Algorithmic complexity and practical performance of Top-k compression for communication efficient distributed optimization

Igor Sokolov, Maksim Makarenko, Rayan Rozi, Sarah Mutahar, Yasir Ghunaim, Zuhair Hafiz

CS 260: Design and Analysis of Algorithms
December 5, 2021

Team

Igor Sokolov
AMCS MS student

Maksim Makarenko
ECE PhD student

Rayan Rozi
CS MS student

Sarah Mutahar
CS MS student

Yasir Ghunaim
CS MS student

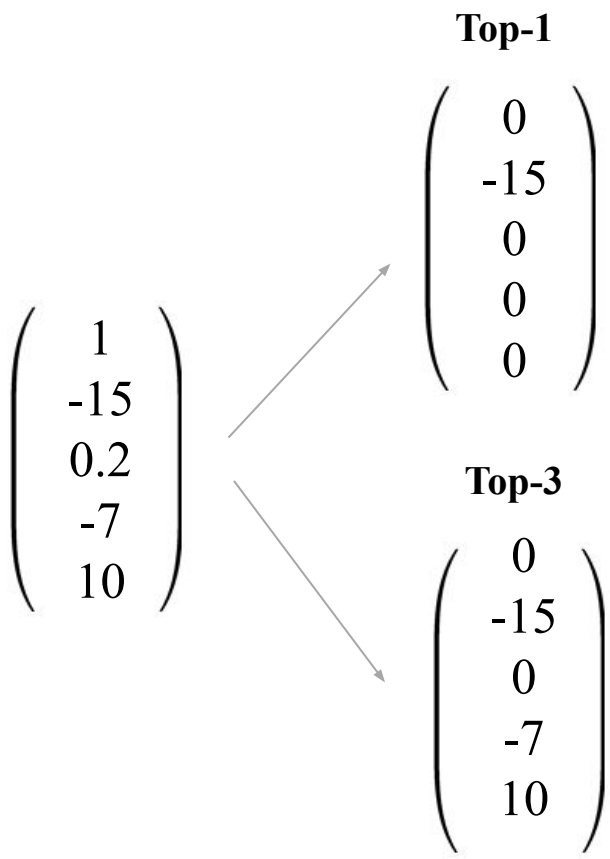
Zuhair Hafiz
CS MS student

Goal of the project

Top-k as a Compressor

$$\begin{pmatrix} 1 \\ -15 \\ 0.2 \\ -7 \\ 10 \end{pmatrix}$$

Top-k as a Compressor



Top-k as a Compressor

$$\begin{pmatrix} X_1 \\ X_2 \\ . \\ . \\ . \\ X_d \end{pmatrix}$$

Assuming that d is large ($d \approx 10^5$)

Top-k as a Compressor

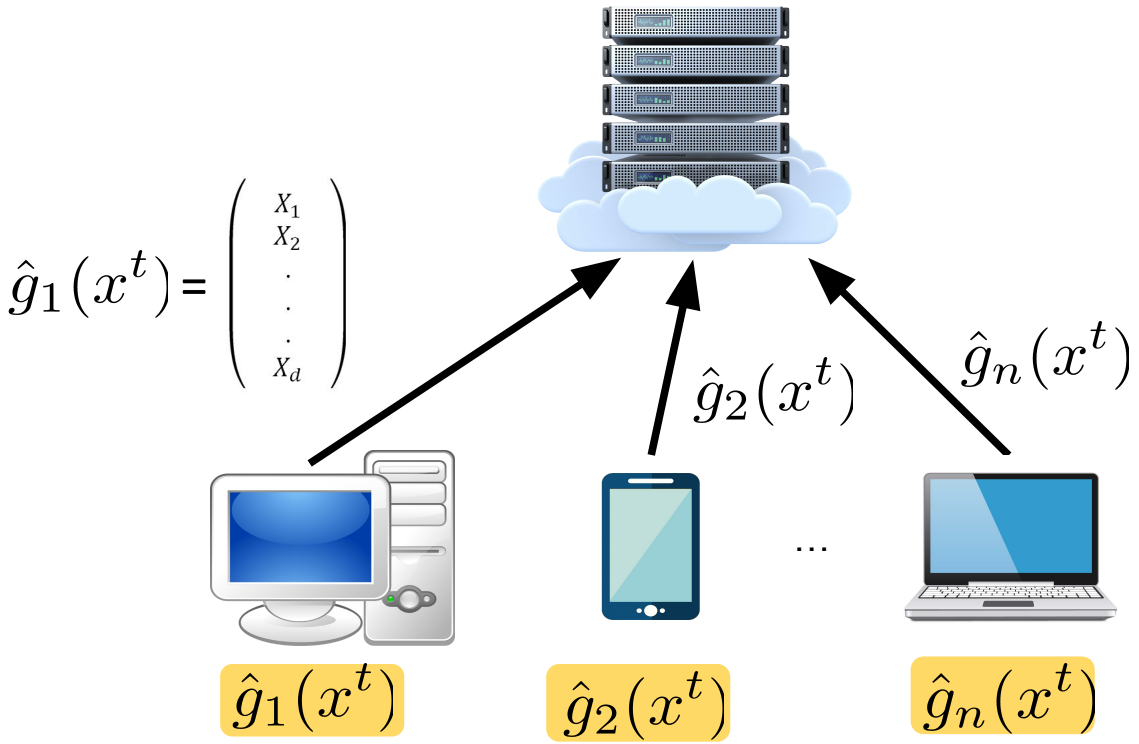
$$\begin{pmatrix} X_1 \\ X_2 \\ . \\ . \\ . \\ X_d \end{pmatrix}$$

Assuming that d is large ($d \approx 10^5$)

Goal:
Measure the time complexity of Top-k compression for large d

Why is Top-k important?

DSGD: Distributed Stochastic Gradient Descent



DSGD: Distributed Stochastic Gradient Descent

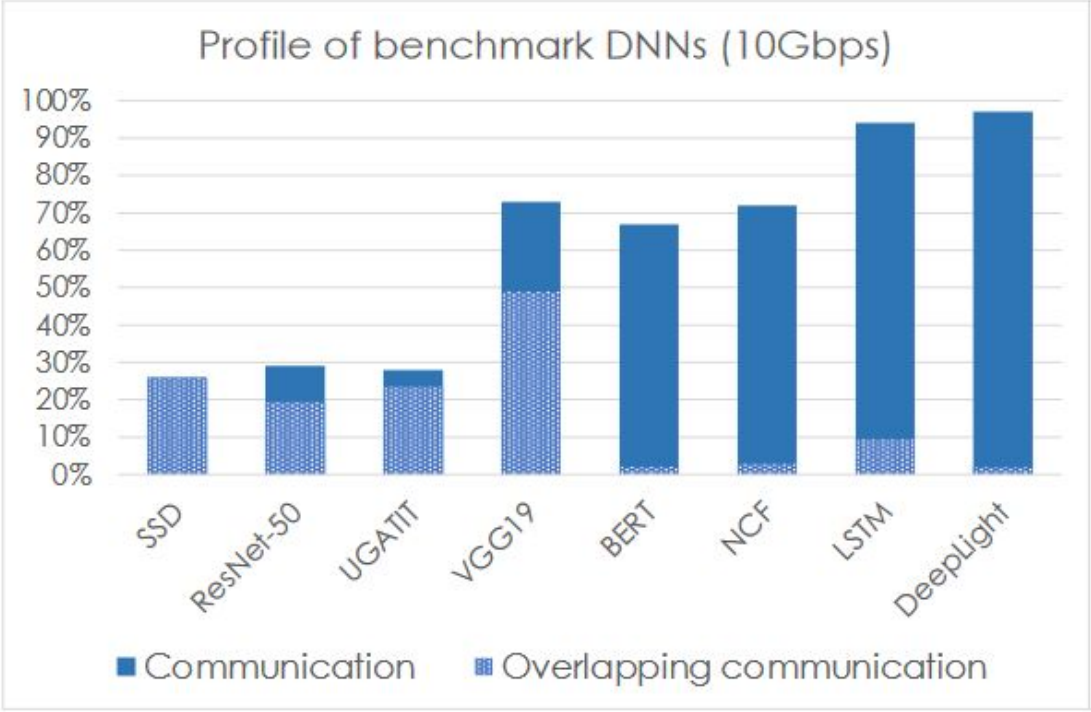
Machine Learning models are increasing in size:

Architecture	Year	Top-1 Accuracy	# Parameters
DenseNet-169	2017	76.2%	14M
Inception-v3	2016	78.8%	24M
Inception-resnet-v2	2017	80.1%	56M
PolyNet	2017	81.3%	92M
SENet	2018	82.7%	146M
GPipe	2018	84.3%	557M
ResNeXt-101 32x48d	2019	85.4%	829M

Source: Efficient inference in Deep Learning - where is the problem? Deci. (2021, November 22). Retrieved December 4, 2021, from https://deci.ai/resources/blog/problem-efficient-inference-deep-learning/.

