

Vectorized Data Computing

— Vector databases, privacy, LLM, big models

Ping Li pingli98@gmail.com

<https://pltrees.github.io/>

Updated Version Available

<https://pltrees.github.io/publication/VecDataComp.pdf>

Summary

- Many data types in machine learning and AI can be viewed as “vectors”.
- Vectorized data computing (VDC) is crucial for machine learning and is also much beyond machine learning. It may grow into its own discipline in the near future.
- Vector databases can be viewed as one component in vectorized data computing.
- In most applications, results from **vector databases** (such as similarity search) are quite crude and can serve as the initial screening step (e.g., ads candidate retrieval). AI and machine learning models are necessary for accurate predictions.
- Keys in big models: 1) accuracy 2) training/serving efficiency 3) distributed training. For example, training **trillion-parameter** models for high-accuracy recommender systems.. Many novel algorithms and infrastructure systems are presented.
- **Privacy** and AI model security have become increasingly critical in AI.

Embeddings (Vectors): Memory for AI and LLM

Chat:



Embedding:

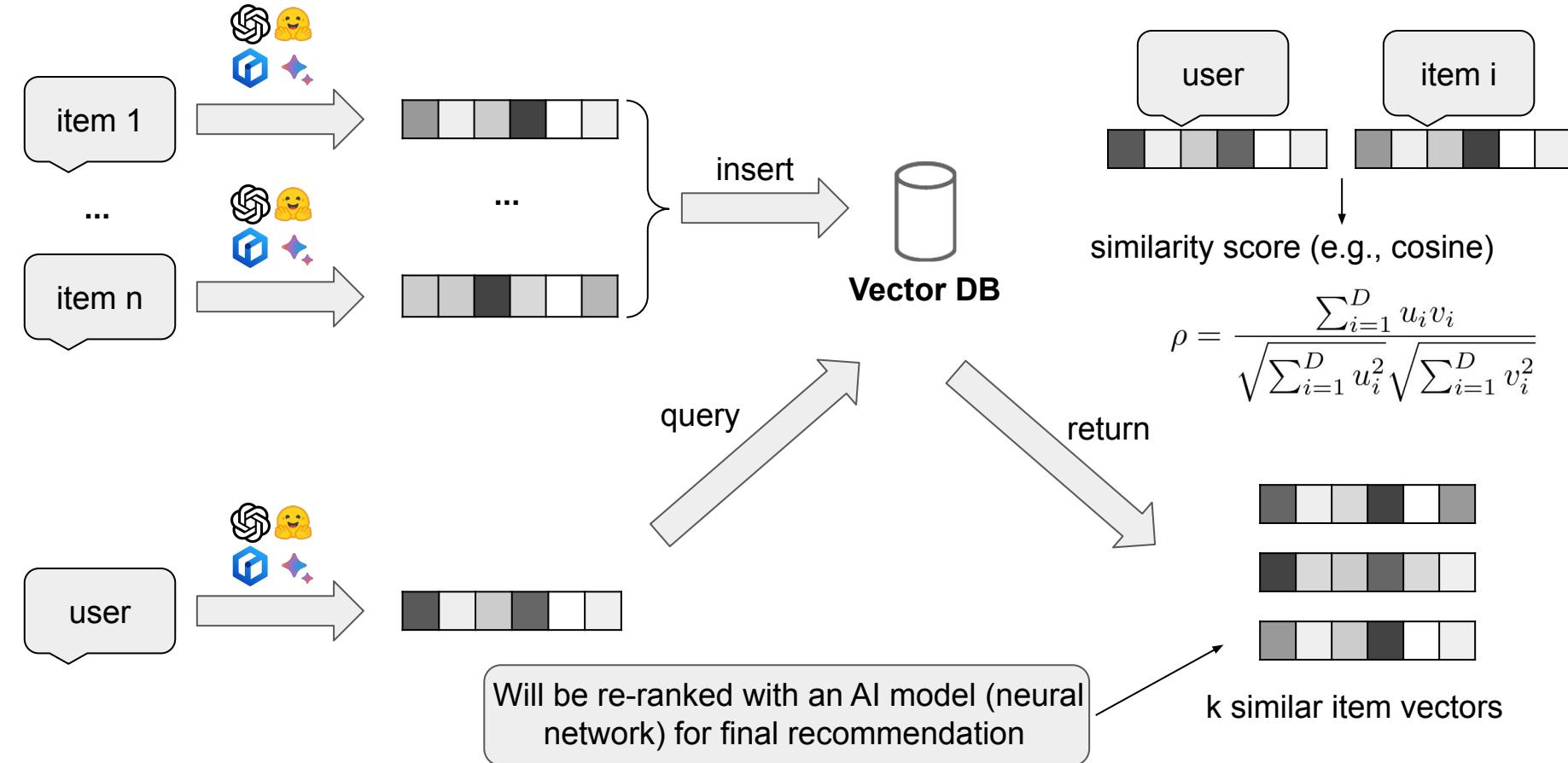


Tasks involving embeddings: fine-tuning models, compact storage, similarity search, weighted similarity search using MIPS (maximum inner product search), fast neural ranking, downstream learning models, privacy protection, and prompt engineering, etc.

Major Use Examples of Vectorized Data

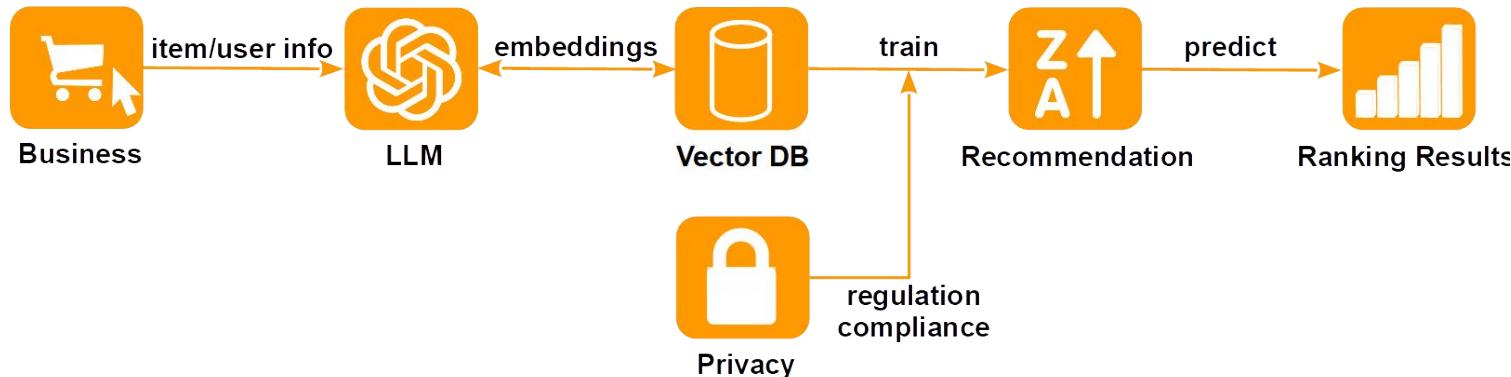
- LLM (large language models) training: A few major players and many startups.
- Retrieval-augmented generation (RAG) for LLM models:
 - More cost-effective than fine-tuning. Reducing hallucinations. Vector DB is a key.
- LLM/AIGC applications:
 - Vector database (DB) is crucial for applications of LLM/AIGC.
- Recommender systems (in the broad sense):
 - Search, advertising (ads), feed, recommendation.
 - Major revenue generators of AI
- Other traditional uses of vectorized data:
 - Machine learning models: risk, fraud, trust, security
 - Knowledge-graph embeddings (KGE), e.g., for question-answering (QA)
 - Vectors generated from raw texts without training, e.g., n-grams, shingles.

Introduction to Typical Vector Database (DB)



The Role of Vector DBs in Search and Ads

1. Embeddings have become the crucial component in search and ads.
2. Embedding-based retrieval (EBR) is the key step for retrieval.
3. However, EBR, ANN, Vector DBs etc can only provide only very crude results and hence they can only serve as an important intermediate step in the pipeline of ads and search.
4. AI models for generating embeddings and models for prediction/rankings are crucial. We will focus on big AI models (as well as privacy) for the rest of the presentation.



A Simple Demo

- Amazon movie review: LLM + embedding + AI models :
 - 11,000 movie reviews (raw texts) => LLM => 11,000 embeddings in 384 dimensions.
 - 1,000 embeddings as test (query) vectors , 10,000 base vectors.
 - HNSW and KNN classifiers to predict review ratings (1-5) => 60% accuracy.
 - ABC-Boost trees with 10,000 base vectors for training => 70% accuracy.
 - MLP neural networks => 70% accuracy.
 - MLP neural networks + DCNV2 (deep feature crossing) => 71%.
 - MLP neural networks + BFI (8-block clockwise feature interactions) => 71%.

Only using embeddings and their similarities is typically not sufficient. Many applications will need to build AI models on top of the embeddings from LLM or other methods.

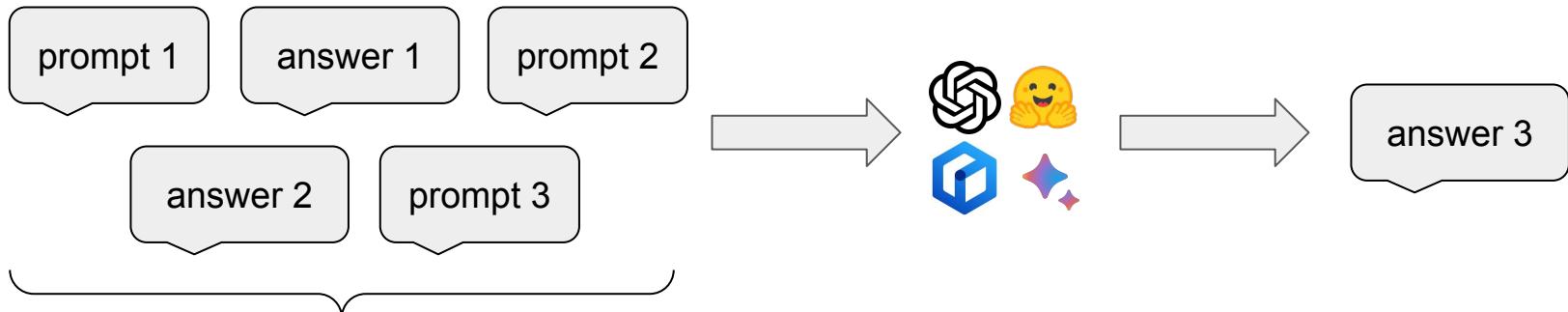
Vector Databases for Improving LLM

Vector databases can also be used for improving LLM. A few examples are:

- History Memory LLM
 - Contextual Memory: By storing contextually relevant information in vector databases, the language model can access and recall it at a later time
- ANN and MIPS for LLM
 - Next word prediction as an MIPS when number of tokens is large
- Multi-Modal Integration in LLM
 - LLM leverage vector databases to store and retrieve multi-modal embeddings. This enables the LLM to associate relevant information across different modalities

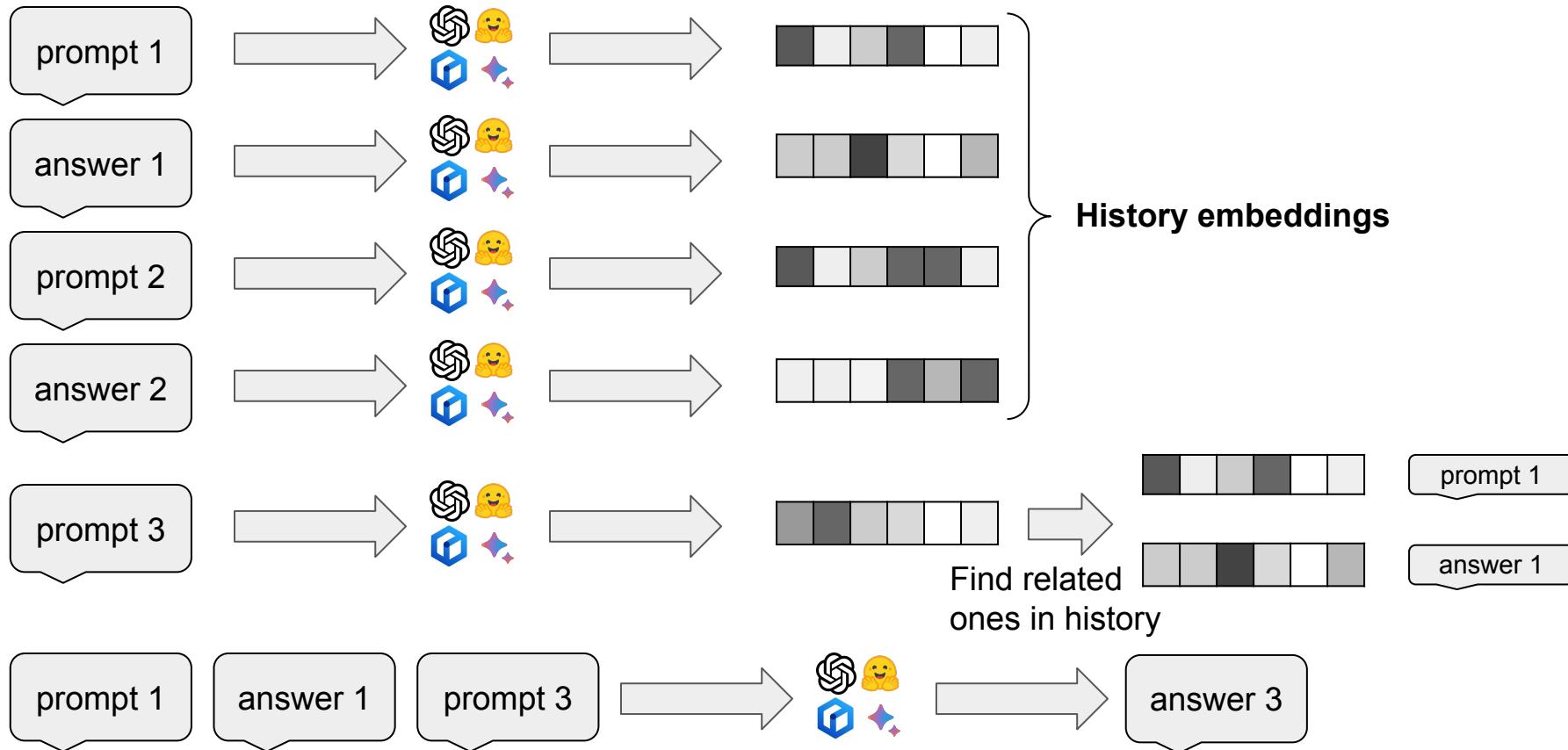
History Memory LLM

**Long-Term History
Memorization**



Too many tokens to be handled

History Memory LLM



History Memory LLM Use Cases

- Customer Support
 - Understand **continuous** customer queries and provide relevant responses
- Data Analysis
 - Process **vast** amounts of data to assist in extracting valuable insights
- Domain-Specific Language Models, e.g., legal
 - Analyze **long** legal contracts, and regulations and assist in drafting legal documents

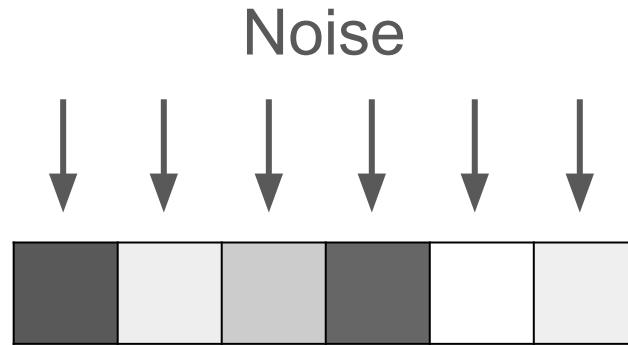
ANN and MIPS in LLM

- Having more tokens (i.e., bigger vocabulary size) can improve the performance of LLM
- Creating more token generates fewer hallucination
- Having too many tokens make the training/inference of the last layer in the LLM slow
 - It becomes an MIPS (max inner product search) task
- Having a lot of tokens also means the token embedding layer super large
- This may require highly efficient ANN search to make the training and inference efficient
- Also, the most work on [Pb-Hash](#) can help deal with super large vocabulary problem.

Privacy for LLM and Vectorized Data

Protecting user data (e.g., embeddings) is a major challenge faced by applications of LLM.

Differential privacy (DP) works by adding (sufficient) noise to every dimension of the vector

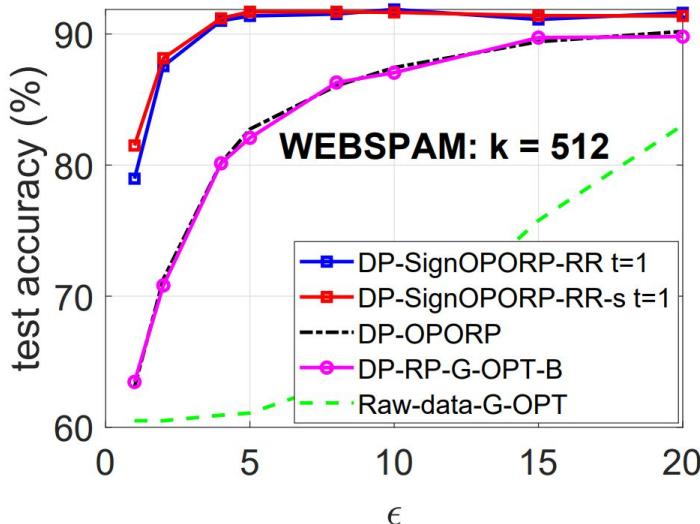


The noises to be added are typically quite significant in order to rigorously satisfy the privacy requirement. Thus, DP-based algorithms usually suffer from poor performance.

Privacy for LLM and Vectorized Data

New DP algorithms based on (e.g.,) random projections (RP) and 1-bit (sign) quantization [arXiv](#).

Green curve: directly adding noise to the original data leads to poor accuracy (y-axis).



Smaller $\epsilon \Rightarrow$ better privacy. $\epsilon \leq 10$ is “cutoff” in [2023 Google survey](#) for reasonable privacy guarantee.

Our methods: At $\epsilon \leq 5 \sim 10$, our new DP methods achieves good accuracy.

Privatized User Data and Embeddings



Our users can securely share privatized embeddings with third parties for building AI models. Alternatively, users can entrust us with the responsibility of constructing models on their behalf, leveraging our state-of-the-art machine learning platforms and expertise.

Our Prior Works Related to Privacy

ICLR'23, [Improved Convergence of Differential Private SGD with Gradient Clipping](#)

ArXiv'23, [Differential Privacy with Random Projections and Sign Random Projections](#)

ArXiv'23, [Differentially Private One Permutation Hashing and Bin-wise Consistent Weighted Sampling](#)

ICML'23, [Regression with Label Permutation in Generalized Linear Model](#)

ICML'23, [One-Step Estimator for Permuted Sparse Recovery](#)

KDD'23, [OPORP: One Permutation + One Random Projection](#)

SIGIR'23, [Building K-Anonymous User Cohorts with Consecutive Consistent Weighted Sampling \(CCWS\)](#)

ArXiv'22, [k-Median Clustering via Metric Embedding: Towards Better Initialization with Differential Privacy](#)

NeurIPS'22, [Breaking the Linear Error Barrier in Differentially Private Graph Distance Release](#)

IEEE CNS'22, [NL2GDPR: Automatically Develop GDPR Compliant Android Application from Natural Language](#)

ISIT'22, [Distances Release with Differential Privacy in Tree and Grid Graph](#)

AI Model Security

In addition to privacy, the AI model security has become increasingly important:

- FACT SHEET: Biden-Harris Administration Secures Voluntary Commitments from Leading Artificial Intelligence Companies to Manage the Risks Posed by AI, [link](#),
- The Blueprint for an AI Bill of Rights, [link](#)

The list of our prior works on AI model security:

AAAI 2023, [Defending Backdoor Attacks on Vision Transformer via Patch Processing](#)

NeurIPS 2022, [Marksman Backdoor: Backdoor Attacks with Arbitrary Target Class](#)

KDD 2022, [Integrity Authentication in Tree Models](#)

ICDE 2022, [Identification for Deep Neural Network: Simply Adjusting Few Weights!](#)

AAAI 2022, [DeepAuth: A DNN Authentication Framework by Model-Unique and Fragile Signature Embedding](#)

NeurIPS 2021, [Backdoor Attack with Imperceptible Input and Latent Modification](#)

ICCV 2021, [LIRA: Learnable, Imperceptible and Robust Backdoor Attacks](#)

ICCV 2021, [Robust Watermarking for Deep Neural Networks via Bi-level Optimization](#)

Outline

1. **Vector similarity functions**
2. Vector compressions
3. Vector similarity search
4. Maximum inner product search (MIPS)
5. Fast neural ranking
6. GPU computing
7. GCWSNet, hashing algorithms
8. Boosted trees, ABC-boost
9. Privacy
10. Security
11. Distributed, adaptive, and federated learning
12. Others: generative AI models, NLP, knowledge graphs, multi-modal, cross-modal, advertising

Vector Similarity Functions

Consider vectors in D dimensions



$$u \in \mathbb{R}^D$$

D can be (e.g.,) 1024 or much larger



$$v \in \mathbb{R}^D$$

Inner product: $a = \sum_{i=1}^D u_i v_i$

Euclidean distance: $d = \sum_{i=1}^D |u_i - v_i|^2$

The popular measure of similarity is the **cosine**

$$\rho = \frac{\sum_{i=1}^D u_i v_i}{\sqrt{\sum_{i=1}^D u_i^2} \sqrt{\sum_{i=1}^D v_i^2}}$$

Another popular similarity is the **RBF** (Gaussian) kernel

$$RBF = e^{-\lambda(1-\rho)}, \lambda > 0$$

More Vector Similarity Functions



$$u \in \mathbb{R}^D$$



$$v \in \mathbb{R}^D$$

L_p distance: $d_p = \sum_{i=1}^D |u_i - v_i|^p, \ p \geq 0$

Chi-square similarity $\rho_{\chi^2} = \sum_{i=1}^D \frac{2u_i v_i}{u_i + v_i}, \quad u_i \geq 0, \ v_i \geq 0, \ \sum_{i=1}^D u_i = 1, \ \sum_{i=1}^D v_i = 1$

Min-Max similarity $MinMax = \frac{\sum_{i=1}^D \min\{u_i, v_i\}}{\sum_{i=1}^D \max\{u_i, v_i\}}, \quad u_i \geq 0, \ v_i \geq 0$

Early Works on L_p Distance (or Norms)

$$\text{L}_p \text{ distance: } d_p = \sum_{i=1}^D |u_i - v_i|^p, \quad p \geq 0$$

$$\text{L}_p \text{ norm: } l_p = \sum_{i=1}^D |u_i|^p, \quad p \geq 0$$

Estimators and Tail Bounds for Dimension Reduction in l_α
 $(0 < \alpha \leq 2)$ Using Stable Random Projections

SODA 2008

Compressed Counting

SODA 2009

Very Sparse Stable Random Projections for Dimension
Reduction in l_α ($0 < \alpha \leq 2$) Norm

KDD 2007

Improving Compressed Counting

UAI 2009

Anomaly Detection using L_p Norms or Entropy

NIPS 2008

A Unified Near-Optimal Estimator For Dimension Reduction in l_α ($0 < \alpha \leq 2$) Using Stable Random Projections

Ping Li

Department of Statistical Science
Faculty of Computing and Information Science
Cornell University
pingli@cornell.edu

Trevor J. Hastie

Department of Statistics
Department of Health, Research and Policy
Stanford University
hastie@stanford.edu

A New Algorithm for Compressed Counting with Applications in Shannon Entropy Estimation in Dynamic Data

COLT 2011

Ping Li
Department of Statistical Science
Cornell University
Ithaca, NY 14853
pingli@cornell.edu

Cun-Hui Zhang
Department of Statistics and Biostatistics
Rutgers University
New Brunswick, NJ 08901
czhang@stat.rutgers.edu

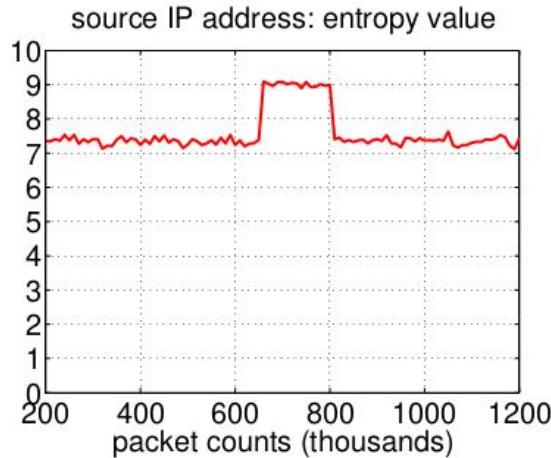
Entropy Estimations Using Correlated Symmetric Stable Random Projections

NIPS 2012

Ping Li
Department of Statistical Science
Cornell University
Ithaca, NY 14853
pingli@cornell.edu

Cun-Hui Zhang
Department of Statistics and Biostatistics
Rutgers University
New Brunswick, NJ 08901
czhang@stat.rutgers.edu

One major application is to detect (e.g.,) DDoS network attacks using entropy. Figure re-generated from a DARPA report.



AISTATS 2017

Binary and Multi-Bit Coding for Stable Random Projections

Compressed Sensing using (Sign) Projections

Exact Sparse Recovery with L0 Projections KDD 2013

Ping Li

Department of Statistical Science
Cornell University
Ithaca, NY 14853, USA
pingli@cornell.edu

Cun-Hui Zhang

Department of Statistics and Biostatistics
Rutgers University
New Brunswick, NJ 08901, USA
cunhui@stat.rutgers.edu

Compressed Counting Meets Compressed Sensing COLT 2014

Ping Li

*Department of Statistics and Biostatistics, Department of Computer Science,
Rutgers University, Piscataway, NJ 08854, USA*

COLT 2014

PINGLI@STAT.RUTGERS.EDU

Cun-Hui Zhang

Department of Statistics and Biostatistics, Rutgers University, Piscataway, NJ 08854, USA

CUNHUI@STAT.RUTGERS.EDU

Tong Zhang

Department of Statistics and Biostatistics, Rutgers University, Piscataway, NJ 08854, USA

TZHANG@STAT.RUTGERS.EDU

Compressed Sensing with Very Sparse Gaussian Random Projections

AISTATS 2015

Ping Li

Department of Statistics and Biostatistics
Department of Computer Science
Rutgers University
Piscataway, NJ 08854, USA
pingli@stat.rutgers.edu

Cun-Hui Zhang

Department of Statistics and Biostatistics
Rutgers University
Piscataway, NJ 08854, USA
cunhui@stat.rutgers.edu

One Scan 1-Bit Compressed Sensing

AISTATS 2016

Machine Learning using L_p Distance

$$\text{L}_p \text{ distance: } d_p = \sum_{i=1}^D |u_i - v_i|^p, \quad p \geq 0$$

Approximating Higher-Order Distances Using Random Projections

UAI 2010

Ping Li*

Department of Statistical Science
Faculty of Computing and Information Science
Cornell University, Ithaca, NY 14853
pingli@cornell.edu

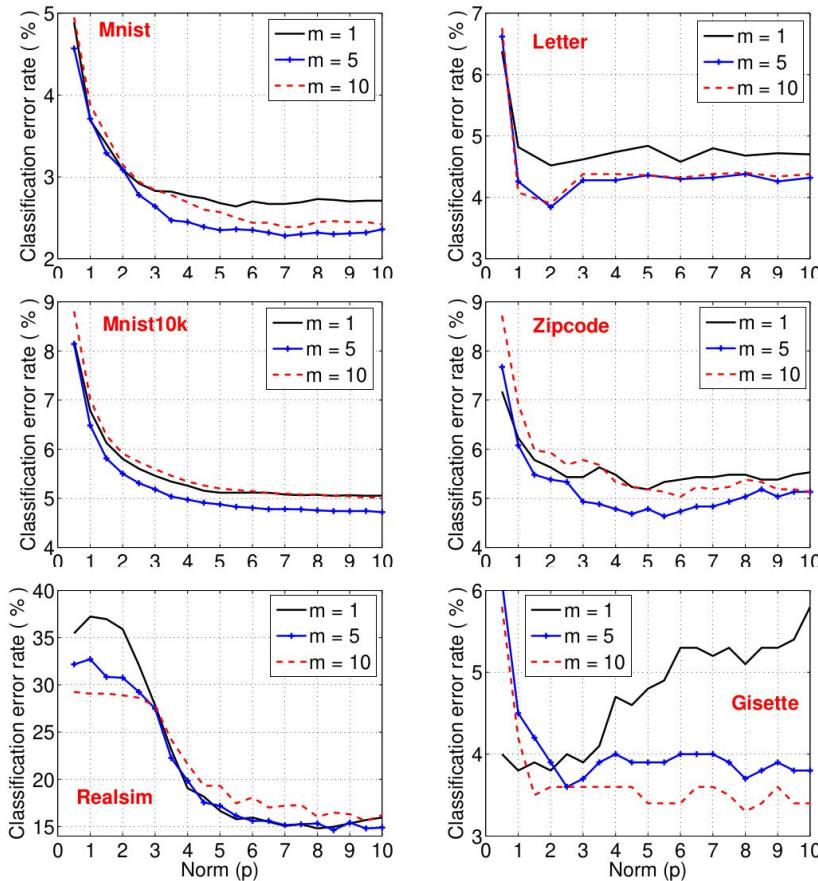
Michael W. Mahoney†

Department of Mathematics
Stanford University
Stanford, CA 94305
mmahoney@cs.stanford.edu

Yiyuan She

Department of Statistics
Florida State University
Tallahassee, FL 32306
yshe@stat.fsu.edu

(Perhaps surprising!) For many datasets, the best classification accuracy is achieved at $p > 2$, using m -nearest neighbor classifier ($m = 1, 5, 10$).



ABC-Boost Trees for L_p Regression

$$L_p = \frac{1}{n} \sum_{i=1}^n L_{p,i} = \frac{1}{n} \sum_{i=1}^n |y_i - F_i|^p, \quad \text{where } F_i = F(\mathbf{x}_i).$$
$$\frac{\partial L_{p,i}}{\partial F_i} = -p|y_i - F_i|^{p-1}\text{sign}(y_i - F_i) \quad \frac{\partial^2 L_{p,i}}{\partial F_i^2} = p(p-1)|y_i - F_i|^{p-2}$$
$$Gain(s) = \frac{\left[\sum_{i=1}^s L'_i \right]^2}{\sum_{i=1}^s L''_i} + \frac{\left[\sum_{i=s+1}^N L'_i \right]^2}{\sum_{i=s+1}^N L''_i} - \frac{\left[\sum_{i=1}^N L'_i \right]^2}{\sum_{i=1}^N L''_i} \quad \text{tree split criterion}$$

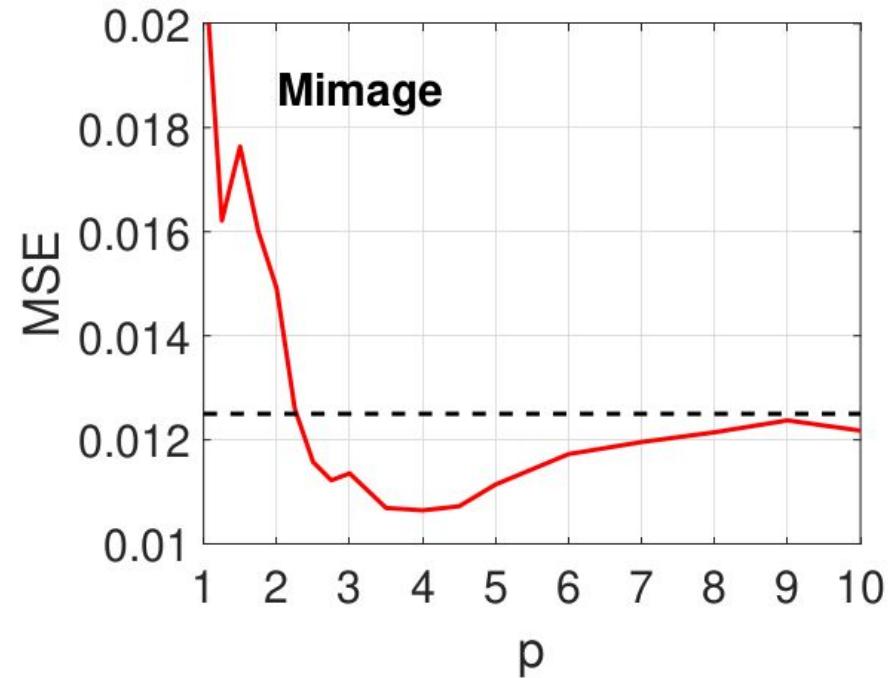
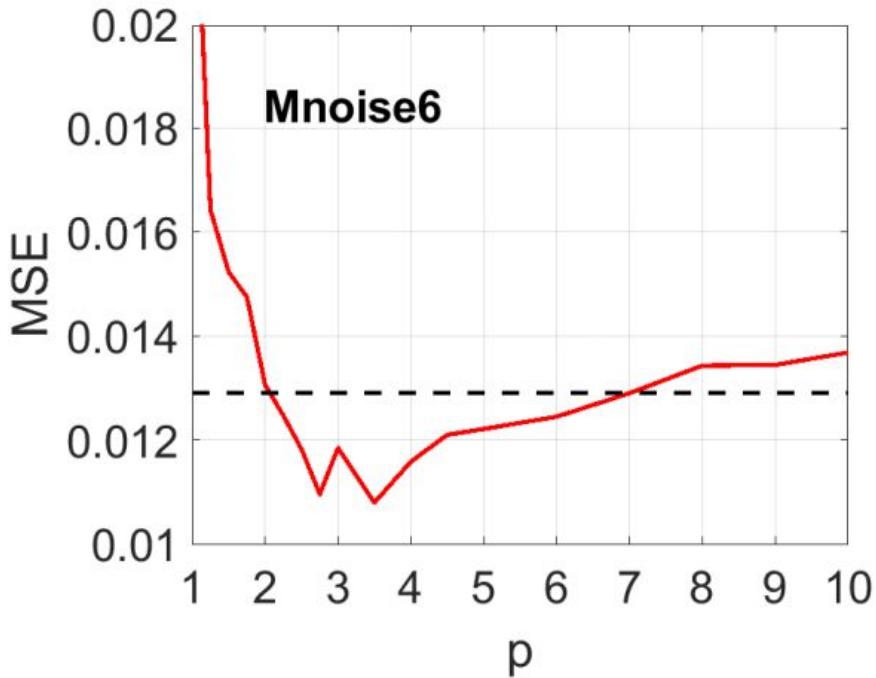
ArXiv'22, [pGMM Kernel Regression and Comparisons with Boosted Trees](#)

ArXiv'22, [Package for Fast ABC-Boost](#)

GitHub, <https://github.com/pltrees/abcboost>

ABC-Boost Trees for L^p Regression

ArXiv'22, [pGMM Kernel Regression and Comparisons with Boosted Trees](#)



pGMM Kernel versus Boosted L2 Regression

ArXiv'22, [pGMM Kernel Regression and Comparisons with Boosted Trees](#)

Table 1: Datasets for testing regression algorithms. We report the best test mean square error (MSE) for each method, over the range of regularization coefficients and parameters.



dataset	# train	# test	dim	LR	RBF	GMM	pGMM	L_2 -Boost
ENBcool	384	384	8	10.24	3.20	1.70	1.28	1.21
ENBheat	384	384	8	9.00	0.495	0.191	0.188	0.186
Airfoil	752	751	5	24.26	8.35	7.50	3.56	3.09
CPUsmall	4096	4096	12	102.12	9.05	7.22	7.05	6.89
CPU	4096	4096	21	98.35	6.42	5.17	5.03	4.69
WECPerth	5000	5000	32	1.1×10^9	2.3×10^8	2.4×10^8	2.3×10^8	2.5×10^8
Cadata	10320	10320	8	4.8×10^9	3.8×10^9	2.4×10^9	2.4×10^9	2.1×10^9
House16H	11392	11392	16	2.1×10^9	1.3×10^9	1.2×10^9	1.1×10^9	1.0×10^9
House16L	11392	11392	16	1.9×10^9	1.1×10^9	9.9×10^8	9.4×10^8	8.6×10^8
CASP	22865	22865	9	26.63	23.94	15.30	15.29	13.51
Splice	1000	2175	60	0.1205	0.0967	0.0589	0.0589	0.0352
Mnoise1	2519	454	784	0.0484	0.0344	0.0311	0.0169	0.0145
Mnoise6	2519	454	784	0.0215	0.0165	0.0195	0.0129	0.0131
Mimage	2538	10524	784	0.0540	0.0323	0.0263	0.0125	0.0149

Min-Max (MM) Kernel and pGMM Kernel



$$u \in \mathbb{R}^D$$



$$v \in \mathbb{R}^D$$

$$\text{MinMax} = \frac{\sum_{i=1}^D \min\{u_i, v_i\}}{\sum_{i=1}^D \max\{u_i, v_i\}}, \quad u_i \geq 0, v_i \geq 0$$

Most natural data vectors are non-negative. Embedding vectors using “ReLu” activation are also non-negative. For general data vectors with negative entries, we do the following:

$$\begin{cases} \tilde{u}_{2i-1} = u_i, \quad \tilde{u}_{2i} = 0 & \text{if } u_i > 0 \\ \tilde{u}_{2i-1} = 0, \quad \tilde{u}_{2i} = -u_i & \text{if } u_i \leq 0 \end{cases} \quad GMM(u, v) = \frac{\sum_{i=1}^{2D} \min(\tilde{u}_i, \tilde{v}_i)}{\sum_{i=1}^{2D} \max(\tilde{u}_i, \tilde{v}_i)}$$

For example, when $D = 2$ and $u = [-5 \ 3]$, the transformed data vector becomes $\tilde{u} = [0 \ 5 \ 3 \ 0]$

A tuning parameter p can also be introduced:

$$pGMM(u, v; p) = \frac{\sum_{i=1}^{2D} (\min\{\tilde{u}_i, \tilde{v}_i\})^p}{\sum_{i=1}^{2D} (\max\{\tilde{u}_i, \tilde{v}_i\})^p}$$

Min-Max (MM) Kernel and pGMM Kernel

0-Bit Consistent Weighted Sampling

KDD 2015

Linearized GMM Kernels and Normalized Random Fourier Features

KDD 2017

Nystrom Method for Approximating the GMM Kernel

ArXiv 2016

Tunable GMM Kernels

ArXiv 2017

Min-Max (MM) Kernel and pGMM Kernel

Table 1: **Public (UCI) classification datasets and l_2 -regularized kernel SVM results.** We report the test classification accuracies for the linear kernel, the best-tuned RBF kernel, the original (tuning-free) GMM kernel, the best-tuned eGMM kernel, and the best-tuned pGMM kernel, at their individually-best SVM regularization C values.

Dataset	# train	# test	# dim	linear	RBF	GMM	eGMM	pGMM
Car	864	864	6	71.53	94.91	98.96	99.31	99.54
Covertype25k	25000	25000	54	62.64	82.66	82.65	88.32	83.25
CTG	1063	1063	35	60.59	89.75	88.81	88.81	100.00
DailySports	4560	4560	5625	77.70	97.61	99.61	99.61	99.61
DailySports2k	2000	7120	5625	72.16	93.71	98.99	99.00	99.07
Dexter	300	300	19999	92.67	93.00	94.00	94.00	94.67
Gesture	4937	4936	32	37.22	61.06	65.50	66.67	66.33
ImageSeg	210	2100	19	83.81	91.38	95.05	95.38	95.57
Isolet2k	2000	5797	617	93.95	95.55	95.53	95.55	95.53
MHealth20k	20000	20000	23	72.62	82.65	85.28	85.33	86.69
MiniBooNE20k	20000	20000	50	88.42	93.06	93.00	93.01	93.72
MSD20k	20000	20000	90	66.72	68.07	71.05	71.18	71.84
Magic	9150	9150	10	78.04	84.43	87.02	86.93	87.57
Musk	3299	3299	166	95.09	99.33	99.24	99.24	99.24
Musk2k	2000	4598	166	94.80	97.63	98.02	98.02	98.06
PageBlocks	2737	2726	10	95.87	97.08	96.56	96.56	97.33
Parkinson	520	520	26	61.15	66.73	69.81	70.19	69.81
PAMAP101	20000	20000	51	76.86	96.68	98.91	98.91	99.00
PAMAP102	20000	20000	51	81.22	95.67	98.78	98.77	98.78
PAMAP103	20000	20000	51	85.54	97.89	99.69	99.70	99.69

ArXiv 2017

Min-Max (MM) Kernel and pGMM Kernel

PAMAP103	20000	20000	51	85.54	97.89	99.69	99.70	99.69
PAMAP104	20000	20000	51	84.03	97.32	99.30	99.31	99.30
PAMAP105	20000	20000	51	79.43	97.34	99.22	99.24	99.22
RobotNavi	2728	2728	24	69.83	90.69	96.85	96.77	98.20
Satimage	4435	2000	36	72.45	85.20	90.40	91.85	90.95
SEMG1	900	900	3000	26.00	43.56	41.00	41.22	42.89
SEMG2	1800	1800	2500	19.28	29.00	54.00	54.00	56.11
Sensorless	29255	29254	48	61.53	93.01	99.39	99.38	99.76
Shuttle500	500	14500	9	91.81	99.52	99.65	99.65	99.66
SkinSeg10k	10000	10000	3	93.36	99.74	99.81	99.90	99.85
SpamBase	2301	2300	57	85.91	92.57	94.17	94.13	95.78
Splice	1000	2175	60	85.10	90.02	95.22	96.46	95.26
Theorem	3059	3059	51	67.83	70.48	71.53	71.69	71.53
Thyroid	3772	3428	21	95.48	97.67	98.31	98.34	99.10
Thyroid2k	2000	5200	21	94.90	97.00	98.40	98.40	98.96
Urban	168	507	147	62.52	51.48	66.08	65.68	83.04
Vertebral	155	155	6	80.65	83.23	89.04	89.68	89.04
Vowel	264	264	10	39.39	94.70	96.97	98.11	96.97
Wholesale	220	220	6	89.55	90.91	93.18	93.18	93.64
Wilt	4339	500	5	62.60	83.20	87.20	87.60	87.40
YoutubeAudio10k	10000	11930	2000	41.35	48.63	50.59	50.60	51.84
YoutubeHOG10k	10000	11930	647	62.77	66.20	68.63	68.65	72.06
YoutubeMotion10k	10000	11930	64	26.24	28.81	31.95	33.05	32.65
YoutubeSaiBoxes10k	10000	11930	7168	46.97	49.31	51.28	51.22	52.15
YoutubeSpectrum10k	10000	11930	1024	26.81	33.54	39.23	39.27	41.23

ArXiv 2017

Min-Max (MM) Kernel and pGMM Kernel

Table 2: Datasets in group 1 are from the LIBSVM website. Datasets in group 2 were used by for testing deep learning algorithms and tree methods.

Group	Dataset	# train	# test	# dim	linear	RBF	GMM	eGMM	pGMM
1	Letter	15000	5000	16	61.66	97.44	97.26	97.68	97.32
	Protein	17766	6621	357	69.14	70.32	70.64	71.03	71.48
	SensIT20k	20000	19705	100	80.42	83.15	84.57	84.69	84.90
	Webspam20k	20000	60000	254	93.00	97.99	97.88	98.21	97.93
ArXiv 2017	M-Basic	12000	50000	784	89.98	97.21	96.34	96.47	96.40
	M-Image	12000	50000	784	70.71	77.84	80.85	81.20	89.53
	M-Noise1	10000	4000	784	60.28	66.83	71.38	71.70	85.20
	M-Noise2	10000	4000	784	62.05	69.15	72.43	72.80	85.40
	M-Noise3	10000	4000	784	65.15	71.68	73.55	74.70	86.55
	M-Noise4	10000	4000	784	68.38	75.33	76.05	76.80	86.88
	M-Noise5	10000	4000	784	72.25	78.70	79.03	79.48	87.33
	M-Noise6	10000	4000	784	78.73	85.33	84.23	84.58	88.15
	M-Rand	12000	50000	784	78.90	85.39	84.22	84.95	89.09
	M-Rotate	12000	50000	784	47.99	89.68	84.76	86.02	86.52
	M-RotImg	12000	50000	784	31.44	45.84	40.98	42.88	54.58

Boosted Trees (ABC-Boost) vs. pGMM Kernel

Table 4: Test error rates of five additional datasets reported in [4, 7]. The results in group 1 are from [4], where they compared kernel SVM, neural nets, and deep learning. The results in group 3 are from [7], which compared four boosted tree methods with deep nets.

Group	Method	M-Basic	M-Rotate	M-Image	M-Rand	M-RotImg
1	SVM-RBF	3.05%	11.11%	22.61%	14.58%	55.18%
	SVM-POLY	3.69%	15.42%	24.01%	16.62%	56.41%
	NNET	4.69%	18.11%	27.41%	20.04%	62.16%
	DBN-3	3.11%	10.30%	16.31%	6.73%	47.39%
	SAA-3	3.46%	10.30%	23.00%	11.28%	51.93%
	DBN-1	3.94%	14.69%	16.15%	9.80%	52.21%
2	Linear	10.02%	52.01%	29.29%	21.10%	68.56%
	RBF	2.79%	10.30%	22.16%	14.61%	54.16%
	GMM	3.80%	15.24%	19.15%	15.78%	59.02%
	eGMM	3.53%	13.98%	18.80%	15.05%	57.12%
	pGMM	3.63%	13.44%	10.47%	10.91%	45.42%
	epGMM	3.29%	11.81%	10.04%	10.57%	44.27%
3	mart	4.12%	15.35%	11.64%	13.15%	49.82%
	abc-mart	3.69%	13.27%	9.45%	10.60%	46.14%
	robust logit	3.45%	13.63%	9.41%	10.04%	45.92%
	abc-robust-logit	3.20%	11.92%	8.54%	9.45%	44.69%

ArXiv 2017

<https://github.com/pltrees/abcboost>

Do We Still Care about Kernels?

Do neural nets learn all nonlinearities? Largely true. For example, we ourselves haven't observed a case in which the Gaussian kernel outperforms the deep nets.

On the other hand, **the family of pGMM kernels is special**. The discontinuity of the kernels is probably (part of) the reason why pGMM kernels can in many cases outperform neural nets. Tree models are another example of discontinuous models, and hence it is not surprising that tree models often outperforms neural nets.

$$\text{MinMax} = \frac{\sum_{i=1}^D \min\{u_i, v_i\}}{\sum_{i=1}^D \max\{u_i, v_i\}}$$

GCWSNet

GCWSNet: Generalized Consistent Weighted Sampling
for Scalable and Accurate Training of Neural Networks

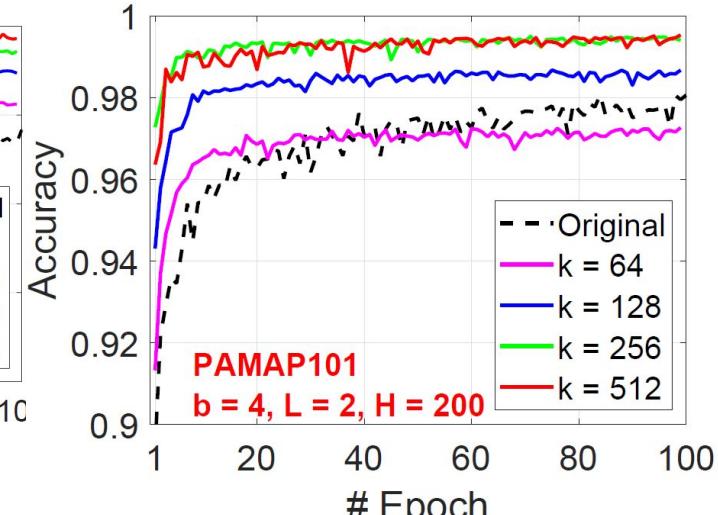
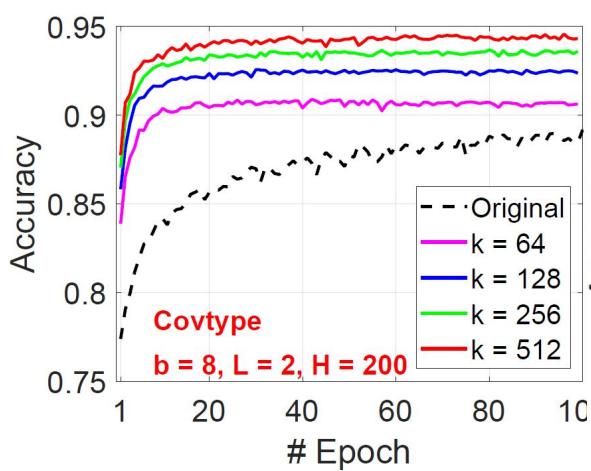
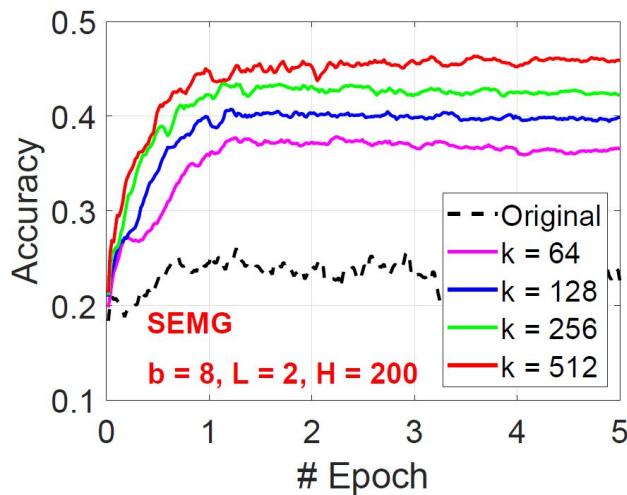
CIKM 2022

Ping Li and Weijie Zhao

Cognitive Computing Lab
Baidu Research

10900 NE 8th St. Bellevue, WA 98004, USA
`{liping11, weijiezhaoy}@baidu.com`

Training neural nets ($L = 2$ layer and $H = 200$ hidden nodes) on CWS hashed data for pGMM kernels, we observe noticeable improvement over directly training neural nets on the original data (black dashed)



Theory of the GMM Kernel

GMM kernel and cosine are surprisingly related, under symmetric data assumption.

$$\left\{ \begin{array}{ll} \tilde{u}_{2i-1} = u_i, & \tilde{u}_{2i} = 0 & \text{if } u_i > 0 \\ \tilde{u}_{2i-1} = 0, & \tilde{u}_{2i} = -u_i & \text{if } u_i \leq 0 \end{array} \right. \quad GMM(u, v) = \frac{\sum_{i=1}^{2D} \min(\tilde{u}_i, \tilde{v}_i)}{\sum_{i=1}^{2D} \max(\tilde{u}_i, \tilde{v}_i)}$$

$$\rho = \frac{\sum_{i=1}^D u_i v_i}{\sqrt{\sum_{i=1}^D u_i^2} \sqrt{\sum_{i=1}^D v_i^2}} \quad GMM(u, v) \rightarrow \frac{1 - \sqrt{(1 - \rho)/2}}{1 + \sqrt{(1 - \rho)/2}}$$

Theory of the GMM Kernel

WWW 2017

Ping Li

Department of Statistics and Biostatistics

Department of Computer Science

Rutgers University

Piscataway, NJ 08854, USA

pingli@stat.rutgers.edu

Cun-Hui Zhang

Department of Statistics and Biostatistics

Rutgers University

Piscataway, NJ 08854, USA

cunhui@stat.rutgers.edu

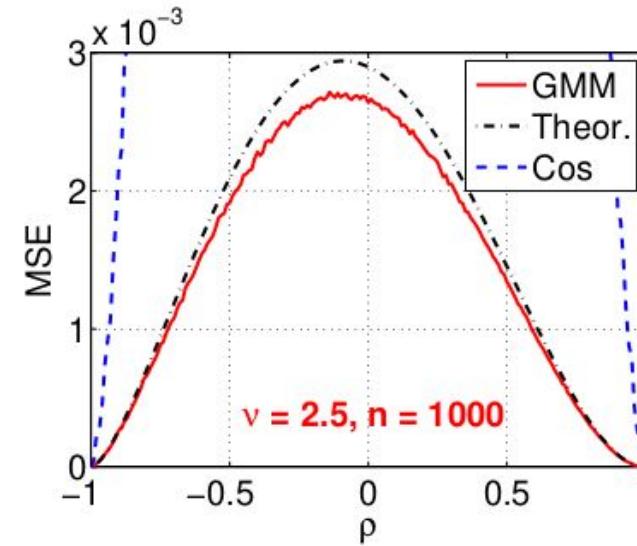
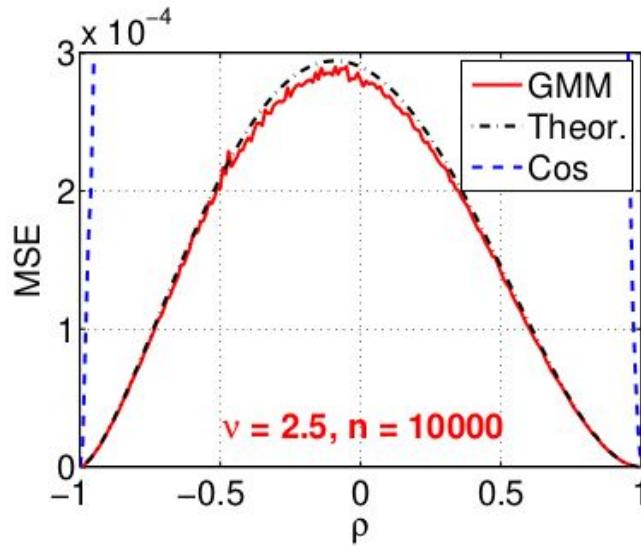
Theory of the GMM Kernel

$$\rho = \frac{\sum_{i=1}^D u_i v_i}{\sqrt{\sum_{i=1}^D u_i^2} \sqrt{\sum_{i=1}^D v_i^2}} \quad GMM(u, v) = \frac{\sum_{i=1}^{2D} \min(\tilde{u}_i, \tilde{v}_i)}{\sum_{i=1}^{2D} \max(\tilde{u}_i, \tilde{v}_i)} \rightarrow \frac{1 - \sqrt{(1 - \rho)/2}}{1 + \sqrt{(1 - \rho)/2}}$$

- We consider the model that the coordinates are i.i.d. samples from a symmetric bivariate distribution. We study the limit when the dimension D goes to infinity.
- Assuming bounded 2nd moment, the cosine converges to the true correlation.
- Assuming only bounded 1st moment, GMM converges to a function of correlation.
- Thus, GMM is substantially more robust than the cosine. In other words, if the data do not have bounded 2nd moment, then it is meaningless to use the cosine.
- Interestingly, even when the data are perfectly Gaussian (in this case, the cosine will be optimal), using GMM only very slightly loses the accuracy.

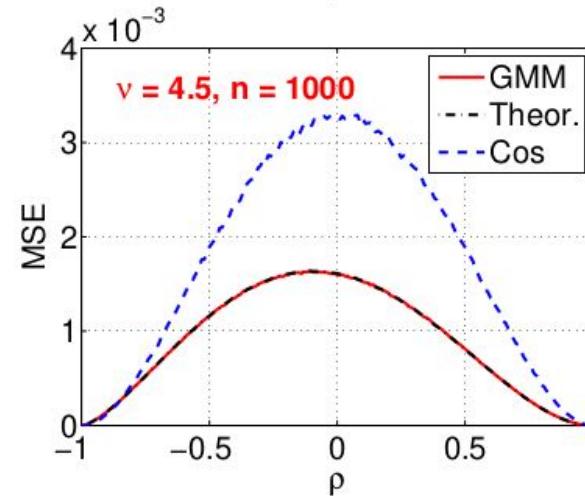
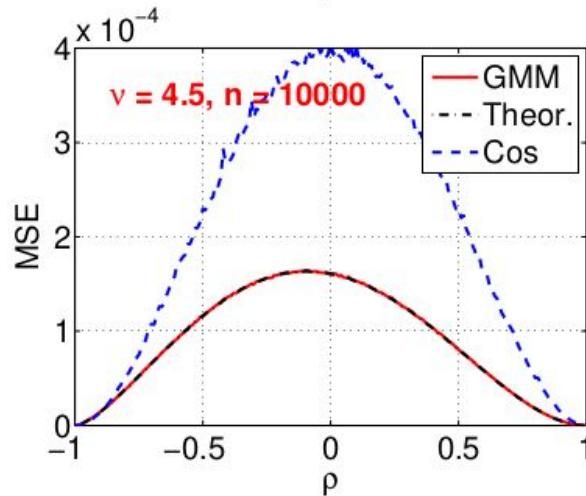
Theory of the GMM Kernel

- Use GMM and cosine to estimate the true correlation. In order to have bounded MSE (variance), GMM and cosine require, respectively, bounded **2nd** and **4th** moments.
- Dimension $D = n = 1000$, or 10000 . $v = 2.5$ means bounded 2.5-th moment.
- The GMM converges nicely and the variance follows the theoretical prediction.
- The estimates by the cosine do not have bounded MSE (variance).
- The GMM kernel is substantially more robust than the cosine.



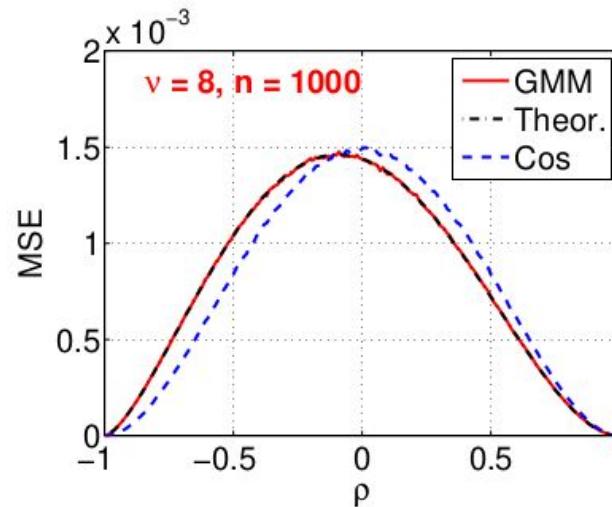
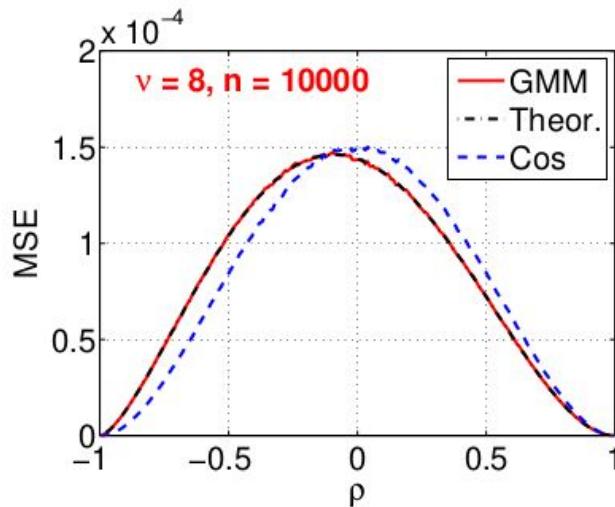
Theory of the GMM Kernel

- When data have bounded $\nu = 4.5$ -th moment, the cosine estimates exhibit bounded MSE (variance), which is substantially larger than the MSE of the GMM estimates.
- Therefore, the GMM estimates will be substantially more accurate in usual realistic situations in where the data have some bounded moments with outliers.



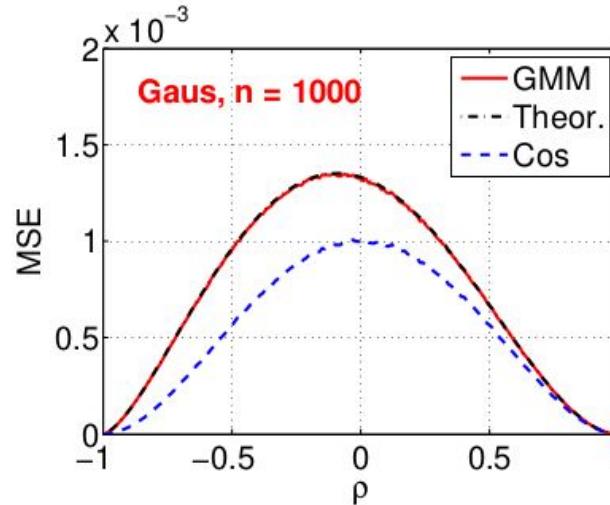
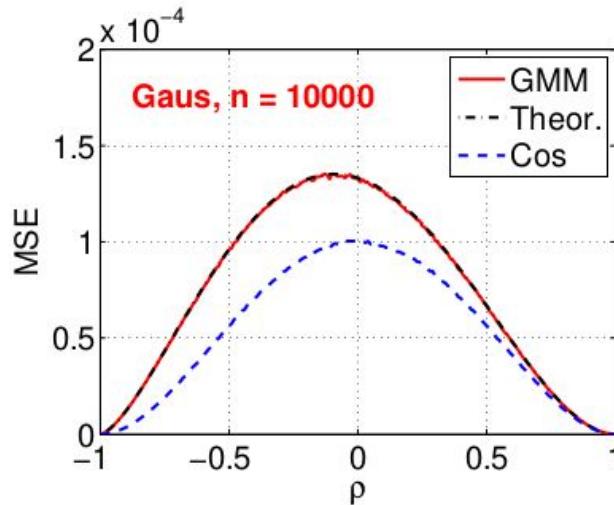
Theory of the GMM Kernel

- For data with bounded $v = 8$ -th moment, the cosine and GMM have similar accuracy.



Theory of the GMM Kernel

- When data are perfectly Gaussian (which is almost never the case in practice), using the GMM only slightly loses accuracy compared to using the cosine.
- Conclusion: practitioners can safely replace the cosine with GMM !



Warning: the GMM kernel is nonlinear (while cosine is linear) => need CWS hashing.

The Chi-Square Similarity

$$\rho_{\chi^2} = \sum_{i=1}^D \frac{2u_i v_i}{u_i + v_i}, \quad u_i \geq 0, \quad v_i \geq 0, \quad \sum_{i=1}^D u_i = 1, \quad \sum_{i=1}^D v_i = 1$$

Sign Cauchy Projections and Chi-Square Kernel

Ping Li

Dept of Statistics & Biostat.
Dept of Computer Science
Rutgers University
pingli@stat.rutgers.edu

Gennady Samorodnitsky

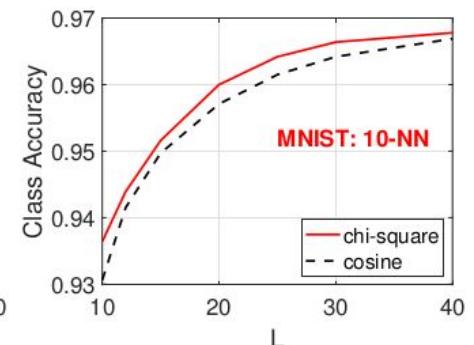
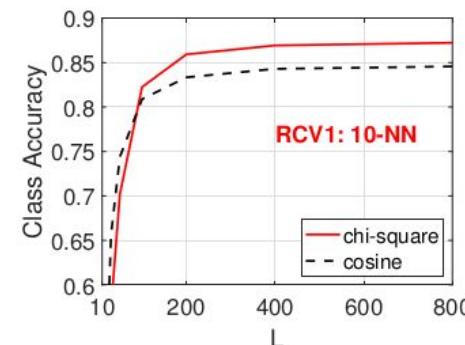
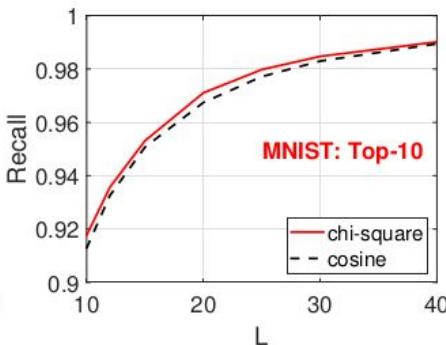
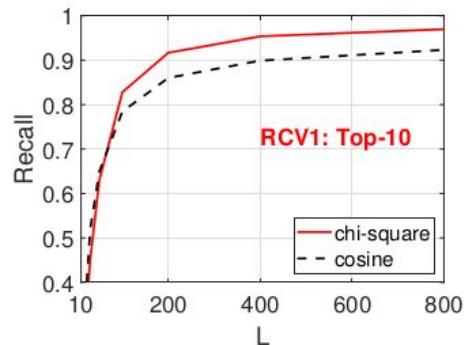
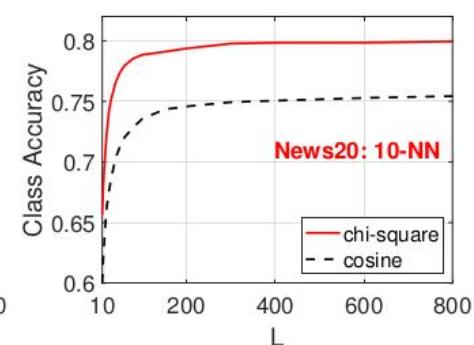
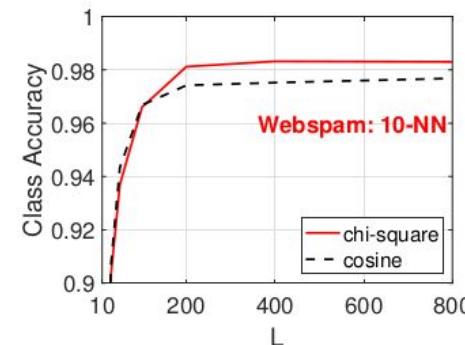
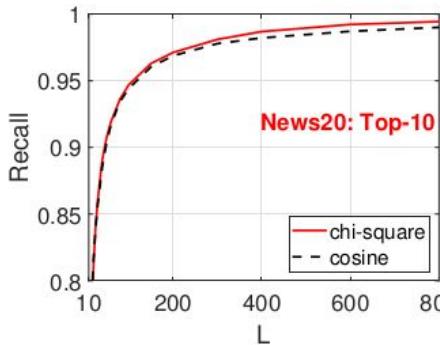
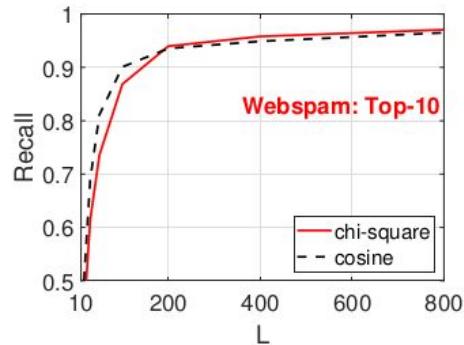
ORIE and Dept of Stat. Science
Cornell University
Ithaca, NY 14853
gs18@cornell.edu

John Hopcroft

Dept of Computer Science
Cornell University
Ithaca, NY 14853
jeh@cs.cornell.edu

The Chi-Square Similarity

[arXiv 2023](#). HNSW + chi-square similarity achieves better retrieval and KNN classification on public data.



