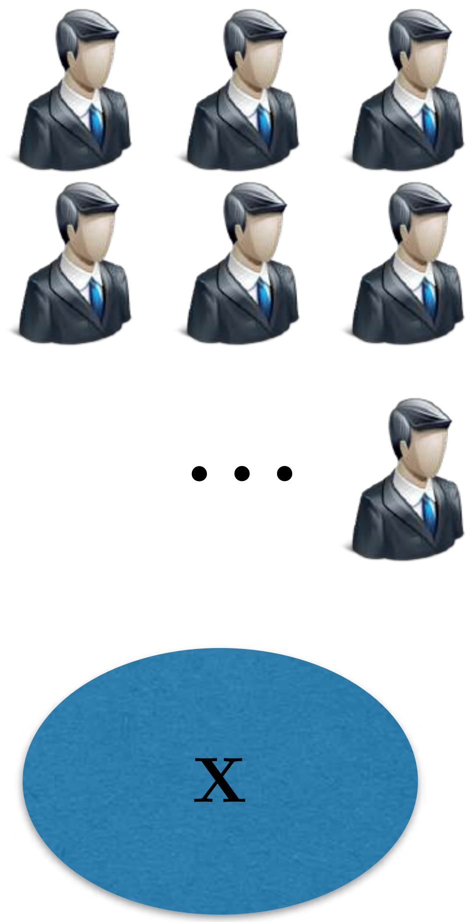


How to Decide to Accept a Paper?



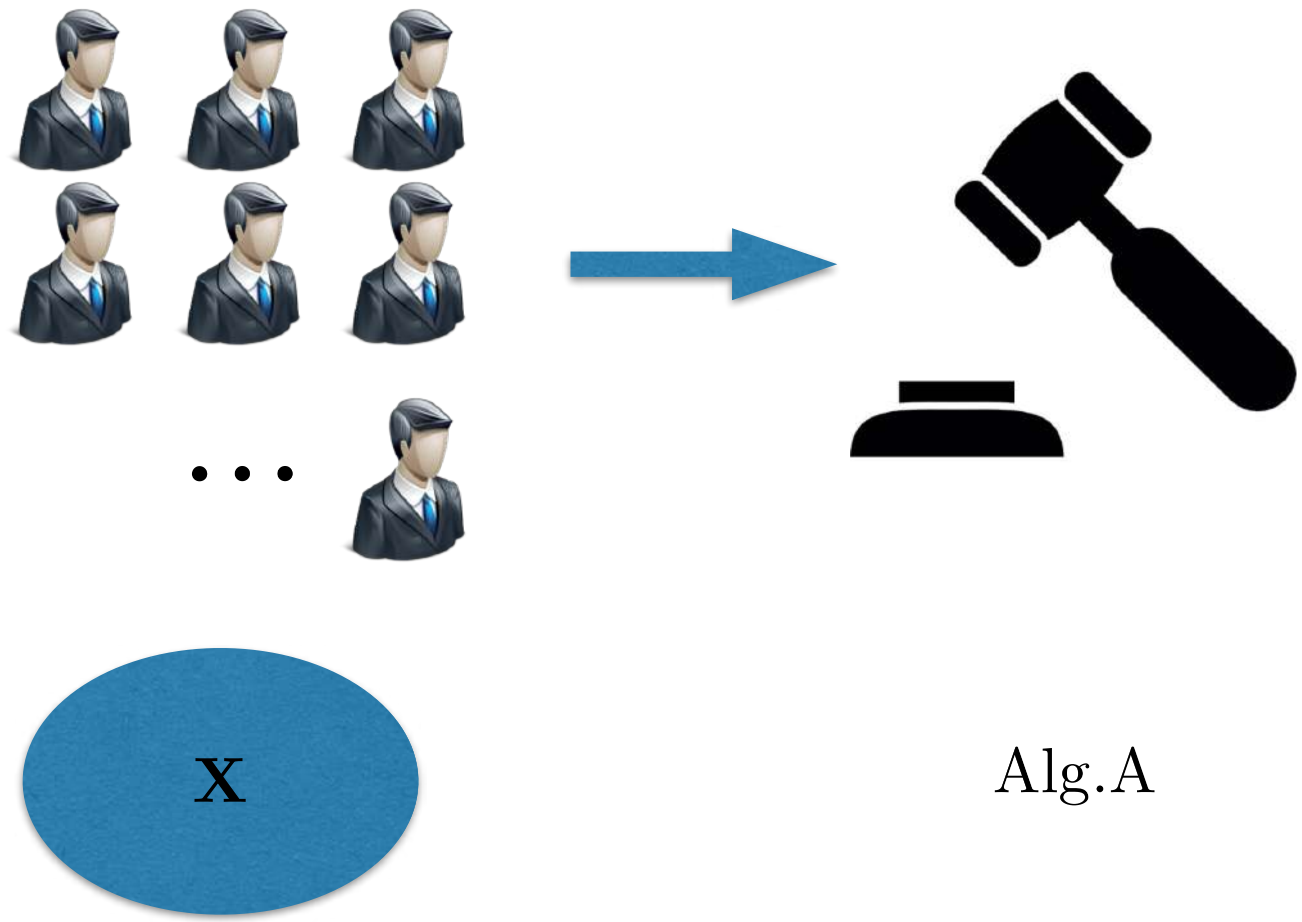
How to Decide to Accept a Paper?

- Use all reviewers that have bid for the paper to review



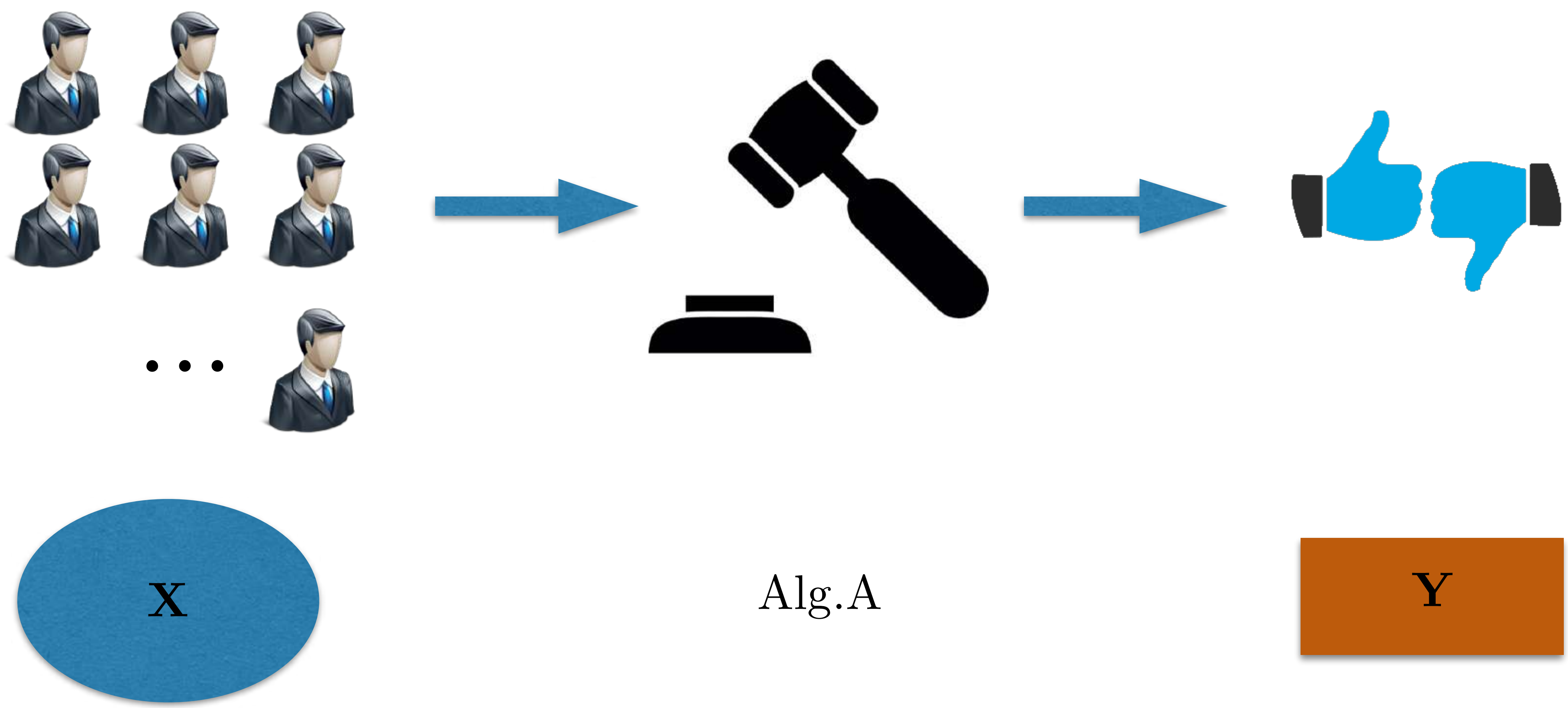
How to Decide to Accept a Paper?

- Use all reviewers that have bid for the paper to decide



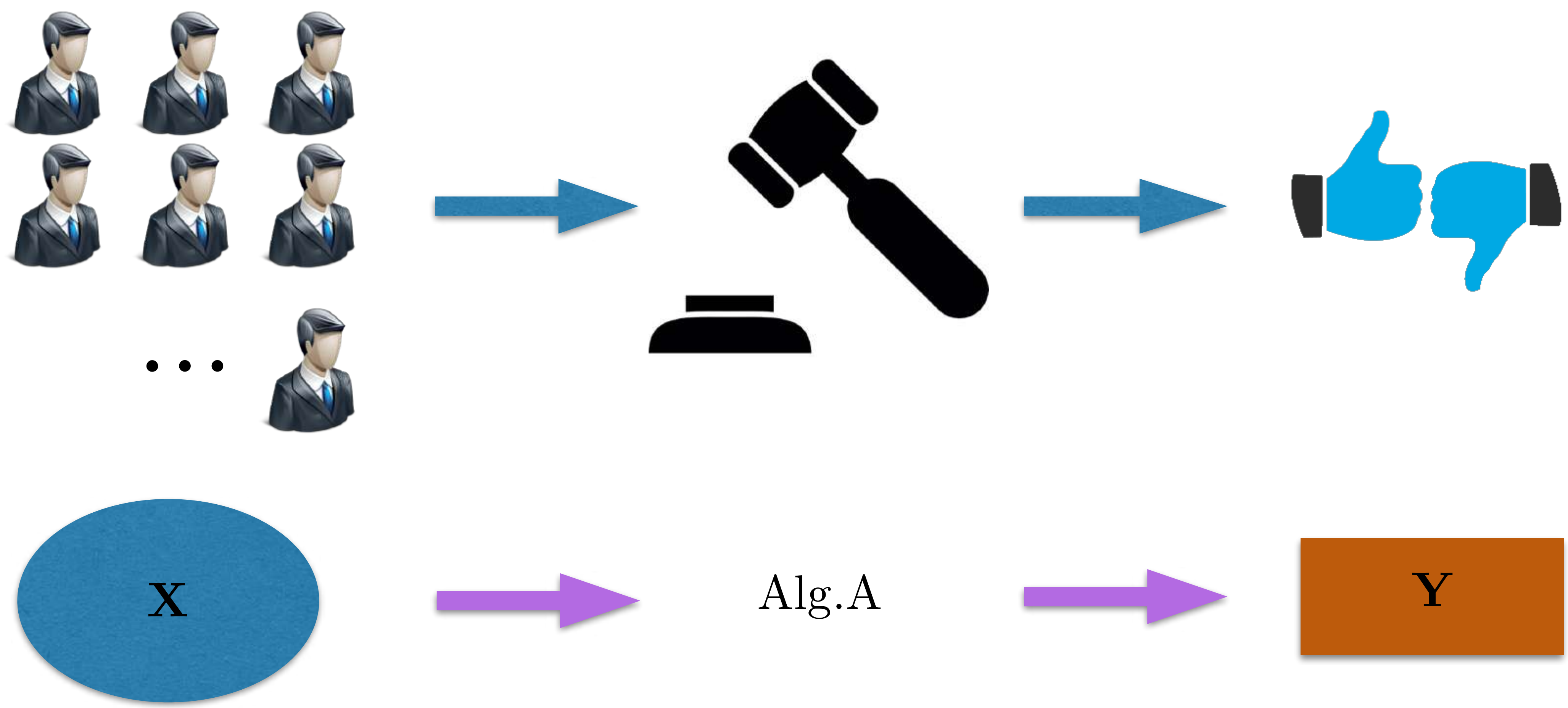
How to Decide to Accept a Paper?

- Use all reviewers that have bid to make a final decision



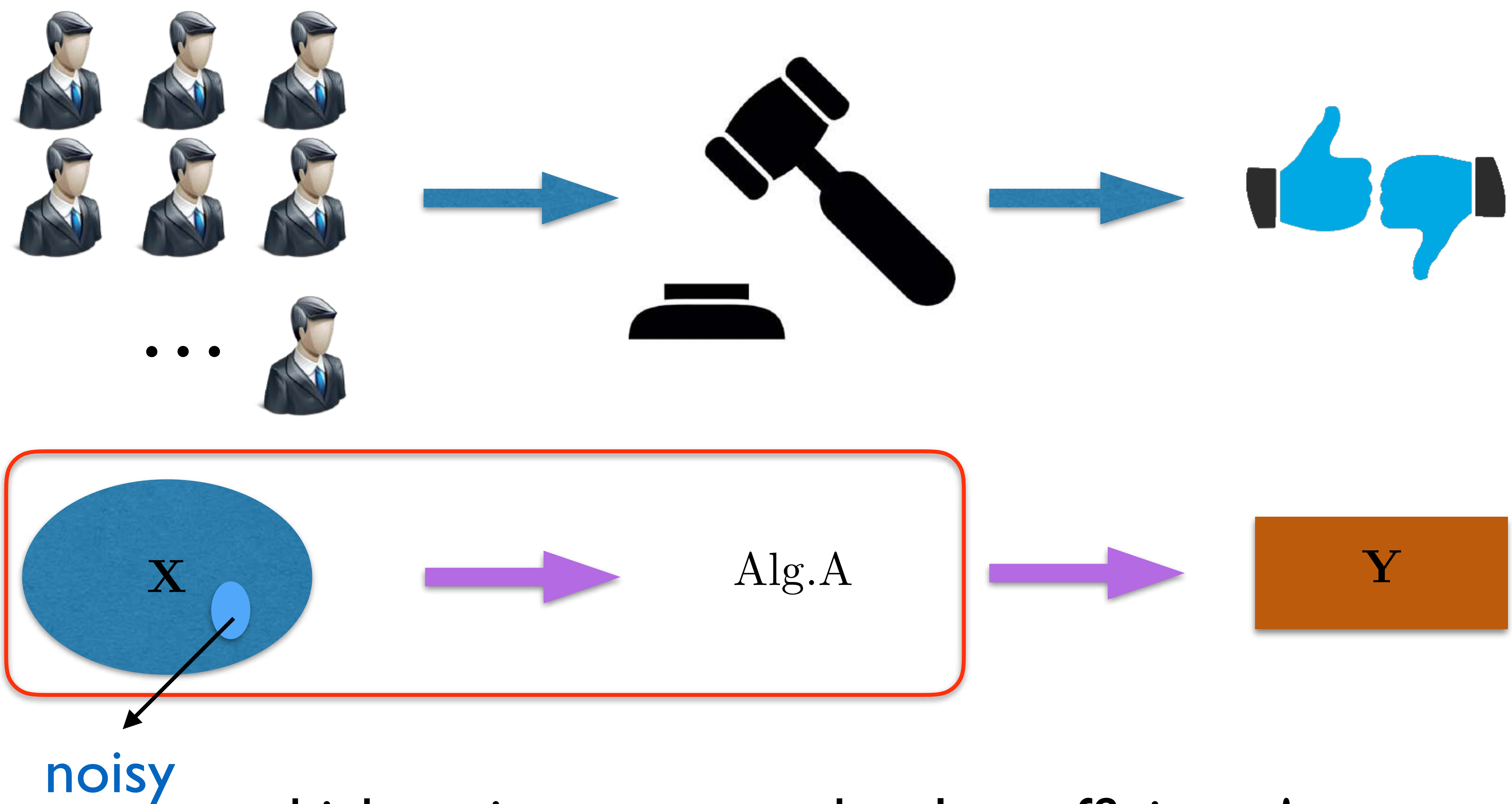
How to Decide to Accept a Paper?

- Use all reviewers that have bid to make a final decision



How to Decide to Accept a Paper?

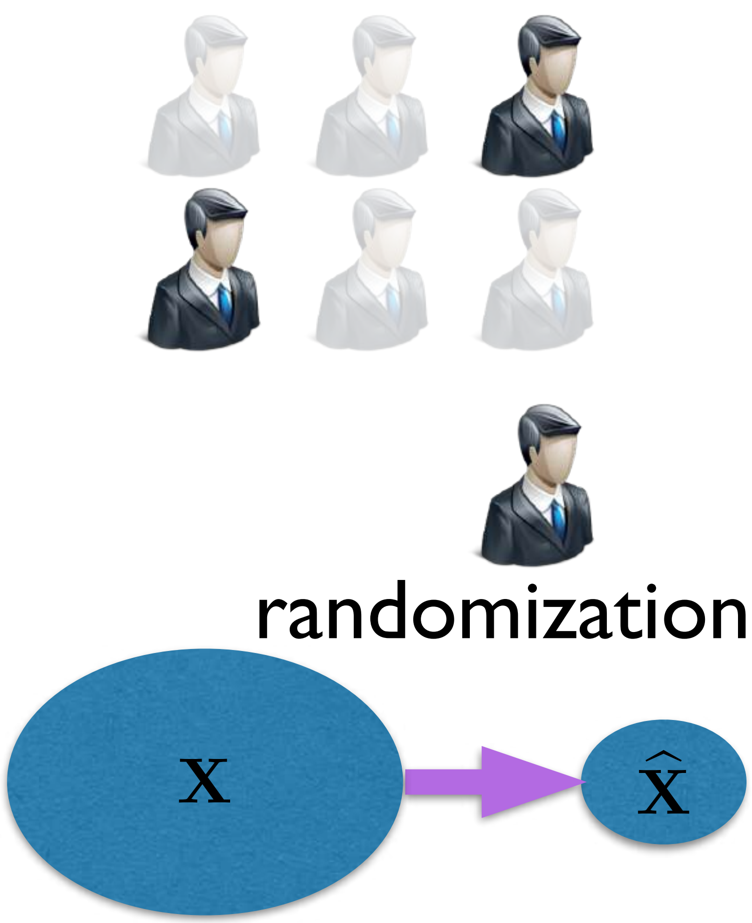
- Use all reviewers that have bid to make a final decision



high review accuracy, but low efficiency!

How to Decide to Accept a Paper?

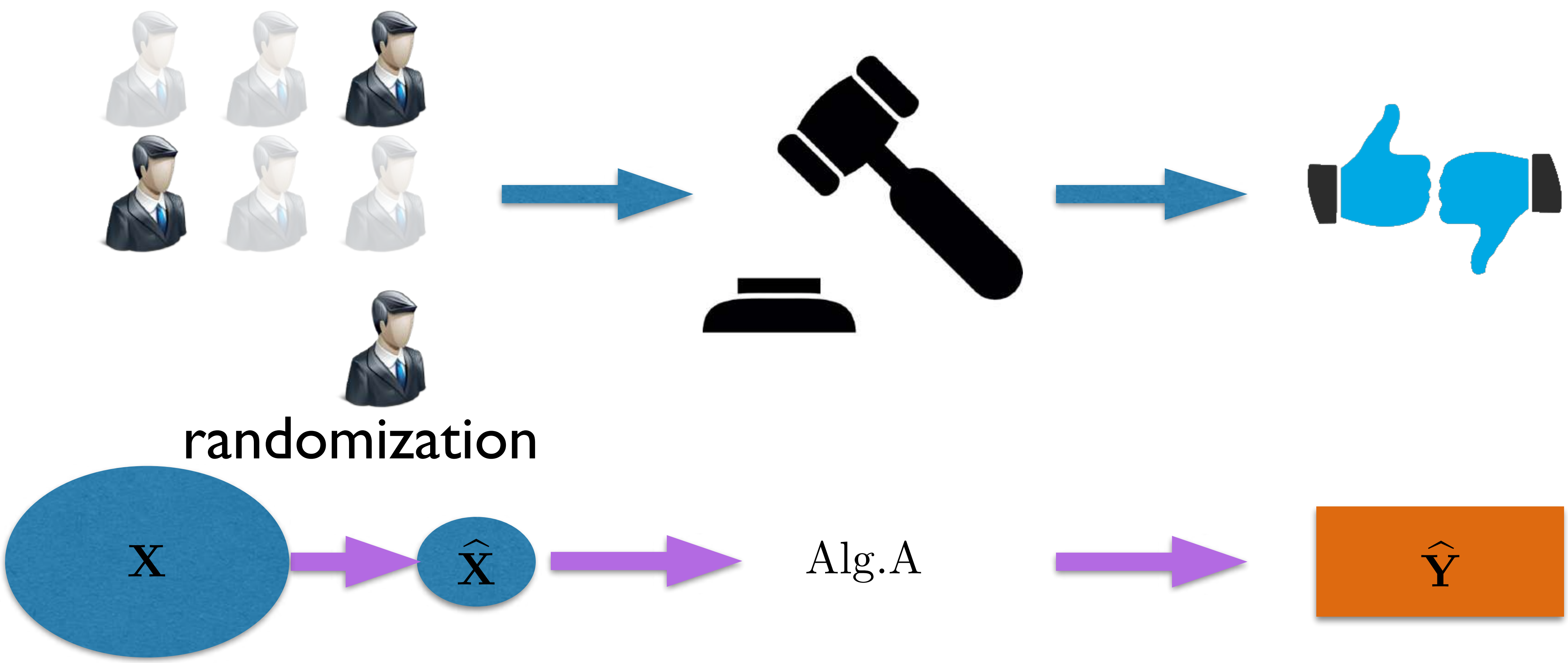
- Randomly sample from reviewers that have bid to review



Randomization: the process of making something random (e.g., random sampling)

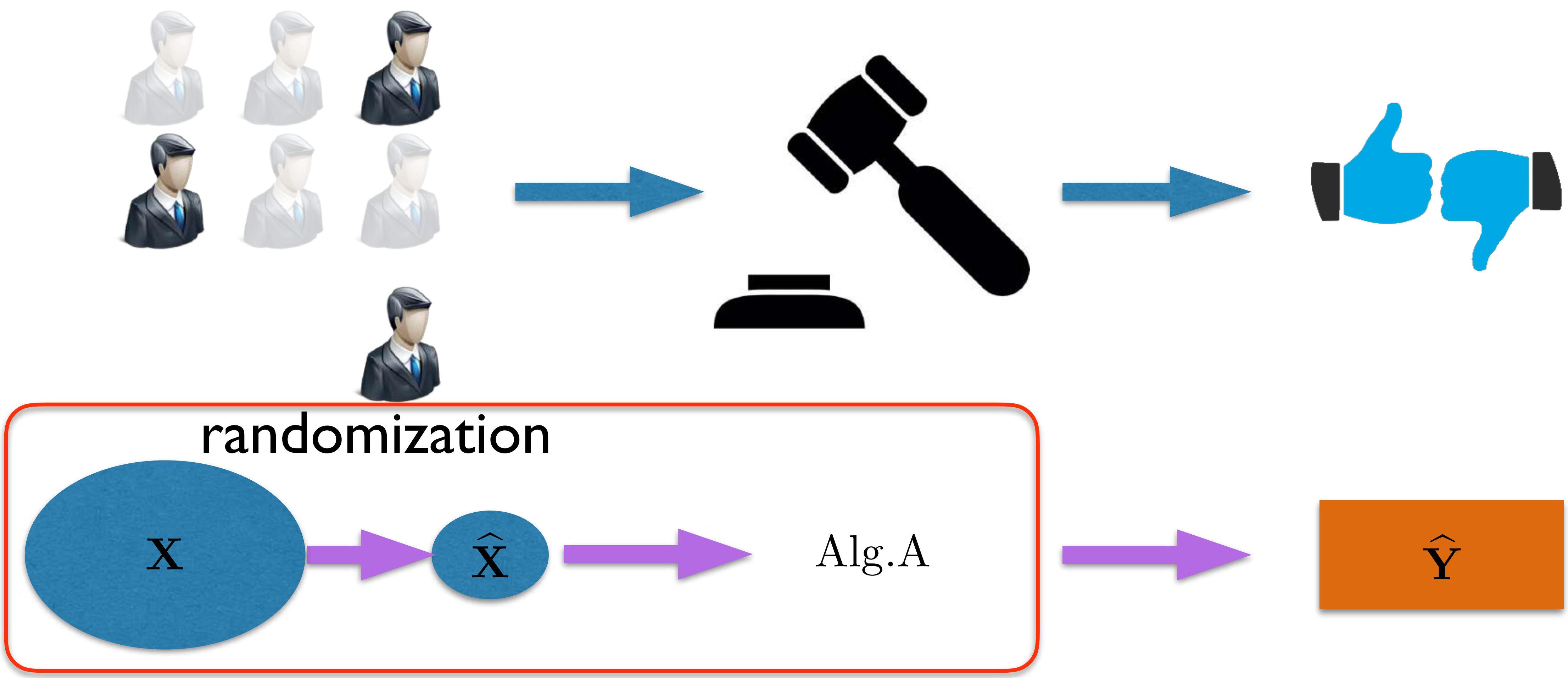
How to Decide to Accept a Paper?

- Randomly sample from reviewers that have bid to decide



How to Decide to Accept a Paper?

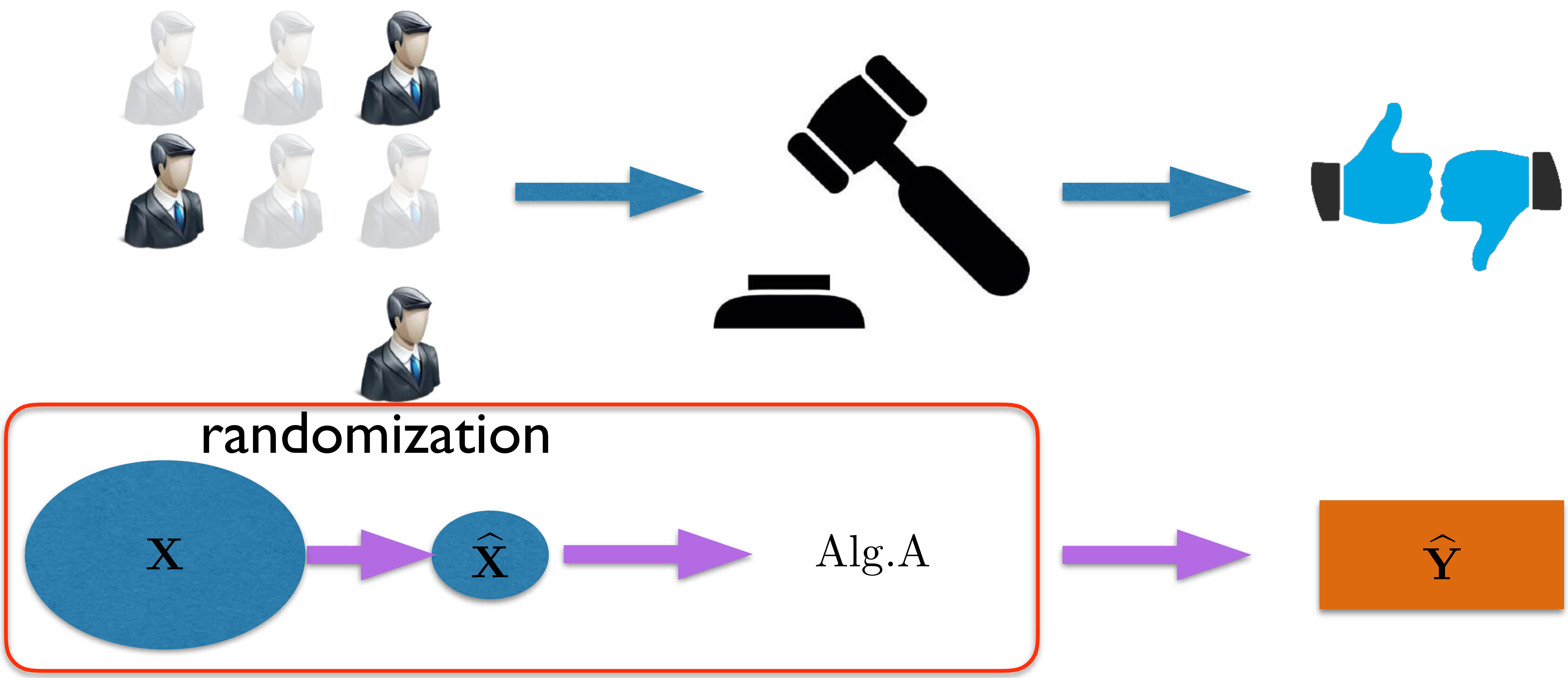
- Randomly sample from reviewers that have bid to make a final decision



Randomized Algorithm (RA): randomization is used additionally to perturb the input and reduce the input size for the algorithm execution

How to Decide to Accept a Paper?

- Randomly sample from reviewers that have bid to make a final decision



high efficiency;
high accuracy? i.e., $\widehat{\mathrm{Y}} \to \mathrm{Y}$?

Randomized Algorithm Helps?

	Efficiency	Accuracy
All reviewers	Low	High
Selected reviewers	High	?

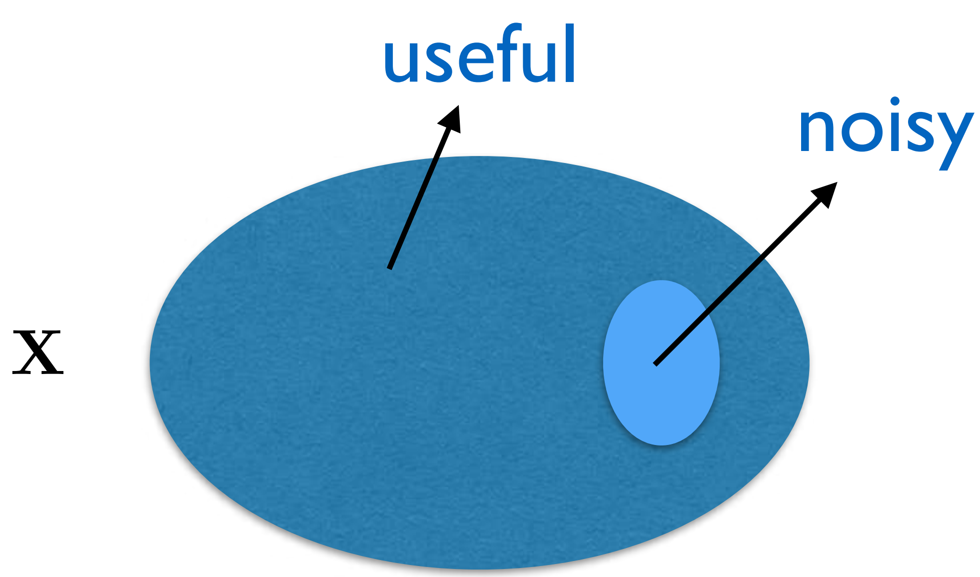
Randomized Algorithm Helps?

- Reviewers mark papers

	P1	P2	P3	P4
R1	+1	+2	-1	-1
R2	-2	+1	+2	-1
R3	-2	-1	-2	-1
R4	+2	-1	+2	+2
R1,2,3,4	-	+	+	-

	P1	P2	P3	P4
R1	+1	+2	-1	-1
R2	-2	+1	+2	-1
R3	-2	-1	-2	-1
R1,2,3	-	+	-	-

ground truth



Randomized Algorithm Helps?

- Reviewers mark papers

	P1	P2	P3	P4
R1	+1	+2	-1	-1
R2	-2	+1	+2	-1
R3	-2	-1	-2	-1
R4	+2	-1	+2	+2
R1,2,3,4	-	+	+	-

	P1	P2	P3	P4
R1	+1	+2	-1	-1
R2	-2	+1	+2	-1
R3	-2	-1	-2	-1
R1,2,3	-	+	-	-

ground truth

	P1	P2	P3	P4	Accuracy
R1	+	+	-	-	3/4
R2	-	+	+	-	3/4
R3	-	-	-	-	3/4
R4	+	-	+	+	0/4
R1,2	-	+	+	-	3/4
R1,3	-	+	-	-	4/4
R1,4	+	+	+	+	1/4
R2,3	-	0	0	-	3/4
R2,4	0	0	+	+	1/4
R3,4	0	-	0	+	1/4

random sampling

Randomized Algorithm Helps?

- NIPS'14 review experiment
  - Half the papers appearing at NIPS are still kept if the review process were rerun

Randomized Algorithm Helps?

- To improve the accuracy
  - Assign more reviewers (enlarge the problem size after randomization)

	P1	P2	P3	P4
R1,2	-	+	+	-
R1,2,3	-	+	-	-

Randomized Algorithm Helps?

- To improve the accuracy
  - Assign more reviewers (enlarge the problem size after randomization)

	P1	P2	P3	P4
R1,2	-	+	+	-
R1,2,3	-	+	-	-

  - Ensure a known expert in the review process in NIPS'16 (design more complicated randomization techniques)

	P1	P2	P3	P4
R1	+	+	-	-
R3	-	-	-	-
R4	+	-	+	+
R1,3	-	+	-	-
R1,4	+	+	+	+

Randomized Algorithm Helps?

- To improve the accuracy
  - Assign more reviewers (enlarge the problem size after randomization)

	P1	P2	P3	P4
R1,2	-	+	+	-
R1,2,3	-	+	-	-

  - Ensure a known expert in the review process in NIPS'16 (design more complicated randomization techniques)

	P1	P2	P3	P4
R1	+	+	-	-
R3	-	-	-	-
R4	+	-	+	+
R1,3	-	+	-	-
R1,4	+	+	+	+

although will decrease the achieved efficiency!

Randomized Algorithm Helps!

