Distributed Distance Learning Algorithms and Applications

Yuxin Su

Ph.D. Oral Defense

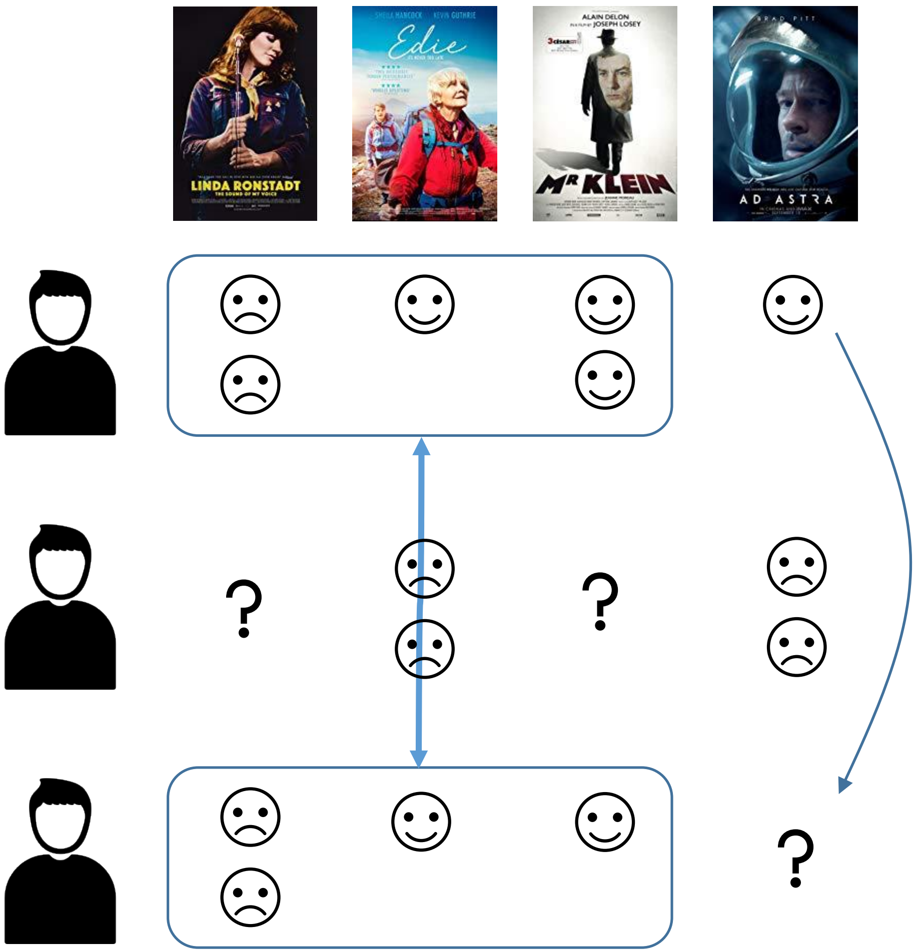
Supervisors: Prof. Michael Lyu and Prof. Irwin King

8/28/2019

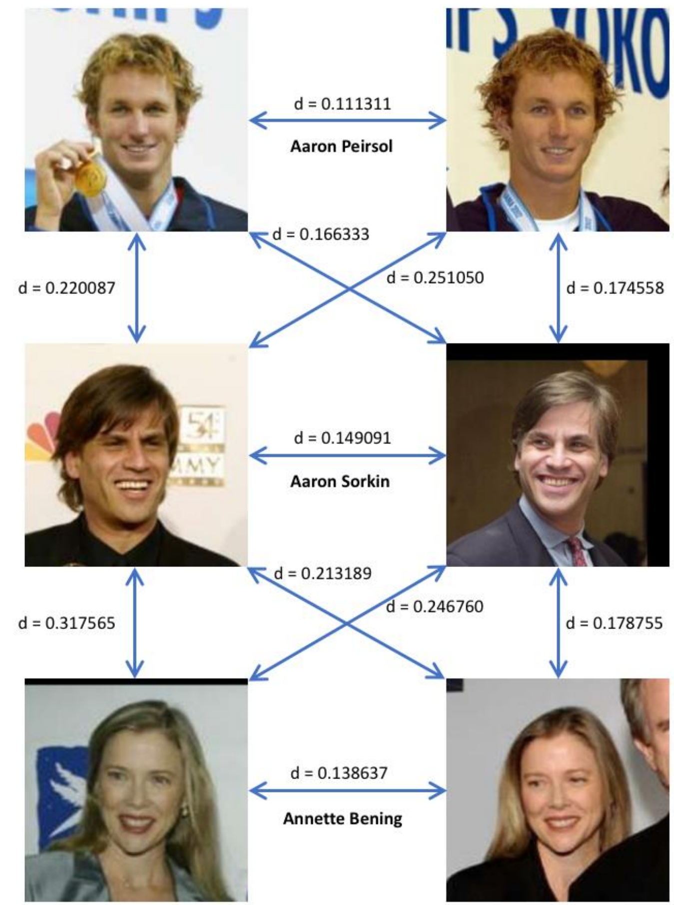
香港中文大學
The Chinese University of Hong Kong