Outline

1. Vector similarity functions
- 2. Vector compressions**
3. Vector similarity search
4. Maximum inner product search (MIPS)
5. Fast neural ranking
6. GPU computing
7. GCWSNet, hashing algorithms
8. Boosted trees, ABC-boost
9. Distributed, adaptive, and federated learning
10. Privacy
11. Security
12. Others: generative AI models, NLP, knowledge graphs, multi-modal, cross-modal, advertising

Benefits of Vector Compressions

- Save storage (memory) space
 - Long vectors => shorter vectors
 - Real value entries => integer or 1-bit entries
 - Examples: [OPORP](#), [sign-full random projections](#), and CWS
- Provide indexing
 - Sign Gaussian random projections
 - Sign Cauchy random projections
 - Sign Stable random projections
 - Sign random Fourier features ([SignRFF](#))
 - b-bit Minwise hashing
 - b-bit consistent weighted sampling (CWS)
- Provide privacy
- Improve re-ranking efficiency
- Reduce dimension of big models

One Permutation + One Random Projection

KDD 2023, [OPORP: One Permutation + One Random Projections](#)

- OPORP is a variant of count-sketch with fixed-length binning and normalization.
- The estimation variance is substantially reduced due to binning scheme and normalization.
- The analysis of the proposed OPORP leads to the normalized estimator and its variance, for “very sparse random projections” (VSRP, [KDD 2006](#)).
- Further developments:
 - OPORP + differential privacy (DP), e.g., [arXiv 2023](#).
 - OPORP + quantization, e.g., sign-full random projections ([AAAI 2019](#)).
 - OPORP + big models, i.e., OPORP samples to replace or augment original features.

An Illustration of the OPORP Procedure

KDD 2023

Original vector	6	11	0		4	3	8		7	1	9
Permuted vector	1	7	4		8	9	0		3	11	6
Random vector	-1	-1	+1		+1	-1	+1		-1	+1	-1
Aggregation			-4				-1				2
Normalized			-0.8729				-0.2182				0.4364

$$-1-7+4 = -4,$$

$$+8-9+0 = -1,$$

$$-3+11-6 = 2$$

$$(-4)^2 + (-1)^2 + (2)^2 = 16 + 1 + 4 = 21 \quad -4/\text{square root}(21) = -0.8729$$

$$(-0.8729)^2 + (-0.2182)^2 + (0.4364)^2 = 1.0$$

OPORP Procedure

KDD 2023

- Generate a permutation $\pi : [D] \longrightarrow [D]$.
- Apply the same permutation to all data vectors, e.g., $u, v \in \mathbb{R}^D$.
- Generate a random vector r of size D , with i.i.d. entries r_i of the following first four moments:

$$E(r_i) = 0, \quad E(r_i^2) = 1, \quad E(r_i^3) = 0, \quad E(r_i^4) = s$$

Our calculation will show that $s = 1$ leads to the smallest variance of OPORP.

- Divide the D columns into k bins. There are two binning strategies:
 1. Fixed-length binning scheme: every bin has a length of D/k .
 2. Variable-length binning scheme: the bin lengths follow a multinomial distribution

OPОРР Samples

KDD 2023

$$x_j = \sum_{i=1}^D u_i r_i I_{ij}, \quad y_j = \sum_{i=1}^D v_i r_i I_{ij}, \quad j = 1, 2, \dots, k$$

where I_{ij} is a random variable determined by one of the following two binning schemes:

1. (*First binning scheme*) Fixed-length binning scheme: every bin has a length of D/k . We assume that D is divisible by k , if not, we can always pad zeros.
2. (*Second binning scheme*) Variable-length binning scheme: the bin lengths follow a multinomial distribution $\text{multinomial}(D, 1/k, 1/k, \dots, 1/k)$ with k bins.

Specifically, $I_{ij} = 1$ if the original coordinate $i \in [1, D]$ is mapped to bin $j \in [1, k]$; $I_{ij} = 0$ otherwise. Wherever necessary, we will use $I_{1,ij}$ and $I_{2,ij}$ to differentiate the two binning schemes.

The Un-normalized Estimator

KDD 2023

Given the OPORP samples:

$$x_j = \sum_{i=1}^D u_i r_i I_{ij}, \quad y_j = \sum_{i=1}^D v_i r_i I_{ij}, \quad j = 1, 2, \dots, k$$

we hope to estimate the inner product and the cosine:

$$\text{inner product: } a = \sum_{i=1}^D u_i v_i \qquad \text{cosine: } \rho = \frac{\sum_{i=1}^D u_i v_i}{\sqrt{\sum_{i=1}^D u_i^2} \sqrt{\sum_{i=1}^D v_i^2}}$$

$\hat{a} = \sum_{j=1}^k x_j y_j$ the un-normalized estimator, ie., the inner product of the samples

The Un-normalized Estimator

KDD 2023

THEOREM 3.2.

$$\hat{a} = \sum_{j=1}^k x_j y_j$$

$$E(\hat{a}) = a,$$

$$Var(\hat{a}_1) = (s - 1) \sum_{i=1}^D u_i^2 v_i^2 + \frac{1}{k} \left(a^2 + \sum_{i=1}^D u_i^2 \sum_{i=1}^D v_i^2 - 2 \sum_{i=1}^D u_i^2 v_i^2 \right) \frac{D - k}{D - 1},$$

$$Var(\hat{a}_2) = (s - 1) \sum_{i=1}^D u_i^2 v_i^2 + \frac{1}{k} \left(a^2 + \sum_{i=1}^D u_i^2 \sum_{i=1}^D v_i^2 - 2 \sum_{i=1}^D u_i^2 v_i^2 \right).$$

\hat{a}_1 and \hat{a}_2 for two binning schemes, respectively.

We must use $s = 1$ for OPORP

$\frac{D - k}{D - 1}$ variance reduction factor due to the fixed-length binning scheme (the 1st scheme)

The Un-normalized Estimator

KDD 2023

What will happen if we repeat OPORP m times?

$$Var(\hat{a}_1; m \text{ repetitions}) = \frac{1}{m} \left[(s - 1) \sum_{i=1}^D u_i^2 v_i^2 + \frac{1}{k} \left(a^2 + \sum_{i=1}^D u_i^2 \sum_{i=1}^D v_i^2 - 2 \sum_{i=1}^D u_i^2 v_i^2 \right) \frac{D - k}{D - 1} \right],$$

$$Var(\hat{a}_2; m \text{ repetitions}) = \frac{1}{m} \left[(s - 1) \sum_{i=1}^D u_i^2 v_i^2 + \frac{1}{k} \left(a^2 + \sum_{i=1}^D u_i^2 \sum_{i=1}^D v_i^2 - 2 \sum_{i=1}^D u_i^2 v_i^2 \right) \right].$$

$$Var(\hat{a}; m \text{ repetitions and } k = 1) = \frac{1}{m} \left(a^2 + \sum_{i=1}^D u_i^2 \sum_{i=1}^D v_i^2 + (s - 3) \sum_{i=1}^D u_i^2 v_i^2 \right)$$

This is exactly the variance formula for “**very sparse random projections**” in KDD 2006.

Very Sparse Random Projections (VSRP)

KDD 2006

Very Sparse Random Projections

Ping Li

Department of Statistics
Stanford University
Stanford CA 94305, USA
pingli@stat.stanford.edu

Trevor J. Hastie

Department of Statistics
Stanford University
Stanford CA 94305, USA
hastie@stanford.edu

Kenneth W. Church

Microsoft Research
Microsoft Corporation
Redmond WA 98052, USA
church@microsoft.com

VSRP uses a very

sparse projection matrix

$$x_j = \sum_{i=1}^D u_i r_{ij}, \quad y_j = \sum_{i=1}^D v_i r_{ij}, \quad j = 1, 2, \dots, k.$$

$$r_{ij} = \sqrt{s} \times \begin{cases} -1 & \text{with prob. } 1/(2s) \\ 0 & \text{with prob. } 1 - 1/s \\ +1 & \text{with prob. } 1/(2s) \end{cases}$$

$$\hat{a}_{vsvp} = \frac{1}{k} \sum_{j=1}^k x_j y_j, \quad \hat{\rho}_{vsvp} = \frac{\sum_{j=1}^k x_j y_j}{\sqrt{\sum_{j=1}^k x_j^2} \sqrt{\sum_{j=1}^k y_j^2}},$$

The variance of the un-normalized estimator of inner product is in KDD 2006.

$$Var(\hat{a}_{vsvp}) = \frac{1}{k} \left(a^2 + \sum_{i=1}^D u_i^2 \sum_{i=1}^D v_i^2 + (s-3) \sum_{i=1}^D u_i^2 v_i^2 \right)$$

The variance of the normalized estimator of VSRP can be shown only in this paper.

The Normalized Estimator of OPORP

KDD 2023

$$\hat{\rho} = \frac{\sum_{j=1}^k x_j y_j}{\sqrt{\sum_{j=1}^k x_j^2} \sqrt{\sum_{j=1}^k y_j^2}}$$

The variance of the normalized estimator is substantially reduced

THEOREM 3.4. For large k , $\hat{\rho}$ converges to ρ , almost surely, with

$$Var(\hat{\rho}_1) = (s - 1)A + \left\{ \frac{1}{k} [(1 - \rho^2)^2 - 2A] + O\left(\frac{1}{k^2}\right) \right\} \frac{D - k}{D - 1},$$

$$Var(\hat{\rho}_2) = (s - 1)A + \left\{ \frac{1}{k} [(1 - \rho^2)^2 - 2A] + O\left(\frac{1}{k^2}\right) \right\}.$$

where

$$A = \sum_{i=1}^D \left(u'_i v'_i - \rho/2(u'_i{}^2 + v'_i{}^2) \right)^2, u'_i = \frac{u_i}{\sqrt{\sum_{t=1}^D u_t^2}}, v'_i = \frac{v_i}{\sqrt{\sum_{t=1}^D v_t^2}}.$$

The Normalized Estimator of VSRP

KDD 2023

Just like OPORP, the variance of the normalized estimator of VSRP is substantially reduced

THEOREM 3.5. *As $k \rightarrow \infty$, $\hat{\rho}_{vsrp} \rightarrow \rho$ almost surely, with*

$$Var(\hat{\rho}_{vsrp}) = \frac{1}{k} \left((1 - \rho^2)^2 + (s - 3)A \right) + O\left(\frac{1}{k^2}\right),$$

where

$$A = \sum_{i=1}^D \left(u'_i v'_i - \rho/2(u'^2_i + v'^2_i) \right)^2, u'_i = \frac{u_i}{\sqrt{\sum_{t=1}^D u_t^2}}, v'_i = \frac{v_i}{\sqrt{\sum_{t=1}^D v_t^2}}.$$

A Simulation Study of OPORP

KDD 2023

MSE = mean square errors

Blue = un-normalized

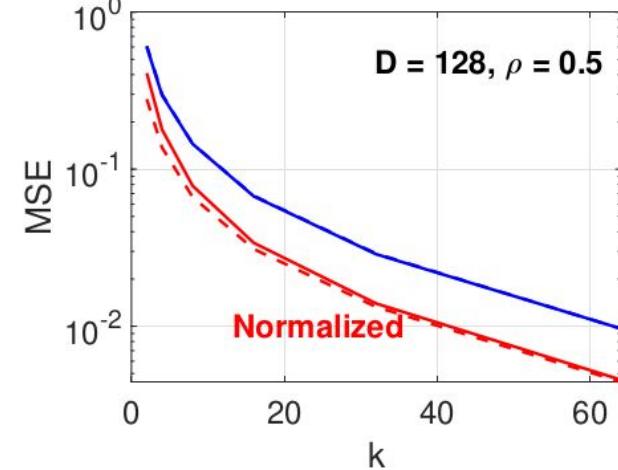
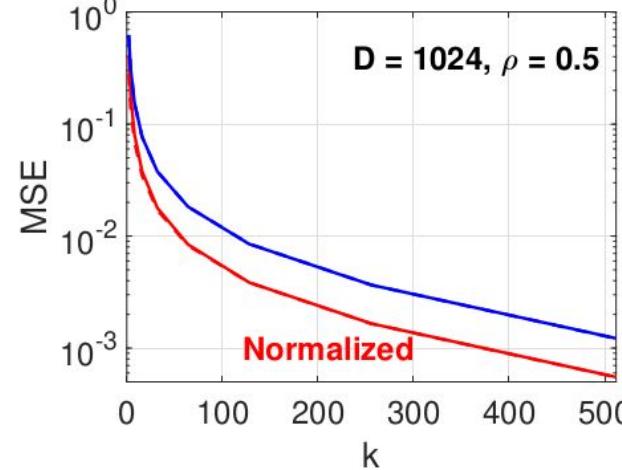
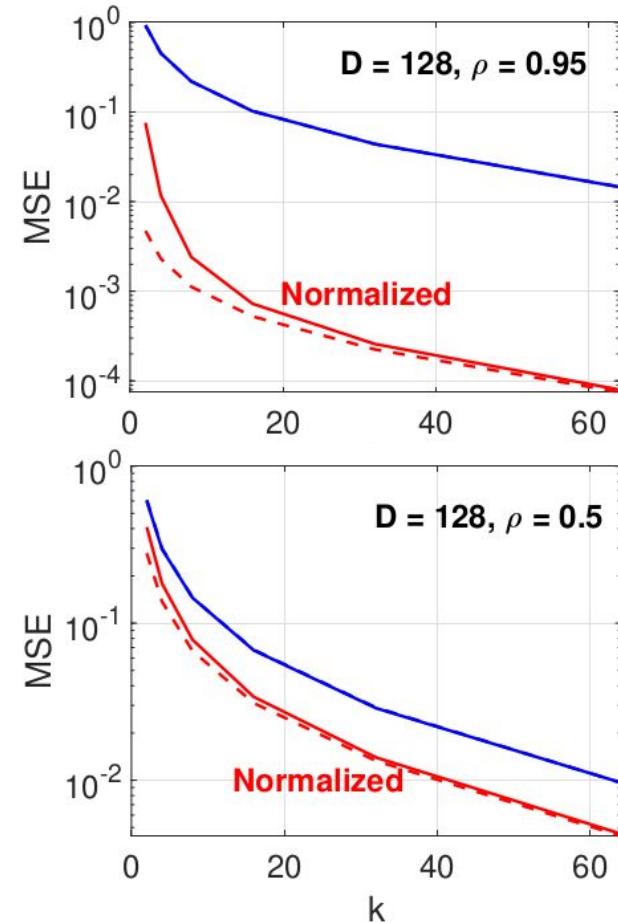
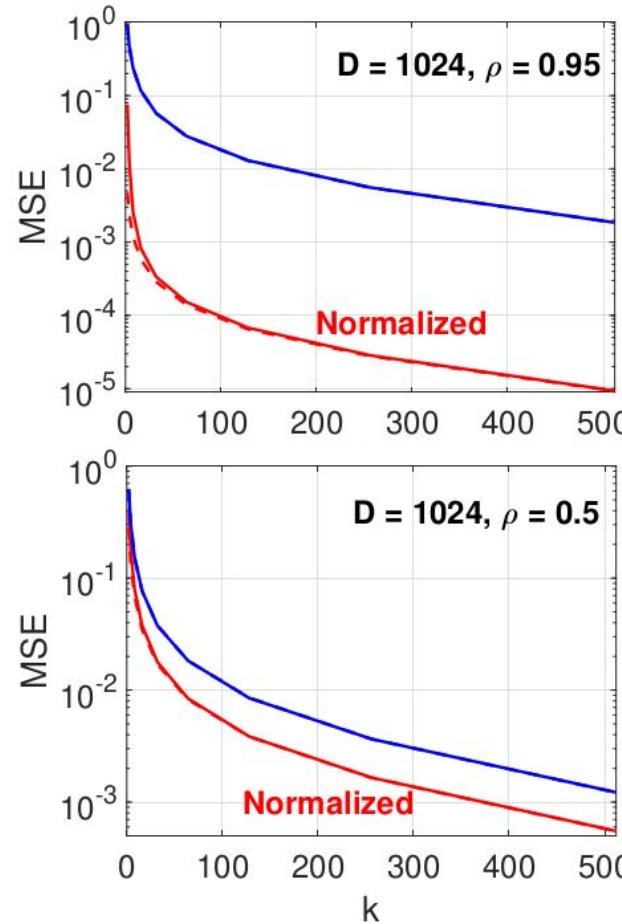
Red = normalized

Dashed = theoretical var

Normalization improves accuracy especially in high-similarity region.

Theory matches empirical

Original data vectors are normalized so that inner product and cosine estimators are the same.



The Inner Product Estimators

KDD 2023

If the original data are not normalized, we have at least two estimators for inner product:

$$\hat{a} = \sum_{j=1}^k x_j y_j$$

$$\hat{a}_n = \hat{\rho} \sqrt{\sum_{i=1}^D u_i^2} \sqrt{\sum_{i=1}^D v_i^2} \quad \hat{\rho} = \frac{\sum_{j=1}^k x_j y_j}{\sqrt{\sum_{j=1}^k x_j^2} \sqrt{\sum_{j=1}^k y_j^2}}$$

Improving Random Projections Using Marginal Information

We can also approximately use the cubic equation for MLE solution (based on dense Gaussian random projections)

COLT 2006 Ping Li¹, Trevor J. Hastie¹, and Kenneth W. Church²

¹ Department of Statistics, Stanford University, Stanford CA 94305, USA,
`{pingli, hastie}@stat.stanford.edu`

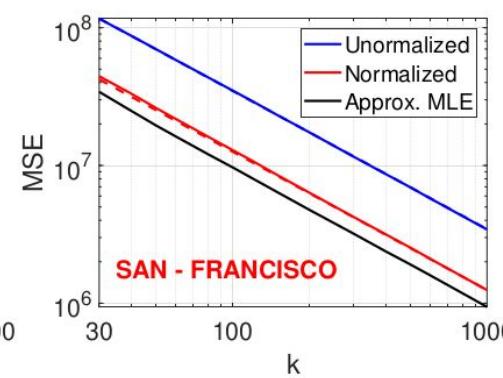
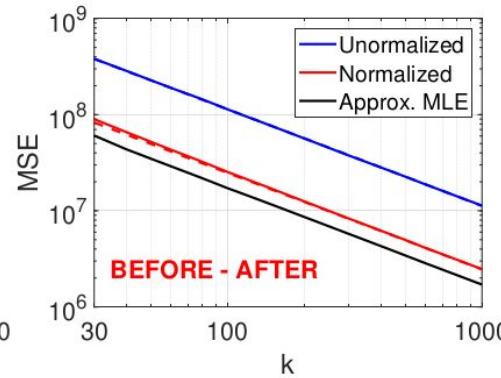
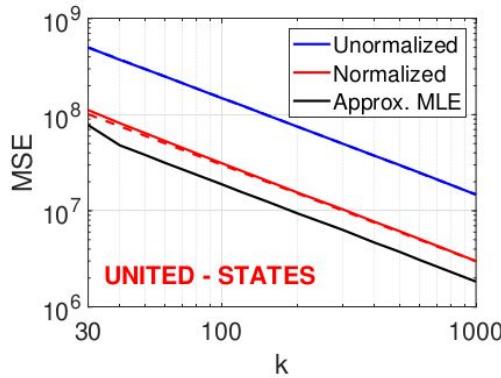
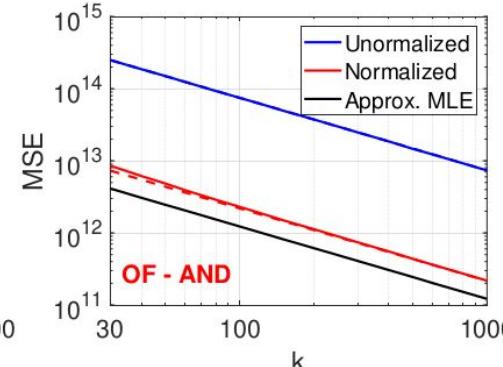
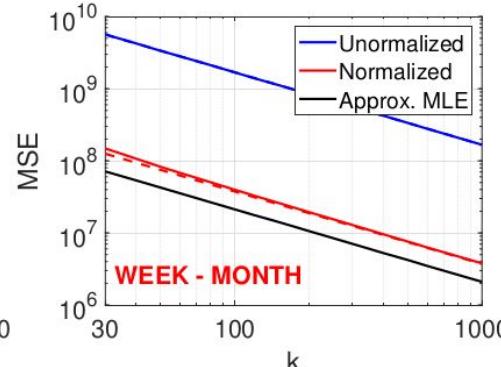
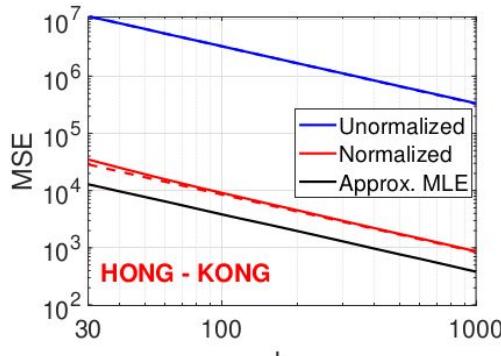
² Microsoft Research, One Microsoft Way, Redmond WA 98052, USA,

$$\hat{a}_m^3 - \hat{a}_m^2 \sum_{i=1}^k x_i y_i + \hat{a}_m \left(- \sum_{i=1}^D u_i^2 \sum_{i=1}^D v_i^2 + \sum_{i=1}^D u_i^2 \sum_{j=1}^k y_j^2 + \sum_{i=1}^D v_i^2 \sum_{j=1}^k x_j^2 \right) - \sum_{i=1}^D u_i^2 \sum_{i=1}^D v_i^2 \sum_{j=1}^k x_j y_j = 0$$

Experiments for Estimating Inner Products

KDD 2023

Three estimators for estimating the inner product between a pair of word vectors, using the “Word dataset” in [EMNLP 2005](#)



The “Word” Dataset

<https://github.com/pltrees/Smallest-K-Sketch>

This dataset contains several thousand of word vectors. Each vector records the number of word occurrences in $2^{16} = 65,536$ documents. This dataset was initially used in 2005 for estimating word associations using “**smallest-K-sketch**” which was later developed into “**Conditional Random Sampling**” (CRS).

- Ping Li' PhD Thesis. [Stable random projections and conditional random sampling, two sampling techniques for modern massive datasets](#)). Department of Statistics, Stanford University, 2007.
- Ping Li, Kenneth Church, Trevor Hastie. [One Sketch For All: Theory and Application of Conditional Random Sampling](#). NIPS 2008.
- Ping Li, Kenneth Church. [A Sketch Algorithm for Estimating Two-Way and Multi-Way Associations](#). Computational Linguistics 2007.
- Ping Li, Kenneth Church, Trevor Hastie. [Conditional Random Sampling: A Sketch-based Sampling Technique for Sparse Data](#). NIPS 2006.
- Ping Li, Kenneth Church. [Using Sketches to Estimate Associations](#). EMNLP 2005.

Retrieval Experiments for OPORP and VSRP

KDD 2023

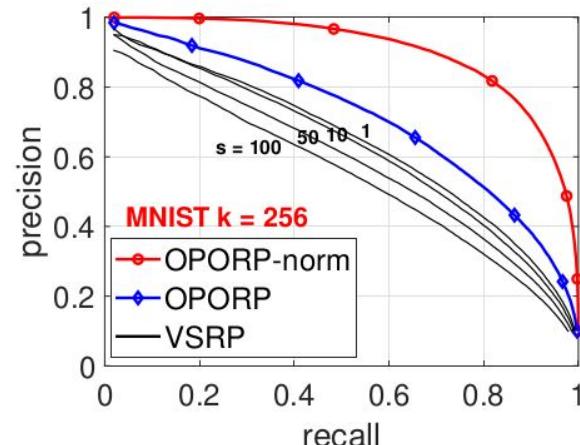
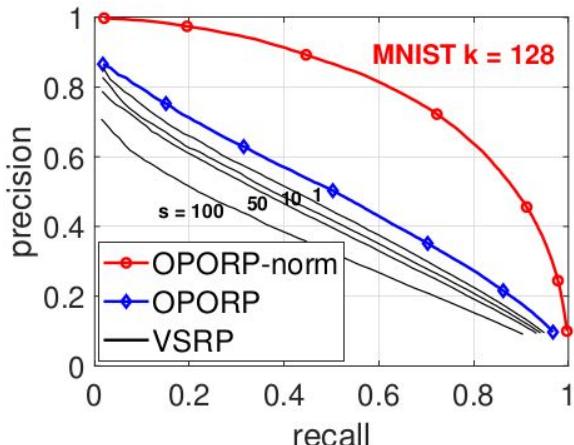
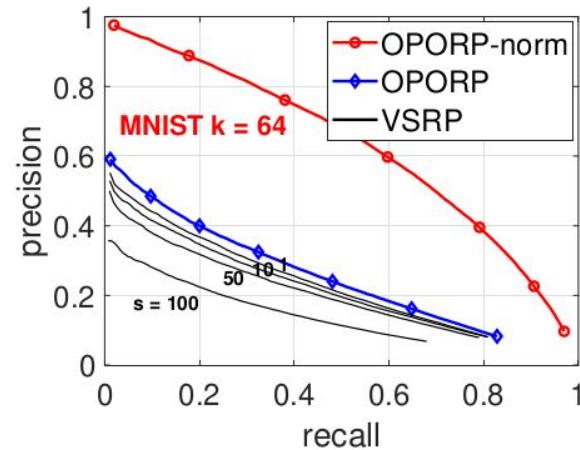
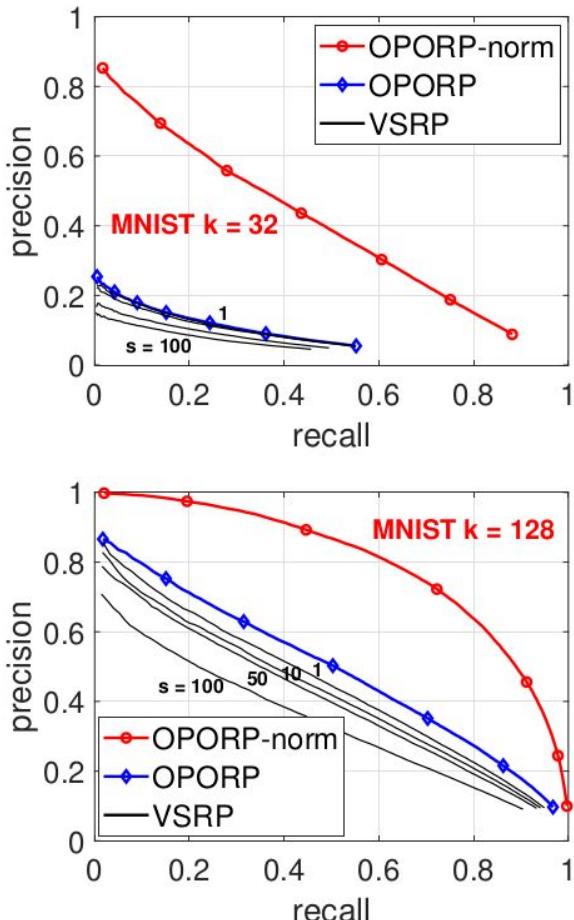
Comparing three algorithms:

- 1 OPORP (un-normalized)
- 2 OPORP-norm
- 3 VSRP

Normalization substantially improves the accuracy.

Without normalization, both VSRP and OPORP have very similar accuracy when $s=1$ (parameter in VSRP).

VSRP performs poorly using large s value (very sparse).



Retrieval Experiments for OPORP and VSRP

KDD 2023

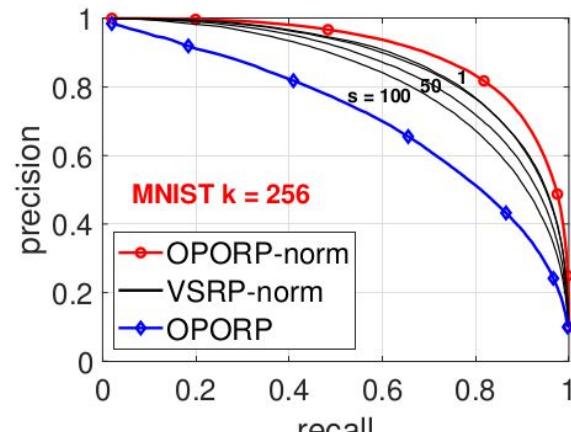
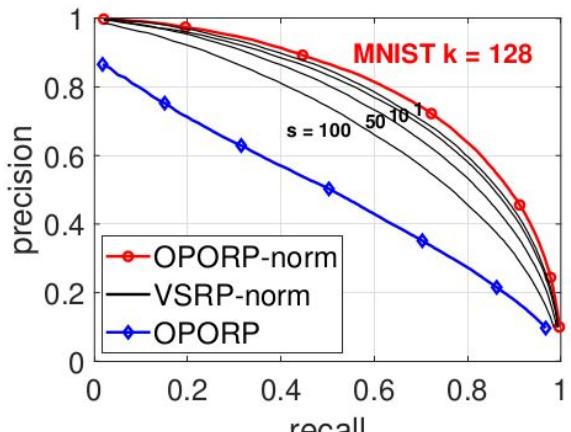
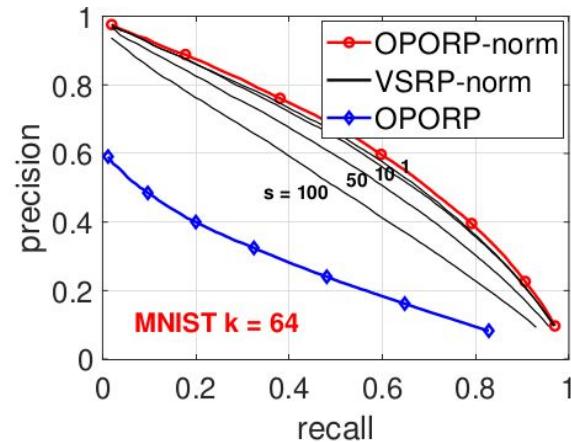
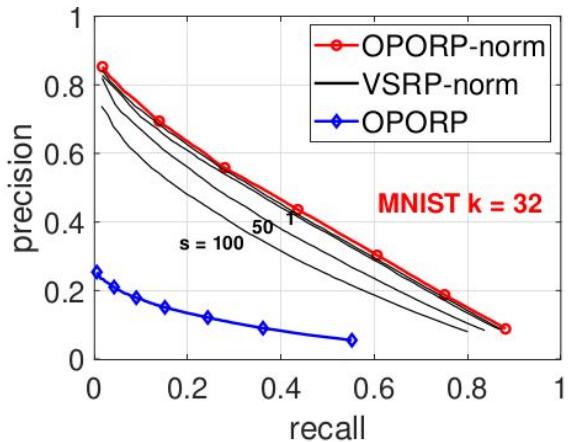
Comparing three algorithms:

- 1 OPORP-norm
- 2 VSRP (un-normalized)
- 3 VSRP-norm

Normalization substantially improves the accuracy.

With normalization, both VSRP and OPORP have very similar accuracy when $s=1$ (parameter in VSRP).

VSRP degrades when using large s value (very sparse).



KNN Classification for OPORP and VSRP

KDD 2023

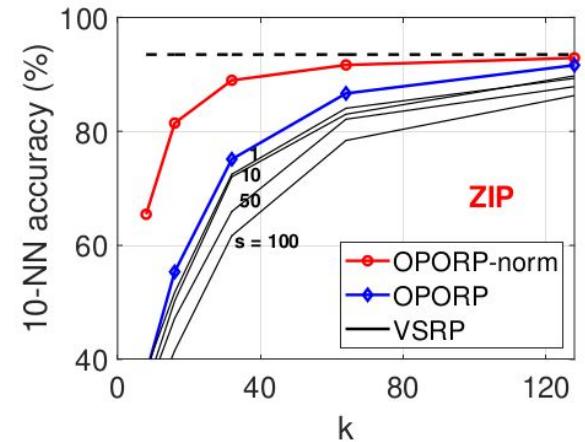
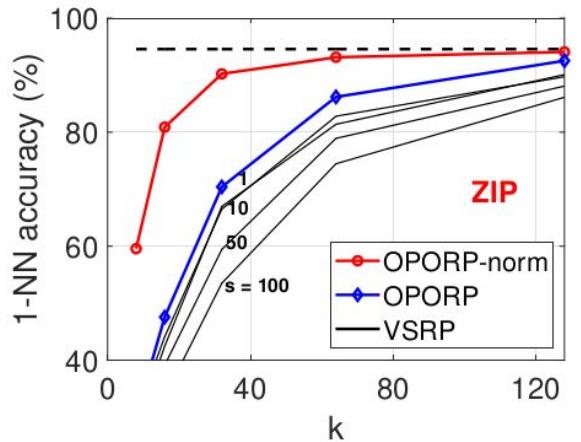
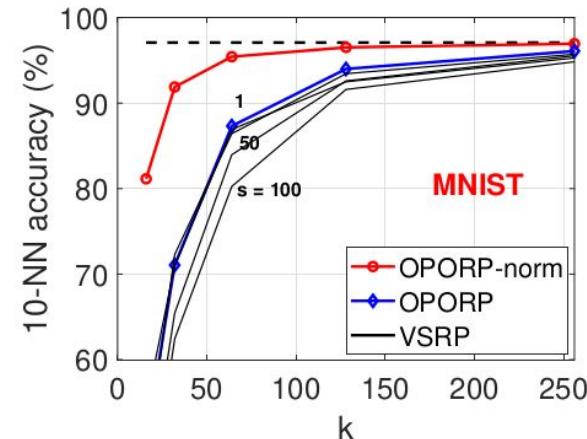
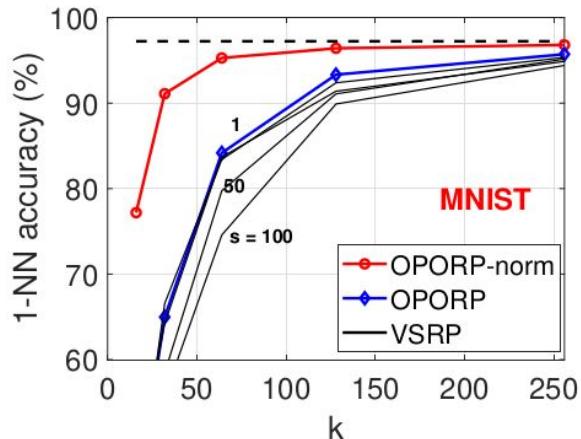
Comparing three algorithms:

- 1 OPORP (un-normalized)
- 2 OPORP-norm
- 3 VSRP

Normalization substantially improves the accuracy.

Without normalization, both VSRP and OPORP have very similar accuracy when $s=1$ (parameter in VSRP).

VSRP performs poorly using large s value (very sparse).



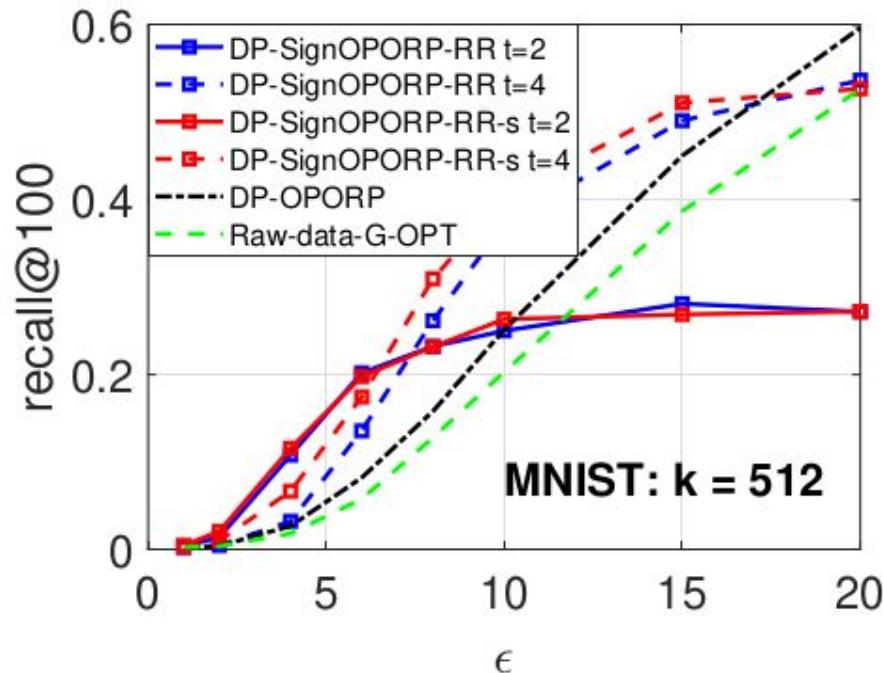
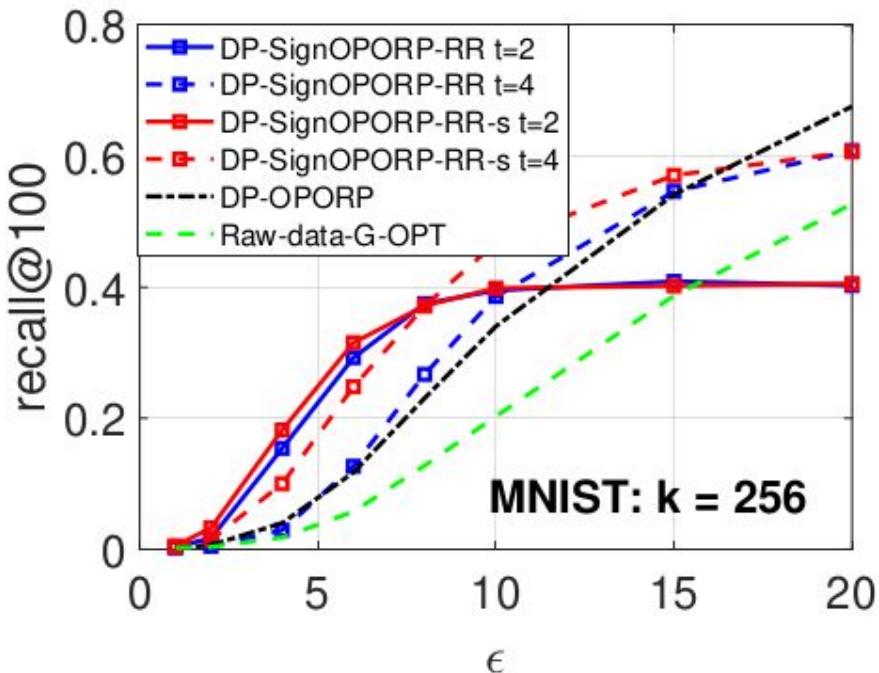
Summary of OPORP

KDD 2023

- Count-sketch with 1. fixed-length binning, and 2. normalization for the estimation.
- The fixed-length binning reduces the variance by $(D-k)/(D-1)$, which is substantial for relatively short embeddings. The fixed-length binning is also more convenient.
- The normalization step reduces the variance from $(1+\rho^2)$ to $(1-\rho^2)^2$, which is drastic especially in high-similarity region when $\rho \rightarrow 1$, e.g., duplicate detection.
- A side result is that we also develop the normalized estimator and provide its variance, for “very sparse random projections” (VSRP, [KDD 2006](#)).
- Directions for further developments:
 - OPORP + differential privacy (DP), e.g., [arXiv 2023](#).
 - OPORP + quantization, e.g., sign-full random projections ([AAAI 2019](#)).
 - OPORP + big models, i.e., OPORP samples to replace or augment original features.

Differential Privacy (DP) with OPORP

- Differential Privacy with Random Projections and Sign Random Projections. [pdf](#)
- Differentially Private One Permutation Hashing and Bin-wise Consistent Weighted Sampling. [pdf](#)

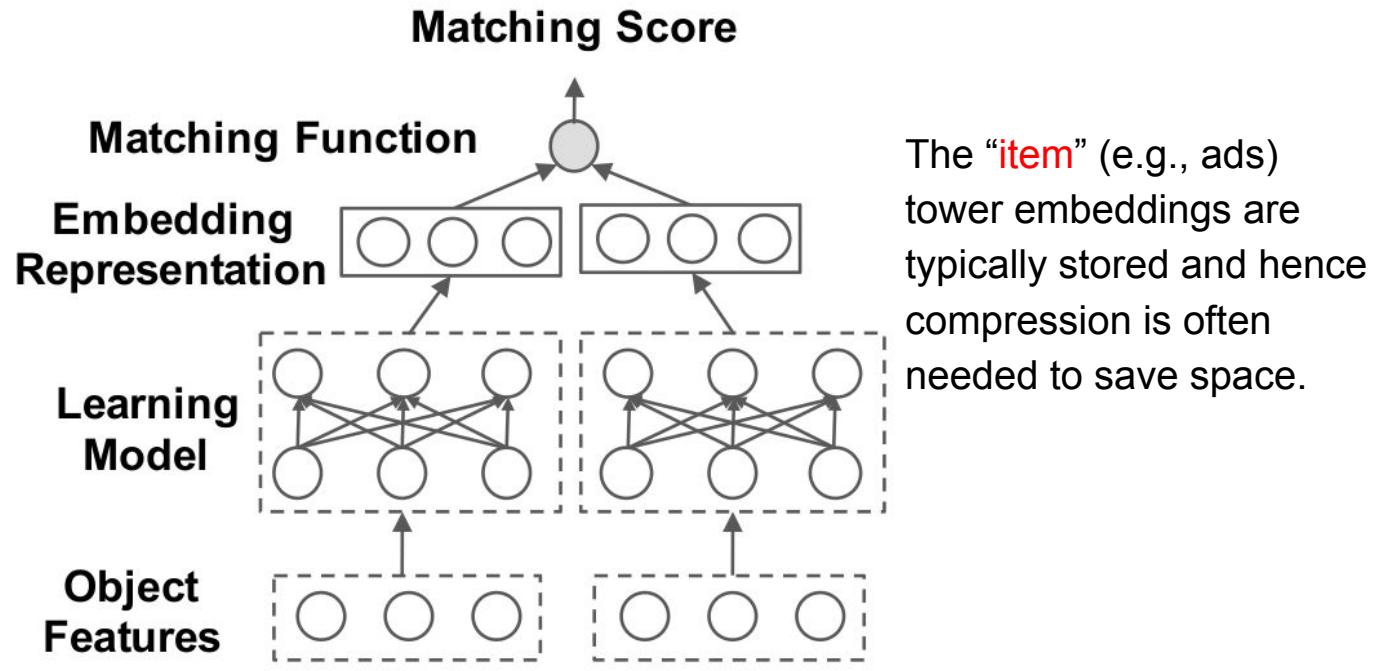


Sign-Full Random Projections

AAAI 2019

Vector compressions via “sign-full random projections” is particularly natural for two-tower models.

The “**query**” tower embeddings are typically generated on demand when a query arrives. It is hence natural to use the “full” precision.



Sign-Full Random Projections

AAAI 2019

- The original theory of sign-full random projections focuses on the standard (dense) Gaussian random projections, although in practice, we can allow sparse projections including OPORP.
- The theory is also built on top of a series of prior research on quantized random projections, by Goemans, Williamson, Charikar, etc., as well as our own prior works on random projections.
- “Sign-Full” means the “query” vectors are not quantized (after random projections) while the “item” vectors are quantized to be 1-bit (i.e., the sign after random projections). It is only a special instance of a more general “asymmetric quantized random projections”; see NeurIPS 2019.

Sign-Full Random Projections

AAAI 2019



$$u \in \mathbb{R}^D$$



$$v \in \mathbb{R}^D$$

$$x_j = \sum_{i=1}^D u_i r_{ij}, \quad y_j = \sum_{i=1}^D v_i r_{ij}, \quad j = 1, 2, \dots, k, \quad r_{ij} \sim N(0, 1)$$

Using a random projection matrix of size $D \times k$, sampled from Gaussian distribution, the projected data $\begin{bmatrix} x_j \\ y_j \end{bmatrix}$ follow the standard bivariate Gaussian distribution. This is un-normalized estimator $\hat{\rho}_f$

$$\begin{bmatrix} x_j \\ y_j \end{bmatrix} \sim N \left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix} \right)$$

$$\hat{\rho}_f = \frac{1}{k} \sum_{j=1}^k x_j y_j, \quad E(\hat{\rho}_f) = \rho$$

$$Var(\hat{\rho}_f) = \frac{V_f}{k}, \quad V_f = 1 + \rho^2$$

Sign-Full Random Projections

AAAI 2019

$$\hat{\rho}_{f,n} = \frac{\sum_{j=1}^k x_j y_j}{\sqrt{\sum_{j=1}^k x_j^2} \sqrt{\sum_{j=1}^k y_j^2}}, \quad E(\hat{\rho}_{f,n}) = \rho + O\left(\frac{1}{k}\right)$$

$$Var(\hat{\rho}_{f,n}) = \frac{V_{f,n}}{k} + O\left(\frac{1}{k^2}\right), \quad V_{f,n} = (1 - \rho^2)^2$$

The normalized estimator and variance

$$\rho^3 - \rho^2 \sum_{j=1}^k x_j y_j + \rho \left(-1 + \sum_{i=1}^k x_i^2 + \sum_{j=1}^k y_j^2 \right) - \sum_{j=1}^k x_j y_j = 0$$

The maximum likelihood estimator (MLE) by solving a cubic equation and the corresponding variance.

$$E(\hat{\rho}_{f,m}) = \rho + O\left(\frac{1}{k}\right)$$

$$Var(\hat{\rho}_{f,m}) = \frac{V_{f,m}}{k} + O\left(\frac{1}{k^2}\right), \quad V_{f,m} = \frac{(1 - \rho^2)^2}{1 + \rho^2}$$

Sign-Full Random Projections

AAAI 2019

$$Pr(sgn(x_j) = sgn(y_j)) = 1 - \frac{1}{\pi} \cos^{-1} \rho,$$

Based on the well-known probability, one can use the “sign-sign” estimator:

$$\hat{\rho}_1 = \cos \pi \left(1 - \frac{1}{k} \sum_{j=1}^k 1_{sgn(x_j)=sgn(y_j)} \right),$$

$$E(\hat{\rho}_1) = \rho + O\left(\frac{1}{k}\right), \quad Var(\hat{\rho}_1) = \frac{V_1}{k} + O\left(\frac{1}{k^2}\right)$$

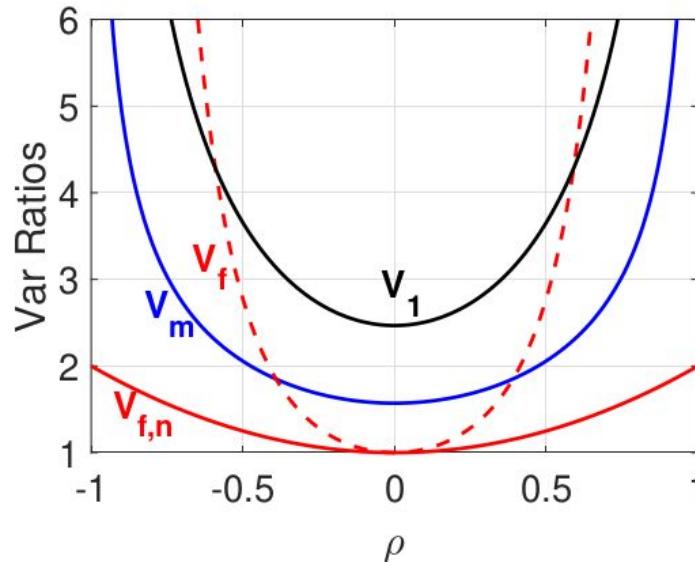
$$V_1 = \cos^{-1} \rho (\pi - \cos^{-1} \rho) (1 - \rho^2)$$

Sign-Full Random Projections

AAAI 2019

The MLE estimator is the most accurate. The ratios of the variances show that the “sign-sign” estimator (V_1) can be substantially improved.

The accuracy of the “sign-full” estimator (V_m) is substantially better than the “sign-sign” (V_1) estimator.



The difference between blue (V_m) and black (V_1) is the amount of improvement we can hope for with “sign-full”.

Ratios of variance factors: $\frac{V_1}{V_{f,m}}$, $\frac{V_f}{V_{f,m}}$, $\frac{V_m}{V_{f,m}}$, $\frac{V_{f,n}}{V_{f,m}}$. Because $V_{f,m}$ is the theoretically smallest variance factor, the ratios are always larger than 1, and we can use them to compare estimators (lower the better). Note that V_m is the variance factor for the MLE of sign-full random projections (see Section 2).

Sign-Full Random Projections

AAAI 2019

The “sign-full” samples $(sign(x_j), y_j), j = 1, 2, \dots, k$

The MLE estimator for sign-full projections

$$\sum_{j=1}^k \frac{\phi\left(\frac{\rho}{\sqrt{1-\rho^2}} sgn(x_j)y_j\right)}{\Phi\left(\frac{\rho}{\sqrt{1-\rho^2}} sgn(x_j)y_j\right)} sgn(x_j)y_j = 0$$

The variance of the MLE estimator for sign-full projections

$$E(\hat{\rho}_m) = \rho + O\left(\frac{1}{k}\right)$$

$$Var(\hat{\rho}_m) = \frac{V_m}{k} + O\left(\frac{1}{k^2}\right)$$

$$\begin{aligned} \frac{1}{V_m} &= E \left\{ \frac{\rho}{(1-\rho^2)^{7/2}} \frac{\phi\left(\frac{\rho}{\sqrt{1-\rho^2}} sgn(x_j)y_j\right)}{\Phi\left(\frac{\rho}{\sqrt{1-\rho^2}} sgn(x_j)y_j\right)} sgn(x_j)y_j^3 \right\} \\ &\quad + E \left\{ \frac{1}{(1-\rho^2)^3} \frac{\phi^2\left(\frac{\rho}{\sqrt{1-\rho^2}} sgn(x_j)y_j\right)}{\Phi^2\left(\frac{\rho}{\sqrt{1-\rho^2}} sgn(x_j)y_j\right)} y_j^2 \right\} \\ &\quad - E \left\{ \frac{3\rho}{(1-\rho^2)^{5/2}} \frac{\phi\left(\frac{\rho}{\sqrt{1-\rho^2}} sgn(x_j)y_j\right)}{\Phi\left(\frac{\rho}{\sqrt{1-\rho^2}} sgn(x_j)y_j\right)} sgn(x_j)y_j \right\} \end{aligned}$$

Sign-Full Random Projections

AAAI 2019

The “sign-full” samples $(\text{sign}(x_j), y_j), j = 1, 2, \dots, k$

Based on this probability result,
we develop the basic “sign-full”
estimator and its variance.

$$E(\text{sgn}(x_j)y_j) = \sqrt{\frac{2}{\pi}}\rho$$

$$\hat{\rho}_g = \frac{1}{k} \sum_{j=1}^k \sqrt{\frac{\pi}{2}} \text{sgn}(x_j)y_j, \quad E(\hat{\rho}_g) = \rho$$

Again, this estimator can be
improved by a normalization step.

$$\text{Var}(\hat{\rho}_g) = \frac{V_g}{k}, \quad V_g = \frac{\pi}{2} - \rho^2$$

Sign-Full Random Projections

AAAI 2019

The normalization step substantially reduces the estimation variance.

$$V_{g,n} \leq V_g$$

$$\hat{\rho}_{g,n} = \sqrt{\frac{\pi}{2}} \left(\frac{\sum_{j=1}^k sgn(x_j)y_j}{\sqrt{k}\sqrt{\sum_{j=1}^k y_j^2}} \right)$$

$$E(\hat{\rho}_{g,n}) = \rho + O\left(\frac{1}{k}\right)$$

$$Var(\hat{\rho}_{g,n}) = \frac{V_{g,n}}{k} + O\left(\frac{1}{k^2}\right)$$

$$V_{g,n} = V_g - \rho^2 (3/2 - \rho^2)$$

$$V_g = \frac{\pi}{2} - \rho^2$$

Sign-Full Random Projections

AAAI 2019

$$\hat{\rho}_s = 1 - \frac{\sqrt{2\pi}}{k} \sum_{j=1}^k [y_j - 1_{x_j \geq 0} + y_j + 1_{x_j < 0}]$$

$$E(\hat{\rho}_s) = \rho, \quad Var(\hat{\rho}_s) = \frac{V_s}{k}$$

$$V_s = 2\pi \left[1_{\rho < 0} + \frac{1}{\pi} \left(\tan^{-1} \left(\frac{\sqrt{1 - \rho^2}}{\rho} \right) - \rho \sqrt{1 - \rho^2} \right) - \frac{(1 - \rho)^2}{2\pi} \right]$$

Two other “sign-full” estimators

$$\hat{\rho}_{s,n} = 1 - \frac{\sum_{j=1}^k \sqrt{2\pi} [y_j - 1_{x_j \geq 0} + y_j + 1_{x_j < 0}]}{\sqrt{k} \sqrt{\sum_{j=1}^k y_j^2}}$$

$$E(\hat{\rho}_{s,n}) = \rho + O\left(\frac{1}{k}\right), \quad Var(\hat{\rho}_{s,n}) = \frac{V_{s,n}}{k} + O\left(\frac{1}{k^2}\right)$$

$$V_{s,n} = V_s - \frac{(1 - \rho)^2}{4\pi} (1 - 2\rho - 2\rho^2)$$

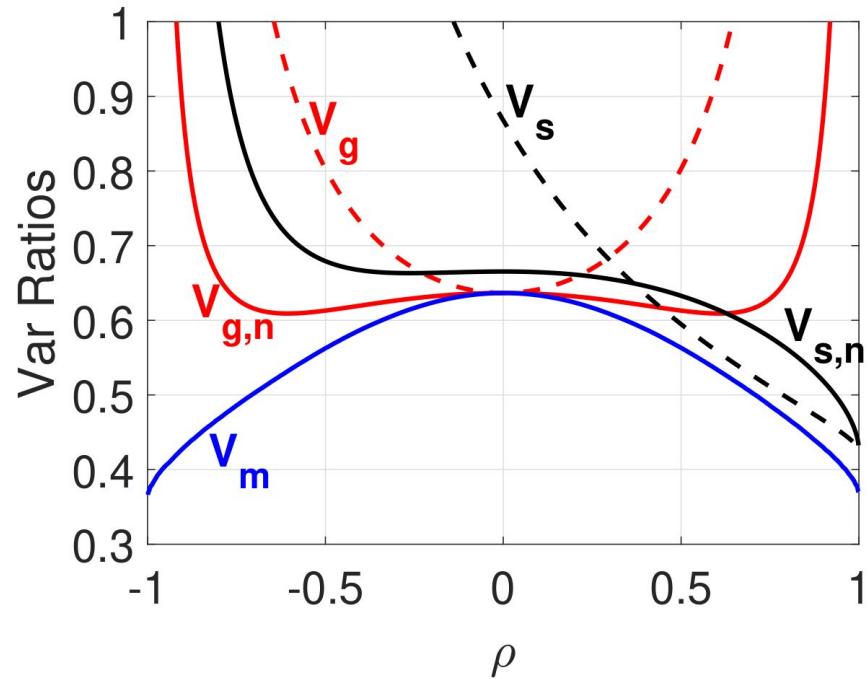
Sign-Full Random Projections

AAAI 2019

Use the “sign-sign” estimator (V_1) as the baseline, to compare the four estimators as well as the MLE (V_m), for “sign-full” random projections.

The MLE (V_m) is the most accurate, but MLE is computationally expensive and is not in a metric (inner product) form.

Overall, $\hat{\rho}_{s,n}$ is a very good estimator, for non-negative data



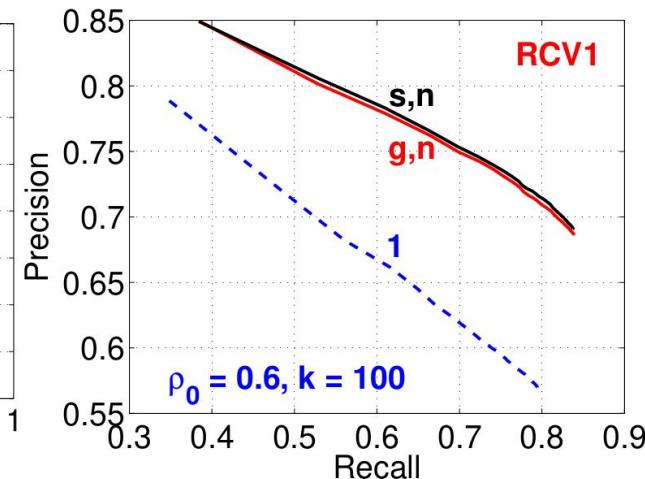
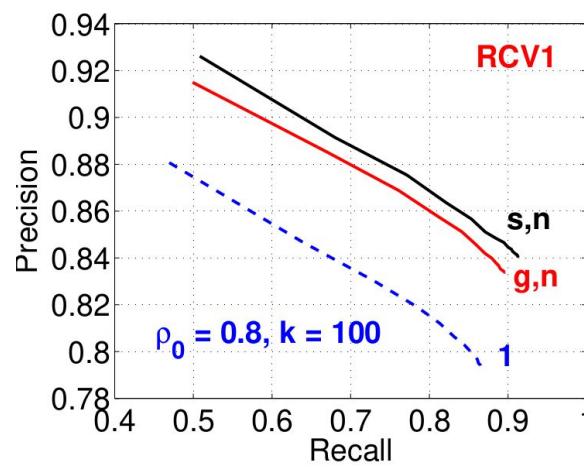
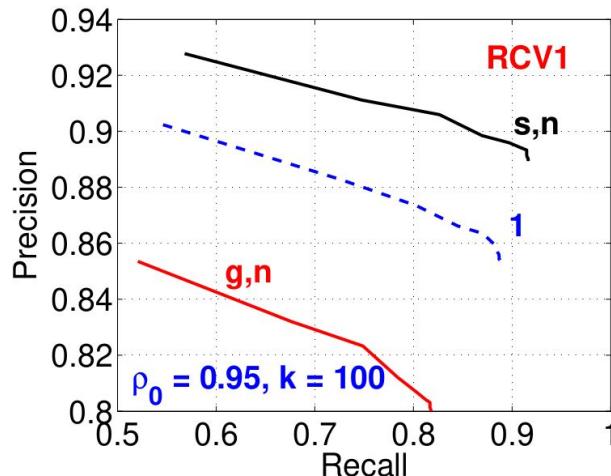
Variance factor ratios: $\frac{V_m}{V_1}$, $\frac{V_g}{V_1}$, $\frac{V_{g,n}}{V_1}$, $\frac{V_s}{V_1}$, $\frac{V_{s,n}}{V_1}$

Sign-Full Random Projections

AAAI 2019

Retrieval experiments on RCV1 dataset to compare the “sign-sign” estimator $\hat{\rho}_1$ with two “sign-full” estimators: $\hat{\rho}_{s,n}$, $\hat{\rho}_{g,n}$ for retrieving similar items exceeding a pre-specified threshold ρ_0 . $\hat{\rho}_{s,n}$ shows the best accuracy. $\hat{\rho}_1$ outperforms $\hat{\rho}_{g,n}$ at very high similarity.

The experiments match the theory very well.



Other Methods for Compressions

1. Stochastic quantizations for data vectors and model weights, e.g., [SIGMOD 2021](#).
2. Many quantization schemes for Gaussian projections, e.g., [ICML 2014](#), [NIPS 2016](#), [NIPS 2017](#).
3. Sign Cauchy random projections, e.g., [NIPS 2013](#).
4. Sign Stable random projections, e.g., [arXiv 2015](#).
5. Sign random Fourier features, e.g., [NeurIPS 2022](#).
6. General quantization methods for random Fourier features, e.g., [AISTATS 2021](#), [ICML 2021](#).

7. Minwise hashing, consistent weighted sampling, and related methods will be covered later.

Outline

1. Vector similarity functions
2. Vector compressions
- 3. Vector similarity search**
4. Maximum inner product search (MIPS)
5. Fast neural ranking
6. GPU computing
7. GCWSNet, hashing algorithms
8. Boosted trees, ABC-boost
9. Distributed, adaptive, and federated learning
10. Privacy
11. Security
12. Others: generative AI models, NLP, knowledge graphs, multi-modal, cross-modal, advertising

Embedding Based Retrieval (EBR)

A typical scenario: given an input image, find a few similar images in the database of billions of images



A scenario of commercial value: query – ads (advertisement) matching

The screenshot shows a mobile search interface. At the top, it displays "中国移动" (China Mobile), the time "10:50", and battery level "77%". The search bar contains the text "阿拉斯加旅游 User search query". Below the search bar, there is a navigation bar with tabs: 全部 (All), 视频 (Video), 图片 (Images), 问答 (Questions & Answers), 贴吧 (Baidu Tieba), 小程序 (Small Programs), and 资讯 (News). A red dashed box highlights the "全部" tab. Below the tabs, a message says "搜索结果涉及价格仅供参考。请以商家官网为准" (Search results involve price reference only. Please refer to the merchant's official website). The main content area features an advertisement for "阿拉斯加旅行团_上纳美_带你享受极光之旅" (Alaska Travel Group_Shang Nami_Take you to enjoy the aurora). The ad includes text about traveling by boat to see marine life, crossing the Arctic Circle, and visiting the North Pole. It also mentions "纳美旅游 美国华人旅行社, 多年境外旅游经验, 专业放心" (Nanmei Travel, Chinese American Travel Agency, many years of overseas travel experience, professional and放心). At the bottom of the ad, there is a button labeled "立即提交" (Submit immediately) and a note "An ad from sponsored search". There are also buttons for "在线咨询" (Online consultation) and "请输入您的微信号" (Please enter your WeChat ID). The footer of the screenshot shows "北京纳美旅行社 vs 广告" (Beijing Nanmei Travel Agency vs Ad).

Query: “Travel in Alaska” (in Chinese)

Advertisement

The screenshot shows search results for the query "Travel in Alaska". The results are framed by a green dashed box. The first result is a link titled "2019阿拉斯加州旅游攻略,11月阿拉斯加州..." (2019 Alaska Travel Guide, November Alaska...). Below the title, there is a snippet of text: "阿拉斯加州旅游攻略指南,携程攻略社区! 靠谱的旅游攻略平台, 最佳的阿拉斯加州自助游、自由行、自驾游、跟团旅..." (Alaska Travel Guide, Ctrip Travel Community! Reliable travel guide platform, best self-tour, independent travel, self-driving, group tour...). The result is attributed to "携程旅行 2018年6月11日" (Ctrip Travel June 11, 2018). To the right of the result, the text "The organic search results" is written. Below the result, there is another snippet: "一篇详尽必收藏的,阿拉斯加州旅游攻略之..." (A detailed and must-collect Alaska Travel Guide...) and "第一次去阿拉斯加旅游必备的就是一份完善的攻略!而你是喜欢夏季去阿拉斯加旅游还是会选择冬季去呢?下面走..." (For your first trip to Alaska, what you need is a comprehensive guide! Do you like summer or winter? Below follow...). At the bottom of the search results, there are navigation icons for back, forward, search, and other functions.

Organic search results

KDD 2019

Embedding Based Retrieval (EBR)

KDD 2019

MOBIUS: Towards the Next Generation of Query-Ad Matching in Baidu's Sponsored Search

¹Miao Fan, ²Jiacheng Guo, ²Shuai Zhu, ²Shuo Miao, ¹Mingming Sun, ¹Ping Li

{fanmiao,guojiacheng,zhushuai,miaoshuo,sunmingming01,iping11}@baidu.com

¹ Cognitive Computing Lab (CCL), Baidu Research, Baidu Inc.

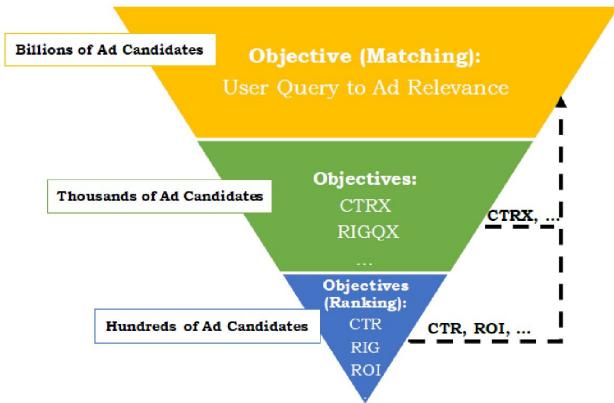
² Baidu Search Ads (Phoenix Nest), Baidu Inc.



Figure 1: "Mobius" is the internal code name of this project. Coincidentally, the well-known "Mobius Loop" is also the bird's-eye view of Baidu's Technology Park in Beijing, China.