DSGD: Distributed Stochastic Gradient Descent

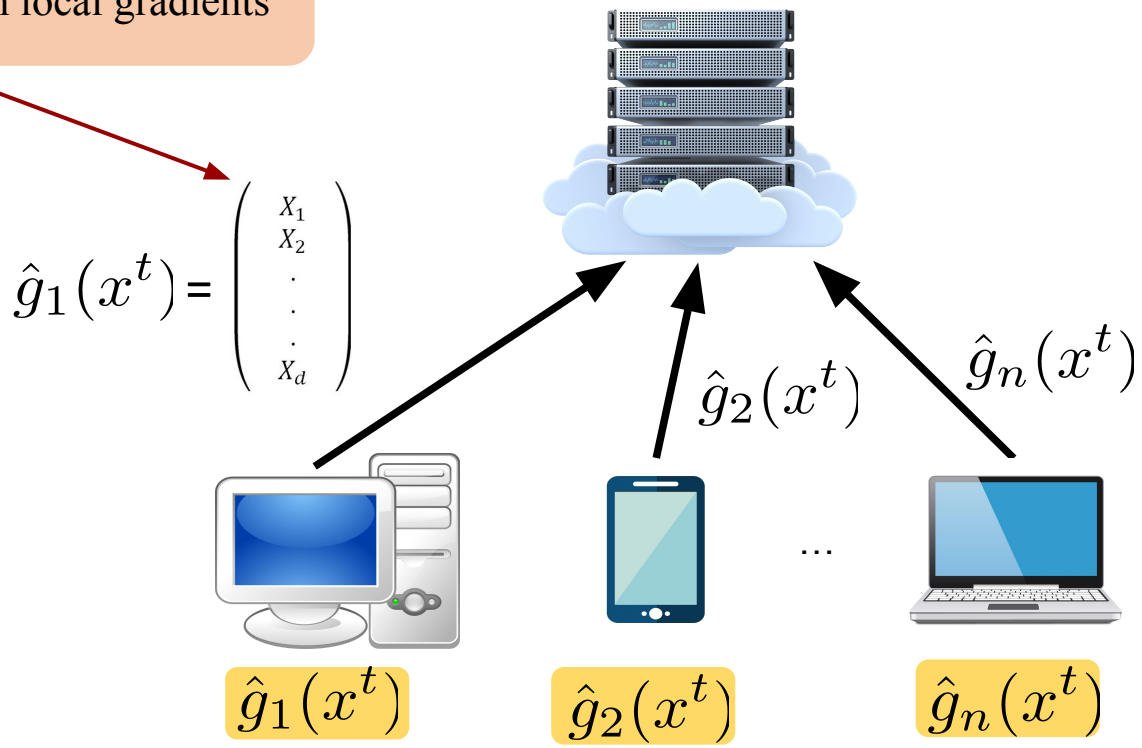
Benchmark on datacenter with 10 Gbps network speed



Source: DeepReduce A Sparse-tensor Communication Framework for Federated Deep Learning Deci. (2021, November 29). Retrieved December 4, 2021, from KAUST CS Graduate Seminar, Professor Panos Kalnis

DSGD: Distributed Stochastic Gradient Descent

Objective:
Apply Top-k compression on local gradients



Algorithms that performs Top-k compression

Algorithms

1. Tournament Method — Worst case: $O(d + k \log(d))$

2. Quickselect
   a. Deterministic Pivot — Worst case: $O(d^2)$ Average: $O(d)$
   b. Random Pivot — Worst case: $O(d^2)$ Average: $O(d)$
   c. Median of medians — Worst case: $O(d)$ Average: $O(d)$
      but worse than other pivots

Algorithms Modifications

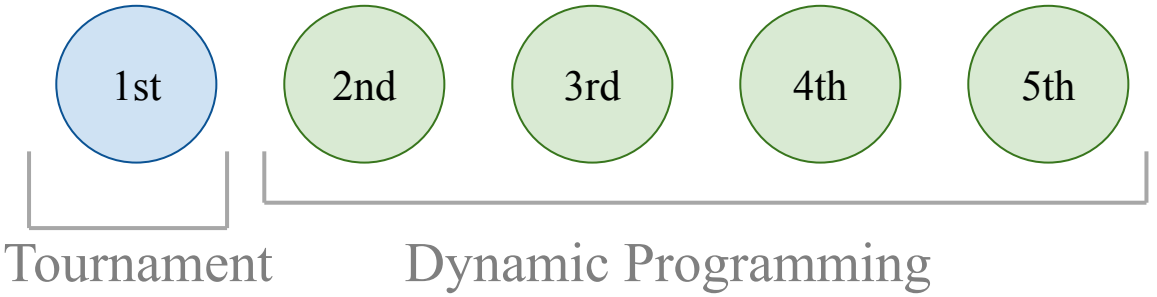
We are interested in finding all elements larger than or equal to kth element.
Not only kth largest element!

Algorithms Modifications

We are interested in finding all elements larger than or equal to kth element.
Not only kth largest element!

Find the Top-5 elements:

Tournament Method + DP

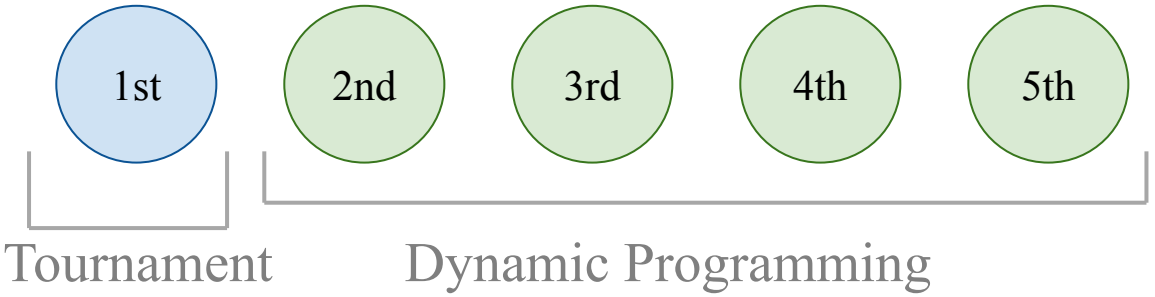


Algorithms Modifications

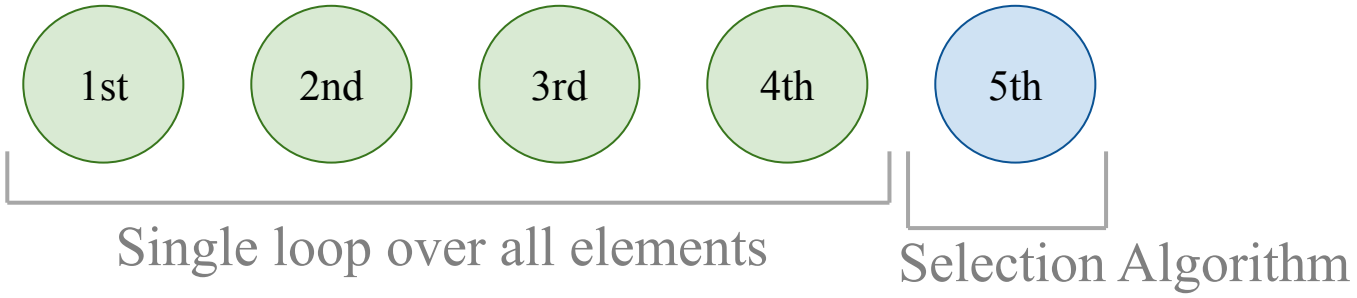
We are interested in finding all elements larger than or equal to kth element.
Not only kth largest element!