Similarity is the Fundamental

Recommender System



Face Verification



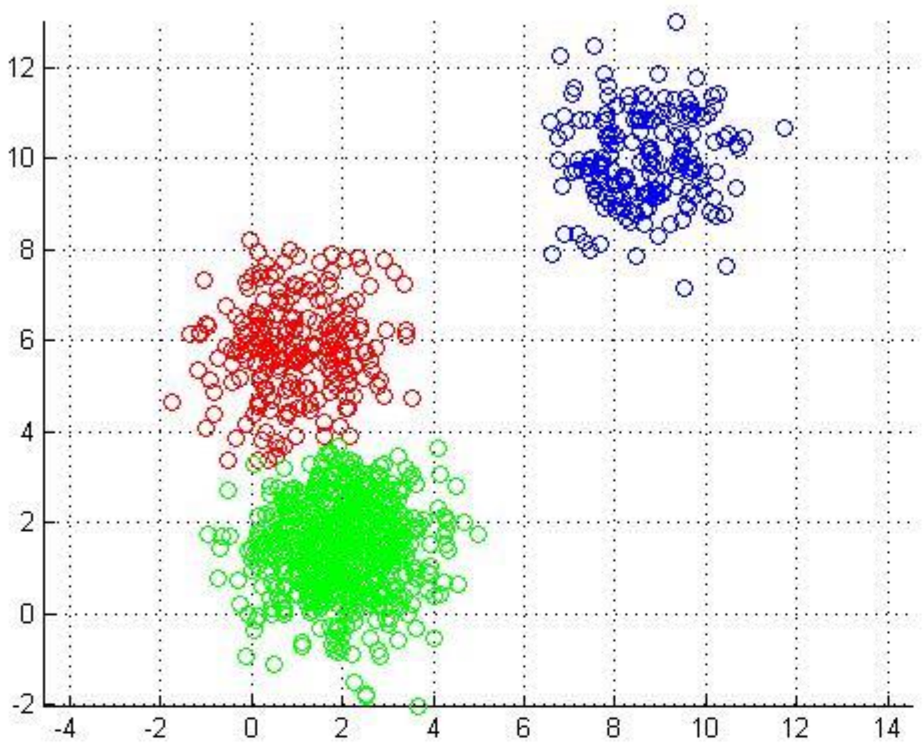
https://github.com/JifuZhao/face-verification

Popular Distance Functions

Name	Expression
Euclidean	$f(x,y) = \sqrt{(x-y)^\top (x-y)}$
Cityblock	$f(x,y) = \sum_i \lvert x_i - y_j \rvert$
Chebyshev	$f(x,y) = \max\{\lvert x_i - y_i \rvert\}$
Minkowski	$f(x,y) = \left(\sum_i \lvert x_i - y_i \rvert^p\right)^{\frac{1}{p}}$
Jaccard	$f(x,y) = 1 - \frac{\sum \min\{x_i, y_i\}}{\sum \max\{x_i, y_i\}}$
Cosine	$f(x,y) = 1 - \frac{x^\top y}{\lvert x \rvert \lvert y \rvert}$
KL Divergence	$f(x,y) = \sum_i x_i \log \frac{x_i}{y_i}$
Mahalanobis	$f(x,y) = \sqrt{(x-y)^\top M (x-y)}$