After query embeddings and ads embeddings are generated, ANN (approximate near neighbor) search is the key technology, because each input query might correspond to millions or more potential ads candidates.

Traditional pipeline: ads candidates generated using different metrics from final CTR model

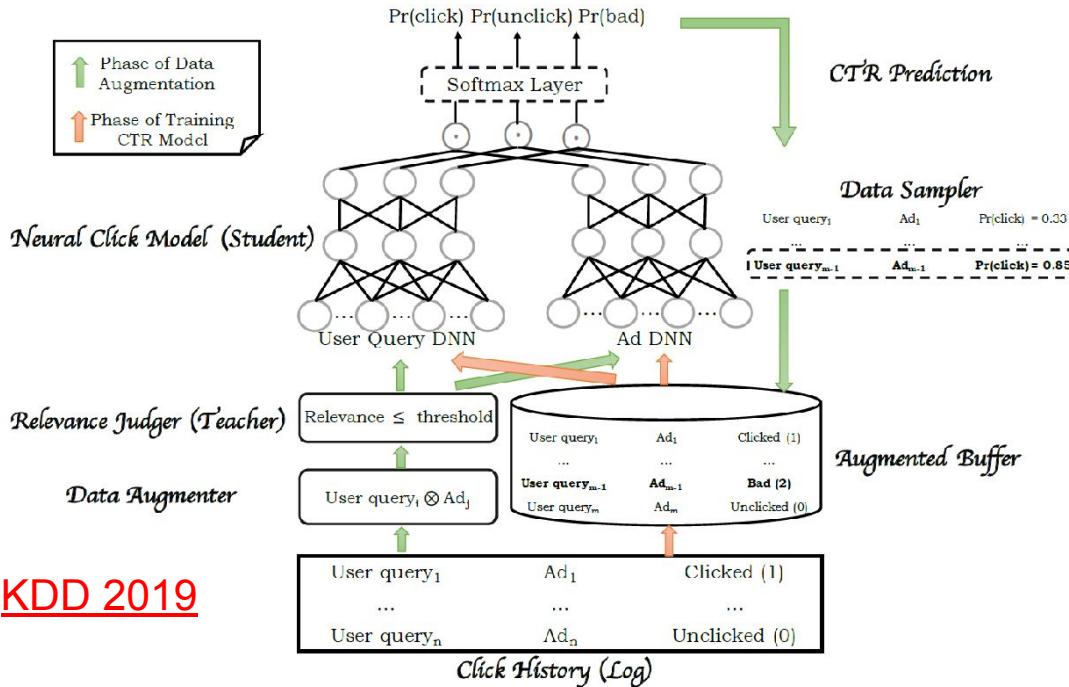


New pipeline: candidates generated by considering business metrics (CTR)



Mobius: Combining Recall with CTR Models

The classical **two-tower model** is used for training query and ads embeddings (by optimizing the **cosine** similarity). At the query time, each query embedding needs to be compared with millions or more ads embeddings. Thus, approximate near neighbor (**ANN**) search has become a standard component of **EBR**.



1. Business-related “weights” (e.g., bid price) can be considered at query time. ANN => approximate maximum inner product search (**MIPS**).
2. We can replace this simple click/cosine model with another deep neural net.
=> **neural ranking**

EBR: better technology => increased revenues

Extremely useful for search, feed and advertising

Table 3: The improvements on CPM, CTR, and ACP of Mobius-V1 compared with the previous system deployed on different websites/apps. The results are based on our 7-day surveillance of the entire online traffic.

Launched Platform	CPM	CTR	ACP
Baidu App on Mobiles	+3.8%	+0.7%	+2.9%
Baidu Search on PCs	+3.5%	+1.0%	+2.2%
Affiliated Websites/Apps	+0.8%	+0.5%	+0.2%

CPM can be viewed as revenues.

CTR and CVR are directly related to revenues

We demonstrate the effects of launching EGM in Baidu video advertising platform. We show its influence on click-through rate (CTR) and conversion ratio (CVR) defined as follows:

$$CTR = \frac{\# \text{ of clicks}}{\# \text{ of impressions}}, \quad CVR = \frac{\# \text{ of conversions}}{\# \text{ of impressions}} \quad (20)$$

There two metrics directly measure the effectiveness of the advertising. The experiments are conducted on an observation window with a length of a week. Before launching EGM, our video advertising platform is based on TDM. After launching EGM, CTR relatively increases by 1.33% and CVR relatively increases by 1.07%.

MOBIUS: Towards the Next Generation of Query-Ad Matching in Baidu's Sponsored Search, KDD 2019

EGM: Enhanced Graph-based Model for Large-scale Video Advertisement Search, KDD 2022

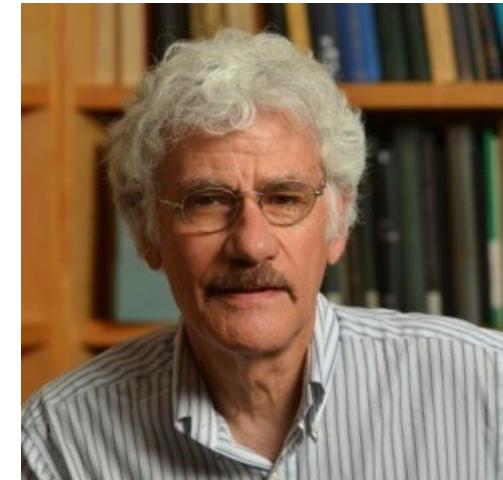
Approximate Near Neighbor (ANN) Search

ANN is an ancient topic in CS, possibly starting in the 1970s; see Prof. Friedman's works:

Jerome H. Friedman, F. Baskett, and L. Shustek. An algorithm for finding nearest neighbors. *IEEE Transactions on Computers*, 24:1000–1006, 1975.

Jerome H. Friedman, J. Bentley, and R. Finkel. An algorithm for finding best matches in logarithmic expected time. *ACM Transactions on Mathematical Software*, 3:209–226, 1977.

<https://www.linkedin.com/feed/update/urn:li:activity:6965026431579951104/> the talk at Google on 8/15/22 also discussed Prof. Friedman's contributions on boosting and trees.



Graph-Based ANN

Graph-based ANN algorithms (such as HNSW) have become very popular, especially in EBR applications. We will introduce HNSW based on our implementations and the GPU version.

Unlike other ANN algorithms such as hashing methods, the GPU version of HNSW is not trivial.

SONG: Approximate Nearest Neighbor Search on GPU

Weijie Zhao, Shulong Tan and Ping Li

ICDE 2020

Cognitive Computing Lab

Baidu Research USA

1195 Bordeaux Dr. Sunnyvale, CA 94089

10900 NE 8th St. Bellevue, WA 98004

{weijiezhao,shulongtan,liping11}@baidu.com

Graph-Based ANN

Algorithm 1 Searching algorithm on the proximity graph.

Input: Graph index $G(V, E)$; a query point p ;

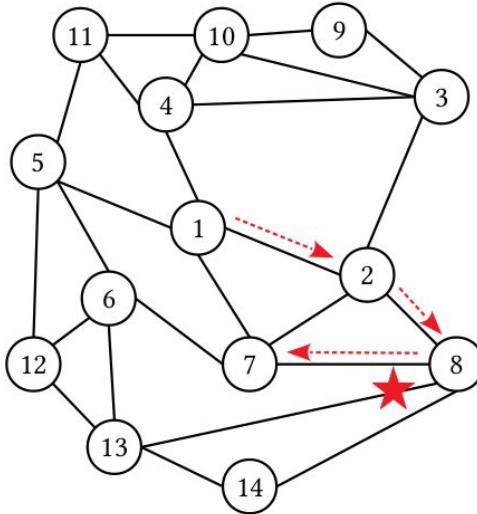
Number of output candidates K

Output: Top K candidates for each query $topk$

```

1. Initialize a binary min-heap as priority queue  $q$  and a hash
   set  $visited$  with the default starting point. Construct an
   empty binary max-heap as priority queue  $topk$ 
2. while  $q \neq \emptyset$  do
3.    $(now\_dist, now\_idx) \leftarrow q.pop\_min()$ 
4.   if  $topk.size = K$  and  $topk.peak\_max() < now\_dist$  then
5.     break
6.   else
7.      $topk.push\_heap((now\_dist, now\_idx))$ 
8.   end if
9.   for each  $(now\_idx, v) \in E$  do
10.    if  $visited.exist(v) \neq \text{true}$  then
11.       $d \leftarrow dist(p, v)$ 
12.       $visited.insert(v)$ 
13.       $q.push\_heap((d, v))$ 
14.    end if
15.   end for
16. end while
17. return  $topk$ 

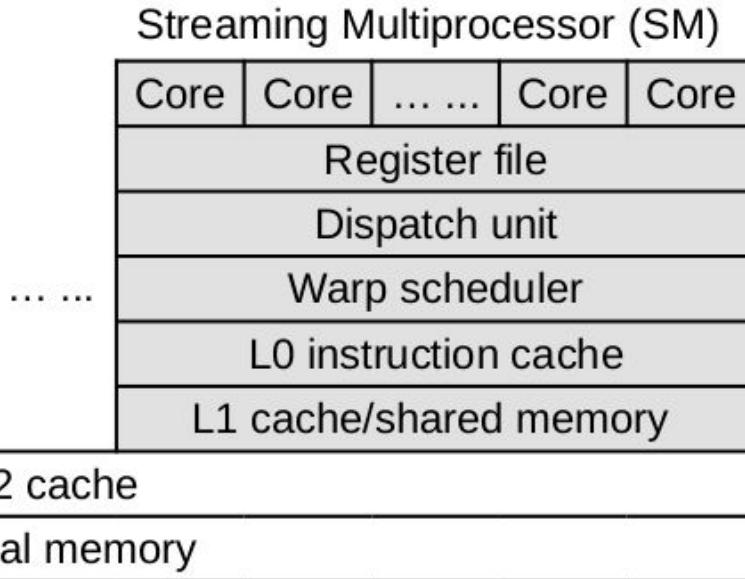
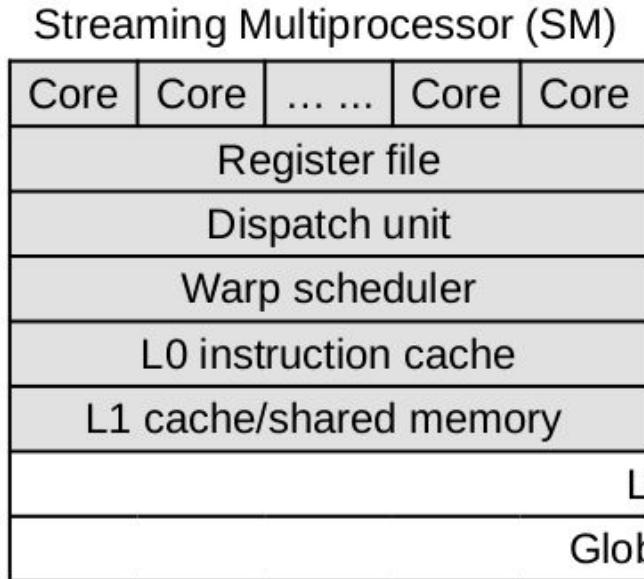
```



	Initialization
	$q: 1$ $topk: \emptyset$ $visited: 1$
Iteration 1	$q: 2\ 7\ 4\ 5$ $topk: 1$ $visited: 1\ 2\ 4\ 5\ 7$
Iteration 2	$q: 8\ 7\ 3\ 4\ 5$ $topk: 1\ 2$ $visited: 1\ 2\ 3\ 4\ 5\ 7\ 8$
Iteration 3	$q: 7\ 3\ 4\ 5\ 14\ 13$ $topk: 1\ 2\ 8$ $visited: 1\ 2\ 3\ 4\ 5\ 7\ 8\ 13\ 14$
Iteration 4	$q: 3\ 4\ 5\ 6\ 14\ 13$ $topk: 7\ 2\ 8$ $visited: 1\ 2\ 3\ 4\ 5\ 6\ 7\ 8\ 13\ 14$

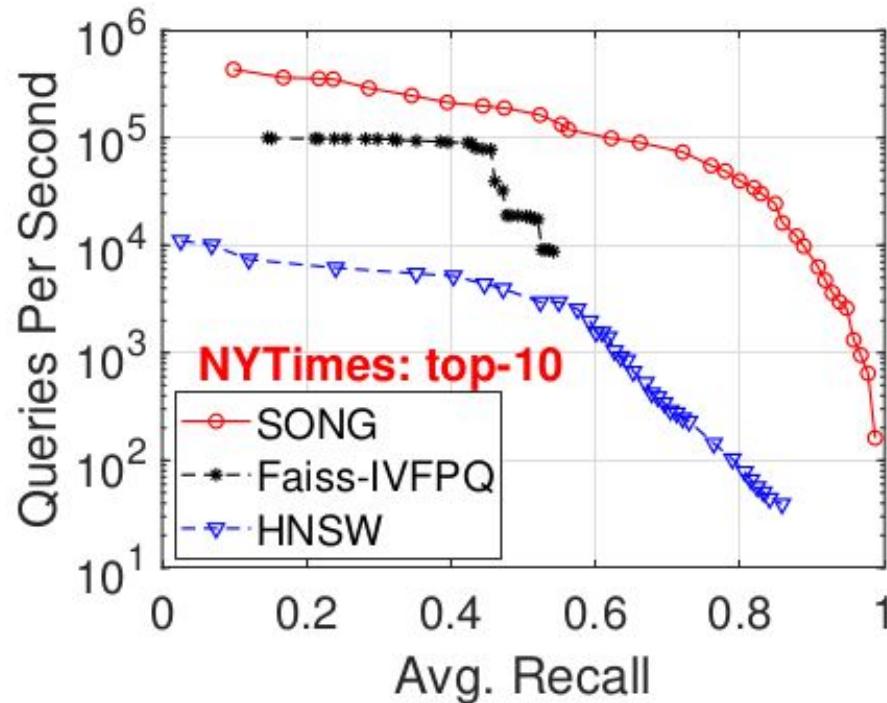
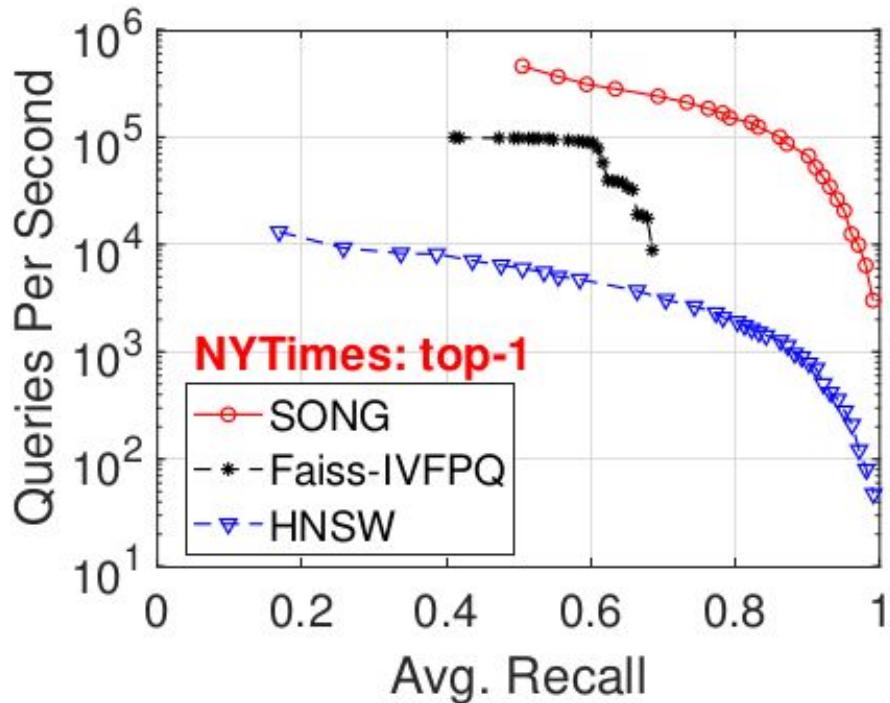
Fig. 1: An example to illustrate the searching algorithm on the proximity graph (Algorithm 1). The star represents the query point p . In this example, we target to find the top $K = 3$ nearest neighbors of p . Vertex 1 is the default starting point. The searching path $1 \rightarrow 2 \rightarrow 8 \rightarrow 7$ is highlighted by the dashed arrows. The right part shows the states of two priority queues— q and $topk$ —and the hash table $visited$ in each iteration.

GPU Architecture

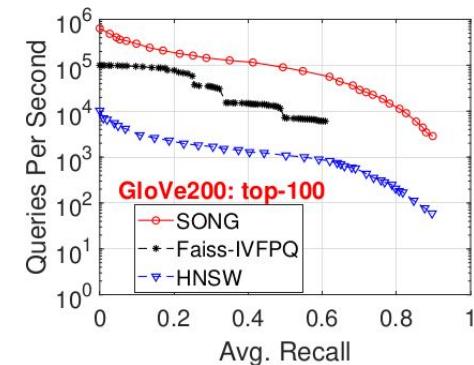
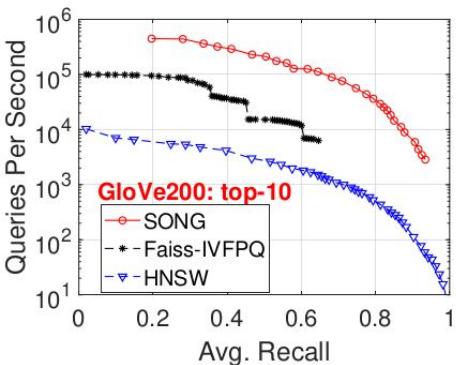
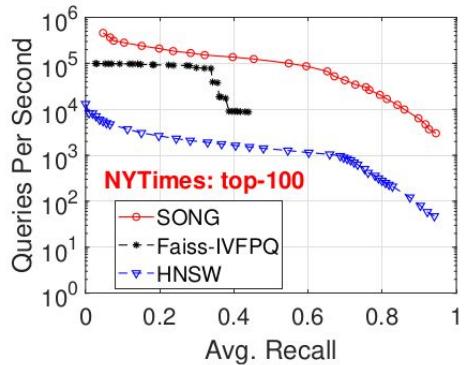
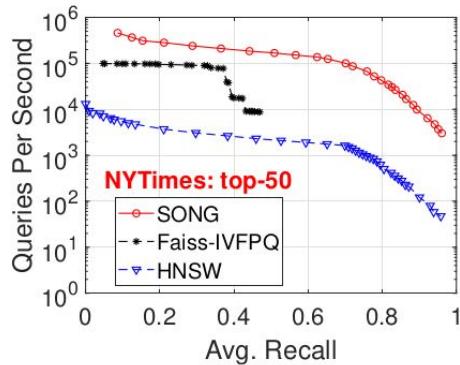
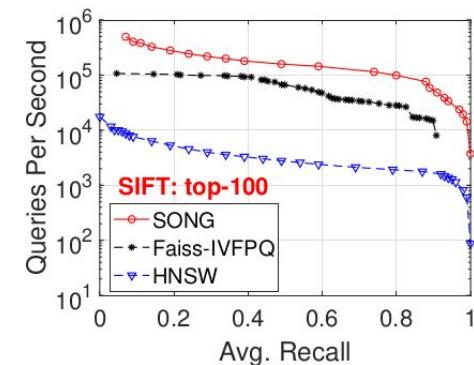
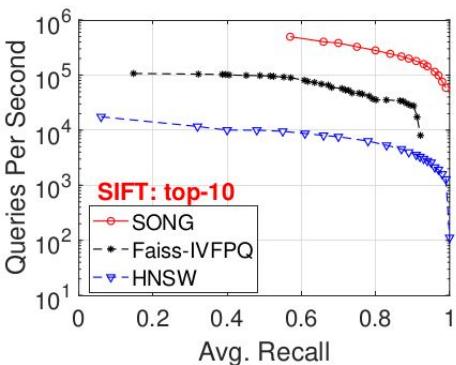
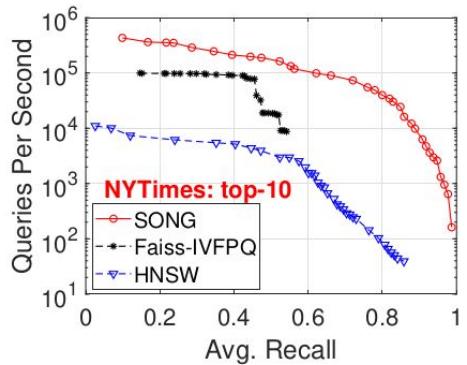
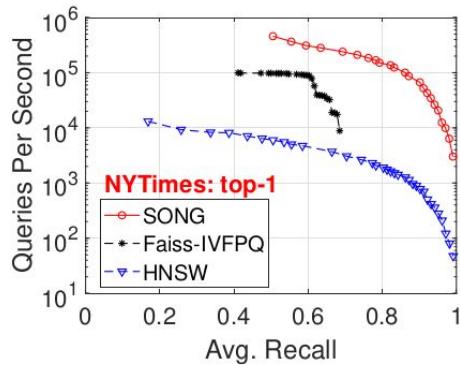


GPU for HNSW is not trivial. Need to understand architecture well for efficient implementation.

ANN Performance Evaluations



ANN Performance Evaluations



Challenges with Graph-Based ANN

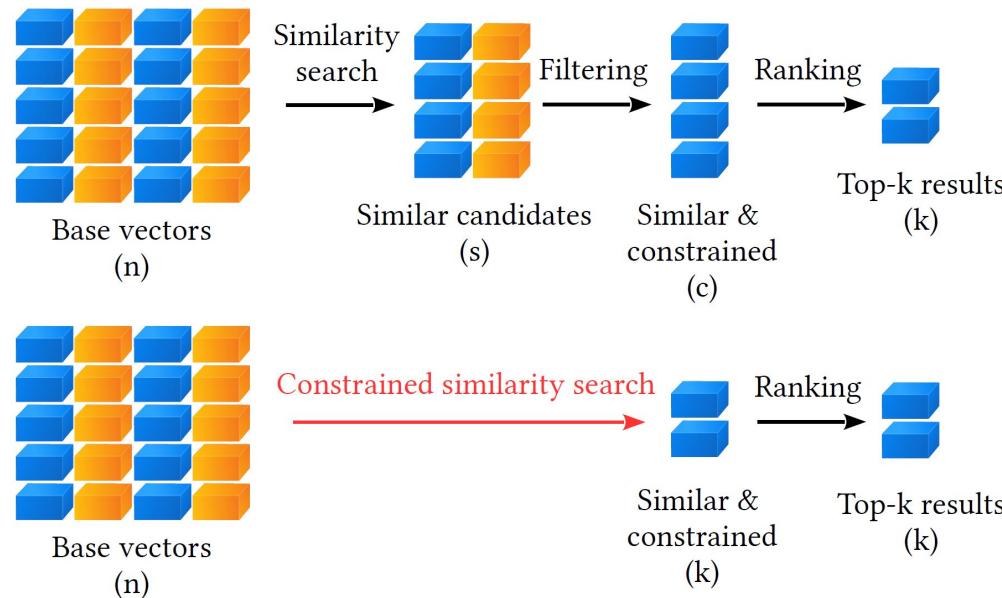
- Building graph indexing can be expensive. Updating graph indexing is also expensive.
- GPU implementation is not trivial (although we have done that).
- ANN + business filter is an urgent demand from industry.
- The search time is often dominated by the similarity computing time on the fly.
- Maximum inner product search (MIPS) with graph-based ANN.
- Storing original embedding vectors can be too expensive, especially for the memory.
- (GPU) Fast neural ranking

Since 2017, we have been working on graph-based ANN algorithms and have developed many solutions to address the above challenges.

ANN + Business Filters

Constrained Approximate Similarity Search on Proximity Graph (first paper on this topic)

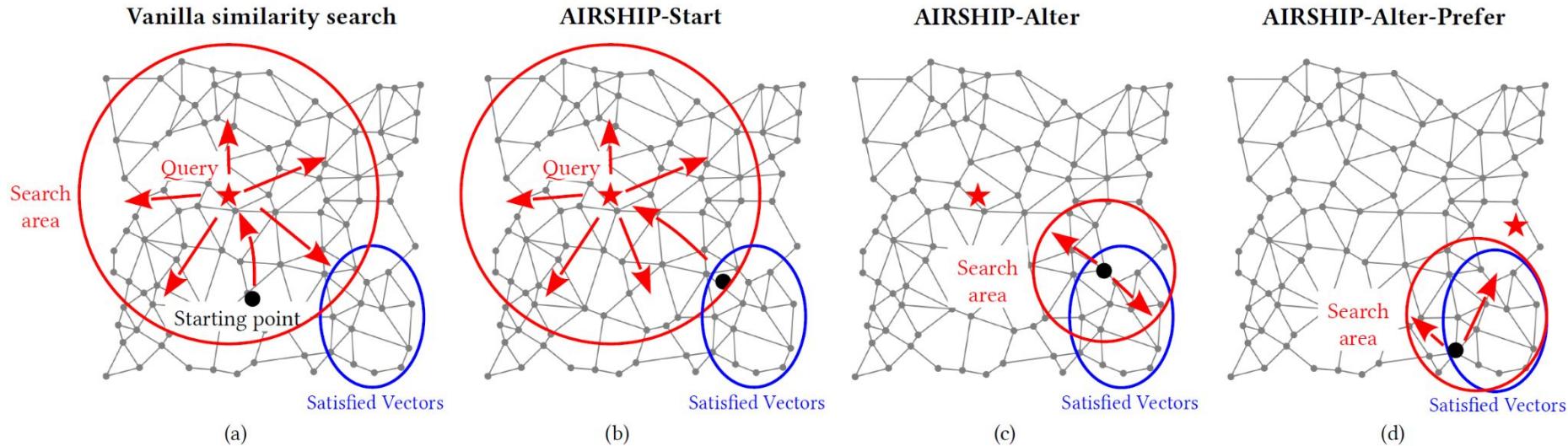
- Retrieving vectors satisfied the filter **without reconstructing** graph index
- More than 100X faster than FAISS (PQ) and HNSW (Vanilla) for most recall levels



ANN + Business Filters

Constrained Approximate Similarity Search on Proximity Graph (first paper on this topic)

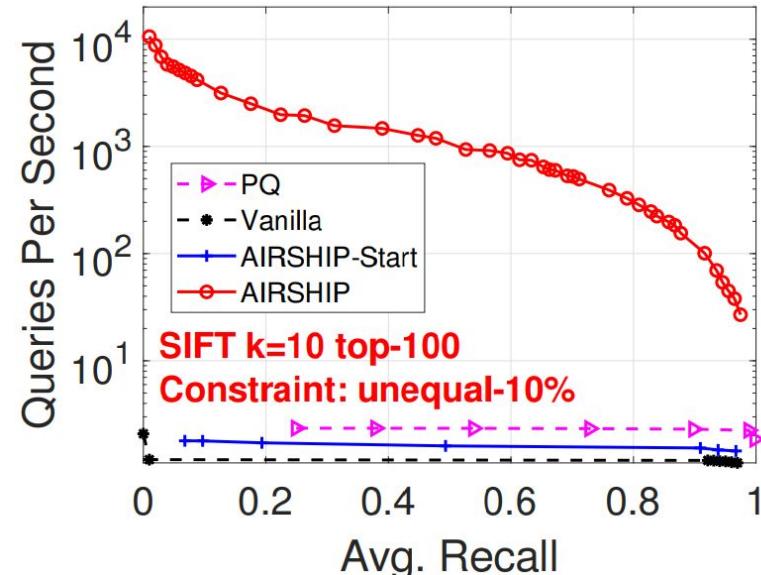
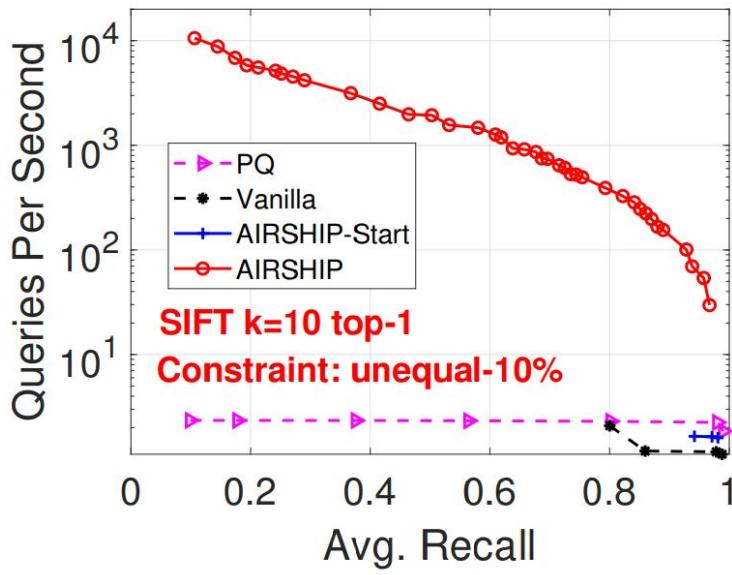
- Retrieving vectors satisfied the filter **without reconstructing** graph index
- More than 100X faster than FAISS (PQ) and HNSW (Vanilla) for most recall levels



ANN + Business Filters

Constrained Approximate Similarity Search on Proximity Graph (first paper on this topic)

- Retrieving vectors satisfied the filter **without reconstructing** graph index
- More than 100X faster than FAISS (PQ) and HNSW (Vanilla) for most recall levels



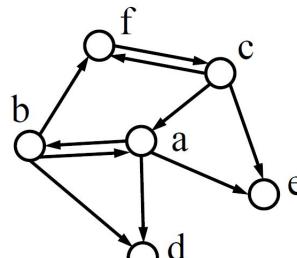
ANN + Business Filters: Example Use Cases

- Marketing
 - Constraints on the customer's preferences, purchase history, or the specific item they are currently viewing
- Fraud detection
 - Include patterns or thresholds for suspicious behavior, such as unusual transaction amounts, frequencies, or locations
- Talent Acquisition
 - Narrow down the pool to candidates/jobs with a particular skill set, a certain level of experience, or specific qualifications
- Healthcare
 - Filter on patient demographics, medical conditions, or treatment history
- Applications in Ads: ad publisher targets on given locations and/or companies

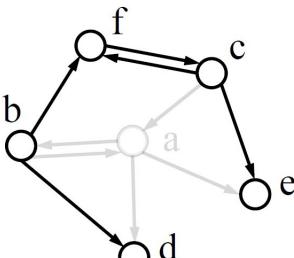
Updating Graph Indexes

Proximity Graph Maintenance for Fast Online Nearest Neighbor Search

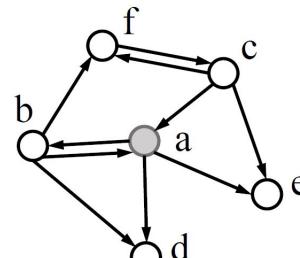
- Adding/deleting data vectors **without reconstructing** graph index
- Our techniques are highly efficient with very little loss on accuracy



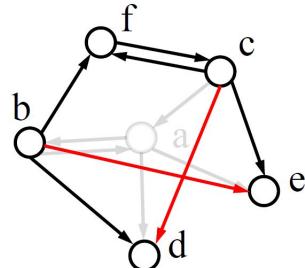
(a) Original Graph



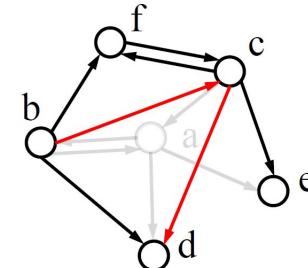
(b) Pure Deletion



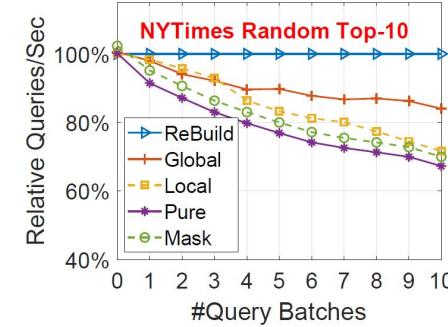
(c) Vertex Masking



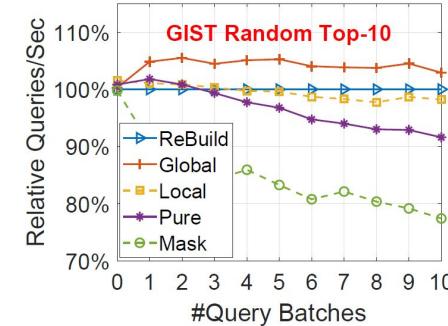
(d) Local Reconnection



(e) Global Reconnection

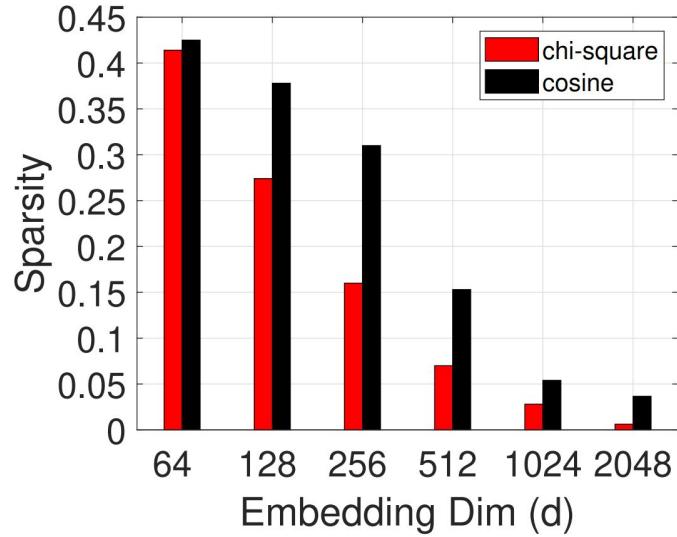


Relative efficiency at 80% recall



Techniques for Sparse Vectors

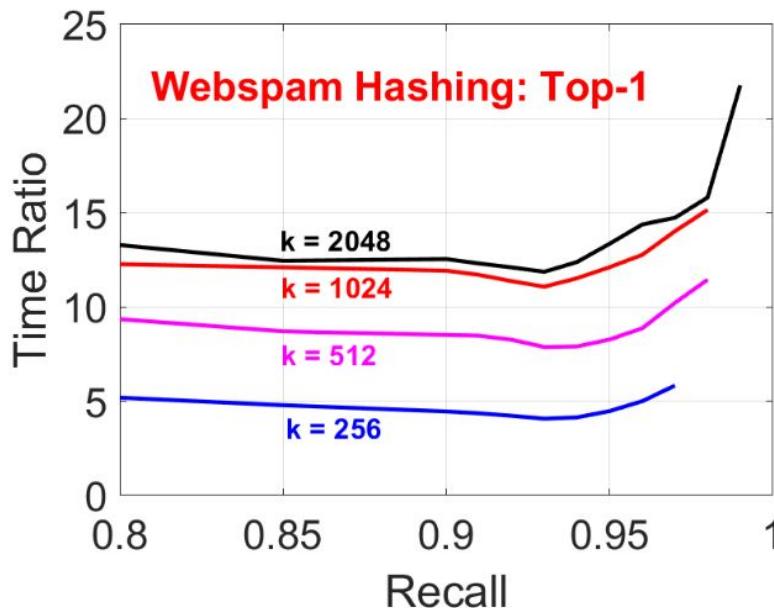
- Natural vectors (e.g., from text n-gram data) are often sparse. Embedding vectors can be dense or sparse. By using our latest “[Chi-square two-tower model](#)”, we obtain highly sparse embeddings vectors.
- HNSW for sparse vectors is a less explored area. We have developed a series of new techniques to improve the efficiency of HNSW on sparse data.
- In particular, HNSW + hashing is proven effective, for example, HNSW + [sign cauchy random projections](#).
- Since 2005, we have developed a wide range of hashing algorithms suitable for sparse data including b-bit minwise hashing, one permutation hashing, circular hashing, consistent weighted sampling (CWS), etc.



In an industry application, the sparsity (fraction of non-zeros) of embedding can be as low as 2%.

Vector Compression and Hashing

Webspam is high-dimensional, sparse, with ~4000 non-zeros per vector. With compression techniques (here we used “[sign cauchy random projections](#)” in NIPS’13), we can reduce each vector to $k = 256$ (or $k = 2048$) bits per vector, corresponding to 500-folder (or 62-fold) reduction in space. The overall similarity computational time for HNSW is reduced to 5-fold (or 13-fold).



HNSW experiments with compression, as reported in

[Practice with Graph-based ANN Algorithms on Sparse Data: Chi-square Two-tower model, HNSW, Sign Cauchy Projections](#).

Hashing-based ANN

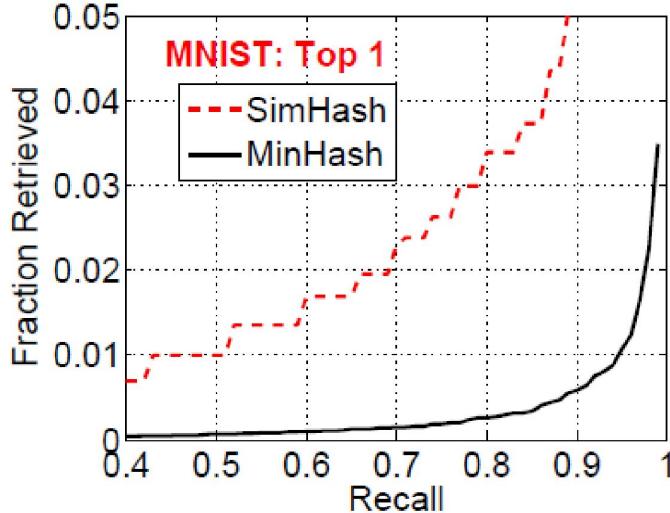
- Example: each vector => 4-bit code => a hash table of $2^4 = 16$ buckets.
- All vectors are stored in the buckets according to the hash code.
- When a new vector arrives, we generate its 4-bit hash code and retrieve the vectors in the corresponding bucket. This way, we can avoid exhaustive search of all data points.
- Only using one hash table may not perform well. Large table => too many buckets => few retrieved points. A common strategy is to build multiple tables and use the union results.

Index	Data Points
00 00	8, 13, 251
00 01	5, 14, 19, 29
00 10	(empty)
11 01	7, 24, 156
11 10	33, 174, 3153
11 11	61, 342

Index	Data Points
00 00	2, 19, 83
00 01	17, 36, 129
00 10	4, 34, 52, 796
11 01	7, 198
11 10	56, 989
11 11	8, 9, 156, 879

Choice of Hash is Crucial

In Defense of MinHash Over SimHash, AISTATS 2014



Fraction retrieved versus recall plots: standard way to evaluate ANN algorithms

Exhaustive search = 100% fraction retrieved

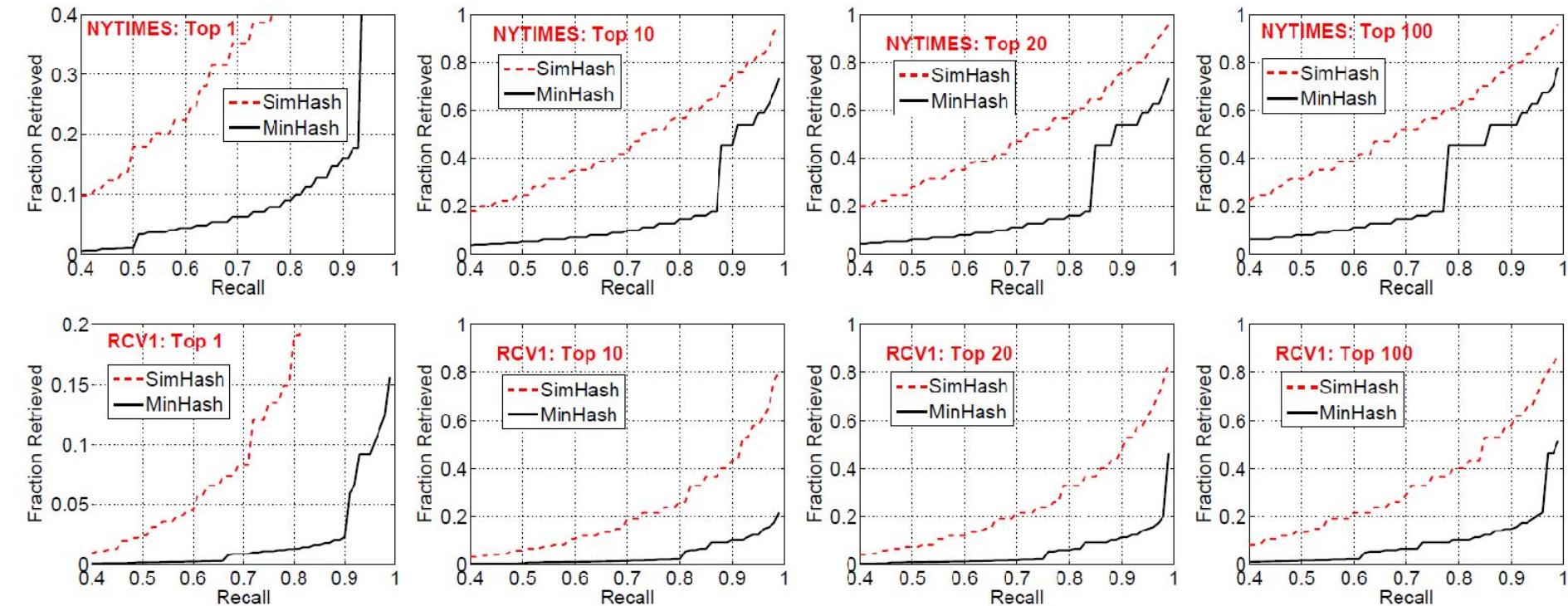
Recall = percentage of ground truths in search results

SimHash: In order to achieve a recall at 90% (0.9), we need to search for **5%** of the data points. 5% means a 20-fold reduction in cost, pretty good.

MinHash: In order to achieve a recall at 90% (0.9), we need to search for **0.5%** of the data points. 0.5% is ten times better than 5%.

MinHash versus SimHash

In Defense of MinHash Over SimHash, AISTATS 2014



Outline

1. Vector similarity functions
2. Vector compressions
3. Vector similarity search
4. **Maximum inner product search (MIPS)**
5. Fast neural ranking
6. GPU computing
7. GCWSNet, hashing algorithms
8. Boosted trees, ABC-boost
9. Distributed, adaptive, and federated learning
10. Privacy
11. Security
12. Others: generative AI models, NLP, knowledge graphs, multi-modal, cross-modal, advertising

Maximum Inner Product Search (MIPS) for Ads

$$\cos(x, y) \times w = \frac{x^T y \times w}{\|x\| \|y\|} = \left(\frac{x}{\|x\|} \right)^T \left(\frac{y \times w}{\|y\|} \right)$$

x: user query embedding vector

y: ads embedding vector

w: weights from business considerations such as bid price

The task is transformed from maximum cosine search to maximum inner product search.

Technical challenge: measures like cosine satisfy triangle inequality but inner products do not.

Our works on MIPS

KDD'21, [Norm Adjusted Proximity Graph for Fast Inner Product Retrieval](#)

EMNLP'19, [On Efficient Retrieval of Top Similarity Vectors](#)

NeurIPS'19, [Möbius Transformation for Fast Inner Product Search on Graph](#)

WWW'15, [Asymmetric Minwise Hashing for Indexing Binary Inner Products and Set Containment](#)

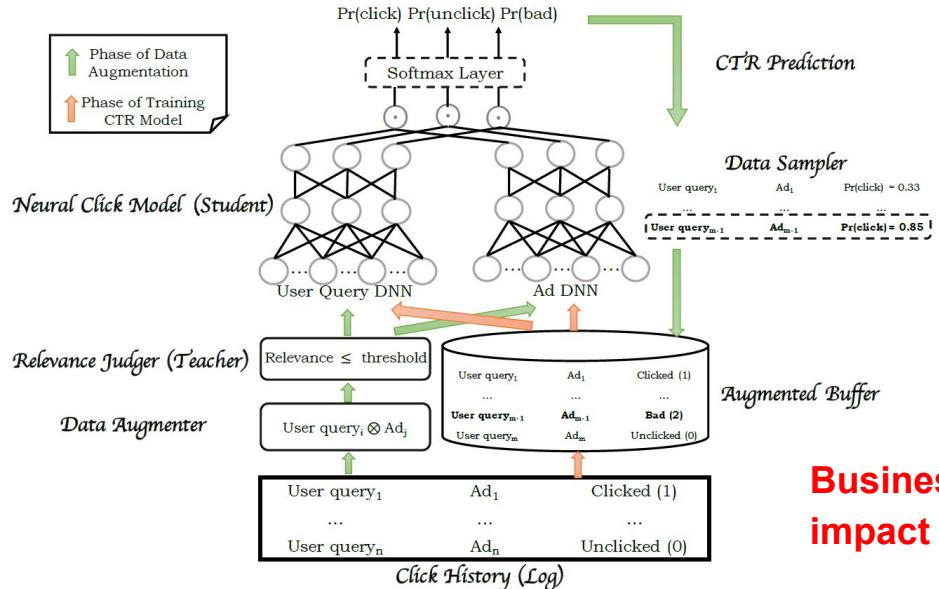
UAI'15, [Improved Asymmetric Locality Sensitive Hashing \(ALSH\) for Maximum Inner Product Search \(MIPS\)](#)

NIPS'14, [Asymmetric LSH \(ALSH\) for Sublinear Time Maximum Inner Product Search \(MIPS\)](#)
(The Best Paper Award in NIPS'14)

Business Motivation for MIPS

KDD'19, [MOBIUS: Towards the Next Generation of Query-Ad Matching in Baidu's Sponsored Search](#)

By adding weights (e.g., bid price) to vectors, the search problem becomes MIPS, which is widely used in (e.g.,) advertising. The figures and tables are from Baidu's published work in KDD'19, for the new CTR retrieval/training system.



Business impact

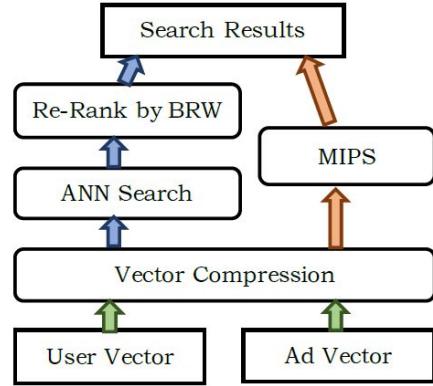


Figure 7: The fast ad retrieval framework. The two types of vectors will be compressed first to save the memory space. After that, two strategies can be applied: (a) ANN search by cosine similarity and then re-rank by the business related weight (BRW); (b) Ranking by exploiting weight, which is a Maximum Inner Product Search (MIPS) problem.

Launched Platform	CPM	CTR	ACP
Baidu App on Mobiles	+3.8%	+0.7%	+2.9%
Baidu Search on PCs	+3.5%	+1.0%	+2.2%
Affiliated Websites/Apps	+0.8%	+0.5%	+0.2%

Möbius Transformation for Fast MIPS on Graph

Möbius Transformation for Fast Inner Product Search on Graph, NeurIPS 2019

Outline

General setting.

Search on graph.

Voronoi Cell and Delaunay graph.

Search on Delaunay graph.

ℓ^2 -Delaunay graph and IP-Delaunay graph.

Möbius transformation and graph isomorphism.

Proposed algorithm.

Experimental results.

Möbius Transformation for Fast MIPS on Graph

[NeurIPS 2019](#)

General Setting

$$X, Y \subset \mathbb{R}^d.$$

$$\text{Dataset } S = \{x_1, \dots, x_n\} \subset X.$$

Real valued function $f : X \times Y \rightarrow \mathbb{R}$.

Aim to solve the optimization problem:

$$\arg \max_{x_i \in S} f(x_i, q) \quad \text{for } q \in Y.$$

Allow prepossessing on dataset S .

Most interesting examples:

$$f(x, y) = -\|x - y\| \quad \text{and} \quad f(x, y) = x^\top y.$$

Möbius Transformation for Fast MIPS on Graph

NeurIPS 2019

Search on graph

Let $f(x, y) = -\|x - y\|$.

$\arg \max_{x_i \in S} f(x_i, q)$ aims to find the nearest point of q in S .

Search on graph strategy

1. Build a proximity graph G on S by connecting certain points, e.g., k -NN graph.
2. Randomly select a vertex x from the graph.
3. Evaluate the ℓ^2 -distance $\|x - q\|$ and $\|y - q\|$, where y 's are neighbors of x on the graph G .
4. If x' is closer to q than x , then replace x by y and repeat step 3.
5. Stop if x is closer to q than x 's neighbors on graph.

This procedure finds the exact solution if and only if G contains *Delaunay graph* as a subgraph.

Möbius Transformation for Fast MIPS on Graph

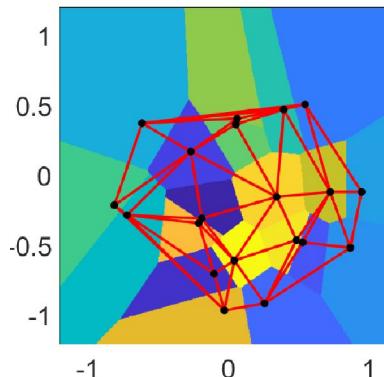
[NeurIPS 2019](#)

Voronoi Cells and Delaunay Graph

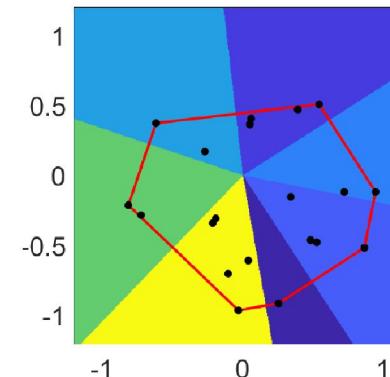
Goal: to find $\arg \max_{x_i \in S} f(x_i, q)$.

Voronoi cells: solution area of the problem.

Delaunay graph: the dual graph of Voronoi diagram.



ℓ^2 -Delaunay graph



IP-Delaunay graph

Every ℓ^2 -Voronoi cell is nonempty, but IP-Voronoi cell can be empty.

Möbius Transformation for Fast MIPS on Graph

[NeurIPS 2019](#)

Sufficiency and Necessity Delaunay Graph

Theorem

For given f , suppose its Voronoi cells w.r.t. any dataset are connected, then

- for $q \in Y$, performing greedy search on Delaunay graph returns the solution of $\arg \max_{x_i \in S} f(x_i, q)$;
- conversely, for any G' does not contain Delaunay graph as a subgraph, there exists a query $q \in Y$ such that greedy search on G' does not always retrieve the exact solution.

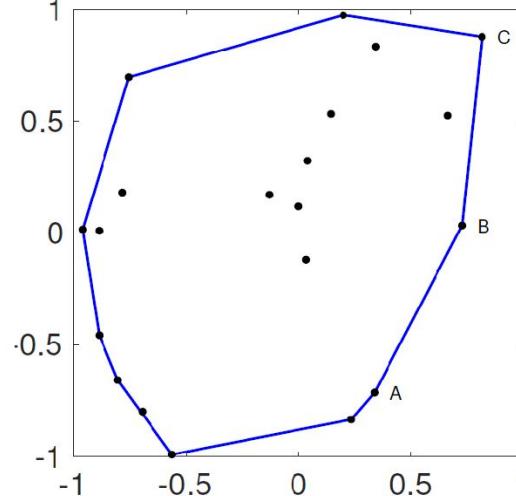
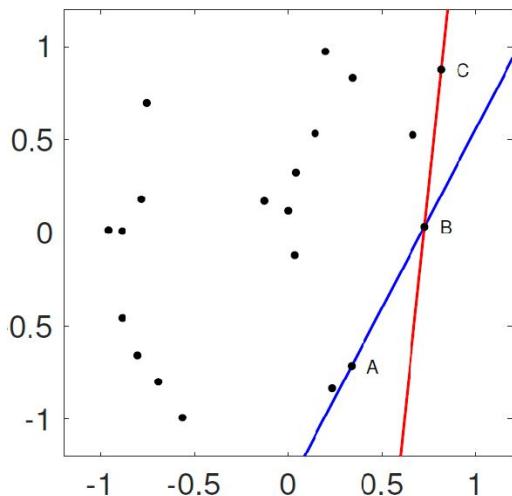
- Any ℓ^p -norm and inner product satisfy the assumption on f .
- Delaunay graph is the smallest graph such that greedy search always return exact solution.

Möbius Transformation for Fast MIPS on Graph

[NeurIPS 2019](#)

Empty Half Space Criterion

- The line AB in divides the plane into two open half-spaces. One of the half-space does not contain any data points, so A and B are connected in IP-Delaunay graph.

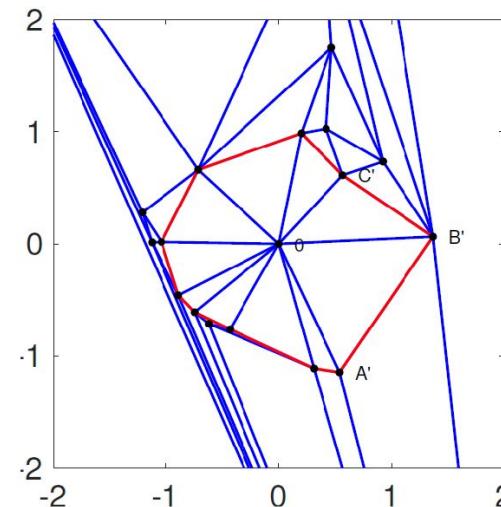
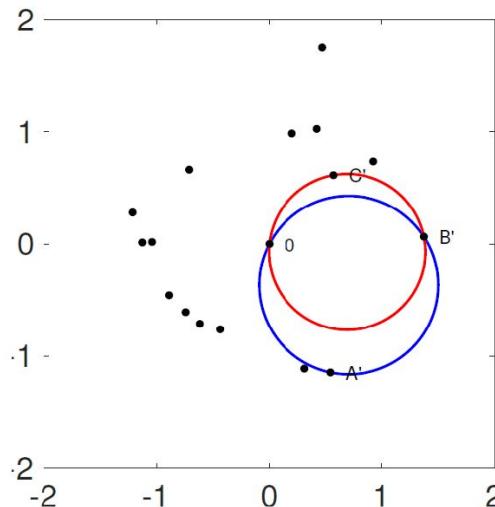


Möbius Transformation for Fast MIPS on Graph

[NeurIPS 2019](#)

Empty Sphere Criterion

The circumcircle of $0, A$ and B does not contain any data points inside, so there is a simplex with vertices $0, A'$ and B' in the ℓ^2 -Delaunay graph.



Möbius Transformation for Fast MIPS on Graph

[NeurIPS 2019](#)

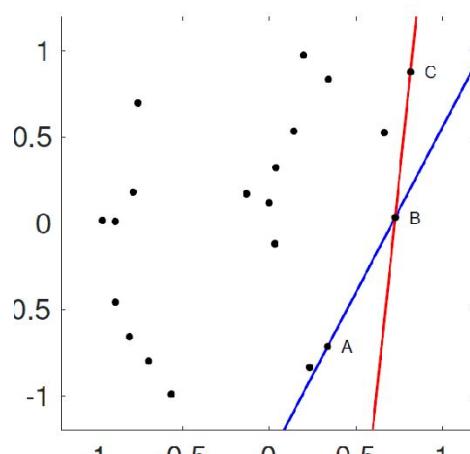
Mobius Transformation

Möbius transformation $y = \frac{x}{\|x\|^2}$.

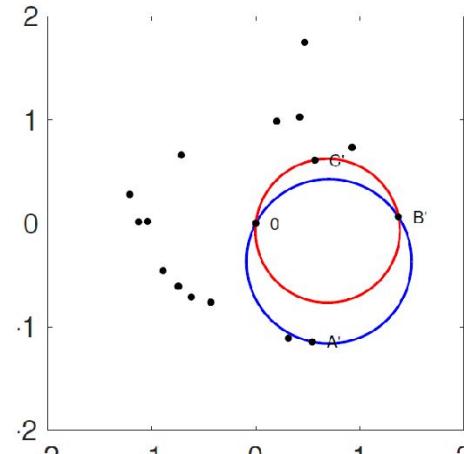
Maps hyperplanes to spheres.

Maps lines to circles in \mathbb{R}^2 .

Let $y_i = x_i / \|x_i\|^2$.



$$S = \{x_1, \dots, x_n\}$$



$$S' = \{0, y_1, \dots, y_n\}$$

Möbius Transformation for Fast MIPS on Graph

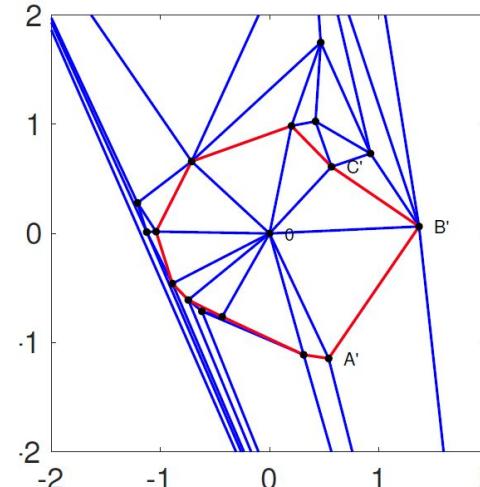
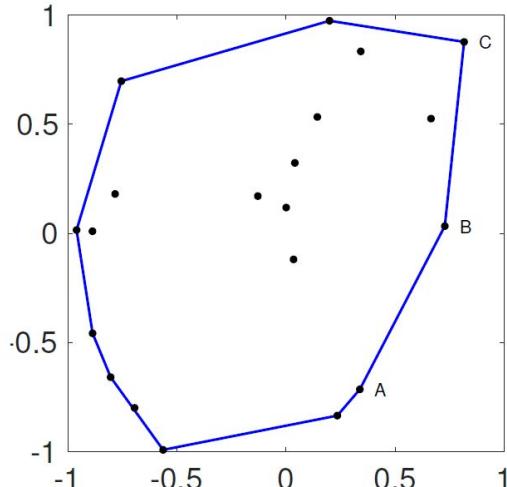
[NeurIPS 2019](#)

Graph Isomorphism

Theorem

The following two graphs are isomorphic:

- (a) IP-Delaunay graph before transformation (blue graph on the left);
- (b) The neighborhood of 0 (in the graph sense) of ℓ^2 -Delaunay graph after transformation (red graph on the right).



Möbius Transformation for Fast MIPS on Graph

[NeurIPS 2019](#)

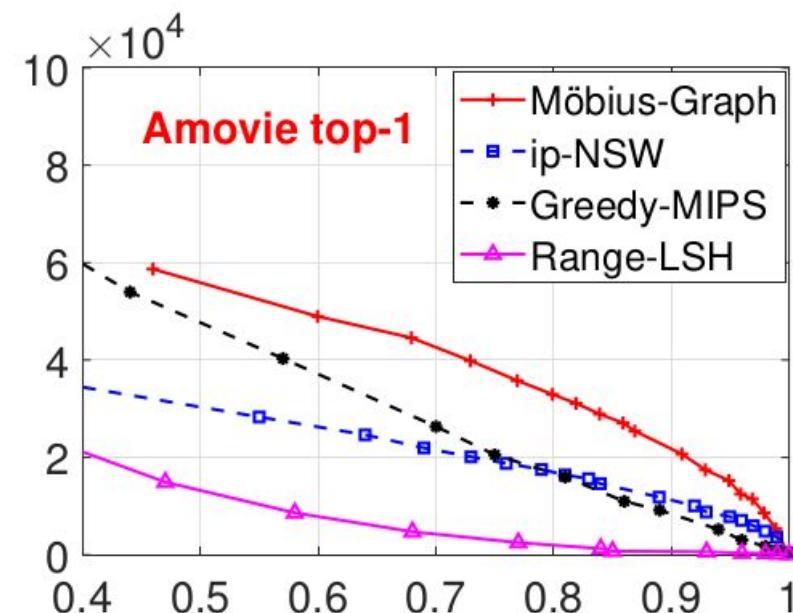
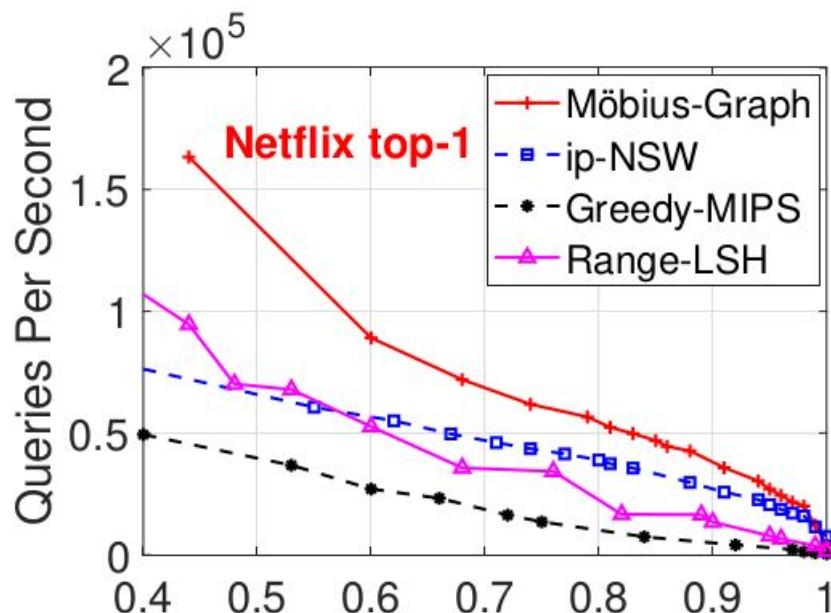
Proposed algorithm

1. Let $\tilde{S} := \{y_i = x_i / \|x_i\|^2 \mid x_i \in S\} \cup \{0\}$ be the transformed dataset.
2. Construct approximate ℓ^2 -Delaunay graph, e.g., HNSW, w.r.t. \tilde{S} .
3. Replace the vertices y_i by original data vectors x_i .
4. Start from 0, perform greedy inner product search on the graph.