Find the Top-5 elements:

Tournament Method + DP



Better Approach



Implementation, Experiments

Experiments

1. **Synthetic experiments: inputs are sampled from known priors**

Experiments

1. **Synthetic experiments: inputs are sampled from known priors**
2. **Real experiments: inputs are taken from ML model training procedure**

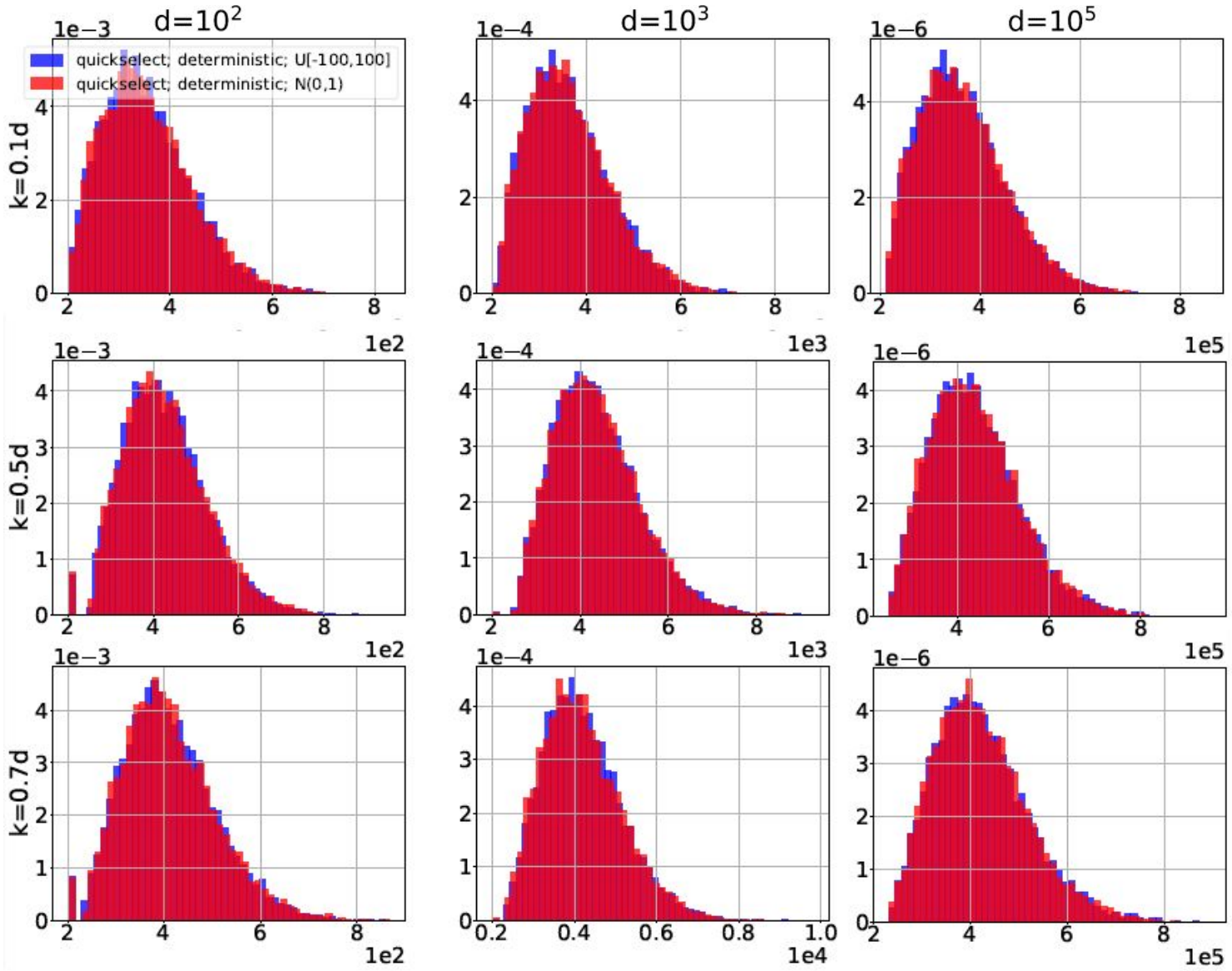
Synthetic Experiments

Synthetic experiments

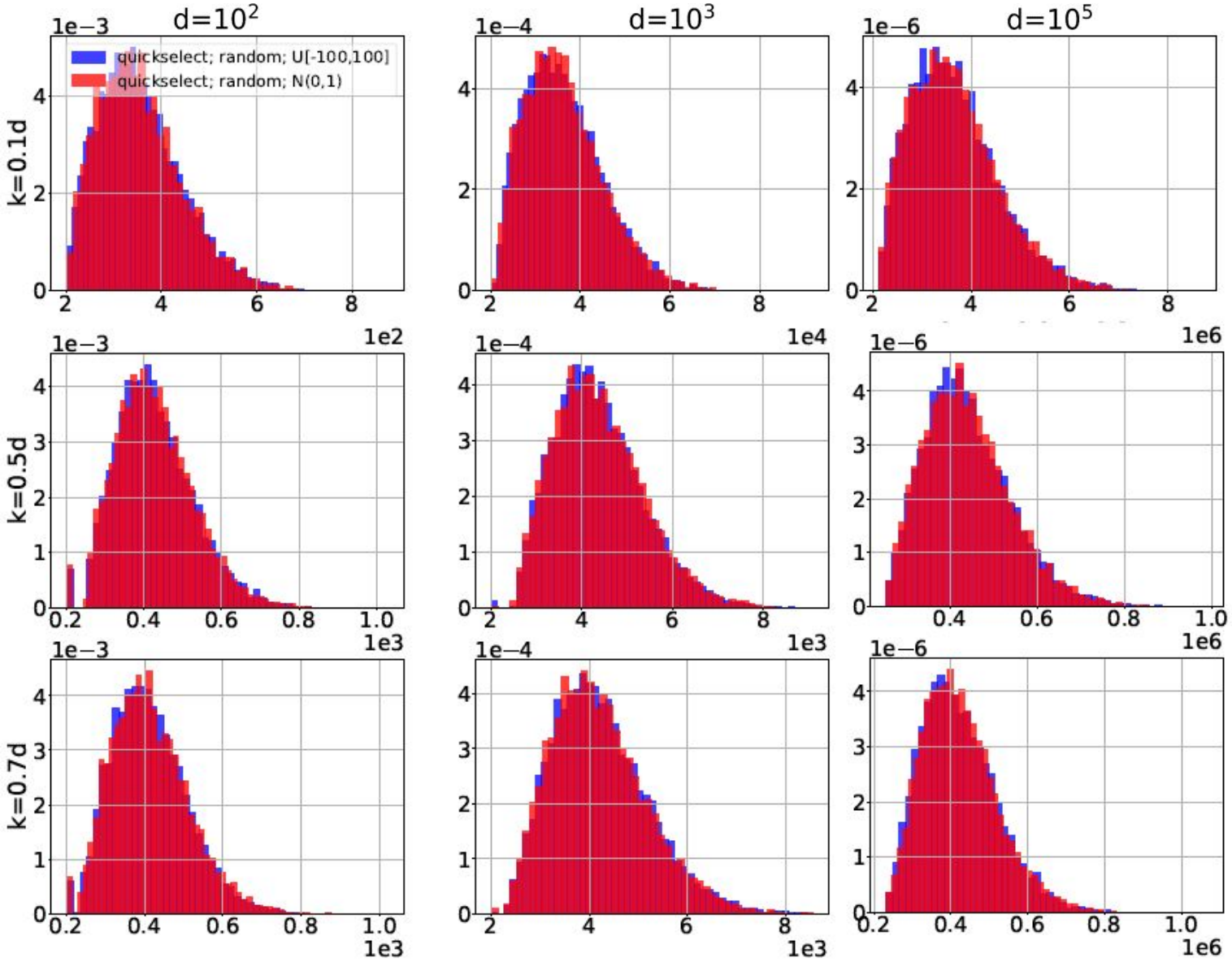
Synthetic data

- Fix some values of d from 100 to 100000 and for each d fix k from 0.01d to 0.9d
- Sample ~10000 vectors from known prior and compute time complexity for each vector
- Plot the empirical distribution of the time complexity
- Compare medians of the empirical distributions