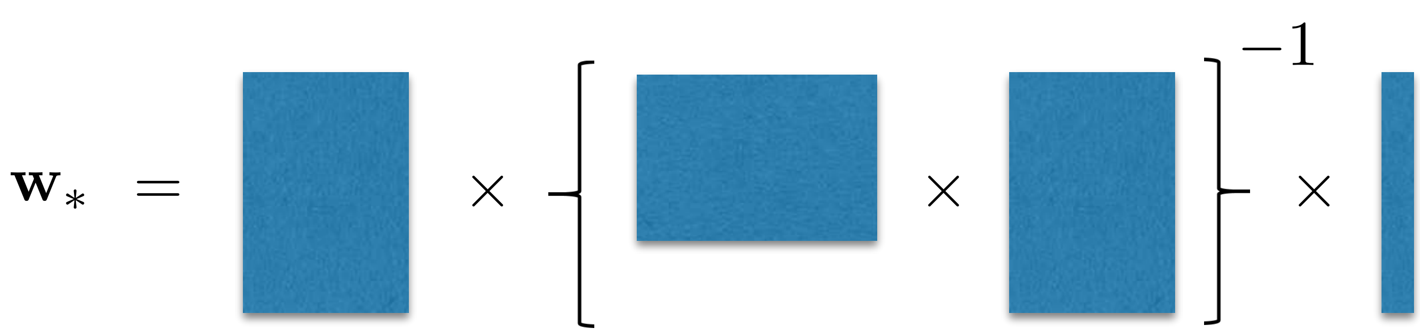
- A tradeoff between accuracy and efficiency in the algorithm design
- Reduce the computational requirements with good outputs

Randomized Algorithm on Learning

- Solving learning problems involves matrix computations

$$\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T, \mathbf{X} \in \mathbb{R}^{d \times n}$$

covariance estimation



$$\mathbf{w}_* = \arg\min_{\mathbf{w} \in \mathbb{R}^d} \|\mathbf{X}^T\mathbf{w} - \mathbf{b}\|_2^2, \ \mathbf{X} \in \mathbb{R}^{d \times n}$$

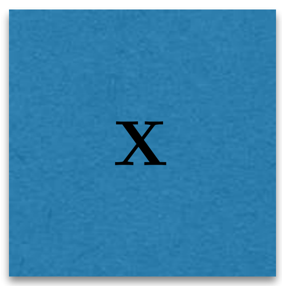
least square regression

Randomized Algorithm on Learning

- Randomization is utilized to obtain a smaller or sparser matrix that represents the essential information in the original matrix for the algorithm execution

Randomized Algorithm on Learning

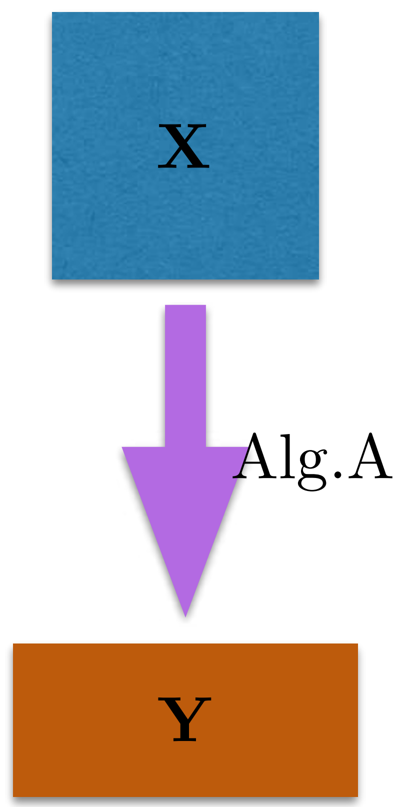
- Randomization is utilized to obtain a smaller or sparser matrix that represents the essential information in the original matrix for the algorithm execution



data matrix

Randomized Algorithm on Learning

- Randomization is utilized to obtain a smaller or sparser matrix that represents the essential information in the original matrix for the algorithm execution



Randomized Algorithm on Learning

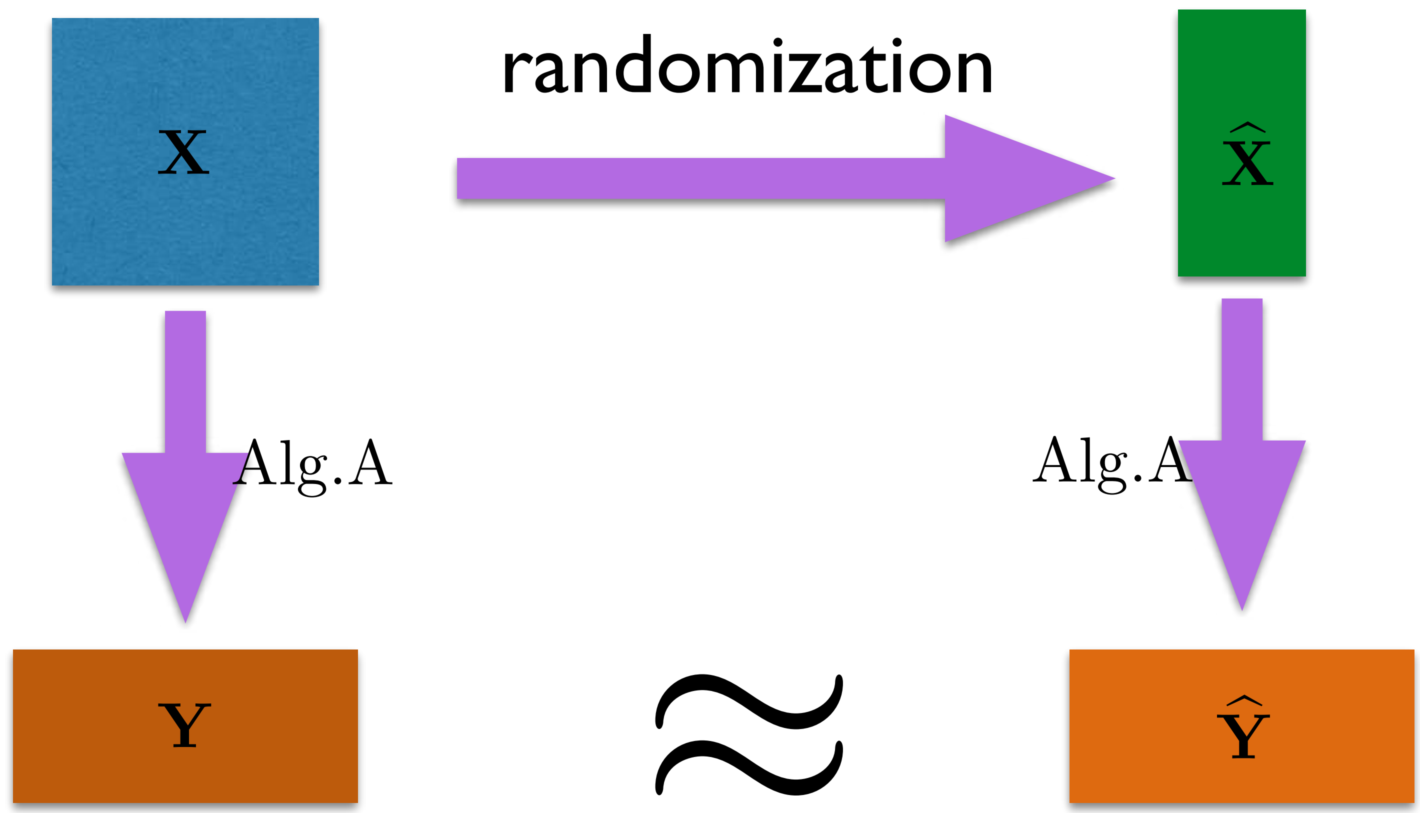
- Randomization is utilized to obtain a smaller or sparser matrix that represents the essential information in the original matrix for the algorithm execution

Randomized Algorithm on Learning

- Randomization is utilized to obtain a smaller or sparser matrix that represents the essential information in the original matrix for the algorithm execution

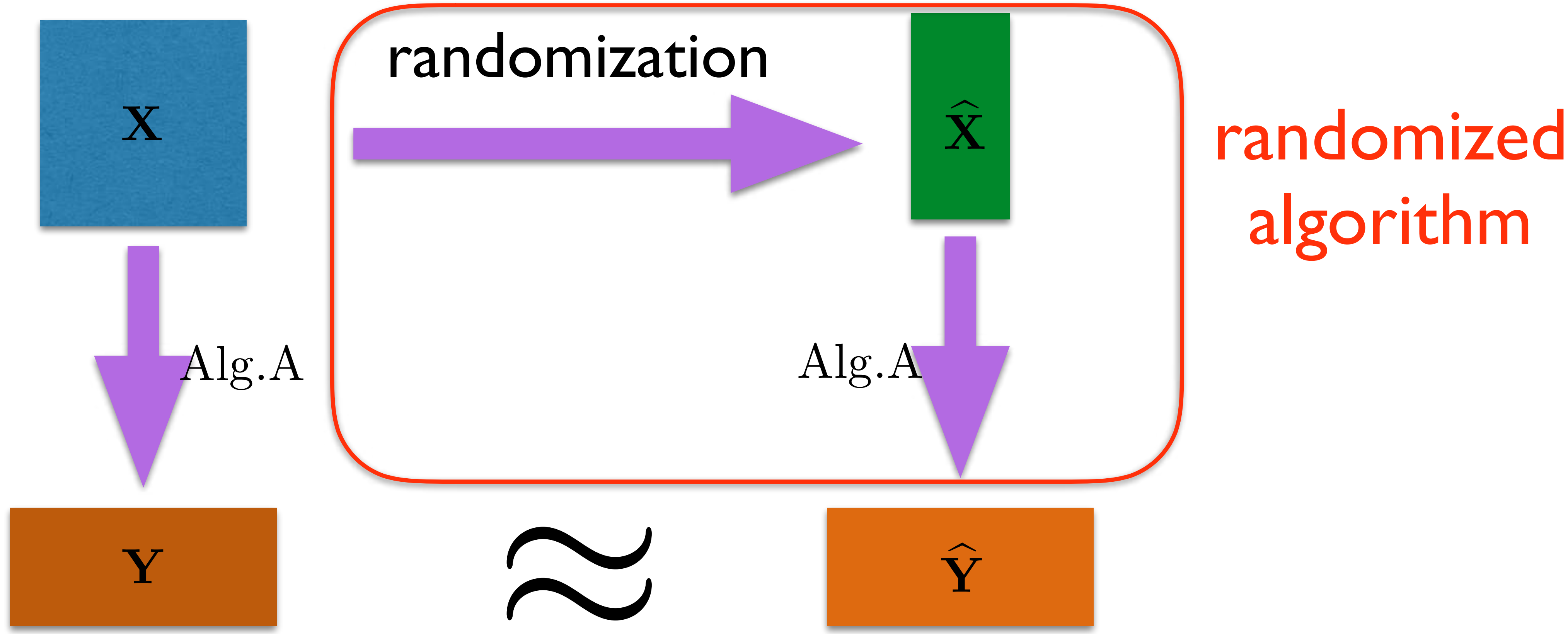
Randomized Algorithm on Learning

- Randomization is utilized to obtain a smaller or sparser matrix that represents the essential information in the original matrix for the algorithm execution



Randomized Algorithm on Learning

- Randomization is utilized to obtain a smaller or sparser matrix that represents the essential information in the original matrix for the algorithm execution

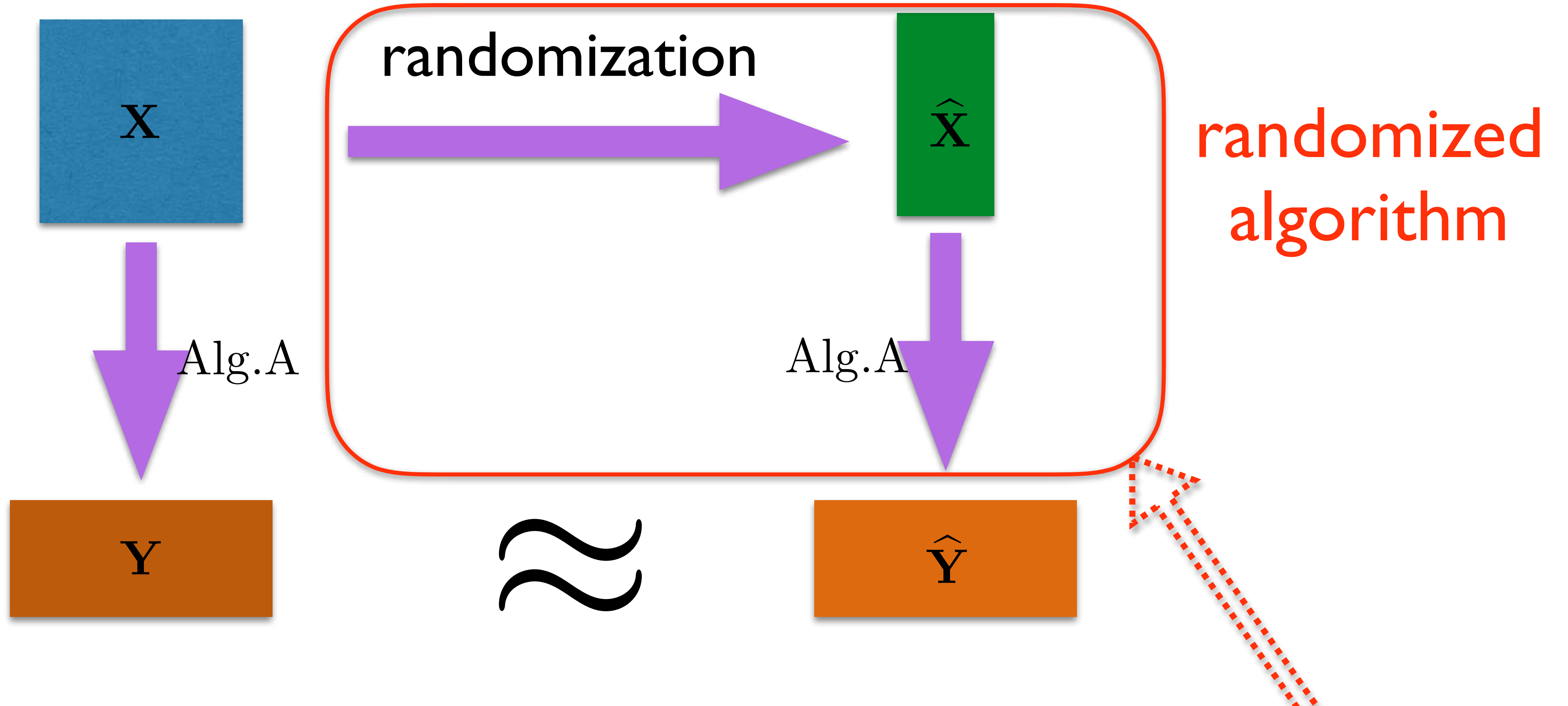


Goal: $\mathbb{P}\{\text{Difference}(\mathbf{Y}, \widehat{\mathbf{Y}}) \leq \epsilon\} \geq 1 - \delta$ holds in a low computational burden!

[M. Mahoney, 2011; T. Yang, 2015]

Randomized Algorithm on Learning

- Randomization is utilized to obtain a smaller or sparser matrix that represents the essential information in the original matrix for the algorithm execution



Goal: $\mathbb{P}\{\text{Difference}(\mathbf{Y}, \hat{\mathbf{Y}}) \leq \epsilon\} \geq 1 - \delta$ holds in a low computational burden!

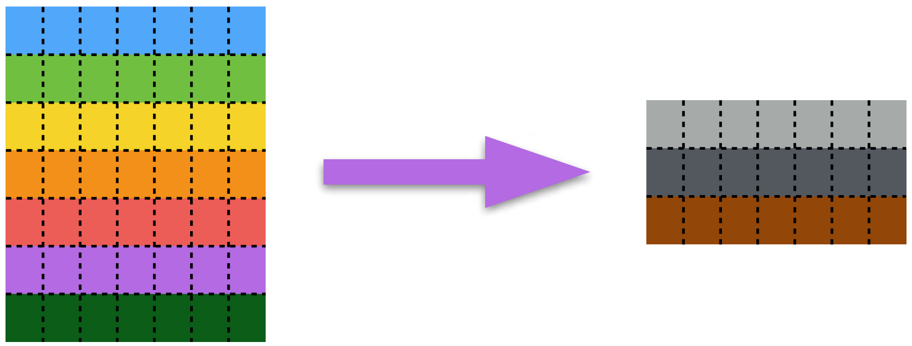
[M. Mahoney, 2011; T. Yang, 2015]

How to Get a Good Randomized Algorithm

- Randomization greatly impacts the accuracy and efficiency:
  - Random projection
  - Random sampling

Random Projection

- Randomly combine rows/columns of data matrix to create a smaller representation



- JL-lemma [Johnson & Lindenstrauss, 1984]
  - Assume $0 < \epsilon, \delta < 1$ and $m = \Omega(\epsilon^{-2} \log(\frac{1}{\delta}))$. There exists a probability distribution on an real matrix $\mathbf{\Phi} \in \mathbb{R}^{m \times d}$. Then, for any fixed vector $\mathbf{x} \in \mathbb{R}^d$ with a probability at least $1 - \delta$, we have

$$(1 - \epsilon)\|\mathbf{x}\|_2^2 \leq \|\mathbf{\Phi x}\|_2^2 \leq (1 + \epsilon)\|\mathbf{x}\|_2^2$$

Random Projection

- $\mathbf{\Phi} \in \mathbb{R}^{m \times d}$: Gaussian matrix [S. Dasgupta, et al., 2003]
  - Satisfy $\phi_{ij} \sim \mathcal{N}(0,1)/\sqrt{m}$
  - Take $O(mdn)$ time for $\mathbf{\Phi X}$ ($\mathbf{X} \in \mathbb{R}^{d \times n}$)

Gaussian matrix is dense, which is not very efficient!

Random Projection

- $\mathbf{\Phi} \in \mathbb{R}^{m \times d}$: sparse matrix [D. Achlioptas, 2003]
- Satisfy $\phi_{ij} = \begin{cases} \sqrt{3/m} & \text{Prob.} = 1/6 \\ 0 & \text{Prob.} = 2/3 \\ -\sqrt{3/m} & \text{Prob.} = 1/6 \end{cases}$
- Faster

Random Projection

- $\mathbf{\Phi} = \mathbf{PHD} \in \mathbb{R}^{m \times d}$: Hadamard transform [N. Ailon, et al., 2009]

  fastest for $\mathbf{\Phi X}$ ($\mathbf{X} \in \mathbb{R}^{d \times n}$): $nd \log(m)$ time

  - $\mathbf{P} \in \mathbb{R}^{m \times d}$: sparse Gaussian matrix

$$p_{ij} = \begin{cases} \mathcal{N}(0, q^{-1}) & \text{Prob.} = q \\ 0 & \text{Prob.} = 1 - q \end{cases}$$

  - $\mathbf{H} \in \mathbb{R}^{d \times d}$: normalized Walsh-Hadamard matrix (for FFT)

$$\mathbf{H} = \frac{1}{\sqrt{d}} \mathbf{H}_d, \ \mathbf{H}_d = \begin{bmatrix} \mathbf{H}_{d/2} & \mathbf{H}_{d/2} \\ \mathbf{H}_{d/2} & -\mathbf{H}_{d/2} \end{bmatrix}, \ \mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

  - $\mathbf{D} \in \mathbb{R}^{d \times d}$: diagonal matrix

$$d_{ii} = \begin{cases} 1 & \text{Prob.} = 1/2 \\ -1 & \text{Prob.} = 1/2 \end{cases}$$

Random Projection

- $\mathbf{\Phi} = \mathbf{PHD} \in \mathbb{R}^{m \times d}$: Hadamard transform [N. Ailon, et al., 2009]
  - $\mathbf{P} \in \mathbb{R}^{m \times d}$: sparse Gaussian matrix

$$p_{ij} = \begin{cases} \mathcal{N}(0, q^{-1}) & \text{Prob.} = q \\ 0 & \text{Prob.} = 1 - q \end{cases}$$

  - $\mathbf{H} \in \mathbb{R}^{d \times d}$: normalized Walsh-Hadamard matrix

$$\mathbf{H} = \frac{1}{\sqrt{d}} \mathbf{H}_d, \ \mathbf{H}_d = \begin{bmatrix} \mathbf{H}_{d/2} & \mathbf{H}_{d/2} \\ \mathbf{H}_{d/2} & -\mathbf{H}_{d/2} \end{bmatrix}, \ \mathbf{H}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

fast: $d \log(m)$ for $\Phi \mathbf{x}_i$

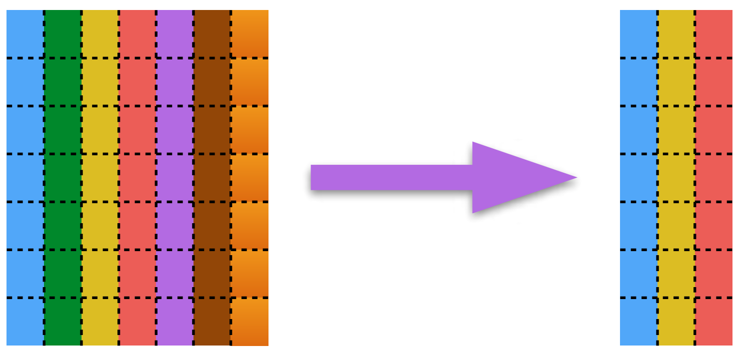
no need for storing $\mathbf{H}$

  - $\mathbf{D} \in \mathbb{R}^{d \times d}$: diagonal matrix

$$d_{ii} = \begin{cases} 1 & \text{Prob.} = 1/2 \\ -1 & \text{Prob.} = 1/2 \end{cases}$$

Random Sampling

- Randomly sample a small number of rows/columns to create a smaller matrix (interpretable, efficient)



- Choose a column $\mathbf{y}$ from $\{\mathbf{x}_i\}_{i=1}^n$ $(\mathbf{X} \in \mathbb{R}^{d \times n})$ based on the sampling probabilities $\{p_i\}_{i=1}^n : \mathbb{P}(\mathbf{y} = \mathbf{x}_i) = p_i$
- How to define p_i ?
  - Uniform: $p_i = \frac{1}{n}$
  - Non-Uniform: $p_i = \frac{\|\mathbf{x}_i\|_2^2}{\|\mathbf{X}\|_\mathrm{F}^2}$, leverage scores [P. Drineas, et al., 2006], etc.

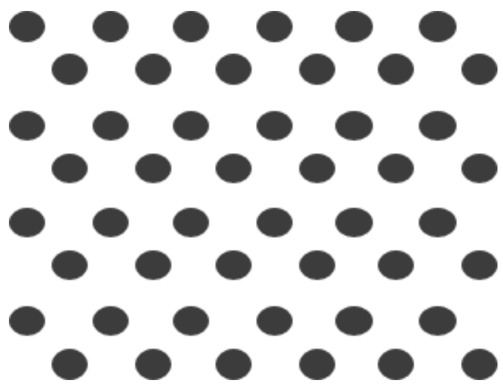
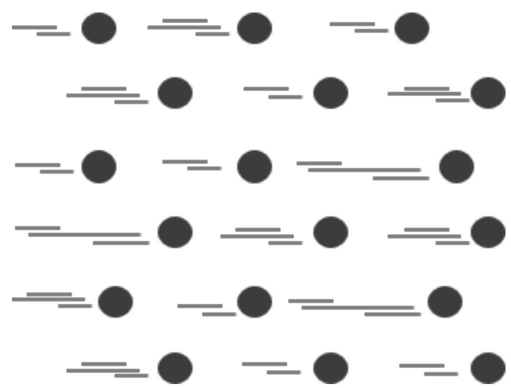
Randomized Algorithm

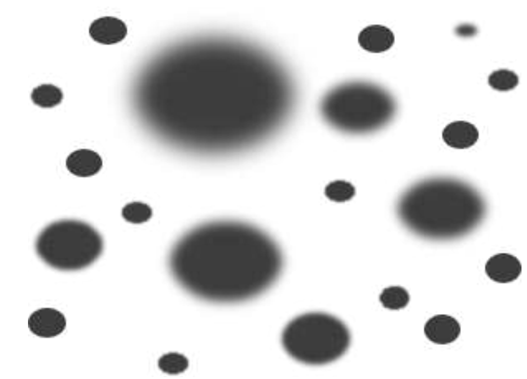
- Summary of principles:
  - Construct a sketch by randomization
    - Sketch: a smaller or sparser matrix that represents the essential information in the original matrix
  - Leverage the sketch as a surrogate for the learning
  - Theoretically analyze the learning accuracy and computational complexity

Why Randomized Algorithm

Why Randomized Algorithm

Volume	Velocity	Variety	Veracity



IDC Analyze the Future

40 ZB (2020)

5.2 TB per person

facebook

500 TB per day
new data

Why Randomized Algorithm

- Can make learning efficient [M. Mahoney, 2011]
  - Reduction in time, space, and communication

Why Randomized Algorithm

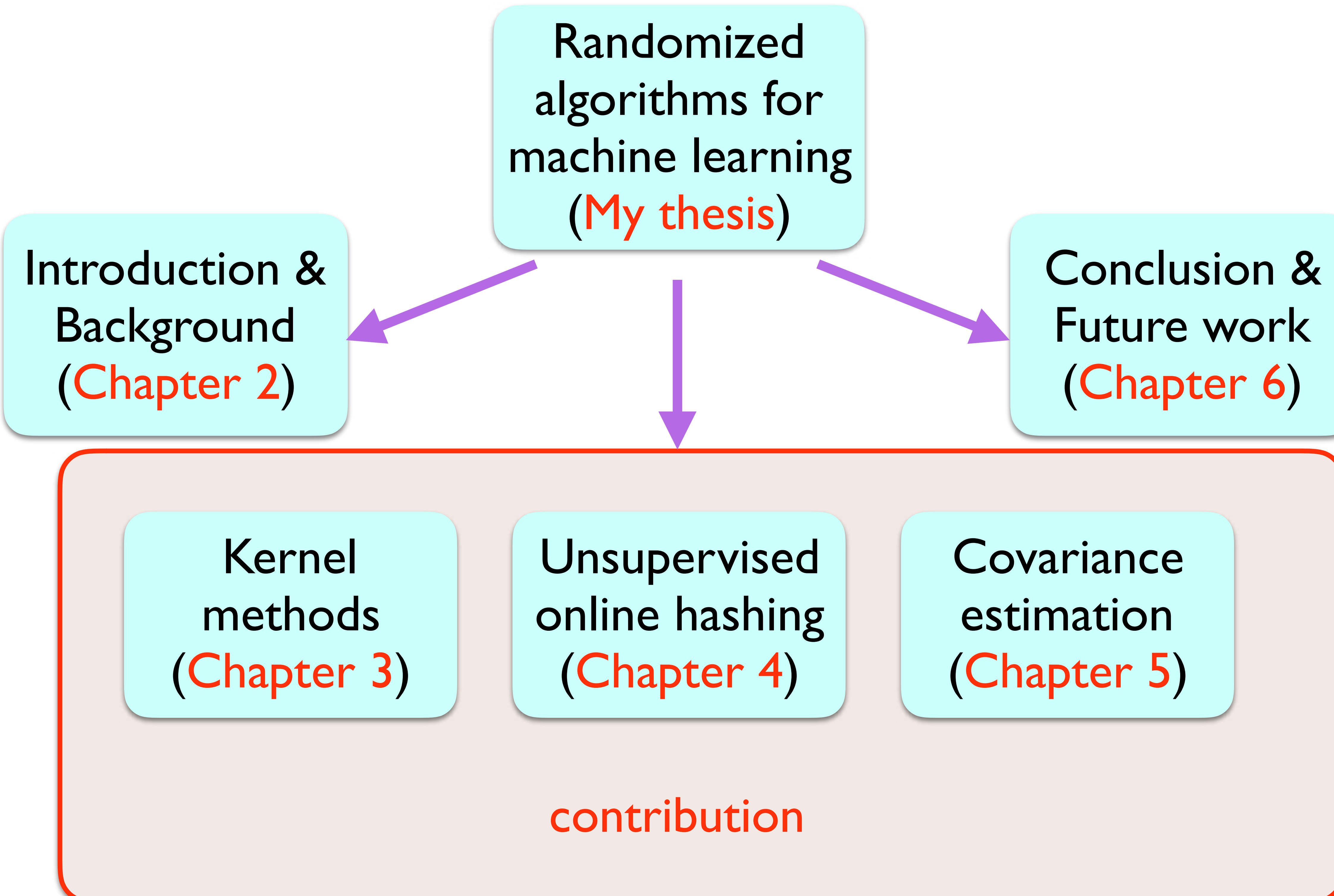
- Can make learning efficient [M. Mahoney, 2011]
  - Reduction in time, space, and communication
  - Simple
  - Effective
  - Theoretically guaranteed
  - Interpretable
  - Parallelizable

Application Taxonomy

randomized algorithms for machine learning

- **regression**
  - $n \gg d$ [O. Maillar, et al., 2009] [P. Dhillon, et al., 2013] [G. Raskutti, et al., 2015] [J. Yang, et al., 2016] [M. Slawski, et al., 2017]
  - $d \gg n$ [Y. Lu, et al., 2013] [S. Chen, et al., 2015] [S. Wang, et al., 2017]
  - $n \gg 1,\ d \gg 1$ [J. Wang, et al., 2017]
- **kernel methods**
  - kernel matrix [C. Williams, et al., 2001] [P. Drineas, et al., 2005] [T. Yang, et al., 2012] [D. G. Anderson, et al., 2015] [D. Oglic, et al., 2017]
  - kernel function [A. Rahimi, et al., 2007] [N. Pham, et al., 2013] [J. Yang, et al., 2014] [X. Chen, et al., 2015] [M. Slawski, et al., 2017] [H. Avron, et al., 2017]
- **hashing**
  - data-independent [M. Datar, et al., 2004] [P. Li, et al., 2011] [Shrivastava, et al., 2014]
  - data-dependent [F. Xu, et al., 2014] [C. Leng, et al., 2015] [Q. Jiang, et al., 2015] [W. Kang, et al., 2016] [X. Chen, et al., 2017]
- **covariance estimation** [H. Qi, et al., 2012] [F. Anaraki, et al., 2014] [M. Azizyan, et al., 2015] [F. Anaraki, et al., 2016] [F. Anaraki, et al., 2017] [X. Chen, et al., 2017]
- **deep learning**
  - after training [R. Spring, et al., 2016] [Y. Wang, et al., 2017]
  - during training [A. Novikov, et al., 2015] [W. Kang, et al., 2016] [K. Cutajar, et al., 2017]
- **k-means clustering** [C. Boutsidis, et al., 2009] [C. Boutsidis., et al., 2015] [S. Wang, et al., 2017]
- **optimization** [L. Zhang, et al., 2013] [A. Gonen, et al., 2016] [P. Xu, et al.,2016] [H. Luo, et al., 2016] [M. Pilanci, et al., 2016] [J. Wang, et al., 2017] [D. Calandriello, et al., 2017]

Outline


Randomized algorithms for machine learning (My thesis)
Introduction & Background (Chapter 2)
Conclusion & Future work (Chapter 6)
Kernel methods (Chapter 3)
Unsupervised online hashing (Chapter 4)
Covariance estimation (Chapter 5)
contribution

Thesis Contribution

- Focus on three learning techniques

Machine learning techniques	Applications
Kernel methods (Chapter 3)	regression; SVM; GP; spectral clustering
Unsupervised online hashing (Chapter 4)	retrieval; matching; clustering
Covariance estimation (Chapter 5)	LDA; QDA; regression; ICA; PCA; policy learning; gene analysis; array signal

Thesis Contribution

- Focus on three learning techniques

Machine learning techniques	Applications	Solutions
Kernel methods (Chapter 3)	regression; SVM; GP; spectral clustering	RKS [A. Rahimi, et al., 2007]
Unsupervised online hashing (Chapter 4)	retrieval; matching; clustering	OSH [C. Leng, et al., 2015]
Covariance estimation (Chapter 5)	LDA; QDA; regression; ICA; PCA; policy learning; gene analysis; array signal	Standard [W. Feller, 1966]

Thesis Contribution

- Focus on three learning techniques

Machine learning techniques	Applications	Solutions	Computational challenges
Kernel methods (Chapter 3)	regression; SVM; GP; spectral clustering	RKS [A. Rahimi, et al., 2007]	time
Unsupervised online hashing (Chapter 4)	retrieval; matching; clustering	OSH [C. Leng, et al., 2015]	time
Covariance estimation (Chapter 5)	LDA; QDA; regression; ICA; PCA; policy learning; gene analysis; array signal	Standard [W. Feller, 1966]	time; space; communication

Thesis Contribution

- Focus on three learning techniques

Machine learning techniques	Applications	Solutions	Computational challenges	Shared structures
Kernel methods (Chapter 3)	regression; SVM; GP; spectral clustering	RKS [A. Rahimi, et al., 2007]	time	$\mathbf{X}^T\mathbf{X}$ ($\mathbf{X} \in \mathbb{R}^{d \times n}$)
Unsupervised online hashing (Chapter 4)	retrieval; matching; clustering	OSH [C. Leng, et al., 2015]	time	
Covariance estimation (Chapter 5)	LDA; QDA; regression; ICA; PCA; policy learning; gene analysis; array signal	Standard [W. Feller, 1966]	time; space; communication	

Thesis Contribution

- Focus on three learning techniques

Machine learning techniques	Applications	Solutions	Computational challenges	Shared structures
Kernel methods (Chapter 3)	regression; SVM; GP; spectral clustering	RKS [A. Rahimi, et al., 2007]	time	$\mathbf{Y}^T\mathbf{Y} \approx \mathbf{X}^T\mathbf{X}$ $\mathbf{Y}?$
Unsupervised online hashing (Chapter 4)	retrieval; matching; clustering	OSH [C. Leng, et al., 2015]	time	
Covariance estimation (Chapter 5)	LDA; QDA; regression; ICA; PCA; policy learning; gene analysis; array signal	Standard [W. Feller, 1966]	time; space; communication	

randomization

Thesis Contribution

- Focus on three learning techniques

Machine learning techniques	Applications	Solutions	Computational challenges	Shared structures
Kernel methods (Chapter 3)	regression; SVM; GP; spectral clustering	RKS [A. Rahimi, et al., 2007]	time	$\mathbf{Y}^T\mathbf{Y} \approx \mathbf{X}^T\mathbf{X}$ error?
Unsupervised online hashing (Chapter 4)	retrieval; matching; clustering	OSH [C. Leng, et al., 2015]	time	
Covariance estimation (Chapter 5)	LDA; QDA; regression; ICA; PCA; policy learning; gene analysis; array signal	Standard [W. Feller, 1966]	time; space; communication	

Thesis Contribution

- Focus on three learning techniques

Machine learning techniques	Applications	Solutions	Computational challenges	Shared structures
Kernel methods (Chapter 3)	regression; SVM; GP; spectral clustering	RKS [A. Rahimi, et al., 2007]	time	$\mathbf{Y}^T\mathbf{Y} \approx \mathbf{X}^T\mathbf{X}$
Unsupervised online hashing (Chapter 4)	retrieval; matching; clustering	OSH [C. Leng, et al., 2015]	time	error?
Covariance estimation (Chapter 5)	LDA; QDA; regression; ICA; PCA; policy learning; gene analysis; array signal	Standard [W. Feller, 1966]	time; space; communication	

propagate

Thesis Contribution

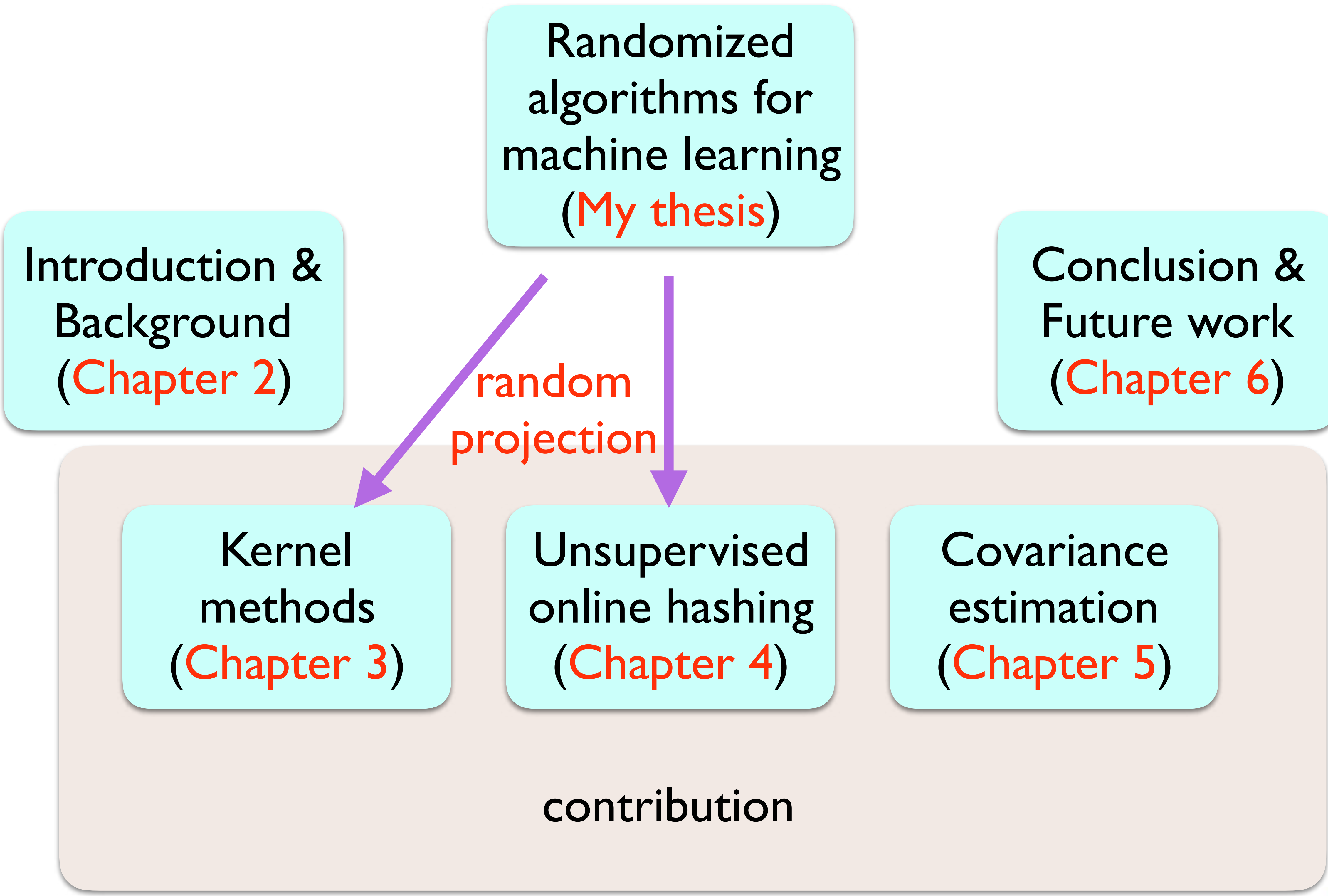
- Focus on three learning techniques

Machine learning techniques	Applications	Solutions	Computational challenges	Shared structures	Different settings
Kernel methods (Chapter 3)	regression; SVM; GP; spectral clustering	RKS [A. Rahimi, et al., 2007]	time		$d \ll n$
Unsupervised online hashing (Chapter 4)	retrieval; matching; clustering	OSH [C. Leng, et al., 2015]	time	$\mathbf{X}^T\mathbf{X}$ $(\mathbf{X} \in \mathbb{R}^{d \times n})$	streaming; fixed memory space; $1 \ll d \ll n$
Covariance estimation (Chapter 5)	LDA; QDA; regression; ICA; PCA; policy learning; gene analysis; array signal	Standard [W. Feller, 1966]	time; space; communication		distributed; streaming

Thesis Contribution

- Design randomized algorithms to reduce the computational costs
- Theoretically analyze the accuracy and efficiency
- Empirically demonstrate the good performance

Outline


Randomized algorithms for machine learning (My thesis)
Introduction & Background (Chapter 2)
Conclusion & Future work (Chapter 6)
random projection
Kernel methods (Chapter 3)
Unsupervised online hashing (Chapter 4)
Covariance estimation (Chapter 5)
contribution

Outline

Randomized algorithms for machine learning (My thesis)

Introduction & Background (Chapter 2)

random sampling

Conclusion & Future work (Chapter 6)

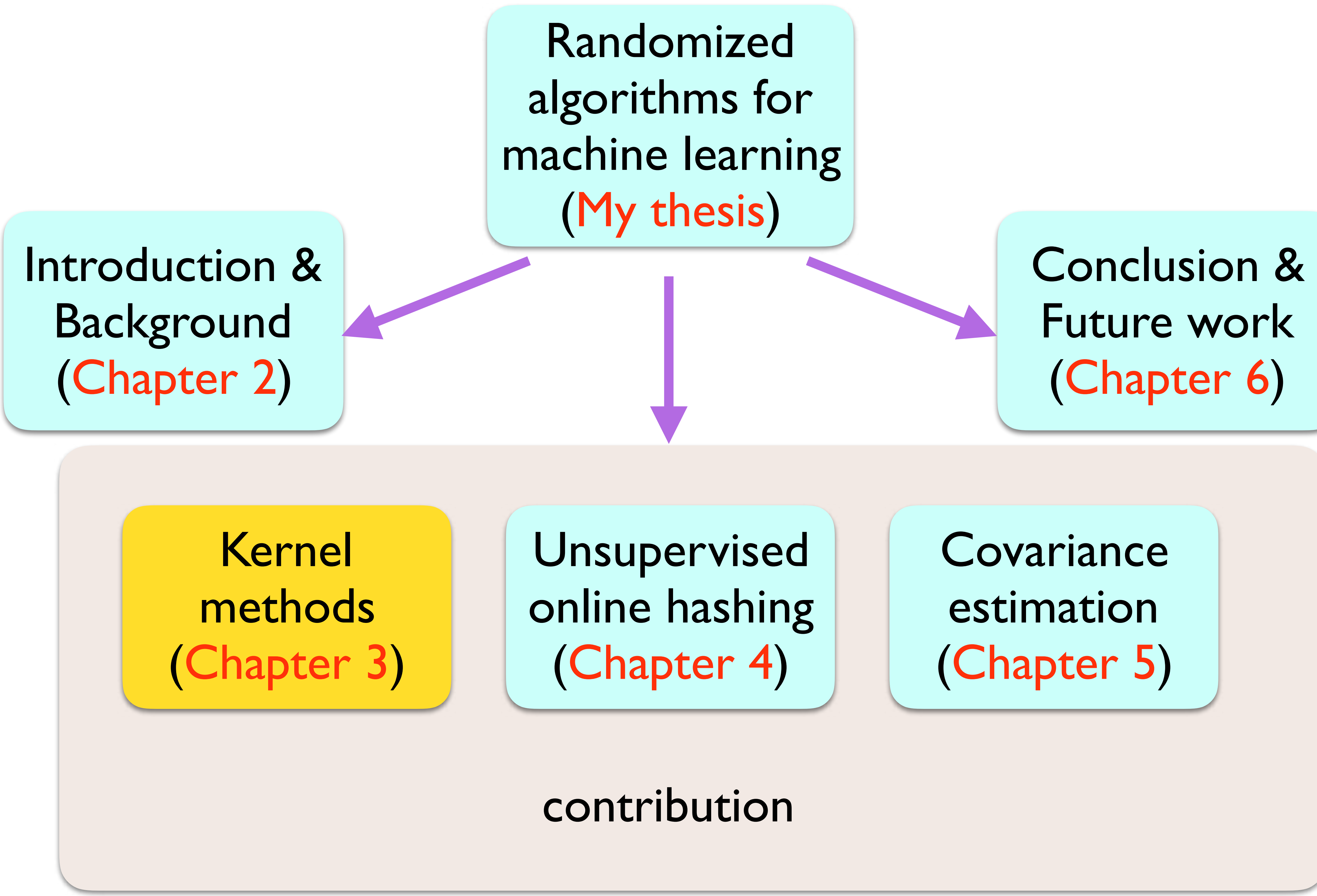
Kernel methods (Chapter 3)