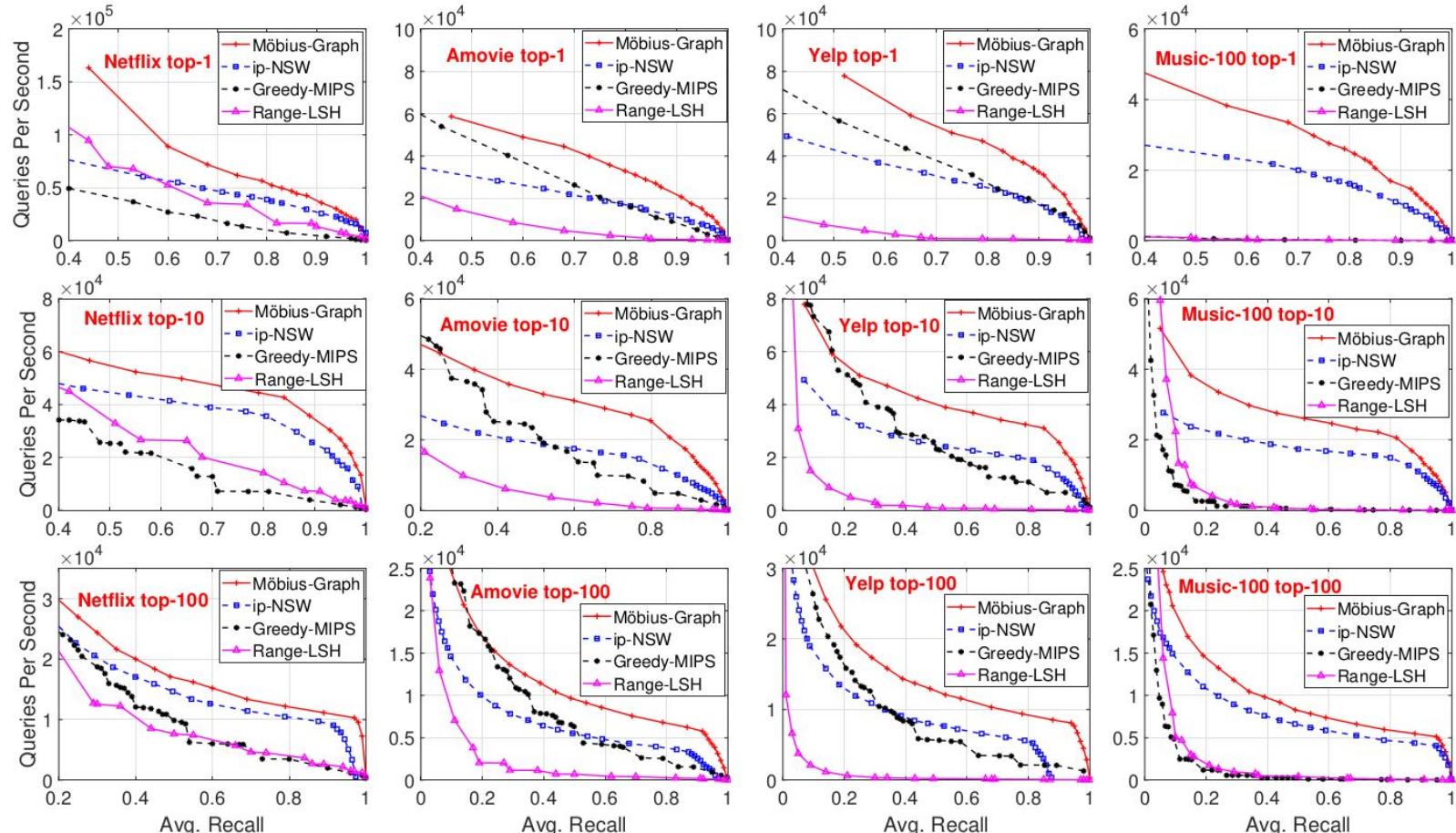
Möbius Transformation for Fast MIPS on Graph

[NeurIPS 2019](#)

Experiments



Möbius Transformation for Fast MIPS on Graph



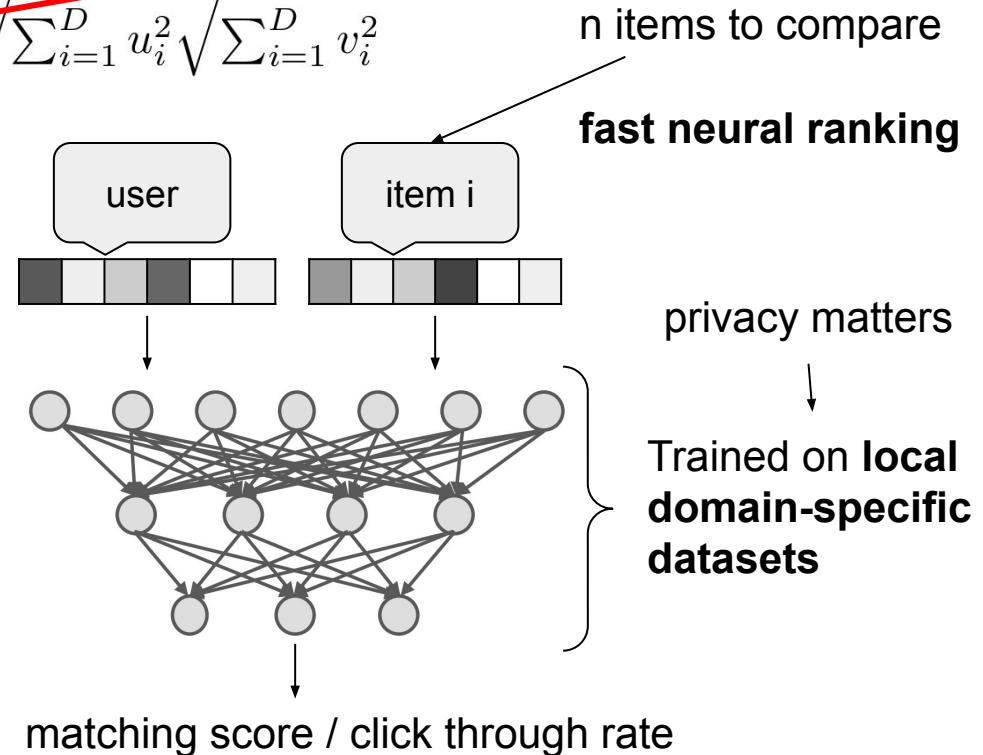
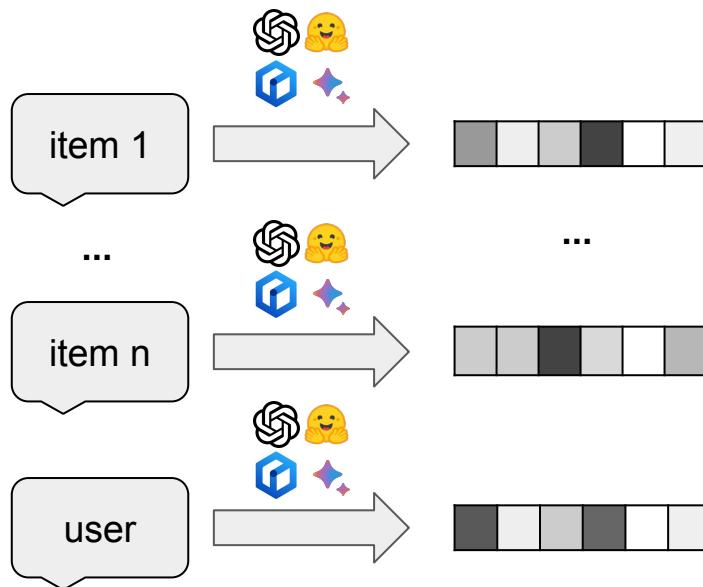
Outline

1. Vector similarity functions
2. Vector compressions
3. Vector similarity search
4. Maximum inner product search (MIPS)
5. **Fast neural ranking**
6. GPU computing
7. GCWSNet, hashing algorithms
8. Boosted trees, ABC-boost
9. Distributed, adaptive, and federated learning
10. Privacy
11. Security
12. Others: generative AI models, NLP, knowledge graphs, multi-modal, cross-modal, advertising

Fast Neural Ranking: Better Accuracy

Neural net model is much more accurate than cosine

$$\rho = \frac{\sum_{i=1}^D u_i v_i}{\sqrt{\sum_{i=1}^D u_i^2} \sqrt{\sum_{i=1}^D v_i^2}}$$



Key Components in Fast Neural Ranking

1. Replace cosine with a neural net and train the embeddings together with the neural net.
2. Store embeddings just like in two-tower models (saving time using space).
3. When a query arrives, generate (or retrieve) its embedding, and evaluate the neural net for every item embedding to find the best item with the maximum score. However, this would be extremely slow by evaluating all item embeddings for every query. This is the same motivation for vector ANN. The challenge however would be substantially more difficult.
4. We have developed a series of fast neural ranking techniques which achieve high accuracy by only evaluating much **less than 1%** of the neural nets.
5. In industry practice, to further boost the serving efficiency, it is often desirable to use GPUs for fast neural ranking. Hence this technique is also known as “**GPU fast neural ranking**”.

Our Works on (GPU) Fast Neural Ranking

WSDM'20, [Fast Item Ranking under Neural Network based Measures](#)

(The proposed **SL2G** algorithm is the simplest algorithm and easy to use in practice)

VLDB'22, [Fast Neural Ranking on Bipartite Graph Indices](#)

KDD'22, [EGM: Enhanced Graph-based Model for Large-scale Video Advertisement Search](#)

SIGIR'23, [Asymmetric Hashing for Fast Ranking via Neural Network Measures](#)

SL2G and Optimal Binary Function Search (OBFS)

Fast Item Ranking under Neural Network based Measures, **WSDM 2020**

DEFINITION 1. (OBFS) Let X and Y be subsets of Euclidean spaces (possibly with different dimensions), given a data set $S = \{x_1, \dots, x_n\} \subset X$ and a continuous binary functional, $f : X \times Y \rightarrow \mathbb{R}$, given $q \in Y$, OBFS aims to find

$$\arg \max_{x_i \in S} f(x_i, q). \quad (1)$$

- No strong assumptions for ranking measures, linear or nonlinear, convex or non-convex
- Traditional ANN Search and MIPS are special cases of OBFS
- Specifically, we focus on neural network based binary functional f

Neural ranking: Search on L2 Graph (SL2G)

Fast Item Ranking under Neural Network based Measures, **WSDM 2020**

To bypass constructing Delaunay graphs with respect to complicated binary functions, SL2G has two steps:

1. No matter what the given binary function f is, SL2G constructs a Delaunay graph (or an approximate one) with respect to ℓ_2 distance (which is defined on searching data X and independent of queries) in the indexing step.
2. In the searching step, SL2G performs the greedy search on this index graph by **the binary function f** .

BEGIN: Fast Neural Ranking by Bipartite Graph

Fast Neural Ranking on Bipartite Graph Indices, VLDB 2022

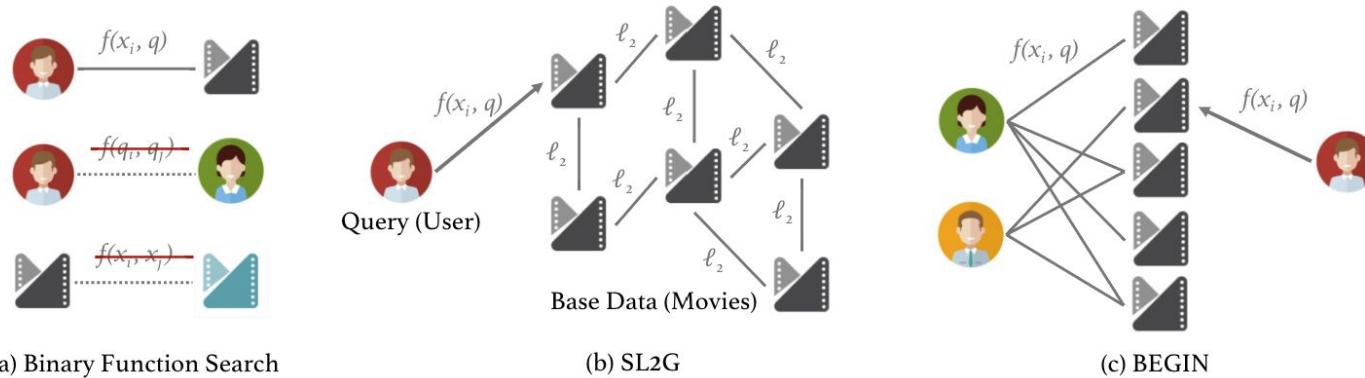


Figure 1: An example of the OBFS problem and solutions: we have collections of users and movies, and a binary function $f(x_i, q)$ learned on historical user-movie preference pairs. Given a user q and a movie x_i , the binary function $f(x_i, q)$ predicts the ranking score of this pair—how the user may like the movie. (a) Challenges in adapting ANN algorithms to the fast OBFS problem. The binary function is defined on movie-user pairs—no user-user nor movie-movie distance/similarity is defined. However, traditional ANN methods, e.g., proximity graph, require the distance between base data vectors to construct an index. (b) SL2G exploits ℓ_2 -graph to approximate the binary function search space. (c) Our proposed solution—BEGIN—builds a bipartite graph that leverages users to bridge relations among movies. This allows us to apply graph-based search algorithms on fast OBFS without knowing the distance between users $f(q_i, q_j)$ or the relationship between movies $f(x_i, x_j)$.

BEGIN Construction

Algorithm 1 BEGIN Construction

1: **input:** Base vector set S , sample query vector set Q , the maximum vertex degree M_x for base data, the maximum vertex degree M_q for queries, the priority queue size k for searching neighbors and the similarity measure $f(x, q)$.

2: Initialize graph $G = \emptyset$

3: **for** each x in S **do**

4: Greedy base data search k vertices $\{p_i\}$ on G by $SearchB(x, G, k, f)$ that have largest values with x in $f(x, p_i)$, place them in descending order.

5: $C \leftarrow \emptyset$. $H \leftarrow \emptyset$.

6: **for** $i \leftarrow 1$ to k **do**

7: **if** p_i not in H **then**

8: $C \leftarrow C \cup \{p_i\}$

9: Add all neighbors' neighbors of p_i to H .

10: add edge $\{x, p_i\}$ to G

11: **if** $|C| = M_x$ **then**

12: **break**

13: **for** each q in Q **do**

14: Greedy Query search k vertices $\{p_i\}$ on G by $SearchQ(q, G, k, f)$ that have largest values with q in $f(p_i, q)$, place them in descending order.

15: $C \leftarrow \emptyset$. $H \leftarrow \emptyset$.

16: **for** $i \leftarrow 1$ to k **do**

17: **if** p_i not in H **then**

18: $C \leftarrow C \cup \{p_i\}$

19: Add all neighbors' neighbors of p_i to H .

20: add edge $\{p_i, q\}$ to G

21: **if** $|C| = M_q$ **then**

22: **break**

23: **output:** index graph G

VLDB 2022

Results for Neural Network Measures

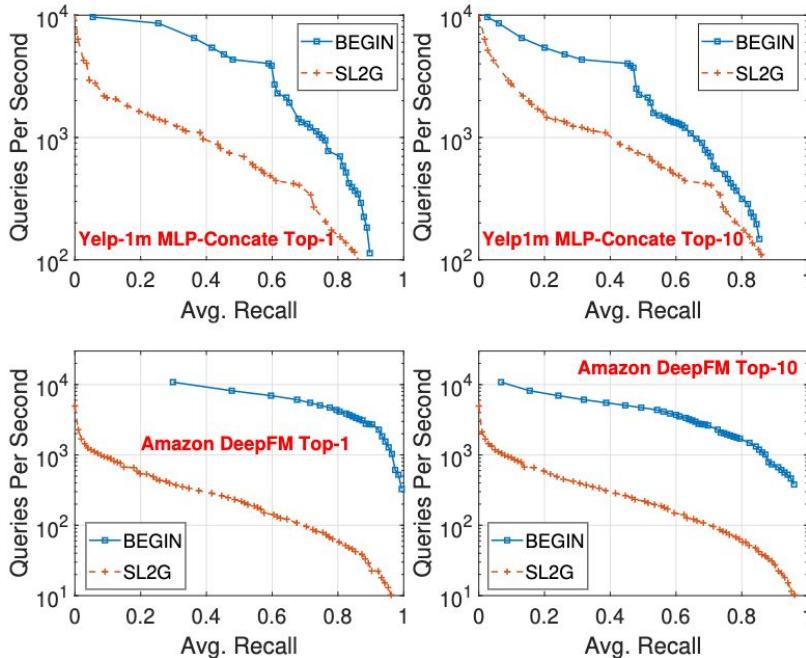


Figure 7: Experimental results for the neural network measures from the view of Recall vs. Time. The best results are in the upper right corner.

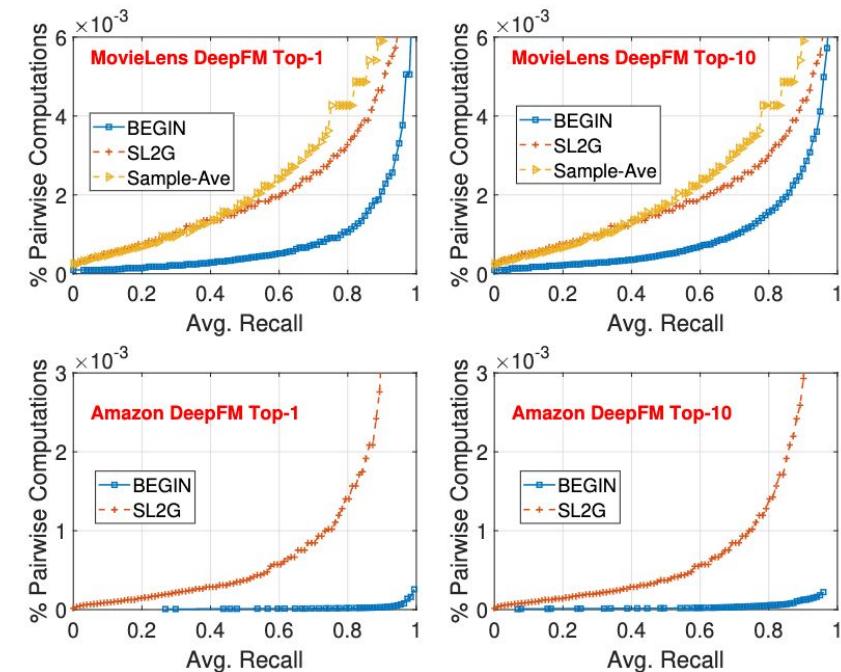


Figure 8: Experimental results for the neural network measures from the view of Recall vs. Computations. The best results are in the lower right corner.

Outline

1. Vector similarity functions
2. Vector compressions
3. Vector similarity search
4. Maximum inner product search (MIPS)
5. Fast neural ranking
- 6. GPU computing**
7. GCWSNet, hashing algorithms
8. Boosted trees, ABC-boost
9. Distributed, adaptive, and federated learning
10. Privacy
11. Security
12. Others: generative AI models, NLP, knowledge graphs, multi-modal, cross-modal, advertising

GPU Computing

1. GPU for HNSW
2. GPU fast neural ranking
3. GPU hierarchical parameter server for training massive-scale CTR models
4. GPU for feature processing
5. GPU for hashing

List of our relevant works on GPUs:

WWW'12, [GPU-Based Minwise Hashing](#)

CIKM'19, [AIBox: CTR Prediction Model Training on a Single Node](#)

MLSys'20, [Distributed Hierarchical GPU Parameter Server for Massive Scale Deep Learning Ads Systems](#)

ICDE'20, [SONG: Approximate Nearest Neighbor Search on GPU](#)

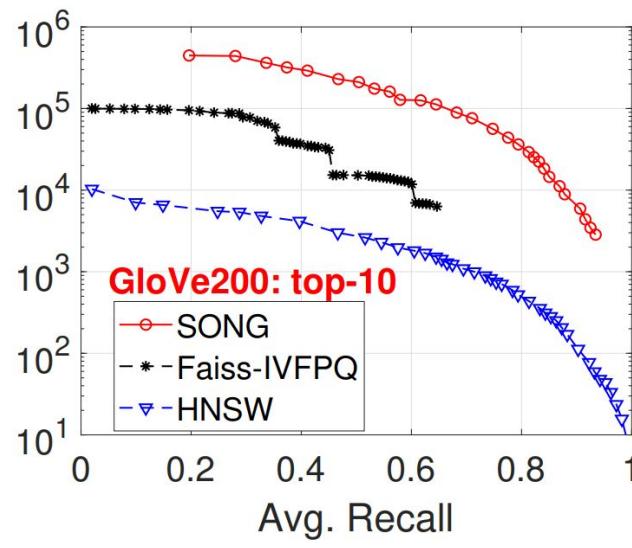
BIGDATA'22, [Communication-Efficient TeraByte-Scale Model Training Framework for Online Advertising](#)

BIGDATA'22, [FeatureBox: Feature Engineering on GPUs for Massive-Scale Ads Systems](#)

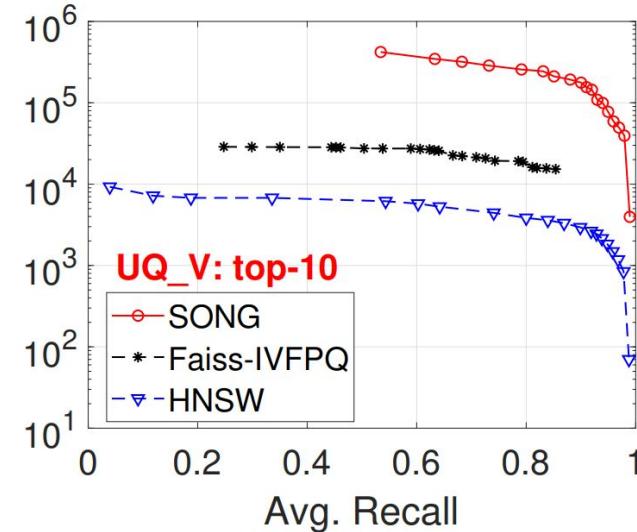
GPU for HNSW

ICDE 2020, [SONG: Approximate Nearest Neighbor Search on GPU](#)

- SONG (our GPU version of HNSW) can be close to 100 times faster than CPU HNSW
- SONG can be an order of magnitude faster than FAISS-IVFPQ (GPU)
- GPU HNSW saves the CPU-GPU data movement overhead when calling LLM



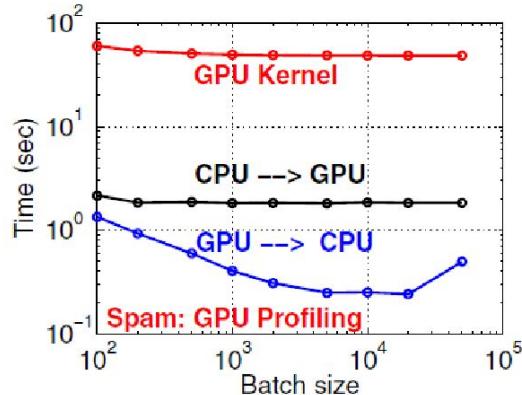
GloVe200: top-10



UQ_V: top-10

GPU for Minwise Hashing

GPU-Based Minwise Hashing, WWW 2012



b-Bit Minwise Hashing in Practice, Internetware 2013

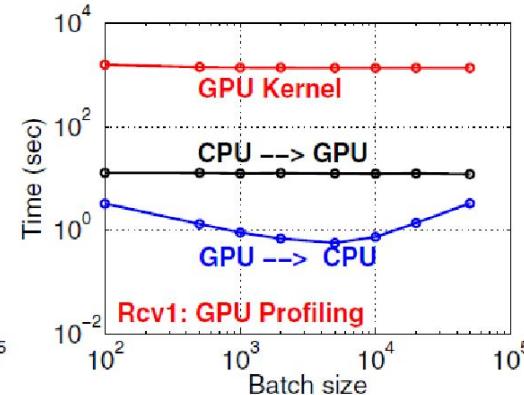


Figure 1: Overhead of three phases of GPU implementation.

Table 2: Data loading and preprocessing ($k = 500$) time (sec)

Dataset	Loading	Permu	2U	4U	2U (GPU)
Webspam	9.7×10^2	6.1×10^3	4.1×10^3	4.4×10^4	51
Rev1	1.0×10^4	—	3.0×10^4	—	1.4×10^3

GPU achieves 100-fold improvements. Example of “embarrassingly parallelizable”.

GPU for (Ads) CTR Model Training

1. (Ads) CTR models are still the major revenue source for AI. CTR = click-through rate
2. CTR models can easily have trillion (1000 billion) parameters, even in 2014.
3. Starting in 2018, training CTR models on GPUs became a reality, through innovations.
4. The key is to carefully design a GPU-CPU-Disk (SSD) hierarchical engine.

Hierarchical GPU parameter server for training massive CTR models. A major breakthrough in industry. The initial version called “AIBox” was mentioned in [NVIDIA](#) Jensen Huang’s [talk in 2019](#). See [media report](#) and publications in [CIKM’19](#) (single GPU-box), [MLSys’20](#) (multi GPU-box), [BIGDATA’22](#) (GPU feature processing), [BIGDATA’22](#) (GPU adaptive training), etc.

GPU for (Ads) CTR Model Training

CIKM 2019

AIBox: CTR Prediction Model Training on a Single Node

Weijie Zhao¹, Jingyuan Zhang¹, Deping Xie², Yulei Qian², Ronglai Jia², Ping Li¹

¹Cognitive Computing Lab, Baidu Research USA

²Baidu Search Ads (Phoenix Nest), Baidu Inc.

1195 Bordeaux Dr, Sunnyvale, CA 94089, USA

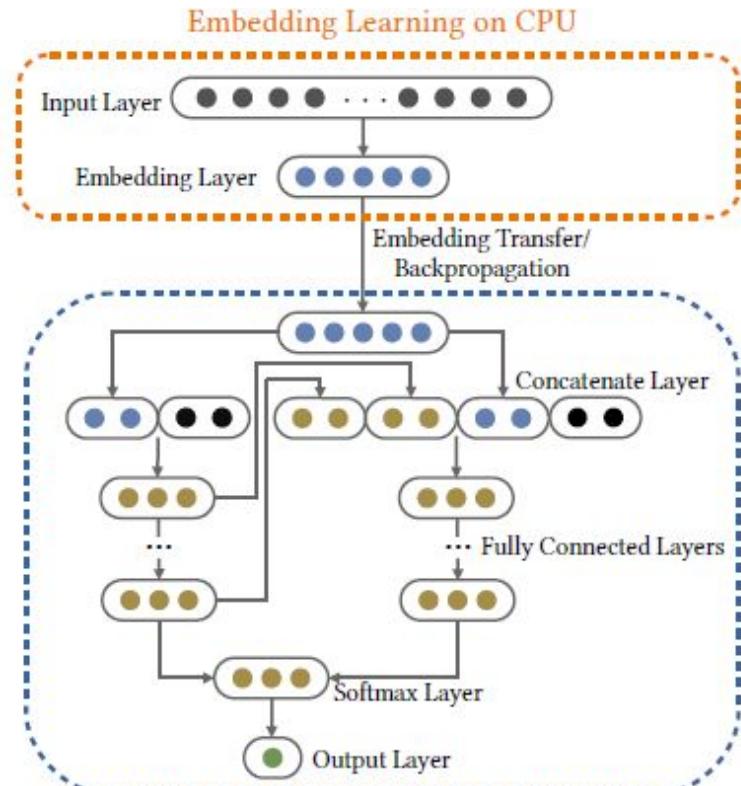
No.10 Xibeiwang East Road, Beijing, 10085, China

10900 NE 8th St, Bellevue, WA 98004, USA

{weijiezha,zhangjingyuan03,xiedeping01,qianyulei,jiaronglai,liping11}@baidu.com

CIKM 2019: Mixed CPU-GPU training.

Massive 10TB-parameter layer trained on CPUs. Offloaded to SSDs.



GPU for (Ads) CTR Model Training

MLSys'20

DISTRIBUTED HIERARCHICAL GPU PARAMETER SERVER FOR MASSIVE SCALE DEEP LEARNING ADS SYSTEMS

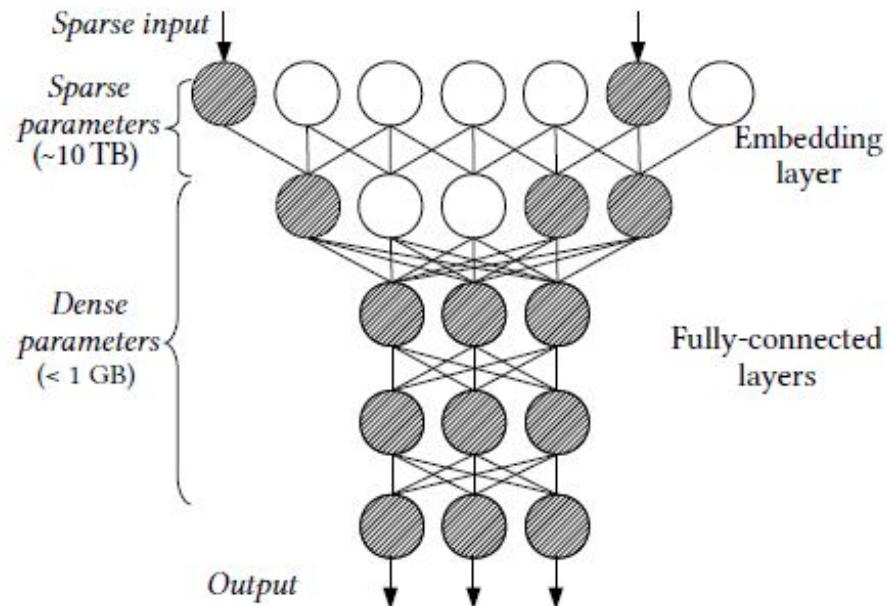
Weijie Zhao¹, Deping Xie², Ronglai Jia², Yulei Qian², Ruiquan Ding³, Mingming Sun¹, Ping Li¹

¹ Cognitive Computing Lab, Baidu Research

² Baidu Search Ads (Phoenix Nest), Baidu Inc.

³ Sys. & Basic Infra., Baidu Inc.

{weijiezha, xiedeping01, jiaronglai, qianyulei, dingruiquan, sunmingming01, liping11}@baidu.com

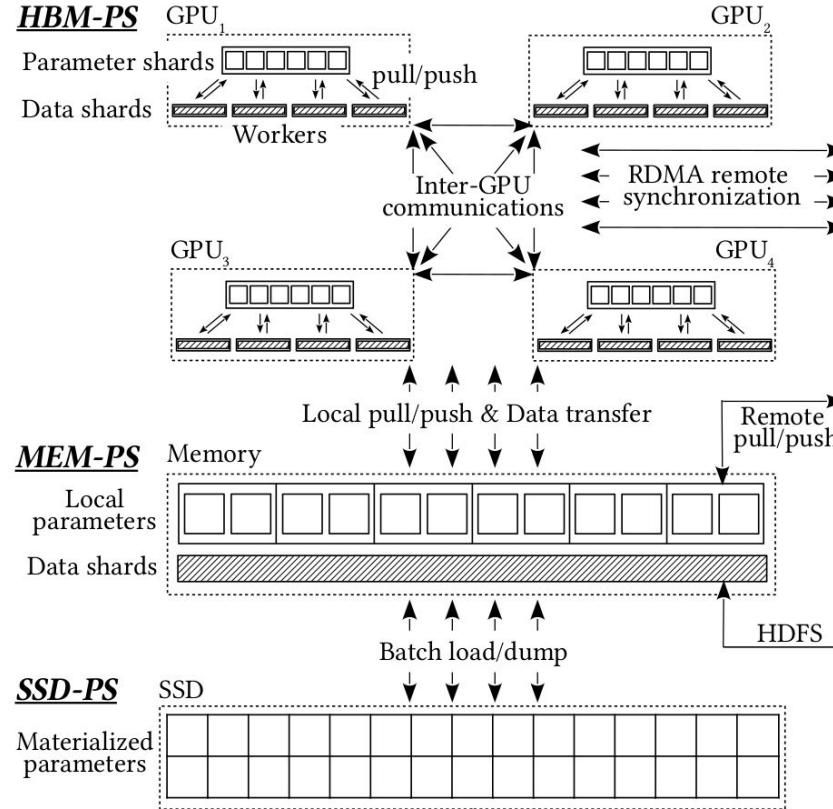


Train the massive layer also on GPUs

GPU for (Ads) CTR Model Training

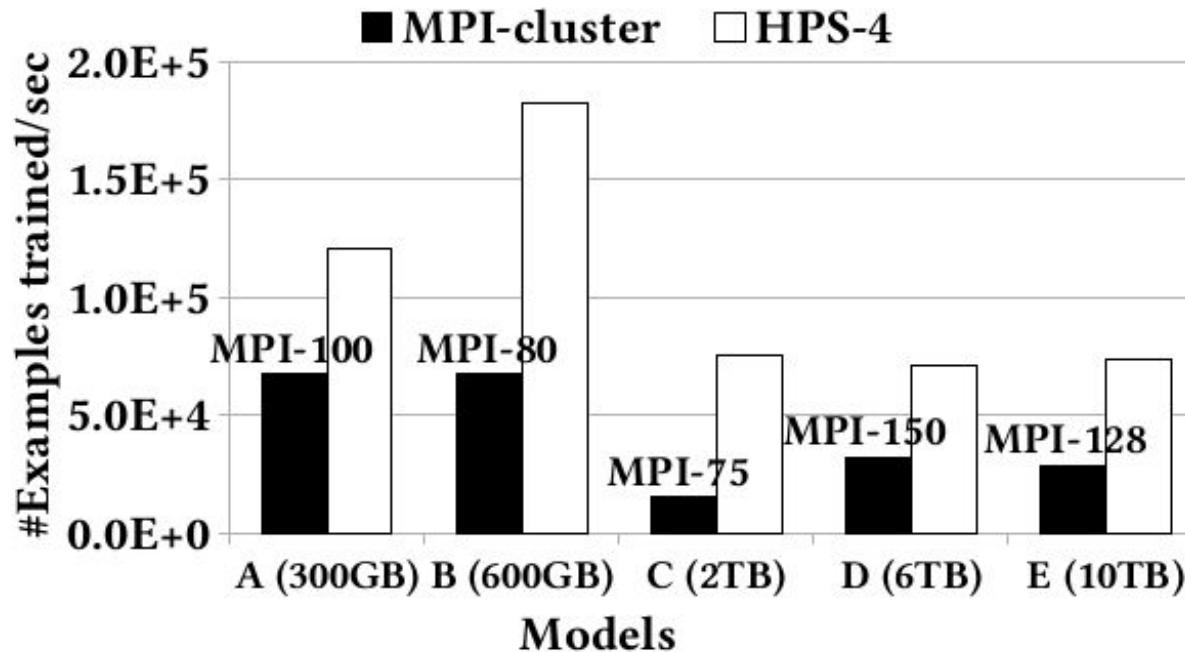
MLSys'20

Key architecture



GPU for (Ads) CTR Model Training

MLSys'20



GPU (HPS 4 boxes) solution can be 2-4 times more efficient compared to CPU (MPI) cluster solution

GPU for Feature Engineering

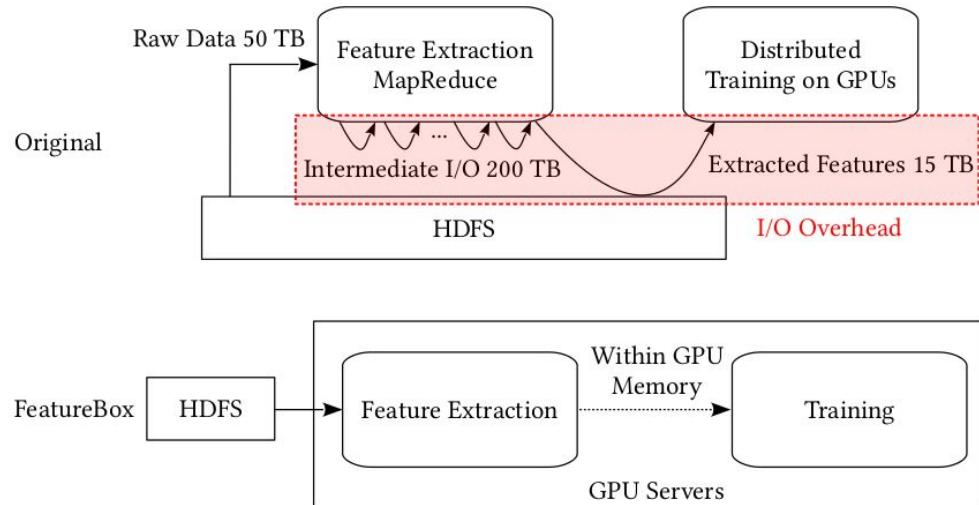
BIGDATA 2022

FeatureBox: Feature Engineering on GPUs for Massive-Scale Ads Systems

Weijie Zhao, Xuewu Jiao, Xinsheng Luo, Jingxue Li, Belhal Karimi, Ping Li

Cognitive Computing Lab, Baidu Research
Baidu Search Ads (Phoenix Nest), Baidu Inc.
10900 NE 8th St. Bellevue, Washington 98004, USA
No. 10 Xibeiwang East Road, Beijing 100193, China

{weijiezhao, jiaoxuewu, luoxinsheng, lijingxue01, belhalkarimi, liping11}@baidu.com



Reduce excessive I/O between HDFS and computing nodes

GPU for Feature Engineering

BIGDATA 2022

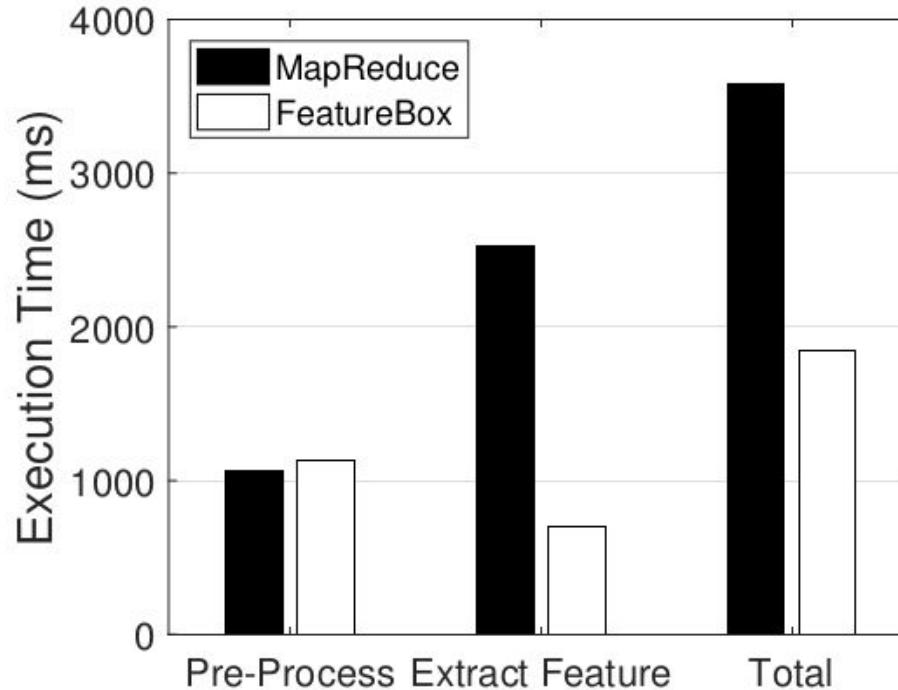


Fig. 6. Feature extraction time of MapReduce and FeatureBox.

GPU Ads CTR Models with Local Training

Communication-Efficient TeraByte-Scale Model Training
Framework for Online Advertising

Weijie Zhao¹, Xuewu Jiao², Mingqing Hu², Xiaoyun Li¹, Xiangyu Zhang², Ping Li¹

¹ Cognitive Computing Lab, Baidu Research

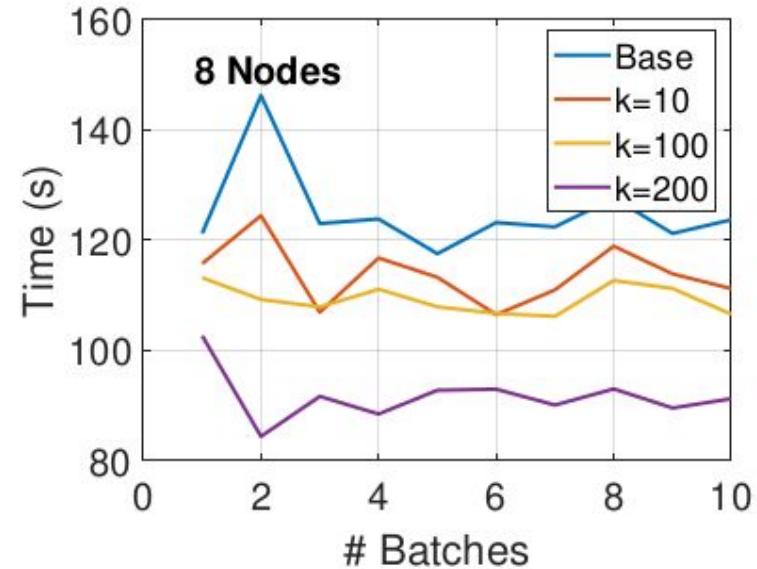
² Baidu Search Ads (Phoenix Nest), Baidu Inc.
10900 NE 8th St. Bellevue, Washington 98004, USA
No. 10 Xibeiwang East Road, Beijing 100193, China

Algorithm 2 k -step Adam (with N workers)

```
1: Input: learning rate  $\alpha$ ,  $\beta_1$ ,  $\beta_2$ 
2: Initialize: local models  $x_{0,i}$ , moment estimators  $m_{0,i} = 0$ ,  $v_{0,i} = \epsilon \mathbf{1}$ ,  $i = 1, \dots, N$ 
3: for  $t = 1, \dots, T$  do
4:   Compute stochastic gradient  $g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$ 
5:    $m_{t,i} \leftarrow \beta_1 m_{t-1,i} + (1 - \beta_1) g_{t,i}$ 
6:    $v_{t,i} \leftarrow \beta_2 v_{t-1,i} + (1 - \beta_2) g_{t,i}^2$ 
7:   if  $t \bmod k \neq 0$  then
8:      $v_t = v_{t-1}$ 
9:      $x_{t+1,i} \leftarrow x_{t,i} - \alpha \frac{m_{t,i}}{\sqrt{v_t}}$ 
10:    //Local Adam Update
11:   else
12:      $v_t = \frac{1}{N} \sum_{i=1}^N v_{t,i}$ 
13:      $x_{t+1,i} \leftarrow \frac{1}{N} \sum_{j=1}^N \left( x_{t,j} - \alpha \frac{m_{t,j}}{\sqrt{v_t}} \right)$ 
14:     //Global averaging
15:   end if
16: end for
```

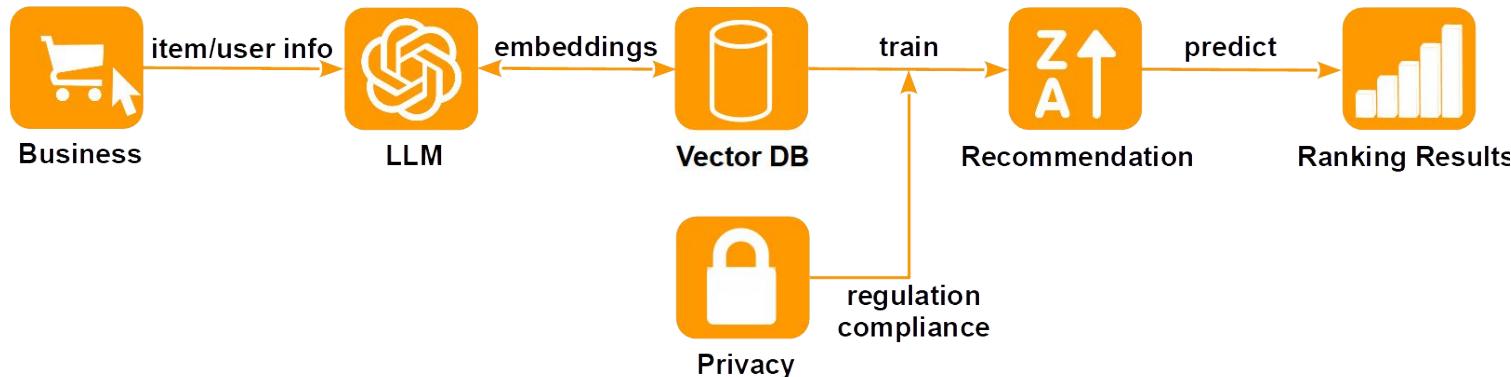
BIGDATA 2022

K-step local adam updates reduces the training time.



The Role of Vector DBs in Search and Ads

1. Embeddings have become the crucial component in search and ads.
2. Embedding-based retrieval (EBR) is the key step for retrieval.
3. However, EBR, ANN, Vector DBs etc can only provide only very crude results and hence they can only serve as an important intermediate step in the pipeline of ads and search.
4. AI models for generating embeddings and models for prediction/rankings are crucial. We will focus on big AI models (as well as privacy) for the rest of the presentation.

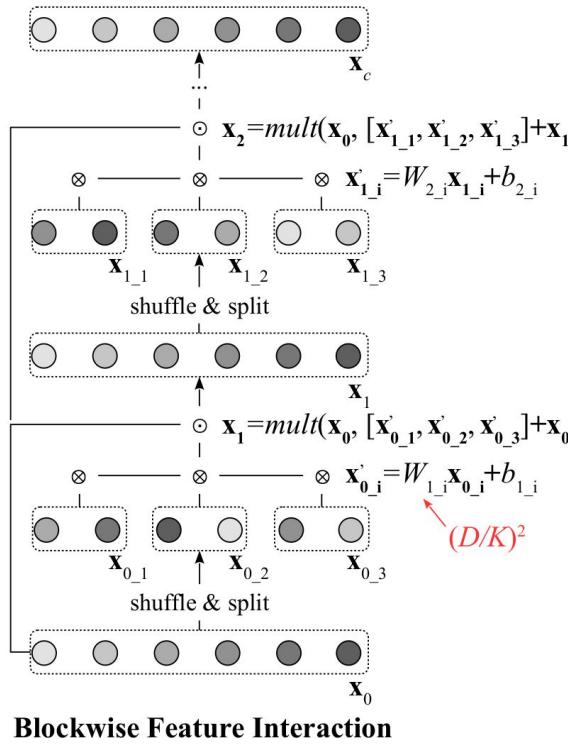
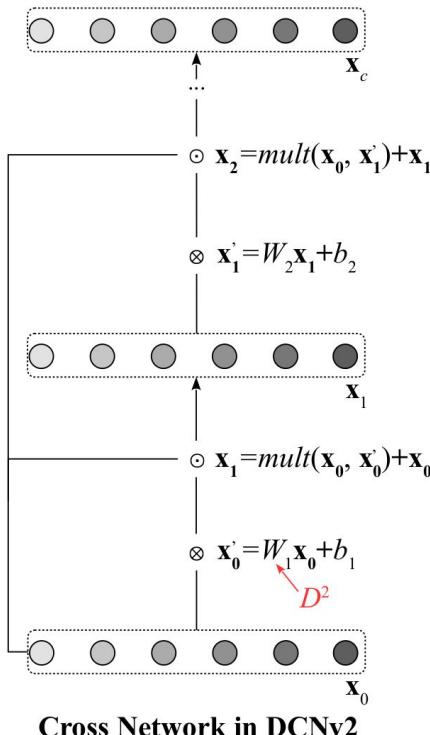


A Simple Demo

- Amazon movie review: LLM + embedding + AI models :
 - 11,000 movie reviews (raw texts) => LLM => 11,000 embeddings in 384 dimensions.
 - 1,000 embeddings as test (query) vectors , 10,000 base vectors.
 - HNSW and KNN classifiers to predict review ratings (1-5) => 60% accuracy.
 - ABC-Boost trees with 10,000 base vectors for training => 70% accuracy.
 - MLP neural networks => 70% accuracy.
 - MLP neural networks + DCNV2 (deep feature crossing) => 71%.
 - MLP neural networks + BFI (8-block clockwise feature interactions) => 71%.

Only using embeddings and their similarities is typically not sufficient. Many applications will need to build AI models on top of the embeddings from LLM or other methods.

BFI: Blockwise Feature Interaction

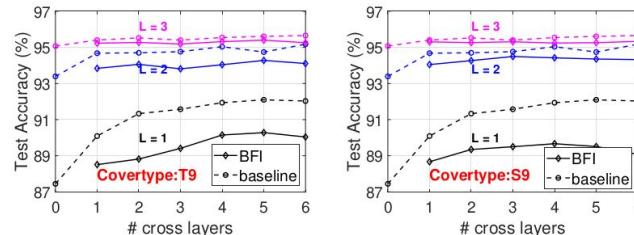
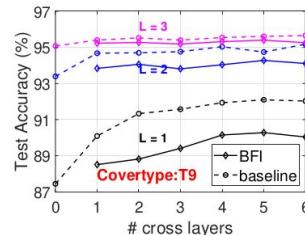
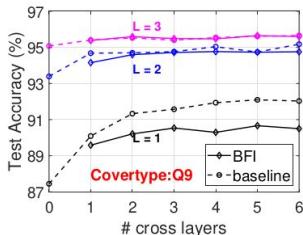
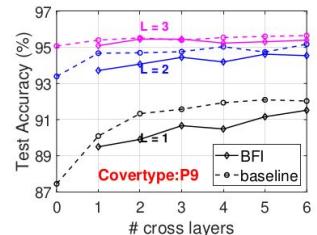
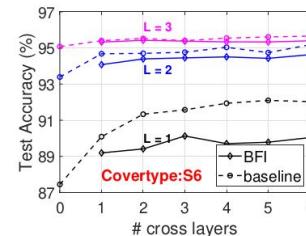
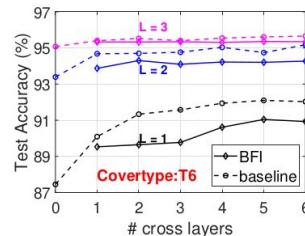
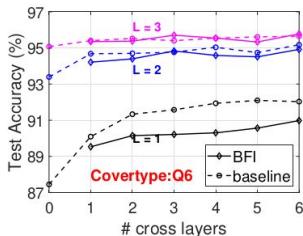
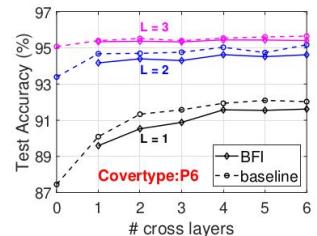
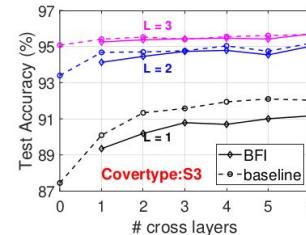
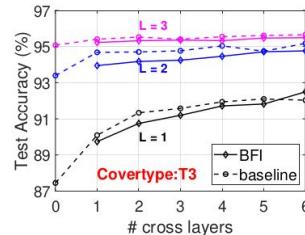
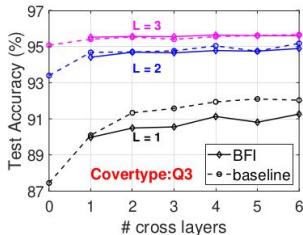
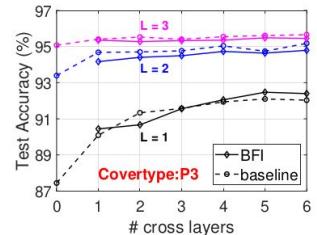


DCNv2 uses D2 parameters for each cross layer

Blockwise Feature Interaction (BFI) shuffles the features and partition them into blocks to compute feature interactions in a smaller scale

The output of each block are merged and reshuffled. The feature interaction across multiple blocks is addressed through multiple cross layers with this reshuffle

BFI: Blockwise Feature Interaction



We implemented 4 variants of BFI: P, Q, T, S.

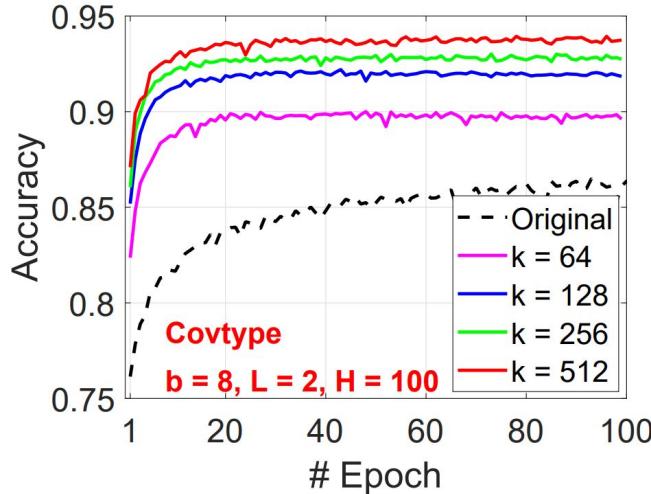
P does the most work and usually performs the best.

T6 further reduces the number of parameters by a factor of 36.

Outline

1. Vector similarity functions
2. Vector compressions
3. Vector similarity search
4. Maximum inner product search (MIPS)
5. Fast neural ranking
6. GPU computing
7. **GCWSNet, hashing algorithms**
8. Boosted trees, ABC-boost
9. Distributed, adaptive, and federated learning
10. Privacy
11. Security
12. Others: generative AI models, NLP, knowledge graphs, multi-modal, cross-modal, advertising

Lightweight and High-Accuracy Deep Learning



GCWSNet

Dashed curve: test accuracy with a 2-hidden-layer neural net on the original data. The best accuracy is about **0.86**.

$k = 64$ solid curve: using **CWS hashing** as features, the accuracy can be improved to **0.9** with faster convergence.

Using more (larger k) hashes improves the accuracy. Training with CWS hashing converges much faster too.

GCWSNet

CIKM 2022

Algorithm 1 Generalized consistent weighted sampling (GCWS) for hashing the pGMM kernel.

Input: Data vector u_i ($i = 1$ to D)

Generate vector \tilde{u} in $2D$ -dim by (1).

For i from 1 to $2D$

$$r_i \sim \text{Gamma}(2, 1), c_i \sim \text{Gamma}(2, 1), \beta_i \sim \text{Uniform}(0, 1)$$

$$t_i \leftarrow \lfloor p \frac{\log \tilde{u}_i}{r_i} + \beta_i \rfloor, a_i \leftarrow \log(c_i) - r_i(t_i + 1 - \beta_i)$$

End For

Output: $i^* \leftarrow \arg \min_i a_i, t^* \leftarrow t_{i^*}$

Given another data vector v , we feed it to GCWS using the same set of random numbers: r_i, c_i, β_i . To differentiate the hash samples, we name them, respectively, (i_u^*, t_u^*) and (i_v^*, t_v^*) . We first present the basic probability result as the following theorem.

Theorem 1.

$$P[(i_u^*, t_u^*) = (i_v^*, t_v^*)] = pGMM(u, v). \quad (5)$$

GCWSNet

CIKM 2022

$$\begin{cases} \tilde{u}_{2i-1} = u_i, \quad \tilde{u}_{2i} = 0 & \text{if } u_i > 0 \\ \tilde{u}_{2i-1} = 0, \quad \tilde{u}_{2i} = -u_i & \text{if } u_i \leq 0 \end{cases}$$

$$GMM(u, v) = \frac{\sum_{i=1}^{2D} \min\{\tilde{u}_i, \tilde{v}_i\}}{\sum_{i=1}^{2D} \max\{\tilde{u}_i, \tilde{v}_i\}}.$$

$$pGMM(u, v; p) = \frac{\sum_{i=1}^{2D} (\min\{\tilde{u}_i, \tilde{v}_i\})^p}{\sum_{i=1}^{2D} (\max\{\tilde{u}_i, \tilde{v}_i\})^p}$$

GMM kernels produce
accurate classification results

Dataset	# train	# test	# dim	# class	linear	RBF	GMM	pGMM (p)
SEMG	1800	1,800	2,500	6	19.3	29.0	54.0	56.1 (2)
DailySports	4,560	4,560	5,625	19	77.7	97.6	99.6	99.6 (0.6)
M-Noise1	10,000	4,000	784	10	60.3	66.8	71.4	85.2 (80)
M-Image	12,000	50,000	784	10	70.7	77.8	80.9	89.5 (50)
PAMAP101	188,209	188,208	51	20	75.3	—	—	—
Covtype	290,506	290,506	54	7	71.5	—	—	—

From Hashed Data to Features

CIKM 2022

$$P[i_u^* = i_v^*] \approx P[(i_u^*, t_u^*) = (i_v^*, t_v^*)] = pGMM(u, v)$$

Hashed data are ID (categorical) features and should be expanded by one-hot representation.

For example, with $k = 3$ hashes (3, 0, 1), each in $b = 2$ bits, we can expand them to be 12-dims

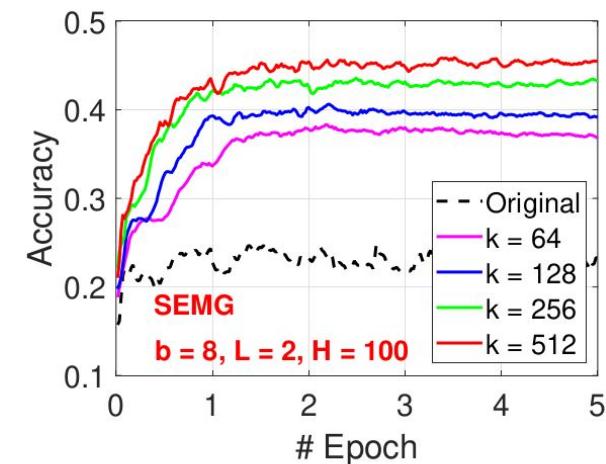
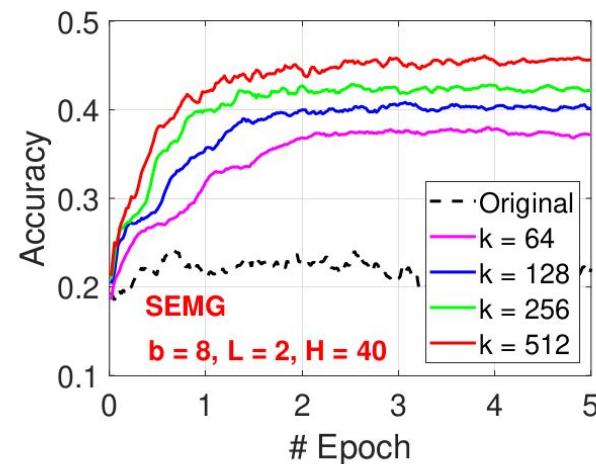
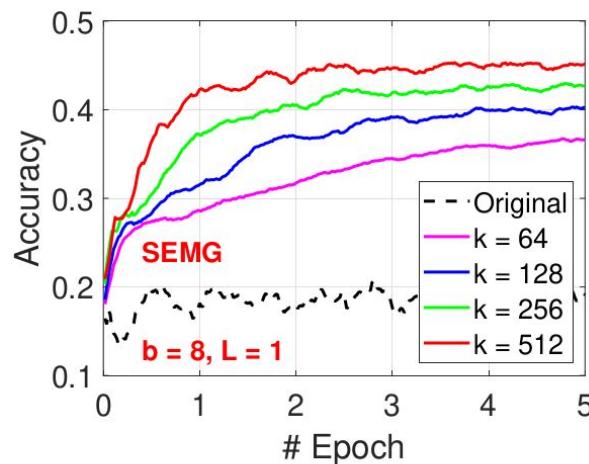
$$(3, 0, 1) \quad \Rightarrow \quad [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1 \ 0]$$

GCWSNet has two main parameters: k = number of hashes, and b = number of bits per hash

GCWSNet: CWS + Neural Nets

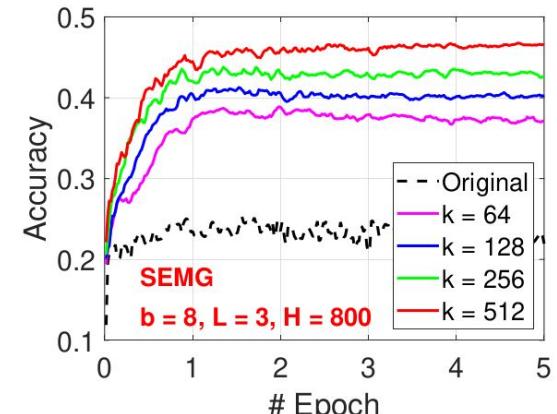
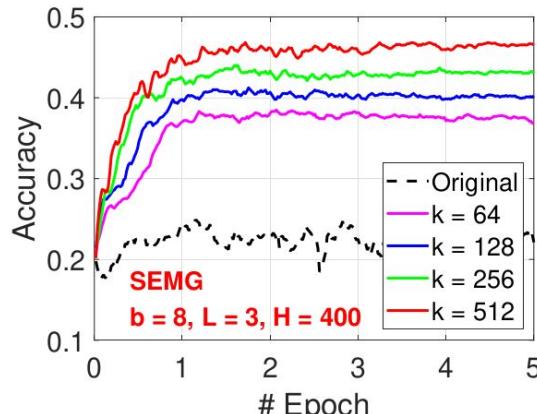
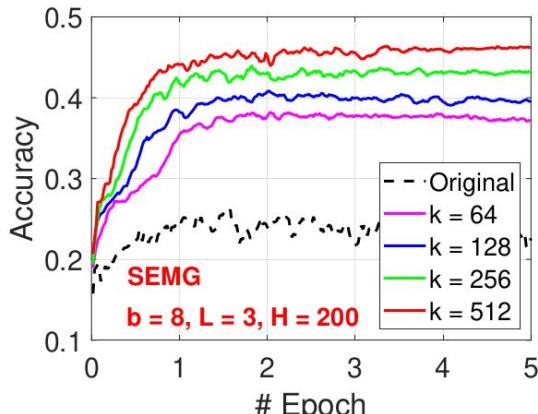
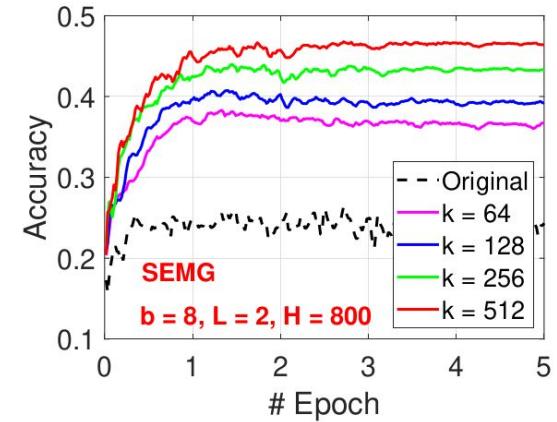
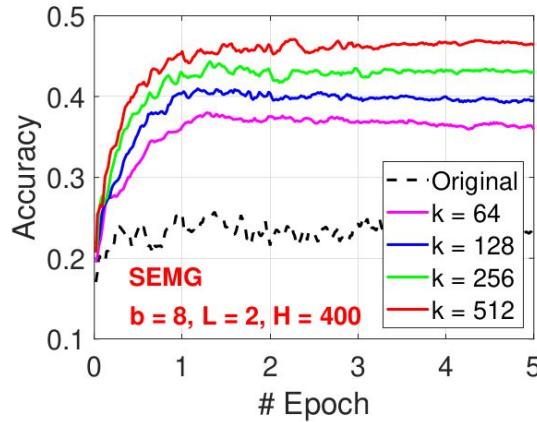
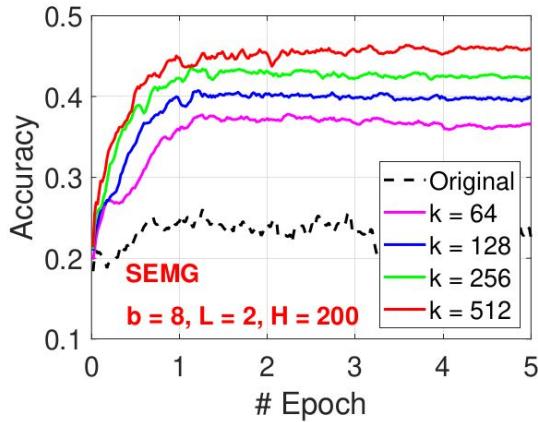
CIKM 2022

- Classification on the SEMG dataset using neural nets with L layers and H hidden units.
 $L = 1$: logistic regression. $L = 2$ means one hidden layer.
- Dashed (black) curve = the original data
- GCWSNet with $k = \{64, 128, 256, 512\}$ hashes and $b = 8$ bits for each hash.
- For this dataset, GCWSNet substantially improves



GCWSNet: CWS + Neural Nets

CIKM 2022

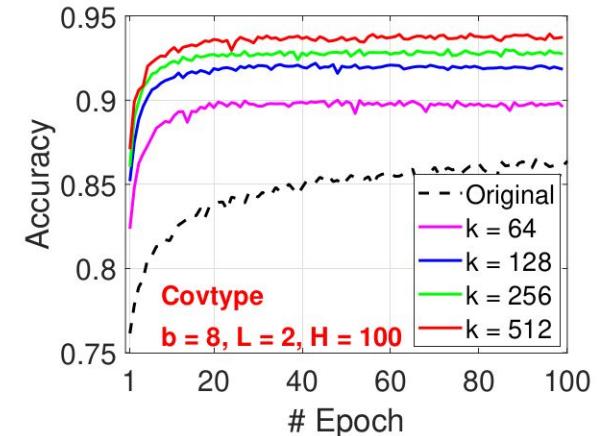
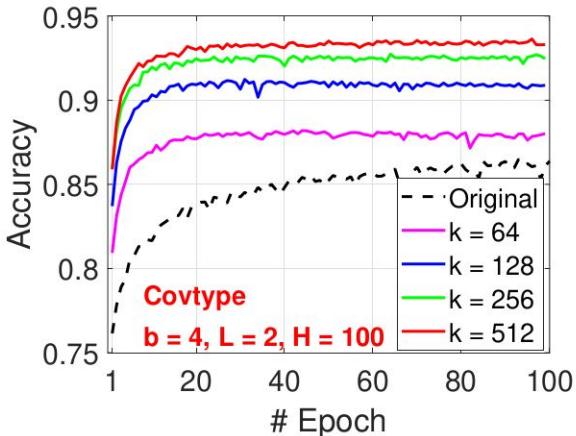
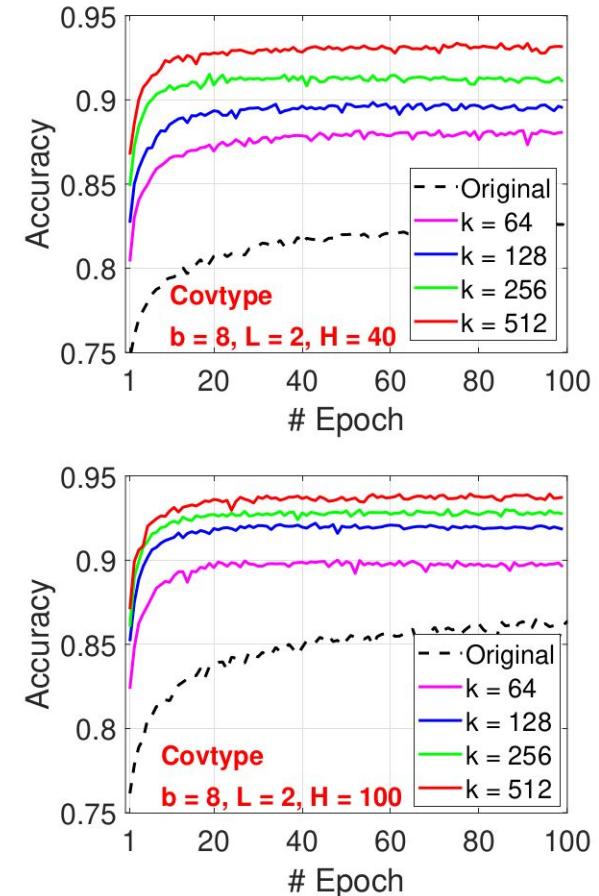
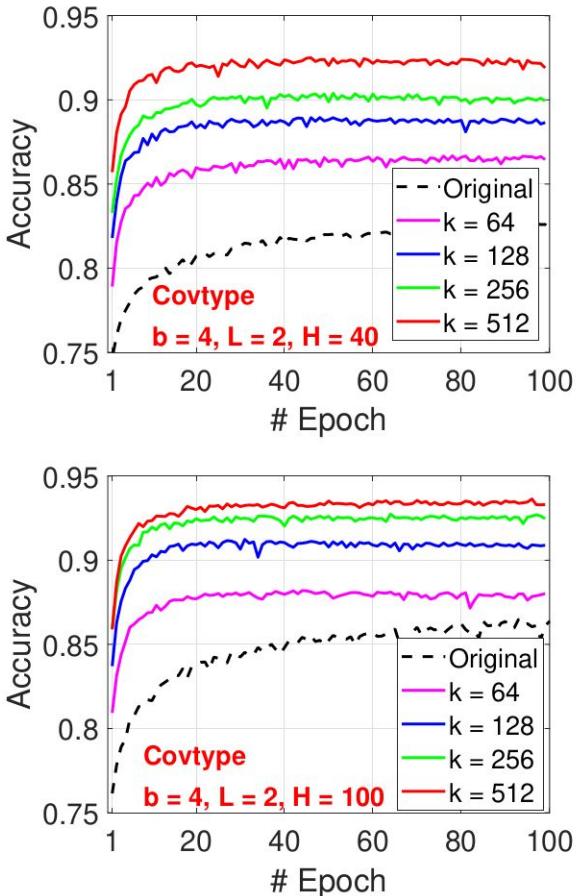


GCWSNet: CWS + Neural Nets

CIKM 2022

On UCI Covtype dataset,
GCWSNet substantially
improves the accuracy.