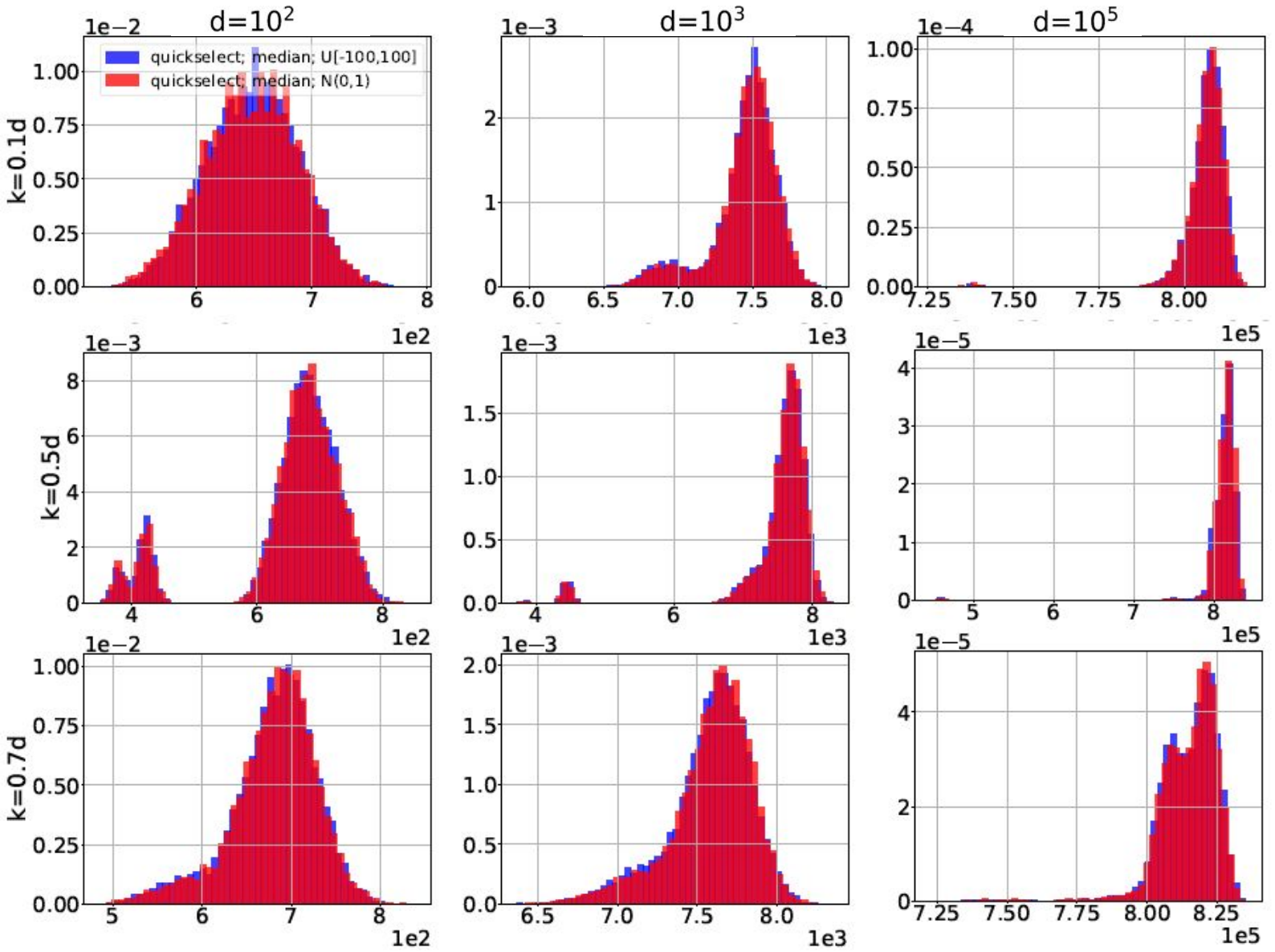
Prior Impact (Deterministic Quickselect)



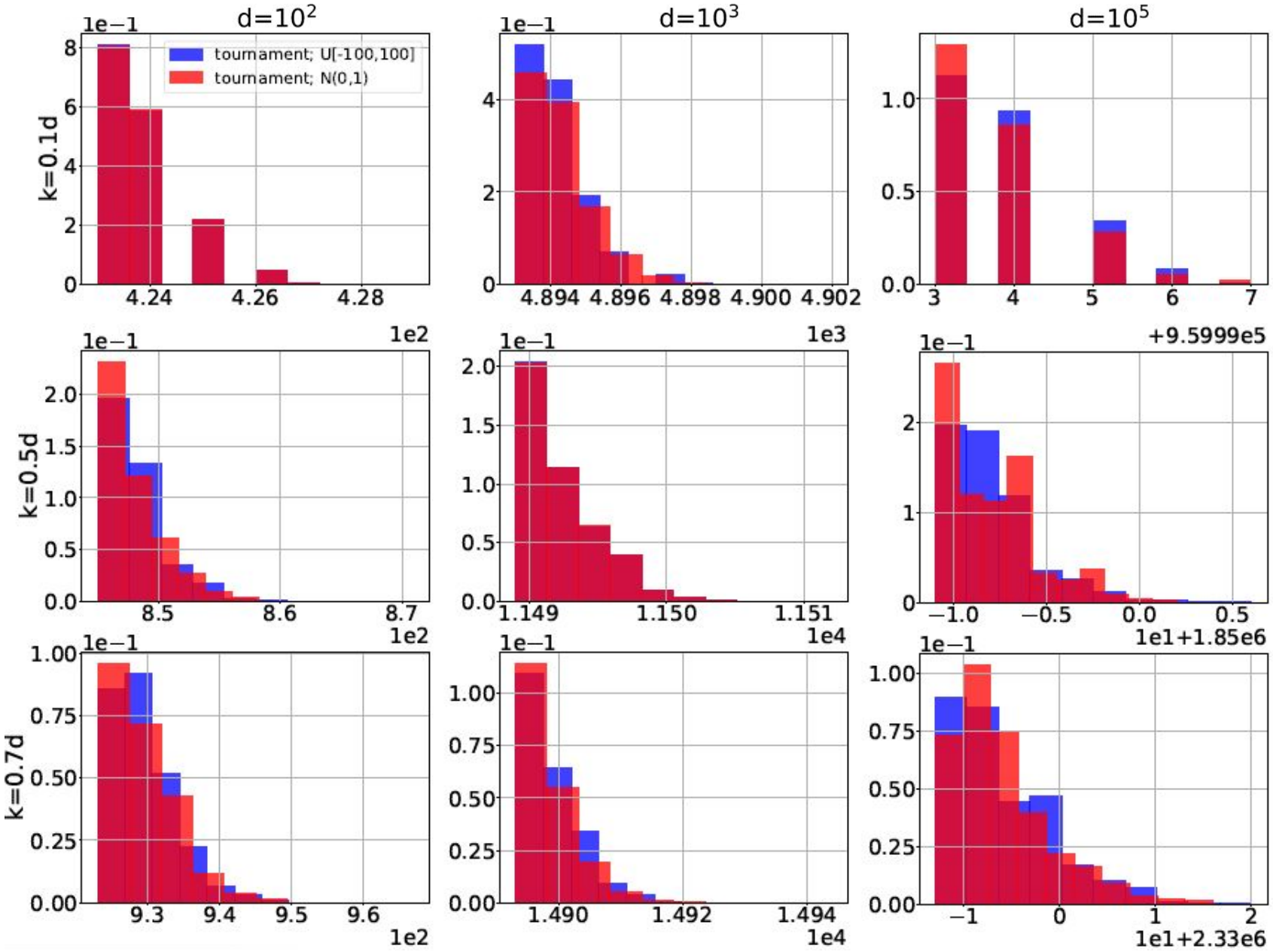
Prior Impact (Random Quickselect)