Unsupervised online hashing (Chapter 4)

Covariance estimation (Chapter 5)

contribution

Outline


Randomized algorithms for machine learning (My thesis)
Introduction & Background (Chapter 2)
Conclusion & Future work (Chapter 6)
Kernel methods (Chapter 3)
Unsupervised online hashing (Chapter 4)
Covariance estimation (Chapter 5)
contribution

Background

- Kernel methods
  - Kernel regression, kernel SVM, kernel PCA, etc.
  - Kernel function: $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle, \ \forall i, j \in [n]$, without knowing $\Phi(\cdot)$

Background

- Kernel methods
  - Kernel regression, kernel SVM, kernel PCA, etc.
  - Kernel function: $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle, \forall i, j \in [n]$, without knowing $\Phi(\cdot)$
  - Shift-invariant kernel function: $k(\mathbf{x}_i, \mathbf{x}_j) = g(\mathbf{x}_i - \mathbf{x}_j)$
    e.g., $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / 2\sigma^2)$

Background

- Kernel methods
  - Kernel regression, kernel SVM, kernel PCA, etc.
  - Kernel function: $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle, \forall i, j \in [n]$, without knowing $\Phi(\cdot)$
  - Shift-invariant kernel function: $k(\mathbf{x}_i, \mathbf{x}_j) = g(\mathbf{x}_i - \mathbf{x}_j)$
    e.g., $k(\mathbf{x}_i, \mathbf{x}_j) = \exp(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / 2\sigma^2)$

powerful but inefficient

Related Work

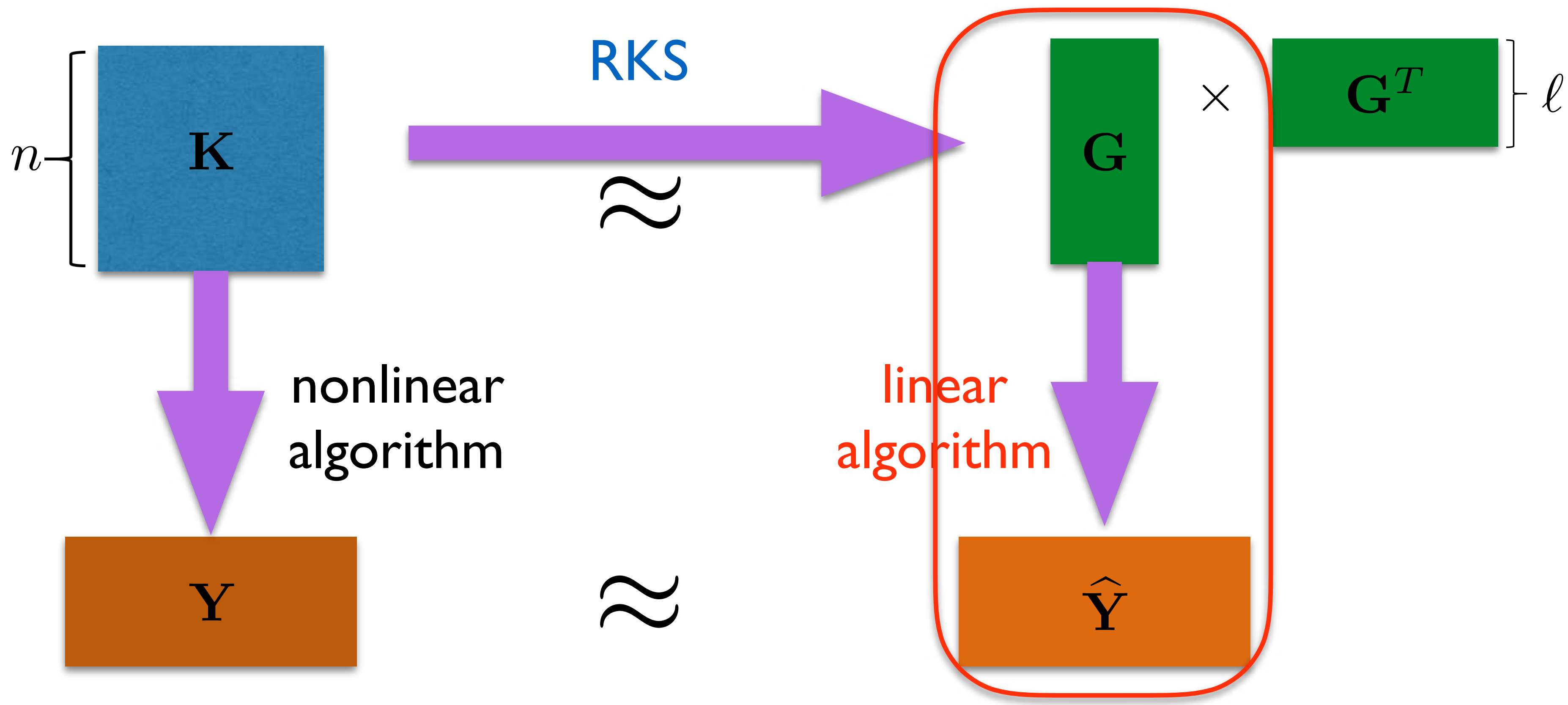
- Random Kitchen Sink (RKS) [A. Rahimi, et al., 2007]
  - Explicitly mapped features $\mathbf{G} = \{\mathbf{Z}(\mathbf{x}_i) \in \mathbb{R}^\ell\}_{i=1}^n$, satisfying

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \approx \langle \mathbf{Z}(\mathbf{x}_i), \mathbf{Z}(\mathbf{x}_j) \rangle, \ \mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^m$$

Related Work

- Random Kitchen Sink (RKS) [A. Rahimi, et al., 2007]
  - Explicitly mapped features $\mathbf{G} = \{\mathbf{Z}(\mathbf{x}_i) \in \mathbb{R}^\ell\}_{i=1}^n$, satisfying

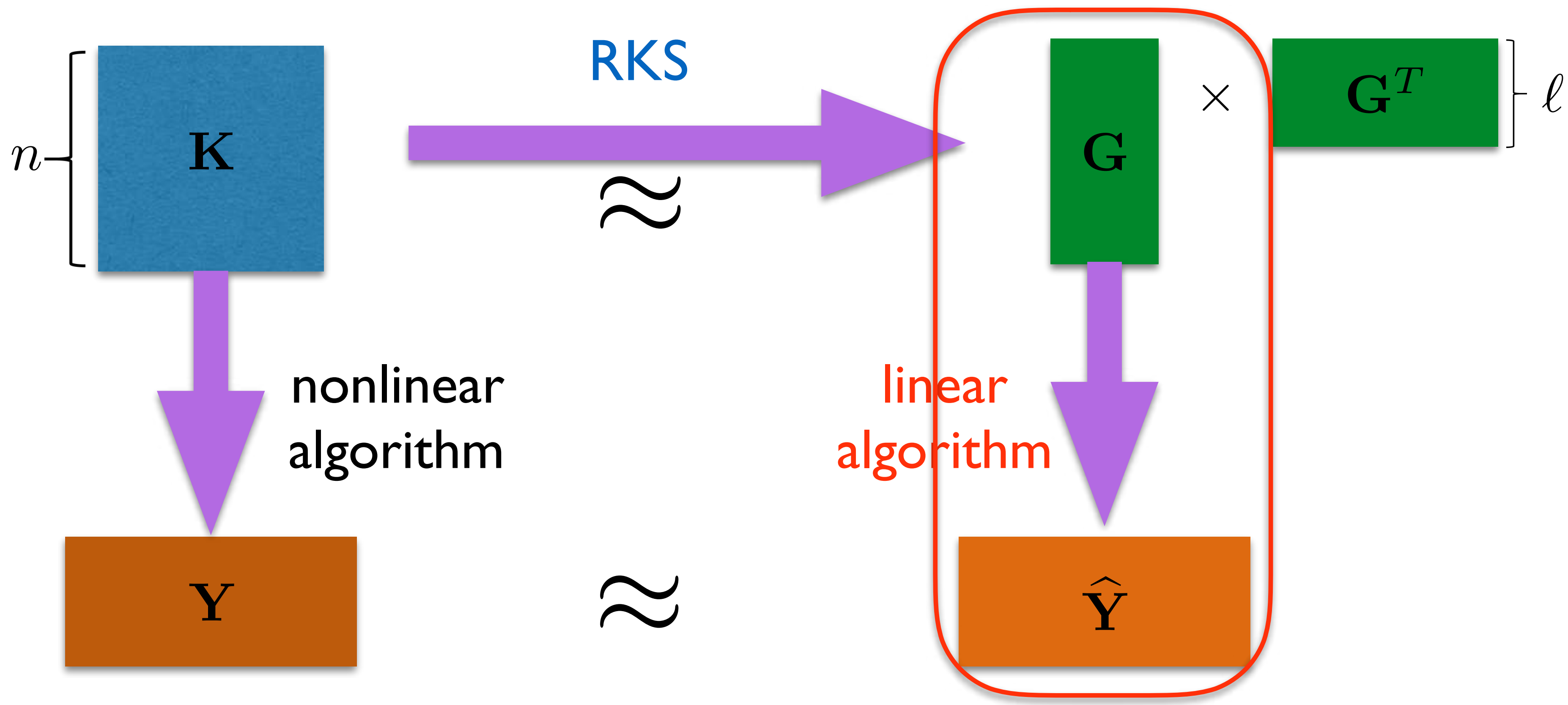
$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \approx \langle \mathbf{Z}(\mathbf{x}_i), \mathbf{Z}(\mathbf{x}_j) \rangle, \ \mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^m$$



Related Work

- Random Kitchen Sink (RKS) [A. Rahimi, et al., 2007]
  - Explicitly mapped features $\mathbf{G} = \{\mathbf{Z}(\mathbf{x}_i) \in \mathbb{R}^\ell\}_{i=1}^n$, satisfying

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}_j) \rangle \approx \langle \mathbf{Z}(\mathbf{x}_i), \mathbf{Z}(\mathbf{x}_j) \rangle, \ \mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^m$$



$$\|\mathbf{K} - \mathbf{G}\mathbf{G}^T\|_2 \leq O(n/\sqrt{\ell})$$

a large ℓ for accurate training, still inefficient!

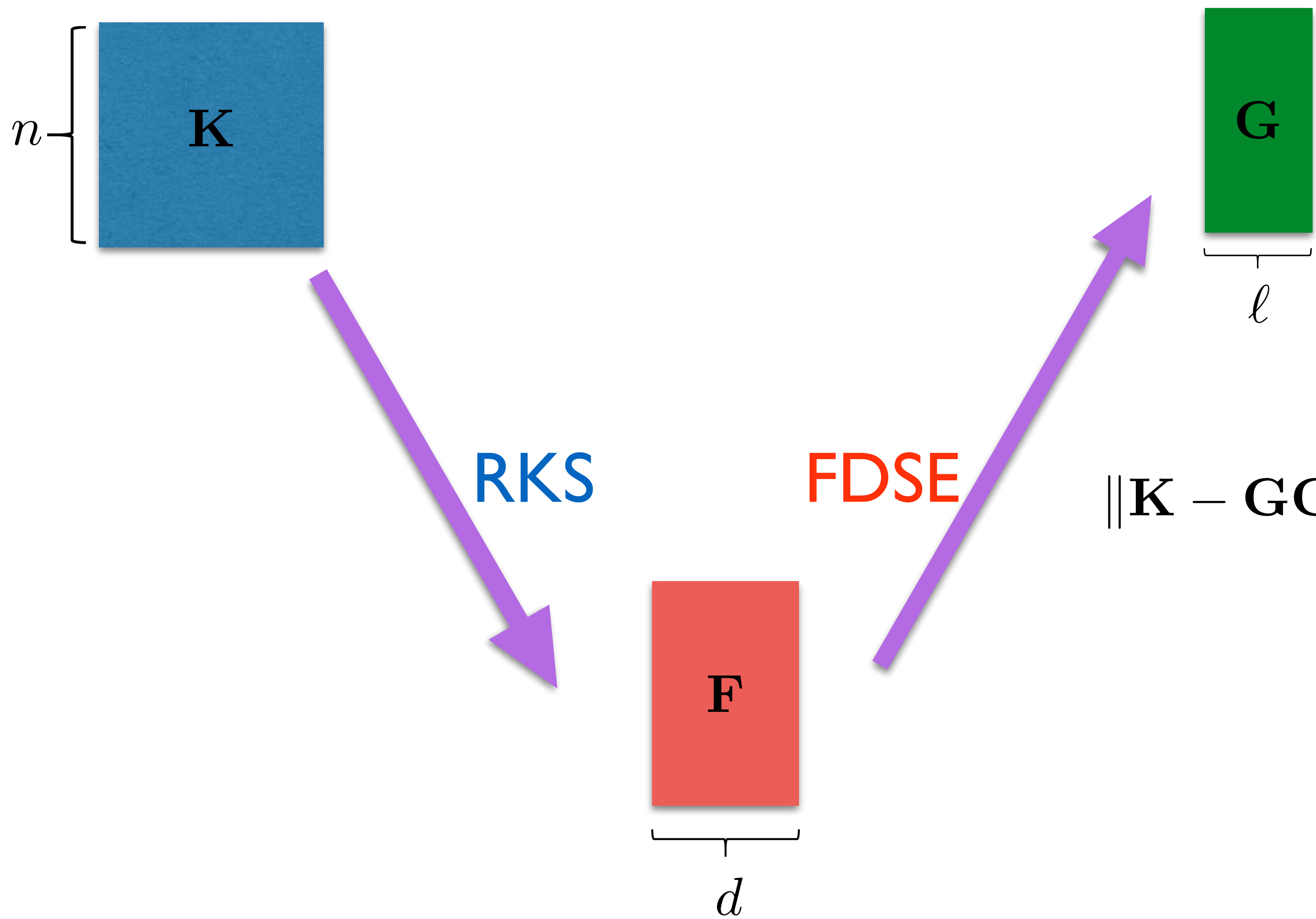
Our Method

- Use small ℓ to maintain information in RKS

$$
\begin{aligned}
k_{ij} &= k(\mathbf{x}_i - \mathbf{x}_j) = \int p(\mathbf{z}) e^{i\mathbf{z}^T(\mathbf{x}_i - \mathbf{x}_j)} d\mathbf{z} && (1)\\
&\approx \frac{2}{\ell} \sum_{s=1}^{\ell/2} \langle e^{i\mathbf{z}_s^T \mathbf{x}_i}, e^{i\mathbf{z}_s^T \mathbf{x}_j} \rangle \\
&= \sum_{s=1}^{\ell/2} \langle \frac{1}{\sqrt{\ell/2}} \cos(\mathbf{z}_s^T \mathbf{x}_i), \frac{1}{\sqrt{\ell/2}} \cos(\mathbf{z}_s^T \mathbf{x}_j) \rangle \\
&\quad + \langle \frac{1}{\sqrt{\ell/2}} \sin(\mathbf{z}_s^T \mathbf{x}_i), \frac{1}{\sqrt{\ell/2}} \sin(\mathbf{z}_s^T \mathbf{x}_j) \rangle \\
&= \langle \mathbf{Z}(\mathbf{x}_i) \in \mathbb{R}^{\ell}, \mathbf{Z}(\mathbf{x}_j) \in \mathbb{R}^{\ell} \rangle && (2)
\end{aligned}
$$

Our Method

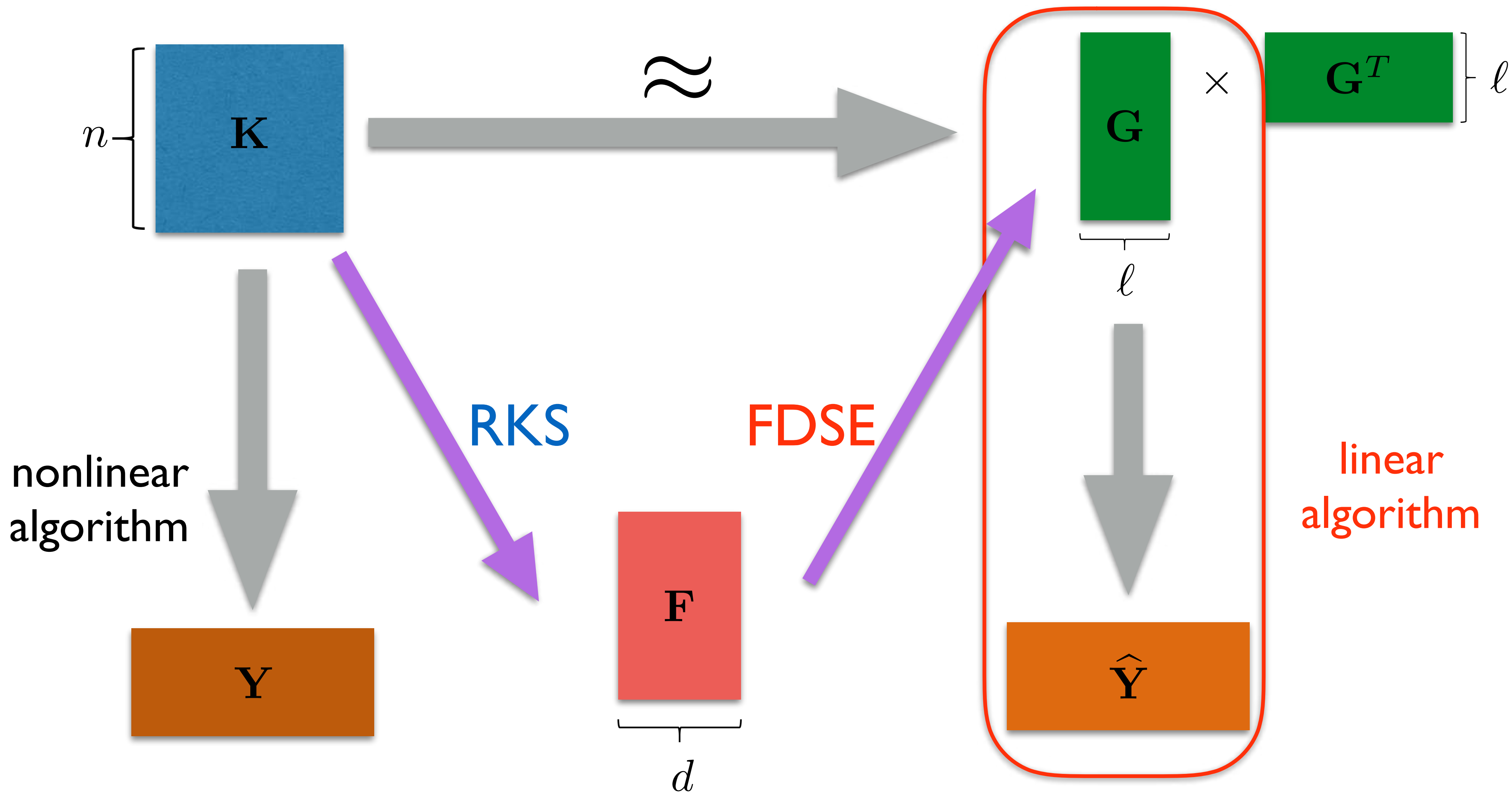
- Improve RKS via FDSE (fast data-dependent subspace embedding)



$$\|\mathbf{K} - \mathbf{G}\mathbf{G}^T\|_2 \leq O(n/\ell)$$

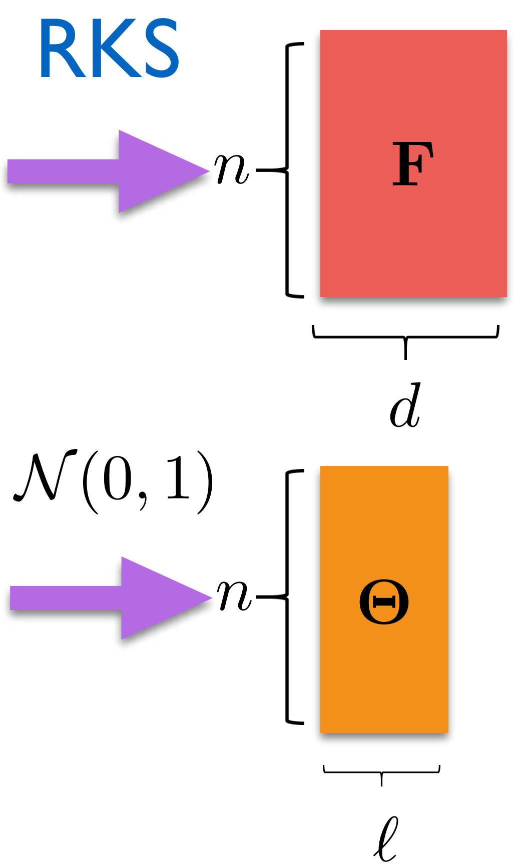
Our Method

- Improve RKS via FDSE (fast data-dependent subspace embedding)



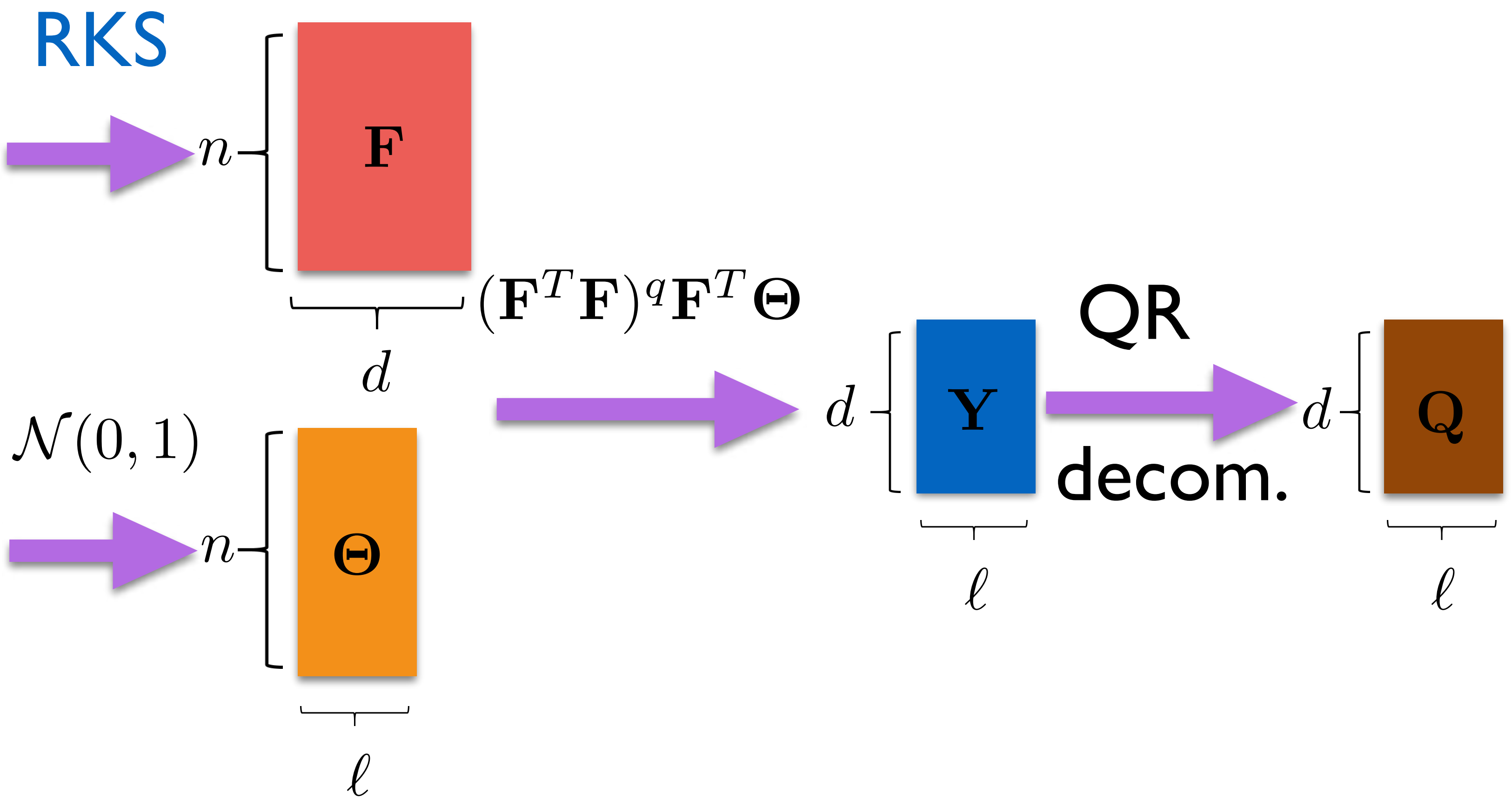
Our Method

- TEFM-G



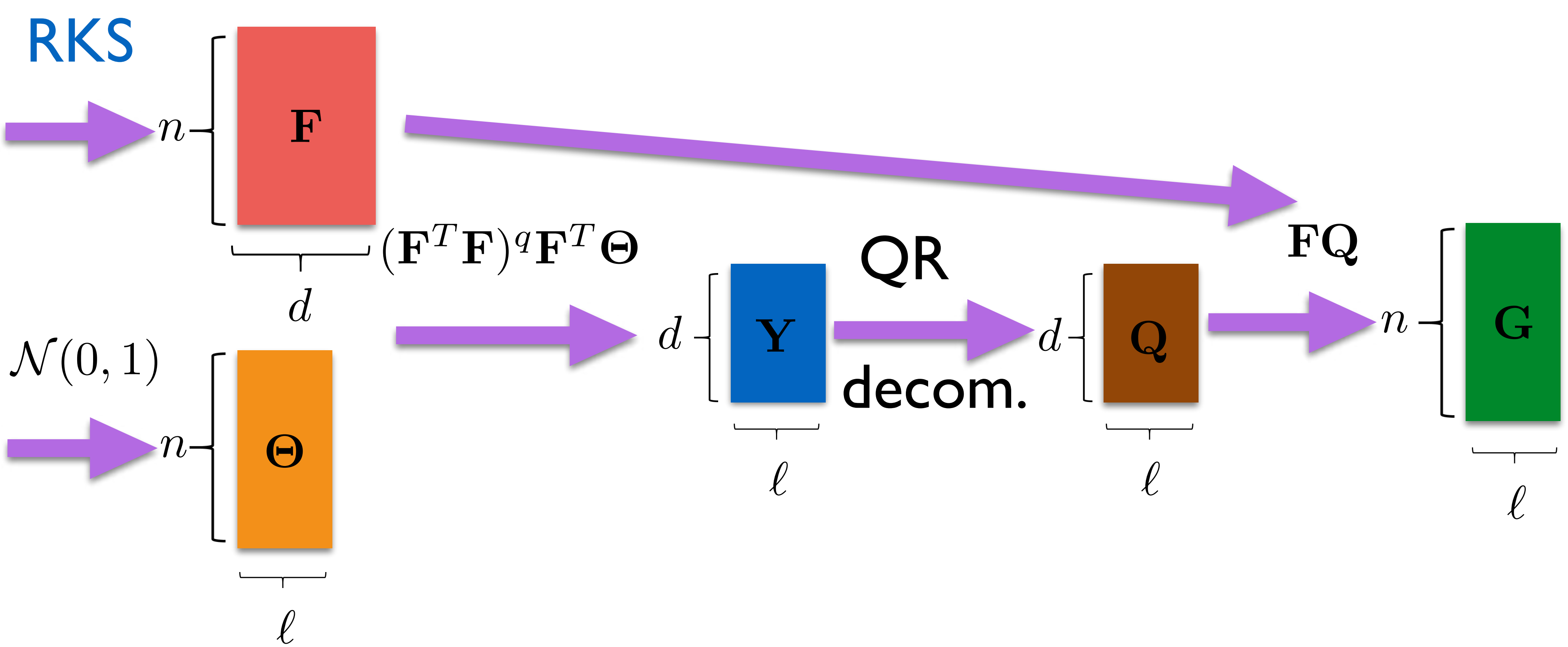
Our Method

- TEFM-G



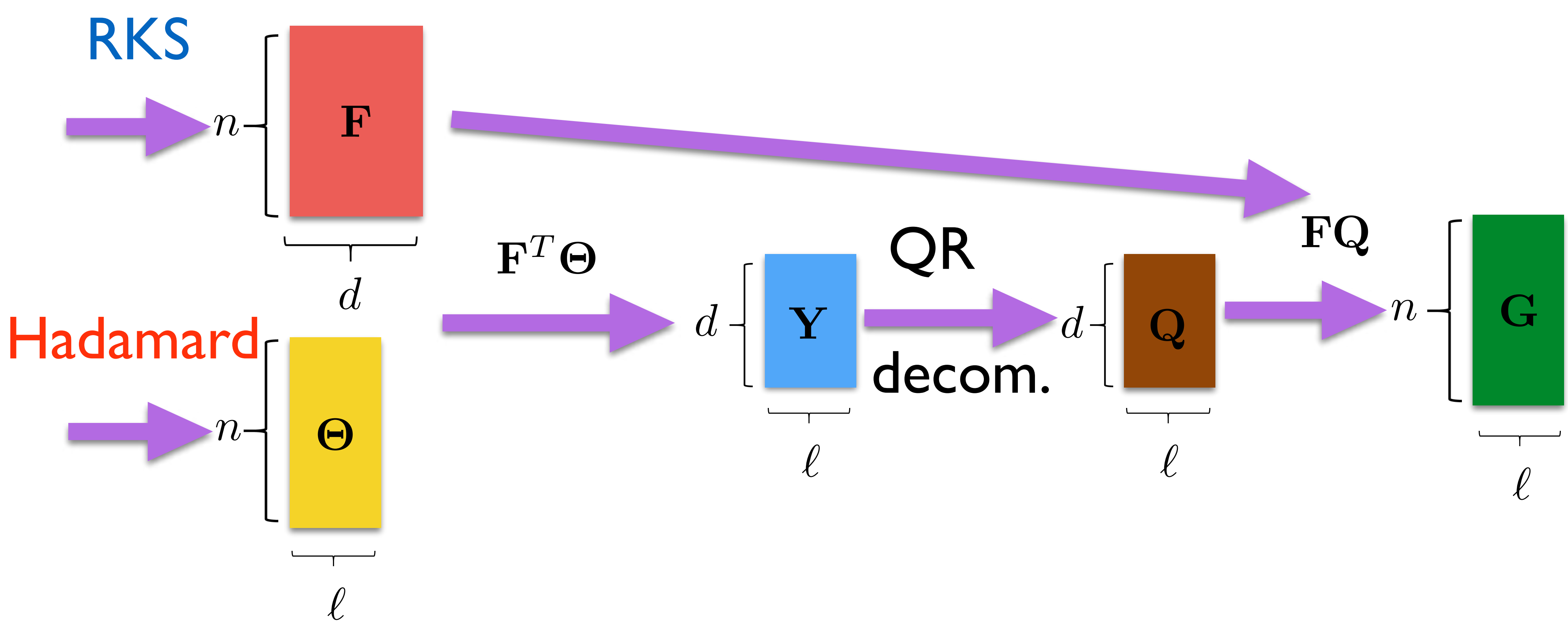
Our Method

- TEFM-G



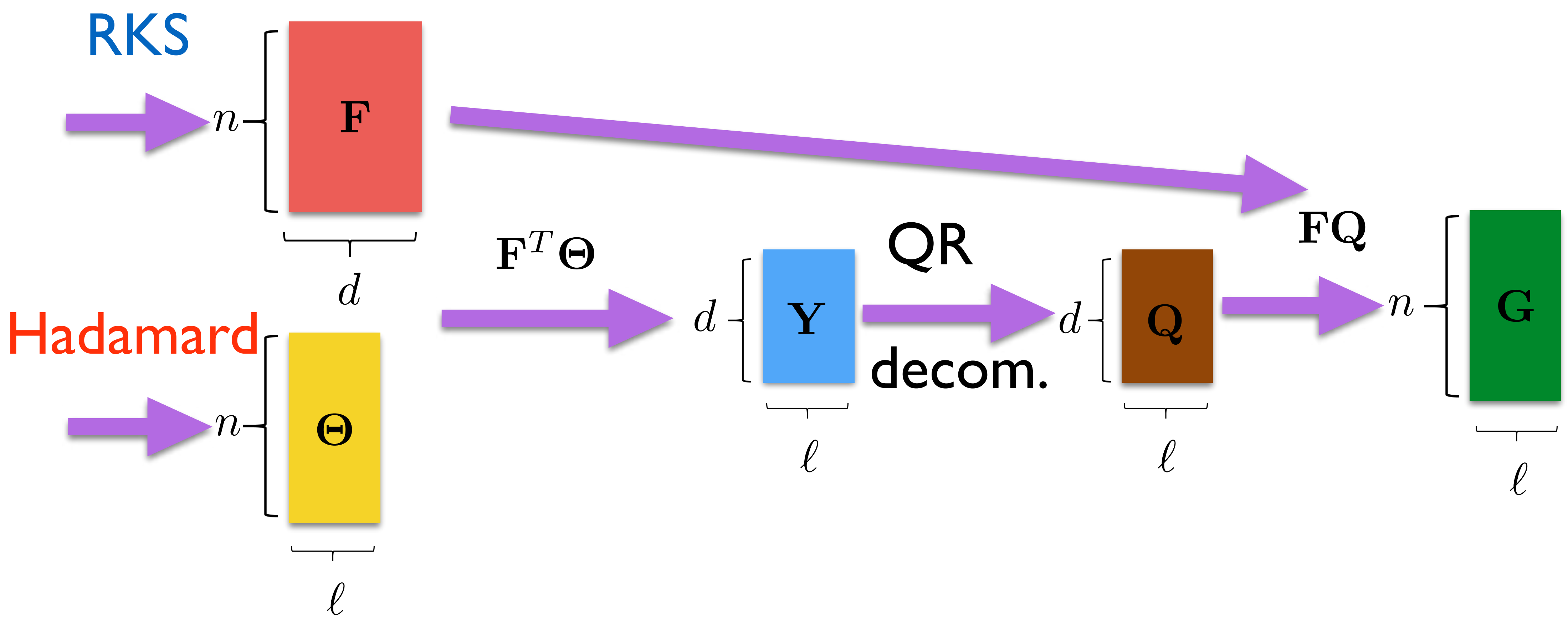
Our Method

- TEFM-S



Our Method

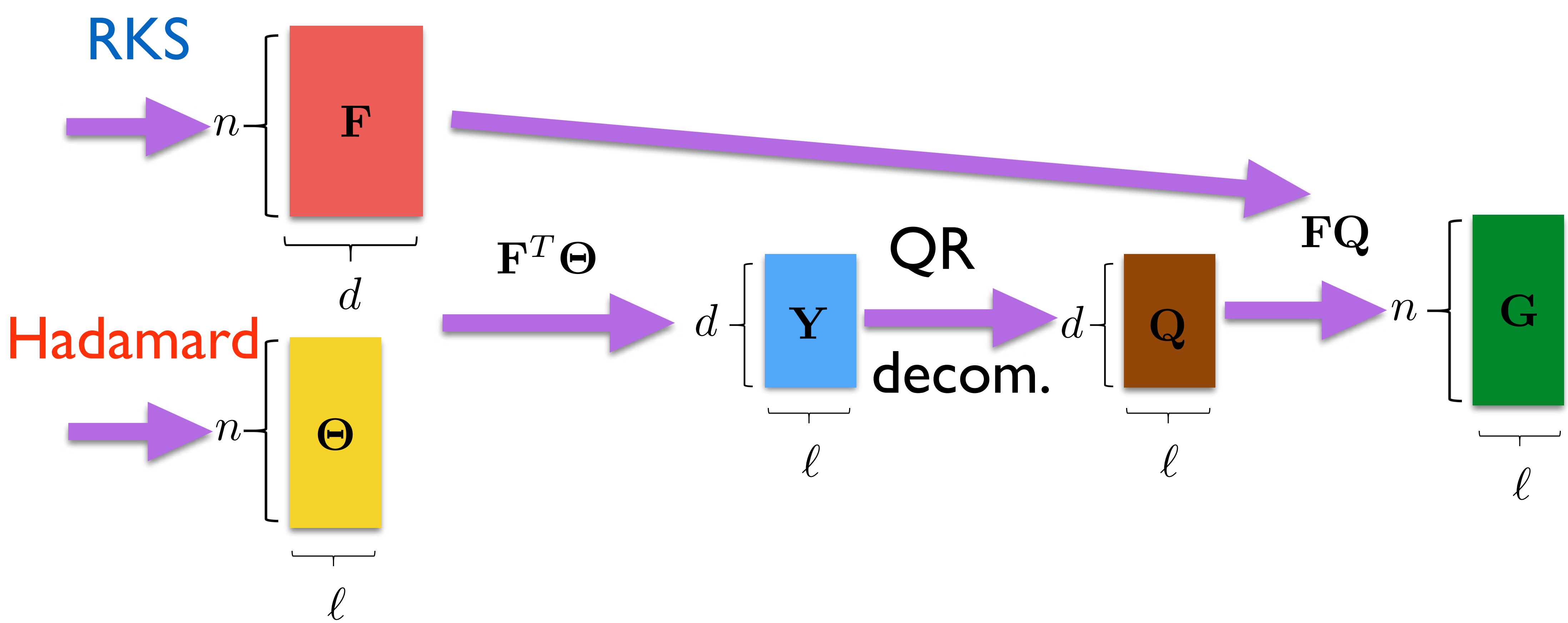
- TEFM-S



$$\mathbf{K} \approx \mathbf{F}\mathbf{F}^T \approx \mathbf{G}\mathbf{G}^T = \mathbf{F}\mathbf{Q}\mathbf{Q}^T\mathbf{F}^T$$