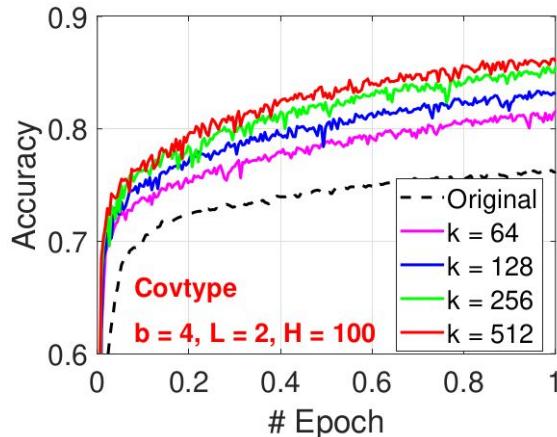
GCWSNet also converges
much faster.



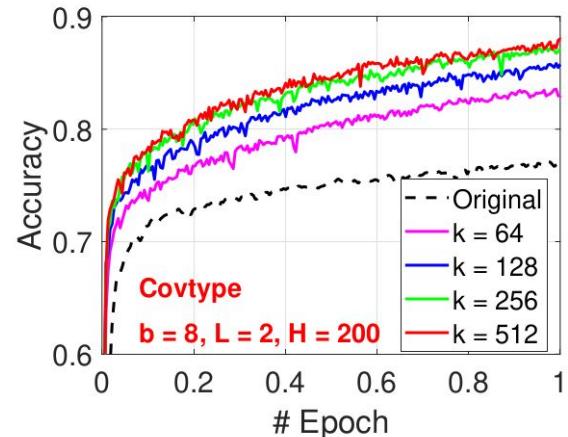
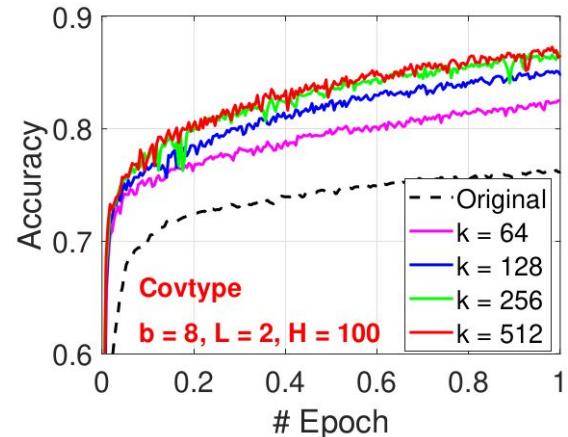
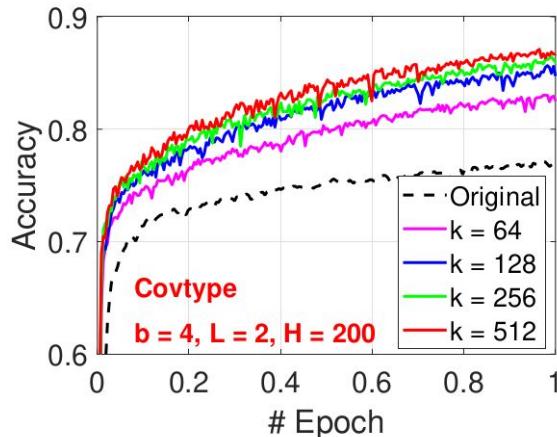
GCWSNet: CWS + Neural Nets

CIKM 2022

Accuracy at the first epoch.

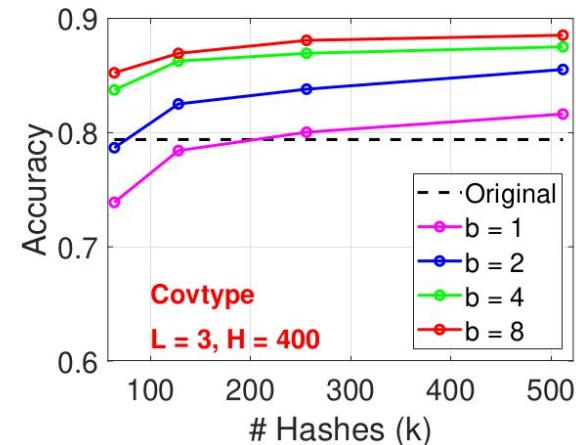
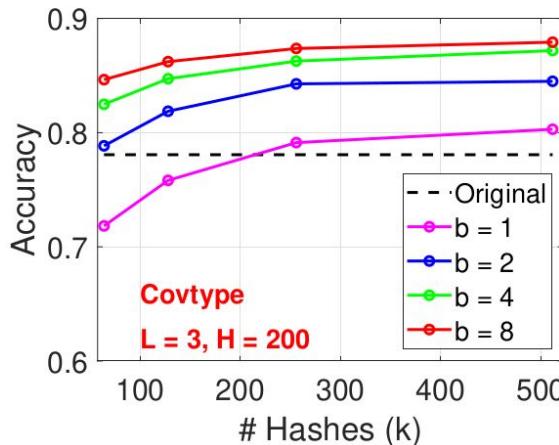
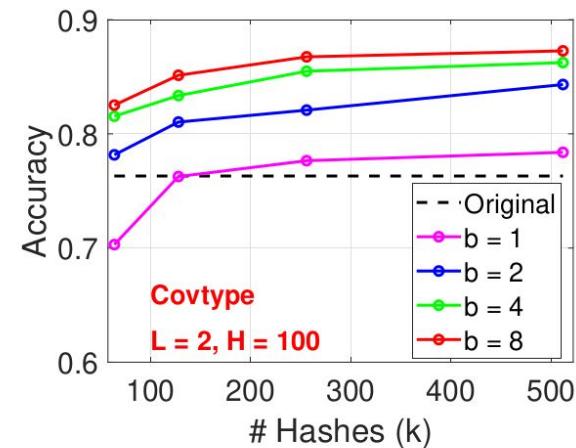
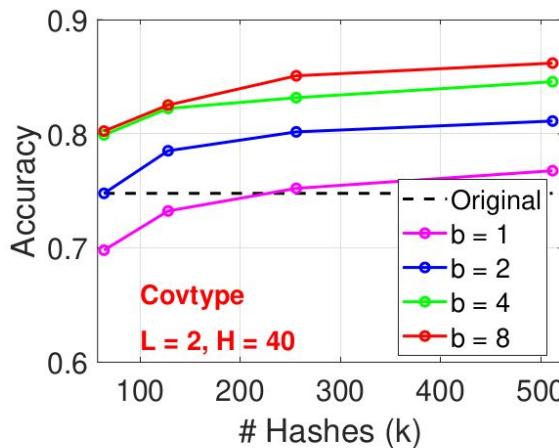


GCWSNet converges fast.



GCWSNet: the Effect of b (# bits)

CIKM 2022



GCWSNet: Stable Power Transformation

CIKM 2022

Algorithm 1 Generalized consistent weighted sampling (GCWS) for hashing the pGMM kernel.

Input: Data vector u_i ($i = 1$ to D)

Generate vector \tilde{u} in $2D$ -dim by (1).

For i from 1 to $2D$

$r_i \sim \text{Gamma}(2, 1)$, $c_i \sim \text{Gamma}(2, 1)$, $\beta_i \sim \text{Uniform}(0, 1)$

$t_i \leftarrow \lfloor p \frac{\log \tilde{u}_i}{r_i} + \beta_i \rfloor$, $a_i \leftarrow \log(c_i) - r_i(t_i + 1 - \beta_i)$

End For

Output: $i^* \leftarrow \arg \min_i a_i$, $t^* \leftarrow t_{i^*}$

$$pGMM(u, v; p) = \frac{\sum_{i=1}^{2D} (\min\{\tilde{u}_i, \tilde{v}_i\})^p}{\sum_{i=1}^{2D} (\max\{\tilde{u}_i, \tilde{v}_i\})^p}$$

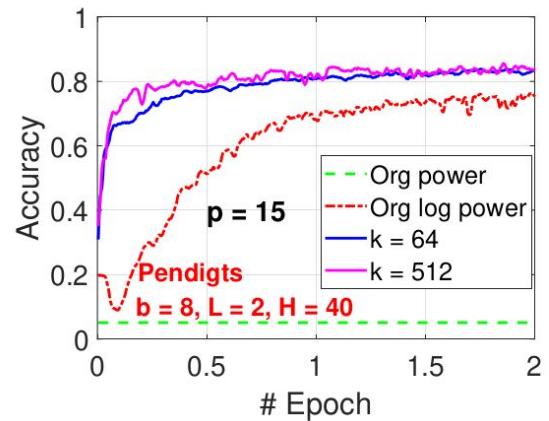
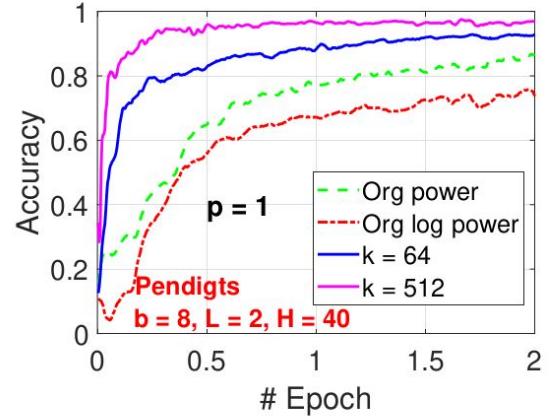
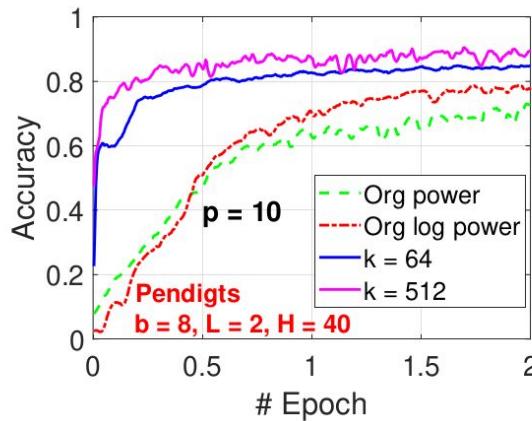
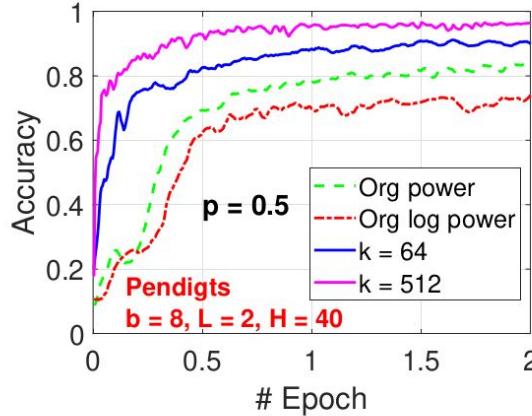
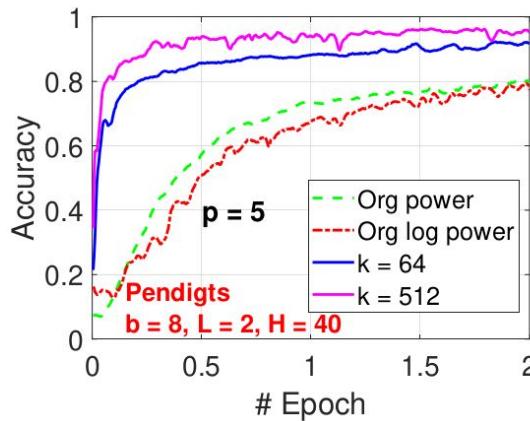
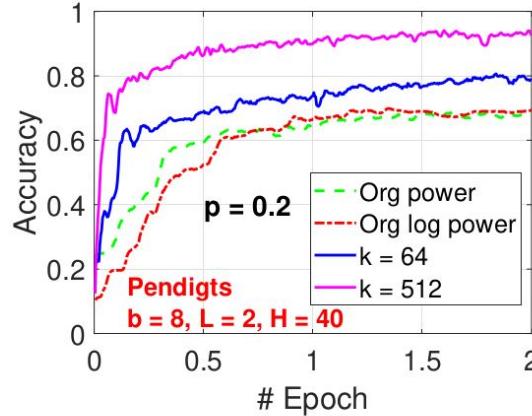
It is mathematically equivalent to applying power transformation on the original data.

However, GCWSNet (with parameter **p**) would be substantially more stable.

GCWSNet: Stable Power Transformation

CIKM 2022

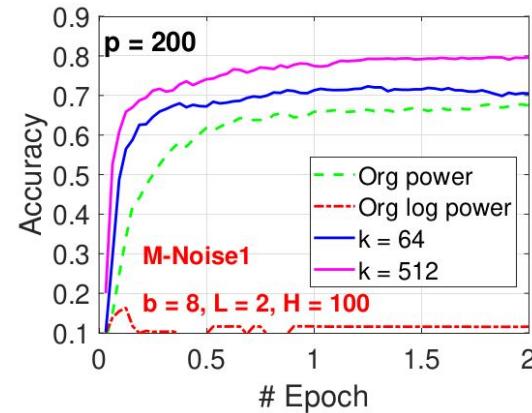
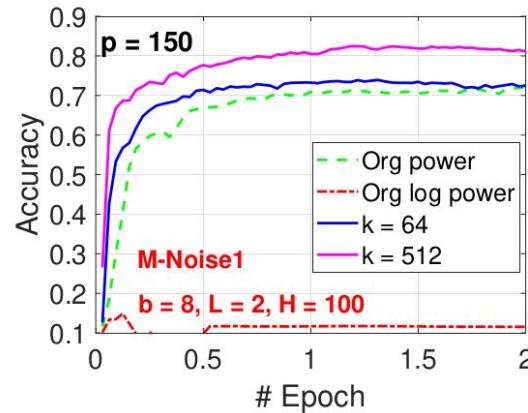
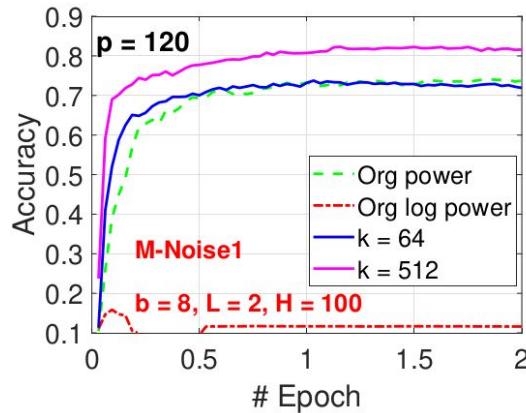
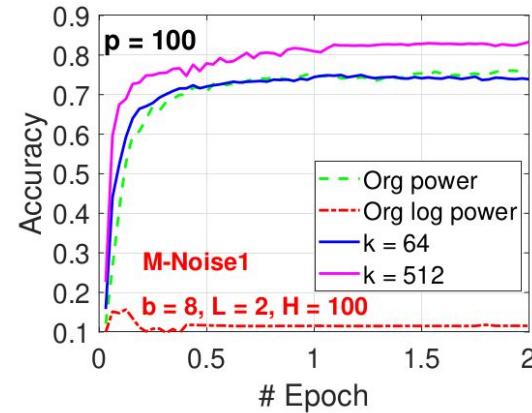
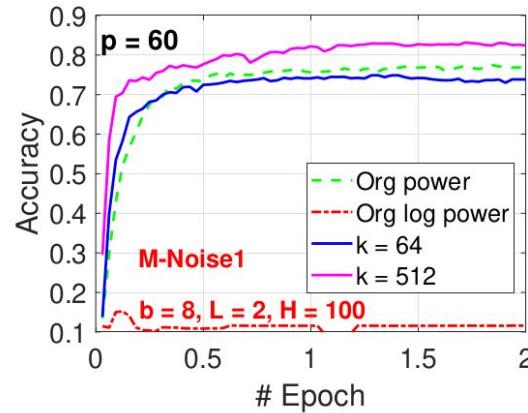
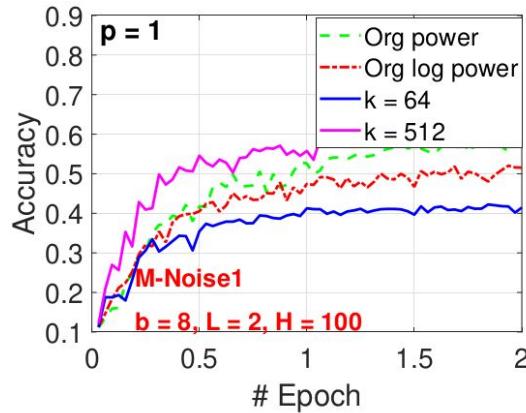
Pendigits dataset have integer values.



GCWSNet: Stable Power Transformation

CIKM 2022

M-Noise1 dataset has values in $[0, 1]$



GCWSNet versus NRFF

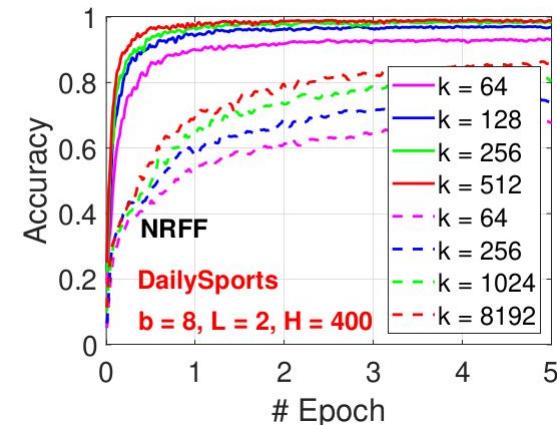
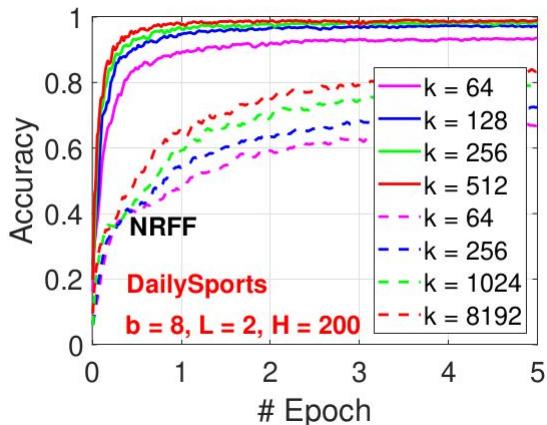
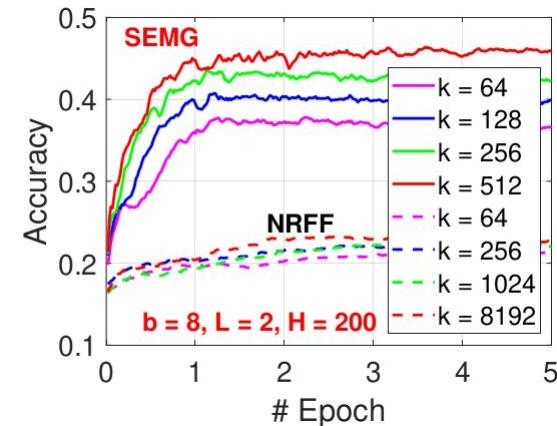
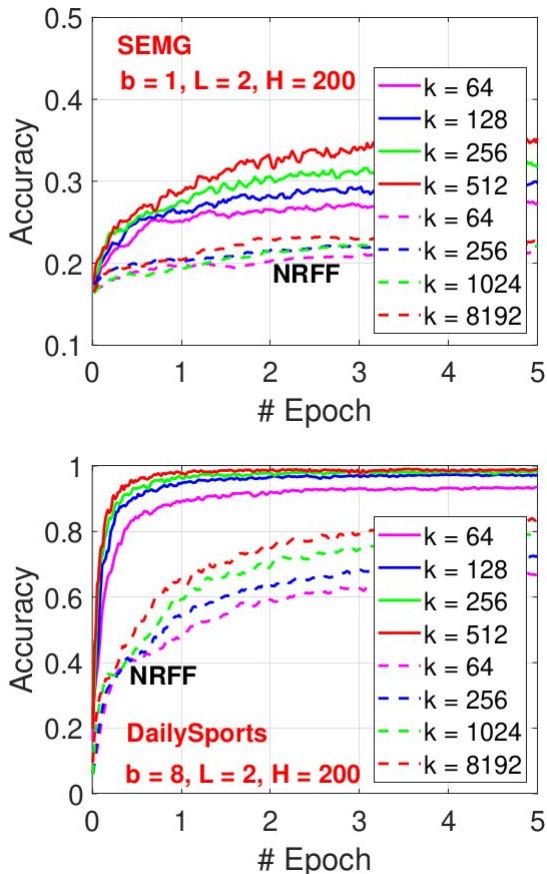
CIKM 2022

NRFF

= normalized random Fourier features

RFF and NRFF perform very poorly compared to GCWSNet.

Even with $k = 8192$ hashes (dashed), NRFF still performs very poorly.



Other Hashing Methods

- Conditional Random Sampling (CRS, smallest-k sketch)
- Minwise hashing
- One permutation hashing (OPH)
- Circulant minwise hashing (C-MinHash)
- Partitioned b-bit hashing (Pb-Hash)
- Extremal processes

Term-based Representations?

- Still very useful and should not be ignored.
- Simple and can be powerful if used properly, e.g., with query augmentation/expansion.
- Often very high-dimensional and even the storage can be a bottleneck if materialized.
- Hashing can be effective if good hashing methods are adopted. Since data are highly sparse, their non-zero locations themselves carry strong information. Making good use of the prior information is beneficial. Certain popular hashing methods like random projections “destroy” the sparse structure and would not expect to work well on sparse data.

A fun experiment in 2004/2005 at Microsoft Research

Table 1

We expected the counts for single words are exact and stored.

Page hits for a few high-frequency words and a few low-frequency words

Query	Hits (MSN.com)	Hits (Google)
A	2,452,759,266	3,160,000,000
The	2,304,929,841	3,360,000,000
Kalevala	159,937	214,000
Griseofulvin	105,326	149,000
Saccade	38,202	147,000



Ken Church

<https://www.linkedin.com/in/kenneth-church-a902772>

Table 2

Estimates of page hits are not always consistent. Joint frequencies ought to decrease monotonically as we add terms to the query, but estimates produced by current state-of-the-art search engines sometimes violate this invariant.

Query	Hits (MSN.com)	Hits (Google)
America	150,731,182	393,000,000
America, China	15,240,116	66,000,000
America, China, Britain	235,111	6,090,000
America, China, Britain, Japan	154,444	23,300,000

We expected the counts for the intersections of two or more words are estimated because there would be way too many.

A fun experiment in 2004/2005 at Microsoft Research

Table 3

This table illustrates the usefulness of joint counts in query planning for databases. To minimize intermediate writes, the optimal order of joins is: ((“Schwarzenegger” \cap “Austria”) \cap “Terminator”) \cap “Governor,” with 136,000 intermediate results. The standard practice starts with the least frequent terms, namely, ((“Schwarzenegger” \cap “Terminator”) \cap “Governor”) \cap “Austria,” with 579,100 intermediate results.

	Query	Hits (Google)	
One-way	Austria	88,200,000	Query execution optimization: Suppose the goal is to find all the intersections among 4 words.
	Governor	37,300,000	
	Schwarzenegger	4,030,000	
	Terminator	3,480,000	
Two-way	Governor, Schwarzenegger	1,220,000	Intuitively we should start with two “shortest” words. But in this example, the two shortest words are almost semantically identical.
	Governor, Austria	708,000	
	Schwarzenegger, Terminator	504,000	
	Terminator, Austria	171,000	
	Governor, Terminator	132,000	
	Schwarzenegger, Austria	120,000	
Three-way	Governor, Schwarzenegger, Terminator	75,100	It would be a lot better to start with “Schwarzenegger” and “Austria”
	Governor, Schwarzenegger, Austria	46,100	
	Schwarzenegger, Terminator, Austria	16,000	
	Governor, Terminator, Austria	11,500	
Four-way	Governor, Schwarzenegger, Terminator, Austria	6,930	

Classical contingency table estimation problem

Let P denote “postings” (or inverted index) for the word, i.e., P contains a list of document IDs which contain that particular word. The size of the postings for single words can be assumed to be known (and stored). The goal is to compute (or estimate) the intersections among two postings.

$$a = |P_1 \cap P_2|, \ b = |P_1 \cap \neg P_2|, \ c = |\neg P_1 \cap P_2|, \ d = |\neg P_1 \cap \neg P_2|$$

where $\neg P_1$ is short-hand for $\Omega - P_1$, and $\Omega = \{1, 2, 3, \dots, D\}$ is the set of all document

	W ₂	$\neg W_2$
W ₁	$a (x_1)$	$b (x_2)$
$\neg W_1$	$c (x_3)$	$d (x_4)$
$f_1 = a + b$	$f_2 = a + c$	
$D = a + b + c + d$		

(a) Contingency table

	W ₂	$\neg W_2$
W ₁	$a_s (s_1)$	$b_s (s_2)$
$\neg W_1$	$c_s (s_3)$	$d_s (s_4)$
$D_s = a_s + b_s + c_s + d_s$		

(b) Sample table

The task is to estimate “a” (and other cells) from the sample table. Obviously, we can estimate “a” by a simple scaling D/D_s , but we hope to make use of the marginal information: $|P_1| = a+b = f_1$, $|P_2| = a+c = f_2$, which are assumed to be known.

MLE (maximum likelihood estimator)

	W ₂	~W ₂
W ₁	a (x ₁)	b (x ₂)
~W ₁	c (x ₃)	d (x ₄)
$f_1 = a + b$	$f_2 = a + c$	
$D = a + b + c + d$		

(a) Contingency table

	W ₂	~W ₂
W ₁	a _s (s ₁)	b _s (s ₂)
~W ₁	c _s (s ₃)	d _s (s ₄)
$D_s = a_s + b_s + c_s + d_s$		

(b) Sample table

$$\frac{\partial \log \Pr(a_s, b_s, c_s, d_s | D_s; a)}{\partial a} = 0$$

Considering the margin constraints, the partial likelihood $\Pr(a_s, b_s, c_s, d_s | D_s; a)$ expressed as a function of a single unknown parameter, a :

$$\begin{aligned}
 \Pr(a_s, b_s, c_s, d_s | D_s; a) &= \frac{\binom{a}{a_s} \binom{b}{b_s} \binom{c}{c_s} \binom{d}{d_s}}{\binom{a+b+c+d}{a_s+b_s+c_s+d_s}} = \frac{\binom{a}{a_s} \binom{f_1-a}{b_s} \binom{f_2-a}{c_s} \binom{D-f_1-f_2+a}{d_s}}{\binom{D}{D_s}} \\
 &\propto \frac{a!}{(a - a_s)!} \times \frac{(f_1 - a)!}{(f_1 - a - b_s)!} \times \frac{(f_2 - a)!}{(f_2 - a - c_s)!} \times \frac{(D - f_1 - f_2 + a)!}{(D - f_1 - f_2 + a - d_s)!} \\
 &= \prod_{i=0}^{a_s-1} (a - i) \times \prod_{i=0}^{b_s-1} (f_1 - a - i) \times \prod_{i=0}^{c_s-1} (f_2 - a - i) \times \prod_{i=0}^{d_s-1} (D - f_1 - f_2 + a - i)
 \end{aligned}$$

We can take log and hope to simplify the expression.

MLE (maximum likelihood estimator)

The MLE solution can be quite complicated:

$$\sum_{i=0}^{a_s-1} \frac{1}{a-i} - \sum_{i=0}^{b_s-1} \frac{1}{f_1-a-i} - \sum_{i=0}^{c_s-1} \frac{1}{f_2-a-i} + \sum_{i=0}^{d_s-1} \frac{1}{D-f_1-f_2+a-i} = 0$$

It can be simplified to be

$$\left(\frac{1}{f_1-a+1} - \frac{1}{f_1-a+1-b_s} \right) + \left(\frac{1}{f_2-a+1} - \frac{1}{f_2-a+1-c_s} \right) \\ + \left(\frac{1}{D-f_1-f_2+a} - \frac{1}{D-f_1-f_2+a-d_s} \right) + \left(\frac{1}{a} - \frac{1}{a-a_s} \right) = 0$$

Still complicated

Simplifying the MLE by sample-with-replacement

With the “sample-with-replacement” assumption, the likelihood function can be simplified

$$\begin{aligned}\Pr(a_s, b_s, c_s, d_s | D_s; a, r) &= \binom{D_s}{a_s, b_s, c_s, d_s} \left(\frac{a}{D}\right)^{a_s} \left(\frac{b}{D}\right)^{b_s} \left(\frac{c}{D}\right)^{c_s} \left(\frac{d}{D}\right)^{d_s} \\ &\propto a^{a_s} (f_1 - a)^{b_s} (f_2 - a)^{c_s} (D - f_1 - f_2 + a)^{d_s}\end{aligned}$$

The MLE solution is then a cubic equation:

$$\frac{a_s}{a} - \frac{b_s}{f_1 - a} - \frac{c_s}{f_2 - a} + \frac{d_s}{D - f_1 - f_2 + a} = 0$$

Still complicated

From cubic to quadratic, a “brave” simplification

We can imagine there are two independent binomial problems for the same “a”.

$$\left[\binom{f_1}{a_s + b_s} \left(\frac{a}{f_1} \right)^{a_s} \left(\frac{f_1 - a}{f_1} \right)^{b_s} \right] \times \left[\binom{f_2}{a_s + c_s} \left(\frac{a}{f_2} \right)^{a_s} \left(\frac{f_2 - a}{f_2} \right)^{c_s} \right]$$
$$\propto a^{2a_s} (f_1 - a)^{b_s} (f_2 - a)^{c_s}$$

The MLE solution is then a quadratic equation:

$$\frac{2a_s}{a} - \frac{b_s}{f_1 - a} - \frac{c_s}{f_2 - a} = 0$$

with a closed-form solution:

$$\hat{a}_{MLE,a} = \frac{f_1 (2a_s + c_s) + f_2 (2a_s + b_s) - \sqrt{(f_1 (2a_s + c_s) - f_2 (2a_s + b_s))^2 + 4f_1 f_2 b_s c_s}}{2 (2a_s + b_s + c_s)}$$

This is a very accurate formula for solving the original contingency table estimation problem.

References to the quadratic formula

- [1] Using Sketches to Estimate Association, EMNLP 2005
- [2] A Sketch Algorithm for Estimating Two-Way and Multi-Way Associations.
Computational Linguistics 2007

How to obtain good samples

Can we simply use a very large sample size (Ds)?

In early 2007, during a visit to AT&T Labs, Divesh asked me this exact question. My answer was that, since the data are sparse and we store only non-zero locations anyway, using a large sample size (Ds) should work reasonably well, but there are better methods.



Andrei Broder

<https://www.linkedin.com/in/andrei-broder-b2159/>



Divesh Srivastava

<https://www.linkedin.com/in/divesh-srivastava-98a5b2/>

We adopted the idea from Andrei and his colleagues on “**minwise hashing**” (actually the method was not “minwise” to start with in 1997). Many prominent researchers including Moses Charikar, Piotr Indyk, Michael Mitzenmacher etc have made contributions to minwise hashing, which has had numerous applications.

Sampling only from non-zero locations

We first apply a random permutation on all postings (inverted indexes) and keep the first a few non-zero locations . Later, we estimate the original similarity (between postings) for each pair of sketches.

$$\frac{|P_1 \cap P_2|}{|P_1 \cup P_2|} = R \quad \text{Resemblance}$$

P ₁	1	4	5	7	11	13	15
P ₂	2	4	7	8	10	11	13
P ₃	1	3	4	6	9	10	14
P ₄	2	5	8				
P ₅	1	2	5	6	8	9	11
	12	13	14	15			

(a) Postings

$$K_1 = \text{MIN}_{k_1}(\pi(P_1)), K_2 = \text{MIN}_{k_2}(\pi(P_2))$$

1 → 11	9 → 5	K ₁	1	6	8	11
2 → 3	10 → 14	K ₂	3	6	10	12
3 → 9	11 → 15	K ₃	2	5	7	9
4 → 13	12 → 4	K ₄	3	6	12	
5 → 1	13 → 6	K ₅	1	2	3	4
6 → 7	14 → 2		5	6	7	8
7 → 12	15 → 8		9	10	11	13
8 → 10			14	15	16	17

(b) Permutation π

(c) Sketches

This step is the same as in the original (bottomm-k version) of minwise hashing by Broder et al.

Our method (published in 2005) differs in the estimation procedure.

Broder's (1997) original estimator

An unbiased estimator based on hypergeometric was proposed in 1997 by throwing away half samples:

$$\hat{R}_B = \frac{|\text{MIN}_k(K_1 \cup K_2) \cap K_1 \cap K_2|}{|\text{MIN}_k(K_1 \cup K_2)|} \quad \frac{|P_1 \cap P_2|}{|P_1 \cup P_2|} = R$$

Note that intersecting by $\text{MIN}_k(K_1 \cup K_2)$ throws out half the samples.

In 2004 (published in EMNLP 2005), we developed “Conditional Random Sampling (CRS)” to avoid throwing away half of the samples. It turns out the CRS generalizes naturally to non-binary data (published in NIPS 2006, 2008).

Conditional Random Sampling (CRS)

Suppose for two postings (inverted indices), we take (after permutation) the first 7 non-zero locations (light gray box) as the sketches.

P ₁	3	4	7	9	10	15	18	19	24	25	28	33
P ₂	2	4	5	8	15	19	21	24	27	28	31	35

We claim that we would obtain exactly the same samples if we simply take the first **18** (=min(18,21)) columns from the equivalent binary (0/1) data matrix. But since we apply the permutation on columns, any columns (including the first 18 columns) would constitute a random sample.

This way, we effectively obtain a random sample for each pair with the sample size determined only at the estimation time. Then, we can apply good estimators from statistical theory. In particular, we still throw away some samples (in this case, “19”, “21”), but not as many as one half.

CRS for Non-Binary Data

A great advantage of CRS is that it is natural applicable to non-binary data. It is also natural to use CRS for three-way and multi-way similarities, because we have a random sample.

	1	2	3	4	5	6	7	8	D
1	■	■	■	■	■	■	■	■	■
2	■	■	■	■	■	■	■	■	■
3	■	■	■	■	■	■	■	■	■
4	■	■	■	■	■	■	■	■	■
5	■	■	■	■	■	■	■	■	■
n	■	■	■	■	■	■	■	■	■

(a) Original

	1	2	3	4	5	6	7	8	D
1	■	■	■	■	■	■	■	■	■
2	■	■	■	■	■	■	■	■	■
3	■	■	■	■	■	■	■	■	■
4	■	■	■	■	■	■	■	■	■
5	■	■	■	■	■	■	■	■	■
n	■	■	■	■	■	■	■	■	■

(b) Permuted

	1	2	3	4	5	6	7	8	D
1	■	■	■	■	■	■	■	■	■
2	■	■	■	■	■	■	■	■	■
3	■	■	■	■	■	■	■	■	■
4	■	■	■	■	■	■	■	■	■
5	■	■	■	■	■	■	■	■	■
n	■	■	■	■	■	■	■	■	■

(c) Postings

	1	2	3	4	5	6	7	8	D
1	■	■	■	■	■	■	■	■	■
2	■	■	■	■	■	■	■	■	■
3	■	■	■	■	■	■	■	■	■
4	■	■	■	■	■	■	■	■	■
5	■	■	■	■	■	■	■	■	■
n	■	■	■	■	■	■	■	■	■

(d) Sketches

Figures from NIPS 2006 paper.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
u ₁	0	1	0	2	0	1	0	0	1	2	1	0	1	0	2
u ₂	1	3	0	0	1	2	0	1	0	0	3	0	0	2	1

(a) Data matrix and random samples

$$\begin{aligned} P_1 : & \quad 2(1) \quad 4(2) \quad 6(1) \quad 9(1) \quad 10(2) \quad 11(1) \quad 13(1) \quad 15(2) \\ P_2 : & \quad 1(1) \quad 2(3) \quad 5(1) \quad 6(2) \quad 8(1) \quad 11(3) \quad 14(2) \quad 15(1) \end{aligned}$$

(b) Postings

$$\begin{aligned} K_1 : & \quad 2(1) \quad 4(2) \quad 6(1) \quad 9(1) \quad 10(2) \\ K_2 : & \quad 1(1) \quad 2(3) \quad 5(1) \quad 6(2) \quad 8(1) \quad 11(3) \end{aligned}$$

(c) Sketches

NIPS 2008 showed it is better to use $10 - 1 = 9$ as the sample size.

Therefore, we have **a one-sketch-for-all** scheme. The drawback of CRS is that the sample size Ds is determined only at the estimation time and Ds is different from different pair (or group) of sketches, meaning that we do not have a metric space and CRS can not be used for certain applications.

References for CRS

[1] Stable random projections and conditional random sampling, two sampling techniques for modern massive datasets, Ping Li's PhD Dissertation, 2007



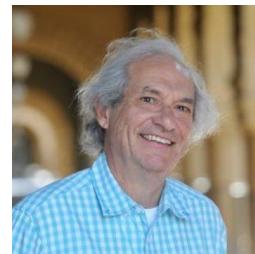
Prof. Trevor Hastie



Prof. Art Owen



Prof. Robert Tibshirani



Prof. Persi Diaconis



Prof. Tim Roughgarden

[2] Using Sketches to Estimate Association, EMNLP 2005

[3] Conditional Random Sampling: A Sketch-based Sampling Technique for Sparse Data. NIPS 2006

[4] A Sketch Algorithm for Estimating Two-Way and Multi-Way Associations. Computational Linguistics 2007

[5] One Sketch For All: Theory and Application of Conditional Random Sampling. NIPS 2008

It appears that CRS (or “bottom-k” sketch) later has become very useful in other areas too.

Another half of the thesis is about “stable random projections”.

Minwise Hashing and Related

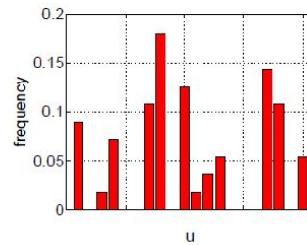
- Originally developed by Andrei Broder and his colleagues in late 1990s, minhash is one of the a few randomized algorithms that is widely used in practice.
- Minhash was originally motivated by the near-duplicate detection task, reported in 1997. It was “bottom-k” and became the “minwse hashing” form since the 1998 STOC paper, by using k permutations.
- b-bit minwise hashing was developed, by storing only the lowest b-bits of each hash value. ([WWW 2010](#))
- A paradigm was developed for using b-bit minwise hashing in large-scale machine learning. ([NIPS 2011](#))
- One permutation hashing (OPH) was developed by breaking data vectors into k bins. ([NIPS 2012](#))
- OPH needs to be densified (filling the empty bins) in order to use OPH for ANN. ([ICML 2014](#), [NIPS 2019](#))
- Circulant-MinHash (C-MinHash) reuses one permutation to replace k permutations. Combined with OPH, C-MinHash only reduces $1/k$ permutation (instead of k or 1 permutations). ([ICML 2022](#) and [arXiv](#) reports).
- Extensions to non-binary data: consistent weighted sampling, extremal processes sampling, etc.

MinHash for high-dimensional 0/1 data

One Major Source of High-Dimensional Data: Histogram

Histogram-based features are very popular in practice, for example, natural language processing (NLP) and computer vision.

It can be viewed as high-dimensional vector: $u_i \geq 0, i = 1, 2, \dots, D$



The size of the space D can often be extremely large. For example, D can be the total number of words, or combinations of words (or characters, or visual words).

In search industry, $D = 2^{64}$ is often used, for convenience.

MinHash for high-dimensional 0/1 data

An Example of Text Data Representation by n -grams

Each document (Web page) can be viewed as a set of n -grams.

For example, after parsing, a sentence “today is a nice day” becomes

- $n = 1$: {“today”, “is”, “a”, “nice”, “day”}
- $n = 2$: {“today is”, “is a”, “a nice”, “nice day”}
- $n = 3$: {“today is a”, “is a nice”, “a nice day”}

It is common to use $n \geq 5$.

Using n -grams generates extremely high dimensional vectors, e.g., $D = (10^5)^n$.

$(10^5)^5 = 10^{25} = 2^{83}$, although in current practice, it seems $D = 2^{64}$ suffices.

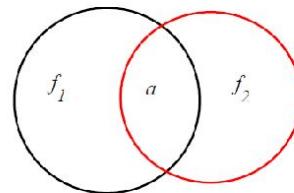
As a highly successful practice, n -gram representations have many variants, e.g., word n -grams, character n -grams, skip n -grams, etc.

MinHash for high-dimensional 0/1 data

Minwise Hashing: Notation

A binary (0/1) vector \iff a set (locations of nonzeros).

Consider two sets $S_1, S_2 \subseteq \Omega = \{0, 1, 2, \dots, D - 1\}$ (e.g., $D = 2^{64}$)



$$f_1 = |S_1|, \quad f_2 = |S_2|, \quad a = |S_1 \cap S_2|.$$

The **resemblance R** is a popular measure of set similarity

$$R = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{a}{f_1 + f_2 - a}. \quad (\text{Is it more rational than } \frac{a}{\sqrt{f_1 f_2}}?)$$

MinHash for high-dimensional 0/1 data

Minwise Hashing: Standard Algorithm in the Context of Search

The standard practice in the search industry:

Suppose a random permutation π is performed on Ω , i.e.,

$$\pi : \Omega \longrightarrow \Omega, \quad \text{where } \Omega = \{0, 1, \dots, D - 1\}.$$

An elementary probability argument shows that

$$\Pr(\min(\pi(S_1)) = \min(\pi(S_2))) = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = R.$$

MinHash for high-dimensional 0/1 data

An Example

$$D = 5, S_1 = \{0, 3, 4\}, S_2 = \{1, 2, 3\}, R = \frac{|S_1 \cap S_2|}{|S_1 \cup S_2|} = \frac{1}{5}.$$

One realization of the permutation π can be

$$0 \implies 3$$

$$1 \implies 2$$

$$2 \implies 0$$

$$3 \implies 4$$

$$4 \implies 1$$

$$\pi(S_1) = \{3, 4, 1\} = \{\textcolor{red}{1}, 3, 4\}, \quad \pi(S_2) = \{2, 0, 4\} = \{\textcolor{red}{0}, 2, 4\}$$

In this example, $\min(\pi(S_1)) \neq \min(\pi(S_2))$.

MinHash for high-dimensional 0/1 data

Minwise Hashing in 0/1 Data Matrix

Original Data Matrix

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

S_1 : 0 1 0 0 1 1 0 0 1 0 0 0 0 0 0 0

S_2 : 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 0

S_3 : 0 0 0 1 0 0 1 1 0 0 0 0 0 0 1 0

Permuted Data Matrix

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

$\pi(S_1)$: 0 0 1 0 1 0 0 1 0 0 0 0 0 1 0 0

$\pi(S_2)$: 1 0 0 1 0 0 1 0 0 0 0 0 0 1 0 0

$\pi(S_3)$: 1 1 0 0 0 0 0 0 0 0 1 0 1 0 0 0

$$\min(\pi(S_1)) = 2, \quad \min(\pi(S_2)) = 0, \quad \min(\pi(S_3)) = 0$$

MinHash for high-dimensional 0/1 data

An Example with $k = 3$ Permutations

Input: sets $S_1, S_2, \dots,$

Hashed values for $S_1 :$ 113 264 1091

Hashed values for $S_2 :$ 2049 103 1091

Hashed values for $S_3 : \dots$

....

MinHash for high-dimensional 0/1 data

Minwise Hashing Estimator

After k permutations, $\pi_1, \pi_2, \dots, \pi_k$, one can estimate R without bias:

$$\hat{R}_M = \frac{1}{k} \sum_{j=1}^k \mathbb{1}\{\min(\pi_j(S_1)) = \min(\pi_j(S_2))\},$$

$$\text{Var}(\hat{R}_M) = \frac{1}{k} R(1 - R).$$

b-bit minwise hashing



Arnd Christian König

Apply a random permutation π on S_1 and S_2 : $\pi : \Omega \longrightarrow \Omega$. Define the minimum values under π to be z_1 and z_2 :

$$z_1 = \min(\pi(S_1)), \quad z_2 = \min(\pi(S_2)).$$

Define $e_{1,i} = i$ th lowest bit of z_1 , and $e_{2,i} = i$ th lowest bit of z_2 .
Theorem 1 derives the main probability formula.

THEOREM 1. Assume D is large.

$$E_b = \Pr \left(\prod_{i=1}^b 1 \{ e_{1,i} = e_{2,i} \} = 1 \right) = C_{1,b} + (1 - C_{2,b}) R$$

Collision probability is still proportional to R , the similarity

where

$$r_1 = \frac{f_1}{D}, \quad r_2 = \frac{f_2}{D},$$

$$C_{1,b} = A_{1,b} \frac{r_2}{r_1 + r_2} + A_{2,b} \frac{r_1}{r_1 + r_2},$$

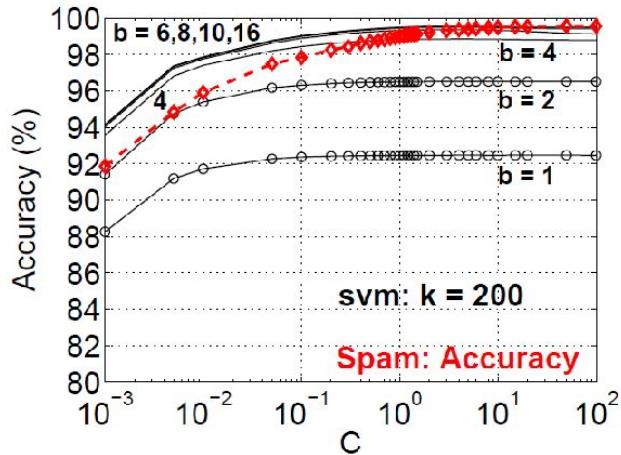
$$C_{2,b} = A_{1,b} \frac{r_1}{r_1 + r_2} + A_{2,b} \frac{r_2}{r_1 + r_2},$$

$$A_{1,b} = \frac{r_1 [1 - r_1]^{2^b - 1}}{1 - [1 - r_1]^{2^b}}, \quad A_{2,b} = \frac{r_2 [1 - r_2]^{2^b - 1}}{1 - [1 - r_2]^{2^b}}.$$

b-bit minwise hashing for learning

Experiments on Webspam (3-gram) Data: Testing Accuracy

Hashing Algorithms for Large-Scale Learning, NIPS 2011



- **Dashed:** using the original data (**24GB** disk space).
- Solid: b -bit hashing. Using $b = 8$ and $k = 200$ achieves about the same test accuracies as using the original data. Space: **70MB** (350000×200)

One Permutation Hashing (OPH)

Intuition: Minwise Hashing Ought to Be Wasteful

Original Data Matrix

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S_1 :	0	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0
S_2 :	0	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0
S_3 :	0	0	0	1	0	0	1	1	0	0	0	0	0	0	1	0

Permuted Data Matrix

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
$\pi(S_1)$:	0	0	1	0	1	0	0	1	0	0	0	0	0	1	0	0
$\pi(S_2)$:	1	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0
$\pi(S_3)$:	1	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0

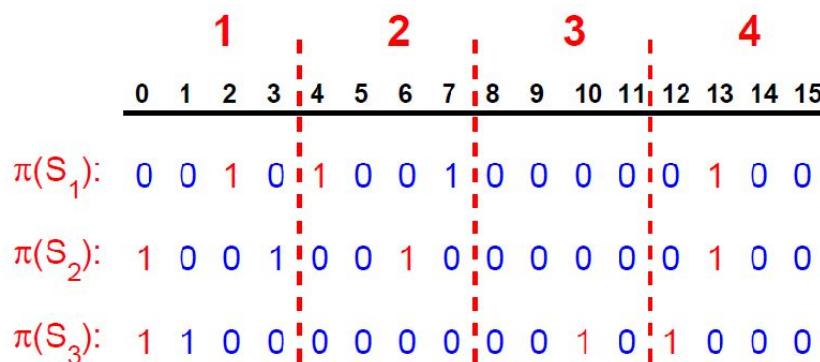
Only store the minimums and repeat the process k (e.g., 500) times.

One Permutation Hashing (OPH)

One Permutation Hashing, NIPS 2012

$S_1, S_2, S_3 \subseteq \Omega = \{0, 1, \dots, 15\}$ (i.e., $D = 16$). The figure presents the permuted sets as three binary (0/1) vectors:

$$\pi(S_1) = \{2, 4, 7, 13\}, \quad \pi(S_2) = \{0, 6, 13\}, \quad \pi(S_3) = \{0, 1, 10, 12\}$$



One permutation hashing: divide the space Ω evenly into $k = 4$ bins and select the smallest nonzero in each bin.

One Permutation Hashing (OPH)

One Permutation Hashing, NIPS 2012

The estimator is simple: count the total number of matches in all bins, count the total number of jointly empty bins. The following ratio estimator is (surprisingly) unbiased:

$$\hat{R}_{mat} = \frac{N_{mat}}{k - N_{emp}}, \quad E(\hat{R}_{mat}) = R$$

Variance is slightly smaller than original minhash

$$Var(\hat{R}_{mat}) = R(1 - R) \left(E\left(\frac{1}{k - N_{emp}}\right) \left(1 + \frac{1}{f - 1}\right) - \frac{1}{f - 1} \right)$$



Prof. Art Owen



Prof. Cun-hui Zhang

Main issue with OPH : empty bins

$$\hat{R}_{mat} = \frac{N_{mat}}{k - N_{emp}}$$

Total number of matches is an inner product.

Total number of jointly empty bins, however, is unknown until estimation time.

The overall estimator cannot be written as an inner product. It does not satisfy the requirement of locality sensitive hashing (**LSH**), unlike the original minhash.

Examples of researchers
who made substantial
contributions to LSH



Prof. Rajeev Motwani



Prof. Piotr Indyk



Prof. Moses Charikar

Minwise hashing for approximate near neighbor search

Specifically, we hash the data points using k random permutations and store each hash value using b bits. For each data point, we concatenate the resultant $B = bk$ bits as a *signature* (e.g., $bk = 16$). This way, we create a table of 2^B buckets and each bucket stores the pointers of the data points whose signatures match the bucket number. In the testing phrase, we apply the same k permutations to a query data point to generate a bk -bit signature and only search data points in the corresponding bucket. Since using only one table will likely miss many true near neighbors, as a remedy, we independently generate L tables. The query result is the union of data points retrieved in L tables.

An example with $L = 2$ tables,
 $b = 2$ bits, and $k = 2$ hashes

Index	Data Points
00 00	6, 110, 143
00 01	3, 38, 217
00 10	(empty)
11 01	5, 14, 206
11 10	31, 74, 153
11 11	21, 142, 329

Index	Data Points
00 00	8, 159, 331
00 01	11, 25, 99
00 10	3, 14, 32, 97
11 01	7, 49, 208
11 10	33, 489
11 11	6, 15, 26, 79

Typically, one might want
to use $b = 2\text{-}4$ bits, $k =$
 $4\text{-}12$, and L as large as
possible

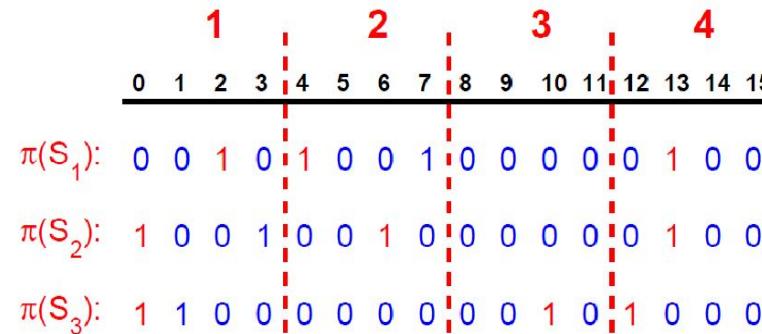
With one permutation hashing (OPH), however, the empty bins would not be able to provide useful information, i.e., we don't know which bin to put the data points into, if part of the hash is empty.

OPH with empty bins for ANN

Densifying One Permutation

Hashing via Rotation for Fast Near

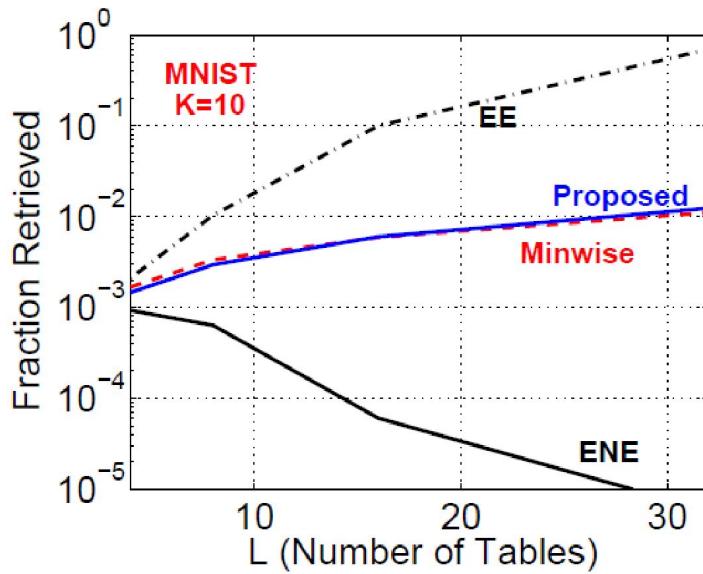
Neighbor Search, ICML 2014



Three strategies to deal with empty bins. First two do not work well.

1. Empty Equal (**EE**): we assign empty bins a fixed symbol. Then any empty will match with any other empty bin. This will create a lot of spurious matches, i.e., will retrieve many unnecessary data points.
2. Empty Not Equal (**ENE**): we assign empty bins random numbers. The chance for two empty bins to match is very small. This strategy will likely retrieve too few data points.
3. Proper **densification**: We can always borrow hash values from the closest (in fixed direction) non-empty bins. There are many variants and “optimal” schemes, after the 2014 ICML paper on densification.

OPH with empty bins for ANN



Densifying One Permutation
Hashing via Rotation for Fast Near
Neighbor Search, ICML 2014

1. We plot the fraction retrieved (compared with the total) versus L , the number of tables. We can see that the proposed densification scheme works well and matches the performance of original minwise hashing.
2. Empty Equal (EE) retrieves too many points and Empty Not Equal (ENE) retrieves too few data points.

The most recent strategy for OPH densification

Re-randomized Densification for One Permutation Hashing and Bin-wise Consistent Weighted Sampling, NeurIPS 2019

In the paper, four different densification schemes are theoretically analyzed and compared.

Rs, RsRe, Den, **DenRe** (Densification with Re-randomization). We will not explain the details here, but nevertheless, we still paste the main variance calculations here to show that they can be analyzed, via non-trivial efforts.

Theorem 1. Let $\tilde{E}_0(m)$ be defined in Lemma 1. Let $f_1 = |S|$, $f_2 = |T|$, $a = |S \cap T|$, $f = |S \cup T| = f_1 + f_2 - a$, and $J = \frac{a}{f}$, $\tilde{J} = \frac{a-1}{f-1}$. N_{emp}^M is the number of empty bins out of first $M \leq K$ bins. If $M > K$, then $N_{emp}^M = N_{emp}^K$. We have

$$\begin{aligned} Var(\hat{J}_{Rs}^M) &= \frac{J}{M} + \frac{M-1}{M}E_1 - J^2, \\ Var(\hat{J}_{RsRe}^M) &= \frac{J}{M} + \frac{M-1}{M}E_2 - J^2. \end{aligned}$$

If $M \leq K$, then

$$Var(\hat{J}_{Den}^M) = \frac{J}{M} + \frac{1}{M^2}\mathbb{E}[(M - N_{emp}^M)(M - N_{emp}^M - 1)J\tilde{J}] + \frac{1}{M^2}\mathbb{E}[N_{emp}^M(2M - N_{emp}^M - 1)]E_1 - J^2,$$

$$Var(\hat{J}_{DenRe}^M) = \frac{J}{M} + \frac{1}{M^2}\mathbb{E}[(M - N_{emp}^M)(M - N_{emp}^M - 1)J\tilde{J}] + \frac{1}{M^2}\mathbb{E}[N_{emp}^M(2M - N_{emp}^M - 1)]E_2 - J^2,$$

If $M > K$, then

$$Var(\hat{J}_{Den}^M) = \frac{1}{M^2}[K^2(Var(\hat{J}_{Den}^K) + J^2) + (M-K)(M+K-1)E_1 + (M-K)J] - J^2,$$

$$Var(\hat{J}_{DenRe}^M) = \frac{1}{M^2}[K^2(Var(\hat{J}_{DenRe}^K) + J^2) + (M-K)(M+K-1)E_2 + (M-K)J] - J^2,$$

where $E_1 = \mathbb{E}[\frac{J}{K-N_{emp}^K} + (1 - \frac{1}{K-N_{emp}^K})J\tilde{J}]$ and $E_2 = \mathbb{E}[\frac{\tilde{E}_0(K-N_{emp}^K)}{K-N_{emp}^K}J + (1 - \frac{\tilde{E}_0(K-N_{emp}^K)}{K-N_{emp}^K})J\tilde{J}]$.

In the paper, four different densification schemes are theoretically analyzed and compared.

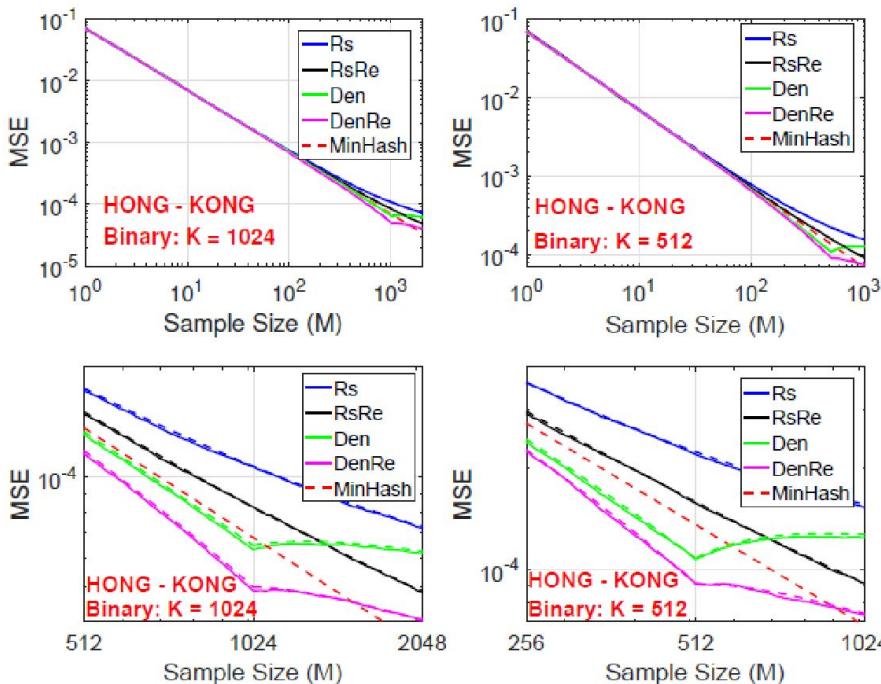
Rs, RsRe, Den, **DenRe** (Densification with Re-randomization). We will not explain the details here, but nevertheless, we still paste the main variance calculations here to show that they can be analyzed, via non-trivial efforts.

Re-randomized Densification for One Permutation Hashing and Bin-wise Consistent Weighted Sampling, NeurIPS 2019

Here we use “J” to stand for resemblance.

The most recent strategy for OPH densification

We estimate resemblance between two word vectors: “Hong” and “Kong”, by generating M samples with K bins and four densification methods to fill empty bins.



Empirical mean square error (MSE) of estimates are compared with the theoretical variances (dashed). They match well.

Take-away message: we will be able to use just one permutation (instead of K permutations), with advanced densification methods to fill the empty bins, to generate M hash values, where M can be even larger than K. The accuracy is in fact better (lower MSE) than original minhash

Circulant MinHash (C-MinHash)

If we don't do densification, we can still just use "one permutation" by circulant hashing trick.

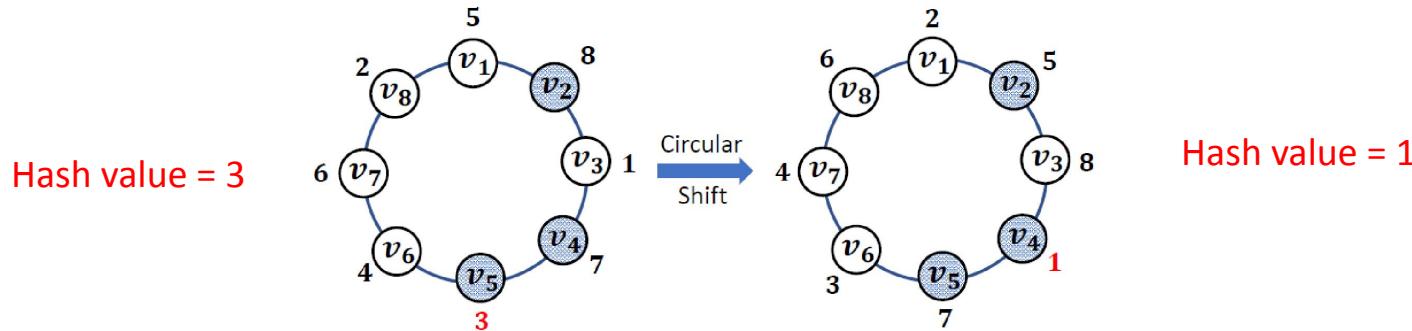
Consider a binary vector with
three non-zero locations at 2, 4, 5

$$v_2 = v_4 = v_5 = 1$$

Suppose we have a permutation. By (circulant) shifting 1, we obtain another permutation.

$$\pi_{\rightarrow k} = \{5, 8, 1, 7, 3, 4, 6, 2\}$$

$$\pi_{\rightarrow k+1} = \{2, 5, 8, 1, 7, 3, 4, 6\}$$



Circulant MinHash (C-MinHash)

Algorithm 1 Minwise-hashing (MinHash)

Input: Binary data vector $v \in \{0, 1\}^D$;
 K independent permutations $\pi_1, \dots, \pi_K: [D] \rightarrow [D]$

Output: K hash values $h_1(v), \dots, h_K(v)$

For $k = 1$ to K

$h_k(v) \leftarrow \min_{i: v_i \neq 0} \pi_k(i)$ Original minhash

End For

Algorithm 3 C-MinHash-(σ, π)

Input: Binary data vector $v \in \{0, 1\}^D$;
 Permutation vectors π and $\sigma: [D] \rightarrow [D]$

Output: Hash values $h_1(v), \dots, h_K(v)$

Initial permutation: $v' = \sigma(v)$

Circulant minhash with an
independent initial permutation

For $k = 1$ to K

 Shift π circulantly rightwards by k units: $\pi_k = \pi_{\rightarrow k}$

$h_k(v) \leftarrow \min_{i: v'_i \neq 0} \pi_{\rightarrow k}(i)$

End For

Algorithm 2 C-MinHash-($0, \pi$)

Input: Binary data vector $v \in \{0, 1\}^D$;
 Permutation vector $\pi: [D] \rightarrow [D]$

Output: Hash values $h_1(v), \dots, h_K(v)$

For $k = 1$ to K

 Shift π circulantly rightwards by k units: $\pi_k = \pi_{\rightarrow k}$

$h_k(v) \leftarrow \min_{i: v_i \neq 0} \pi_{\rightarrow k}(i)$

End For

Circulant minhash
without initial
permutation

Algorithm 4: the same
initial permutation can
be re-used for hashing

C-MinHash-(π, π)

C-MinHash: Improving Minwise Hashing with
Circulant Permutation, ICML 2022

C-MinHash is provably more accurate than MinHash

Theorem 3.1. Let a, f be defined as in (6). When $0 < a < f \leq D$ ($J \notin \{0, 1\}$), we have

$$\text{Var}[\hat{J}_{\sigma, \pi}] = \frac{J}{K} + \frac{(K-1)\tilde{\mathcal{E}}}{K} - J^2, \quad (9)$$

where $l = \max(0, D - 2f + a)$, and

$$\begin{aligned} \tilde{\mathcal{E}} = & \sum_{\Xi} \left(\frac{l_0}{f + g_0 + g_1} + \frac{a(g_0 + l_2)}{(f + g_0 + g_1)f} \right) \left(\sum_{s=l}^{D-f-1} \frac{\binom{D-f}{s}}{\binom{D-a-1}{D-f-1}} \cdot \right. \\ & \left. \frac{\binom{f-a-1}{D-f-s-1} \binom{s}{n_1} \binom{D-f-s}{n_2} \binom{D-f-s}{n_3} \binom{f-a-(D-f-s)}{n_4} \binom{a-1}{a-l_1-l_2}}{\binom{D-1}{a}} \right). \end{aligned} \quad (10)$$

The feasible set $\Xi = \{l_0, l_2, g_0, g_1\}$ satisfies the intrinsic constraints (7), and

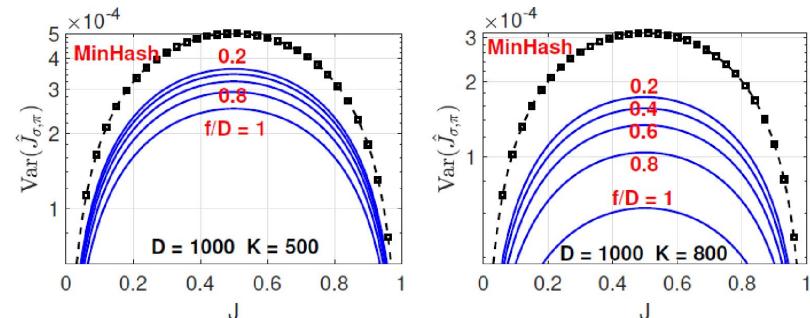
$$\begin{aligned} n_1 &= g_0 - (D - f - s - g_1), & n_2 &= D - f - s - g_1, \\ n_3 &= l_2 - g_0 + (D - f - s - g_1), \\ n_4 &= l_1 - (D - f - s - g_1). \end{aligned}$$

When $a = 0$ or $f = a$ ($J = 0$ or 1), $\text{Var}[\hat{J}_{\sigma, \pi}] = 0$.