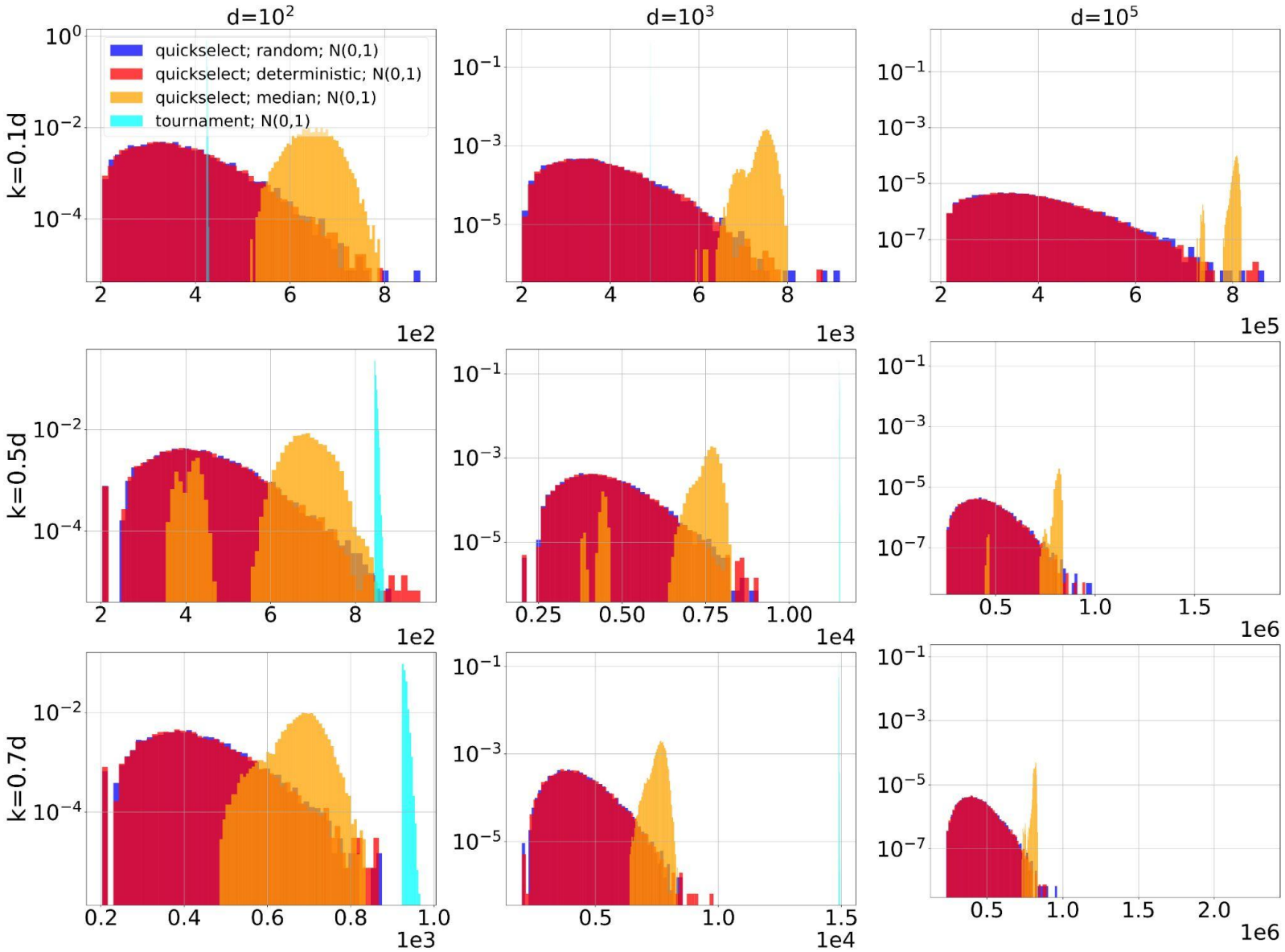
Prior Impact (Median Quickselect)



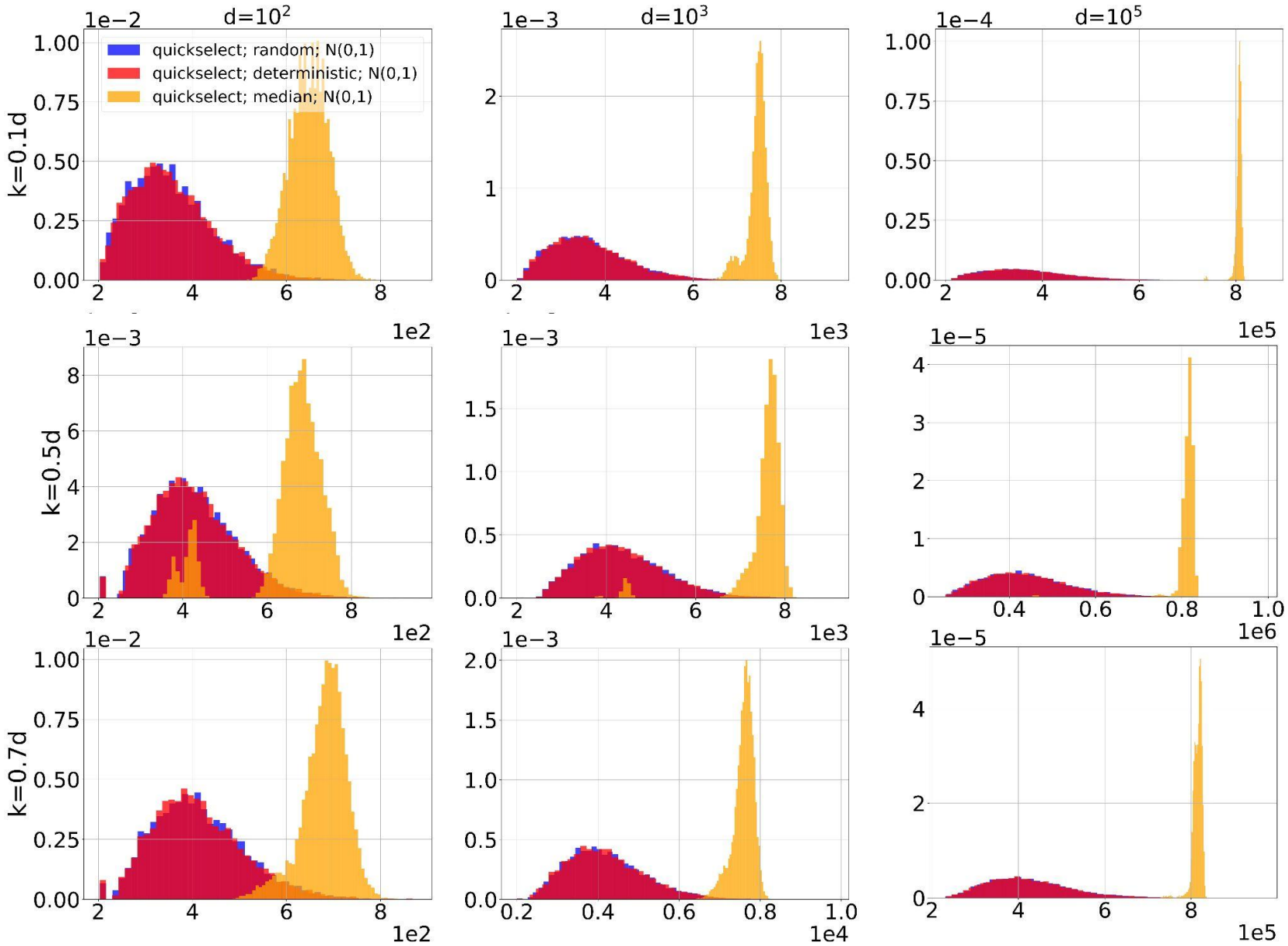
Prior Impact (Tournament)