Our Method

- TEFM-S

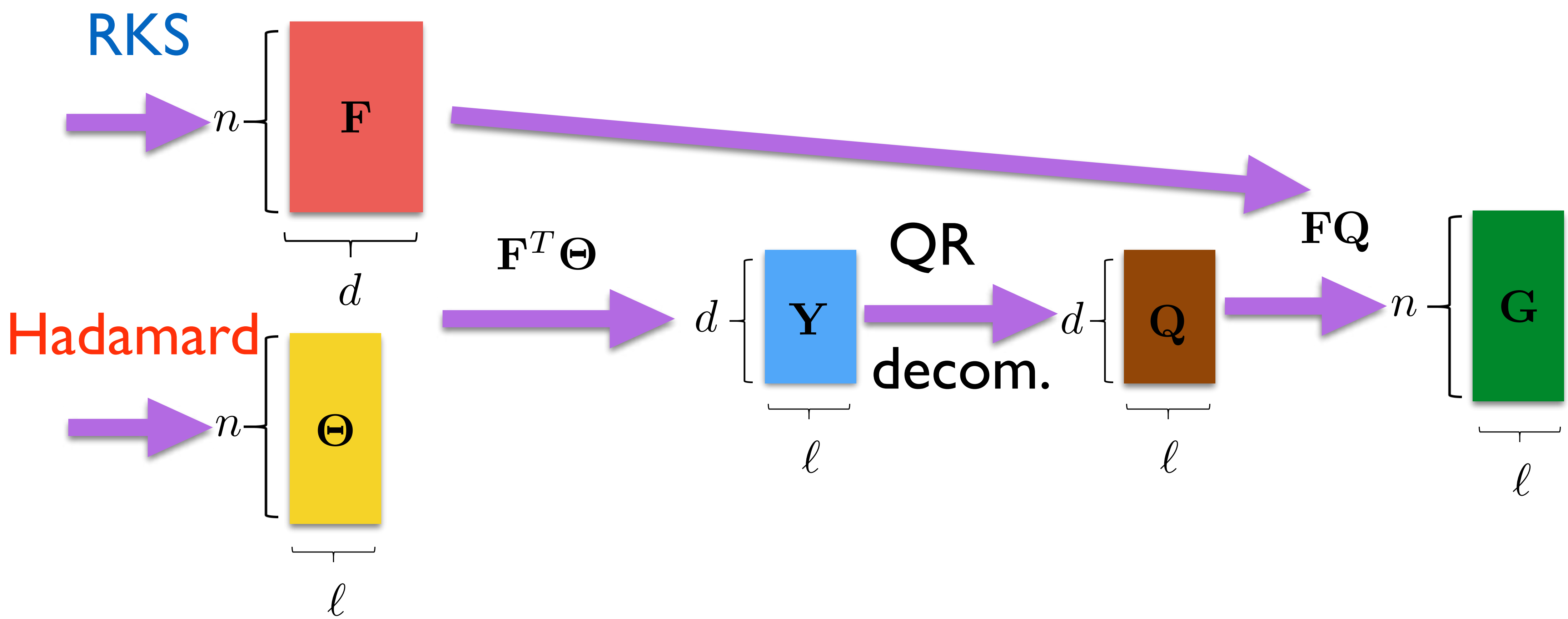


$$\mathbf{K} \approx \mathbf{F}\mathbf{F}^T \approx \mathbf{G}\mathbf{G}^T = \mathbf{F}\mathbf{Q}\mathbf{Q}^T\mathbf{F}^T \longrightarrow \mathbf{Q}\mathbf{Q}^T \approx \mathbf{I}_d,\ \mathbf{Q} \in \mathbf{F}^T$$

[N. Halko, et al., 2011]

Our Method

- TEFM-S



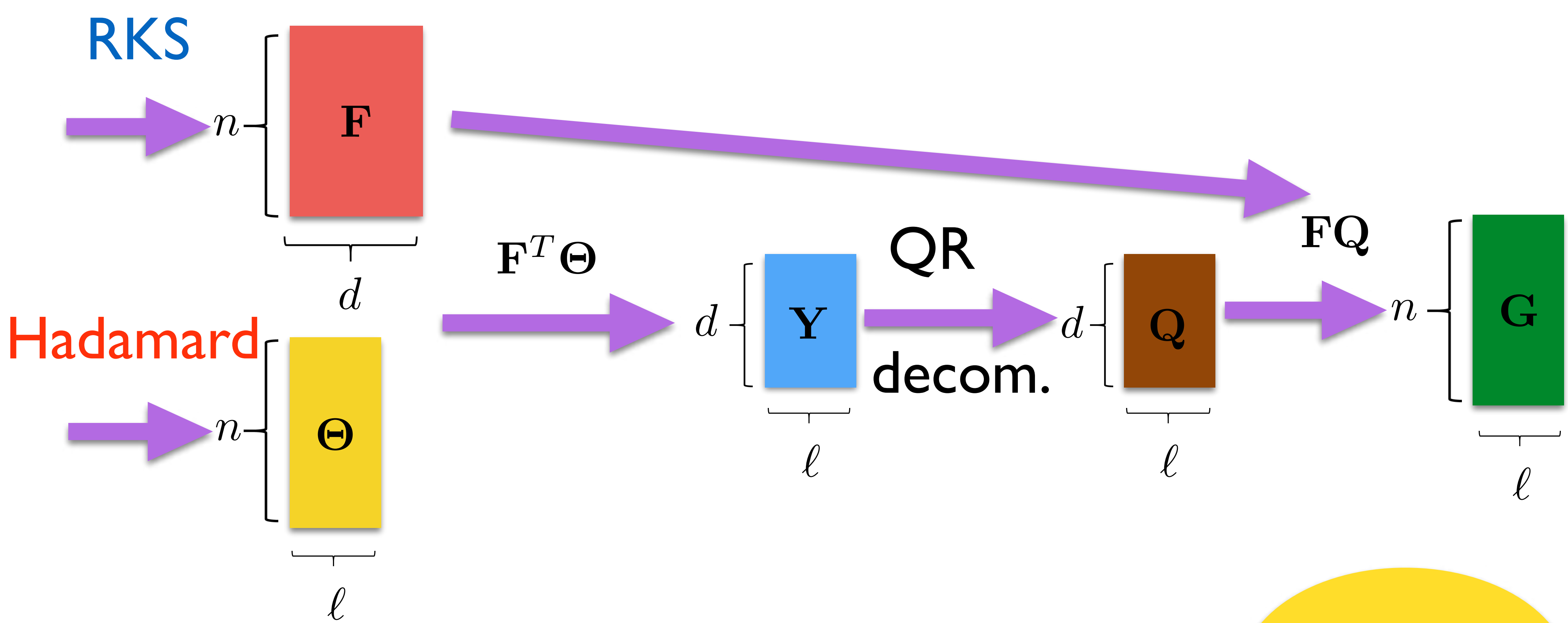
$$\mathbf{K} \approx \mathbf{F}\mathbf{F}^T \approx \mathbf{G}\mathbf{G}^T = \mathbf{F}\mathbf{Q}\mathbf{Q}^T\mathbf{F}^T \longrightarrow \mathbf{Q}\mathbf{Q}^T \approx \mathbf{I}_d,\ \mathbf{Q} \in \mathbf{F}^T$$

$$\longrightarrow \mathbf{Q} \in \mathbf{F}^T\mathbf{\Theta} \longrightarrow \mathbf{F}^T\mathbf{\Theta}\mathbf{\Theta}^T\mathbf{F} \approx \mathbf{F}^T\mathbf{F}$$

[N. Halko, et al., 2011]

Our Method

- TEFM-S



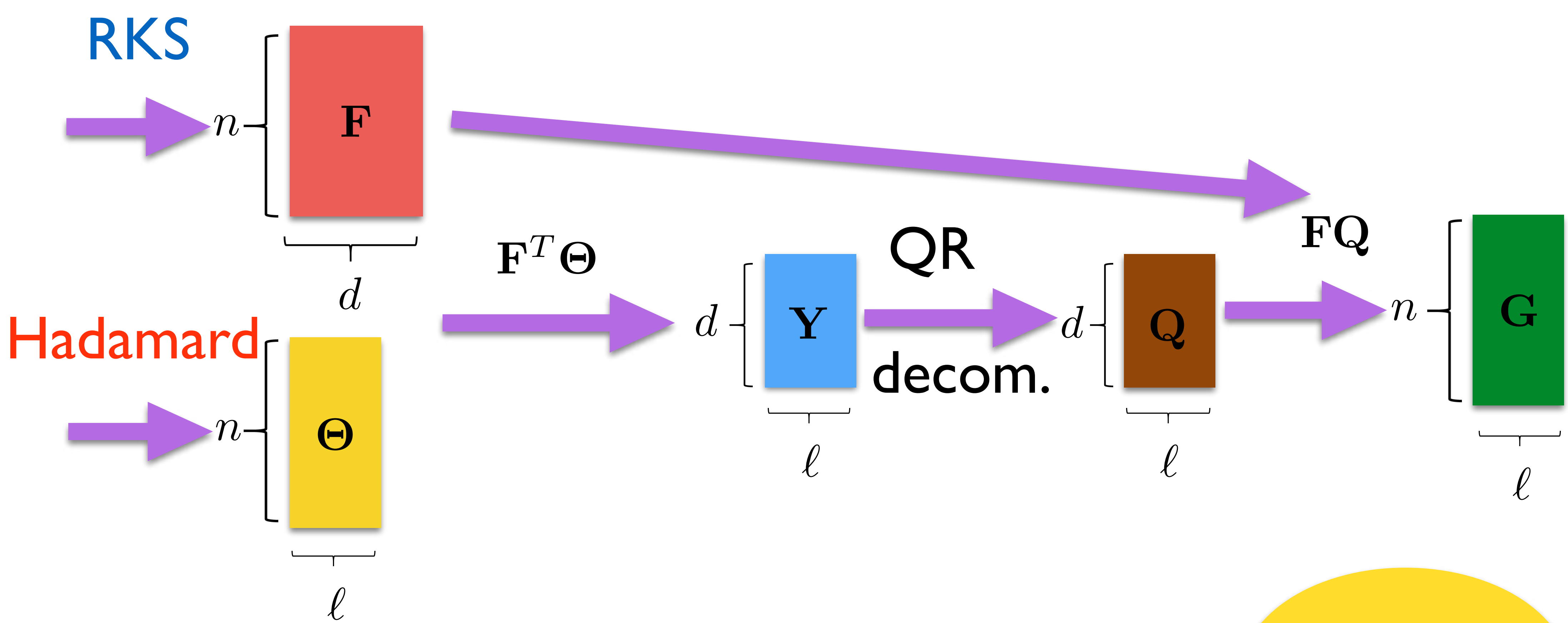
$$\mathbf{K} \approx \mathbf{F}\mathbf{F}^T \approx \mathbf{G}\mathbf{G}^T = \mathbf{F}\mathbf{Q}\mathbf{Q}^T\mathbf{F}^T \longrightarrow \mathbf{Q}\mathbf{Q}^T \approx \mathbf{I}_d,\ \mathbf{Q} \in \mathbf{F}^T$$

$$\longrightarrow \mathbf{Q} \in \mathbf{F}^T\boldsymbol{\Theta} \longrightarrow \mathbf{F}^T\boldsymbol{\Theta}\boldsymbol{\Theta}^T\mathbf{F} \approx \underline{\mathbf{F}^T\mathbf{F}}$$

$\mathbf{X}^T\mathbf{X}$

Our Method

- TEFM-S



$\mathbf{K} \approx \mathbf{F}\mathbf{F}^T \approx \mathbf{G}\mathbf{G}^T = \mathbf{F}\mathbf{Q}\mathbf{Q}^T\mathbf{F}^T$ ← $\mathbf{Q}\mathbf{Q}^T \approx \mathbf{I}_d$, $\mathbf{Q} \in \mathbf{F}^T$

← $\mathbf{Q} \in \mathbf{F}^T\boldsymbol{\Theta}$ ← $\mathbf{F}^T\boldsymbol{\Theta}\boldsymbol{\Theta}^T\mathbf{F} \approx \underline{\mathbf{F}^T\mathbf{F}}$

$\mathbf{X}^T\mathbf{X}$

error propagates

Results

- Theorem 3.1 & 3.2 (Kernel matrix approximation). Suppose we have a kernel matrix $\mathbf{K} \in \mathbb{R}^{n \times n}$ based on shift-invariant functions and get features $\mathbf{G} \in \mathbb{R}^{n \times \ell}$ via Algorithm TEFM-G or TEFM-S. Then the following inequality holds with limited failure probability

$$\|\mathbf{K} - \mathbf{G}\mathbf{G}^T\|_2 \leq O(n/\ell).$$

Results

- Theorem 3.1 & 3.2 (Kernel matrix approximation). Suppose we have a kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ based on shift-invariant functions and get features $\mathbf{G} \in \mathbb{R}^{n\times \ell}$ via Algorithm TEFM-G or TEFM-S. Then the following inequality holds with limited failure probability

$$\|\mathbf{K} - \mathbf{G}\mathbf{G}^T\|_2 \leq O(n/\ell).$$

- Theorem 3.3 (Impact on learning tasks). Suppose we get a kernel matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$ by operating shift-invariant functions on the data $\mathbf{X}^T = \{\mathbf{x}_i \in \mathbb{R}^m\}_{i=1}^n$ and a feature matrix $\mathbf{G}^T = \{\mathbf{g}_i \in \mathbb{R}^\ell\}_{i=1}^n$ by Algorithm TEFM-G or TEFM-S. Then the following inequality holds with limited failure probability

$$F(\mathbf{w}_{\mathbf{G}}^*) \leq F(\mathbf{w}_{\mathbf{K}}^*) + O(1/\ell),$$

where $F(\mathbf{w}_{\mathbf{Z}(\mathbf{x})}^*) = \min_{\mathbf{w}} \frac{\lambda}{2}\|\mathbf{w}\|_2^2 + \frac{1}{n}\sum_{i=1}^n \hbar\{\mathbf{w}^T\mathbf{Z}(\mathbf{x}_i), y_i\}$, and training on $\{\mathbf{Z}(\mathbf{x}_i) = \mathbf{g}_i\}_{i=1}^n$ gets $F(\mathbf{w}_{\mathbf{G}}^*)$ and training on $\mathbf{K}$ ($\{\mathbf{Z}(\mathbf{x}_i) = \Phi(\mathbf{x}_i)\}_{i=1}^n$) gets $F(\mathbf{w}_{\mathbf{K}}^*)$.

Results

- Kernel matrix approximation
  - Our method: $\|\mathbf{K}-\mathbf{G}\mathbf{G}^T\|_2 \leq O(n/\ell)$ (Theorem 3.1 & 3.2)
  - RKS: $\|\mathbf{K}-\mathbf{G}\mathbf{G}^T\|_2 \leq O(n/\sqrt{\ell})$
- Impact on learning tasks
  - Training on our features: $O(F(\mathbf{w}^*_{\mathbf{K}})+1/\ell)$ (Theorem 3.3)
  - Training on RKS: $O(F(\mathbf{w}^*_{\mathbf{K}})+1/\sqrt{\ell})$

ℓ: $\mathbf{G} \in \mathbb{R}^{n\times\ell}$

Results

- Time cost for ridge regression

	Mapping	Training	Prediction
Kernel	$O(\mathrm{nnz}(\mathbf{X})n)$	$O(n^3)$	$O(tmn)$
RKS	$O(\mathrm{nnz}(\mathbf{X})\ell^2)$	$O(n\ell^4)$	$O(tm\ell^2)$
TEFM-G	$O(\mathrm{nnz}(\mathbf{X})\ell^2 + \ell^4 + n\ell^3)$	$O(n\ell^2)$	$O(tm\ell^2 + \ell^3)$
TEFM-S	$O(\mathrm{nnz}(\mathbf{X})\ell^2 + \ell^4 + n\ell^2 \log \ell)$	$O(n\ell^2)$	$O(tm\ell^2 + \ell^3)$

- $\mathbf{X} \in \mathbb{R}^{n \times m}$: input data
- t : the number of test points
- $\ell \ll n$: the number of mapped features

Results

- Time cost for ridge regression

	Mapping	Training	Prediction
Kernel	$O(\text{nnz}(\mathbf{X})n)$	$O(n^3)$	$O(tmn)$
RKS	$O(\text{nnz}(\mathbf{X})\ell^2)$	$O(n\ell^4)$	$O(tm\ell^2)$
TEFM-G	$O(\text{nnz}(\mathbf{X})\ell^2 + \ell^4 + n\ell^3)$	$O(n\ell^2)$	$O(tm\ell^2 + \ell^3)$
TEFM-S	$O(\text{nnz}(\mathbf{X})\ell^2 + \ell^4 + n\ell^2 \log \ell)$	$O(n\ell^2)$	$O(tm\ell^2 + \ell^3)$

- $\mathbf{X} \in \mathbb{R}^{n \times m}$: input data
- t : the number of test points
- $\ell \ll n$: the number of mapped features

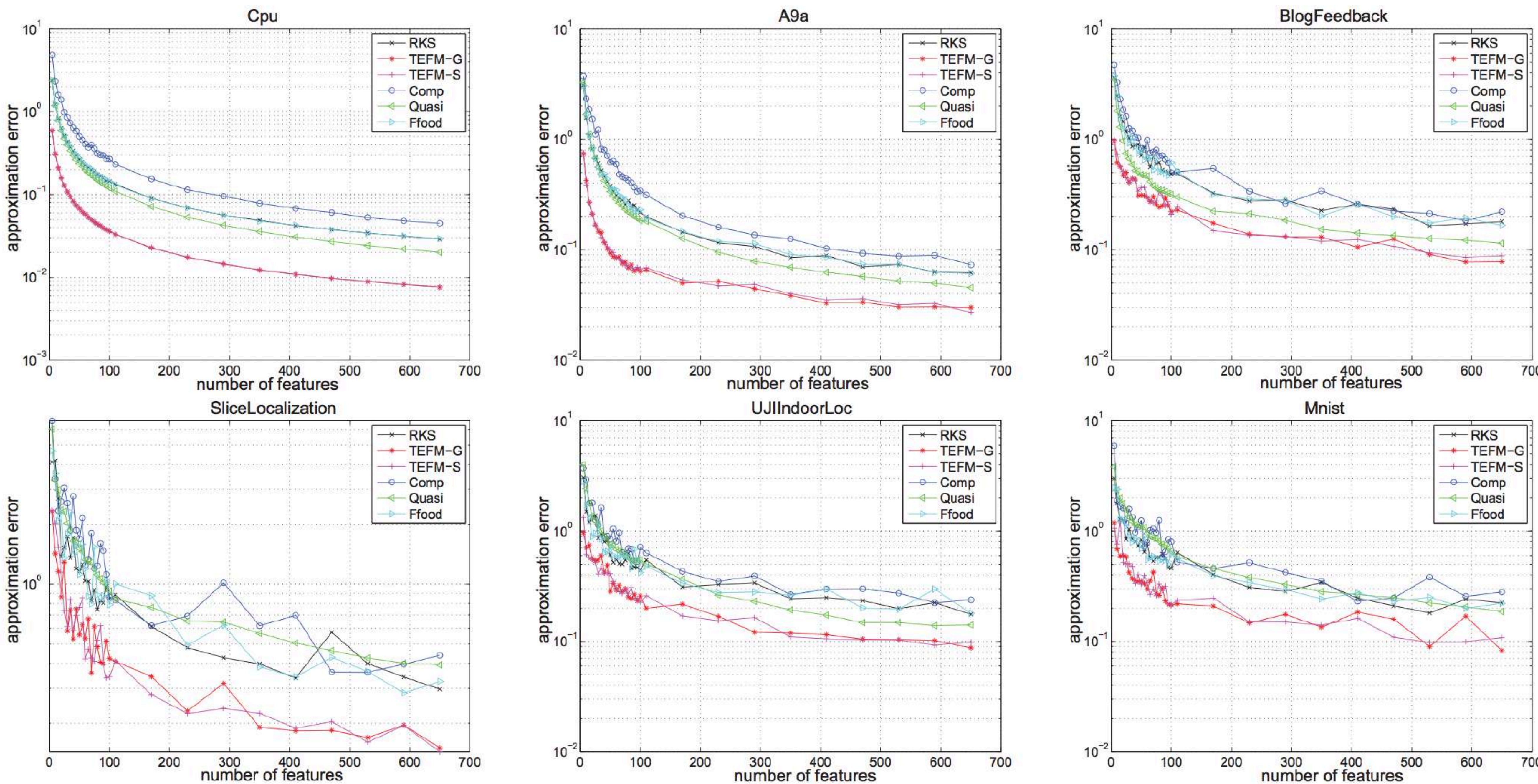
Experiments

- Compared methods
  - Random Kitchen Sinks (denoted by RKS) [A. Rahimi, et al., 2007]
  - Our proposed algorithms TEFM-G and TEFM-S
  - Compact feature maps (denoted by Comp) [R. Hamid, et al., 2014]
  - Quasi-Monte Carlo method (denoted by Quasi) [J. Yang, et al., 2014]
  - Fastfood method (denoted by Ffood) [Q. Le, et al., 2013]

Real Data

Dataset	Size	Dimension
Mnist	70,000	784
BlogFeedback	60,021	280
SliceLocalization	53,500	384
UJIIndoorLoc	21,048	520
Cpu	6,554	21
A9a	48,842	123

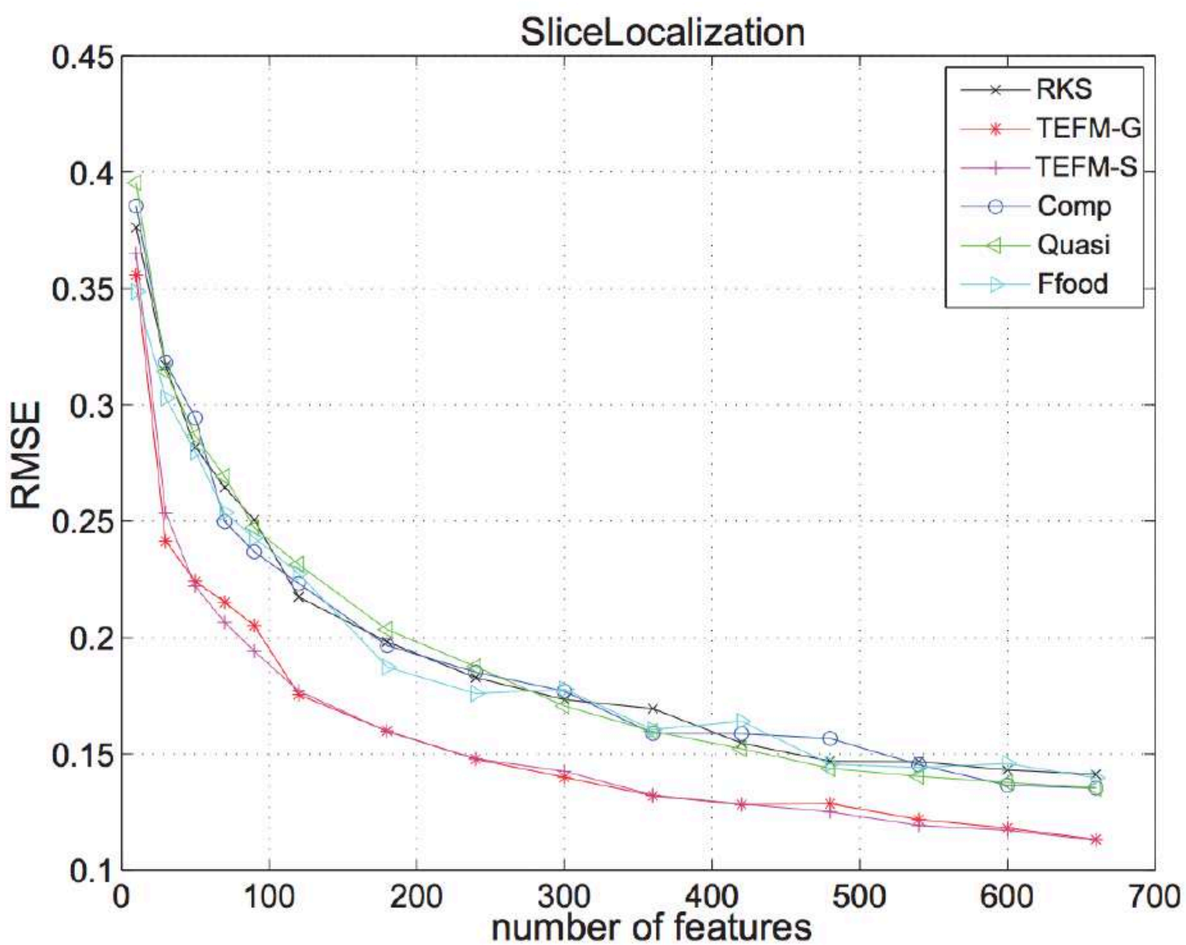
Kernel Matrix Approximation



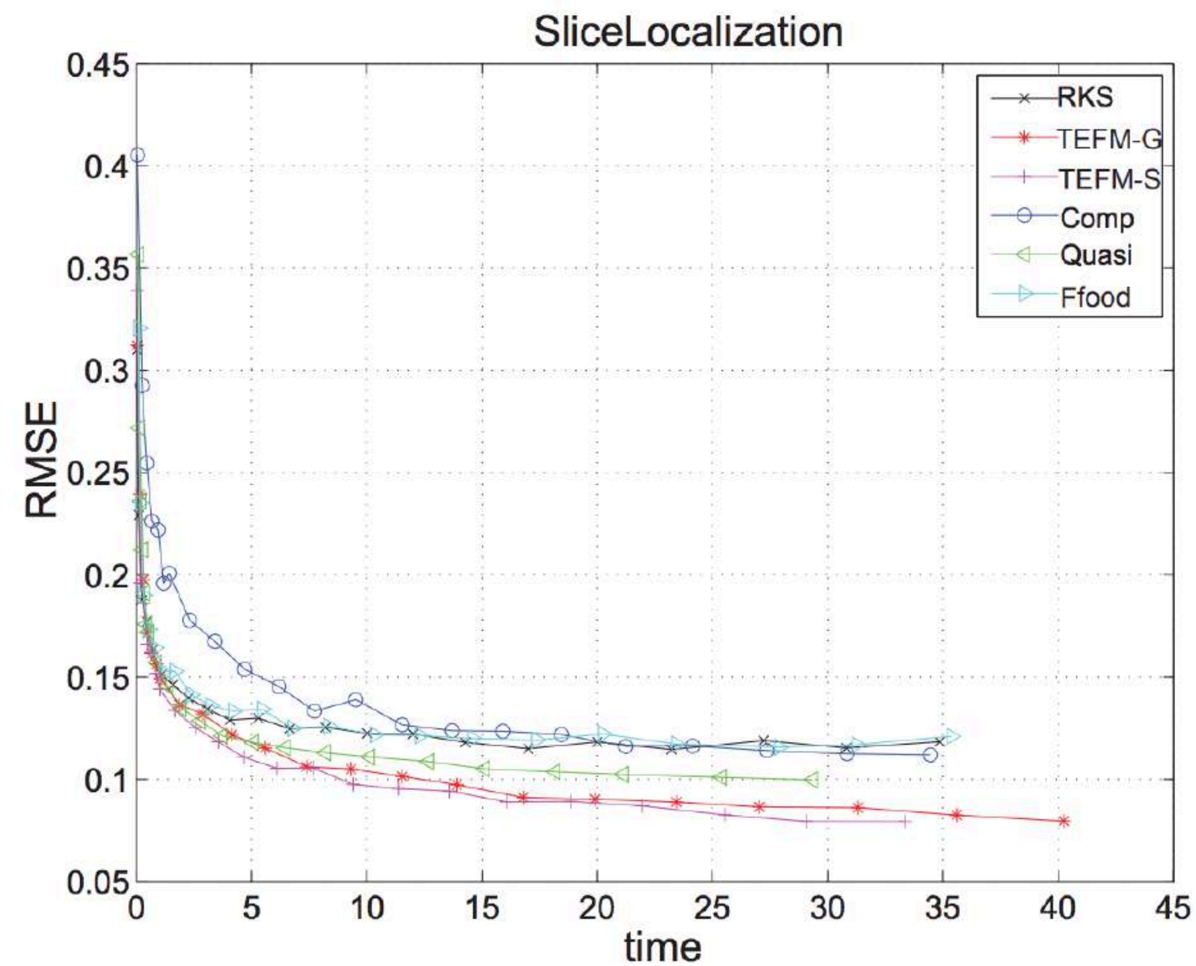
approximation error vs. feature number ℓ

Ridge Regression Task

- RMSE (root mean square error)



RMSE vs. feature number ℓ



RMSE vs. time (mapping+training) in sec.

Conclusion

- Adopt randomized algorithms to get a better kernel matrix approximation and efficient training on down-stream learning algorithms
- Demonstrate the good performance by provable results, complexity analysis, and experiments

Outline

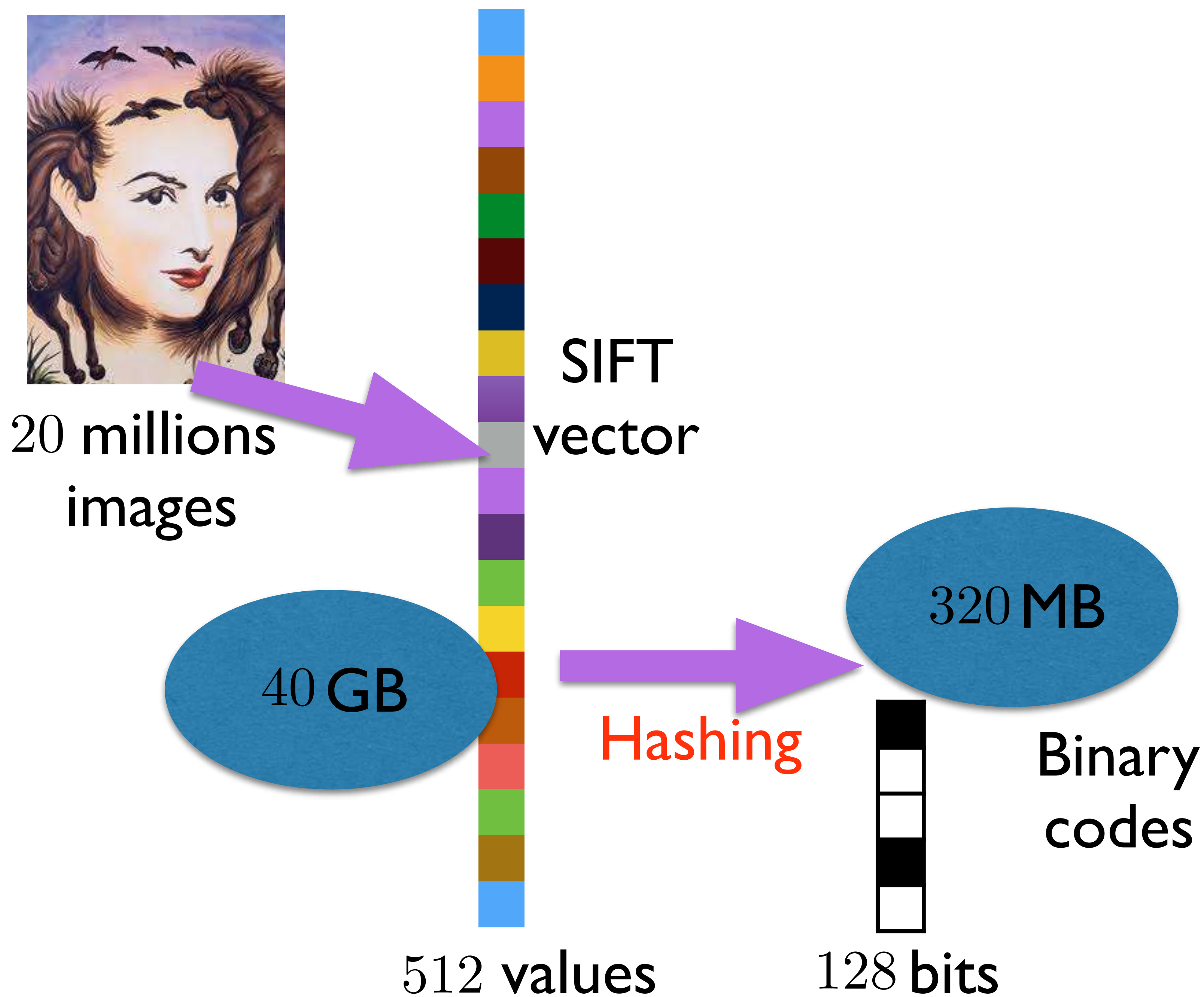
Randomized algorithms for machine learning (My thesis)

- Introduction & Background (Chapter 2)
- Kernel methods (Chapter 3)
- Unsupervised online hashing (Chapter 4)
- Covariance estimation (Chapter 5)
- Conclusion & Future work (Chapter 6)

contribution

Background

- Hashing



Background

- PCA-based hashing (Unsupervised batch-based)
  - PCA (Principal Component Analysis) step

$$\max_{\mathbf{W}\in\mathbb{R}^{d\times r}} \quad \mathrm{Tr}(\mathbf{W}^T(\mathbf{A}-\boldsymbol{\mu})^T(\mathbf{A}-\boldsymbol{\mu})\mathbf{W})$$
$$\text{s.t.} \quad \mathbf{W}^T\mathbf{W} = \mathbf{I}_r$$

  - Quantization step

$$h_k(\mathbf{a}^i) = \mathrm{sgn}((\mathbf{a}^i - \boldsymbol{\mu})\mathbf{w}_k), \ k \in [r]$$

PCA step for $\mathbf{A} \in \mathbb{R}^{n\times d}$: $O(nd^2)$ time $O(nd)$ space!

Background

- PCA-based hashing (Unsupervised batch-based)
    - PCA (Principal Component Analysis) step

$$\max_{\mathbf{W}\in\mathbb{R}^{d\times r}} \quad \mathrm{Tr}(\mathbf{W}^T(\mathbf{A}-\boldsymbol{\mu})^T(\mathbf{A}-\boldsymbol{\mu})\mathbf{W})$$
$$\text{s.t.} \quad \mathbf{W}^T\mathbf{W}=\mathbf{I}_r$$

$\mathbf{X}^T\mathbf{X}$

- Quantization step

$$h_k(\mathbf{a}^i)=\mathrm{sgn}((\mathbf{a}^i-\boldsymbol{\mu})\mathbf{w}_k),\ \ k\in[r]$$

PCA step for $\mathbf{A}\in\mathbb{R}^{n\times d}$: $O(nd^2)$ time $O(nd)$ space!

Background

- Unsupervised online hashing
  - Label-free
  - Adaptive
  - Space-efficient
  - Single-pass

Related Work

- Online Sketching Hashing (OSH) [C. Leng, et al., 2015]
  - Sketch $\mathbf{A} - \boldsymbol{\mu} \in \mathbb{R}^{n \times d}$ into $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ with $\mathbf{B}^T\mathbf{B} \approx (\mathbf{A} - \boldsymbol{\mu})^T(\mathbf{A} - \boldsymbol{\mu})$ in an online fashion which requires $(nd\ell)$ time and $O(d\ell)$ space

Related Work

- Online Sketching Hashing (OSH) [C. Leng, et al., 2015]
  - Sketch $\mathbf{A}-\boldsymbol{\mu}\in\mathbb{R}^{n\times d}$ into $\mathbf{B}\in\mathbb{R}^{\ell\times d}$ with $\mathbf{B}^T\mathbf{B}\approx(\mathbf{A}-\boldsymbol{\mu})^T(\mathbf{A}-\boldsymbol{\mu})$ in an online fashion which requires $(nd\ell)$time and $O(d\ell)$ space

$\mathbf{X}^T\mathbf{X}$

Related Work

- Online Sketching Hashing (OSH) [C. Leng, et al., 2015]
  - Sketch $\mathbf{A} - \boldsymbol{\mu} \in \mathbb{R}^{n \times d}$ into $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ with $\mathbf{B}^T\mathbf{B} \approx (\mathbf{A} - \boldsymbol{\mu})^T(\mathbf{A} - \boldsymbol{\mu})$ in an online fashion which requires $(nd\ell)$time and $O(d\ell)$ space
  - Compute the right eigenvectors of $\mathbf{B}$ instead of $\mathbf{A} - \boldsymbol{\mu}$ which requires $O(d\ell^2)$ time and $O(d\ell)$ space

Related Work

- Online Sketching Hashing (OSH) [C. Leng, et al., 2015]
  - Sketch $\mathbf{A} - \boldsymbol{\mu} \in \mathbb{R}^{n \times d}$ into $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ with $\mathbf{B}^T\mathbf{B} \approx (\mathbf{A} - \boldsymbol{\mu})^T(\mathbf{A} - \boldsymbol{\mu})$ in an online fashion which requires $(nd\ell)$ time and $O(d\ell)$ space
  - Compute the right eigenvectors of $\mathbf{B}$ instead of $\mathbf{A} - \boldsymbol{\mu}$ which requires $O(d\ell^2)$ time and $O(d\ell)$ space

$(nd\ell + d\ell^2)$ time and $O(d\ell)$ space costs in total!