Theorem 3.4 (Uniform Superiority). For any two binary vectors $v, w \in \{0, 1\}^D$ with $J \neq 0$ or 1 , it holds that $\text{Var}[\hat{J}_{\sigma, \pi}(v, w)] < \text{Var}[\hat{J}_{MH}(v, w)]$.

"J" stands for resemblance

C-MinHash: Improving Minwise Hashing with Circulant Permutation, ICML 2022



An example to illustrate that the theoretical variance of C-MinHash is smaller, compared with MinHash.

Integration of C-MinHash with OPH

C-OPH: Improving the Accuracy of One Permutation Hashing (OPH) with Circulant Permutations, arXiv 2021

Original MinHash : K permutations

OPH : 1 permutation, densification needed

C-MinHash: 1 permutation (2 permutations merely for theoretical analysis)

C-MinHash + OPH: $1/K$ permutation

Note that these are exact permutations instead of approximations (such as universal hashing). In prior practice, users cannot store K permutations hence resort to approximate permutations. Now, it is expected that users can store 1 permutation, or at least $1/K$ permutation. **Therefore, it is probably the time to move the practice from approximate to exact permutations.**

Pb-Hash: Partitioned b-Bit Hash

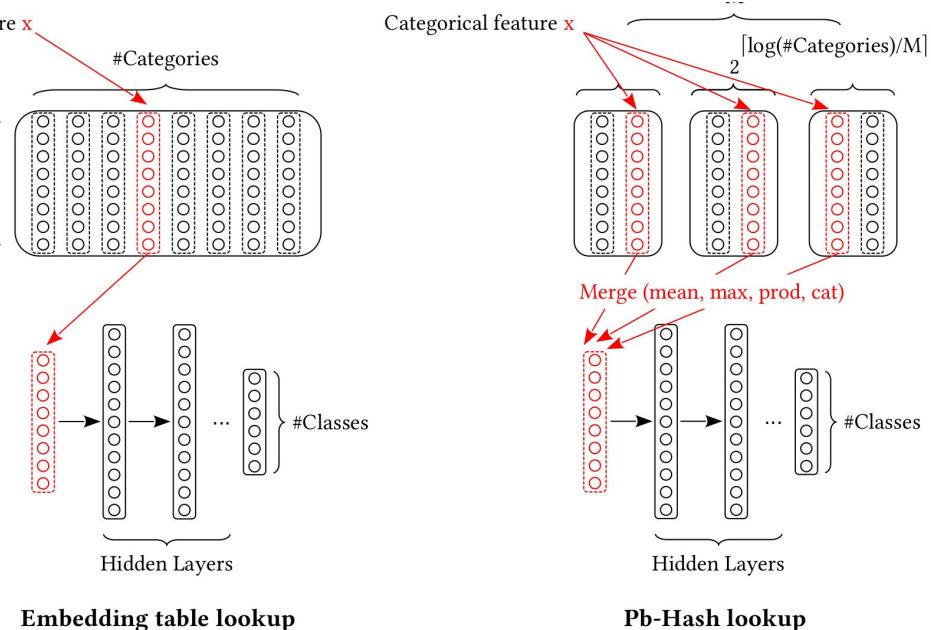
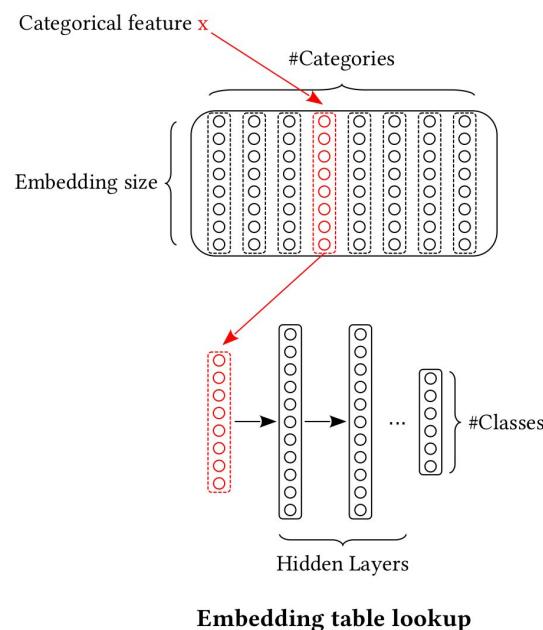
<https://arxiv.org/pdf/2306.15944.pdf>

Pb-Hash = break the bits of a hash code into m chunks and use them as separate hashes.

Interestingly, Pb-hash can be applied to dealing with massive ID features, not just from hash.

In this motivating example, massive ID features correspond to big models in search and ads. They typically require a huge embedding layer. If we break the ID feature into $m = 3$ chunks, the model size and embedding table can be drastically reduced.

QR-hash ($m=2$) is a special instance for this particular case.



Pb-Hash: Partitioned b-Bit Hash

<https://arxiv.org/pdf/2306.15944.pdf>

- Many hash algorithms (e.g., minwise hashing) produce **k hashes of B bits**, where B might be originally very large such as 64, or 32. They correspond to ID features in 2^B dimensions.
- The idea of b-bit hashing is to use the lowest b bits out of the B bits. This significantly reduces the storage and the dimensionality, from 2^B to 2^b . The loss of accuracy can typically be compensated by somewhat increasing k, the number of hashes.
- The idea of Pb-hash is to divide the B bits into m chunks with $b \times m = B$, and treat each chunk as a separate hash. This way, we can re-use the hashes in a more effective way.
- However, re-using the same original hash will hurt the performance due to correlation (because they are not independent hashes). This effect can be fairly accurately calculated, which provides guidance to the design such as the choice of m.

Benefits of Pb-Hash

<https://arxiv.org/pdf/2306.15944.pdf>

- Generating hashes can be expensive for industrial-scale systems especially for many user-facing applications. Thus, engineers may hope to make use of each hash as much as possible, instead of generating more hashes (i.e., by increasing the k).
- To protect user privacy, the hashes might be artificially “polluted” and the differential privacy (DP) budget is proportional to k. See [arXiv 2023](#).
- After hashing, the original data are not necessarily stored and hence it might not be even possible to generate more hashes.
- One special scenario is that we can also apply Pb-Hash to the original categorical (ID) features, not just limited to hashed data. This is also the motivation for QR-Hash

The Basic Assumption of Pb-Hash

<https://arxiv.org/pdf/2306.15944.pdf>

Basic Assumption: Apply the hash function h to two data vectors u and v to obtain $h(u)$ and $h(v)$, respectively, where $h(.) \in [0, 1, 2, \dots, 2^B - 1]$. The collision probability is $\Pr(h(u) = h(v)) = J$. $h^{(b)}(u)$ and $h^{(b)}(v)$ denote the values by taking b bits of $h(u)$ and $h(v)$, respectively, with

$$P_b = \Pr\left(h^{(b)}(u) = h^{(b)}(v)\right) = c_b + (1 - c_b)J, \quad c_b = \frac{1}{2^b} \quad (2)$$

Theoretical Analysis of Pb-Hash

<https://arxiv.org/pdf/2306.15944.pdf>

Recall the **Basic Assumption**: $P_b = \Pr(h^{(b)}(u) = h^{(b)}(v)) = c_b + (1 - c_b)J$, $c_b = \frac{1}{2^b}$. With Pb-Hash, the basic idea is to break the total B bits into m chunks. Let $\sum_{i=1}^m b_i = B$, and later we can assume $b_1 = b_2 = \dots = b_m$ to simplify the expressions. Then, we have the following expectations:

$$E(\hat{P}_{b_i}) = c_{b_i} + (1 - c_{b_i})J \quad (5)$$

$$E\left(\sum_{i=1}^m \hat{P}_{b_i}\right) = \sum_{i=1}^m c_{b_i} + J \sum_{i=1}^m (1 - c_{b_i}). \quad (6)$$

which allows us to write down an unbiased estimator of J :

$$\hat{J}_m = \frac{\sum_{i=1}^m \hat{P}_{b_i}}{\sum_{i=1}^m (1 - c_{b_i})} - \frac{\sum_{i=1}^m c_{b_i}}{\sum_{i=1}^m (1 - c_{b_i})}. \quad (7)$$

Theorem 2.

$$E(\hat{J}_m) = J, \quad (8)$$

$$\text{Var}(\hat{J}_m) = \frac{\sum_{i=1}^m P_{b_i}(1 - P_{b_i}) + \sum_{i \neq i'} (P_{b_i+b_{i'}} - P_{b_i}P_{b_{i'}})}{(\sum_{i=1}^m (1 - c_{b_i}))^2}. \quad (9)$$

$$\text{where } c_{b_i} = \frac{1}{2^{b_i}}, \quad P_{b_i} = c_{b_i} + (1 - c_{b_i})J, \quad P_{b_i+b_{i'}} = c_{b_i+b_{i'}} + (1 - c_{b_i+b_{i'}})J \quad (10)$$

Theoretical Analysis of Pb-Hash

<https://arxiv.org/pdf/2306.15944.pdf>

Proof of Theorem 2. Firstly, it is easy to show that

$$E(\hat{J}_m) = J, \quad Var(\hat{J}_m) = Var\left(\sum_{i=1}^m \hat{P}_{b_i}\right) / \left(\sum_{i=1}^m (1 - c_{b_i})\right)^2.$$

Then we expand the variance of the sum:

$$\begin{aligned} Var\left(\sum_{i=1}^m \hat{P}_{b_i}\right) &= \sum_{i=1}^m Var(\hat{P}_{b_i}) + \sum_{i \neq i'} Cov(\hat{P}_{b_i}, \hat{P}_{b_{i'}}) \\ &= \sum_{i=1}^m P_{b_i}(1 - P_{b_i}) + \sum_{i \neq i'} (P_{b_i+b_{i'}} - P_{b_i}P_{b_{i'}}). \end{aligned}$$

Here we have used the **Basic Assumption.**

□

The key in the analysis of Pb-Hash is the covariance term $Cov(\hat{P}_{b_i}, \hat{P}_{b_{i'}})$, which in the independence case would be just zero. With Pb-Hash, however, the covariance is always non-negative. This is the reason why the accuracy of using m chunks of b -bits from the same hash value would not be as good as using m independent b -bits (i.e., m independent hashes).

Theoretical Analysis of Pb-Hash

Lemma 3.

<https://arxiv.org/pdf/2306.15944.pdf>

$$P_{b_1+b_2} - P_{b_1} P_{b_2} \geq 0$$

is a concave function in $J \in [0, 1]$. Its maximum is $\frac{1}{4} \left(1 - \frac{1}{2^{b_1}}\right) \left(1 - \frac{1}{2^{b_2}}\right)$, attained at $J = 1/2$.

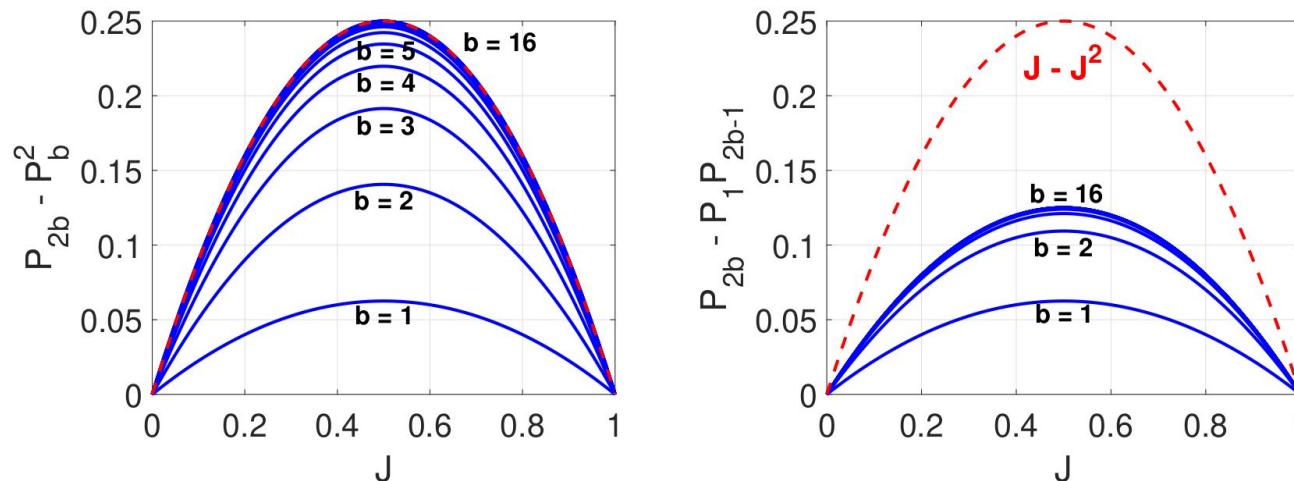


Figure 2: Plots to verify Lemma 3 that $P_{b_1+b_2} - P_{b_1} P_{b_2} \geq 0$. Left panel: $P_{2b} - P_b^2$. Right panel: $P_{2b} - P_1 P_{2b-1}$. It is interesting that in both cases, the maximums are attained at $J = 1/2$.

Theoretical Analysis of Pb-Hash

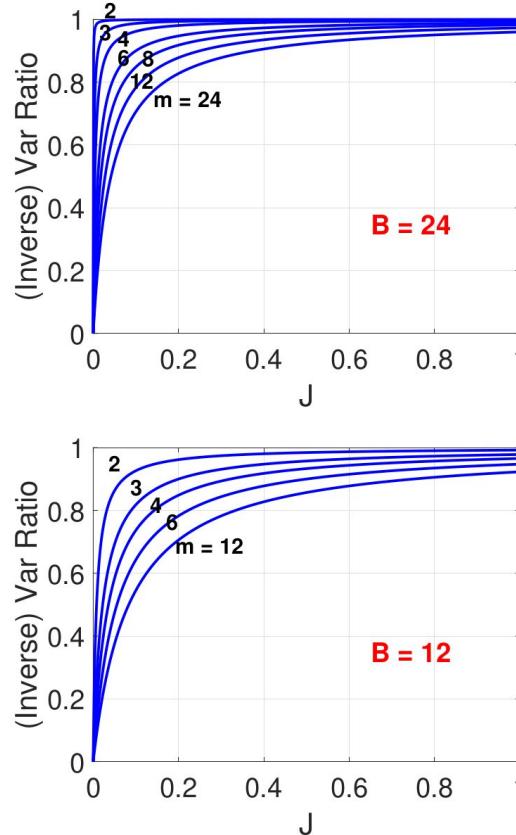
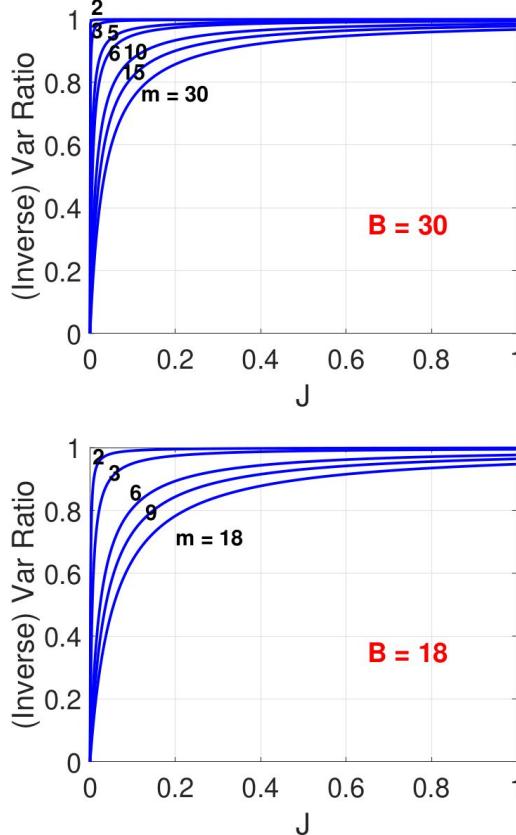
<https://arxiv.org/pdf/2306.15944.pdf>

$$R_{m,b} = \frac{Var(\hat{J}_m)}{J(1-J)} = \frac{P_b(1-P_b) + (m-1)(P_{2b} - P_b^2)}{m(1-c_b)^2 J(1-J)}, \quad m \times b = B. \quad (14)$$

When $R_{m,b}$ is close to 1, it means that Pb-Hash does not lose accuracy as much. Recall that, if we have hashed values for building learning models, the model size is $2^B \times k$, where k is the number of hashes. By Pb-Hash, we can (substantially) reduce the model size to be $m \times 2^b \times k$. In practice, the ID features can have very high cardinality, for example, a million (i.e., $B = 20$) or billion (i.e., $B = 30$). Figure 3 implies that, as long as B is not too small, we do not expect a significant loss of accuracy if $m = 2 \sim 4$.

Theoretical Analysis of Pb-Hash

<https://arxiv.org/pdf/2306.15944.pdf>



$$R_{m,b} = \frac{Var(\hat{J}_m)}{J(1-J)} = \frac{P_b(1-P_b) + (m-1)(P_{2b} - P_b^2)}{m(1-c_b)^2 J(1-J)}$$

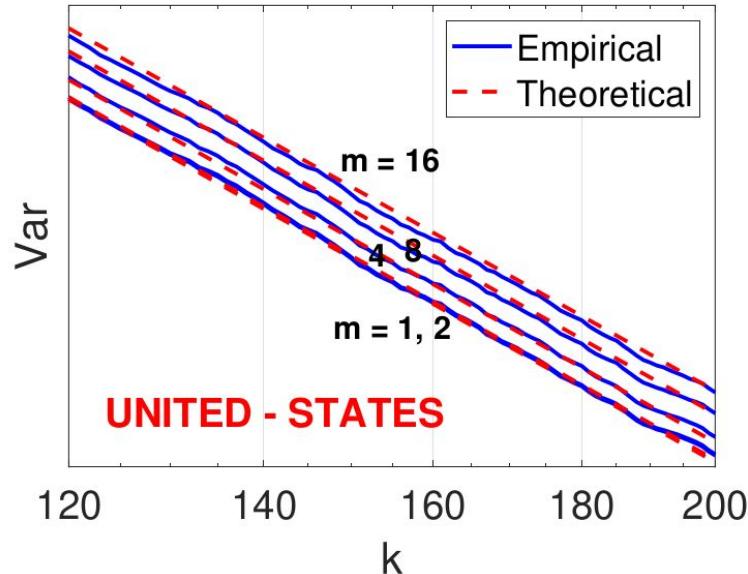
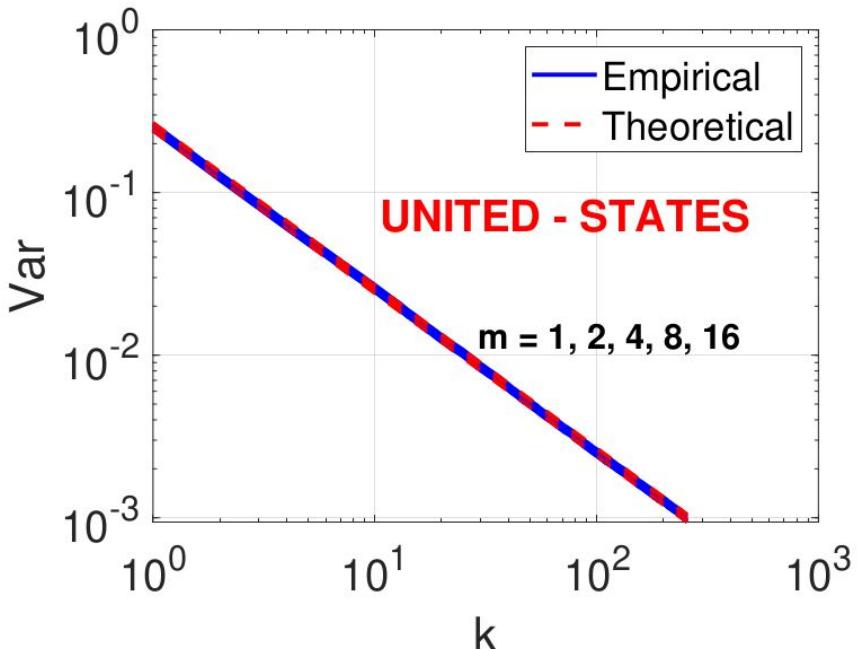
Variance of Pb-hash over variance of original hash.

The ratio ≈ 1 means no loss of accuracy.

Figure 3: Plots for $B \in \{30, 24, 18, 12\}$ to illustrate the variance ratio $R_{m,b}$ in (14).

Experiment of Pb-Hash on Minwise Hashing

<https://arxiv.org/pdf/2306.15944.pdf>



Left panel indicates that even with $m = 16$ (chunks), we do not observe loss of accuracy. Right panel is the zoomed-in view to better tell the differences among due to different m .

Experiment of Pb-Hash on CWS

<https://arxiv.org/pdf/2306.15944.pdf>

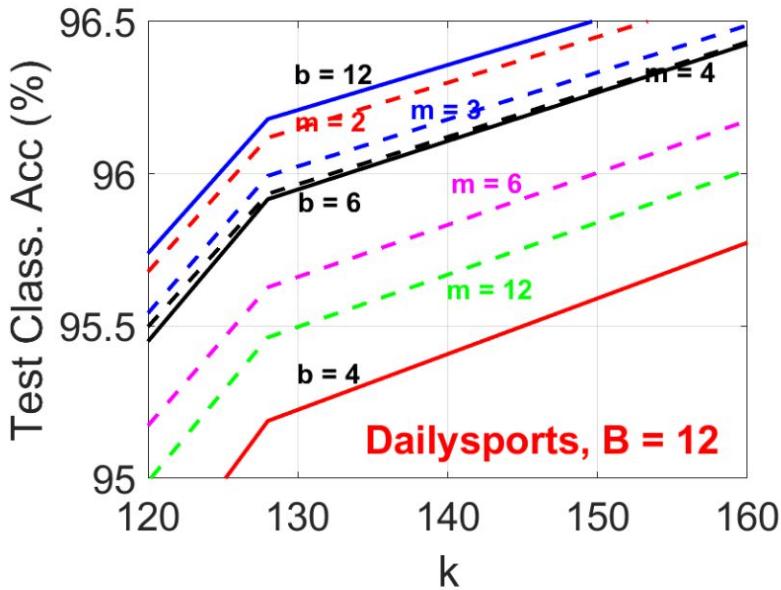
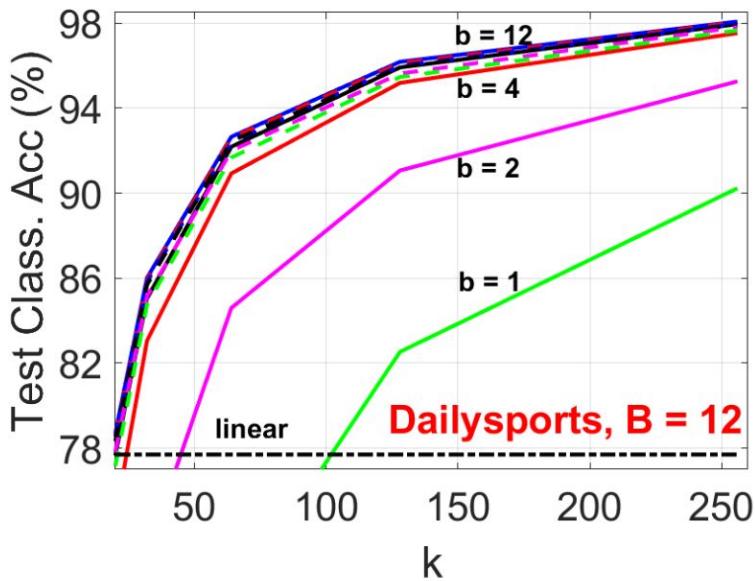


Figure 6: For the “Dailysports” dataset, we apply CWS and keep $B = 12$ bits for each hash value. We choose $b \in \{1, 2, 3, 4, 6, 12\}$ to run the linear SVM classifier. The left panel shows that when $b = 1$ and $b = 2$, we observe a substantial loss of accuracy. In the right panel, we zoom in to show the Pb-Hash results (i.e., dashed curves). We can see with $m = 2 \sim 4$ the loss of accuracy is small.

Experiment on CWS and NN Embeddings

<https://arxiv.org/pdf/2306.15944.pdf>

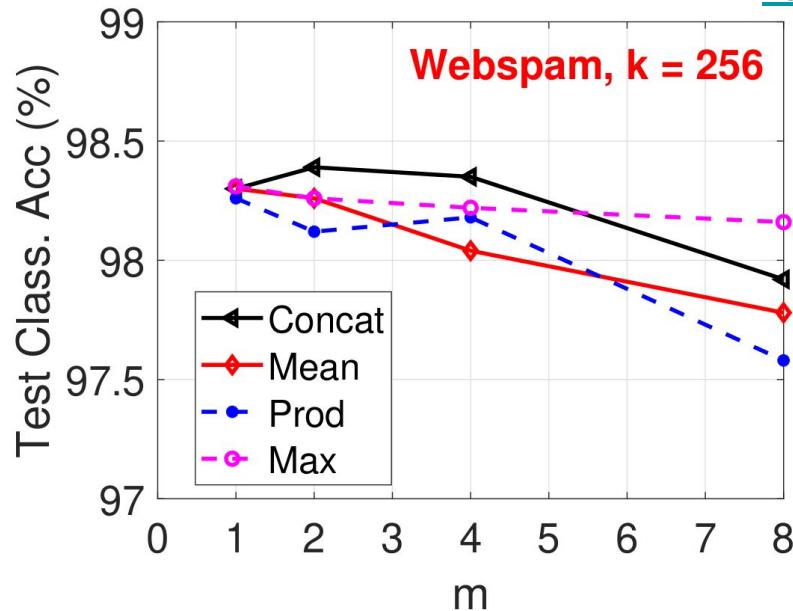


Figure 7: For the “Webspam” (3-gram) dataset, we apply CWS and keep $B = 16$ bits for each hash value. For every hash, we apply Pb-Hash with $m \in \{1, 2, 4, 8\}$. We connect every (sub)-hash to an embedding of size 16. Next we aggregate m embeddings via four different pooling strategies: concatenate, mean, product, and max. Then we connect the pooled embeddings with one hidden layer of size 256.

Consistent Sampling via Extremal Process

WWW 2021

Consistent Sampling Through Extremal Process

Ping Li, Xiaoyun Li, Gennady Samorodnitsky, Weijie Zhao

Baidu Research and Cornell University

10900 NE 8th St. Bellevue, WA 98004, USA

{liping11, v_lxiaoyun02, weijiezhao}@baidu.com, gs18@cornell.edu

- Replace real-value computations with integers. Speed up the computations of CWS.
 - Provide theoretical understanding on the origin and evolution of CWS.
 - Deliver insight for developing more efficient CWS algorithms in the near future.
 - Solve the notorious difficult problem of CWS (i.e., “0-bit CWS”), under the new setting.
- 1.

Consistent Sampling via Extremal Process

[WWW 2021](#)

Algorithm 1: Consistent Weighted Sampling (CWS). The algorithm is shown using a vector $\mathbf{v} \in \mathbb{R}_+^D$ as an example.

- 1 **Input:** Data vector v_i ($i = 1$ to D)
 - 2 **Output:** Consistent uniform sample as a tuple (i^*, t^*)
 - 3 **For** every non-zero v_i
 - 4 $r_i \sim \text{Gamma}(2, 1)$, $c_i \sim \text{Gamma}(2, 1)$, $\beta_i \sim \text{Unif}(0, 1)$
 - 5 $t_i \leftarrow \lfloor \frac{\log v_i}{r_i} + \beta_i \rfloor$, $a_i \leftarrow \log(c_i) - r_i(t_i + 1 - \beta_i)$
 - 6 **End For**
 - 7 $i^* \leftarrow \arg \min_i a_i$, $t^* \leftarrow t_{i^*}$
-

Note that, the line 5 of Algorithm 1,

$$t_i \leftarrow \lfloor \frac{\log v_i}{r_i} + \beta_i \rfloor, \quad a_i \leftarrow \log(c_i) - r_i(t_i + 1 - \beta_i),$$

involves some mathematical operations, e.g., logarithm, floor and multiplication. In real implementation, these operations can be slow.

0-bit CWS (“Relaxed CWS”)

$$P[(i_{\mathbf{v}}^*, t_{\mathbf{v}}^*) = (i_{\mathbf{w}}^*, t_{\mathbf{w}}^*)] \approx P[(i_{\mathbf{v}}^*) = (i_{\mathbf{w}}^*)].$$

is a highly useful empirical observation in [KDD 2015](#). However, no theoretical justification has been conducted.

We solve the analogous problem under the setting of extremal processes.

Consistent Sampling via Extremal Process

[WWW 2021](#)

Operations only involve integers.

Algorithm 3: Extremal sampling (ES) for one hash sample.

```
1 Input: Data vector  $\mathbf{v} \in \mathbb{R}^D$ ; data range  $[m, M]$ 
2 Output: Consistent uniform sample  $(i^*, t^*)$ 
3 For every non-zero  $v_i$ 
4     Generate an extremal process  $X_i(t) = (\mathcal{X}, \mathcal{T})$ ,
5         with  $\mathcal{X} = [x_0, x_1, \dots, x_{k-1}]$ ,  $\mathcal{T} = [t_0, t_1, \dots, t_k]$ ;
6         Find  $j$  such that  $t_j \leq v_i < t_{j+1}$ ,  $j \in [0, k - 1]$ ;
7          $z_i \leftarrow x_j$ ,  $s_i \leftarrow t_j$ 
8 End For
9  $i^* \leftarrow \arg \max_i z_i$ ,  $t^* \leftarrow s_i$ 
```

ES and relaxed ES estimators

$$\hat{J}_{ES}(\mathbf{v}, \mathbf{w}) = \frac{\sum_{j=1}^K \mathbb{1}\{i_j^*(\mathbf{v}) = i_j^*(\mathbf{w}), t_j^*(\mathbf{v}) = t_j^*(\mathbf{w})\}}{K},$$

Algorithm 2: The procedure for simulating extremal process containing $[m, M]$.

```
1 Input: borders  $m, M$ 
2 Output: extremal process  $X(t)$  composed of records set  $\mathcal{X}$ 
    and jump times set  $\mathcal{T}$ 
3 Initialize: generate random number  $r \sim \exp(1)$ , and set
     $x_0 = m/r$ ,  $t_0 = m$ ;  $k = 0$ 
4 While  $t_k < M$ 
5      $k \leftarrow k + 1$ 
6      $r \sim \exp(1)$ ,  $u \sim \text{Unif}(0, 1)$ 
7      $t_k \leftarrow t_{k-1} + x_{k-1}r$ ,  $x_k \leftarrow x_{k-1}/u$ 
8 End While
9  $\mathcal{X} = [x_0, x_1, \dots, x_{k-1}]$ ,  $\mathcal{T} = [t_0, t_1, \dots, t_k]$ 
```

$$\hat{J}_{ES,r}(\mathbf{v}, \mathbf{w}) = \frac{\sum_{j=1}^K \mathbb{1}\{i_j^*(\mathbf{v}) = i_j^*(\mathbf{w})\}}{K}$$

Consistent Sampling via Extremal Process

[WWW 2021](#)

Operations only involve integers.

THEOREM 4.4. (*Relaxed ES estimator*) Suppose $\mathbf{v}, \mathbf{w} \in \mathbb{R}^D$ are non-negative vectors, and $\hat{J}_{ES,r}$ is defined in Eq. (8). We have,

$$\mathbb{E}[\hat{J}_{ES,r}(\mathbf{v}, \mathbf{w})] = P(i^*(\mathbf{v}) = i^*(\mathbf{w})) = J(\mathbf{v}, \mathbf{w}) + R(\mathbf{v}, \mathbf{w}),$$

where the bias term

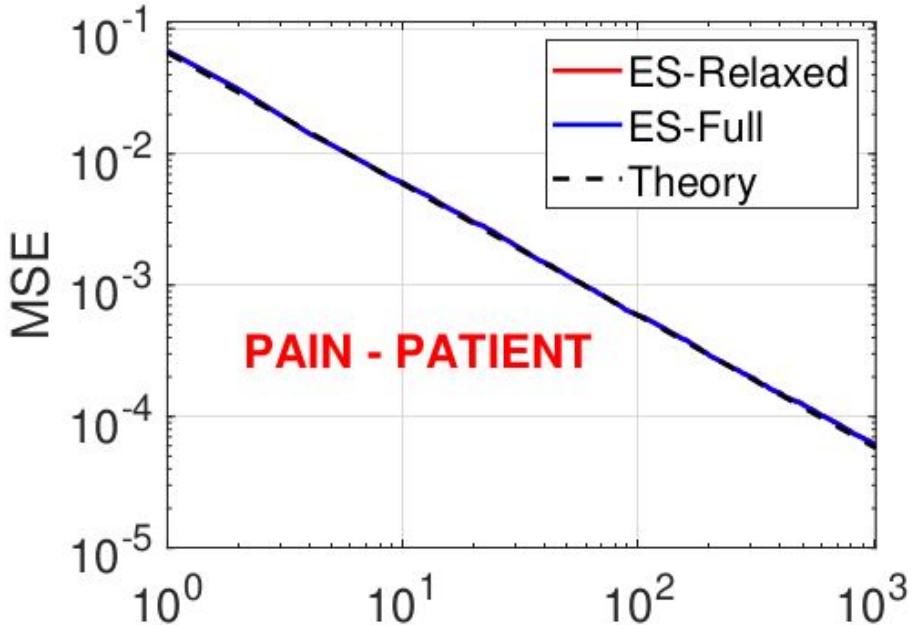
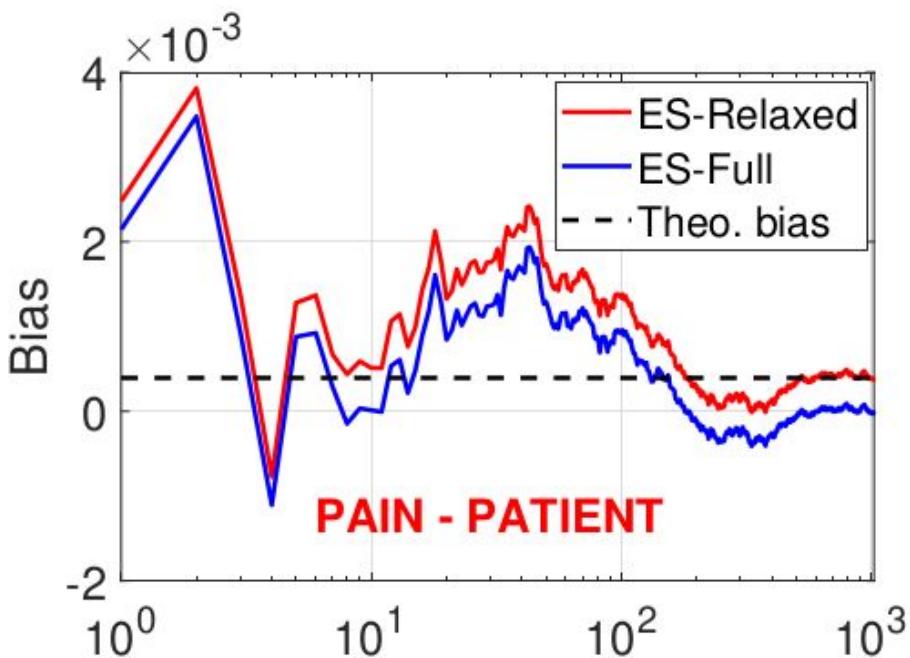
$$R(\mathbf{v}, \mathbf{w}) = \frac{\sum_{G_{\mathbf{v}, \mathbf{w}}} v_i(w_i - v_i)}{\sum_{i=1}^D v_i \sum_{i=1}^D v_i \vee w_i} + \frac{\sum_{i \in G_{\mathbf{v}, \mathbf{w}}^c} w_i(v_i - w_i)}{\sum_{i=1}^D w_i \sum_{i=1}^D v_i \vee w_i}, \quad (9)$$

with $G_{\mathbf{v}, \mathbf{w}} = \{i = 1, \dots, D : 0 \leq v_i \leq w_i\}$, and $G_{\mathbf{v}, \mathbf{w}}^c = \{i = 1, \dots, D : 0 \leq w_i < v_i\}$. In addition, the MSE of $\hat{J}_{ES,r}(\mathbf{v}, \mathbf{w})$ is

$$MSE = \frac{(J(\mathbf{v}, \mathbf{w}) + R(\mathbf{v}, \mathbf{w}))[1 - (J(\mathbf{v}, \mathbf{w}) + R(\mathbf{v}, \mathbf{w}))]}{K} + R^2(\mathbf{v}, \mathbf{w}).$$

Consistent Sampling via Extremal Process

WWW 2021



The bias of relaxed ES is precisely predicted by theory. MSE can tell the difference.

Summary of Extremal Process (ES)

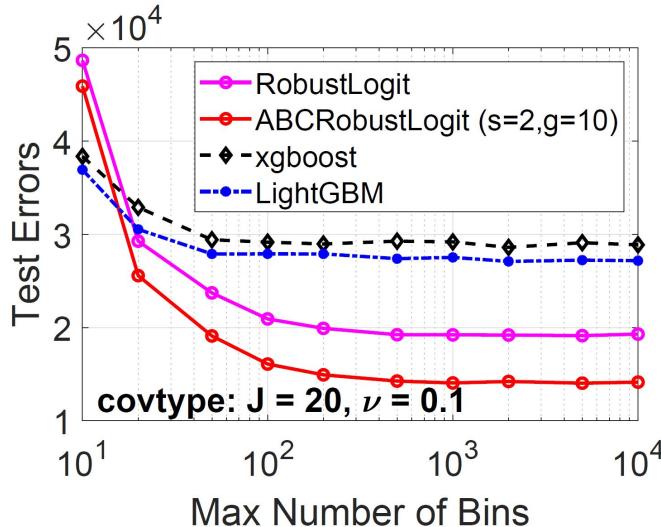
WWW 2021

- ES has a number of advantages over CWS:
 - It avoided real-valued operations and hence can be computationally more efficient.
 - We have carried out the theory for the “relaxed” version of ES.
 - Our study provides insights for further improvement of CWS in the near future.
- ES exhibits one major disadvantage in that it needs to know the range of the data, otherwise additional sampling steps are needed to expand the range during sampling.
- Another disadvantage of ES is that we have observe empirically that the relaxed version of ES is slightly less accurate (i.e., higher bias) than the relaxed version of CWS.

Outline

1. Vector similarity functions
2. Vector compressions
3. Vector similarity search
4. Maximum inner product search (MIPS)
5. Fast neural ranking
6. GPU computing
7. GCWSNet, hashing algorithms
- 8. Boosted trees, ABC-boost**
9. Distributed, adaptive, and federated learning
10. Privacy
11. Security
12. Others: generative AI models, NLP, knowledge graphs, multi-modal, cross-modal, advertising

Accurate and Efficient Boosted Tree Models



ABC-Boost tree models

- The series of works in 2007 - 2010 are behind the success of several popular tree platforms.
- ABC-Boost can be substantially more accurate and more efficient in many datasets.

- <https://hunch.net/?p=1467>, Discussions in 2010 about Ping Li's boosting algorithms
- Ping Li. [ABC-Boost: Adaptive Base Class Boost for Multi-Class Classification](#). ICML 2009.
- Ping Li. [Robust LogitBoost and Adaptive Base Class \(ABC\) LogitBoost](#). UAI 2010.
- Ping Li and Weijie Zhao. [Fast ABC-Boost: A Unified Framework for Selecting the Base Class in Multi-Class Classification](#). arXiv:2205.10927 2022.
- Ping Li and Weijie Zhao. [Package for Fast ABC-Boost](#). arXiv:2207.08770, 2022.
- Ping Li and Weijie Zhao. [pGMM Kernel Regression and Comparisons with Boosted Trees](#). arXiv:2207.08667, 2022.
- Lecture notes on trees & boosting (pages 14-77) <https://statweb.rutgers.edu/pingli/doc/PingLiTutorial.pdf>

Recent developments of trees and boosting

<https://hunch.net/?p=1467>

MACHINE LEARNING (THEORY)

Machine learning and learning theory research

8/23/2010 BY JOHN LANGFORD

Boosted Decision Trees for Deep Learning

About 4 years ago, I speculated that decision trees qualify as a deep learning algorithm because they can make decisions which are substantially nonlinear in the input representation. Ping Li has proved this correct, empirically at UAI by showing that boosted decision trees can beat deep belief networks on versions of Mnist which are artificially hardened so as to make them solvable only by deep learning algorithms.

Recent developments of trees and boosting

<https://hunch.net/?p=1467>

Laurens van der Maaten

8/23/2010 AT 3:05 PM

After seeing Ping's talk about this work in February, I spent some time reproducing his results. Getting the same results as Ping turned out to be fairly easy. It certainly required a lot less tweaking than reproducing the results of DBNs and the like on the same data sets.

This is certainly a huge advantage of trees!

2010 Yahoo! Learning to Rank Grand Challenge

<http://proceedings.mlr.press/v14/chapelle11a/chapelle11a.pdf> (pages 18 – 19)

Ping Li recently proposed Robust LogitBoost ([Li, 2010](#)) to provide a numerically stable implementation of the highly influential LogitBoost algorithm ([Friedman et al., 2000](#)), for classifications. Unlike the widely-used MART algorithm, (robust) LogitBoost use both the first and second-order derivatives of the loss function in the tree-splitting criterion. The five-level ranking problem was viewed as a set of four binary classification problems. The predicted class probabilities were then mapped to a relevance score as in ([Li et al., 2008](#)). For transfer learning, classifiers were learned on each set and a linear combination of the class probabilities from both sets was used.

Recent developments of trees and boosting

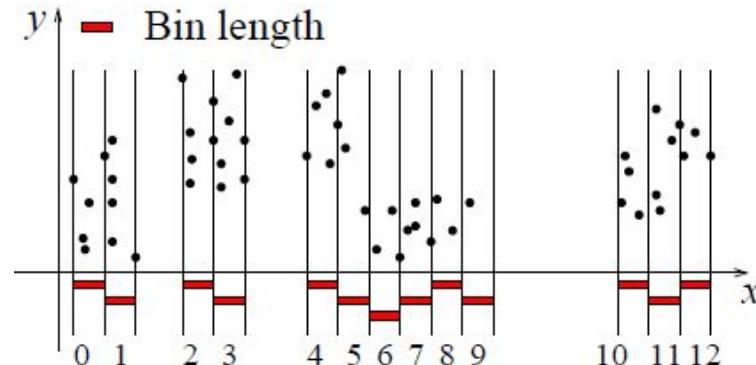
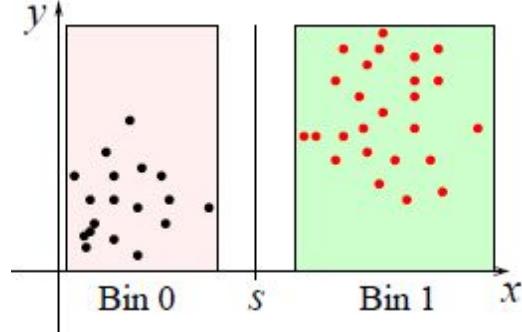
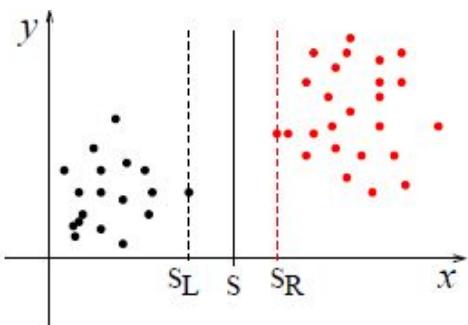
Classical works on boosting and trees before early 2000 were developed by pioneers including Schapire, Freund, Bartlett, Singer, Friedman, Hastie, Tibshirani, etc.

More recently, **three** major developments have made boosted trees more practical and more accurate.

1. The **adaptive binning strategy** for effectively transforming any data types into integers. This makes tree implementations much more convenient and more efficient. [Ping Li et al, **NIPS 2007**]
2. The **gain formula** for tree-split using 2nd-order information. This (in retrospect) simple formula has often made trees substantially more accurate. Using only 1st-order information sometimes did not beat kernel SVMs. This formula is now the standard implementation of popular tree platforms. [Ping Li, **UAI 2010**]
3. The new derivatives (different from textbooks) of logistic regression by assuming a base class and strategies for selecting the base class. These have improved the accuracy of many multi-class classification tasks. [Ping Li, **ICML 2009**]

The adaptive binning algorithm

McRank: Learning to Rank Using Multiple Classification and Gradient Boosting, NIPS 2007



Tree algorithm only splits where there are data and makes arbitrary decision where they are no data

For this split, two bins (0,1) might be sufficient.

Therefore, we can divide the each dimension (feature) into equal-length bins (for simplicity) but we only assign bins where there are data. This strategy is simple and effective.

Why this simple binning method works well?

We have been using this simple (fixed-length) binning method for 15 years. We expect there is ample room to improve the algorithm, but surprisingly we haven't found one that is universally (or largely) better, except that when the maximum number of bins (MaxBin) is too small such as 10 or 20.

1. For discrete features such as {1, 2, 3,...}, this method does not impact the data.
2. The maximum allowed number of bins (i.e., the MaxBin parameter) should not be too small any way for boosted trees. Too much information would be lost if the data are too coarsely quantized. With that many bins (e.g., MaxBin = 1000), it is probably not so easy to improve this fixed-length strategy, as far as the performance of boosting trees is concerned.
3. We should not expect all features would use the same number of bins. Typically, in one dataset, the features can differ a lot. For example, some features might be binary (i.e., even using MaxBin = 1000 would only generate two values), some features may have just 100 distinct values (i.e., using MaxBin = 1000 would still just generate at most 100 values), and some features really need more quantization levels. Therefore, the parameter MaxBin is just a crude guideline. Trying too hard to ``optimize'' the binning procedure according to a given MaxBin is likely counter-productive in real datasets.

Gain formula for tree-split using 2nd-order information

Robust LogitBoost and Adaptive Base Class (ABC) LogitBoost, UAI 2010

This formula is behind the success of popular tree platforms:

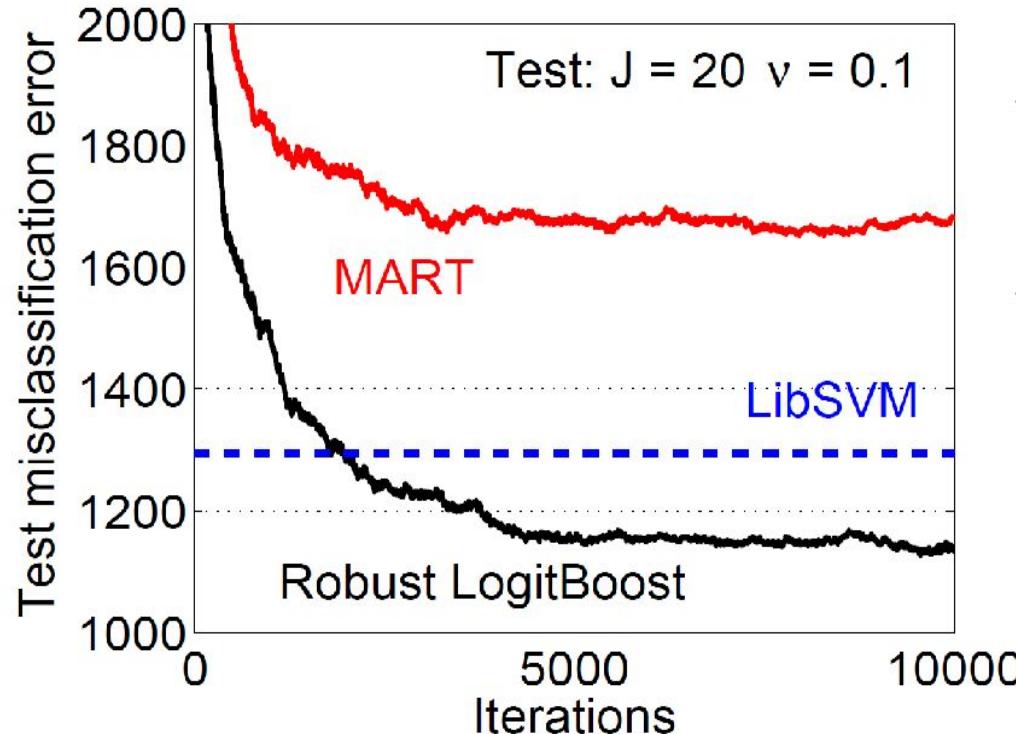
$$Gain(t) = \frac{\left[\sum_{i=1}^t (r_{i,k} - p_{i,k}) \right]^2}{\sum_{i=1}^t p_{i,k}(1 - p_{i,k})} + \frac{\left[\sum_{i=t+1}^N (r_{i,k} - p_{i,k}) \right]^2}{\sum_{i=t+1}^N p_{i,k}(1 - p_{i,k})} - \frac{\left[\sum_{i=1}^N (r_{i,k} - p_{i,k}) \right]^2}{\sum_{i=1}^N p_{i,k}(1 - p_{i,k})}.$$

Boosting for binary classification

IJCNN1 Test Errors

Robust LogitBoost and Adaptive Base Class (ABC) LogitBoost, UAI 2010

This is the result from the 2010 UAI paper (in Appendix), which might be very slightly different from the output of the current package.



This largely explains the success of boosting in practice in past decade:

The second-order tree split formula is the key.

2nd-order tree split formula for general loss functions

$$\begin{aligned} Gain(s) &= SE_{total} - (SE_{left} + SE_{right}) \\ &= \sum_{i=1}^N (z_i - \bar{z})^2 w_i - \left[\sum_{i=1}^s (z_i - \bar{z}_L)^2 w_i + \sum_{i=s+1}^N (z_i - \bar{z}_R)^2 w_i \right], \end{aligned}$$

where $\bar{z} = \frac{\sum_{i=1}^N z_i w_i}{\sum_{i=1}^N w_i}$, $\bar{z}_{left} = \frac{\sum_{i=1}^s z_i w_i}{\sum_{i=1}^s w_i}$, $\bar{z}_{right} = \frac{\sum_{i=s+1}^N z_i w_i}{\sum_{i=s+1}^N w_i}$. With some algebra, we can obtain

$$Gain(s) = \frac{\left[\sum_{i=1}^s z_i w_i \right]^2}{\sum_{i=1}^s w_i} + \frac{\left[\sum_{i=s+1}^N z_i w_i \right]^2}{\sum_{i=s+1}^N w_i} - \frac{\left[\sum_{i=1}^N z_i w_i \right]^2}{\sum_{i=1}^N w_i}$$

Plugging in $z_i = -L'_i/L''_i$, $w_i = L''_i$ yields,

$$Gain(s) = \frac{\left[\sum_{i=1}^s L'_i \right]^2}{\sum_{i=1}^s L''_i} + \frac{\left[\sum_{i=s+1}^N L'_i \right]^2}{\sum_{i=s+1}^N L''_i} - \frac{\left[\sum_{i=1}^N L'_i \right]^2}{\sum_{i=1}^N L''_i}.$$

General formula for any loss function with second derivatives: Lp regression boosting, ranking algorithms, etc.

L_p boosting for regression

$$L_p = \frac{1}{n} \sum_{i=1}^n L_{p,i} = \frac{1}{n} \sum_{i=1}^n |y_i - F_i|^p, \quad \text{where } F_i = F(\mathbf{x}_i).$$

$$\frac{\partial L_{p,i}}{\partial F_i} = -p|y_i - F_i|^{p-1} \text{sign}(y_i - F_i)$$

$$\frac{\partial^2 L_{p,i}}{\partial F_i^2} = p(p-1)|y_i - F_i|^{p-2}. \text{ = 2 if } p = 2 \text{ (the usual L2 boosting)}$$

$$Gain(s) = \frac{\left[\sum_{i=1}^s L'_i \right]^2}{\sum_{i=1}^s L''_i} + \frac{\left[\sum_{i=s+1}^N L'_i \right]^2}{\sum_{i=s+1}^N L''_i} - \frac{\left[\sum_{i=1}^N L'_i \right]^2}{\sum_{i=1}^N L''_i}.$$

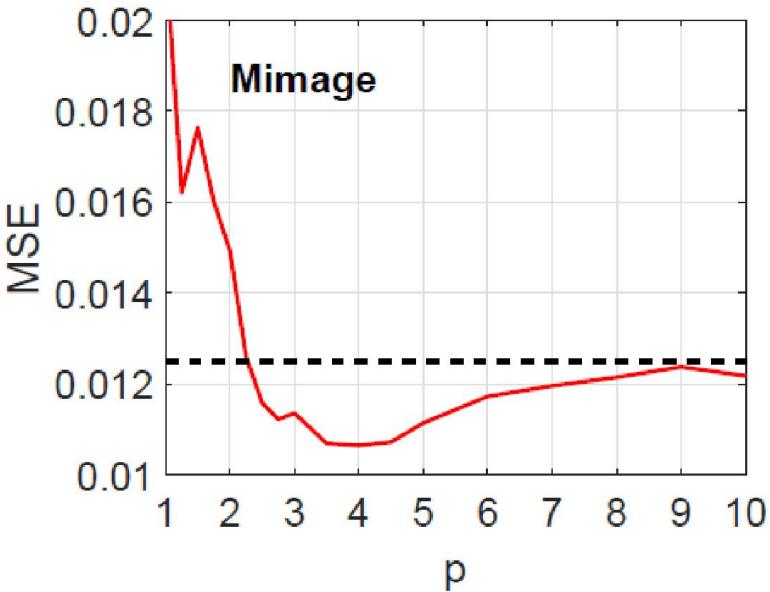
Regression comparison results

Detailed comparisons: LR, RBF, GMM, pGMM, and L2-Boost <https://arxiv.org/pdf/2207.08667.pdf>

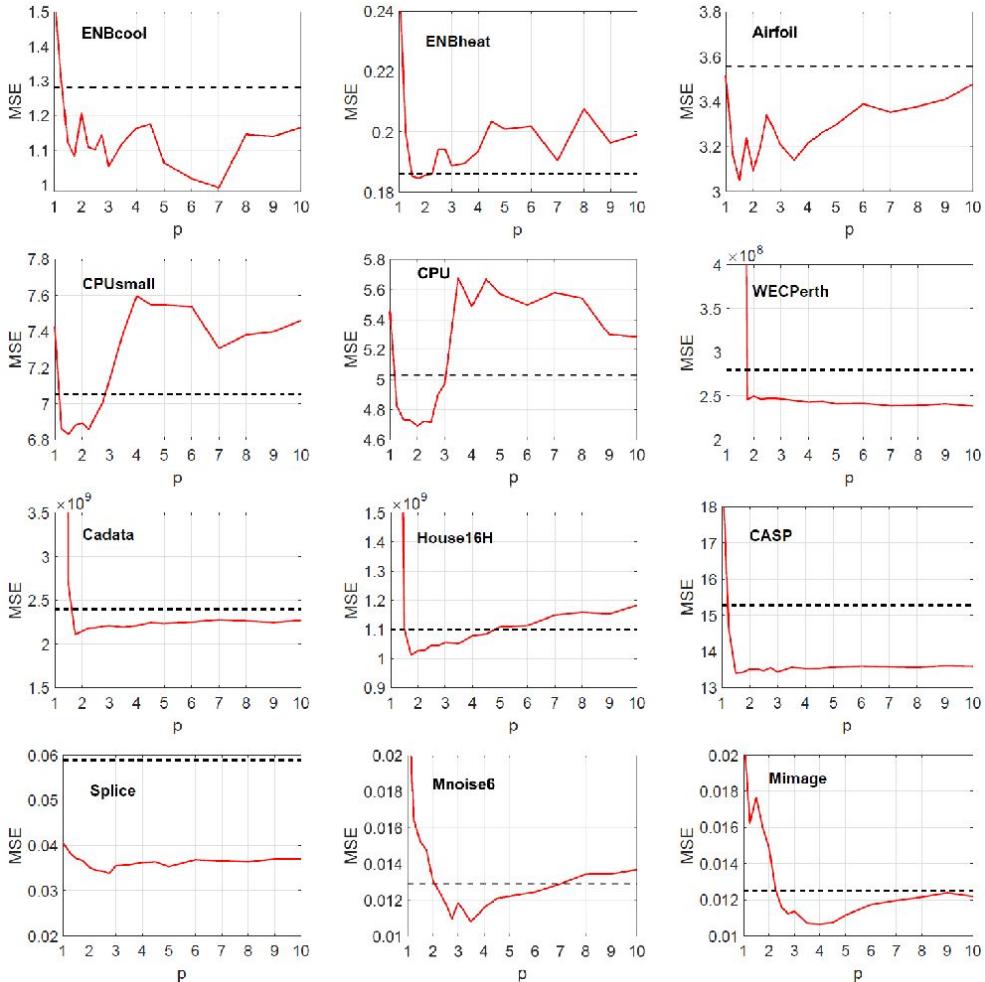
Table 1: Datasets for testing regression algorithms. We report the best test mean square error (MSE) for each method, over the range of regularization coefficients and parameters.

dataset	# train	# test	dim	LR	RBF	GMM	pGMM	L_2 -Boost
ENBcool	384	384	8	10.24	3.20	1.70	1.28	1.21
ENBheat	384	384	8	9.00	0.495	0.191	0.188	0.186
Airfoil	752	751	5	24.26	8.35	7.50	3.56	3.09
CPUsmall	4096	4096	12	102.12	9.05	7.22	7.05	6.89
CPU	4096	4096	21	98.35	6.42	5.17	5.03	4.69
WECPerth	5000	5000	32	1.1×10^9	2.3×10^8	2.4×10^8	2.3×10^8	2.5×10^8
Cadata	10320	10320	8	4.8×10^9	3.8×10^9	2.4×10^9	2.4×10^9	2.1×10^9
House16H	11392	11392	16	2.1×10^9	1.3×10^9	1.2×10^9	1.1×10^9	1.0×10^9
House16L	11392	11392	16	1.9×10^9	1.1×10^9	9.9×10^8	9.4×10^8	8.6×10^8
CASP	22865	22865	9	26.63	23.94	15.30	15.29	13.51
Splice	1000	2175	60	0.1205	0.0967	0.0589	0.0589	0.0352
Mnoise1	2519	454	784	0.0484	0.0344	0.0311	0.0169	0.0145
Mnoise6	2519	454	784	0.0215	0.0165	0.0195	0.0129	0.0131
Mimage	2538	10524	784	0.0540	0.0323	0.0263	0.0125	0.0149

L_p boosting



For some datasets, the difference between L2 boost and L_p boost can be substantial
<https://arxiv.org/pdf/2207.08667.pdf>



ABC-Boost for multi-class classification

ABC-Boost: Adaptive Base Class Boost for Multi-class Classification, ICML 2009

In textbooks, these are the first and second derivatives of logistic regression loss function:

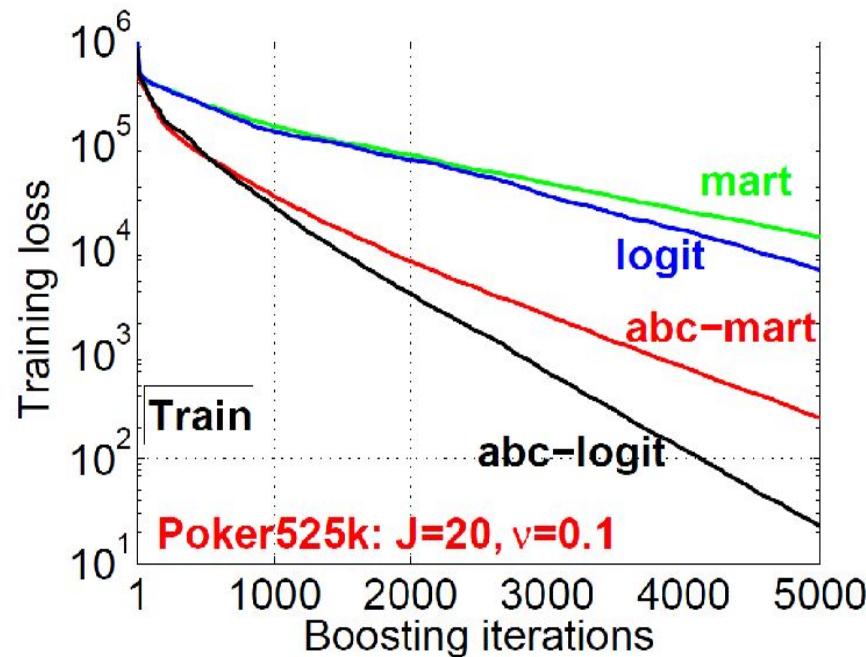
$$\frac{\partial L_i}{\partial F_{i,k}} = -(r_{i,k} - p_{i,k}), \quad \frac{\partial^2 L_i}{\partial F_{i,k}^2} = p_{i,k}(1 - p_{i,k}).$$

The ICML 2009 paper derived a new set of derivatives, by assuming class 0 is “base class”:

$$\begin{aligned}\frac{\partial L_i}{\partial F_{i,k}} &= (r_{i,0} - p_{i,0}) - (r_{i,k} - p_{i,k}), \\ \frac{\partial^2 L_i}{\partial F_{i,k}^2} &= p_{i,0}(1 - p_{i,0}) + p_{i,k}(1 - p_{i,k}) + 2p_{i,0}p_{i,k}.\end{aligned}$$

Training Loss Vs Boosting Iterations

<https://arxiv.org/pdf/1001.1020.pdf> Figure 2



An Empirical Evaluation of Four Algorithms for Multi-Class

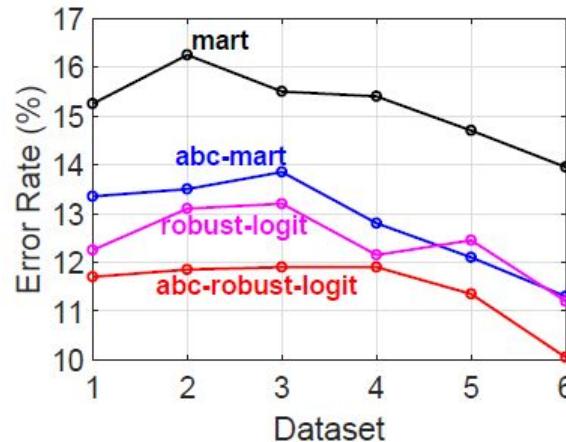
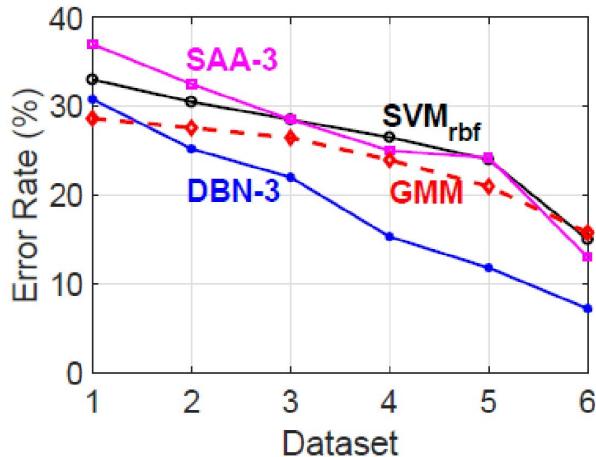
Classification: Mart, ABC-Mart, Robust LogitBoost, and ABC-LogitBoost

arXiv 2010

Comparisons with GMM kernels, trees and deep nets

<https://arxiv.org/pdf/1701.02046.pdf> Six datasets created by CIFAR for deep learning study
<https://arxiv.org/pdf/1805.02830.pdf>

Error rate, lower the better

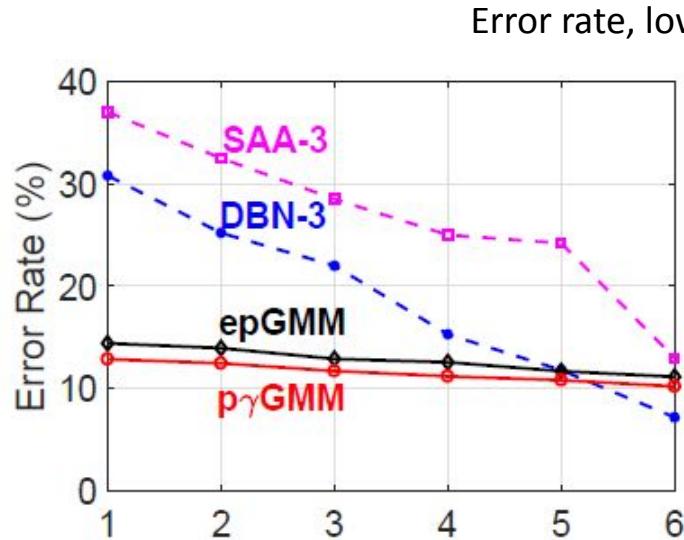


Min-Max (GMM) without tuning
achieved similar accuracy as SVM

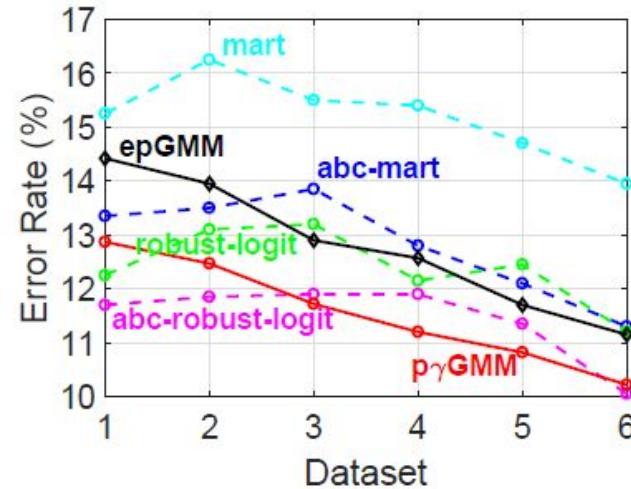
Boosted tree models can do much better

Comparisons with GMM kernels, trees and deep nets

<https://arxiv.org/pdf/1701.02046.pdf> Six additional datasets created by CIFAR for deep learning study
<https://arxiv.org/pdf/1805.02830.pdf>



Tunable Min-Max Kernels do better than deep nets in most cases.