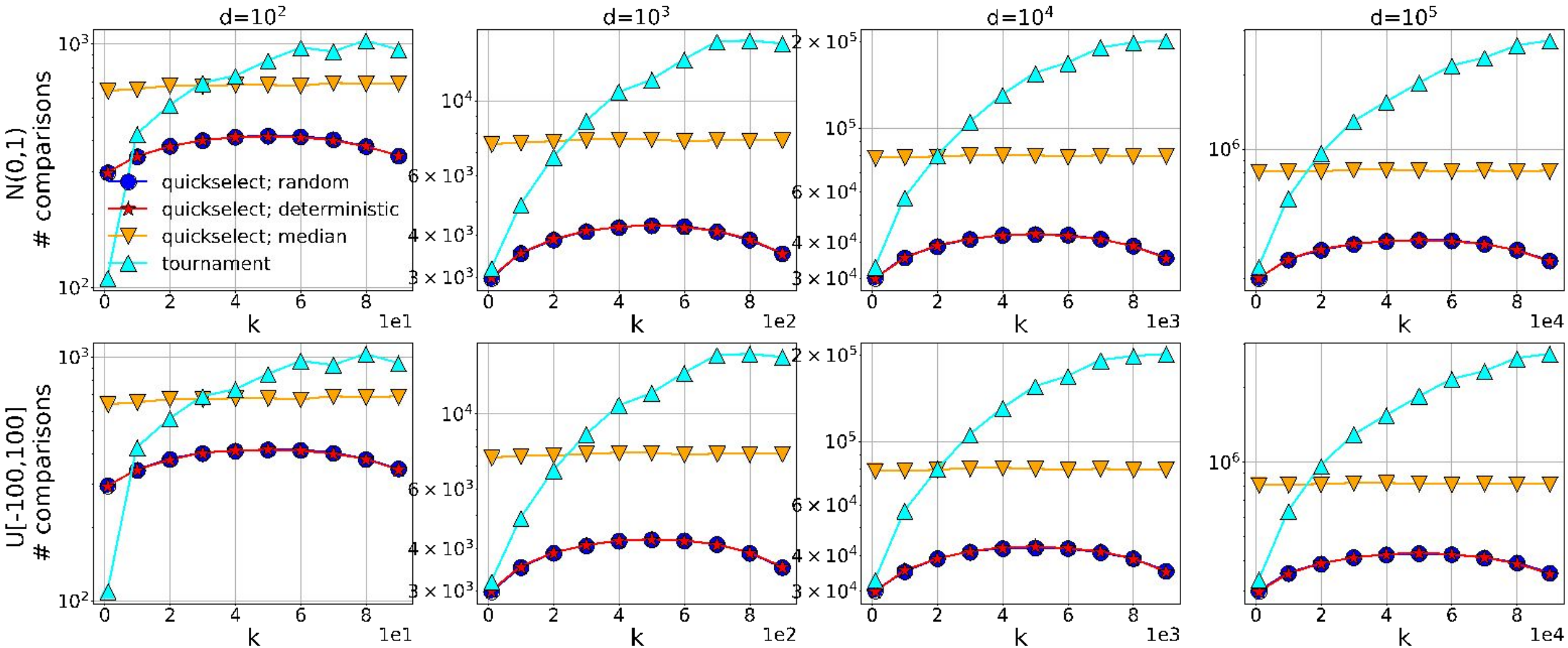
Number of comparison histograms (log scale)