($\ell \ll d \ll n$ is close to the size of the hashing coding)

Related Work

- Online Sketching Hashing (OSH) [C. Leng, et al., 2015]
  - $(nd\ell + d\ell^2)$ time cost is still large because $1 \ll d \ll n$

Our Method

- Propose a FasteR Online Sketching Hashing (FROSH): a randomized algorithm for OSH
  - Speed up the data sketching of OSH

$$\mathbf{B}^T\mathbf{B} \approx \underline{(\mathbf{A}-\boldsymbol{\mu})^T(\mathbf{A}-\boldsymbol{\mu})}$$

$$\mathbf{X}^T\mathbf{X}$$

Our Method

- Propose a FasteR Online Sketching Hashing (FROSH): a randomized algorithm for OSH
    - Speed up the data sketching of OSH

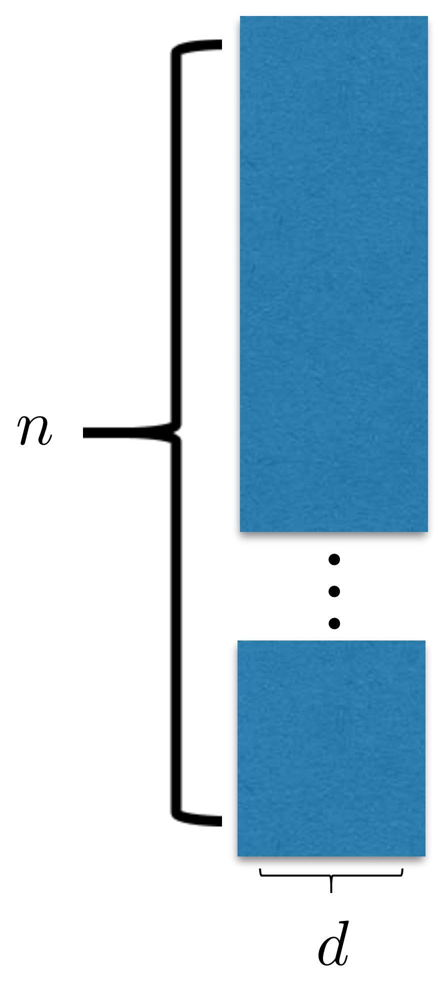
$$\mathbf{B}^T\mathbf{B} \approx (\mathbf{A} - \boldsymbol{\mu})^T(\mathbf{A} - \boldsymbol{\mu})$$

$$\mathbf{X}^T\mathbf{X}$$

also in an online fashion with a small fixed space cost

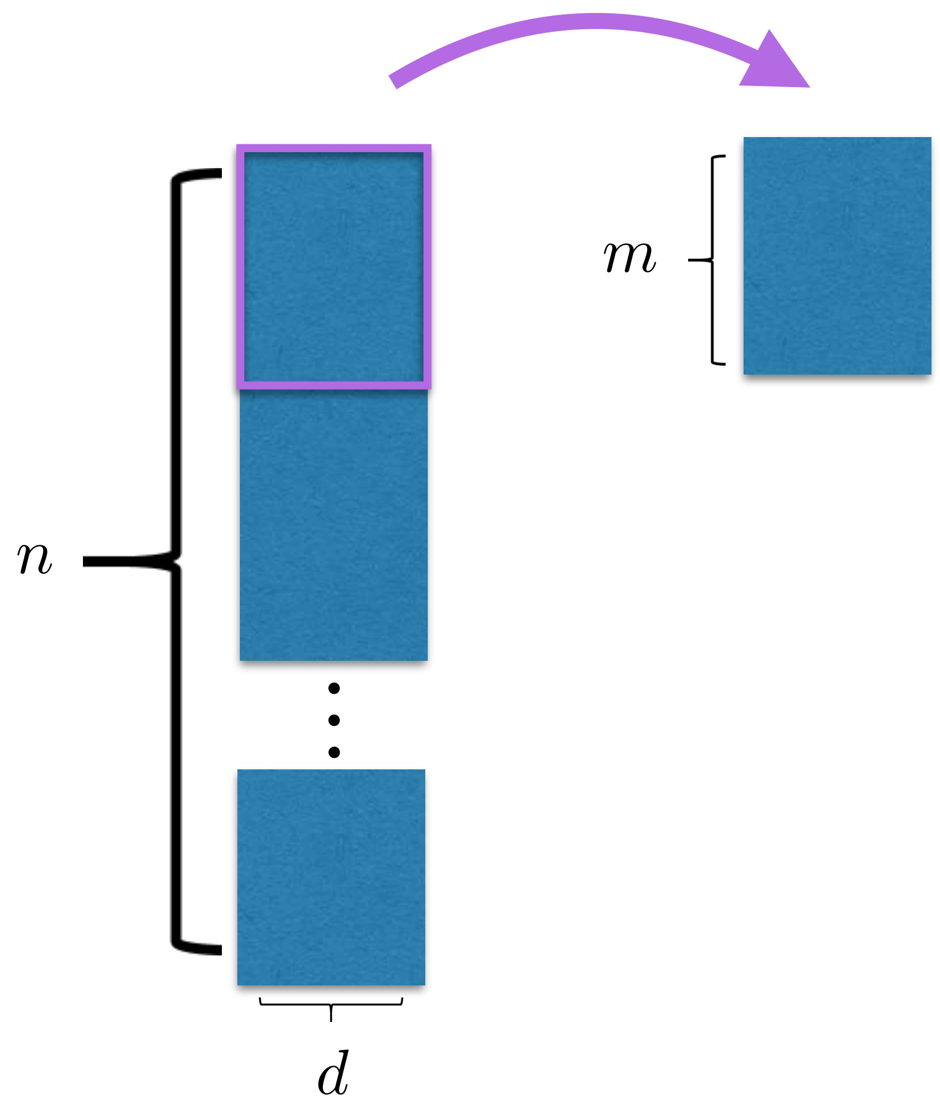
Our Method

- Online instance compression



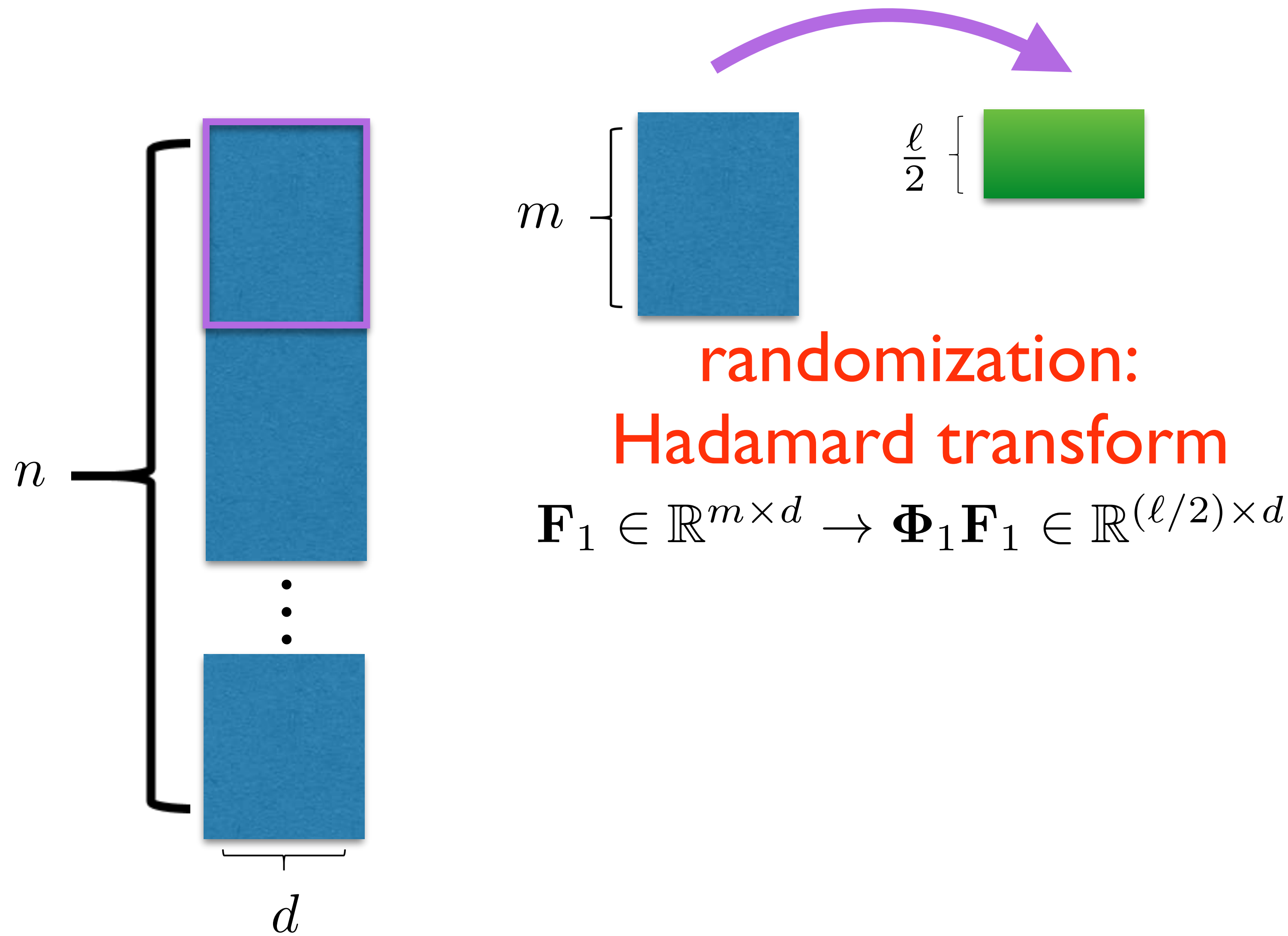
Our Method

- Online instance compression



Our Method

- Online instance compression

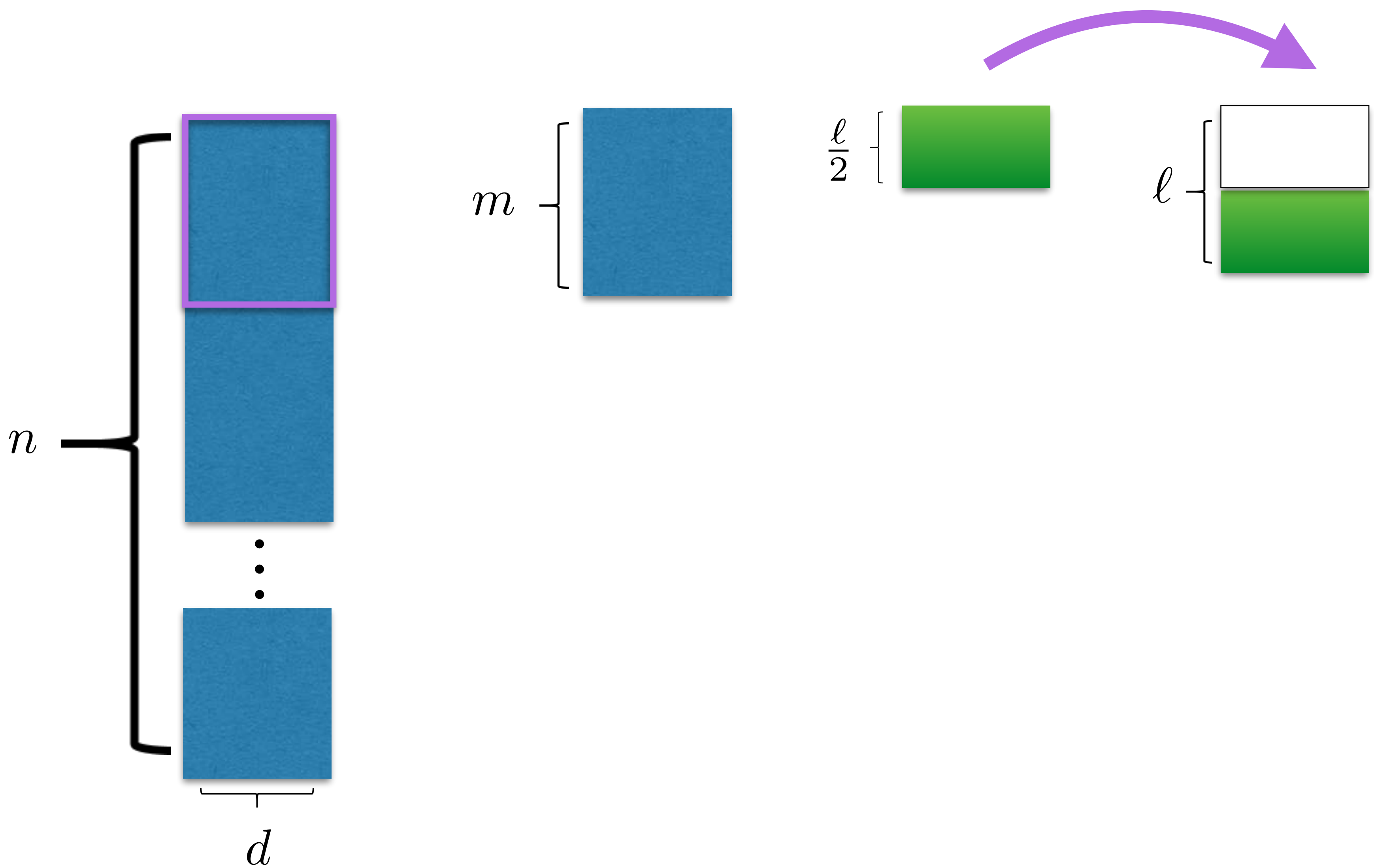


randomization:
Hadamard transform

$$\mathbf{F}_1 \in \mathbb{R}^{m \times d} \to \mathbf{\Phi}_1 \mathbf{F}_1 \in \mathbb{R}^{(\ell/2) \times d}$$

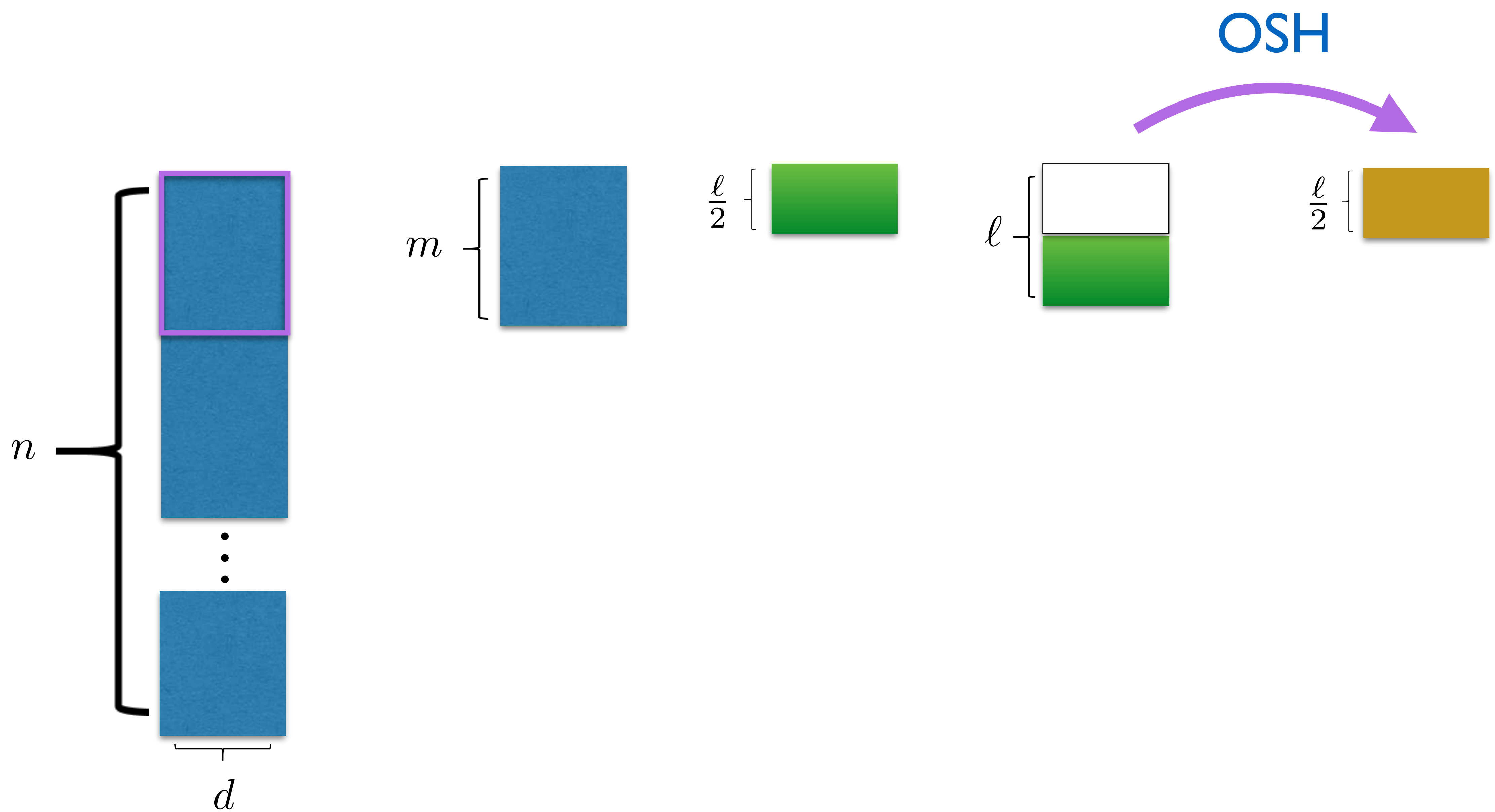
Our Method

- Online instance compression



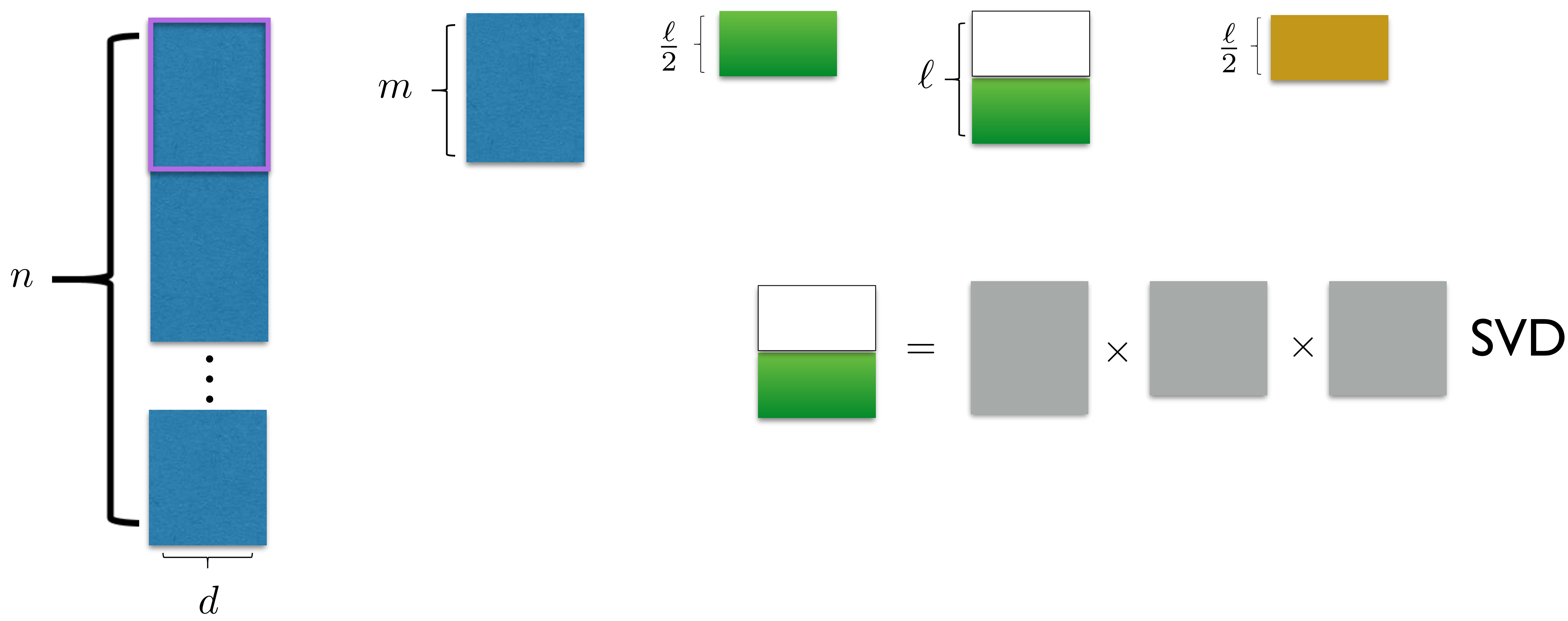
Our Method

- Online instance compression



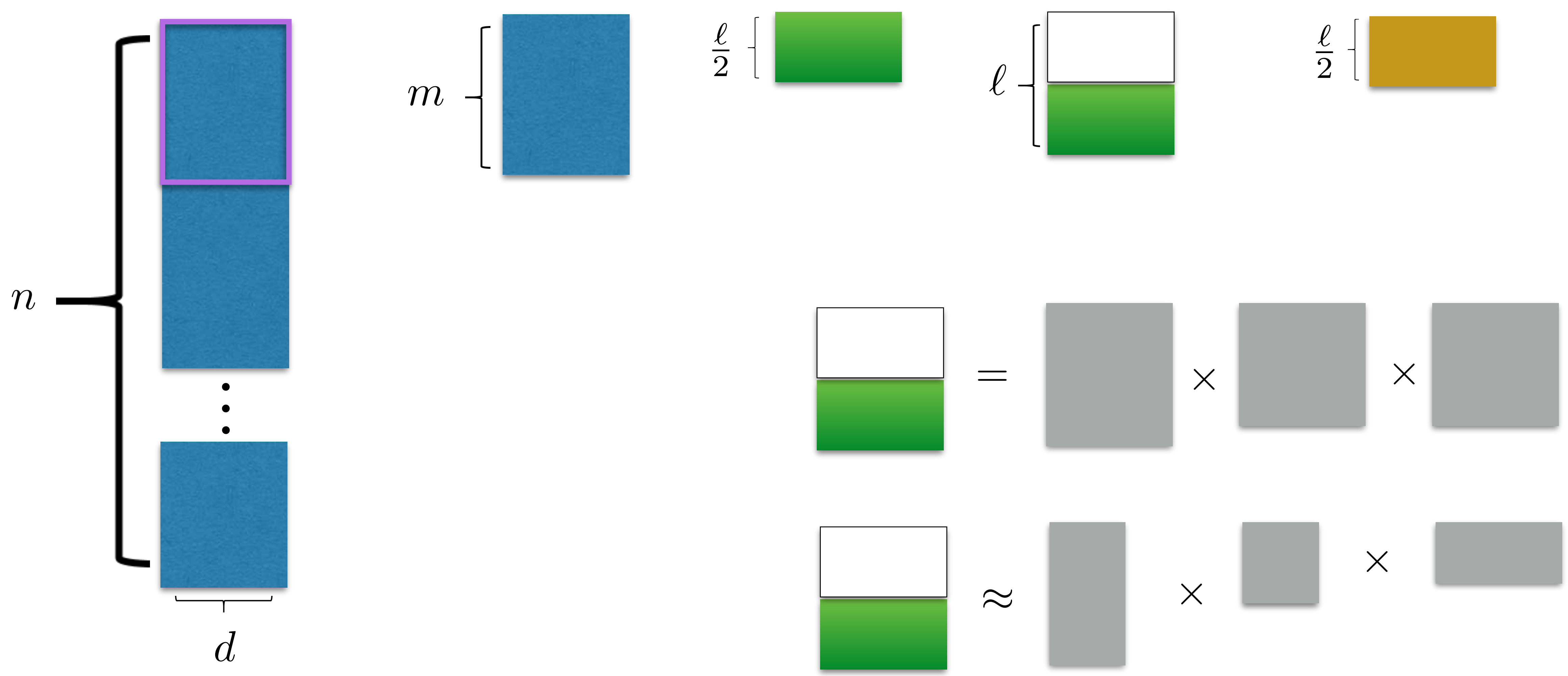
Our Method

- Online instance compression



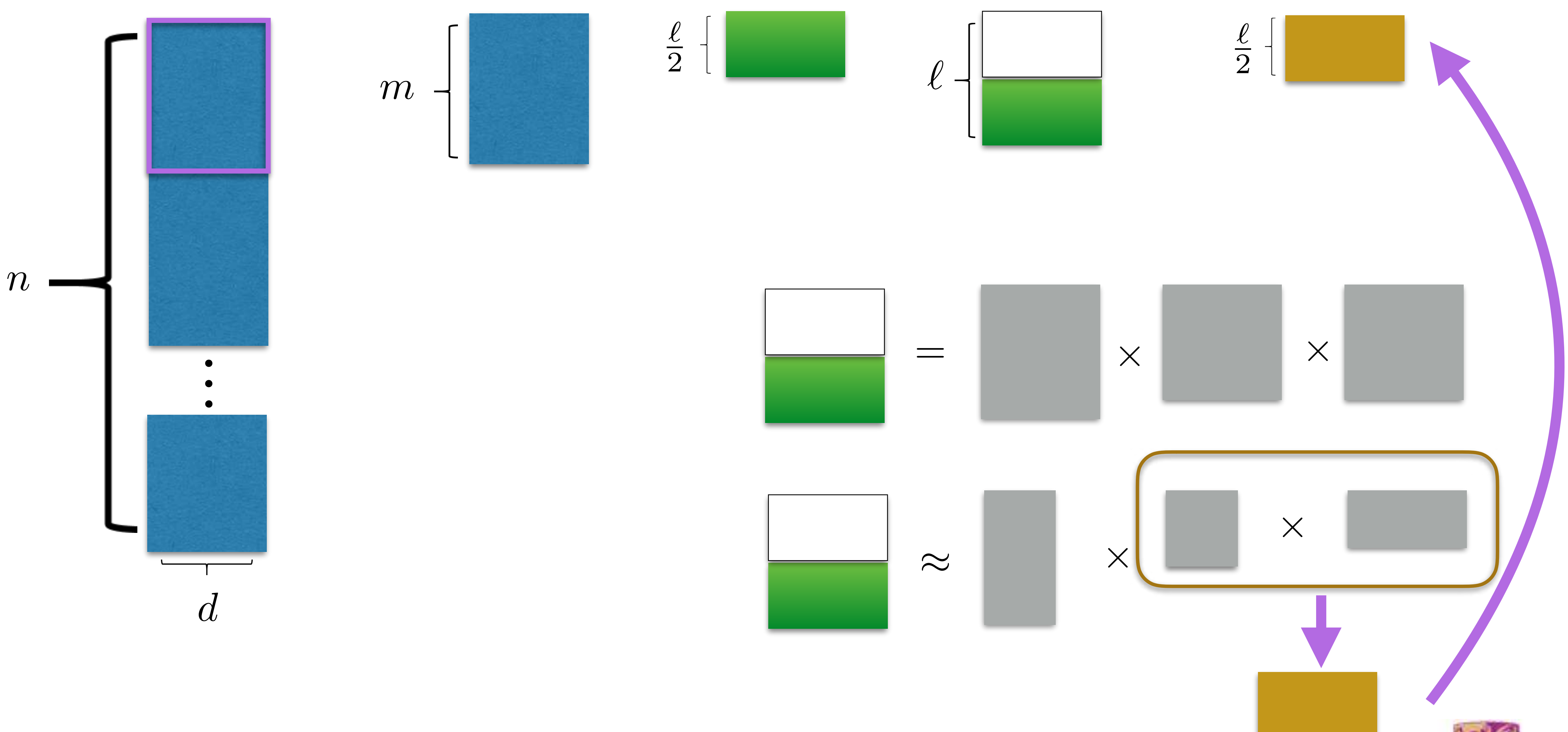
Our Method

- Online instance compression



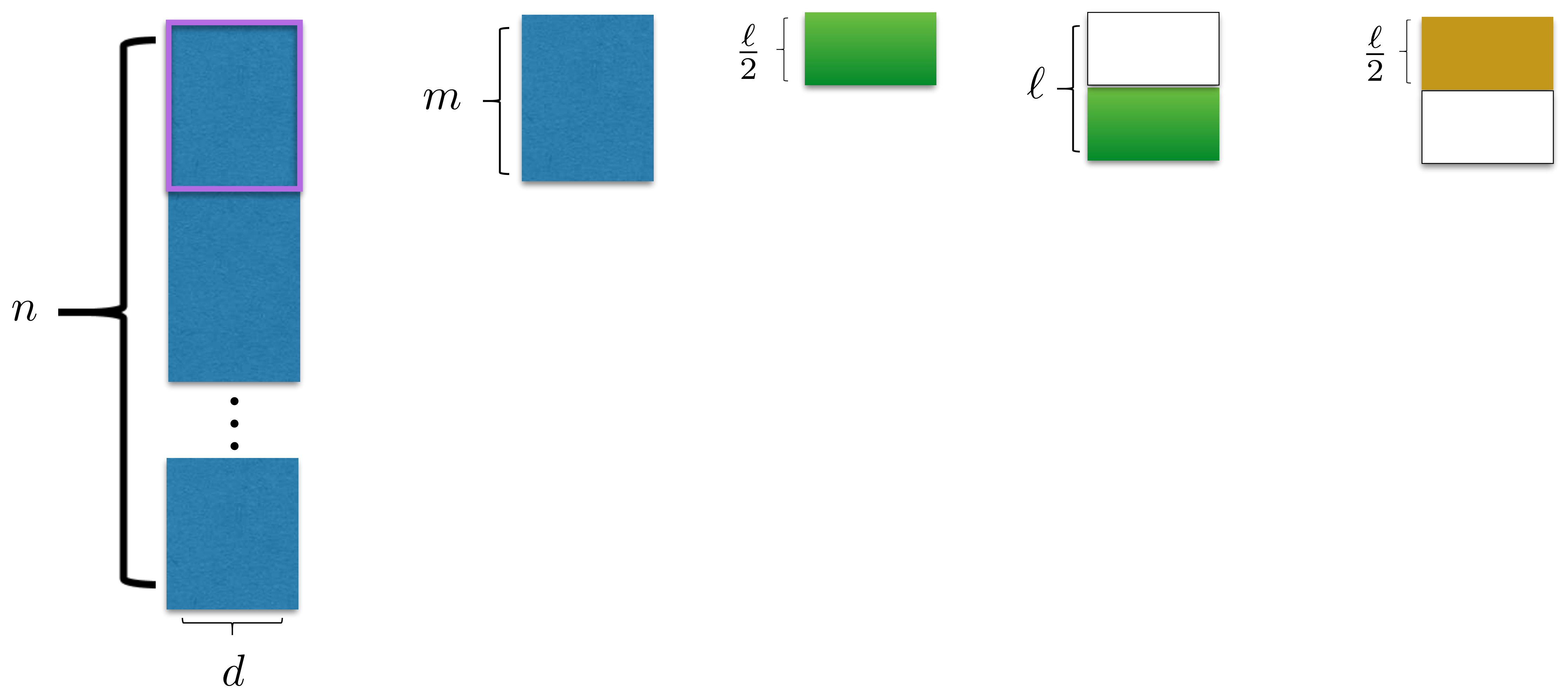
Our Method

- Online instance compression



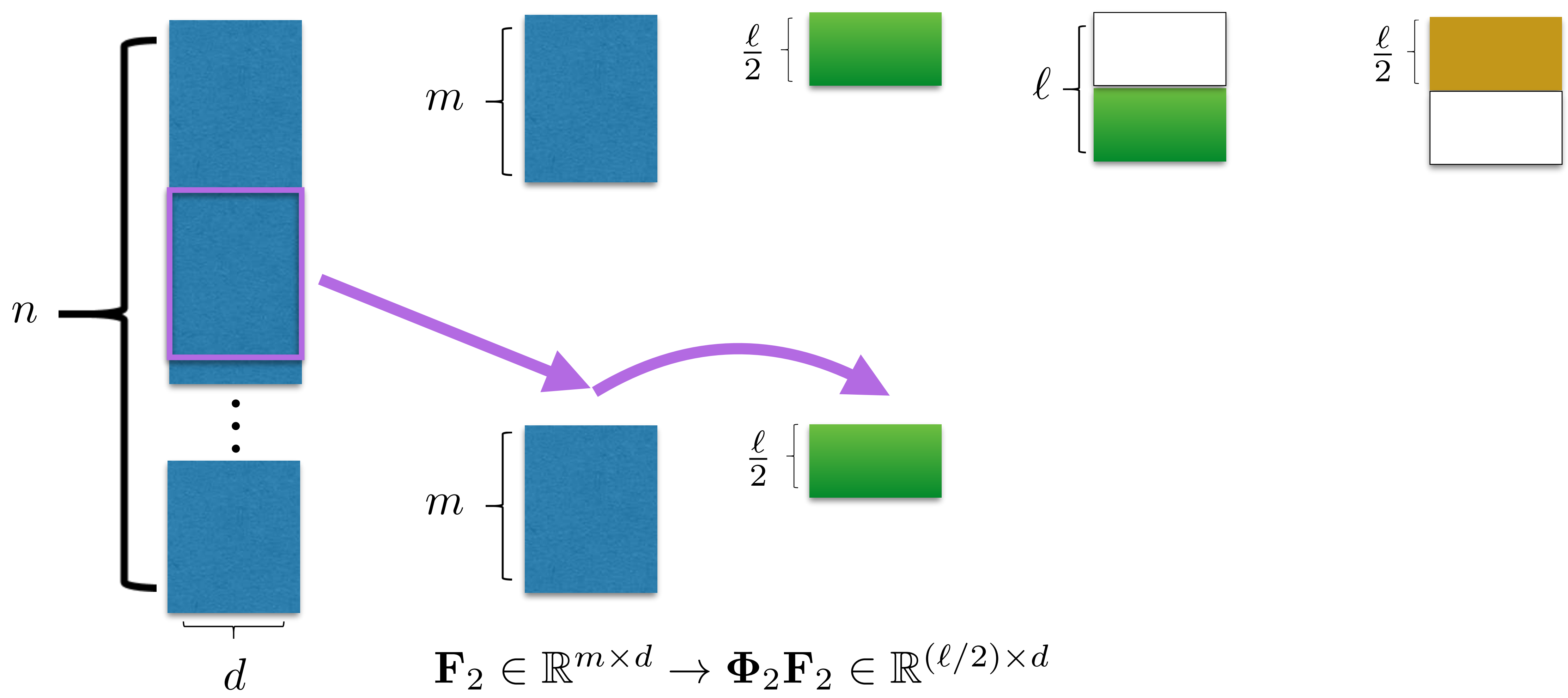
Our Method

- Online instance compression



Our Method

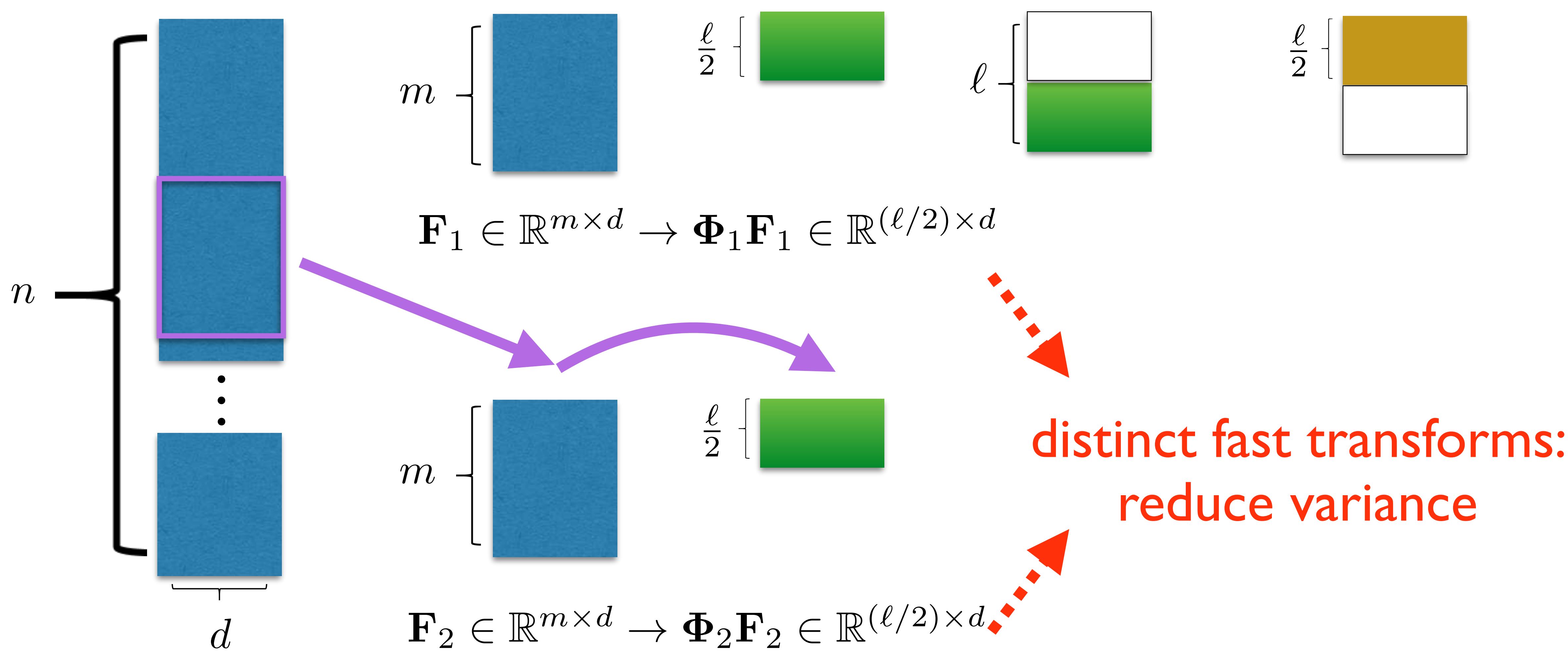
- Online instance compression



$$\mathbf{F}_2 \in \mathbb{R}^{m\times d} \rightarrow \mathbf{\Phi}_2\mathbf{F}_2 \in \mathbb{R}^{(\ell/2)\times d}$$