Tunable Min-Max Kernel can be close to boosted tree models

A recent talk at Google: <https://www.linkedin.com/feed/update/urn:li:activity:6965026431579951104/>

- [1] <https://github.com/pltrees/abcboost> , open source package
- [2] <http://statistics.rutgers.edu/home/pingli/STSCI6520/Lecture/ABC-LogitBoost.pdf> , Lecture notes in 2012
- [3] <http://www.stat.rutgers.edu/home/pingli/doc/PingLiTutorial.pdf> , Tutorial edited during 2012-2015

- [4] [McRank: Learning to Rank Using Multiple Classification and Gradient Boosting](#), NIPS 2007
- [5] [Adaptive Base Class Boost for Multi-class Classification](#), arXiv 2008 (worst-class search)
- [6] [ABC-Boost: Adaptive Base Class Boost for Multi-class Classification](#), ICML 2009 (exhaustive search)
- [7] [Robust LogitBoost and Adaptive Base Class \(ABC\) LogitBoost](#), UAI 2010 (second-order tree-split formula)
- [8] [Fast ABC-Boost for Multi-Class Classification](#), arXiv 2010
- [9] [An Empirical Evaluation of Four Algorithms for Multi-Class Classification](#), arXiv 2010
- [10] [pGMM Kernel Regression and Comparisons with Boosted Trees](#), arXiv 2022
- [11] [Classification Acceleration via Merging Decision Trees](#), FODS 2020
- [12] [Fast ABC-Boost: A Unified Framework for Selecting the Base Class in Multi-Class Classification](#), arXiv 2022

Outline

1. Vector similarity functions
2. Vector compressions
3. Vector similarity search
4. Maximum inner product search (MIPS)
5. Fast neural ranking
6. GPU computing
7. GCWSNet, hashing algorithms
8. Boosted trees, ABC-boost
- 9. Privacy**
10. Security
11. Distributed, adaptive, and federated learning
12. Others: generative AI models, NLP, knowledge graphs, multi-modal, cross-modal, advertising

Privacy Has Become Increasingly Critical



NATIONAL STRATEGY TO ADVANCE PRIVACY-PRESERVING DATA SHARING AND ANALYTICS

A Report by the

FAST-TRACK ACTION COMMITTEE ON ADVANCING
PRIVACY-PRESERVING DATA SHARING AND ANALYTICS

NETWORKING AND INFORMATION TECHNOLOGY
RESEARCH AND DEVELOPMENT SUBCOMMITTEE

of the

NATIONAL SCIENCE AND TECHNOLOGY COUNCIL

<https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Strategy-to-Advance-Privacy-Preserving-Data-Sharing-and-Analytics.pdf>

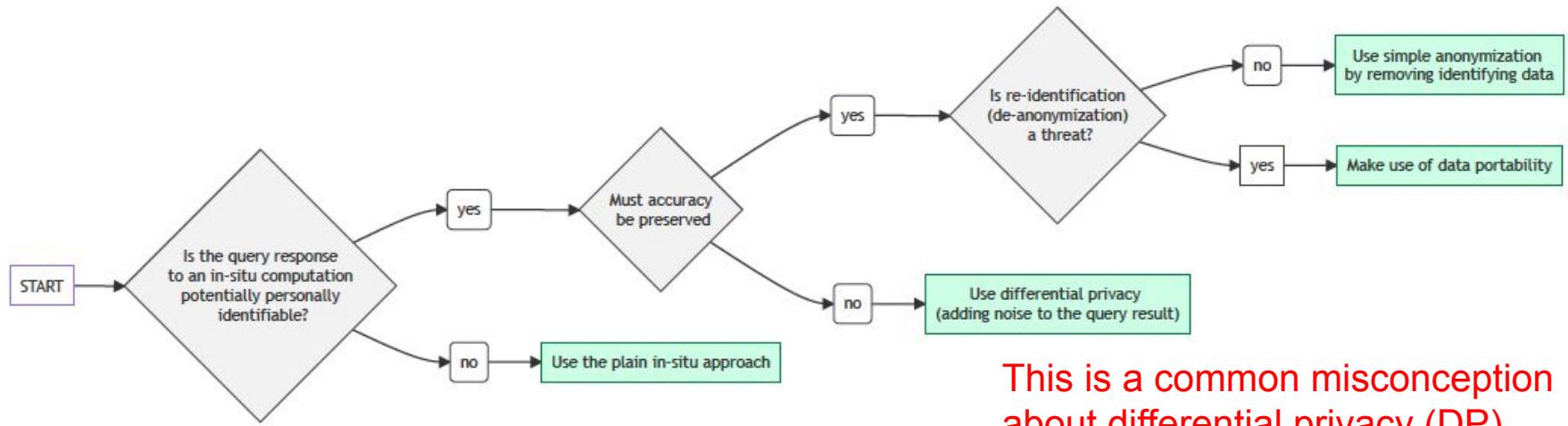
Differential Privacy (DP) and Other Techniques

Table 1. Overview of Key Technical Approaches Essential for PPDSA.

Technique	Description	Value	Limitations
K-anonymity	Transforms a given set of k records in such a way that in the published version, each individual is indistinguishable from the others	Reduces the risk of re-identification	Vulnerable to reidentification attack if additional public information is available
Differential Privacy	Adds noise to the original data in such a way that an adversary cannot tell whether any individual's data was or was not included in the original dataset	Provides formal guarantee of privacy by reducing the likelihood of data reconstruction or linkage attacks	Limited to simpler data types; challenge in managing tradeoff between privacy, accuracy, or utility of data
Synthetic Data	Information that is artificially manufactured as an alternative to real-world data	Preserves the overall properties or characteristics of the original dataset	May still disclose privacy-sensitive information contained in the original dataset; difficult to mirror real-world data

<https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Strategy-to-Advance-Privacy-Preserving-Data-Sharing-and-Analytics.pdf>

Differential Privacy (DP) and Other Techniques



This is a common misconception about differential privacy (DP)

Figure 1. Tentative decision tree.

<https://www.whitehouse.gov/wp-content/uploads/2023/03/National-Strategy-to-Advance-Privacy-Preserving-Data-Sharing-and-Analytics.pdf>

DP and Levels of Protections

<https://arxiv.org/pdf/2303.00654.pdf>

How to DP-fy ML: A Practical Guide to Machine Learning with Differential Privacy

Natalia Ponomareva ^{*1}, Hussein Hazimeh¹, Alex Kurakin², Zheng Xu², Carson Denison³, H. Brendan McMahan³, Sergei Vassilvitskii¹, Steve Chien², and Abhradeep Thakurta²

¹Google Research, NYC

²Google Research, MTV

³Google Research, Seattle

March 3, 2023

- Strong protection: $\epsilon \leq 1$
- Reasonable projection: $\epsilon \leq 10$
- No projection: $\epsilon > 10$

Our Prior Works Related to Privacy

ICLR'23, [Improved Convergence of Differential Private SGD with Gradient Clipping](#)

ArXiv'23, [Differential Privacy with Random Projections and Sign Random Projections](#)

ArXiv'23, [Differentially Private One Permutation Hashing and Bin-wise Consistent Weighted Sampling](#)

ICML'23, [Regression with Label Permutation in Generalized Linear Model](#)

ICML'23, [One-Step Estimator for Permuted Sparse Recovery](#)

KDD'23, [OPORP: One Permutation + One Random Projection](#)

SIGIR'23, [Building K-Anonymous User Cohorts with Consecutive Consistent Weighted Sampling \(CCWS\)](#)

ArXiv'22, [k-Median Clustering via Metric Embedding: Towards Better Initialization with Differential Privacy](#)

NeurIPS'22, [Breaking the Linear Error Barrier in Differentially Private Graph Distance Release](#)

IEEE CNS'22, [NL2GDPR: Automatically Develop GDPR Compliant Android Application from Natural Language](#)

ISIT'22, [Distances Release with Differential Privacy in Tree and Grid Graph](#)

DP Algorithms Based on Random Projections

ArXiv 2023, [Differential Privacy with Random Projections and Sign Random Projections](#)

DP-RP: DP algorithm based on random projections (RP) with many variants

DP-OPORP: A variant of DP-RP Differential Privacy with Random Projections and Sign Random Projections

DP-SignOPORP: DP algorithm based on using signs of OPORP output

iDP-SignRP: another interesting which only needs extremely small ϵ , but it is not strict DP

DP Algorithms Based on Permutations

ArXiv 2023, [Differentially Private One Permutation Hashing and Bin-wise Consistent Weighted Sampling](#)

DP-BCWS: DP algorithm for bin-wise consistent weighted sampling

DP-OPH: DP algorithm for one permutation hashing

DP-MH: DP algorithm for minwise hashing (minhash)

Intuitions behind These New DP Algorithms

Data are often in a vector format, i.e., data vectors in p-dimensions, where p does not have to be small.

The uses of data vectors are often through some "aggregated" form. For example, in similarity search, we often use the "cosine" value (or other similarity measures) between vectors. When training machine learning models, we essentially use their dot products, kernels, or more sophisticated aggregations.

We design randomized algorithms to "aggregate" data, and the resultant values can be further quantized. Interestingly, even after these drastic data transformations, data similarities are still preserved in some form.

The aggregation and quantization operations can significantly facilitate privacy protection.

The randomizations (e.g., hash functions, permutations, random projections) for aggregation are assumed to be known, for achieving stronger privacy protection in the real-world situation.

A series of novel ideas have paved the way for good DP algorithms (good utility with privacy guarantee).

DP-RP Family of Algorithms: the Basic Idea

ArXiv 2023, [Differential Privacy with Random Projections and Sign Random Projections](#)

Instead of using the original data u in p -dimensions, we do rand projections to k dimensions.

$$X = \frac{1}{\sqrt{k}} W^T u, \quad W \in \mathbb{R}^{p \times k}.$$

The distribution of W can be Gaussian, very sparse with various parameters (s), uniform etc. We recommend Rademacher, i.e., very sparse with $s = 1$, or OPORP, the variant of the count-sketch.

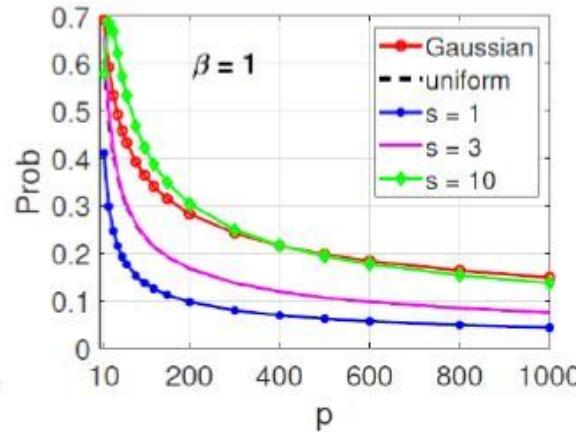
We can use the projected numbers x , to have algorithms called DP-RP (DP-OPORP), but it would be better to use the signs, as the signs will only have a small probability to be changed if the original data u is changed (according to differential privacy (DP) definition). This probability $P+$ can be as small as 0.05.

Sign Flipping Probability of RP

We can use the projected numbers x , to have algorithms called DP-RP (DP-OPORP), but it would be better to use the signs, as the signs will only have a small probability to be changed if the original data u is changed (according to differential privacy (DP) definition). This probability P_+ can be as small as 0.05.

ArXiv 2023, [Differential Privacy with Random Projections and Sign Random Projections](#)

$$P_+ = \Pr \left(\beta \max_{i=1,\dots,p} |w_i| \geq |w^T u| \right)$$



At this point, there are two ways to proceed:

1. Strict DP approach, it works very well, compared to existing DP algorithms.
2. Relaxed DP approach called “individual DP” (iDP). It works remarkably well.

DP-RP and DP-OPORP

ArXiv 2023, [Differential Privacy with Random Projections and Sign Random Projections](#)

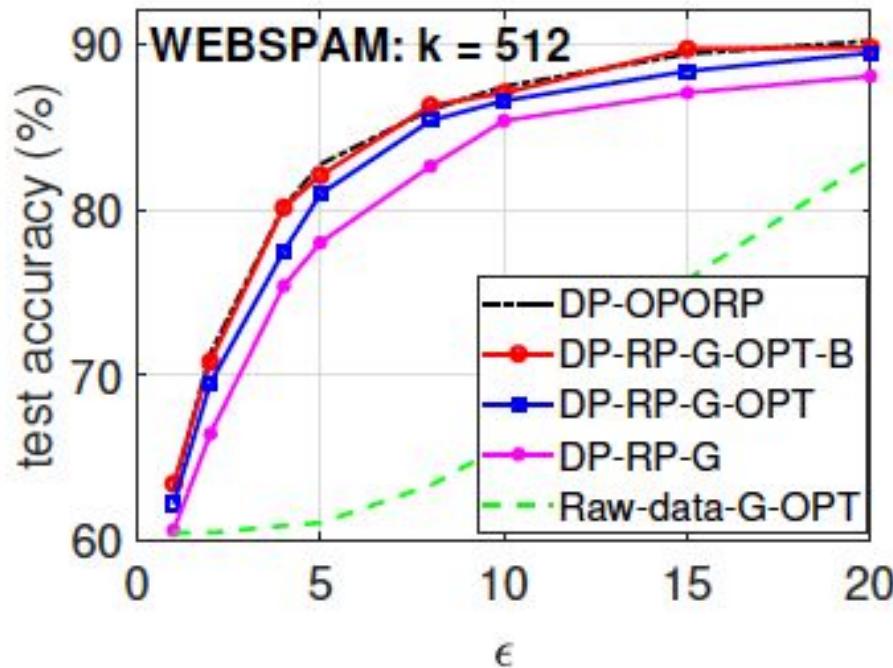
Algorithm 1: DP-RP-G and DP-RP-G-OPT

- 1 **Input:** Data $u \in [-1, 1]^p$, privacy parameters $\epsilon > 0$, $\delta \in (0, 1)$, number of projections k
 - 2 **Output:** (ϵ, δ) -differentially private random projections $\tilde{x} \in \mathbb{R}^k$
 - 3 Apply RP $x = \frac{1}{\sqrt{k}} W^T u$, where $W \in \mathbb{R}^{p \times k}$ has iid entries sampled from $N(0, 1)$
 - 4 Compute the sensitivity Δ_2 by (8)
 - 5 Generate the random noise vector $G \in \mathbb{R}^k$ whose entries are iid samples from $N(0, \sigma^2)$
where σ is obtained by Theorem 3.1 (DP-RP-G) or Theorem 3.5 (DP-RP-G-OPT)
 - 6 Return $\tilde{x} = x + G$
-

DP-RP and DP-OPORP

ArXiv 2023, [Differential Privacy with Random Projections and Sign Random Projections](#)

Compared to the adding DP to the original data (green curve), DP-RP and DP-OPORP considerably improve the accuracy at the same privacy budget (ϵ).



DP-SignOPORP

ArXiv 2023, [Differential Privacy with Random Projections and Sign Random Projections](#)

Algorithm 8: DP-SignOPORP-RR and DP-SignOPORP-RR-smooth

- 1 **Input:** Data $u \in [-1, 1]^p$; $\epsilon > 0$; Number of projections k
- 2 **Output:** Differentially private sign OPORP
- 3 Apply Algorithm 4 with a random Rademacher projection vector to get the OPORP x

DP-SignOPORP-RR:

- 4 Compute $\tilde{s}_j = \begin{cases} sign(x_j), & \text{with prob. } \frac{e^\epsilon}{e^\epsilon + 1} \\ -sign(x_j), & \text{with prob. } \frac{1}{e^\epsilon + 1} \end{cases}$ for $j = 1, \dots, k$

DP-SignOPORP-RR-smooth:

- 5 Compute $L_j = \lceil \frac{|x_j|}{\beta} \rceil$ for $j = 1, \dots, k$

- 6 Compute $\tilde{s}_j = \begin{cases} sign(x_j), & \text{with prob. } \frac{e^{\epsilon'_j}}{e^{\epsilon'_j} + 1} \\ -sign(x_j), & \text{with prob. } \frac{1}{e^{\epsilon'_j} + 1} \end{cases}$ for $j = 1, \dots, k$, with $\epsilon'_j = L_j \epsilon$

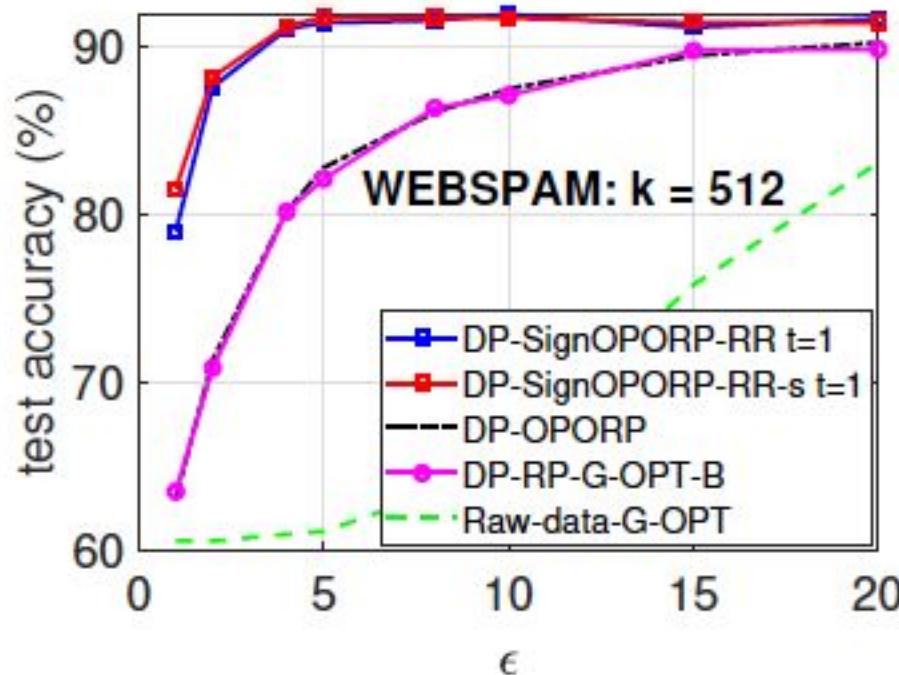
- 7 For $\tilde{s}_j = 0$, assign a random coin in $\{-1, 1\}$

- 8 Return \tilde{s} as the DP-SignOPORP of u
-

DP-SignOPORP

ArXiv 2023, [Differential Privacy with Random Projections and Sign Random Projections](#)

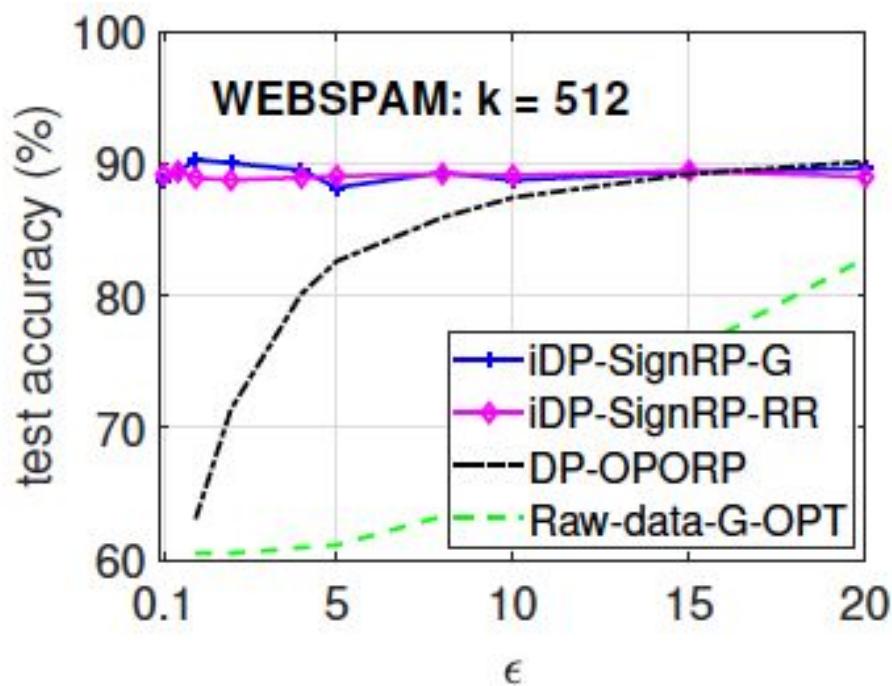
With quantization and using smoothed local sensitivity, DP-SignOPORP further considerably improves the accuracy at the same ϵ .



iDP-SignRP

ArXiv 2023, [Differential Privacy with Random Projections and Sign Random Projections](#)

If applications accept iDP (individual DP), then DP-SignRP can achieve excellent accuracy even at really small ϵ . iDP is a relaxed definition of DP.



DP-BCWS

ArXiv 2023, [Differentially Private One Permutation Hashing and Bin-wise Consistent Weighted Sampling](#)

NeurIPS 2019, [Re-randomized Densification for One Permutation Hashing and Bin-wise Consistent Weighted Sampling](#)

Algorithm 8 Differential Private Bin-wise Consistent Weighted Sampling (DP-BCWS)

Input: Binary vector $\mathbf{u} \in \{0, 1\}^D$; number of hash values K ; number of bits per hash b

Output: DP-BCWS hash values $\tilde{h}_1(\mathbf{u}), \dots, \tilde{h}_K(\mathbf{u})$

```
1: Generate length- $D$  random vectors  $\mathbf{r} \sim \text{Gamma}(2, 1)$ ,  $\mathbf{c} \sim \text{Gamma}(2, 1)$ ,  $\boldsymbol{\beta} \sim \text{Uniform}(0, 1)$ 
2: Let  $d = D/K$ . Use a permutation  $\pi : [D] \mapsto [D]$  with fixed seed to randomly split  $[D]$  into  $K$  equal-size bins  $\mathcal{B}_1, \dots, \mathcal{B}_K$ , with  $\mathcal{B}_k = \{j \in [D] : (k-1)d + 1 \leq \pi(j) \leq kd\}$ 
3: for  $k = 1$  to  $K$  do
4:   if Bin  $\mathcal{B}_k$  is non-empty then
5:      $h_k(\mathbf{u}) \leftarrow \text{CWS}(\mathbf{u}_{\mathcal{B}_k}; \mathbf{r}_{\mathcal{B}_k}, \mathbf{c}_{\mathcal{B}_k}, \boldsymbol{\beta}_{\mathcal{B}_k})$                                  $\triangleright$  Run CWS within each non-empty bin
6:      $h_k(\mathbf{u}) \leftarrow$  last  $b$  bits of  $h_k(\mathbf{u})$ 
7:      $\tilde{h}_k(\mathbf{u}) = \begin{cases} h_k(\mathbf{u}), & \text{with probability } \frac{e^\epsilon}{e^\epsilon + 2^b - 1} \\ i, & \text{with probability } \frac{1}{e^\epsilon + 2^b - 1}, \text{ for } i \in \{0, \dots, 2^b - 1\}, i \neq h_k(\mathbf{u}) \end{cases}$ 
8:   else
9:      $h_k(\mathbf{u}) \leftarrow E$ 
10:     $\tilde{h}_k(\mathbf{u}) = i$  with probability  $\frac{1}{2^b}$ , for  $i = 0, \dots, 2^b - 1$        $\triangleright$  Assign random bits to empty bin
11:   end if
12: end for
```

DP-BCWS

ArXiv 2023, [Differentially Private One Permutation Hashing and Bin-wise Consistent Weighted Sampling](#)

NeurIPS 2019, [Re-randomized Densification for One Permutation Hashing and Bin-wise Consistent Weighted Sampling](#)

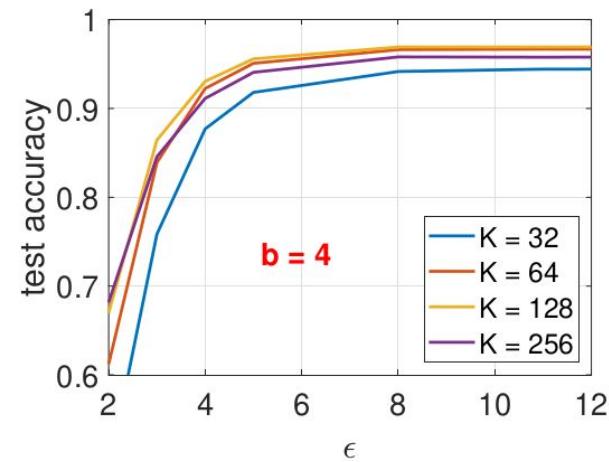
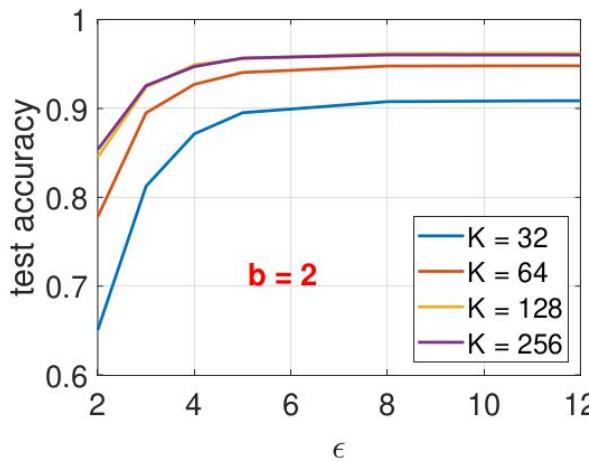
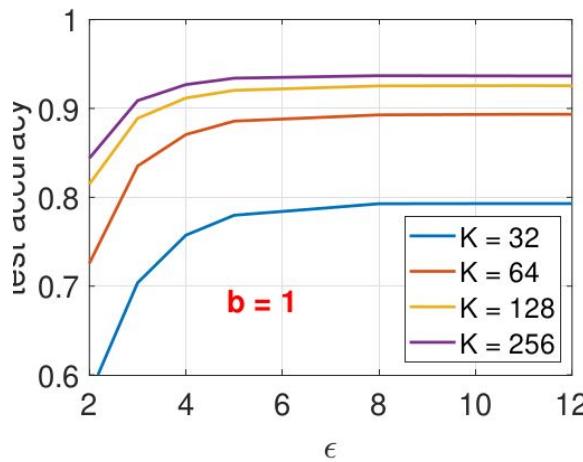


Figure 7: Test classification accuracy of DP-BCWS on MNIST dataset (LeCun, 1998) with 2-hidden layer neural network.

Fines for GDPR Incompliance



Google continues to send data from EU websites to the US - despite two Court of Justice rulings. Austrian Data Protection Authority could fine Google up to €6 billion.

Following the introduction of GDPR in May 2018, initial reports showed that **data breach complaints increased by 160%**. This rate is alarming and indicates just how critical it is to ensure staff receive **comprehensive GDPR training**.

Top 20 GDPR fines so far

1. Amazon Europe - €746m fine (2021)
2. WhatsApp Ireland - €225m fine (2021)
3. Google Inc - €50m fine (2019)
4. H&M - €35.3m fine (2020)
5. TIM - €27.8m fine (2020)
6. British Airways - €22m fine (2020)

GDPR: Fines increased by 40% last year, and they're about to get a lot bigger

Non-compliant businesses, beware: analysts say that regulators are about to get much tougher with GDPR enforcement.

<https://www.skillcast.com/blog/20-biggest-gdpr-fines>

<https://www.zdnet.com/article/gdpr-fines-increased-by-40-last-year-and-theyre-about-to-get-a-lot-bigger/>

<https://noyb.eu/en/austrian-dpa-has-option-fine-google-eu6-billion>

NL2GDPR: Automatically Develop GDPR Compliant Android Application Features from Natural Language

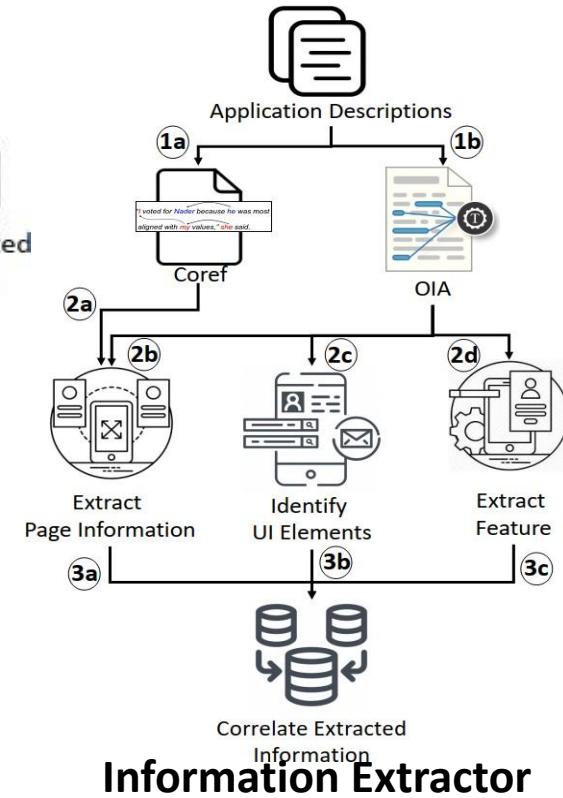
System Architecture



End-to-end Performance

Policy	Accuracy
Retention	89.6%
Consent	89.6%
Privacy Policy	87.5%
Access	89.6%
Deletion	87.5%
Sharing	91.7%
Security	91.7%

CNS 2022



Outline

1. Vector similarity functions
2. Vector compressions
3. Vector similarity search
4. Maximum inner product search (MIPS)
5. Fast neural ranking
6. GPU computing
7. GCWSNet, hashing algorithms
8. Boosted trees, ABC-boost
9. Privacy
- 10. Security**
11. Distributed, adaptive, and federated learning
12. Others: generative AI models, NLP, knowledge graphs, multi-modal, cross-modal, advertising

AI Model Security

In addition to privacy, the AI model security has become increasingly important:

- FACT SHEET: Biden-Harris Administration Secures Voluntary Commitments from Leading Artificial Intelligence Companies to Manage the Risks Posed by AI, [link](#),
- The Blueprint for an AI Bill of Rights, [link](#)

Since 2019, we have worked on watermarking, integrity authentication, and backdoor attacks.

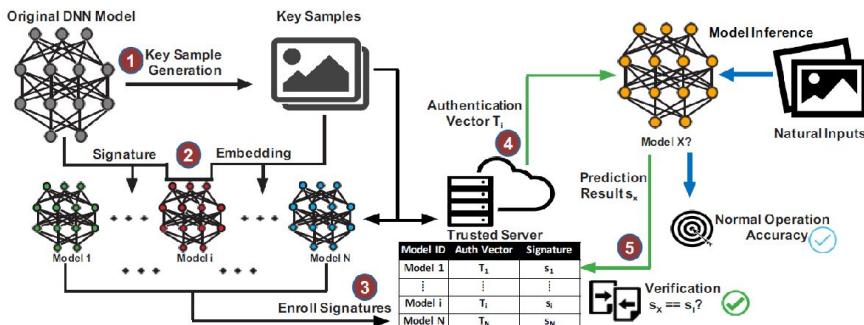


Figure 1: Proposed DeepAuth mainly consists of key sample generation (Step 1) and signature embedding (Step 2). Enrollment (Step 3) and authentication (Steps 4 and 5) are then performed based on the unique signature of each protected model.

AI Model Security

Since 2019, we have worked on watermarking, integrity authentication, and backdoor attacks.

The list of our prior works on AI model security:

AAAI 2023, [Defending Backdoor Attacks on Vision Transformer via Patch Processing](#)

NeurIPS 2022, [Marksman Backdoor: Backdoor Attacks with Arbitrary Target Class](#)

KDD 2022, [Integrity Authentication in Tree Models](#)

ICDE 2022, [Identification for Deep Neural Network: Simply Adjusting Few Weights!](#)

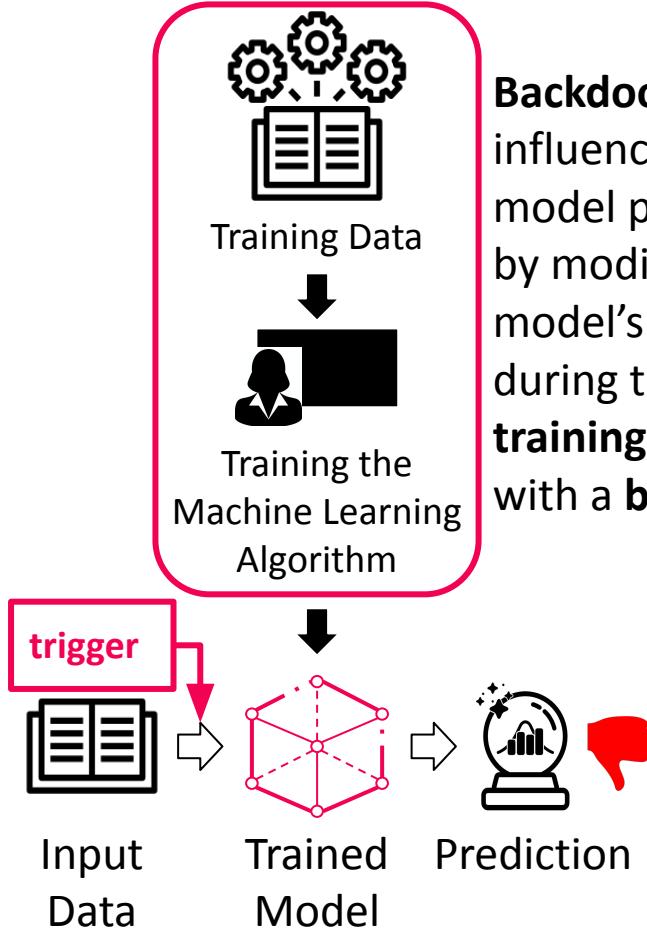
AAAI 2022, [DeepAuth: A DNN Authentication Framework by Model-Unique and Fragile Signature Embedding](#)

NeurIPS 2021, [Backdoor Attack with Imperceptible Input and Latent Modification](#)

ICCV 2021, [LIRA: Learnable, Imperceptible and Robust Backdoor Attacks](#)

ICCV 2021, [Robust Watermarking for Deep Neural Networks via Bi-level Optimization](#)

Backdoor Attacks on Deep Neural Network (DNN)



Backdoor Attack influences the model prediction by modifying the model's behavior during the **training process** with a **backdoor**.



Prediction: **STOP**

Prediction: **GO**

This is a paramount security concern in the model building supply chain, as the increasing complexity of machine learning models has promoted training outsourcing and machine learning as a service (MLaaS).

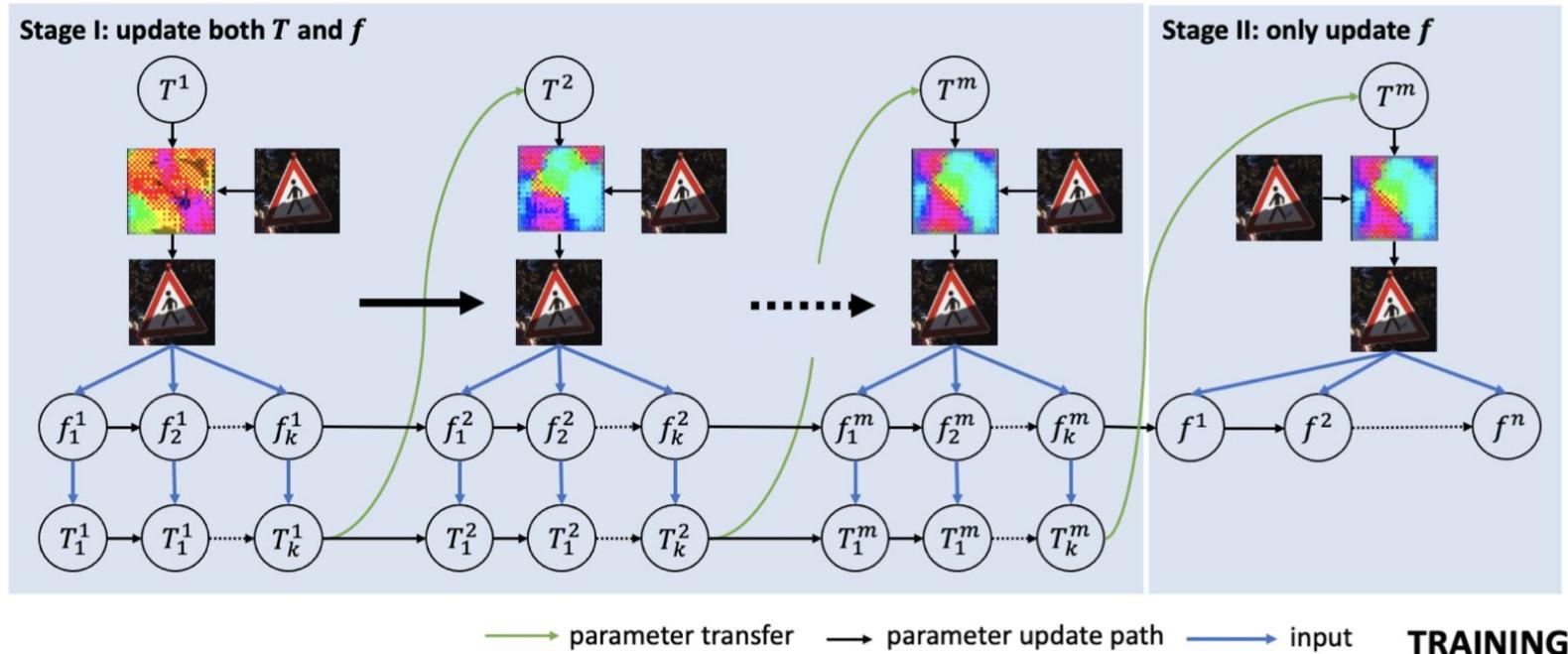
Prior Works: Fixed Trigger/Transformation Function



Limitation: The transformation function is predetermined

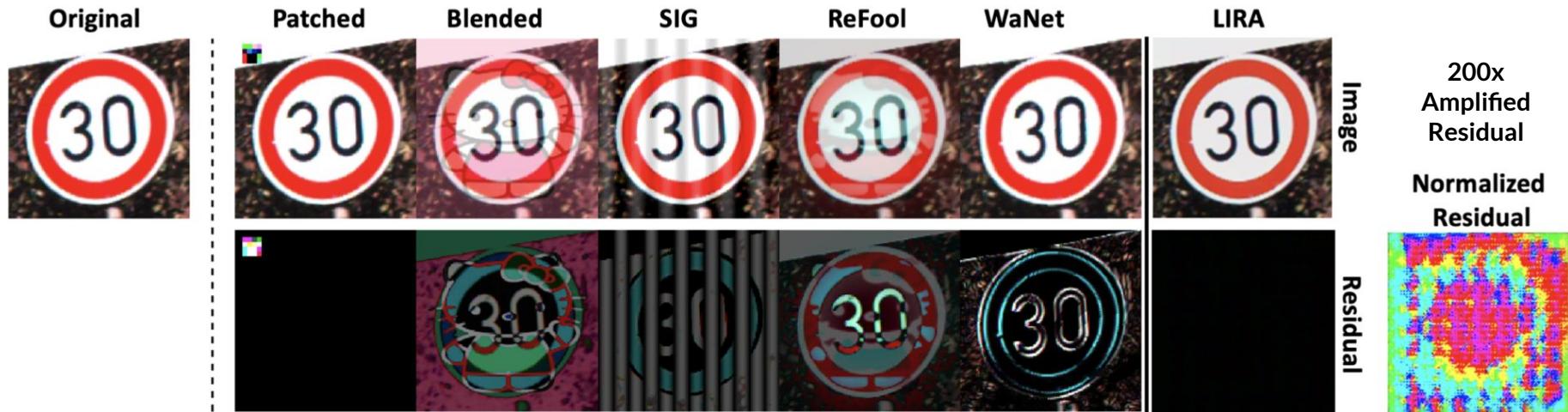
- Limits the attack visual stealthiness
- Results in lower attack success rates

LIRA: Learnable, Imperceptible BackdooR Attack



LIRA's learning process is separated in 2 stages.

- Stage I: both f and T are trained (**trigger generation**).
- Stage II: only f is trained while T is fixed (**backdoor injection**).



Images	Patched	Blended	ReFool	WaNet	LIRA
Backdoor	8.7	1.4	2.3	38.6	60.8
Clean	6.1	10.1	13.1	17.4	40.0
Both	7.4	5.7	7.7	28.0	50.4

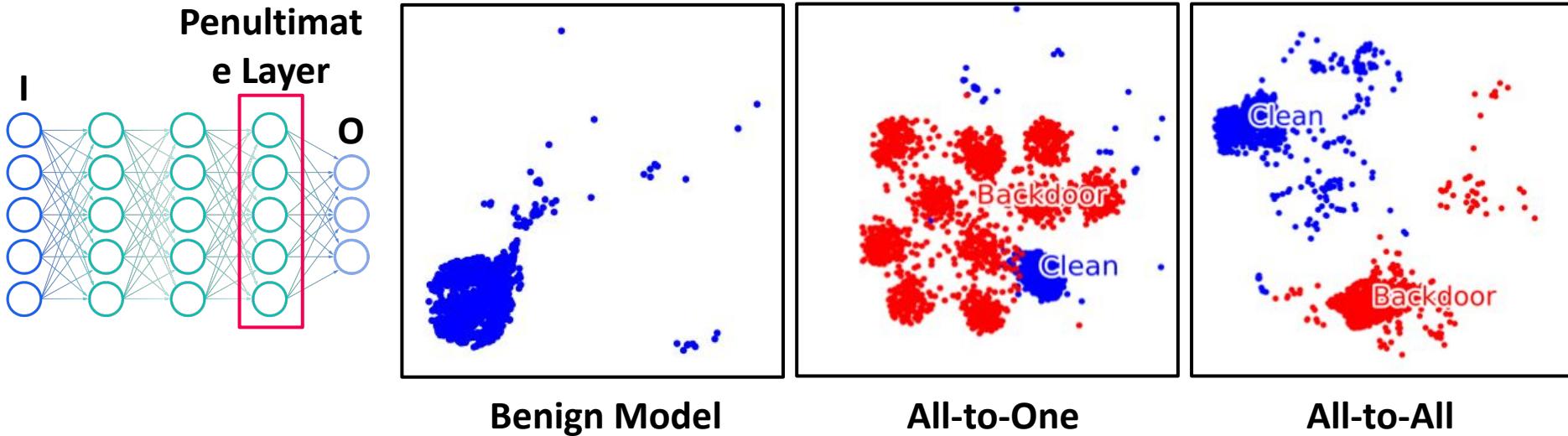
Human Inspection Tests - Each tester is trained to recognize the triggered image. Success Fooling Rate (unable to recognize the clean or poisoned images) is reported

Conclusions:

- LIRA has significantly higher success fooling rates.
- LIRA's stealthiness causes increasing confusion between the testers.

Input Space is Optimized, How About Latent Space?

Activations of the last hidden layer (penultimate) with 2-dimensional t-SNE projections. There exists a clear separation between the poisoned and clean data of a **predicted** class. Activation Clustering detects such separations and removes poisoned data, then re-trains the model.



We observe such separations in the existing methods, including BadNets [Gu et al 2017], WaNet [Nguyen et al 2021] & LIRA [Doan et al 2022].

Wasserstein Backdoor: Imperceptible Input And Latent Modification

- Solve the constrained optimization problem:

$$\arg \min_{\theta} \sum_{i=1}^N \alpha \mathcal{L}(f_{\theta}(x_i), y_i) + \beta \mathcal{L}(f_{\theta}(\mathcal{T}_{\xi}(\theta)(x_i)), \eta(y_i))$$

s. t. (1) $\xi = \arg \min_{\xi} \sum_{i=1}^N \mathcal{L}(f_{\theta}(\mathcal{T}_{\xi}(x_i)), \eta(y_i)) + \mathcal{R}_{\phi}(\mathcal{F}_c, \mathcal{F}_b)$

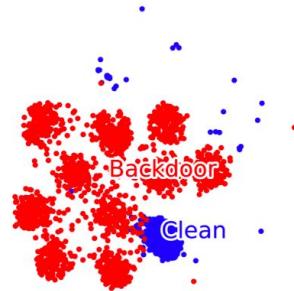
(2) $d(T(x), x) \leq \epsilon$

minimize the difference in the latent space

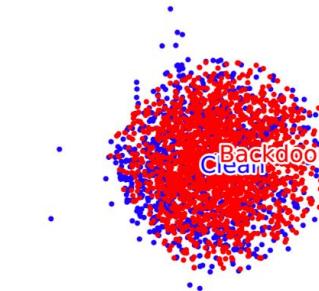
- The trigger function can be defined as:

$$T_{\xi}(x) = x + g_{\xi}(x), \|g_{\xi}(x)\|_{\infty} \leq \epsilon$$

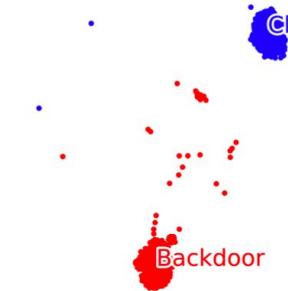
Learned Latent Space is Inseparable



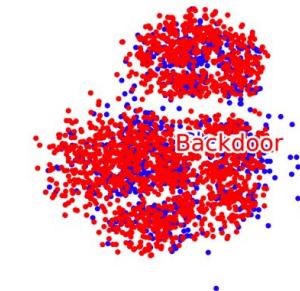
(a) All-to-one: LIRA



(b) All-to-one: WB

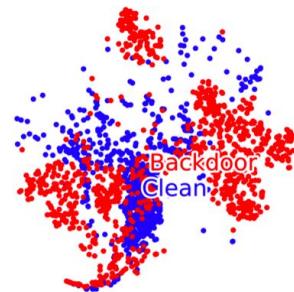


(c) All-to-all: LIRA

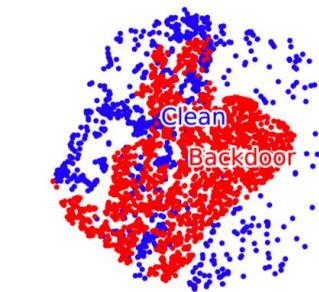


(d) All-to-all: WB

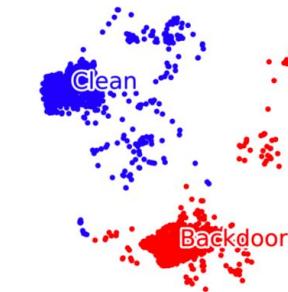
MNIST: t-SNE embedding in the latent space.



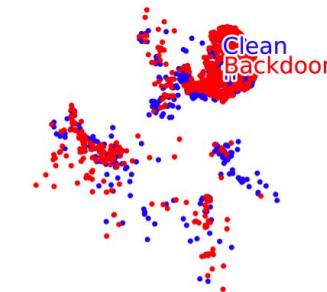
(a) All-to-one: LIRA



(b) All-to-one: WB



(c) All-to-all: LIRA



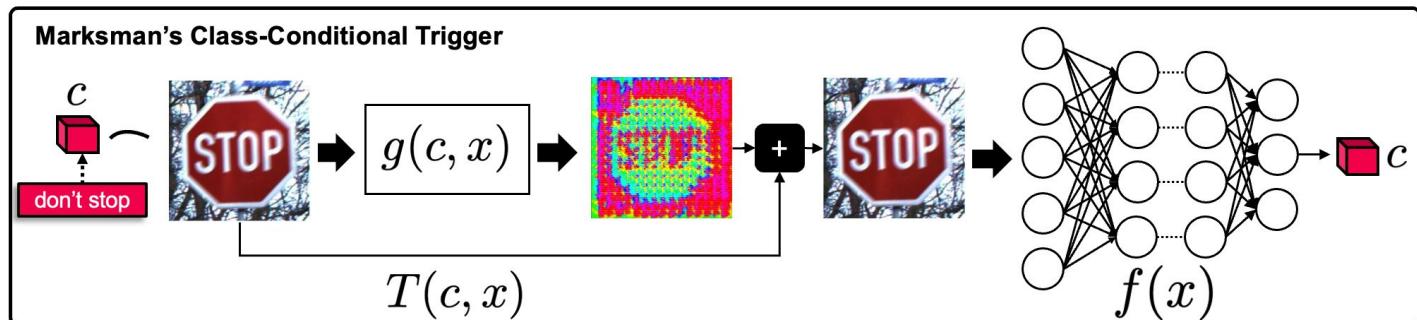
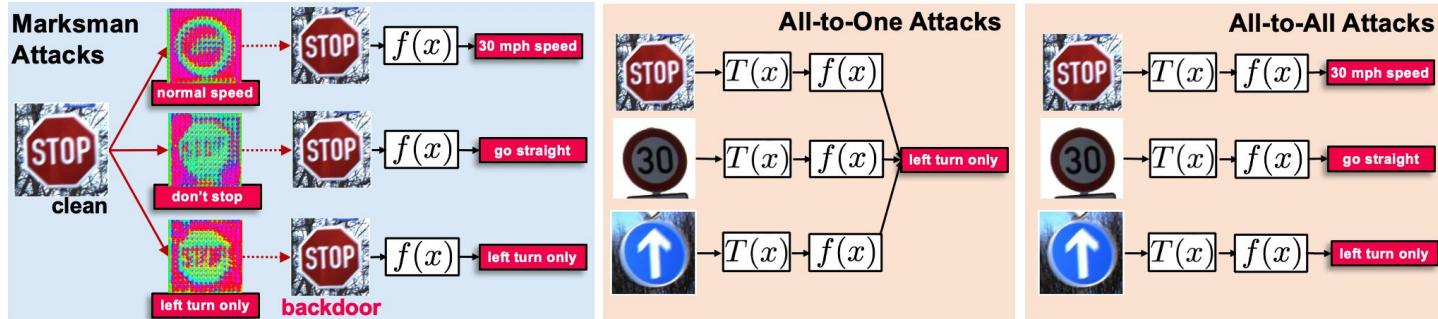
(d) All-to-all: WB

CIFAR10: t-SNE embedding in the latent space.

Marksman Backdoor:

Backdoor Attacks with Arbitrary Target Class

The first work to extend single-payload attack to **multi-trigger** and **multi-payload** backdoor with the capability of misclassifying an input to any target class.



Watermarking

- Protect intellectual property (IP)
- Image, video, ..., **Deep Neural Network (DNN)**



(Source: online)

- Training DNN models can be very expensive

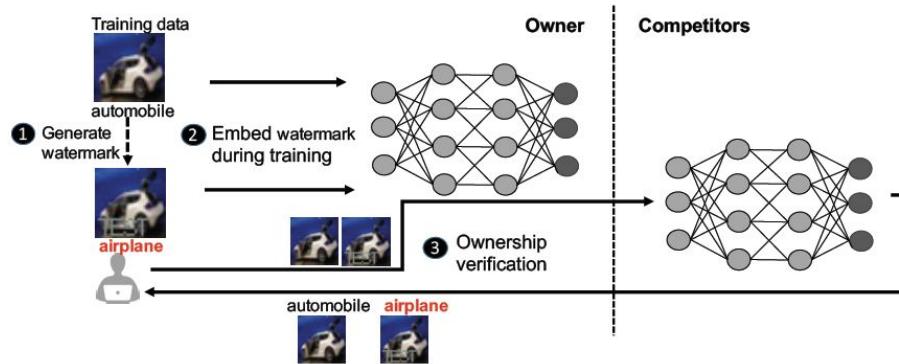
BERT: 256 TPU-chip days ~ \$6,912

GPT-3: 355 Tesla-V100 years ~ \$4,600,000



Prior Methods for Watermarking

- Inserts to feature space (Uchida et al. & Rouhani et al.)
- Exploits adversarial example (Le Merrer et al.)
- Utilizes backdoor (Adi et al.)



rely on end-to-end
retraining key samples

Adi, Y.; Baum, C.; Cisse, M.; Pinkas, B.; and Keshet, J. 2018. Turning your weakness into a strength: Watermarking deep neural networks by backdooring. In Proceedings of 27th USENIX Security Symposium (USENIX Security), 1615–1631.

Uchida, Y.; Nagai, Y.; Sakazawa, S.; and Satoh, S. 2017. Embedding watermarks into deep neural networks. In Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval (ICMR), 269–277.

Darvish Rouhani, B.; Chen, H.; and Koushanfar, F. 2019. DeepSigns: an end-to-end watermarking framework for ownership protection of deep neural networks. In Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 485–497.

Le Merrer, E.; Perez, P.; and Trédan, G. 2020. Adversarial frontier stitching for remote neural network watermarking. Neural Computing and Applications, 32(13): 9233–9244.

Identification for Deep Neural Network: Simply Adjusting a Few Weights

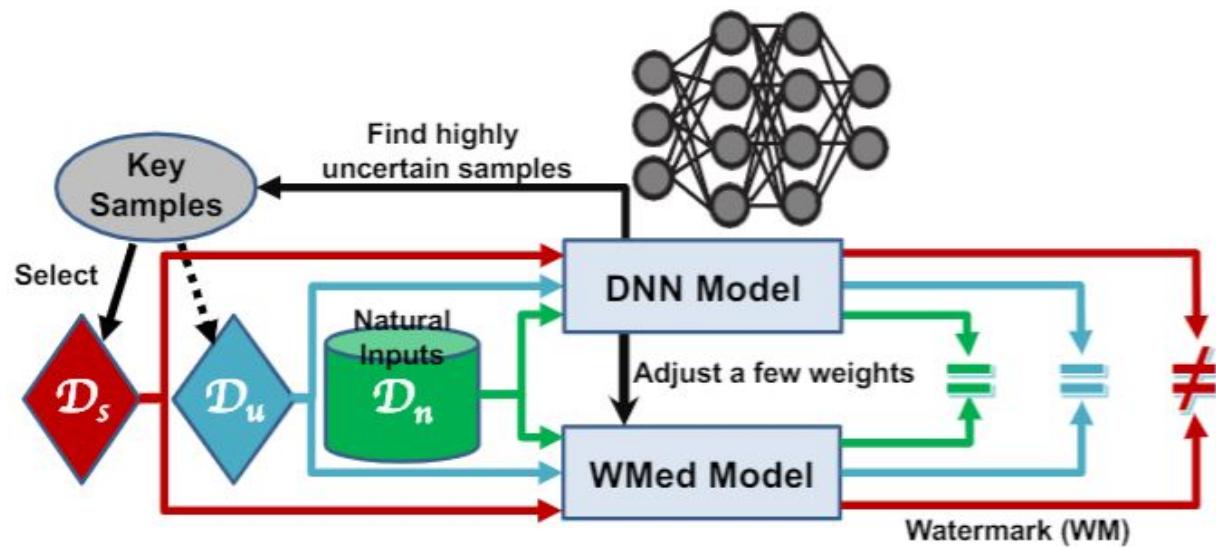
ICDE 2022

- Watermarking by only **adjusting a few weights**, and extend watermarking to **identification**
- Adopt the parameter searching concept from fault attacks

D_s : selected key samples

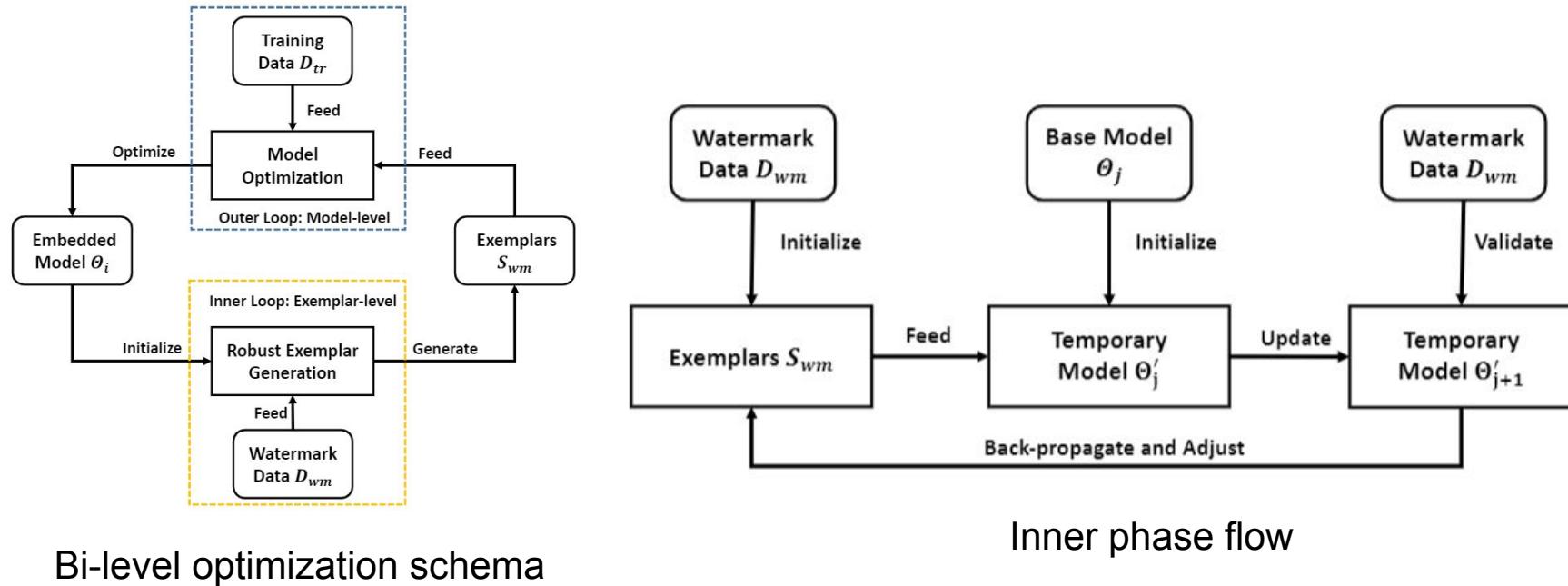
D_u : unselected key samples

D_n : natural inputs



Robust Watermarking for Deep Neural Networks via Bi-level Optimization

- Inner phase: Optimizing the example-level problem to generate robust key samples;
- Outer phase: mask adaptive optimization to achieve robustness of the projected DNN models.



Robustness of Watermarking

Watermark is used for IP ownership verification. It should not be easily removed or overwritten.

Transformation Attacks: attempt to remove the watermark while retaining the accuracy of the DNN model

- Fine-tuning
- Model-pruning
- Watermark overwriting

Comparison to prior works on fidelity and robustness against overwriting (20 key samples)

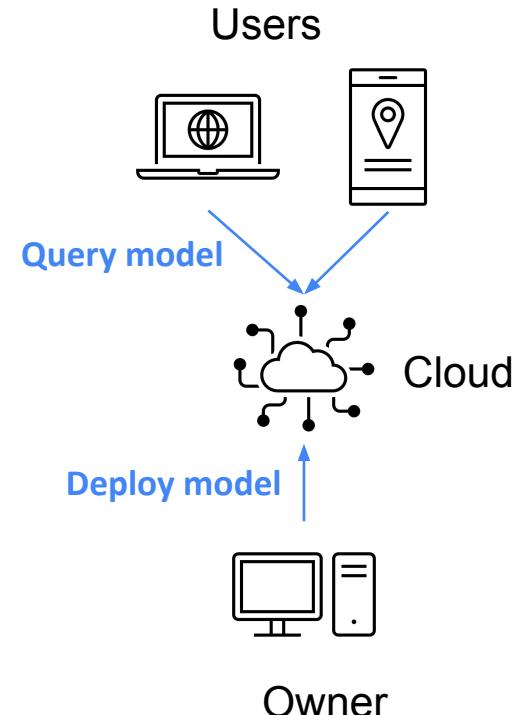
Method	Accuracy	Robustness
[Uchida et al.]	~0.3%	70~96%
[Rouhani et al.]	~0.5%	58%
[Adi et al.]	~0.3%	95%
Proposed	~0.05%	100%

Types of Watermarking

Types	Fragile	Robust
Photo/Media	Disappears when the photo is processed, e.g., with compression or resizing	Survives a variety of transformations, e.g., photo filter, conversion to a different format
DNN	Any malicious modification will destroy the watermark	Robust against a certain degree of modification that may include fine-tuning, transfer learning, pruning, etc.
Application	Integrity Verification	Ownership Verification

Integrity Authentication

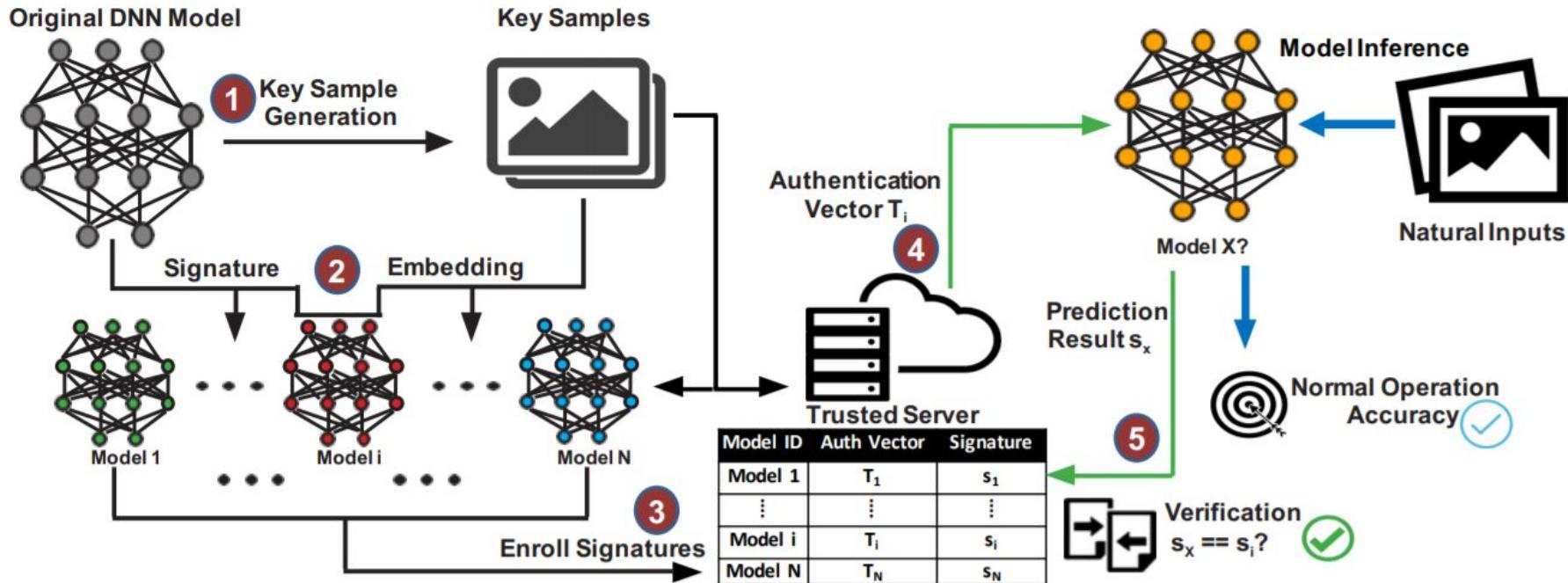
- Machine learning as a service (MLaaS)
- The supply chain of models:
 - multiple parties and vendors
 - data, algorithm, and infrastructure are vulnerable to breach
- Maliciously altered models
 - poisoning or backdoor attacks
 - impair the integrity, reputation, and profit of the model owner



DeepAuth: A DNN Authentication Framework by Model-Unique and Fragile Signature Embedding

Embed a fragile signature without affecting the performance

[AAAI 2022](#)



Integrity Authentication in Tree Models

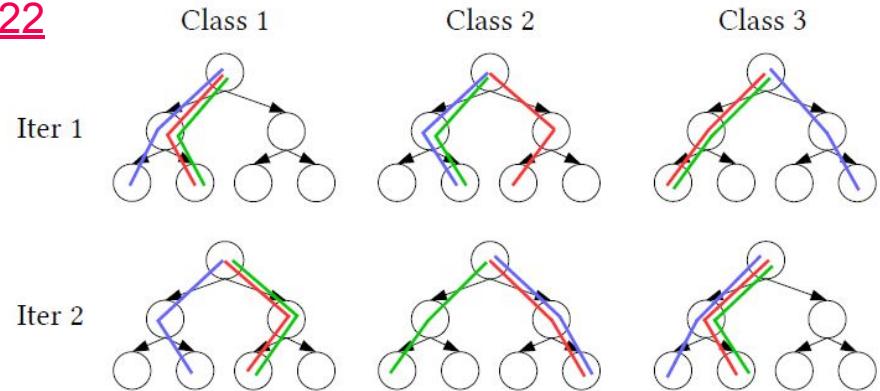
KDD 2022

Boosted Tree Models

- Ensemble of decision trees
- Typically produce robust and fairly accurate results
- Interpretability

Challenges

- Deep learning integrity authentication methods require gradients
 - tree models are indifferentiable
- Many deep learning signature embedding methods require retraining
 - appending more trees increases model size and hurts the inference performance
- Replacing a subset of existing trees is still an open research
 - a tree is generated on the results of the previous trees



Instance	Top class	Second class
—	1	2
—	1	3
—	3	1

An example for signature key selection

Outline

1. Vector similarity functions
2. Vector compressions
3. Vector similarity search
4. Maximum inner product search (MIPS)
5. Fast neural ranking
6. GPU computing
7. GCWSNet, hashing algorithms
8. Boosted trees, ABC-boost
9. Privacy
10. Security
- 11. Distributed, adaptive, and federated learning**
12. Others: generative AI models, NLP, knowledge graphs, multi-modal, cross-modal, advertising

Distributed, Adaptive, or Federated Learning

ICML'23, [Analysis of Error Feedback in Federated Non-Convex Optimization with Biased Compression](#)

JMLR'23, [Sharper Analysis for Minibatch Stochastic Proximal Point Method: Stability, Smoothness, and Deviation](#)

UAI'23, [Fed-LAMB: Layer-wise and Dimension-wise Locally Adaptive Federated Learning](#)

ICLR'23, [Improved Convergence of Differential Private SGD with Gradient Clipping](#)

ICLR'22, [On Distributed Adaptive Optimization with Gradient Compression](#)

NeurIPS'22, [On Convergence of FedProx: Local Dissimilarity Invariant Bounds, Non-smoothness and Beyond](#)

BIGDATA'22, [Communication-Efficient TeraByte-Scale Model Training Framework for Online Advertising](#)

ACML'22, [On the Convergence of Decentralized Adaptive Gradient Methods](#)

ACML'21, [An Optimistic Acceleration of AMSGrad for Nonconvex Optimization](#)

FODS'20, [Toward Communication Efficient Adaptive Gradient Method](#)

JMLR'20, [On Convergence of Distributed Approximate Newton Methods: Globalization, Sharper Bounds and](#)

NeurIPS'20, [Towards Better Generalization of Adaptive Gradient Methods](#)

MLSys'20, [Distributed Hierarchical GPU Parameter Server for Massive Scale Deep Learning Ads Systems](#)

CIKM'19, [AIBox: CTR Prediction Model Training on a Single Node](#)

Communication Efficient Adaptive Gradient Method

FODS'20, [Toward Communication Efficient Adaptive Gradient Method](#)

we consider the following formulation for distributed training, with N works (nodes)

$$\min_x \frac{1}{N} \sum_{i=1}^N f_i(x)$$

where f_i can be considered as the averaged loss over data at worker i and the function can only be accessed by the worker itself. For instance, for training neural nets, f_i can be viewed as the average loss of data located at the i -th node.