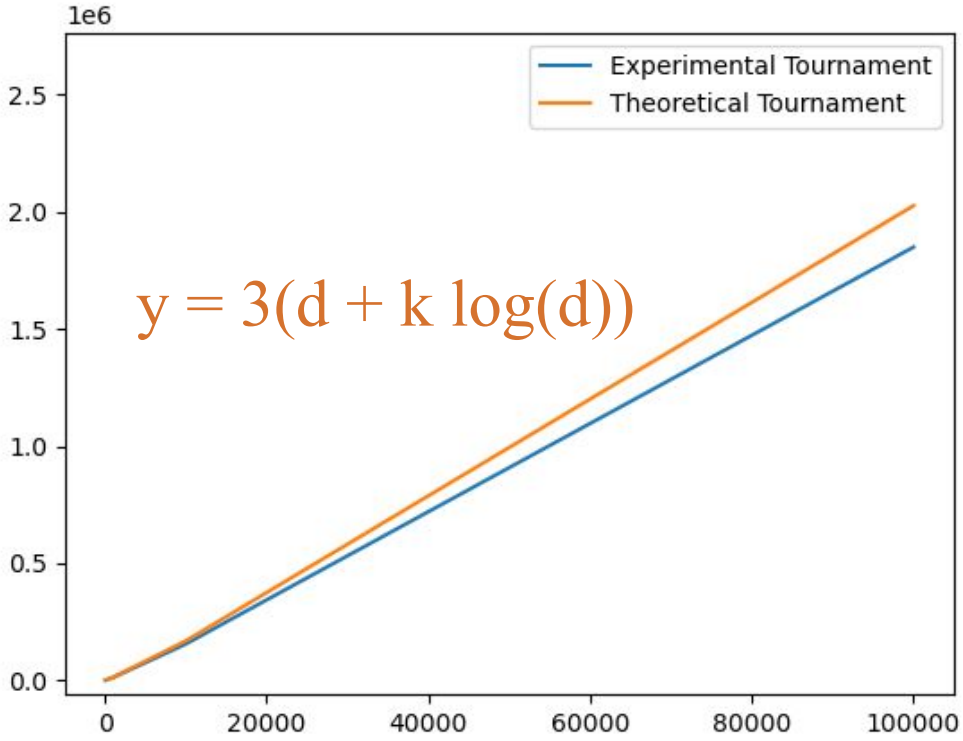
Number of comparison histograms

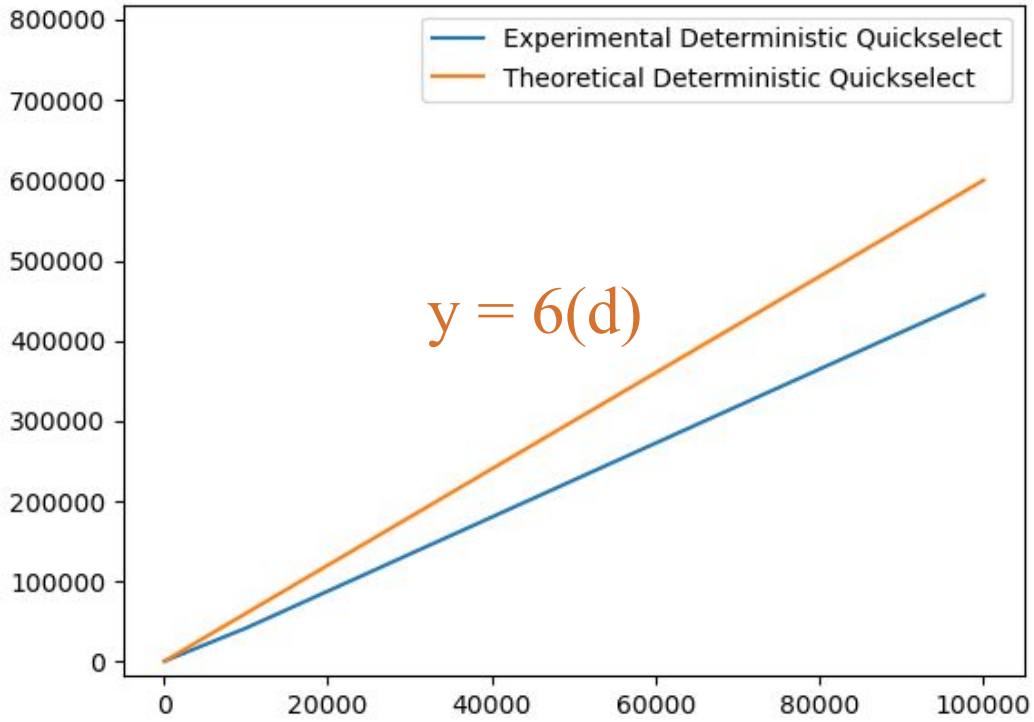


Number of comparisons over k values

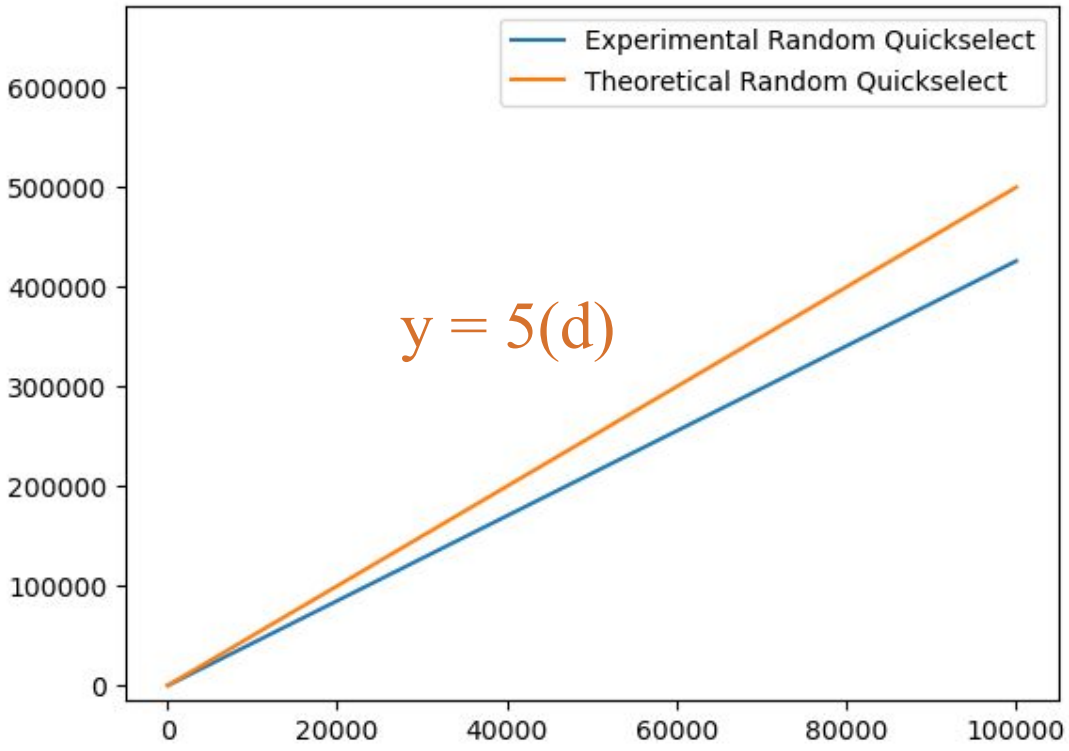


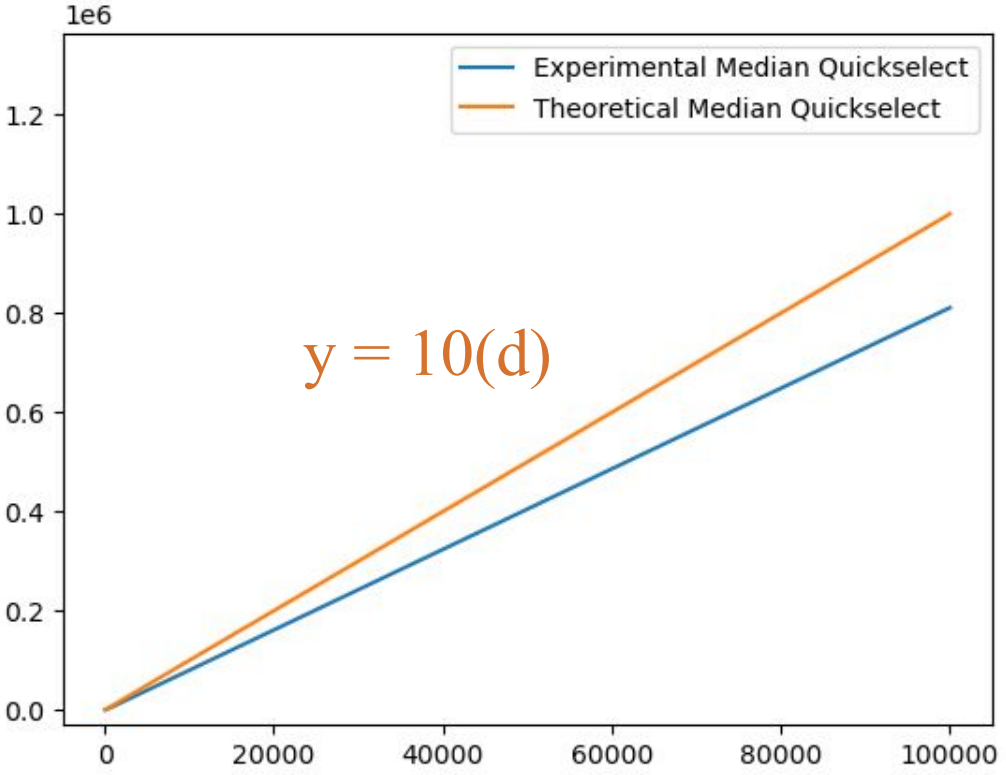
Theoretical vs Experimental Time Complexity





Theoretical vs Experimental Time Complexity





Real Experiments

Real experiments

$$\min_{x\in\mathbb{R}^d}\left\{f(x)=\frac{1}{N}\sum_{i=1}^{N}\log\left(1+\exp\left(-b_i a_i^\top x\right)\right)+\lambda\sum_{j=1}^{d}\frac{x_j^2}{1+x_j^2}\right\}$$

where $a_i\in\mathbb{R}^d, b_i\in\{-1,1\}$ are the training data provided by [1] and $\lambda>0$ is the regularization parameter;

[1] Chih-Chung Chang and Chih-Jen Lin. **Libsvm: a library for support vector machines.** ACM TIST, 2(3):1–27, 2011.

Real experiments

$$\min_{x\in\mathbb{R}^d}\left\{f(x)=\frac{1}{N}\sum_{i=1}^{N}\log\left(1+\exp\left(-b_i a_i^\top x\right)\right)+\lambda\sum_{j=1}^{d}\frac{x_j^2}{1+x_j^2}\right\}$$

where $a_i\in\mathbb{R}^d, b_i\in\{-1,1\}$ are the training data provided by [1] and $\lambda>0$ is the regularization parameter;

Dataset	n	N (total # of datapoints)	d (# of features)	N_i
w8a	100	49,749	300	497
a9a	100	32,560	123	325

[1] Chih-Chung Chang and Chih-Jen Lin. **Libsvm: a library for support vector machines.** ACM TIST, 2(3):1–27, 2011.