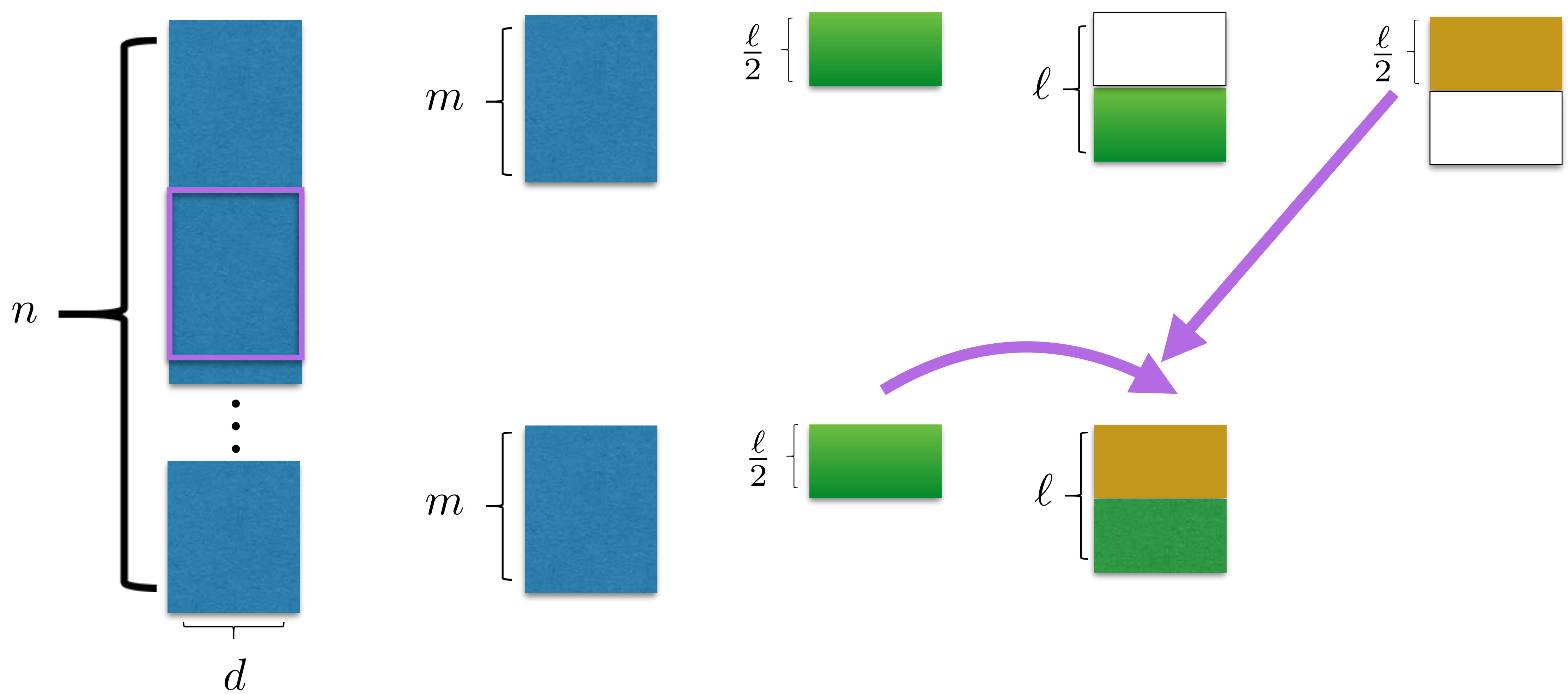
Our Method

- Online instance compression



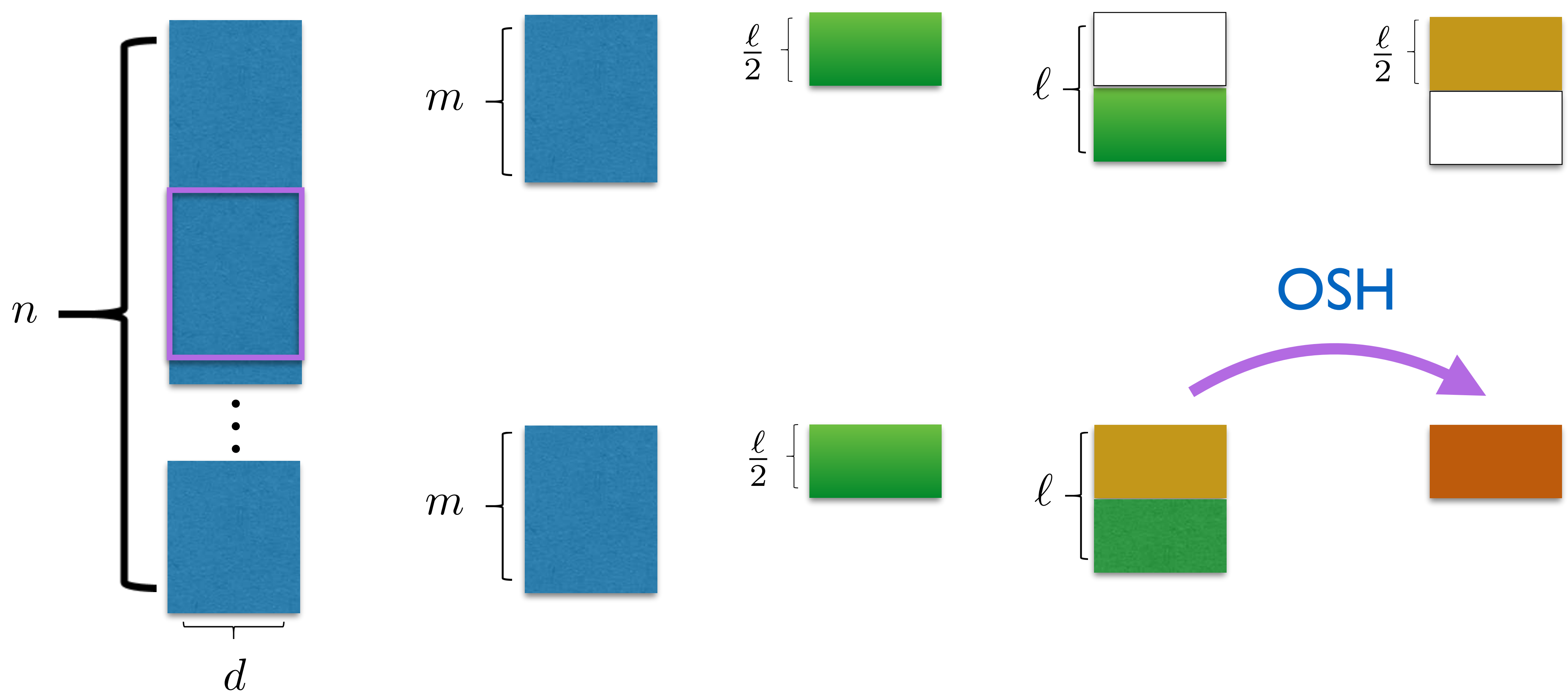
Our Method

- Online instance compression



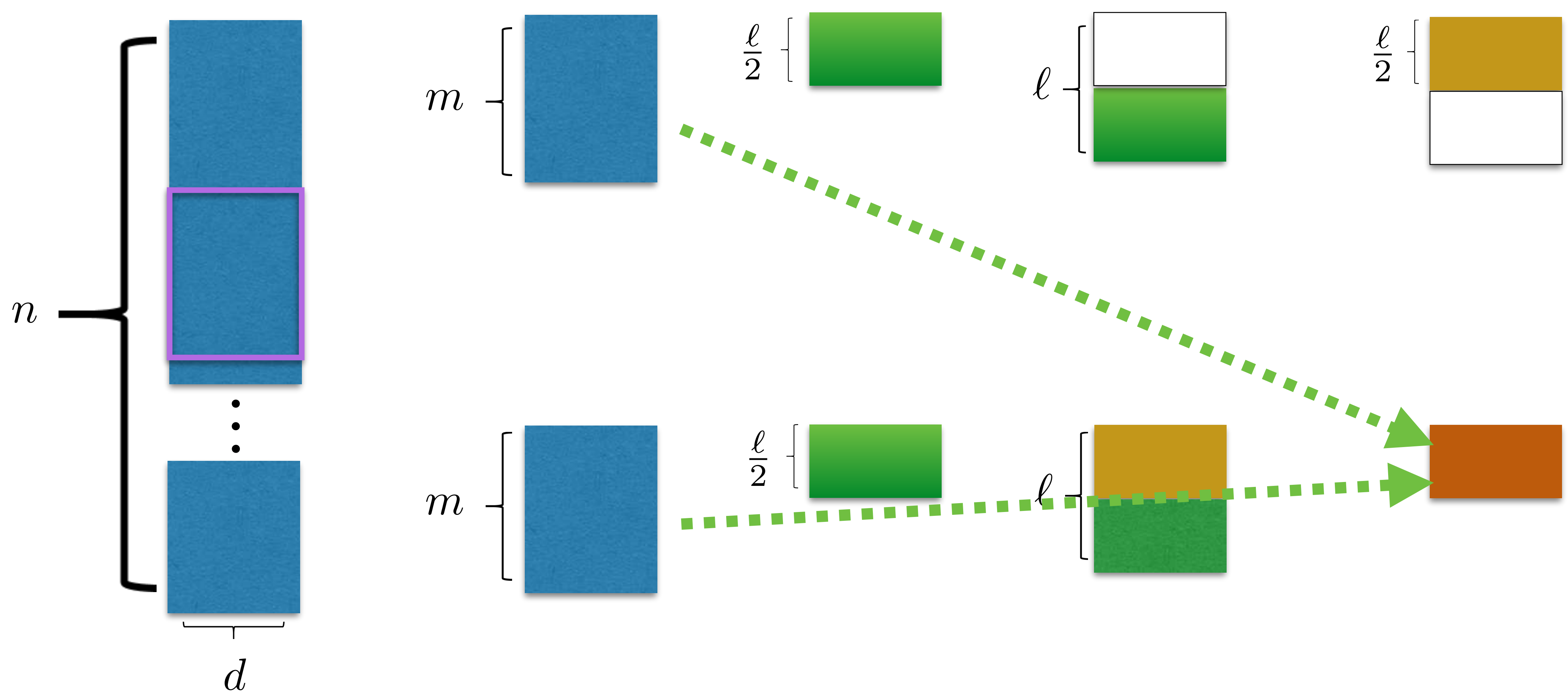
Our Method

- Online instance compression



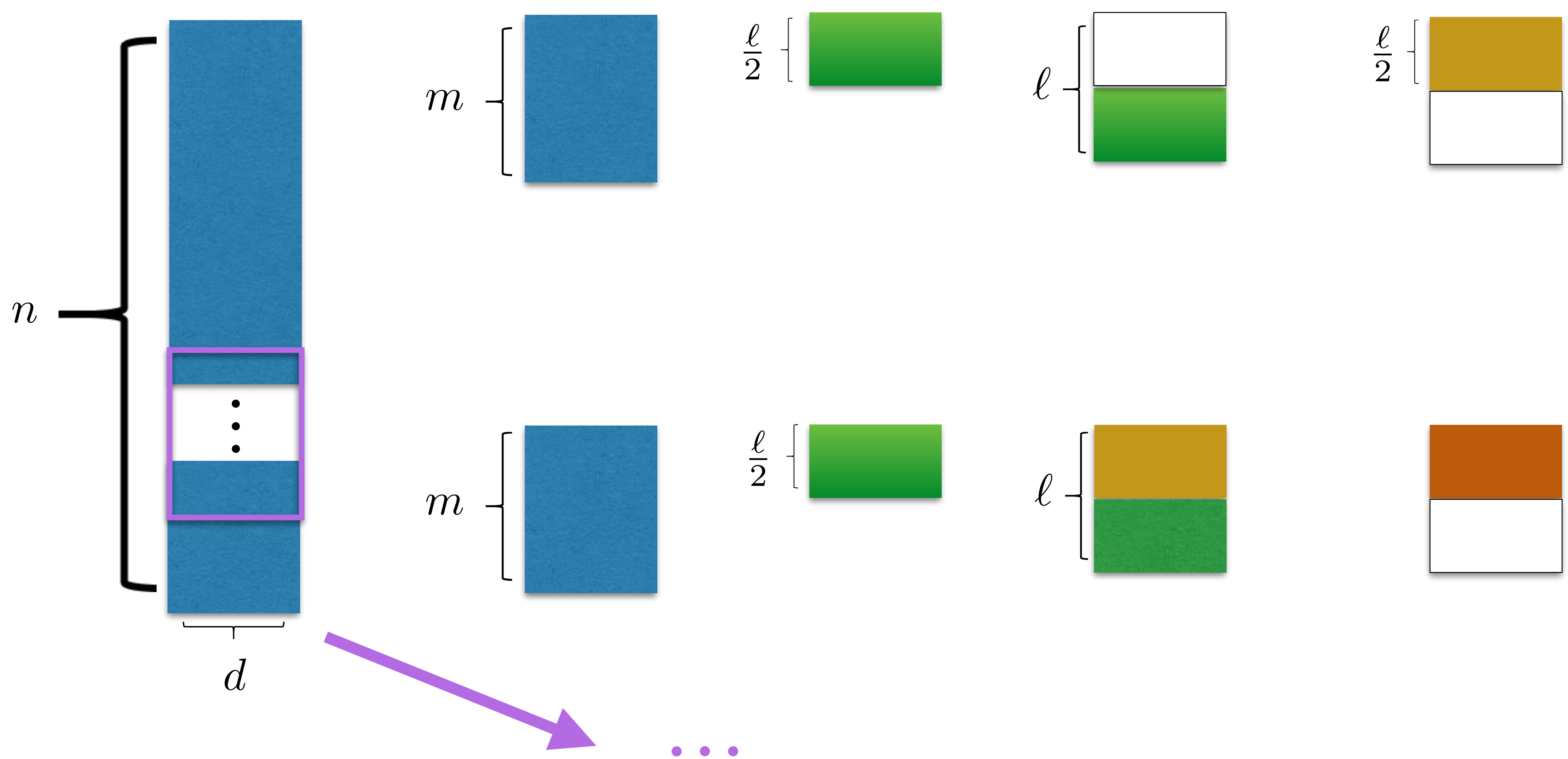
Our Method

- Online instance compression



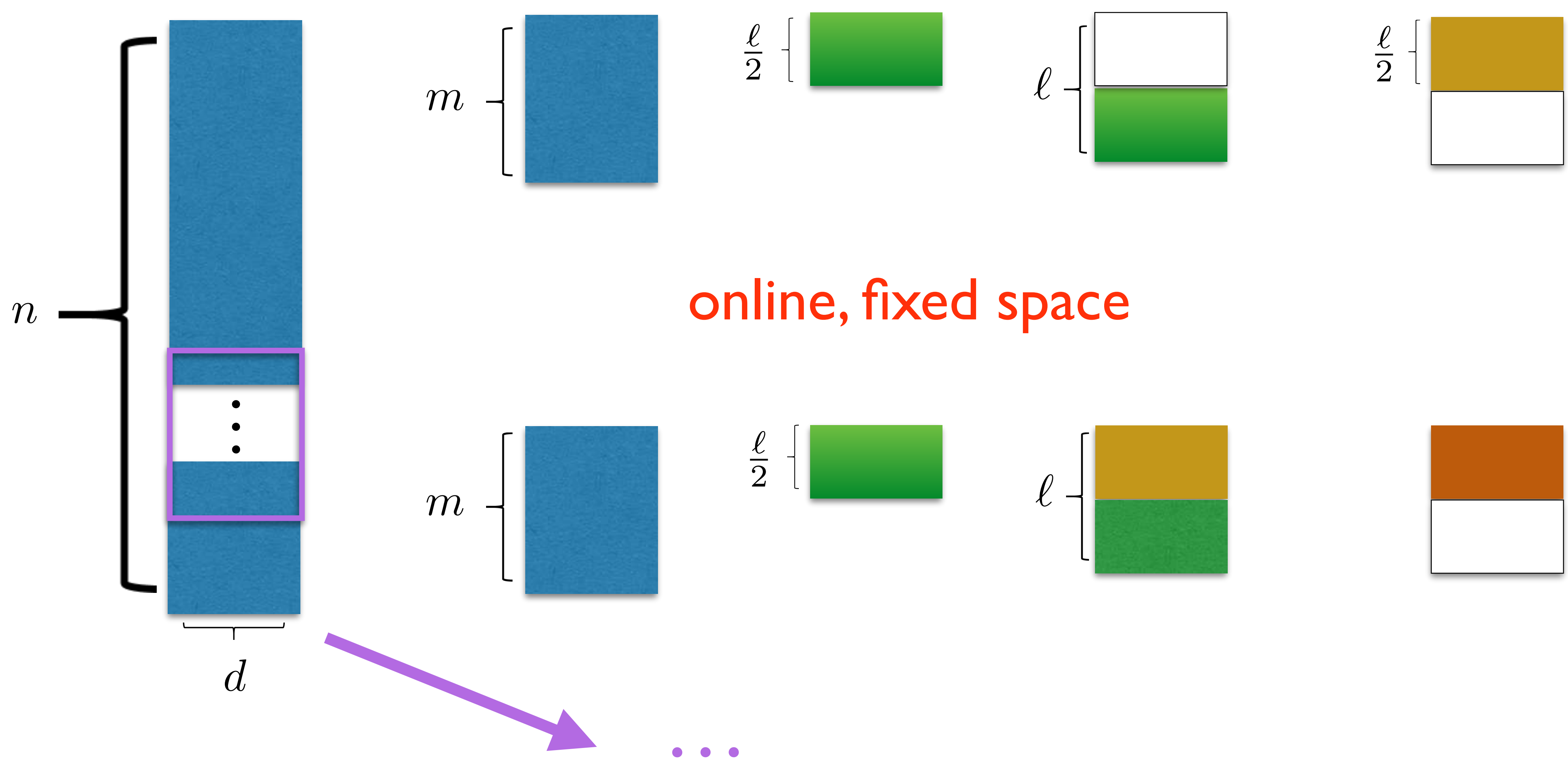
Our Method

- Online instance compression



Our Method

- Online instance compression

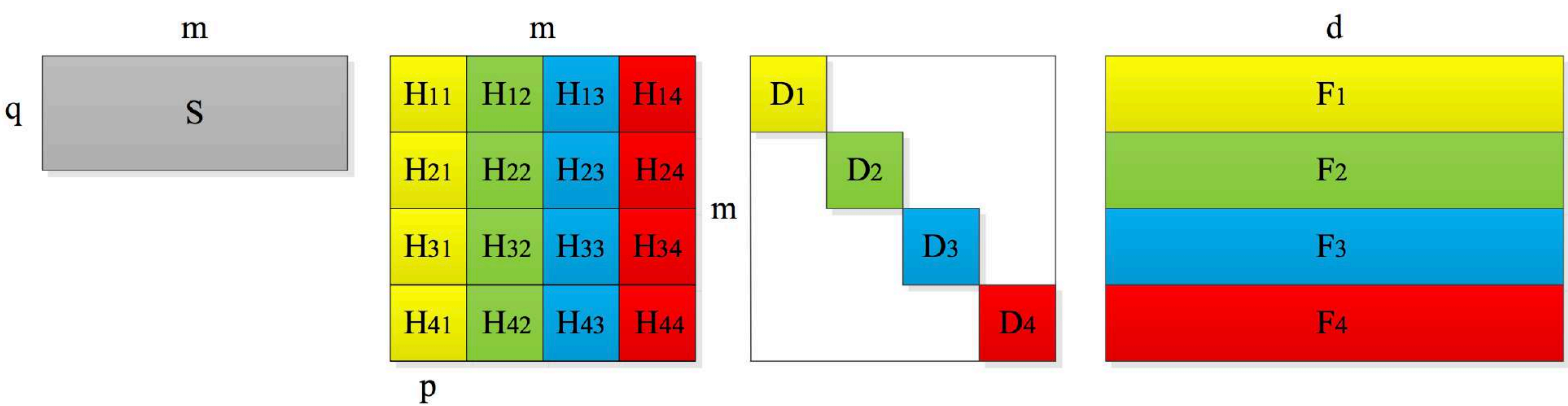


Our Method

- Compress $\mathbf{F} \in \mathbb{R}^{m \times d}$ via fast transform $\mathbf{\Phi F} \in \mathbb{R}^{(\ell/2) \times d}$
  - Typical: $O(md \log \ell)$ time and $O(md)$ space
  - Our: $O(md \log \ell)$ time and $O(\ell d)$ space

Our Method

- Compress $\mathbf{F} \in \mathbb{R}^{m \times d}$ via fast transform $\mathbf{\Phi F} \in \mathbb{R}^{(\ell/2) \times d}$
  - Typical: $O(md \log \ell)$ time and $O(md)$ space
  - Our: $O(md \log \ell)$ time and $O(\ell d)$ space
- Our implementation of $\mathbf{\Phi F} = \mathbf{SHDF}$



$$\{\mathbf{H}_{ij}\}_{i \geq 2} = \mathbf{H}_{1j} \text{ or } -\mathbf{H}_{1j}$$

$$\mathbf{H}_{ij} = (-1)^{\langle i-1, j-1 \rangle}$$

Results

- Theorem 4.2 (FROSH). Given data $\mathbf{A} \in \mathbb{R}^{n\times d}$ with with its row mean vector $\boldsymbol{\mu} \in \mathbb{R}^{1\times d}$, let the sketch $\mathbf{B} \in \mathbb{R}^{\ell\times d}$ be generated by FROSH. Then, with probability at least $1 - p\beta - (2p+1)\delta - \frac{2n}{e^k}$, we have

$$\|(\mathbf{A}-\boldsymbol{\mu})^T(\mathbf{A}-\boldsymbol{\mu}) - \mathbf{B}^T\mathbf{B}\|_2 \leq \widetilde{O}\Big(\frac{1}{\ell} + \Gamma(\ell, p, k)\Big)\|\mathbf{A}-\boldsymbol{\mu}\|_F^2,$$

  where $(\mathbf{A}-\boldsymbol{\mu}) \in \mathbb{R}^{n\times d}$ means subtracting each row of $\mathbf{A}$ by $\boldsymbol{\mu}$, $\widetilde{O}(\cdot)$ hides logarithmic factors on (β, δ, k, d, m), $\Gamma(\ell, p, k) = \sqrt{\frac{k}{\ell p^2}} + \sqrt{\frac{1+\sqrt{k/\ell}}{p}}$ with $p = \frac{n}{m}$, the top r right singular vectors of $\mathbf{B} \in \mathbb{R}^{\ell\times d}$ are for hashing projections $\mathbf{W}^T \in \mathbb{R}^{r\times d}$, and the algorithm requires $O(d\ell)$ space and $\widetilde{O}(n\ell^2 + nd + d\ell^2)$ running time after taking $m = \Theta(d)$.

Results

- Theorem 4.2 (FROSH). Given data $\mathbf{A} \in \mathbb{R}^{n\times d}$ with with its row mean vector $\boldsymbol{\mu} \in \mathbb{R}^{1\times d}$, let the sketch $\mathbf{B} \in \mathbb{R}^{\ell\times d}$ be generated by FROSH. Then, with probability at least $1 - p\beta - (2p+1)\delta - \frac{2n}{e^k}$, we have

$$\|(\mathbf{A}-\boldsymbol{\mu})^T(\mathbf{A}-\boldsymbol{\mu}) - \mathbf{B}^T\mathbf{B}\|_2 \leq \widetilde{O}\Big(\frac{1}{\ell} + \Gamma(\ell,p,k)\Big)\|\mathbf{A}-\boldsymbol{\mu}\|_F^2,$$

$\mathbf{X}^T\mathbf{X}$

where [illegible]^d means subtracting each row of $\mathbf{A}$ by $\boldsymbol{\mu}$, $\widetilde{O}(\cdot)$ hides logarithmic factors on (β,δ,k,d,m), $\Gamma(\ell,p,k) = \sqrt{\frac{k}{\ell p^2}} + \sqrt{\frac{1+\sqrt{k/\ell}}{p}}$ with $p = \frac{n}{m}$, the top r right singular vectors of $\mathbf{B} \in \mathbb{R}^{\ell\times d}$ are for hashing projections $\mathbf{W}^T \in \mathbb{R}^{r\times d}$, and the algorithm requires $O(d\ell)$ space and $\widetilde{O}(n\ell^2 + nd + d\ell^2)$ running time after taking $m = \Theta(d)$.

Results

- Corollary 4.1 (FROSH). Given data $\mathbf{A} \in \mathbb{R}^{n \times d}$ with with its row mean vector $\boldsymbol{\mu} \in \mathbb{R}^{1 \times d}$, let the sketch $\mathbf{B} \in \mathbb{R}^{\ell \times d}$ be generated by FROSH. Let $m = \Theta(d)$, and assume $n = \Omega(\ell^{3/2} d^{3/2})$ for simplicity. Given $(\mathbf{A} - \boldsymbol{\mu}) \in \mathbb{R}^{n \times d}$ that means subtracting each row of $\mathbf{A}$ by $\boldsymbol{\mu}$, let $h = \|(\mathbf{A} - \boldsymbol{\mu})\|_F^2 / \|(\mathbf{A} - \boldsymbol{\mu})\|_2^2$ and σ_i be the i-th largest singular value of $(\mathbf{A} - \boldsymbol{\mu})$. If the sketching size $\ell = \widetilde{\Omega}(\frac{h\sigma_1^2}{\epsilon \sigma_{r+1}^2})$, then with probability defined in Theorem 4.2 we have

$$\|(\mathbf{A} - \boldsymbol{\mu}) - (\mathbf{A} - \boldsymbol{\mu})\mathbf{W}_\mathbf{B}\mathbf{W}_\mathbf{B}^T\|_2^2 \\ \leq (1+\epsilon)\|(\mathbf{A} - \boldsymbol{\mu}) - (\mathbf{A} - \boldsymbol{\mu})\mathbf{W}\mathbf{W}^T\|_2^2,$$

where $0 < \epsilon < 1$, $\mathbf{W}_\mathbf{B}^T \in \mathbb{R}^{r \times d}$ contains the top r right singular vectors of $\mathbf{B} \in \mathbb{R}^{\ell \times d}$, and $\mathbf{W}^T \in \mathbb{R}^{r \times d}$ contains the top r right singular vectors of $(\mathbf{A} - \boldsymbol{\mu})$.

Results

- Theorem 4.2 & Corollary 4.1 vs. OSH
  - Less time cost ($\widetilde{O}(n\ell^2 + nd + d\ell^2)$ vs. $O(nd\ell + d\ell^2)$) for $m = O(d)$
  - Equal space cost
  - Comparable hashing accuracy

Experiments

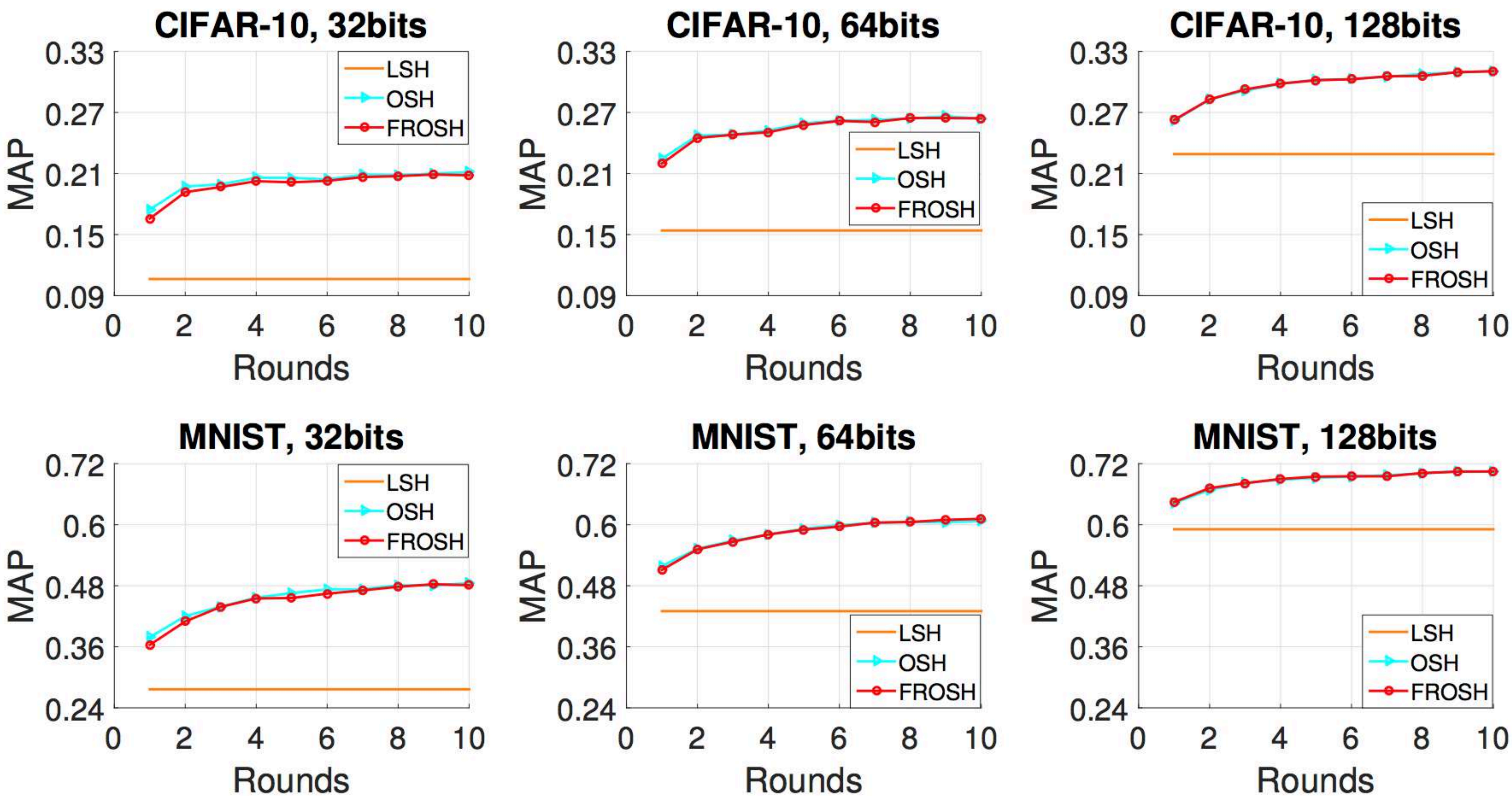
- Setting
  - $m = 4d$ for $\mathbf{\Phi} \in \mathbb{R}^{(\ell/2)\times m}$ and $\mathbf{F} \in \mathbb{R}^{m\times d}$
  - $\ell = 2r$, where $r \sim \{32, 64, 128\}$ is the hashing code length
- Compared methods
  - Unsupervised online hashing: LSH [M. Charikar, et al., 2002], OSH [C. Leng, et al., 2015], FROSH
  - Unsupervised batch-based hashing: SGH [Q. Jiang, et al., 2015], OCH [H. Liu, et al., 2017]

Real Data

Dataset	Size	Dimension
CIFAR-10	60,000	512
MNIST	70,000	784
GIST-1M	1,000,000	960
FLICKR-25600	100,000	25,600

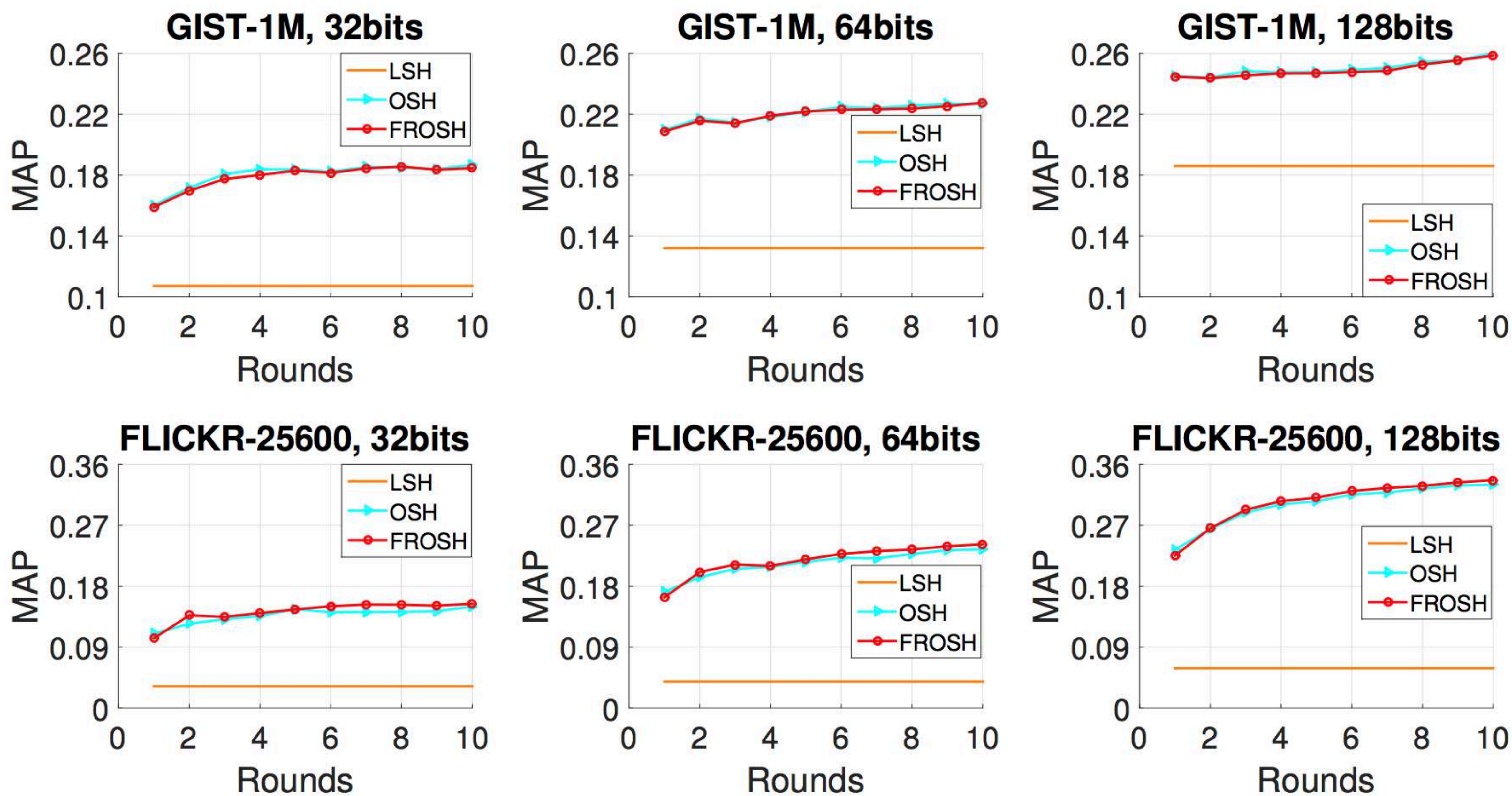
Experiments

- MAP comparisons with unsupervised online hashing



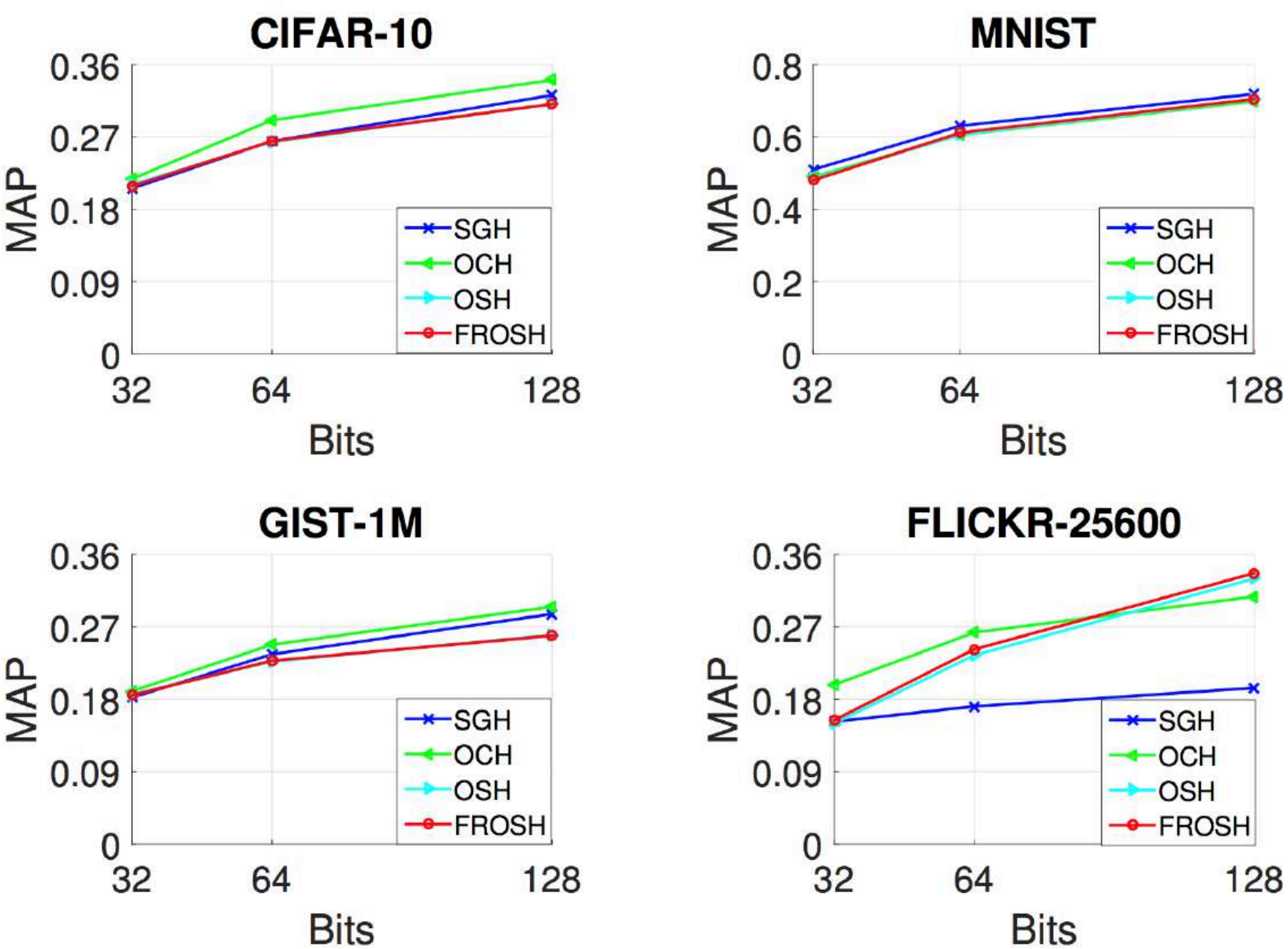
Experiments

- MAP comparisons with unsupervised online hashing



Experiments

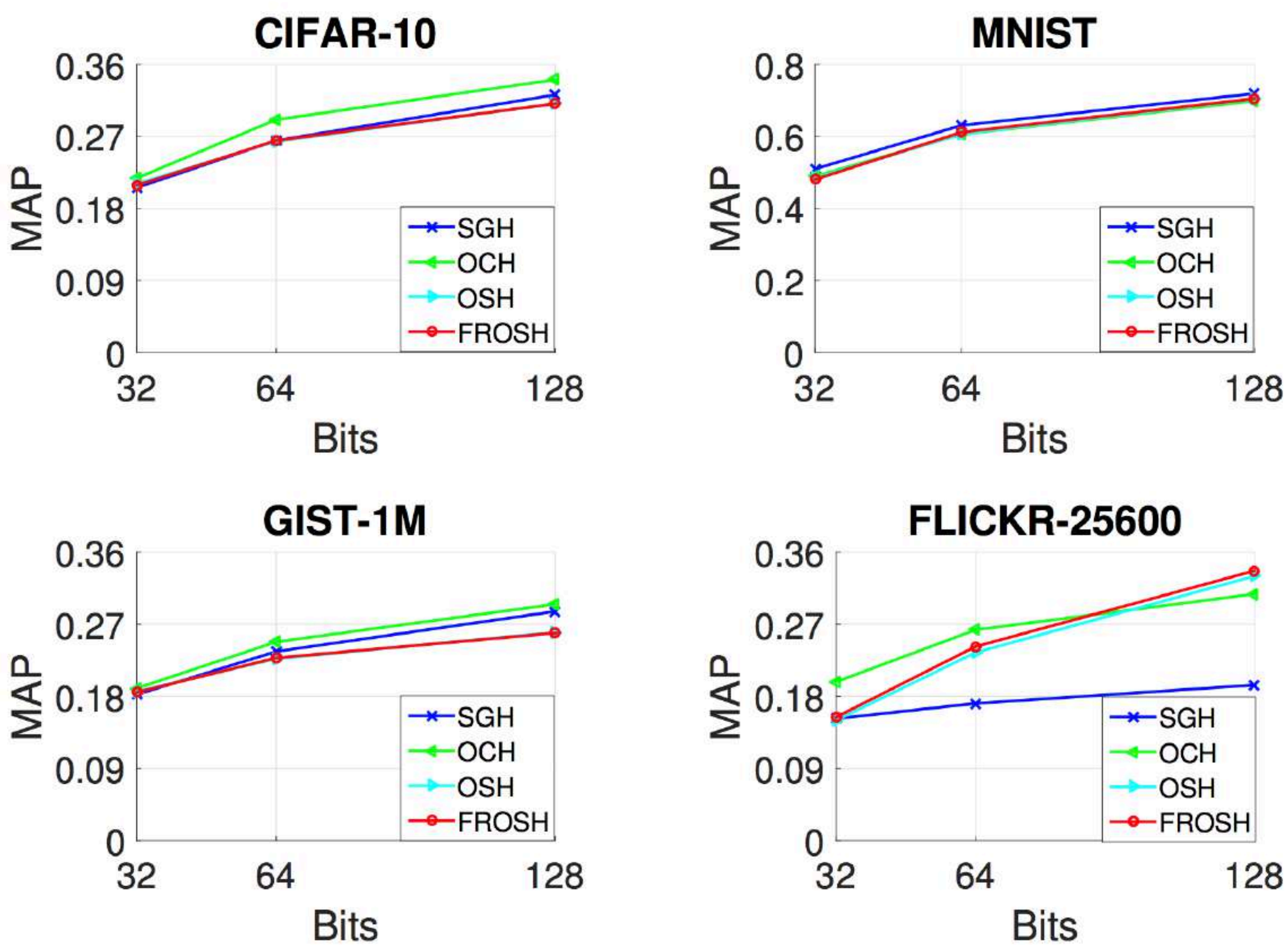
- MAP comparisons
- $10 \sim 70$ times speed-up