Local SGD and Periodic Model Averaging

FODS'20, [Toward Communication Efficient Adaptive Gradient Method](#)

Algorithm 1 Local SGD (with N nodes)

```
1: Input: learning rate  $\alpha$ , current point  $x_t$ 
2:  $g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$ 
3: if  $t \bmod k \neq 0$  then
4:    $x_{t+1,i} \leftarrow x_{t,i} - \alpha g_{t,i}$ 
5: else
6:    $x_{t+1,i} \leftarrow \frac{1}{N} \sum_{j=1}^N (x_{t,j} - \alpha g_{t,i})$ 
7: end if
```

Local SGD is heavily used for training neural nets in federated learning.

We consider adaptive gradient methods to improve over local SGD.

Naive Local AMSGrad

FODS'20, [Toward Communication Efficient Adaptive Gradient Method](#)

Algorithm 2 Naive local AMSGrad (with N nodes)

```
1: Input: learning rate  $\alpha$ , point  $x_t$ ,  $m_{0,i} = 0$ ,  $\hat{v}_{0,i} = \epsilon \mathbf{1}$ ,  $\forall i$ 
2:  $g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$ 
3:  $m_{t,i} = \beta_1 m_{t-1,i} + (1 - \beta_1) g_{t,i}$ 
4:  $v_{t,i} = \beta_2 v_{t-1,i} + (1 - \beta_2) g_{t,i}^2$ 
5:  $\hat{v}_{t,i} = \max(v_{t,i}, \hat{v}_{t-1,i})$ 
6: if  $t \bmod k \neq 0$  then
7:    $x_{t+1,i} \leftarrow x_{t,i} - \alpha \frac{m_{t,i}}{\sqrt{\hat{v}_{t,i}}}$ 
8: else
9:    $x_{t+1,i} \leftarrow \frac{1}{N} \sum_{j=1}^N \left( x_{t,j} - \alpha \frac{m_{t,j}}{\sqrt{\hat{v}_{t,i}}} \right)$ 
10: end if
```

Naive local
AMSGrad fails to
converge

1. Each node runs AMSGrad locally.
2. The variables $\{x_{t,i}\}_{i=1}^N$ are averaged every k iterations.

Local AMSGrad That Works

FODS'20, [Toward Communication Efficient Adaptive Gradient Method](#)

Algorithm 3 Local AMSGrad (with N nodes)

```
1: Input: learning rate  $\alpha$ , point  $x_t$ ,  $m_{t,i} = 0$ ,  $\hat{v}_{0,i} = \epsilon \mathbf{1}$ ,  $\forall i$ 
2:  $g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$ 
3:  $m_{t,i} = \beta_1 m_{t-1,i} + (1 - \beta_1) g_{t,i}$ 
4:  $v_{t,i} = \beta_2 v_{t-1,i} + (1 - \beta_2) g_{t,i}^2$ 
5: if  $t \bmod k \neq 0$  then
6:    $\hat{v}_t = \hat{v}_{t-1}$ 
7:    $x_{t+1,i} \leftarrow x_{t,i} - \alpha \frac{m_{t,i}}{\sqrt{\hat{v}_t}}$ 
8: else
9:    $\hat{v}_t = \max\left(\frac{1}{N} \sum_{i=1}^N v_{t,i}, \hat{v}_{t-1}\right)$ 
10:   $x_{t+1,i} \leftarrow \frac{1}{N} \sum_{j=1}^N \left( x_{t,j} - \alpha \frac{m_{t,j}}{\sqrt{\hat{v}_t}} \right)$ 
11: end if
```

The key divergence mechanism is due to different adaptive learning rates on different nodes.

We force different nodes to have the same adaptive learning rate

Local AMSGrad That Works

FODS'20, [Toward Communication Efficient Adaptive Gradient Method](#)

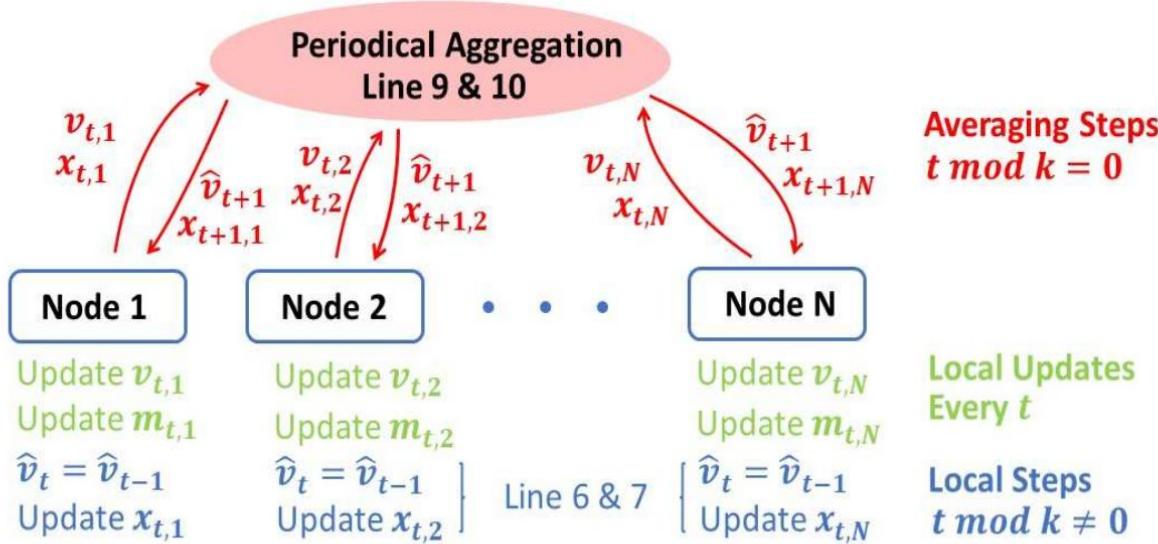


Figure 1: Illustration of the proposed local AMSGrad scheme (Algorithm 3) with shared adaptive learning rate. The local servers employ AMSGrad updates locally, and the global server aggregates the model parameters and second moment \hat{v} every k steps.

Experiments on Local AMSGrad

FODS'20, [Toward Communication Efficient Adaptive Gradient Method](#)

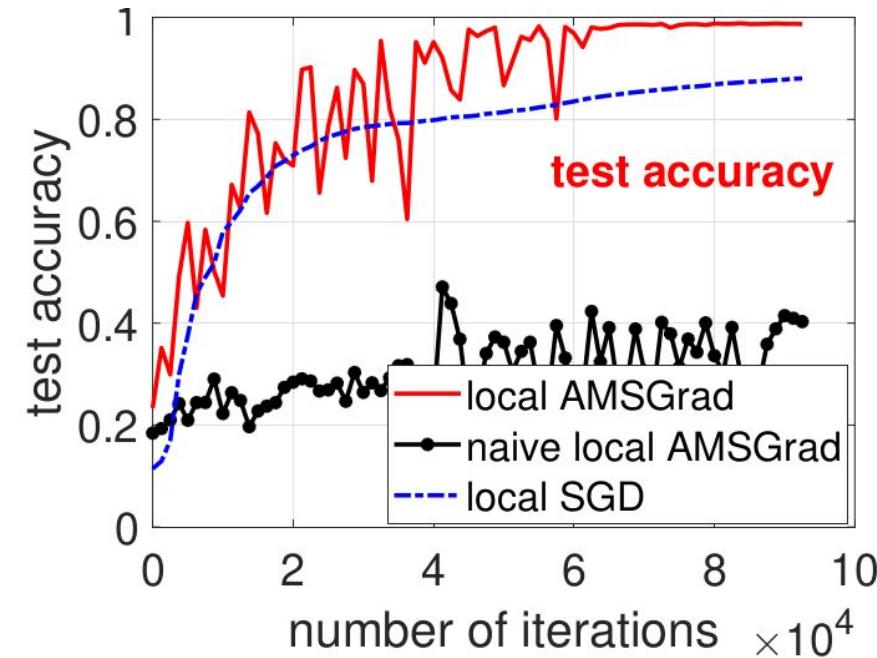
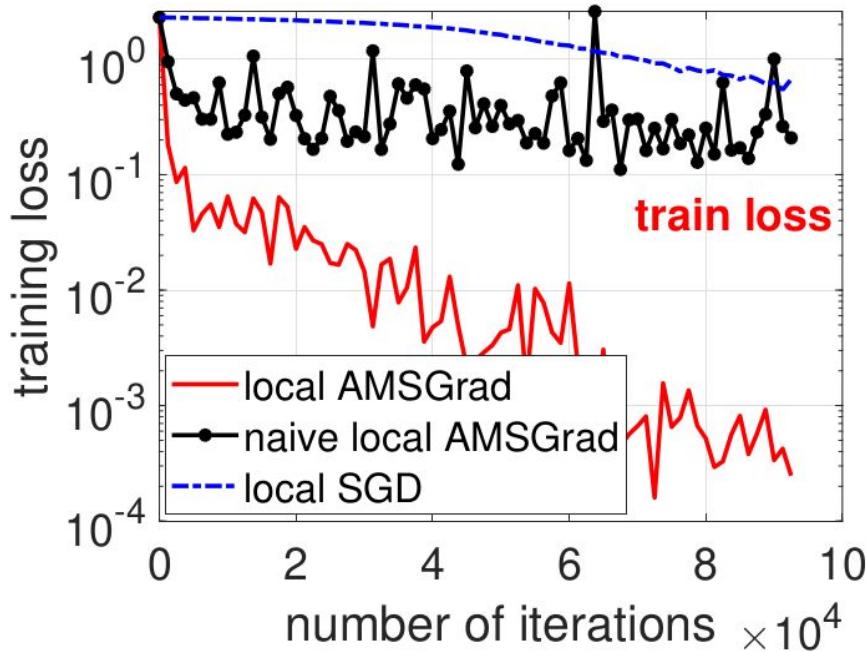


Figure 4: MNIST dataset: Performance comparison of three different algorithms.

Decentralized Adaptive Gradient Methods

ACML'22, [On the Convergence of Decentralized Adaptive Gradient Methods](#)

- Communication cost has become a major concern in (centralized) distributed computing.
- Decentralized distributed computing can be highly beneficial in (e.g.,) mobile computing.
- It is difficult to develop decentralized adaptive gradient methods that provably converge.
- We propose a general algorithmic framework that can convert existing adaptive gradient methods to their decentralized counterparts. We develop the generic decentralized framework on prototype methods: AMSGrad and AdaGrad.

DADAM Fails to Converge

ACML'22, [On the Convergence of Decentralized Adaptive Gradient Methods](#)

Algorithm 1 DADAM (with N nodes)

- 1: **Input:** α , current point X_t , $u_{\frac{1}{2},i} = \hat{v}_{0,i} = \epsilon \mathbf{1}$,
 $m_0 = 0$ and mixing matrix W
- 2: **for** $t = 1, 2, \dots, T$ **do**
- 3: **for all** $i \in [N]$ **do in parallel**
- 4: $g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$
- 5: $m_{t,i} = \beta_1 m_{t-1,i} + (1 - \beta_1) g_{t,i}$
- 6: $v_{t,i} = \beta_2 v_{t-1,i} + (1 - \beta_2) g_{t,i}^2$
- 7: $\hat{v}_{t,i} = \beta_3 \hat{v}_{t-1,i} + (1 - \beta_3) \max(\hat{v}_{t-1,i}, v_{t,i})$
- 8: $x_{t+\frac{1}{2},i} = \sum_{j=1}^N W_{ij} x_{t,j}$
- 9: $x_{t+1,i} = x_{t+\frac{1}{2},i} - \alpha \frac{m_{t,i}}{\sqrt{\hat{v}_{t,i}}}$
- 10: **end for**

Proposed by Nazari et al. (2019), the Decentralized ADAM (DADAM) is a decentralized version of ADAM. It admits a non-standard regret bound in the online setting. However, we can show the convergence failure of DADAM in the offline settings.

Unified Decentralized Adaptive Gradient Framework

ACML'22, [On the Convergence of Decentralized Adaptive Gradient Methods](#)

Algorithm 2 Decentralized Adaptive Gradient Method (with N nodes)

- 1: **Input:** α , initial point $x_{1,i} = x_{init}$, $u_{\frac{1}{2},i} = \hat{v}_{0,i}$, $m_{0,i} = 0$, mixing matrix W
 - 2: **for** $t = 1, 2, \dots, T$ **do**
 - 3: **for all** $i \in [N]$ **do in parallel**
 - 4: $g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$
 - 5: $m_{t,i} = \beta_1 m_{t-1,i} + (1 - \beta_1) g_{t,i}$
 - 6: $\hat{v}_{t,i} = r_t(g_{1,i}, \dots, g_{t,i})$
 - 7: $x_{t+\frac{1}{2},i} = \sum_{j=1}^N W_{ij} x_{t,j}$
 - 8: $\tilde{u}_{t,i} = \sum_{j=1}^N W_{ij} \tilde{u}_{t-\frac{1}{2},j}$
 - 9: $u_{t,i} = \max(\tilde{u}_{t,i}, \epsilon)$
 - 10: $x_{t+1,i} = x_{t+\frac{1}{2},i} - \alpha \frac{m_{t,i}}{\sqrt{u_{t,i}}}$
 - 11: $\tilde{u}_{t+\frac{1}{2},i} = \tilde{u}_{t,i} - \hat{v}_{t-1,i} + \hat{v}_{t,i}$
 - 12: **end for**
-

Theorem 2 Assume A1-A4. When $\alpha \leq \frac{\epsilon^{0.5}}{16L}$, Algorithm 2 yields the following regret bound

$$\begin{aligned} \frac{1}{T} \sum_{t=1}^T \mathbb{E} \left[\left\| \frac{\nabla f(\bar{X}_t)}{\bar{U}_t^{1/4}} \right\|^2 \right] &\leq C_1 \left(\frac{1}{T\alpha} (\mathbb{E}[f(Z_1)] - \min_x f(x)) + \alpha \frac{d\sigma^2}{N} \right) + C_2 \alpha^2 d \\ &\quad + C_3 \alpha^3 d + \frac{1}{T\sqrt{N}} (C_4 + C_5 \alpha) \mathbb{E} \left[\sum_{t=1}^T \|(-\hat{V}_{t-2} + \hat{V}_{t-1})\|_{abs} \right] \end{aligned}$$

Decentralized AMSGrad and AdaGrad

ACML'22, [On the Convergence of Decentralized Adaptive Gradient Methods](#)

Algorithm 3 Decentralized AMSGrad (N nodes)

- 1: **Input:** learning rate α , initial point $x_{1,i} = x_{init}$, $u_{\frac{1}{2},i} = \hat{v}_{0,i} = \epsilon \mathbf{1}$ (with $\epsilon \geq 0$), $m_{0,i} = 0$, mixing matrix W
- 2: **for** $t = 1, 2, \dots, T$ **do**
- 3: **for all** $i \in [N]$ **do in parallel**
- 4: $g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$
- 5: $m_{t,i} = \beta_1 m_{t-1,i} + (1 - \beta_1) g_{t,i}$
- 6: $v_{t,i} = \beta_2 v_{t-1,i} + (1 - \beta_2) g_{t,i}^2$
- 7: $\hat{v}_{t,i} = \max(\hat{v}_{t-1,i}, v_{t,i})$
- 8: $x_{t+\frac{1}{2},i} = \sum_{j=1}^N W_{ij} x_{t,j}$
- 9: $\tilde{u}_{t,i} = \sum_{j=1}^N W_{ij} \tilde{u}_{t-\frac{1}{2},j}$
- 10: $u_{t,i} = \max(\tilde{u}_{t,i}, \epsilon)$
- 11: $x_{t+1,i} = x_{t+\frac{1}{2},i} - \alpha \frac{m_{t,i}}{\sqrt{u_{t,i}}}$
- 12: $\tilde{u}_{t+\frac{1}{2},i} = \tilde{u}_{t,i} - \hat{v}_{t-1,i} + \hat{v}_{t,i}$
- 13: **end for**

Algorithm 4 Decentralized AdaGrad (N nodes)

- 1: **Input:** learning rate α , initial point $x_{1,i} = x_{init}$, $u_{\frac{1}{2},i} = \hat{v}_{0,i} = \epsilon \mathbf{1}$ (with $\epsilon \geq 0$), $m_{0,i} = 0$, mixing matrix W
- 2: **for** $t = 1, 2, \dots, T$ **do**
- 3: **for all** $i \in [N]$ **do in parallel**
- 4: $g_{t,i} \leftarrow \nabla f_i(x_{t,i}) + \xi_{t,i}$
- 5: $m_{t,i} = \beta_1 m_{t-1,i} + (1 - \beta_1) g_{t,i}$
- 6: $\hat{v}_{t,i} = \frac{t-1}{t} \hat{v}_{t-1,i} + \frac{1}{t} g_{t,i}^2$
- 7: $x_{t+\frac{1}{2},i} = \sum_{j=1}^N W_{ij} x_{t,j}$
- 8: $\tilde{u}_{t,i} = \sum_{j=1}^N W_{ij} \tilde{u}_{t-\frac{1}{2},j}$
- 9: $u_{t,i} = \max(\tilde{u}_{t,i}, \epsilon)$
- 10: $x_{t+1,i} = x_{t+\frac{1}{2},i} - \alpha \frac{m_{t,i}}{\sqrt{u_{t,i}}}$
- 11: $\tilde{u}_{t+\frac{1}{2},i} = \tilde{u}_{t,i} - \hat{v}_{t-1,i} + \hat{v}_{t,i}$
- 12: **end for**

Compressed Distributed Adaptive Optimization

ICLR'22, [On Distributed Adaptive Optimization with Gradient Compression](#)

In distributed optimization (e.g., DSGD), transmitting the gradients between local nodes and server could be costly and slow for large models. Gradient compression has been a popular solution to resolve this issue.

Unbiased compressors:

- Stochastic quantization, e.g., QSGD (Alistarh et al., 2017)
- Stochastic sparsification

Biased compressors:

- Random-K, Top-K
- Fixed quantization, e.g., SignSGD

COMP-AMS Algorithm

ICLR'22, [On Distributed Adaptive Optimization with Gradient Compression](#)

We focus on gradient compression with adaptive AMSGrad (Reddi et al. 2018)

Algorithm 2 Distributed COMP-AMS with error feedback (EF)

```
1: Input: parameters  $\beta_1, \beta_2, \epsilon$ , learning rate  $\eta_t$ 
2: Initialize: central server parameter  $\theta_1 \in \mathbb{R}^d \subseteq \mathbb{R}^d$ ;  $e_{1,i} = 0$  the error accumulator for each
   worker;  $m_0 = 0, v_0 = 0, \hat{v}_0 = 0$ 
3: for  $t = 1, \dots, T$  do
4:   parallel for worker  $i \in [n]$  do:
5:     Receive model parameter  $\theta_t$  from central server
6:     Compute stochastic gradient  $g_{t,i}$  at  $\theta_t$ 
7:     Compute the compressed gradient  $\tilde{g}_{t,i} = \mathcal{C}(g_{t,i} + e_{t,i})$ 
8:     Update the error  $e_{t+1,i} = e_{t,i} + g_{t,i} - \tilde{g}_{t,i}$ 
9:     Send  $\tilde{g}_{t,i}$  back to central server
10:  end parallel
11:  Central server do:
12:     $\bar{g}_t = \frac{1}{n} \sum_{i=1}^n \tilde{g}_{t,i}$ 
13:     $m_t = \beta_1 m_{t-1} + (1 - \beta_1) \bar{g}_t$ 
14:     $v_t = \beta_2 v_{t-1} + (1 - \beta_2) \bar{g}_t^2$ 
15:     $\hat{v}_t = \max(v_t, \hat{v}_{t-1})$ 
16:    Update the global model  $\theta_{t+1} = \theta_t - \eta_t \frac{m_t}{\sqrt{\hat{v}_t + \epsilon}}$ 
17: end for
```

We consider biased compressors
with Error Feedback (EF)

In each round, the local node:

1. Compute and send stochastic gradient
2. Update local error tracker

The server:

1. Aggregate the local gradients
2. Perform AMSGrad update

Convergence Rates

ICLR'22, [On Distributed Adaptive Optimization with Gradient Compression](#)

Convergence rate for non-convex optimization

Theorem 1. Denote $C_0 = \sqrt{\frac{4(1+q^2)^3}{(1-q^2)^2} G^2 + \epsilon}$, $C_1 = \frac{\beta_1}{1-\beta_1} + \frac{2q}{1-q^2}$, $\theta^* = \arg \min f(\theta)$ defined as (1). Under Assumptions 1 to 4, with $\eta_t = \eta \leq \frac{\epsilon}{3C_0\sqrt{2L\max\{2L,C_1\}}}$, Algorithm 2 satisfies

$$\begin{aligned} \frac{1}{T} \sum_{t=1}^T \mathbb{E}[\|\nabla f(\theta_t)\|^2] &\leq 2C_0 \left(\frac{\mathbb{E}[f(\theta_1) - f(\theta^*)]}{T\eta} + \frac{\eta L\sigma^2}{n\epsilon} + \frac{3\eta^2 LC_0 C_1^2 \sigma^2}{n\epsilon^2} \right. \\ &\quad \left. + \frac{12\eta^2 q^2 LC_0 \sigma_g^2}{(1-q^2)^2 \epsilon^2} + \frac{(1+C_1)G^2 d}{T\sqrt{\epsilon}} + \frac{\eta(1+2C_1)C_1 L G^2 d}{T\epsilon} \right). \end{aligned}$$

Convergence Rates

ICLR'22, [On Distributed Adaptive Optimization with Gradient Compression](#)

Linear speedup in number of nodes n

Corollary 2. *Under the same setting as Theorem 1, set $\eta = \min\{\frac{\epsilon}{3C_0\sqrt{2L\max\{2L,C_1\}}}, \frac{\sqrt{n}}{\sqrt{T}}\}$. The COMP-AMS iterates admit*

$$\frac{1}{T} \sum_{t=1}^T \mathbb{E}[\|\nabla f(\theta_t)\|^2] \leq \mathcal{O}\left(\frac{1}{\sqrt{nT}} + \frac{\sigma^2}{\sqrt{nT}} + \frac{n(\sigma^2 + \sigma_g^2)}{T}\right). \quad (2)$$

Single-machine rate matches the full-precision AMSGrad

Corollary 1. *When $n = 1$, under Assumption 1 to Assumption 4, setting the stepsize as $\eta = \min\{\frac{\epsilon}{3C_0\sqrt{2L\max\{2L,C_1\}}}, \frac{1}{\sqrt{T}}\}$, Algorithm 2 satisfies*

$$\frac{1}{T} \sum_{t=1}^T \mathbb{E}[\|\nabla f(\theta_t)\|^2] \leq \mathcal{O}\left(\frac{1}{\sqrt{T}} + \frac{\sigma^2}{\sqrt{T}} + \frac{d}{T}\right).$$

Error feedback fixes the convergence issue of biased compression for AMSGrad!

Experiments

ICLR'22, [On Distributed Adaptive Optimization with Gradient Compression](#)

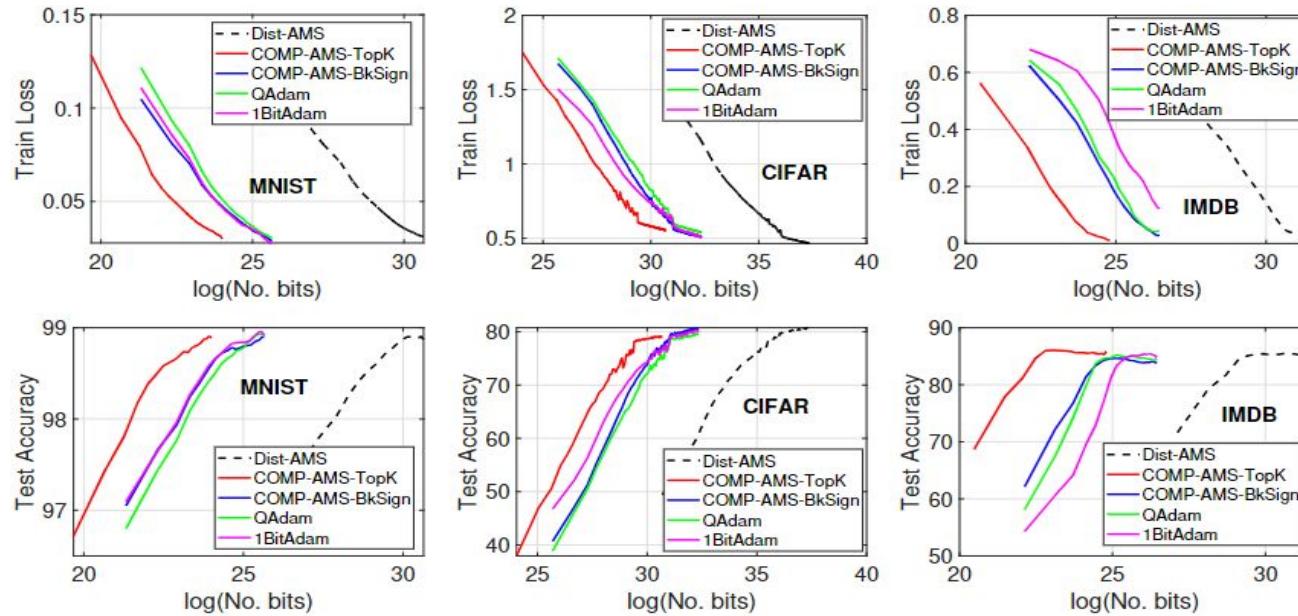


Figure 2: Train loss and Test accuracy vs. No. bits transmitted, on MNIST + CNN, CIFAR-10 + LeNet and IMDB + LSTM with $n = 16$ local workers.

COMP-AMS matches full-precision training with 30 - 100x communication reduction

Experiments

ICLR'22, [On Distributed Adaptive Optimization with Gradient Compression](#)

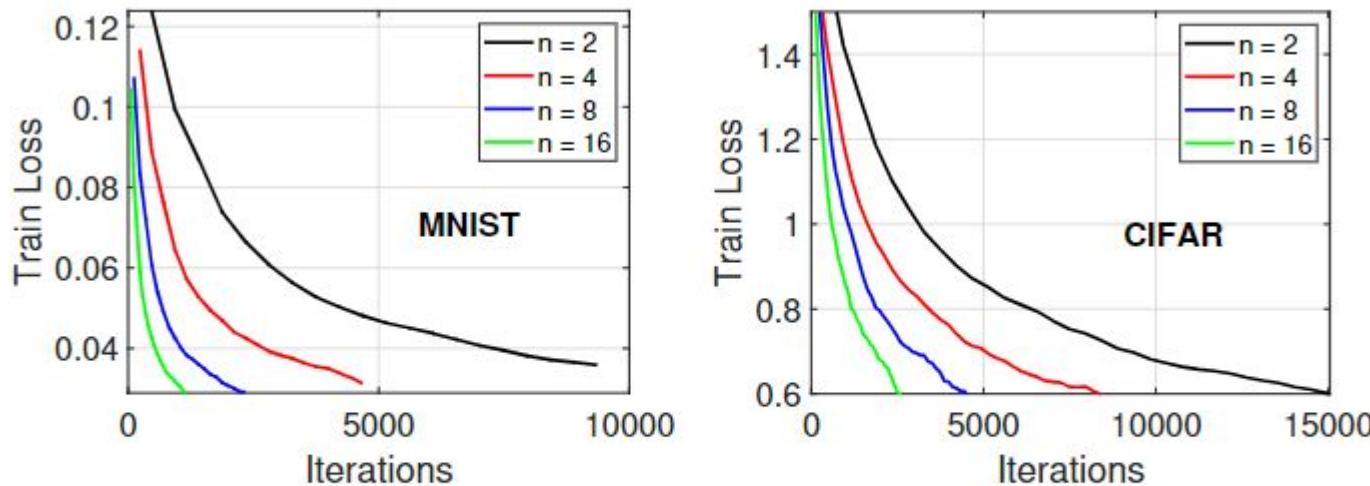


Figure 3: The linear speedup of COMP-AMS with varying n . **Left:** MNIST with **Block-Sign** compressor on CNN. **Right:** CIFAR-10 with **Top- k -0.01** compression on LeNet.

Validate the linear speedup property of convergence of COMP-AMS vs. # of local nodes

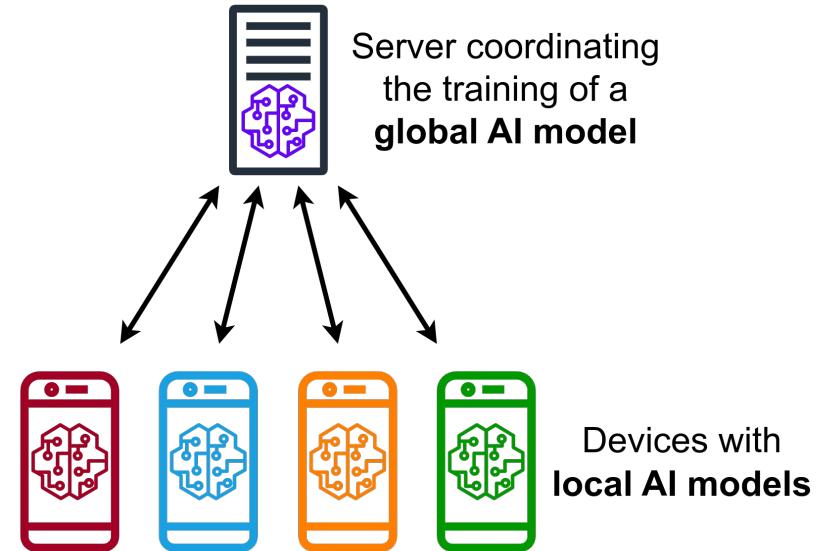
Compression for Federated Learning

ICML'23, [Analysis of Error Feedback in Federated Non-Convex Optimization with Biased Compression](#)

Federated learning (FL) has been an important topic in ML and has seen many applications in 5G/6G wireless communications, Internet of Things (IoT), financial fraud detection, input method editor (IME), advertising (ads), health records, ...

In this paper, we consider a standard centralized FL system, with a global server and many local clients (data silos, mobile phones, IoT devices).

In each round, the clients train the models locally, and send back the model updates to the server for aggregation.



Compression for Federated Learning

ICML'23, [Analysis of Error Feedback in Federated Non-Convex Optimization with Biased Compression](#)

Three key challenges in FL algorithm design, theory, and deployment:

1. **Communication cost**: Limited wireless bandwidth often cannot afford transmitting full-precision large models.
2. **Data heterogeneity**: Local clients' data are non-iid. Thus, the local training loss (expectation over local data distribution) are different from the global training loss.
3. **Partial participation**: In cross-device FL, clients may drop and join in each round, thus partially participating in FL training.

Compression for Federated Learning

ICML'23, [Analysis of Error Feedback in Federated Non-Convex Optimization with Biased Compression](#)

Error Feedback (Seide et al. 2014; Stich et al. 2018) has not been studied under the practical FL setting, thoroughly.

- We focus on the **Fed-EF** framework and provide the analysis of EF with **local steps, data heterogeneity, and communication compression**, to achieve a sharp convergence rate compared with state-of-the-art FL methods.
- We propose **Fed-EF-AMS**, the first adaptive (Adam-type) FL algorithm with communication compression.
- We develop the analysis of EF under **partial participation**, showing an extra slow down factor which is related to the client sampling ratio.

[1] [Seide et al.](#), 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech DNNs, *INTERSPEECH* 2014

[2] [Stich et al.](#), Sparsified SGD with memory, *NeurIPS* 2018

Fed-EF Algorithm

ICML'23, [Analysis of Error Feedback in Federated Non-Convex Optimization with Biased Compression](#)

Fed-EF-SGD: The server performs SGD updates

Fed-EF-AMS: The server performs AMSGrad (Reddi et al. 2019) updates

For distributed gradient compression with adaptive optimizers, see Li et al. 2022.

[1] [Reddi et al.](#), On the convergence of Adam and Beyond, *ICLR* 2019

[2] [Li et al.](#), On distributed adaptive optimization with gradient compression, *ICLR* 2022

Algorithm 1 Fed-EF: Compressed FL with Error Feedback

```
1: Input: learning rates  $\eta, \eta_l$ ; parameters  $\beta_1, \beta_2, \epsilon$ 
2: Initialize: global model  $\theta_1 \in \mathbb{R}^d \subseteq \mathbb{R}^d$ ; local error
   accumulator  $e_{1,i} = 0$ ;  $m_0 = 0, v_0 = 0, \hat{v}_0 = 0$ 
3: for  $t = 1, \dots, T$  do
4:   parallel for worker  $i \in [n]$  do:
5:     Receive global model  $\theta_t$  from server, set  $\theta_{t,i}^{(1)} = \theta_t$ 
6:     for  $k = 1, \dots, K$  do
7:       Compute stochastic gradient  $g_{t,i}^{(k)}$  at  $\theta_{t,i}^{(k)}$ 
8:       Local update  $\theta_{t,i}^{(k+1)} = \theta_{t,i}^{(k)} - \eta_l g_{t,i}^{(k)}$ 
9:     end for
10:    Compute local update  $\Delta_{t,i} = \theta_t - \theta_{t,i}^{(K+1)}$ 
11:    Send  $\tilde{\Delta}_{t,i} = \mathcal{C}(\Delta_{t,i} + e_{t,i})$  to server
12:    Update the error  $e_{t+1,i} = e_{t,i} + \Delta_{t,i} - \tilde{\Delta}_{t,i}$ 
13:  end parallel
14:  Central server do:
15:    Global aggregation  $\bar{\Delta}_t = \frac{1}{n} \sum_{i=1}^n \tilde{\Delta}_{t,i}$ 
16:    Global update  $\theta_{t+1} = \theta_t - \eta \bar{\Delta}_t$  { Fed-EF-SGD }
17:     $m_t = \beta_1 m_{t-1} + (1 - \beta_1) \bar{\Delta}_t$  { Fed-EF-AMS }
18:     $v_t = \beta_2 v_{t-1} + (1 - \beta_2) \bar{\Delta}_t^2, \hat{v}_t = \max(v_t, \hat{v}_{t-1})$ 
19:    Global update  $\theta_{t+1} = \theta_t - \eta \frac{m_t}{\sqrt{\hat{v}_t + \epsilon}}$ 
20:  end for
```

Convergence Analysis

ICML'23, [Analysis of Error Feedback in Federated Non-Convex Optimization with Biased Compression](#)

Contrastive compressor: $\|\mathcal{C}(x) - x\|^2 \leq q^2 \|x\|^2$, n: # of clients m: # of active clients

- Fed-SGD with biased compression, **without EF**: $O(\frac{1+q^2}{\sqrt{TKn}} + q^2 \cdot const)$
- **Fed-EF-SGD and Fed-EF-AMS (full participation):**

$$O(\frac{1+q^2}{\sqrt{TKn}})$$

Matches full-precision rates
when q=0 (no compression)

- Fed-EF under **partial client participation (uniform sampling assumption)**

$$O(\frac{\sqrt{n}}{\sqrt{m}} \frac{(1+q^2)\sqrt{K}}{\sqrt{Tm}})$$

The full-precision rate under PP is
[Yang et al. ICLR'21]

An extra slow-down factor $\frac{\sqrt{n}}{\sqrt{m}}$: “**delayed error compensation**”

Partial participation introduces staleness to the local error accumulator. Updating with the stale information slows down convergence.

Convergence Analysis

ICML'23, [Analysis of Error Feedback in Federated Non-Convex Optimization with Biased Compression](#)

Table 1. Summary of theoretical convergence results from some existing works on distributed and federated learning with communication compression for non-convex optimization. “PP” stands for “partial participation”, and “# of Rounds” is the number of communication rounds required to achieve linear speedup, which is a common measure of the communication complexity of FL algorithms. T is the number of communication rounds, K is the number of local steps, and n is the total number of clients.

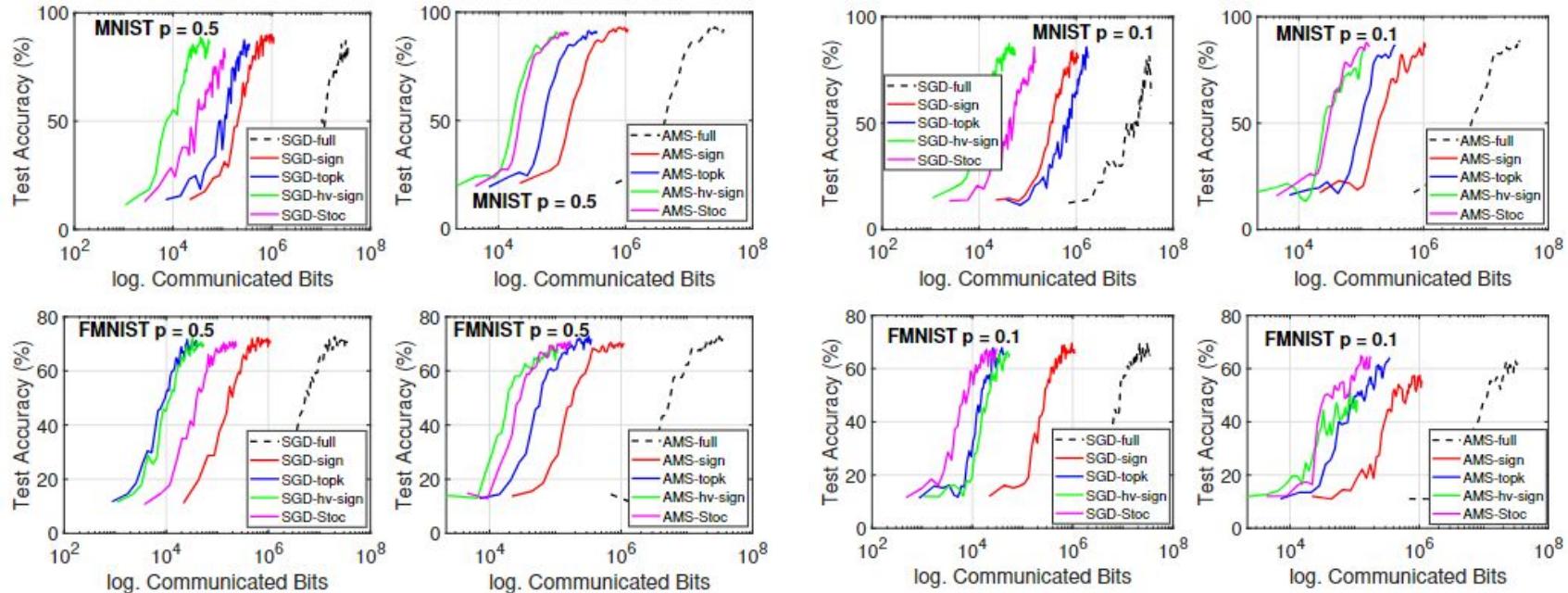
Reference	Local Step	Non-iid Data	PP	Adaptive Opt.	Compression	# of Rounds
Jiang and Agrawal (2018) ^a Li et al. (2022b)		✓			Unbiased Biased + EF	$T = \mathcal{O}(n)$ $T = \mathcal{O}(n^3)$
Reisizadeh et al. (2020)	✓				Unbiased	— ^b
Haddadpour et al. (2021)	✓	✓			Unbiased	$T = \mathcal{O}(Kn)$
Basu et al. (2019)	✓				Biased + EF	$T = \mathcal{O}(K^3 n^3)$
Gao et al. (2021)	✓				Biased + EF	$T = \mathcal{O}(Kn^3)$
Fed-EF (our paper) ^c	✓	✓	✓	✓	Biased + EF	$T = \mathcal{O}(Kn)$

Our algorithm and analysis cover local steps, data heterogeneity, partial participation and adaptive optimizer.

Experiments

ICML'23, [Analysis of Error Feedback in Federated Non-Convex Optimization with Biased Compression](#)

$p = m/n$ is the client participation rate



Fed-EF matches the performance of full-precision training with substantially reduced communication cost (30 - 100x)

Experiments

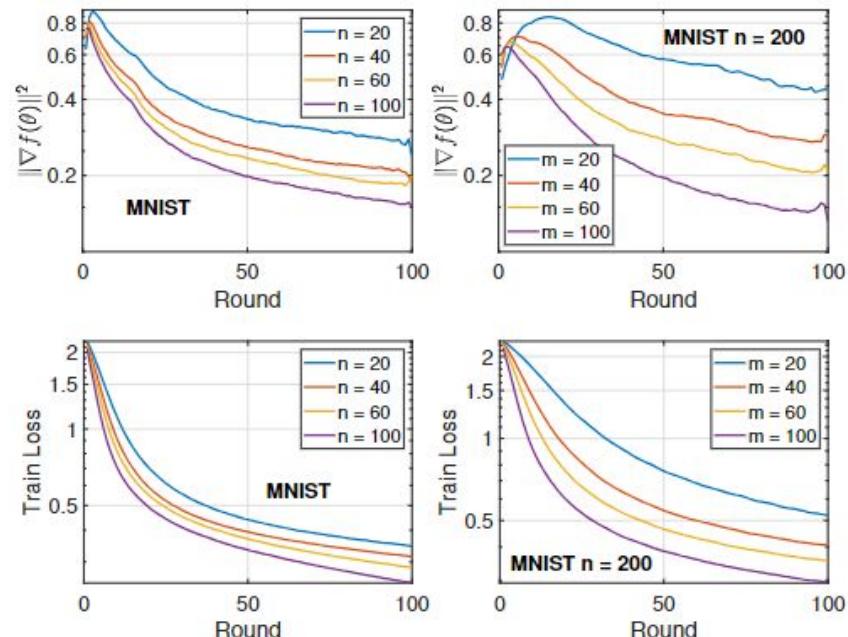
ICML'23, [Analysis of Error Feedback in Federated Non-Convex Optimization with Biased Compression](#)

Fed-EF with Topk-0.01 compression

Left: full participation, $n = 20, 40, 60, 100$

Right: Partial participation, $n = 200$

$m = 20, 40, 60, 100$



Linear speedup with n in full participation case

Faster speedup with m in partial participation (validating the extra $\frac{\sqrt{n}}{\sqrt{m}}$ factor)

Outline

1. Vector similarity functions
2. Vector compressions
3. Vector similarity search
4. Maximum inner product search (MIPS)
5. Fast neural ranking
6. GPU computing
7. GCWSNet, hashing algorithms
8. Boosted trees, ABC-boost
9. Privacy
10. Security
11. Distributed, adaptive, and federated learning
12. Others: generative AI models, NLP, knowledge graphs, multi-modal, cross-modal, advertising

Multi-Modal and Cross-Modal Retrieval

Embeddings from cross-modal and multi-modal applications are increasing more common in industry (e.g., text-image, text-video). We have accumulated rich experiences with multi-modal/cross-modal training and retrieval, especially for search and advertising industry.

NAACL'22, [Cross-Lingual Cross-Modal Consolidation for Effective Multilingual Video Corpus Moment Retrieval](#)

SIGIR'22, [Cross-Probe BERT for Fast Cross-Modal Search](#)

CIKM'22, [Multi-scale Multi-modal Dictionary BERT For Effective Text-image Retrieval in Multimedia Advertising](#)

CIKM'22, [Texture BERT for Cross-modal Texture Image Retrieval](#)

BIGDATA'22, [Tree-based Text-Vision BERT for Video Search in Baidu Video Advertising](#)

ICTIR'22, [U-BERT for Fast and Scalable Text-Image Retrieval](#)

NAACL'21, [Cross-lingual Cross-modal Pretraining for Multimodal Retrieval](#)

EMNLP'21, [Inflate and Shrink: Enriching and Reducing Interactions for Fast Text-Image Retrieval](#)

SIGIR'21, [Heterogeneous Attention Network for Effective and Efficient Cross-modal Retrieval](#)

CIKM'21, [Assorted Attention Network for Cross-Lingual Language-to-Vision Retrieval](#)

CIKM'21, [Multi-modal Dictionary BERT for Cross-modal Video Search in Baidu Advertising](#)

CIKM'21, [MixBERT for Multi-modal Matching in Image Advertising](#)

KDD'20, [Combo-Attention Network for Baidu Video Advertising](#)

Our Works on Generative AI Models

- Likelihood-Based Generative Radiance Field with Latent Space Energy-Based Model for 3D-Aware Disentangled Image Representation, *AISTATS* 2023. [pdf](#)
- CoopInit: Initializing Generative Adversarial Networks via Cooperative Learning, *AAAI* 2023. [pdf](#)
- A Tale of Two Latent Flows: Learning Latent Space Normalizing Flow with Short-run Langevin Flow for Approximate Inference, *AAAI* 2023. [pdf](#)
- Learning Latent Structural Relations with Message Passing Prior, *WACV* 2023. [pdf](#)
- A Tale of Two Flows: Cooperative Learning of Langevin Flow and Normalizing Flow Toward Energy-Based Model, *ICLR* 2022. [pdf](#)
- Degenerate Swin to Win: Plain Window-based Transformer without Sophisticated Operations, *Preprint* 2022. [pdf](#)
- Flow-based Perturbation for Cause-effect Inference, *CIKM* 2022. [pdf](#)
- Variational Flow Graphical Model, *KDD* 2022. [pdf](#)
- Causal Effect Prediction with Flow-based Inference, *ICDM* 2022. [pdf](#)
- Learning Energy-Based Model with Variational Auto-Encoder as Amortized Sampler, *AAAI* 2021. [pdf](#)
- Patchwise Generative ConvNet: Training Energy-Based Models from a Single Natural Image for Internal Learning, *CVPR* 2021. [pdf](#)
- Learning Deep Latent Variable Models by Short-Run MCMC Inference with Optimal Transport Correction, *CVPR* 2021. [pdf](#)
- Learning Energy-Based Generative Models via Coarse-to-Fine Expanding and Sampling, *ICLR* 2021. [pdf](#)
- Learning Generative Vision Transformer with Energy-Based Latent Space for Saliency Prediction, *NeurIPS* 2021. [pdf](#)
- Causal Discovery with Flow-based Conditional Density Estimation, *ICDM* 2021. [pdf](#)
- Estimate the Implicit Likelihoods of GANs with Application to Anomaly Detection, *WWW* 2020. [pdf](#)
- Meta-CoGAN: A Meta Cooperative Training Paradigm for Improving Adversarial Text Generation, *AAAI* 2020. [pdf](#)
- Multi-Agent Discussion Mechanism for Natural Language Generation, *AAAI* 2019. [pdf](#)
- Graph to Graph: a Topology Aware Approach for Graph Structures Learning and Generation, *AISTATS* 2019. [pdf](#)
- On Random Deep Weight-Tied Autoencoders: Exact Asymptotic Analysis, Phase Transitions, and Implications to Training, *ICLR* 2019. [pdf](#)

Our Works on NLP

- A Semi-Autoregressive Graph Generative Model for Dependency Graph Parsing, *ACL 2023 (Findings)*. [pdf](#)
- Denoising Enhanced Distantly Supervised Ultrafine Entity Typing, *ACL 2023 (Findings)*. [pdf](#)
- Learning to Selectively Learn for Weakly Supervised Paraphrase Generation with Model-based Reinforcement Learning, *NAACL 2022*.
[pdf](#)
- PromptGen: Automatically Generate Prompts using Generative Models, *NAACL 2022 (Findings)*. [pdf](#)
- Cross-Lingual Cross-Modal Consolidation for Effective Multilingual Video Corpus Moment Retrieval, *NAACL 2022 (Findings)*. [pdf](#)
- Cross-Probe BERT for Fast Cross-Modal Search, *SIGIR 2022*. [pdf](#)
- Continual Learning for Natural Language Generations with Transformer Calibration, *CoNLL 2022*. [pdf](#)
- Texture BERT for Cross-modal Texture Image Retrieval, *CIKM 2022*. [pdf](#)
- U-BERT for Fast and Scalable Text-Image Retrieval, *ICTIR 2022*. [pdf](#)
- Cross-lingual Language Model Pretraining for Retrieval, *WWW 2021*. [pdf](#)
- Cross-lingual Cross-modal Pretraining for Multimodal Retrieval, *NAACL 2021*. [pdf](#)
- Cross-Lingual Unsupervised Sentiment Classification with Multi-View Transfer Learning, *ACL 2020*. [pdf](#)
- Inflate and Shrink: Enriching and Reducing Interactions for Fast Text-Image Retrieval, *EMNLP 2021*. [pdf](#)
- A Deep Decomposable Model for Disentangling Syntax and Semantics in Sentence Representation, *EMNLP 2021 (Findings)*. [pdf](#)
- Hierarchical Multi-Task Word Embedding Learning for Synonym Prediction, *KDD 2019*. [pdf](#)
- Coreference Aware Representation Learning for Neural Named Entity Recognition, *IJCAI 2019*. [pdf](#)
- End-to-end Deep Reinforcement Learning Based Coreference Resolution, *ACL 2019*. [pdf](#)
- Reinforced Product Metadata Selection for Helpfulness Assessment of Customer Reviews, *EMNLP 2019*. [pdf](#)

Our Works on Knowledge Graph (KG)

- OIE@OIA: an Adaptable and Efficient Open Information Extraction Framework, *ACL* 2022. [pdf](#)
- SpaceE: Knowledge Graph Embedding by Relational Linear Transformation in the Entity Space, *HT* 2022. [pdf](#)
- Explainable Concept Graph Completion by Bridging Open-Domain Relations and Concepts, *SDM* 2022. [pdf](#)
- MQuadE: a Unified Model for Knowledge Fact Embedding, *WWW* 2021. [pdf](#)
- ReadsRE: Retrieval-Augmented Distantly Supervised Relation Extraction, *SIGIR* 2021. [pdf](#)
- Extracting Knowledge from Web Text with Monte Carlo Tree Search, *WWW* 2020. [pdf](#)
- A Predicate-Function-Argument Annotation of Natural Language for Open-Domain Information eXpression, *EMNLP* 2020. [pdf](#)
- Learning Interpretable Relationships between Entities, Relations and Concepts via Bayesian Structure Learning on Open Domain Facts, *ACL* 2020. [pdf](#)
- Integration of Knowledge Graph Embedding into Topic Modeling with Hierarchical Dirichlet Process, *NAACL* 2019. [pdf](#)
- Knowledge Graph Embedding Based Question Answering, *WSDM* 2019. [pdf](#)
- Logician and Orator: Learning from the Duality between Language and Knowledge in Open Domain, *EMNLP* 2018. [pdf](#)
- Logician: A Unified End-to-End Neural Approach for Open-Domain Information Extraction, *WSDM* 2018. [pdf](#)

Our Works on Ads, Search, Recommendation

- Media report: A Look at Baidu's Industrial-Scale GPU Training Architecture. [link](#)
- EGM: Enhanced Graph-based Model for Large-scale Video Advertisement Search, *KDD 2022*. [pdf](#)
- Multi-scale Multi-modal Dictionary BERT For Effective Text-image Retrieval in Multimedia Advertising, *CIKM 2022*. [pdf](#)
- Communication-Efficient TeraByte-Scale Model Training Framework for Online Advertising, *BIGDATA 2022*. [pdf](#)
- Boost CTR Prediction for New Advertisements via Modeling Visual Content, *BIGDATA 2022*. [pdf](#)
- FeatureBox: Feature Engineering on GPUs for Massive-Scale Ads Systems, *BIGDATA 2022*. [pdf](#)
- Decomposing User-APP Graph into Subgraphs for Effective APP and User Embedding Learning. [pdf](#)
- Tree-based Text-Vision BERT for Video Search in Baidu Video Advertising. [pdf](#)
- Agile and Accurate CTR Prediction Model Training for Massive-Scale Online Advertising Systems, *SIGMOD 2021*. [pdf](#)
- GemNN: Gating-Enhanced Multi-Task Neural Networks with Feature Interaction Learning for CTR Prediction, *SIGIR 2021*. [pdf](#)
- Heterogeneous Attention Network for Effective and Efficient Cross-modal Retrieval, *SIGIR 2021*. [pdf](#)
- TIRA in Baidu Image Advertising, *ICDE 2021*. [pdf](#)
- Efficient Learning to Learn a Robust CTR Model for Web-scale Online Sponsored Search Advertising, *CIKM 2021*. [pdf](#)
- Multi-modal Dictionary BERT for Cross-modal Video Search in Baidu Advertising, *CIKM 2021*. [pdf](#)
- MixBERT for Multi-modal Matching in Image Advertising, *CIKM 2021*. [pdf](#)
- Assorted Attention Network for Cross-Lingual Language-to-Vision Retrieval, *CIKM 2021*. [pdf](#)
- Combo-Attention Network for Baidu Video Advertising, *KDD 2020*. [pdf](#)
- Video Recommendation with Multi-gate Mixture of Experts Soft Actor Critic, *SIGIR 2020*. [pdf](#)
- Distributed Hierarchical GPU Parameter Server for Massive Scale Deep Learning Ads Systems, *MLSys 2020*. [pdf](#)
- Sample Optimization For Display Advertising, *CIKM 2020*. [pdf](#)
- AIBox: CTR Prediction Model Training on a Single Node, *CIKM 2019*. [pdf](#)
- MOBIUS: Towards the Next Generation of Query-Ad Matching in Baidu's Sponsored Search, *KDD 2019*. [pdf](#)

Example projects in collaboration with ads teams

“Phoenix Nest” is the legacy name of Baidu (search) Ads

EGM: Enhanced Graph-based Model for Large-scale Video Advertisement Search

Tan Yu¹, Jie Liu², Yi Yang², Yi Li², Hongliang Fei¹, and Ping Li¹

1. Cognitive Computing Lab, Baidu Research

2. Baidu Search Ads (Phoenix Nest), Baidu Inc.

10900 NE 8th St. Bellevue Washington, 98004, USA

No. 10 Xibeiwang East Road, Beijing, 100193, China

{tanyu01,liujie34,yangyi15,liyi01,hongliangfei,iping11}@baidu.com

KDD 2022

Heterogeneous Attention Network for Effective and Efficient Cross-modal Retrieval

Tan Yu¹, Yi Yang², Yi Li², Lin Liu², Hongliang Fei¹, Ping Li¹

1. Cognitive Computing Lab, Baidu Research

2. Baidu Search Ads (Phoenix Nest), Baidu Inc.

10900 NE 8th St. Bellevue, Washington 98004, USA

No. 10 Xibeiwang East Road, Beijing 10193, China

{tanyu01, yangyi15, liyi01, liulin03, hongliangfei, liping11}@baidu.com

SIGIR 2021

Assorted Attention Network for Cross-Lingual Language-to-Vision Retrieval

Tan Yu¹, Yi Yang², Hongliang Fei¹, Yi Li², Xiaodong Chen², Ping Li¹

1. Cognitive Computing Lab, Baidu Research

2. Baidu Search Ads (Phoenix Nest), Baidu Inc.

10900 NE 8th St. Bellevue, Washington 98004, USA

No. 10 Xibeiwang East Road, Beijing 10193, China

{tanyu01, yangyi15, hongliangfei, liyi01, chenxiaodong, liping11}@baidu.com

CIKM 2021

Agile and Accurate CTR Prediction Model Training for Massive-Scale Online Advertising Systems

Zhiqiang Xu¹, Dong Li², Weijie Zhao¹, Xing Shen², Tianbo Huang², Xiaoyun Li¹, Ping Li¹

1 Cognitive Computing Lab, Baidu Research

2 Baidu Search Ads (Phoenix Nest)

No. 10 Xibeiwang East Road, Beijing 100193, China

10900 NE 8th St. Bellevue, Washington 98004, USA

{xuzhiqiang04, lidong06, weijiezha, shenxing01, huangtianbo, v_liaoyun02, liping11}@baidu.com

SIGMOD 2021

GemNN: Gating-Enhanced Multi-Task Neural Networks with Feature Interaction Learning for CTR Prediction

Hongliang Fei¹, Jingyuan Zhang¹, Xingxuan Zhou², Junhao Zhao², Xinyang Qi², Ping Li¹

1. Cognitive Computing Lab, Baidu Research

2. Baidu Search Ads (Phoenix Nest), Baidu Inc.

10900 NE 8th St. Bellevue, Washington 98004, USA

No. 10 Xibeiwang East Road, Beijing 100193, China

{hongliangfei, zhangjingyuan03, zhouxingxuan, zhaojunhao, qixinyang, liping11}@baidu.com

SIGIR 2021

MixBERT for Multi-modal Matching in Image Advertising

Tan Yu¹, Xiaokang Li², Jianwen Xie¹, Ruiyang Yin², Qing Xu², Ping Li¹

1. Cognitive Computing Lab, Baidu Research

2. Baidu Search Ads (Phoenix Nest), Baidu Inc.

10900 NE 8th St. Bellevue, Washington 98004, USA

No. 10 Xibeiwang East Road, Beijing 10193, China

{tanyu01, lixiaokang04, jianwenxie, yinruiyang, xuqing06, liping11}@baidu.com

CIKM 2021

Example projects in collaboration with ads teams

Multi-modal Dictionary BERT for Cross-modal Video Search in Baidu Advertising

Tan Yu¹, Yi Yang², Yi Li², Lin Liu², Mingming Sun¹, and Ping Li¹

1. Cognitive Computing Lab, Baidu Research
2. Baidu Search Ads (Phoenix Nest), Baidu Inc.
10900 NE 8th St. Bellevue Washington, 98004, USA
No. 10 Xibeiwang East Road, Beijing, 100193, China
{tanyu01,yangyi15,liy01,liulin03,sunmingming01,liping11}@baidu.com

CIKM 2021

Efficient Learning to Learn a Robust CTR Model for Web-scale Online Sponsored Search Advertising

Xin Wang¹, Peng Yang¹, Shaopeng Chen², Lin Liu², Lian Zhao², Jiacheng Guo², Mingming Sun¹,
Ping Li¹

¹Cognitive Computing Lab, Baidu Research
²Baidu Sponsored Search (Phoenix Nest)
No.10 Xibeiwang East Road, Beijing 100193, China
10900 NE 8th St. Bellevue, Washington 98004, USA
{wangxin60, pengyang01, chenshaopeng, liulin03, zhaolian, guojacheng, sunmingming01, liping11}@baidu.com

CIKM 2021

Combo-Attention Network for Baidu Video Advertising

Tan Yu¹, Yi Yang², Yi Li², Xiaodong Chen², Mingming Sun¹, Ping Li¹

1. Cognitive Computing Lab, Baidu Research
2. Baidu Search Ads (Phoenix Nest), Baidu Inc.
10900 NE 8th St. Bellevue WA, 98004, USA
No. 10 Xibeiwang East Road, Beijing, 100085, China
{tanyu01,yangyi15,liy01,chenxiaodong,sunmingming01,liping11}@baidu.com

KDD 2020

TIRA in Baidu Image Advertising

Tan Yu*, Xuemeng Yang†, Yan Jiang†, Hongfang Zhang†, Weijie Zhao*, Ping Li*

*Cognitive Computing Lab, Baidu Research

†Baidu Search Ads (Phoenix Nest), Baidu Inc

10900 NE 8th St. Bellevue, Washington 98004, USA

701 Na Xian Road, Pudong, Shanghai 201210, China

ICDE 2021

Multi-Task and Multi-Scene Unified Ranking Model for Online Advertising

Shulong Tan¹, Meifang Li², Weijie Zhao¹, Yandan Zheng², Xin Pei², Ping Li¹

¹Cognitive Computing Lab, Baidu Research

²Baidu Feed Ads (Phoenix Nest), Baidu Inc.

10900 NE 8th St. Bellevue, Washington 98004, USA

No. 10 Xibeiwang East Road, Beijing 10193, China

{shulongtan, limeifang, weijiezhao, zhengyandan, peixin, liping11}@baidu.com

Video Recommendation with Multi-gate Mixture of Experts Soft Actor Critic

SIGIR 2020

Dingcheng Li¹, Xu Li¹, Jun Wang², Ping Li¹

1. Cognitive Computing Lab, Baidu Research

2. Haokan Videos, Baidu Inc.

10900 NE 8th St. Bellevue, WA, 98004, USA

Sample Optimization For Display Advertising

Hongliang Fei¹, Shulong Tan¹, Pengju Guo², Wenbo Zhang², Hongfang Zhang², Ping Li¹

1. Cognitive Computing Lab, Baidu Research

2. Baidu Search Ads (Phoenix Nest), Baidu Inc.

1195 Bordeaux Dr, Sunnyvale, CA 94089, USA

701 Na Xian Road, Pudong, Shanghai 201210, China

10900 NE 8th St. Bellevue, WA 98004, USA

{hongliangfei, shulongtan, guopengju, zhangwenbo03, zhanghongfang, liping11}@baidu.com

CIKM 2020

Examples of non-ads product projects

FastInput: Improving Input Efficiency on Mobile Devices

Jingyuan Zhang, Xin Wang, Yue Feng, Mingming Sun and Ping Li

Cognitive Computing Lab

Baidu Research

10900 NE 8th St. Bellevue, Washington 98004, USA

No.10 Xibeiwang East Road, Beijing 100193, China

{zhangjingyuan03, wangxin60, fengyue04, sunmingming01, liping11}@baidu.com

CIKM 2018, Chinese Language Input Method Editor

Improved Touch-screen Inputting Using Sequence-level Prediction Generation

Deployed boosted tree models on cell phones

Xin Wang, Xu Li, Jinxing Yu, Mingming Sun, Ping Li

Cognitive Computing Lab

Baidu Research

No.10 Xibeiwang East Road, Beijing, China

10900 NE 8th St. Bellevue, WA 98004, USA

{wangxin60, lixu13, yujinxing, sunmingming01, liping11}@baidu.com

ABSTRACT

Recent years have witnessed the continuing growth of people's dependence on touchscreen devices. As a result, input speed with the onscreen keyboard has become crucial to communication efficiency and user experience. In this work, we formally discuss the general problem of input expectation prediction with a touch-screen input method editor (IME). Taken input efficiency as the optimization target, we proposed a neural end-to-end candidates generation solution to handle automatic correction, reordering, insertion, deletion as well as completion. Evaluation metrics are also discussed base on real use scenarios. For a more thorough comparison, we also provide a statistical strategy for mapping touch coordinate sequences to text input candidates. The proposed model and baselines are

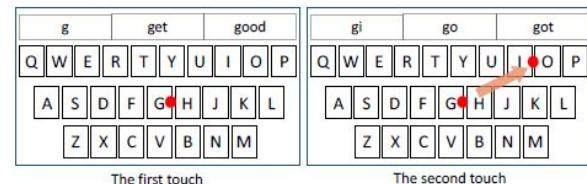


Figure 1: An illustration of the input process and the corresponding candidate lists. The first touch falls into the area of 'g', so the candidate list shows high-probability words beginning with 'g'. Then the second touch falls on the board of 'i' which is close to 'o', so the candidate list contains both possible strings 'gi' and 'go'.

WWW 2020

International Langue Input Method Editor

Summary

- Many data types in machine learning and AI can be viewed as “vectors”.
- Vectorized data computing (VDC) is crucial for machine learning and is also much beyond machine learning. It may grow into its own discipline in the near future.
- Vector databases can be viewed as one component in vectorized data computing.
- In most applications, results from **vector databases** (such as similarity search) are quite crude and can serve as the initial screening step (e.g., ads candidate retrieval). AI and machine learning models are necessary for accurate predictions.
- Keys in big models: 1) accuracy 2) training/serving efficiency 3) distributed training. For example, training **trillion-parameter** models for high-accuracy recommender systems.. Many novel algorithms and infrastructure systems are presented.
- **Privacy** and AI model security have become increasingly critical in AI.

Updated Version Available

<https://pltrees.github.io/publication/VecDataComp.pdf>