Real experiments

$$\min_{x\in\mathbb{R}^d}\left\{f(x)=\frac{1}{N}\sum_{i=1}^{N}\log\left(1+\exp\left(-b_i a_i^\top x\right)\right)+\lambda\sum_{j=1}^{d}\frac{x_j^2}{1+x_j^2}\right\}$$

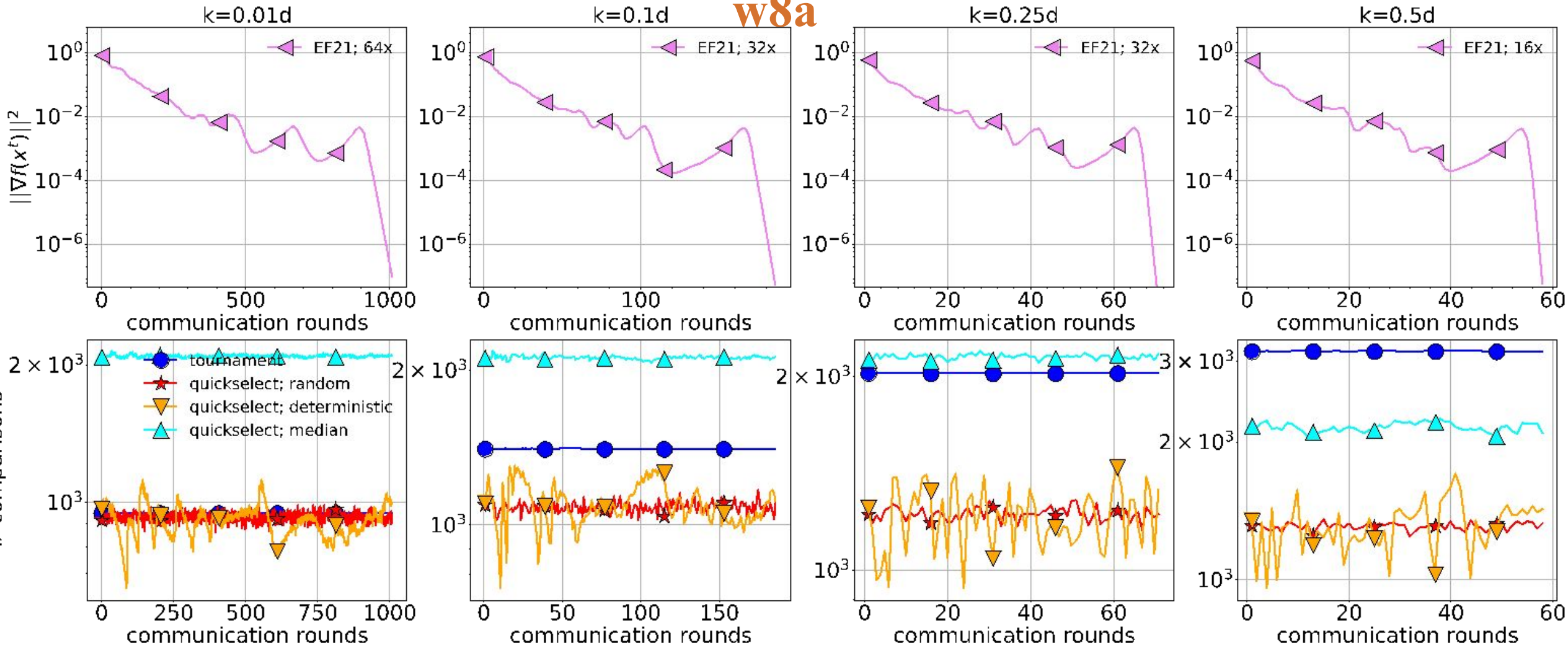
where $a_i\in\mathbb{R}^d, b_i\in\{-1,1\}$ are the training data provided by [1] and $\lambda>0$ is the regularization parameter;

Dataset	n	N (total # of datapoints)	d (# of features)	N_i
w8a	100	49,749	300	497
a9a	100	32,560	123	325

Instead of the classical distributed sgd we will consider more stable method - **EF21** [2],[3]

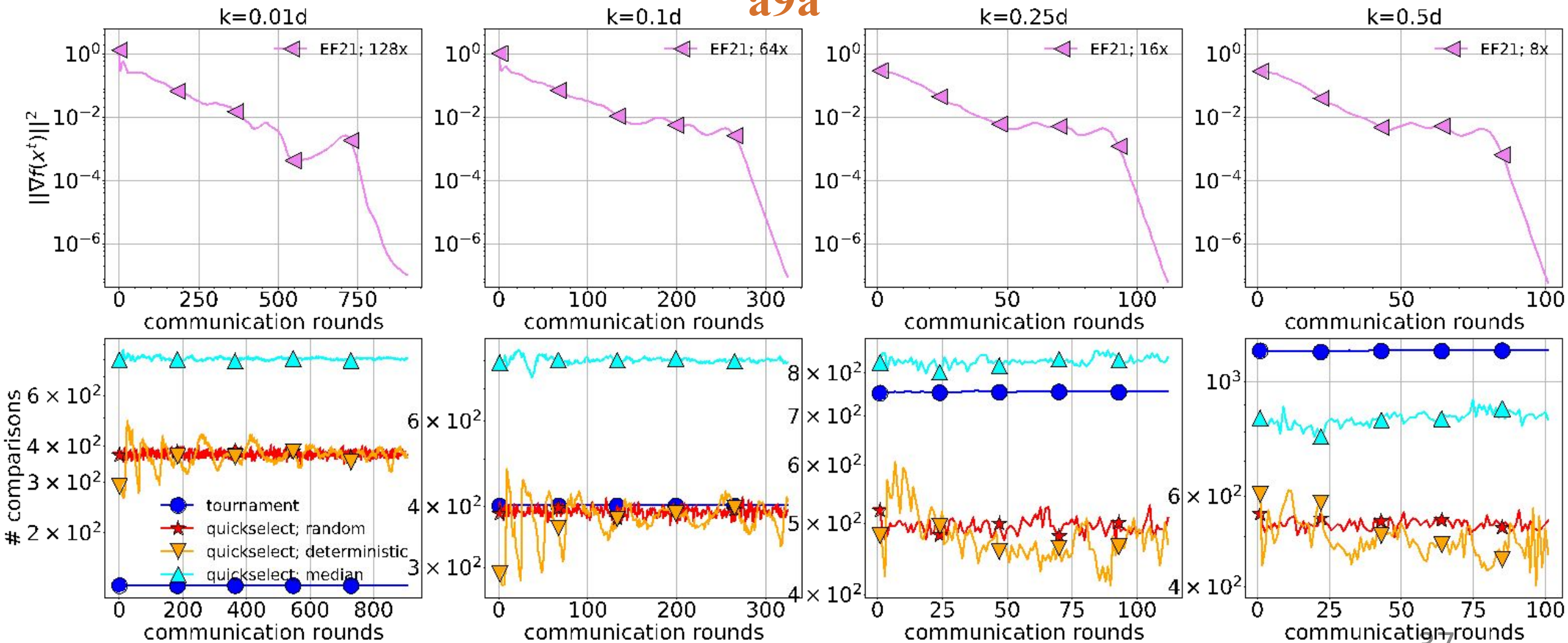
[1] Chih-Chung Chang and Chih-Jen Lin. **Libsvm: a library for support vector machines.** ACM TIST, 2(3):1–27, 2011.
[2] Peter Richtárik, Igor Sokolov and Ilyas Fatkhullin **EF21: A new, simpler, theoretically better, and practically faster error feedback**, NeurIPS, 2021 (oral)
[3] Ilyas Fatkhullin, Igor Sokolov, Eduard Gorbunov, Zhize Li, Peter Richtárik **EF21 with Bells & Whistles: Practical Algorithmic Extensions of Modern Error Feedback**, NeurIPS, 2021 (OptML workshop)

Real experiments



Real experiments

a9a



Conclusion

1. Synthetic experiments: inputs are sampled from known priors

- Choice of prior does not affect on the distribution of the time complexity
- For small d and $k < 0.1d$ Tournament method has the lowest time complexity, whereas for $k > 0.2d$ it is the slowest method
- Quick select with random and deterministic pivots show similar results, outperforming median one and Tournament method

Conclusion

2. Real experiments: inputs are taken from ML model training procedure

- For both datasets, for $k<0.1d$ Tournament methods shows the best time complexity and degrades with the increase of k

- Quick select with random and deterministic pivots shows similar results, and do not increase significantly with the increase of k

Thank you for your attention!

Questions?