Dataset	Method	32bits	64bits	128bits
CIFAR-10	SGH	7.83	11.35	19.49
	OCH	26.89	26.95	27.49
	OSH	7.78	11.88	22.09
	FROSH	**0.63**	**0.94**	**2.11**
MNIST	SGH	10.47	14.59	23.47
	OCH	40.45	40.49	41.10
	OSH	13.25	18.93	30.75
	FROSH	**1.17**	**1.49**	**2.56**
GIST-1M	SGH	231	275	290
	OCH	1042	1089	1192
	OSH	228	331	520
	FROSH	**21**	**27**	**45**
FLICKR-25600	SGH	3032	3541	4903
	OCH	4981	5300	5441
	OSH	679	1283	2570
	FROSH	**72**	**92**	**134**

Experiments

- MAP comparisons



- $10 \sim 70$ times speed-up

Dataset	Method	32bits	64bits	128bits
CIFAR-10	SGH	7.83	11.35	19.49
	OCH	26.89	26.95	27.49
	OSH	7.78	11.88	22.09
	FROSH	**0.63**	**0.94**	**2.11**
MNIST	SGH	10.47	14.59	23.47
	OCH	40.45	40.49	41.10
	OSH	13.25	18.93	30.75
	FROSH	**1.17**	**1.49**	**2.56**
GIST-1M	SGH	231	275	290
	OCH	1042	1089	1192
	OSH	228	331	520
	FROSH	**21**	**27**	**45**
FLICKR-25600	SGH	3032	3541	4903
	OCH	4981	5300	5441
	OSH	679	1283	2570
	FROSH	**72**	**92**	**134**

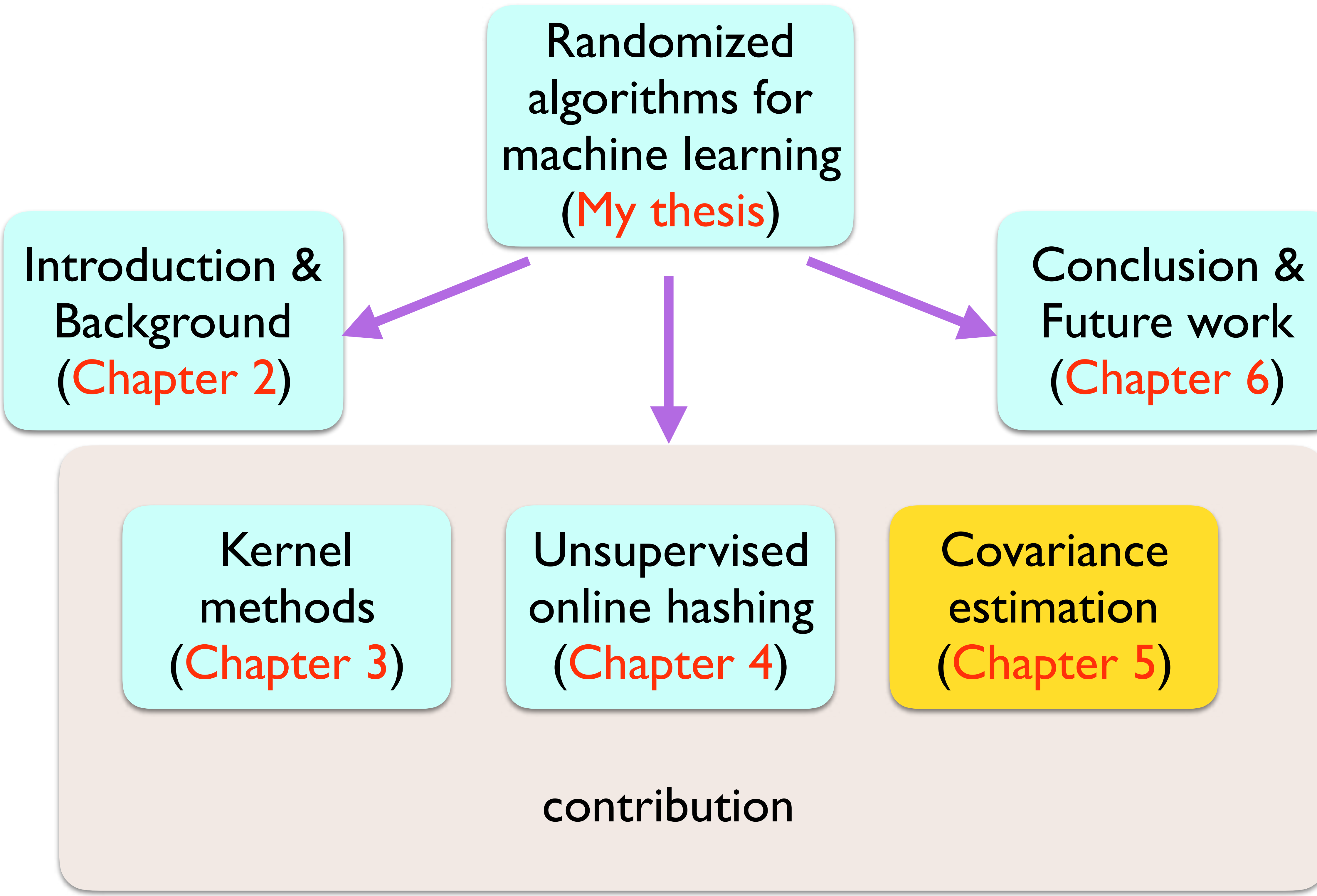
Experiments

- Space cost on FLICKR-25600
  - Batch-based hashing: $> 19\mathrm{GB}$
  - OSH, FROSH: $> 0.05\mathrm{GB}$

Conclusion

- Present a faster online sketching hashing method by designing randomized algorithms
- Demonstrate the good performance with provable results, complexity analysis, and extensive experiments

Outline

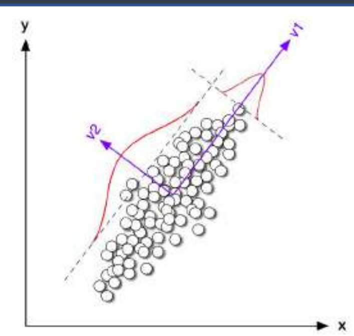


Randomized algorithms for machine learning (My thesis)
Introduction & Background (Chapter 2)
Conclusion & Future work (Chapter 6)
Kernel methods (Chapter 3)
Unsupervised online hashing (Chapter 4)
Covariance estimation (Chapter 5)
contribution

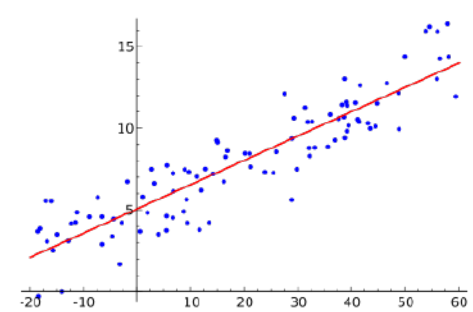
Background

- Covariance matrix:
  - Definition: $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T \; (\mathbf{X} \in \mathbb{R}^{d \times n})$ [W. Feller, 1966]
  - Applications:

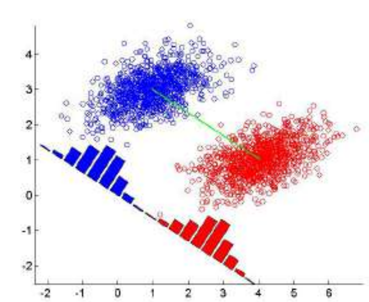
Principal Component Analysis



Generalized Least Squares



Linear Discriminant Analysis

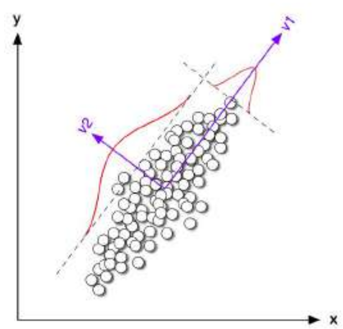


Background

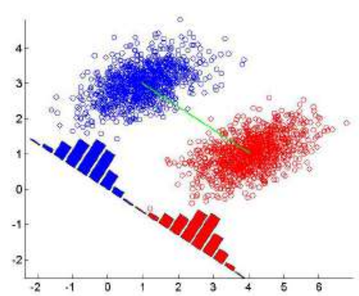
- Covariance matrix:
  - Definition: $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T \ (\mathbf{X} \in \mathbb{R}^{d \times n})$
  - Applications:

$\mathbf{X}^T\mathbf{X}$

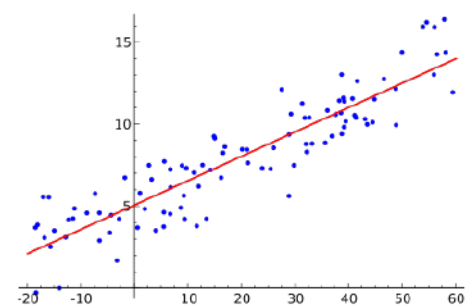
Principal Component



Linear Discriminant Analysis



Generalized Least Squares

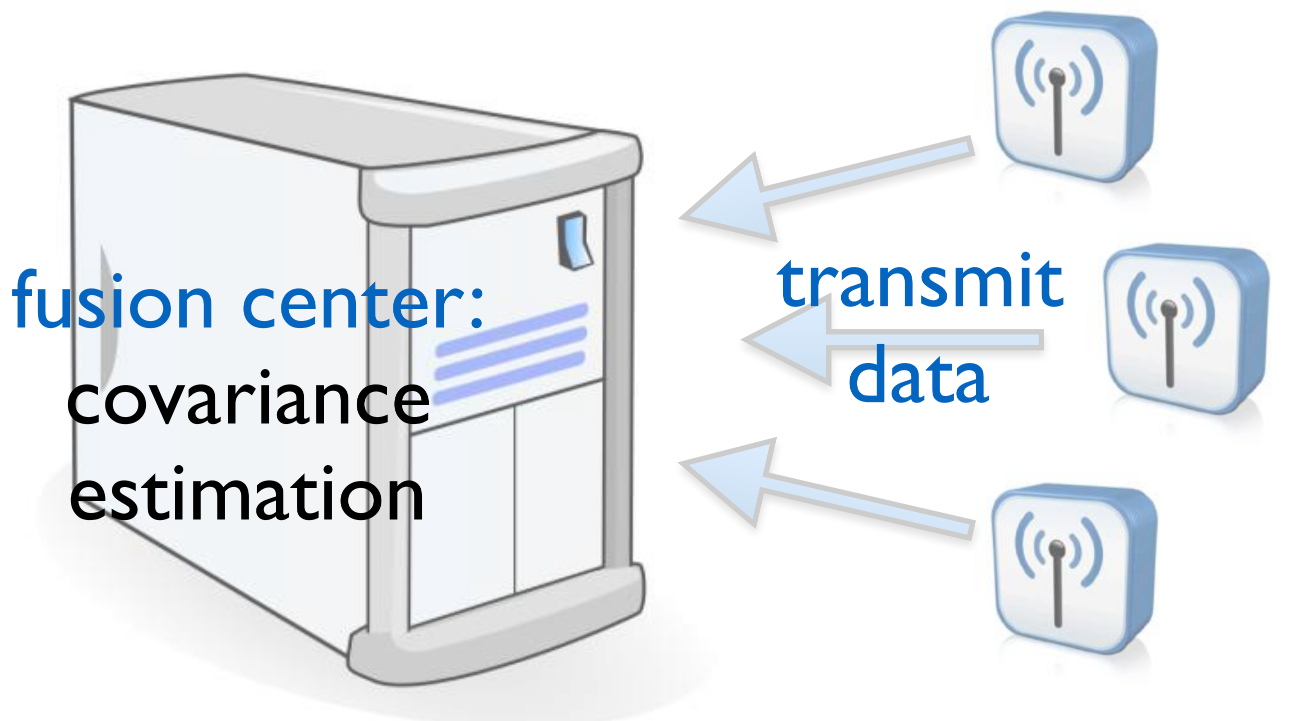


Background

- For $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$ $(\mathbf{X} \in \mathbb{R}^{d\times n})$
  - $O(nd)$ communication burden
  - $O(nd + d^2)$ storage
  - $O(nd^2)$ calculation time

Background

- For $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$ $(\mathbf{X} \in \mathbb{R}^{d\times n})$
  - $O(nd)$ communication burden: data gathered in many distributed remote sites are transmitted to the fusion center to form $\mathbf{C}$
  - $O(nd + d^2)$ storage
  - $O(nd^2)$ calculation time



Background

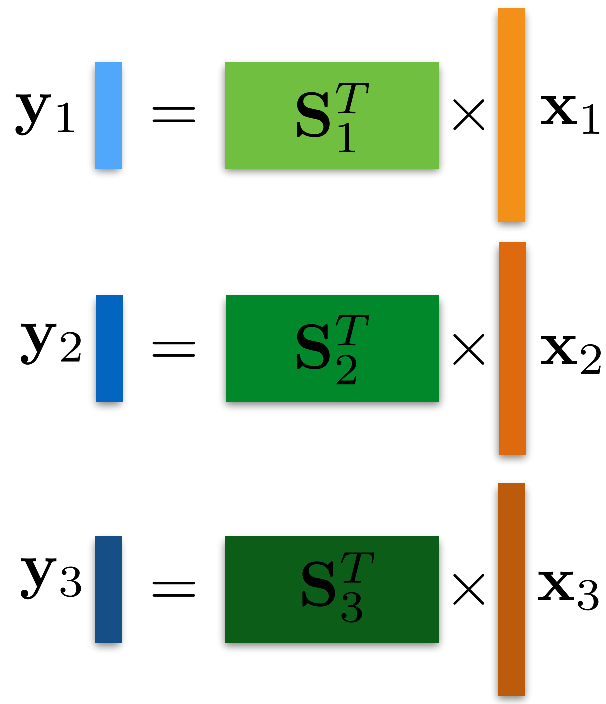
- For $\mathbf{C} = \frac{1}{n}\mathbf{X}\mathbf{X}^T$ $(\mathbf{X} \in \mathbb{R}^{d \times n})$
  - $O(nd)$ communication burden
  - $O(nd + d^2)$ storage
  - $O(nd^2)$ calculation time

computationally expensive, when $n, d \gg 1$

Related Work

- Data compression

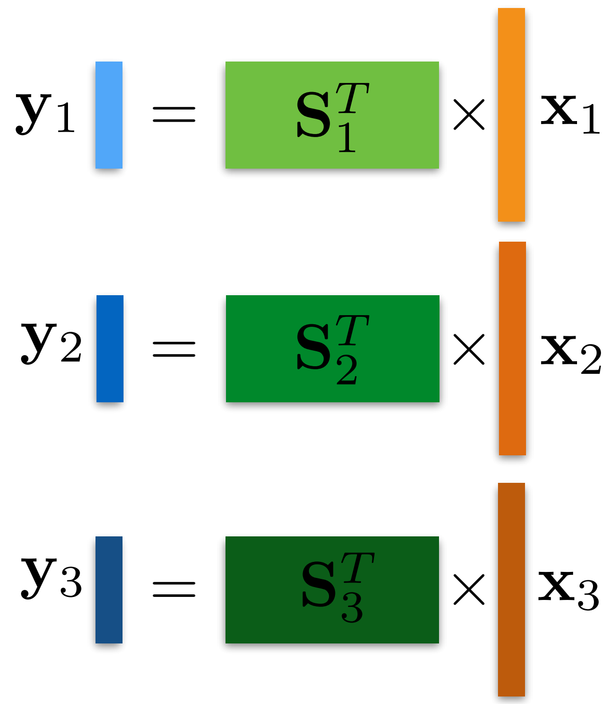
$\mathbf{X} \in \mathbb{R}^{d \times n} \rightarrow \mathbf{Y} \in \mathbb{R}^{m \times n}$ $(\mathbf{y}_i = \mathbf{S}_i^T \mathbf{x}_i \in \mathbb{R}^m$, $\mathbf{S}_i \in \mathbb{R}^{d \times m}$ and $m < d)$



Related Work

- Data compression

$\mathbf{X} \in \mathbb{R}^{d \times n} \rightarrow \mathbf{Y} \in \mathbb{R}^{m \times n}$ $(\mathbf{y}_i = \mathbf{S}_i^T \mathbf{x}_i \in \mathbb{R}^m$, $\mathbf{S}_i \in \mathbb{R}^{d \times m}$ and $m < d)$

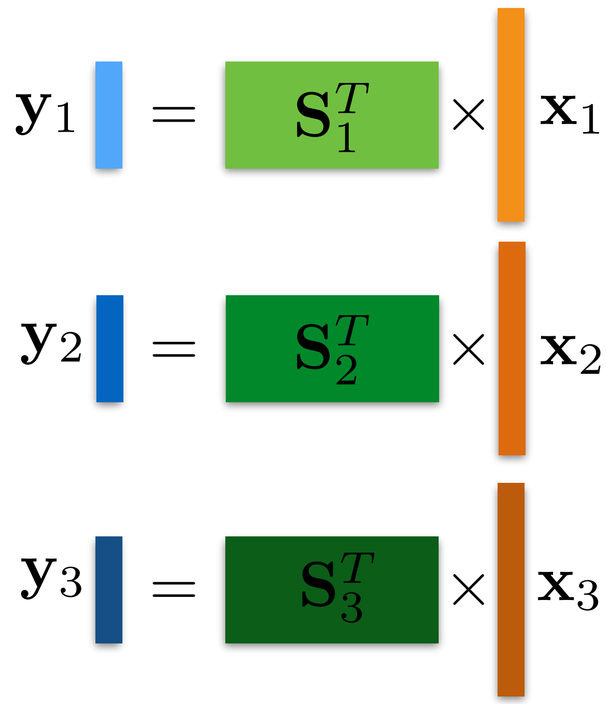


transmit compressed data:
reduce communication

Related Work

- Data compression

$\mathbf{X} \in \mathbb{R}^{d\times n} \rightarrow \mathbf{Y} \in \mathbb{R}^{m\times n}$ $(\mathbf{y}_i = \mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^m$, $\mathbf{S}_i \in \mathbb{R}^{d\times m}$ and $m < d)$



- Recovery

$\mathbf{C}_e = \frac{1}{n}\sum_{i=1}^{n} \mathbf{S}_i\mathbf{y}_i\mathbf{y}_i^T\mathbf{S}_i^T$ with debiasing

Related Work

- Data compression

$\mathbf{X} \in \mathbb{R}^{d\times n} \rightarrow \mathbf{Y} \in \mathbb{R}^{m\times n}$ $(\mathbf{y}_i = \mathbf{S}_i^T\mathbf{x}_i \in \mathbb{R}^m$, $\mathbf{S}_i \in \mathbb{R}^{d\times m}$ and $m < d)$

$$\mathbf{y}_1 = \mathbf{S}_1^T \times \mathbf{x}_1$$

$$\mathbf{y}_2 = \mathbf{S}_2^T \times \mathbf{x}_2$$

$$\mathbf{y}_3 = \mathbf{S}_3^T \times \mathbf{x}_3$$

might reduce
space and time costs

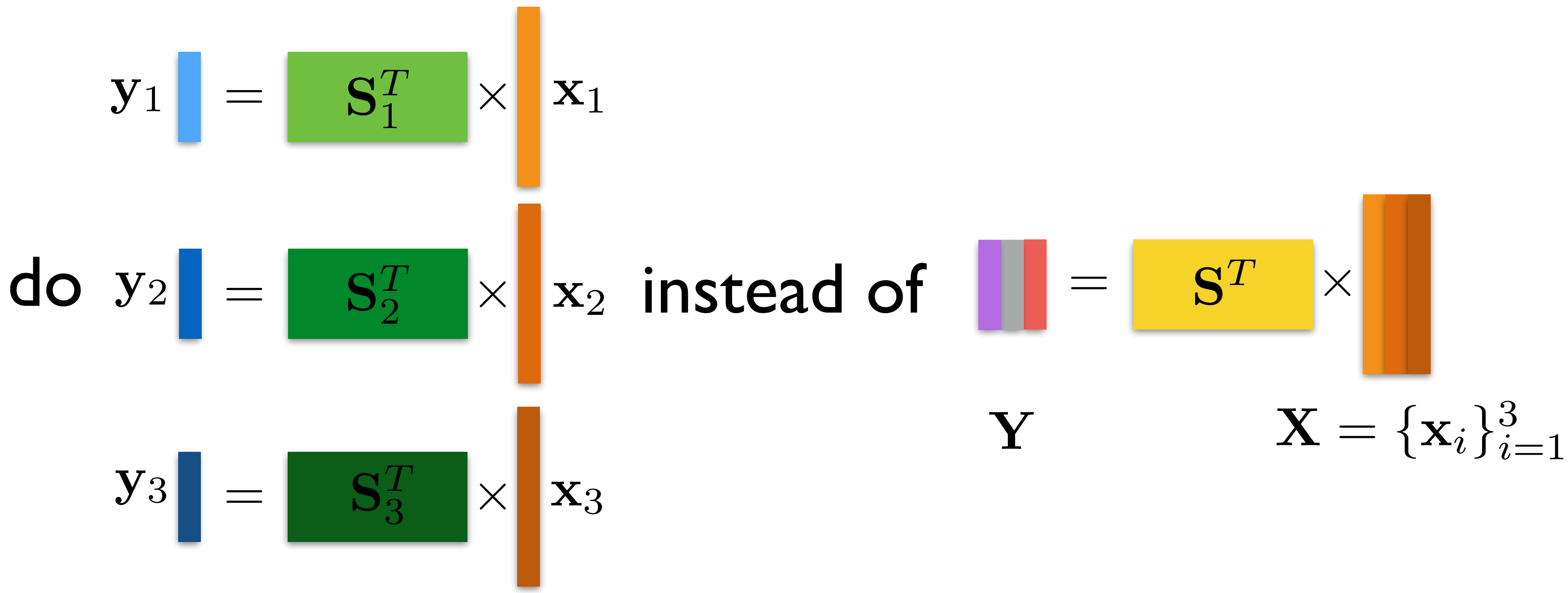
- Recovery

$\mathbf{C}_e = \frac{1}{n}\sum_{i=1}^{n} \mathbf{S}_i\mathbf{y}_i\mathbf{y}_i^T\mathbf{S}_i^T$ with debiasing

Related Work

- Data compression

$\mathbf{X} \in \mathbb{R}^{d \times n} \to \mathbf{Y} \in \mathbb{R}^{m \times n}$ $(\mathbf{y}_i = \mathbf{S}_i^T \mathbf{x}_i \in \mathbb{R}^m$, $\mathbf{S}_i \in \mathbb{R}^{d \times m}$ and $m < d)$



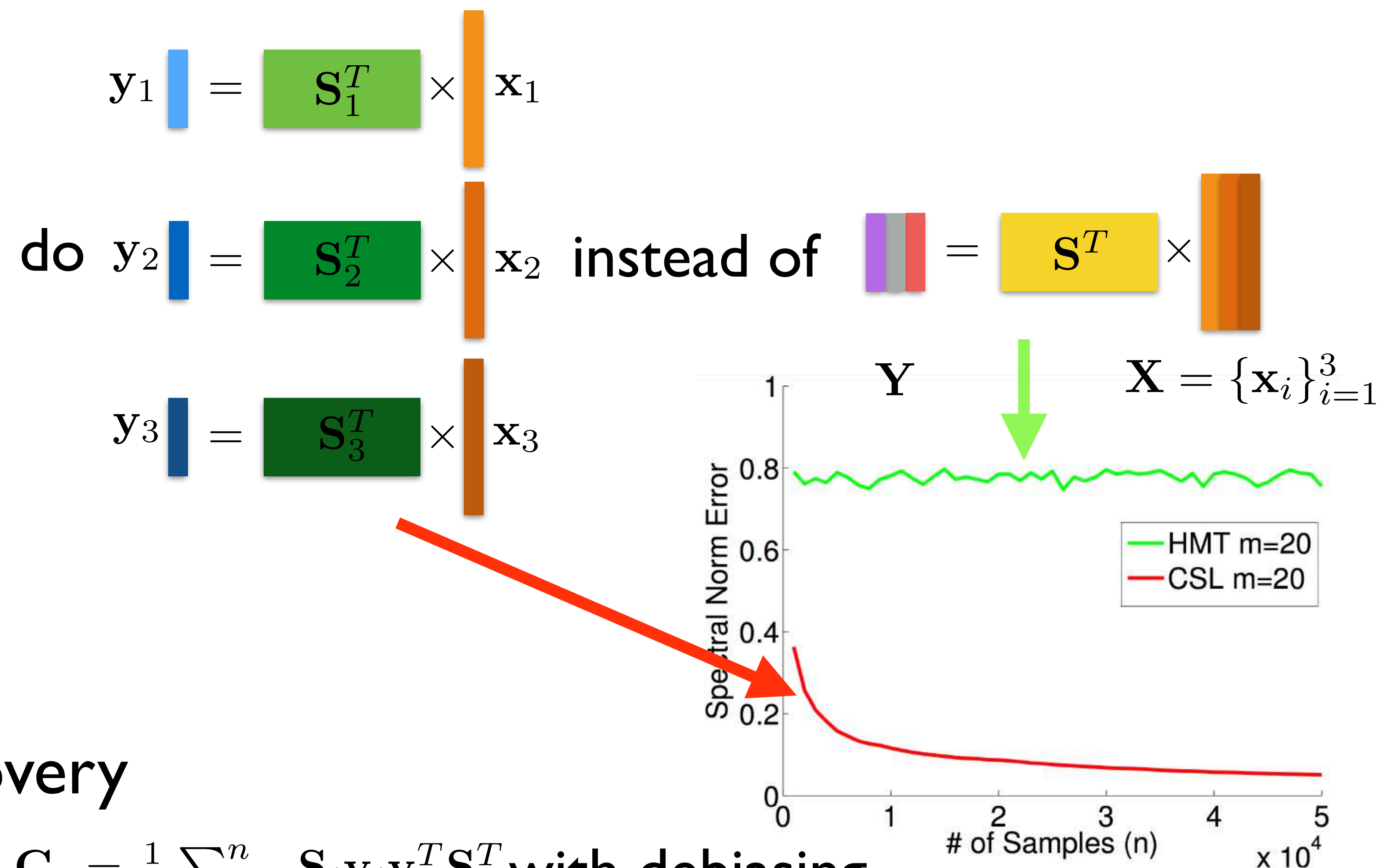
- Recovery

$\mathbf{C}_e = \frac{1}{n} \sum_{i=1}^{n} \mathbf{S}_i \mathbf{y}_i \mathbf{y}_i^T \mathbf{S}_i^T$ with debiasing

Related Work

- Data compression

$\mathbf{X} \in \mathbb{R}^{d \times n} \to \mathbf{Y} \in \mathbb{R}^{m \times n}$ ($\mathbf{y}_i = \mathbf{S}_i^T \mathbf{x}_i \in \mathbb{R}^m$, $\mathbf{S}_i \in \mathbb{R}^{d \times m}$ and $m < d$)



- Recovery

$\mathbf{C}_e = \frac{1}{n}\sum_{i=1}^n \mathbf{S}_i \mathbf{y}_i \mathbf{y}_i^T \mathbf{S}_i^T$ with debiasing

Related Work

- Related work concerning $\mathbf{y}_i = \mathbf{S}_i^T \mathbf{x}_i \in \mathbb{R}^m$, $\mathbf{S}_i \in \mathbb{R}^{d \times m}$
  - Gauss-Inverse: $\frac{1}{n} \sum_{i=1}^{n} \mathbf{S}_i (\mathbf{S}_i^T \mathbf{S}_i)^{-1} \mathbf{S}_i^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{S}_i (\mathbf{S}_i^T \mathbf{S}_i)^{-1} \mathbf{S}_i^T$ [M. Azizyan, et al., 2015]
    - $\mathbf{S}_i$: a Gaussian matrix
    - accurate, computationally expensive
  - Sparse: $\frac{1}{n} \sum_{i=1}^{n} \mathbf{S}_i \mathbf{S}_i^T \mathbf{x}_i \mathbf{x}_i^T \mathbf{S}_i \mathbf{S}_i^T$ [F. Anaraki, et al., 2016]
    - a sparse matrix
    - less accurate, less computationally expensive, not error-bounded
  - UniSample-HD: $\frac{1}{n} \sum_{i=1}^{n} \mathbf{S}_i \mathbf{S}_i^T \mathbf{z}_i \mathbf{z}_i^T \mathbf{S}_i \mathbf{S}_i^T$, $\mathbf{z}_i = \mathbf{H}\mathbf{D}\mathbf{x}_i$ [F. Anaraki, et al., 2017]
    - $\mathbf{S}_i$: a sampling matrix (uniform sampling without replacement)
    - less accurate, efficient

Our Work

- Improve both the estimation accuracy and computational efficiency compared with all previous work

Our Method

- $\mathbf{S}_i$: a weighted sampling matrix
- Sampling probabilities in $\mathbf{S}_i$ to tighten $\|\mathbf{C} - \mathbf{C}_e\|_2$
  - $p_{ki} = \alpha \frac{|x_{ki}|}{\|\mathbf{x}_i\|_1} + (1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$

Our Method

- $\mathbf{S}_i$: a weighted sampling matrix
- Sampling probabilities in $\mathbf{S}_i$ to tighten $\|\mathbf{C}-\mathbf{C}_e\|_2$
  - $p_{ki} = \alpha \frac{|x_{ki}|}{\|\mathbf{x}_i\|_1} + (1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$

for all $i \in [n]$ **do**
  Load $\mathbf{x}_i$ into memory, let $v_i = \|\mathbf{x}_i\|_1 = \sum_{k=1}^{d} |x_{ki}|$ and $w_i = \|\mathbf{x}_i\|_2^2 = \sum_{k=1}^{d} x_{ki}^2$
  for all $j \in [m]$ **do**
    Pick $t_{ji} \in [d]$ with $p_{ki} \equiv \mathbb{P}(t_{ji} = k) = \alpha\frac{|x_{ki}|}{v_i} + (1-\alpha)\frac{x_{ki}^2}{w_i}$, and let $y_{ji} = x_{t_{ji}i}$
  end for
end for

Results

- Theorem 5.1 (Unbiased estimator). The unbiased estimator for the covariance $\mathbf{C} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i\mathbf{x}_i^T = \frac{1}{n}\mathbf{X}\mathbf{X}^T$ can be recovered as

$$\mathbf{C}_e = \widehat{\mathbf{C}}_1 - \widehat{\mathbf{C}}_2,$$

where we have that $\mathbb{E}\left[\mathbf{C}_e\right] = \mathbf{C}$, $\widehat{\mathbf{C}}_1 = \frac{m}{nm-n}\sum_{i=1}^{n}\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T$, and $\widehat{\mathbf{C}}_2 = \frac{m}{nm-n}\sum_{i=1}^{n}\mathbb{D}(\mathbf{S}_i\mathbf{S}_i^T\mathbf{x}_i\mathbf{x}_i^T\mathbf{S}_i\mathbf{S}_i^T)\mathbb{D}(\mathbf{b}_i)$ with $b_{ki} = \frac{1}{1+(m-1)p_{ki}}$.

in the recovery stage, at most m entries of $\mathbf{S}_i$ and $\mathbf{b}_i$ must be calculated, respectively

Results

- Theorem 5.2 (Upper bound). Let $\mathbf{C}_e$ be defined as Theorem 5.1 with the sampling probabilities $p_{ki} = \alpha\frac{|x_{ki}|}{\|\mathbf{x}_i\|_1} + (1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$. Then, with probability at least $1-\eta-\delta$,

$$\|\mathbf{C}_e - \mathbf{C}\|_2 \leq \log(\tfrac{2d}{\delta})\tfrac{2R}{3} + \sqrt{2\sigma^2\log(\tfrac{2d}{\delta})},$$

where we define the range $R = \max_{i\in[n]}\left[\frac{7\|\mathbf{x}_i\|_2^2}{n} + \log^2(\frac{2nd}{\eta})\frac{14\|\mathbf{x}_i\|_1^2}{nm\alpha^2}\right]$, and the variance $\sigma^2 = \sum_{i=1}^n\left[\frac{8\|\mathbf{x}_i\|_2^4}{n^2m^2(1-\alpha)^2} + \frac{4\|\mathbf{x}_i\|_1^2\|\mathbf{x}_i\|_2^2}{n^2m^3\alpha^2(1-\alpha)} + \frac{9\|\mathbf{x}_i\|_2^4}{n^2m(1-\alpha)} + \frac{2\|\mathbf{x}_i\|_2^2\|\mathbf{x}_i\|_1^2}{n^2m^2\alpha(1-\alpha)}\right] + \|\sum_{i=1}^n\frac{\|\mathbf{x}_i\|_1^2\mathbf{x}_i\mathbf{x}_i^2}{n^2m\alpha}\|_2$.

Results

- Corollary 5.1 (Upper bound). Let $\mathbf{C}_e$ be defined as Theorem 5.1. Define $\frac{\|\mathbf{x}_i\|_1}{\|\mathbf{x}_i\|_2} \leq \varphi$ with $1 \leq \varphi \leq \sqrt{d}$, and $\|\mathbf{x}_i\|_2 \leq \tau$ for all $i \in [n]$. Then, with probability at least $1 - \eta - \delta$ we have

$$\|\mathbf{C}_e - \mathbf{C}\|_2 \leq \widetilde{O}\Big(\frac{\tau^2}{n} + \frac{\tau^2\varphi^2}{nm} + \tau\varphi\sqrt{\frac{\|\mathbf{C}\|_2}{nm}} + f\Big),$$

where $f = \min\{\frac{\tau^2\varphi}{m}\sqrt{\frac{1}{n}} + \tau^2\sqrt{\frac{1}{nm}}, \frac{\tau\varphi}{m}\sqrt{\frac{d\|\mathbf{C}\|_2}{n}} + \tau\sqrt{\frac{d\|\mathbf{C}\|_2}{nm}}\}$, and $\widetilde{O}(\cdot)$ hides the logarithmic factors on η, δ, m, n, d, and α.

as good as Gauss-Inverse asymptotically when $\varphi = \sqrt{d}$, and improve Gauss-Inverse by $\sqrt{d/m}$ times when $\varphi = 1$; improve UniSample-HD by a factor of 1 to $\sqrt{d/m}$ when $\varphi = \sqrt{d}$ and at least d/m if $\varphi = 1$, given a small m/d

Results

- Corollary 5.2. Given $\mathbf{X} \in \mathbb{R}^{d \times n}$ and an unknown population covariance matrix $\mathbf{C}_p \in \mathbb{R}^{d \times d}$ with each column vector $\mathbf{x}_i \in \mathbb{R}^d$ i.i.d. generated from the Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{C}_p)$. Let $\mathbf{C}_e$ be constructed by Theorem 5.1. Then, with the probability at least $1 - \eta - \delta - \zeta$,

$$\frac{\|\mathbf{C}_e - \mathbf{C}_p\|_2}{\|\mathbf{C}_p\|_2} \leq \widetilde{O}\left(\frac{d^2}{nm} + \frac{d}{m}\sqrt{\frac{d}{n}}\right);$$

statistical setting

Additionally, assuming rank$(\mathbf{C}_p) \leq r$, then with the probability at least $1 - \eta - \delta - \zeta$ we have

$$\frac{\|[\mathbf{C}_e]_r - \mathbf{C}_p\|_2}{\|\mathbf{C}_p\|_2} \leq \widetilde{O}\left(\frac{rd}{nm} + \frac{r}{m}\sqrt{\frac{d}{n}} + \sqrt{\frac{rd}{nm}}\right),$$

structural setting

where $[\mathbf{C}_e]_r$ is the solution to $\min_{\mathrm{rank}(A) \leq r} \|\mathbf{A} - \mathbf{C}_e\|_2$, and $\widetilde{O}(\cdot)$ hides the logarithmic factors on η, δ, ζ, m, n, d, and α.

Results

- Corollary 5.3 (Subspace). Given the notations in Corollary 5.2. Let $\prod_k = \sum_{i=1}^{k} \mathbf{u}_i\mathbf{u}_i^T$ and $\widehat{\prod}_k = \sum_{i=1}^{k} \hat{\mathbf{u}}_i\hat{\mathbf{u}}_i^T$ with $\{\mathbf{u}_i\}_{i=1}^k$ and $\{\hat{\mathbf{u}}_i\}_{i=1}^k$ being the leading k eigenvectors of C_p and C_e, respectively. Denote the k-th largest eigenvalue of C_p by λ_k. Then, with probability at least $1-\eta-\delta-\zeta$,

$$\frac{\|\widehat{\prod}_k - \prod_k\|_2}{\|\mathbf{C}_p\|_2} \leq \frac{1}{\lambda_k - \lambda_{k+1}} \widetilde{O}\Big(\frac{d^2}{nm} + \frac{d}{m}\sqrt{\frac{d}{n}}\Big),$$

where the eigengap $\lambda_k - \lambda_{k+1} > 0$ and $\widetilde{O}(\cdot)$ hides the logarithmic factors on η, δ ,ζ,m,n, d, and α.

Results

- Unbiased estimator $\mathbf{C}_e : \mathbb{E}[\mathbf{C}_e] = \mathbf{C}$ (Theorem 5.1)
- Upper bound $\|\mathbf{C} - \mathbf{C}_e\|_2$ (Theorem 5.2 & Corollary 5.1)
  - Outperform all related work
- Applicable to low-rank setting (Corollary 5.2)
  - Polynomially equal with the state-of-the-art methods that must use assumptions in algorithms design [Y. Chen, et al., 2013; T. Cai, et al., 2015]

Results

- Computational costs on the storage, communication, and time

Method	Storage	Communication	Time
Standard	$O(nd + d^2)$	$O(nd)$	$O(nd^2)$
Gauss-Inverse	$O(nm + d^2)$	$O(nm)$	$O(nmd + nm^2 d + nd^2) + T_G$
Sparse	$O(nm + d^2)$	$O(nm)$	$O(d + nm^2) + T_S$
UniSample-HD	$O(nm + d^2)$	$O(nm)$	$O(nd \log d + nm^2)$
Our method	$O(nm + d^2)$	$O(nm)$	$O(nd + nm \log d + nm^2)$

- $T_G \sim O(nmd)$, $T_S \sim O(nd^2)$
- Standard [W. Feller, 1966]

Results

- Computational costs on the storage, communication, and time

Method	Storage	Communication	Time
Standard	$O(nd+d^2)$	$O(nd)$	$O(nd^2)$
Gauss-Inverse	$O(nm+d^2)$	$O(nm)$	$O(nmd+nm^2d+nd^2)+T_G$
Sparse	$O(nm+d^2)$	$O(nm)$	$O(d+nm^2)+T_S$
UniSample-HD	$O(nm+d^2)$	$O(nm)$	$O(nd\log d+nm^2)$
Our method	$O(nm+d^2)$	$O(nm)$	$O(nd+nm\log d+nm^2)$

- $T_G \sim O(nmd)$, $T_S \sim O(nd^2)$
- Standard [W. Feller, 1966]

Experiments

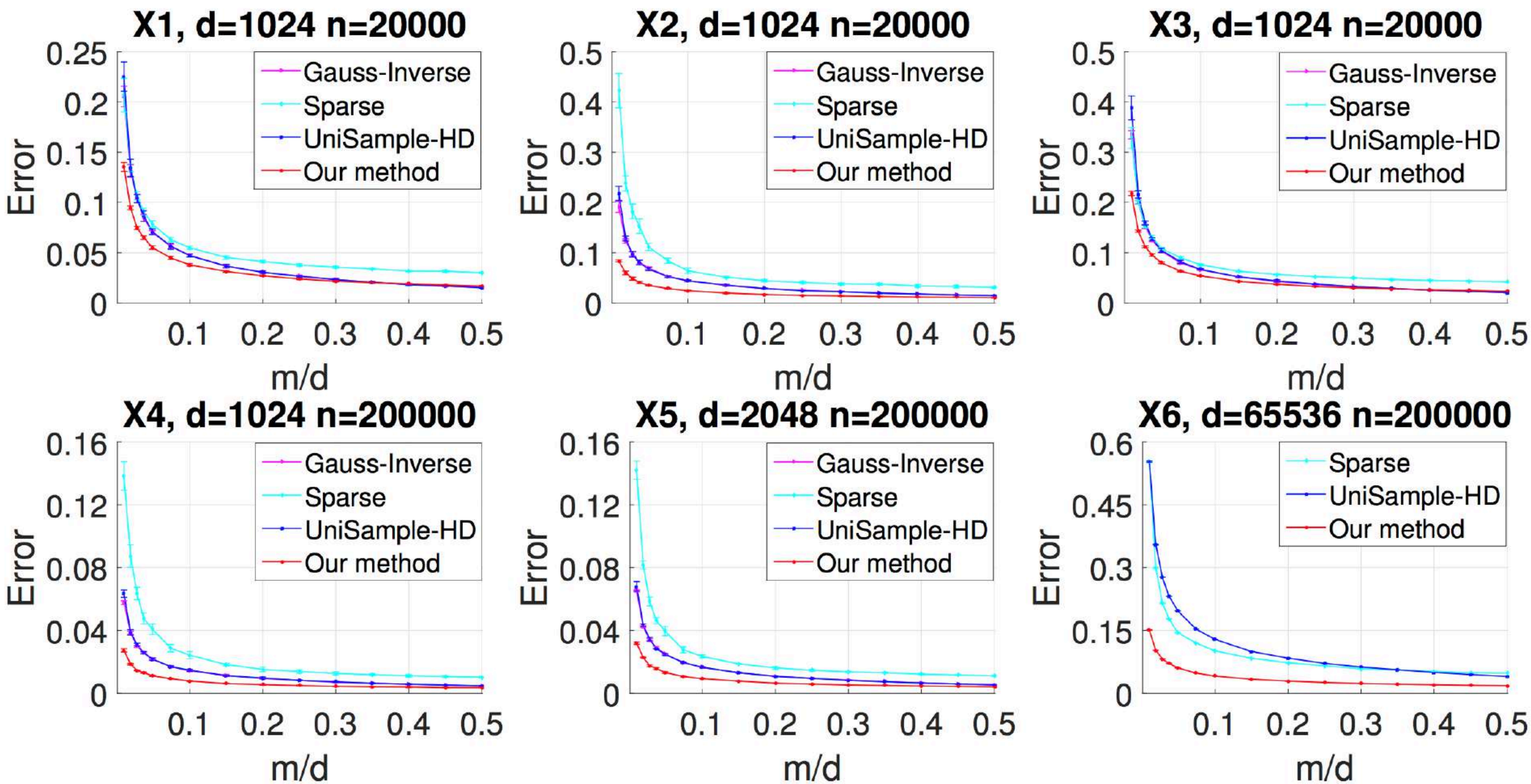
- Setting
  - $\alpha = 0.9$ in $p_{ki} = \alpha \frac{|x_{ki}|}{\|\mathbf{x}_i\|_1} + (1-\alpha)\frac{x_{ki}^2}{\|\mathbf{x}_i\|_2^2}$
- Compared methods
  - Gauss-Inverse [M. Azizyan, et al., 2015]
  - Sparse [F. Anaraki, et al., 2016]
  - UniSample-HD [F. Anaraki, et al., 2017]
  - Our method

Synthetic Data

- Probabilistic generative model: $\mathbf{X} = \mathbf{UFG} \in \mathbb{R}^{d \times n}$
  - $\mathbf{U} \in \mathbb{R}^{d \times k}$ with $\mathbf{U}^T\mathbf{U} = \mathbf{I}_k$ and $k \approx 0.005d$
  - $\mathbf{F} \in \mathbb{R}^{k \times k}$ with $f_{ii} = 1 - (i-1)/k$
  - $\mathbf{G} \in \mathbb{R}^{k \times n}$ with $g_{ij} \sim \mathcal{N}(0, 1)$
- Synthetic data: $\{\mathbf{X}_i\}_{i=1}^{3} \in \mathbb{R}^{1024 \times 20000}$, $\mathbf{X}_4 \in \mathbb{R}^{1024 \times 200000}$, $\mathbf{X}_5 \in \mathbb{R}^{2048 \times 200000}$ and $\mathbf{X}_6 \in \mathbb{R}^{65536 \times 200000}$
  - $\mathbf{X}_1 \sim \mathbf{X}$; $\mathbf{X}_3 \sim \mathbf{X}$ except that $\mathbf{F} = \mathbf{I}_k$
  - $\mathbf{X}_2 \sim \mathbf{DX}$ with $d_{ii} = 1/\beta_i$ and $\beta_i \sim [15]$; $\{\mathbf{X}_i\}_{i=4}^{6} \sim \mathbf{X}_2$

Covariance Estimation

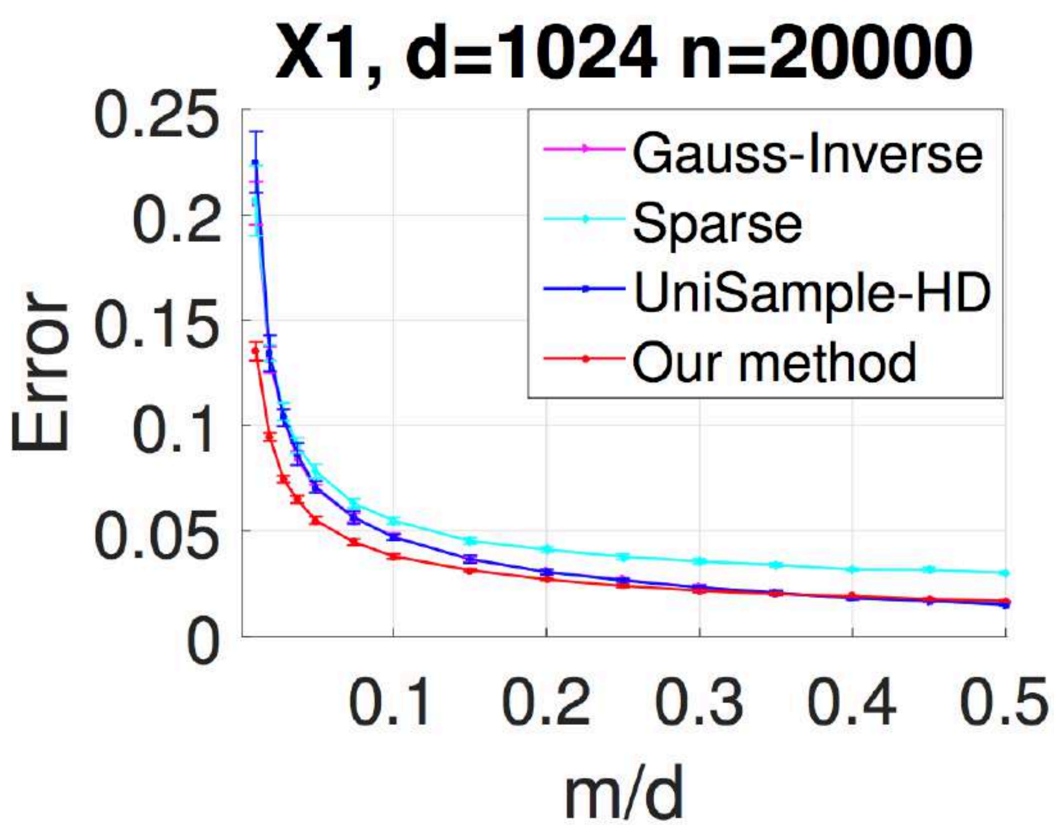
- Error: $\|\mathbf{C}_e - \mathbf{C}\|_2 / \|\mathbf{C}\|_2$

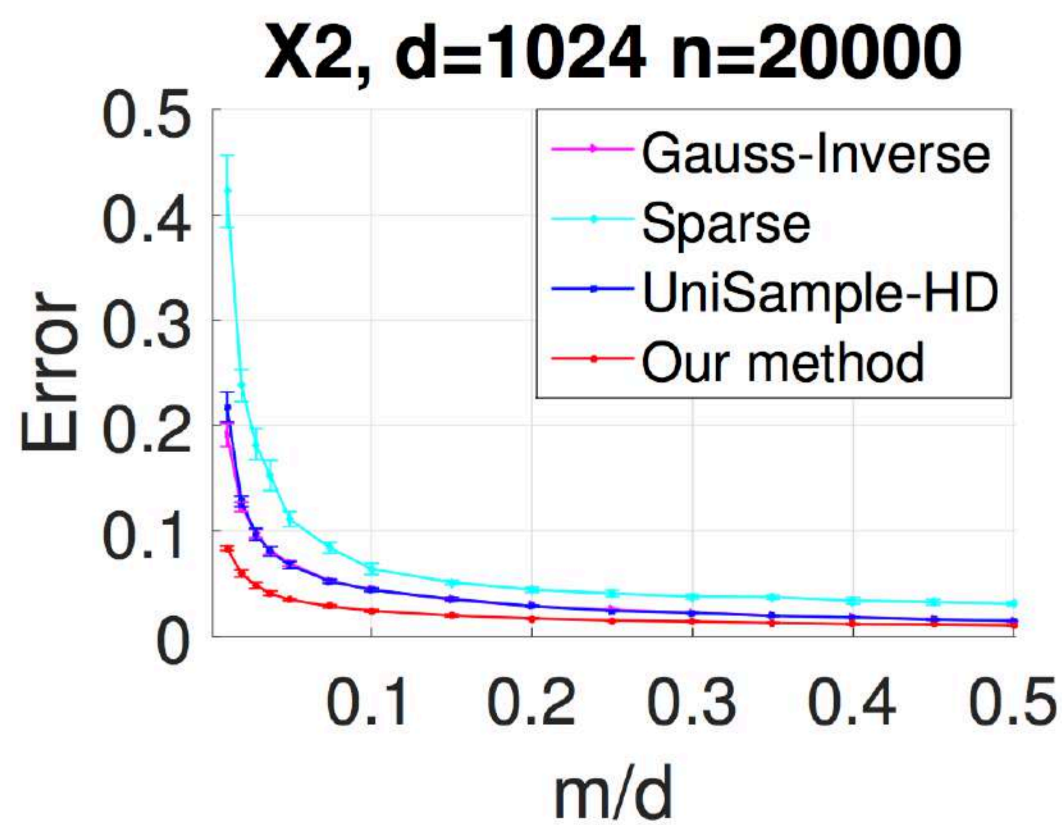


Covariance Estimation

- Error: $\|\mathbf{C}_e - \mathbf{C}\|_2 / \|\mathbf{C}\|_2$
- φ: $\|\mathbf{x}_i\|_1 / \|\mathbf{x}_i\|_2$
  - $\mathbf{X}_1$: $\varphi = 0.81\sqrt{d}$
  - $\mathbf{X}_2$: $\varphi = 0.55\sqrt{d}$

φ ↘Error ↘only for our method

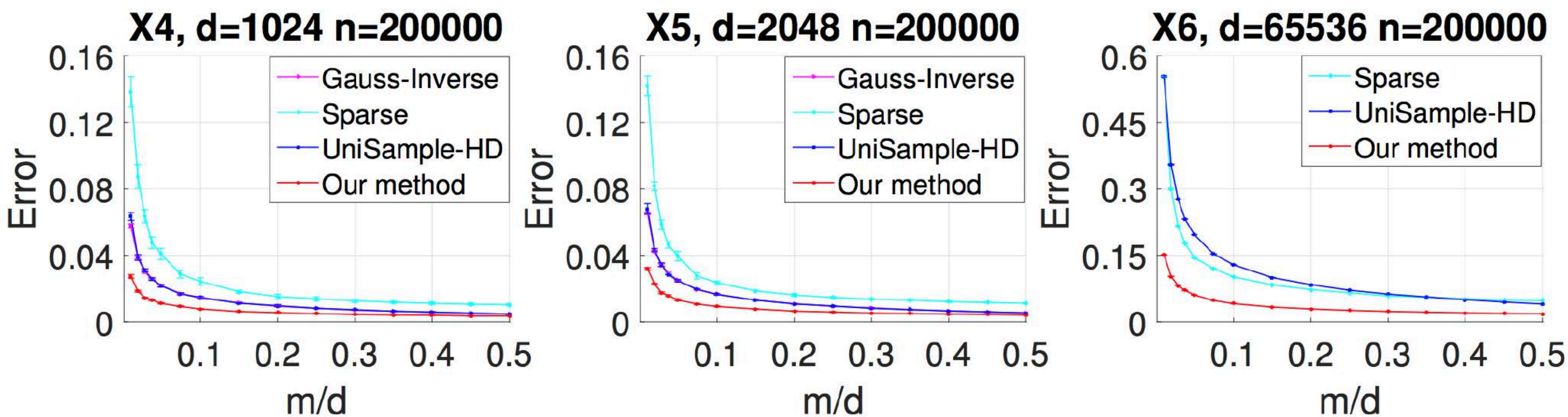




Covariance Estimation

- Error: $\|\mathbf{C}_e - \mathbf{C}\|_2 / \|\mathbf{C}\|_2$
  - $\mathbf{X}_4$: $d = 1024$
  - $\mathbf{X}_5$: $d = 2048$
  - $\mathbf{X}_6$: $d = 65536$

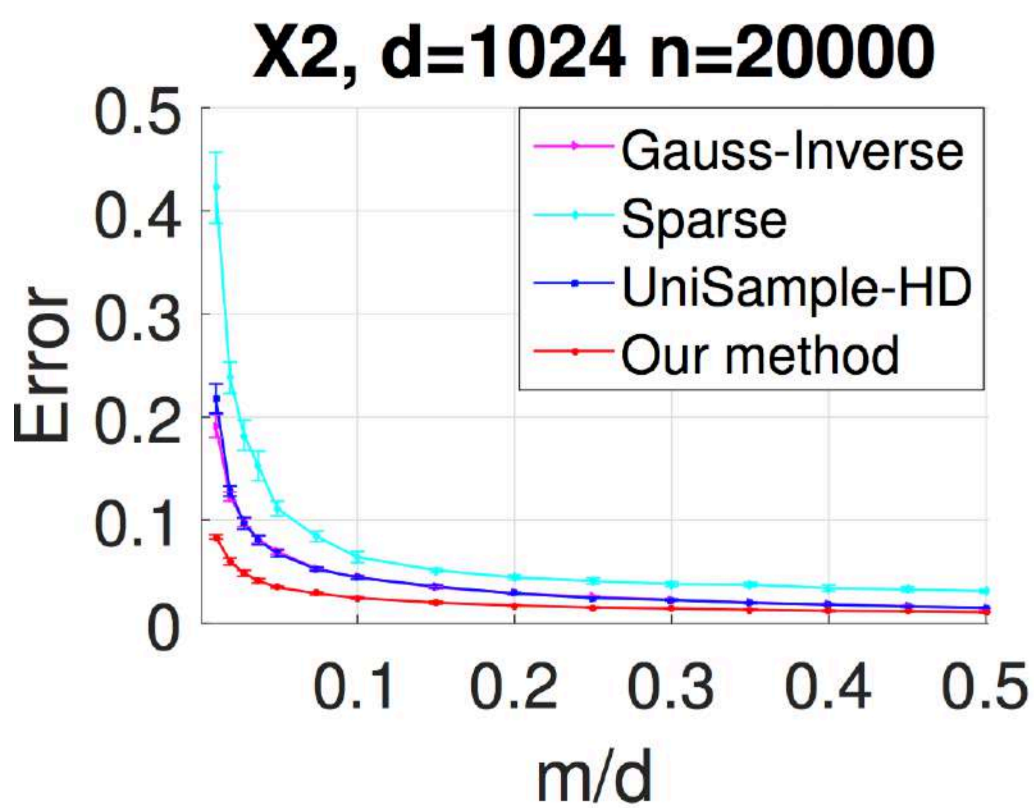
best for all d

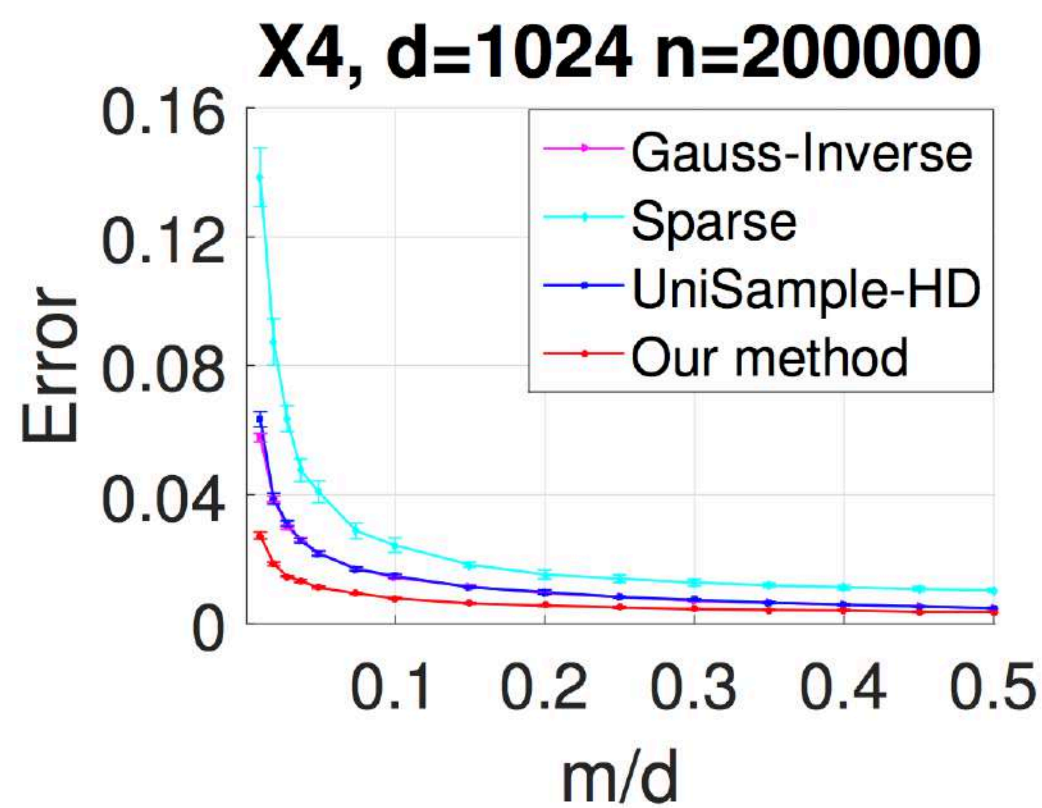


Covariance Estimation

- Error: $\|\mathbf{C}_e - \mathbf{C}\|_2 / \|\mathbf{C}\|_2$
  - $\mathbf{X}_2$: $n = 20000$
  - $\mathbf{X}_4$: $n = 200000$

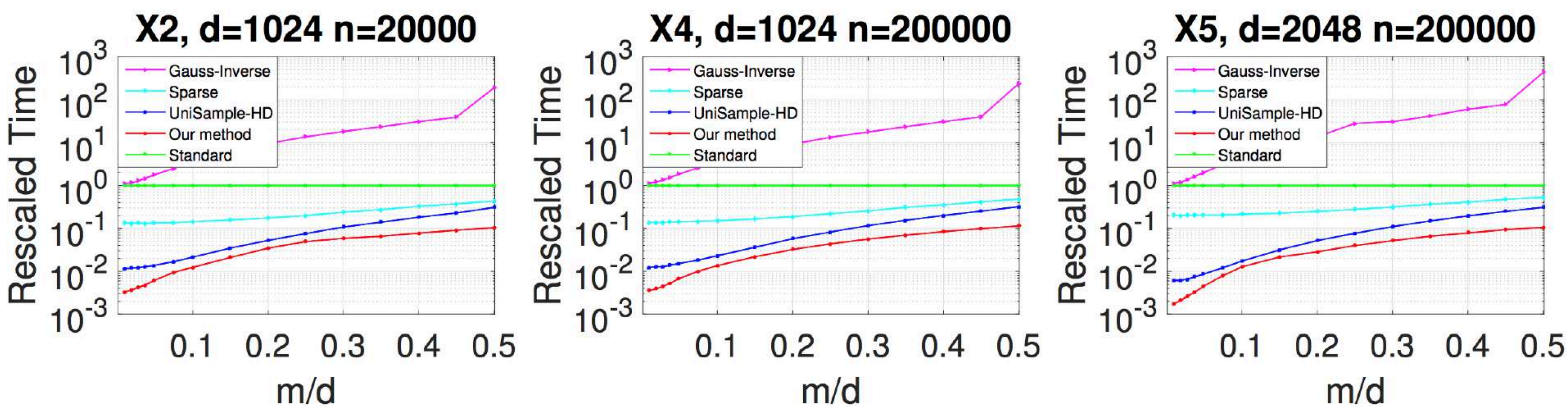
$n \nearrow$ Error $\searrow$





Covariance Estimation

- Time comparison

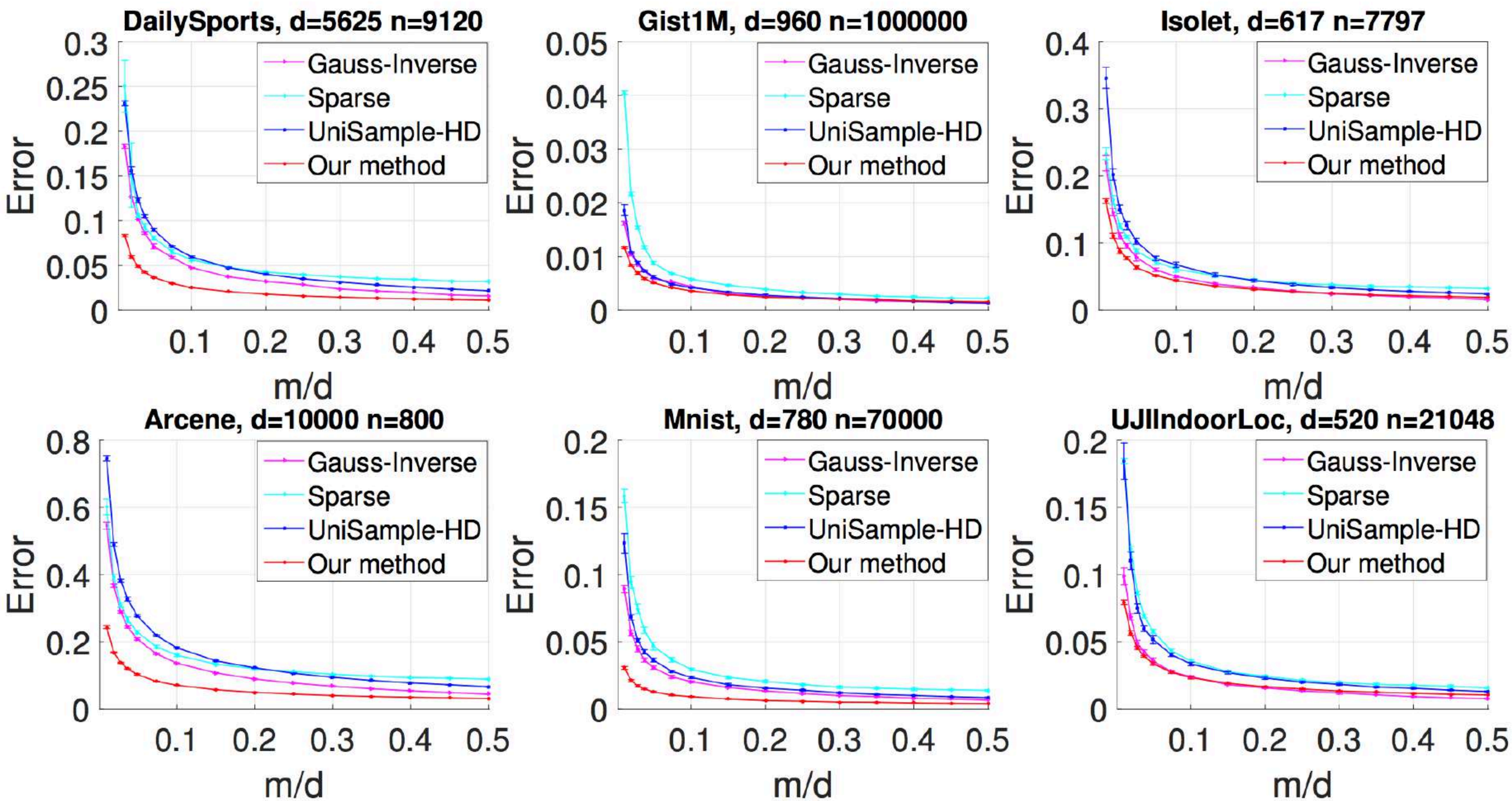


Real Data

Dataset	Size	Dimension
DailySports	9,120	5,625
Gist1M	1,000,000	960
Isolet	7,797	617
Arcene	800	10,000
Mnist	70,000	780
UJIIndoorLoc	21,048	520

Covariance Estimation

- Error: $\|\mathbf{C}_e - \mathbf{C}\|_2 / \|\mathbf{C}\|_2$

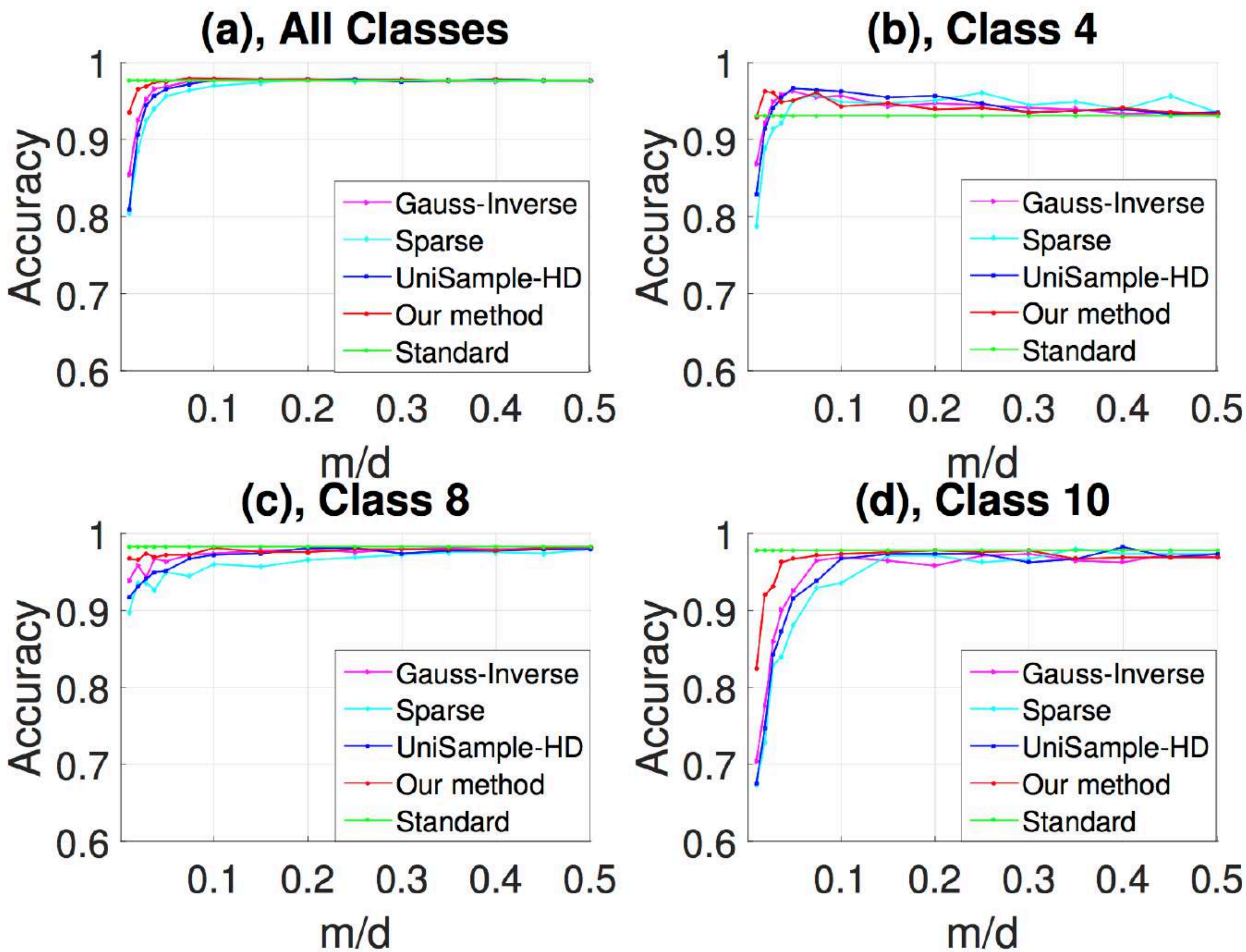


Multiclass Classification

- MNIST data - 10 classes
  - Center data for each individual class
- Classifier
  - Get $\{\mathbf{C}_t\}_{t=1}^{10}$ by different estimation methods
  - Compute $\prod_{k,t} = \sum_{j=1}^{k} \mathbf{u}_{j,t}\mathbf{u}_{j,t}^T$ from $\{\mathbf{C}_t\}_{t=1}^{10}$
  - Find a solution to $\max_t \mathbf{x}^T \prod_{k,t} \mathbf{x}$ for all $t \in [10]$

Multiclass Classification

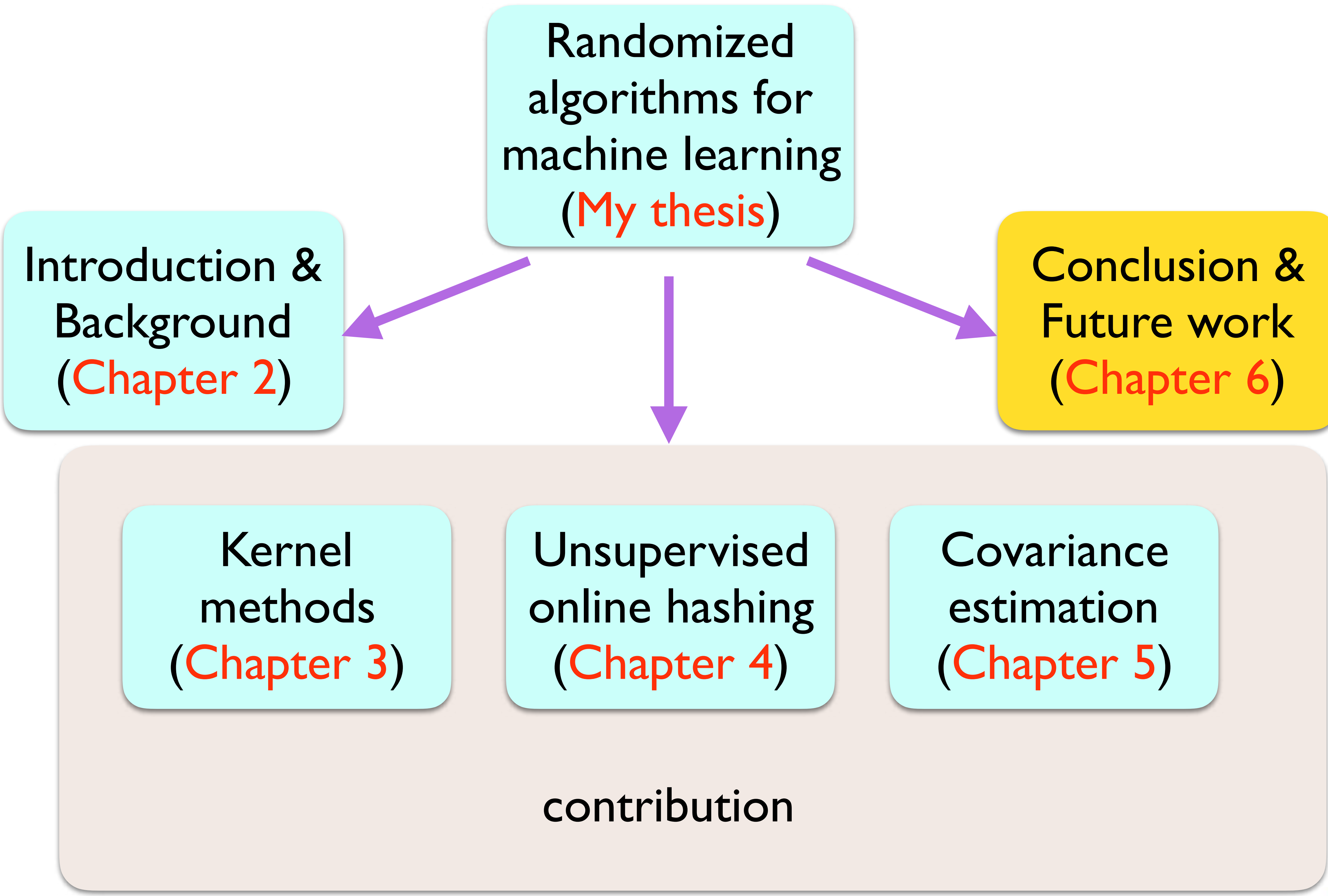
- Accuracy comparison - MNIST data



Conclusion

- Improve both the accuracy and efficiency of covariance matrix estimation on compressed data
- Demonstrate the good performance by provable results, complexity analysis, and extensive experiments

Outline


Randomized algorithms for machine learning (My thesis)
Introduction & Background (Chapter 2)
Conclusion & Future work (Chapter 6)
Kernel methods (Chapter 3)
Unsupervised online hashing (Chapter 4)
Covariance estimation (Chapter 5)
contribution

Comparisons on Chapters

	$\mathbf{X}^T\mathbf{X}$ $\mathbf{X} \in \mathbb{R}^{d \times n}$
projection; compress n	Kernel methods (Chapter 3)
projection; compress n	Unsupervised online hashing (Chapter 4)
sampling; compress d	Covariance estimation (Chapter 5)

Comparisons on Chapters

$\mathbf{X}^T\mathbf{X}$ $\mathbf{X} \in \mathbb{R}^{d \times n}$		apply to	Chapter 3	Chapter 4	Chapter 5
projection; compress n	Kernel methods (Chapter 3)	→		😭optimal accuracy; streaming setting	
projection; compress n	Unsupervised online hashing (Chapter 4)	→	😃		
sampling; compress d	Covariance estimation (Chapter 5)	→			

Comparisons on Chapters

$\mathbf{X}^T\mathbf{X}$	$\mathbf{X} \in \mathbb{R}^{d \times n}$	apply to	Chapter 3	Chapter 4	Chapter 5
projection; compress n	Kernel methods (Chapter 3)	→			😭consistent estimation; streaming setting
projection; compress n	Unsupervised online hashing (Chapter 4)	→			
sampling; compress d	Covariance estimation (Chapter 5)	→	😃		

Comparisons on Chapters

$\mathbf{X}^T\mathbf{X}$	$\mathbf{X} \in \mathbb{R}^{d \times n}$	apply to	Chapter 3	Chapter 4	Chapter 5
projection; compress n	Kernel methods (Chapter 3)	→			
projection; compress n	Unsupervised online hashing (Chapter 4)	→			😃UniSample-HD [F. Anaraki, 2017] with compressing d
sampling; compress d	Covariance estimation (Chapter 5)	→		😭optimal accuracy	

Comparisons on Chapters

$\mathbf{X}^T\mathbf{X}$ $\mathbf{X} \in \mathbb{R}^{d \times n}$		apply to	Chapter 3	Chapter 4	Chapter 5
projection; compress n	Kernel methods (Chapter 3)	→		😭optimal accuracy; streaming setting	😭consistent estimation; streaming setting
projection; compress n	Unsupervised online hashing (Chapter 4)	→	😃		😃UniSample-HD [F. Anaraki, 2017] with compressing d
sampling; compress d	Covariance estimation (Chapter 5)	→	😃	😭optimal accuracy	

Way Forward

- $\mathrm{X}^T\mathrm{X}$ involved:
  - "Approximate Newton Methods and Their Local Convergence" [H. Ye, et al., ICML 2017]

Algorithm 1 Sketch Newton.

Input: $x^{(0)}, 0 < \delta < 1, 0 < \epsilon_0 < 1$;
for $t = 0, 1, \ldots$ until termination **do**
  Construct an ϵ_0-subspace embedding matrix S for $B(x^{(t)})$ and where $\nabla^2 F(x)$ is of the form $\nabla^2 F(x) = (B(x^{(t)}))^T B(x^{(t)})$, and calculate $H^{(t)} = [B(x^{(t)})]^T S^T S B(x^{(t)})$;
  Calculate $p^{(t)} \approx \mathrm{argmin}_p \frac{1}{2} p^T H^{(t)} p - p^T \nabla F(x^{(t)})$;
  Update $x^{(t+1)} = x^{(t)} - p^{(t)}$;
end for

  - Other 4 similar randomized algorithm papers in [J. Tang, et al., ICML 2017; S. Wang, et al., ICML 2017; D. Calandriello, et al., ICML 2017; D. Calandriello, et al., NIPS 2017]
  - Chapter 5 can improve their accuracy, but how to prove?

Way Forward

- Randomized algorithms and implicit regularization
- Randomized algorithms for deep neural networks
- Randomized algorithms for parallel/distributed computation

Publications

- Conference
  - **Xixian Chen**, Michael R. Lyu, Irwin King. Toward Efficient and Accurate Covariance Matrix Estimation on Compressed Data. In Proceedings of the 34th International Conference on Machine Learning (ICML 2017).
  - **Xixian Chen**, Irwin King, Michael R. Lyu. FROSH: FasteR Online Sketching Hashing. In Proceedings of the 33rd International Conference on Uncertainty in Artificial Intelligence (UAI 2017).
  - **Xixian Chen**, Haiqin Yang, Irwin King, and Michael R. Lyu. Training-Efficient Feature Map for Shift-Invariant Kernels. In Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015).

Publications

- Conference
  - Shenglin Zhao, **Xixian Chen**, Michael R. Lyu, Irwin King. Personalized Sequential Check-In Prediction: Beyond Geographical and Temporal Contexts. Submitted to International Conference on Multimedia and Expo (ICME 2018).
- Journal
  - **Xixian Chen**, Haiqin Yang, Irwin King, Michael R. Lyu. Faster Online Sketching Hashing. Submitted to IEEE Transactions on Neural Networks and Learning Systems (TNNLS).
  - **Xixian Chen**, Haiqin Yang, Michael R. Lyu, Irwin King. Estimation of Covariance Matrix on Compressed Data. Submitted to IEEE Transactions on Neural Networks and Learning Systems (TNNLS).

Thanks!

Q&A