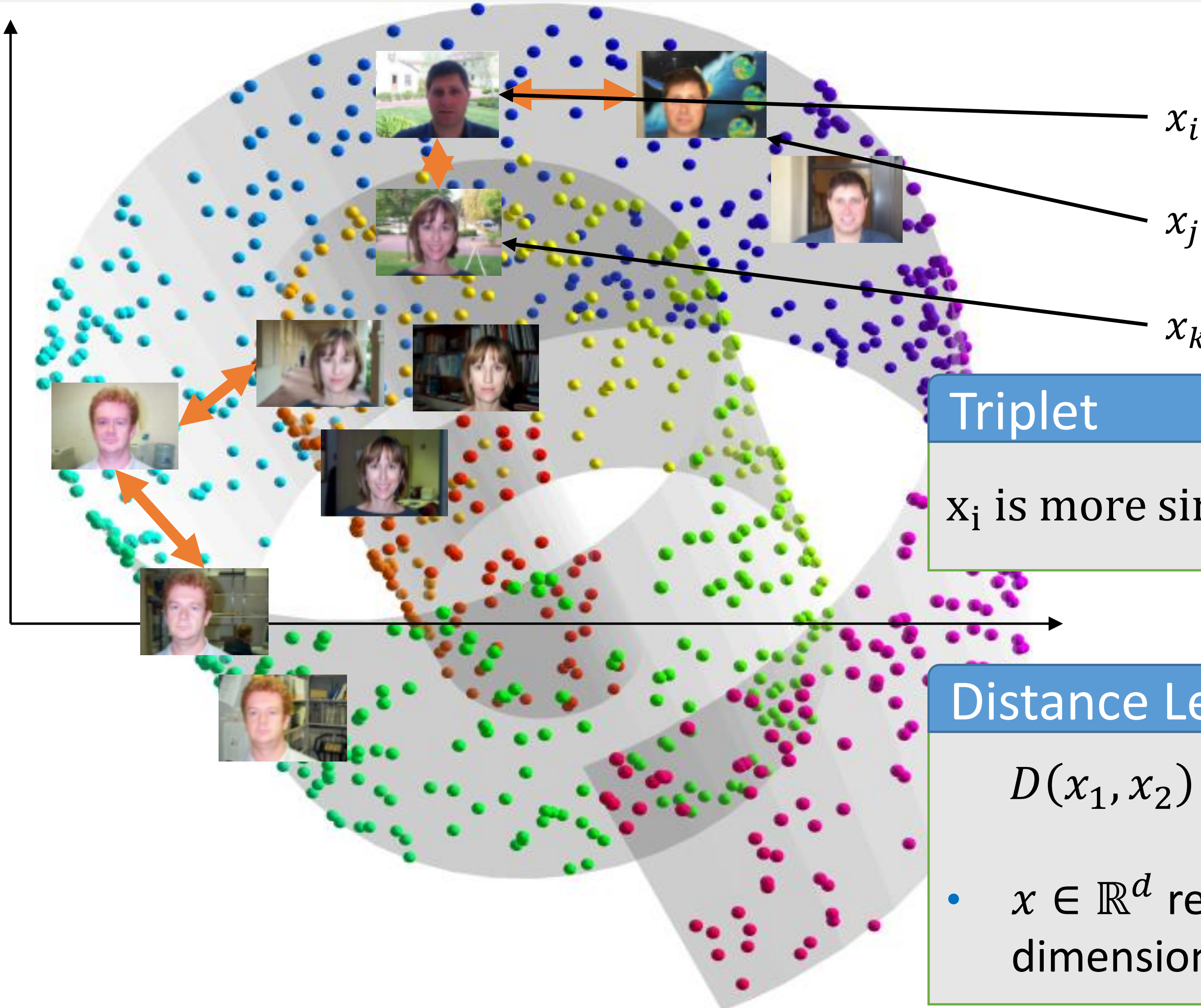
Euclidean Distance

$$f(\boldsymbol{x}_i, \boldsymbol{x}_j) = \sqrt{(\boldsymbol{x}_i - \boldsymbol{x}_j)^{\top}(\boldsymbol{x}_i - \boldsymbol{x}_j)}$$



not suitable for high-dimensional data

Similarity between Images



Triplet

x_i is more similar to x_j than to x_k

Distance Learning

$$D(x_1, x_2) = \|f(x_1) - f(x_2)\|$$

- $x \in \mathbb{R}^d$ represents high-dimensional data

Bellet, Aurelien, Amaury Habrard, and Marc Sebban. "A Survey on Metric Learning for Feature Vectors and Structured Data." *arXiv: Learning* (2013).

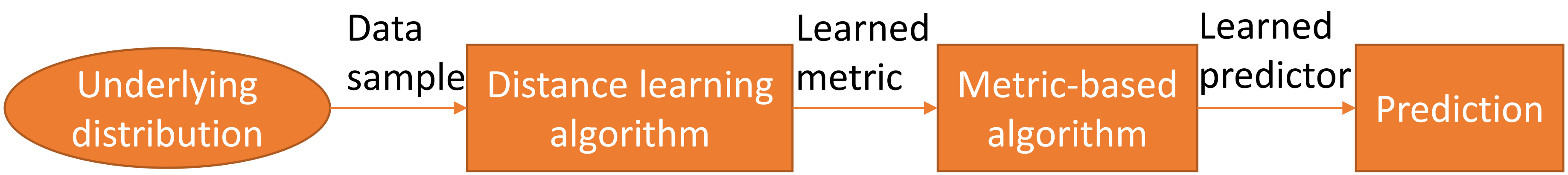
Distance Learning

- Beyond the Euclidean distance:

$$f(\boldsymbol{x}_1, \boldsymbol{x}_2) = \|\phi(\boldsymbol{x}_1) - \phi(\boldsymbol{x}_2)\|$$

- e.g. Mahalanobis distance:

$$f_M(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sqrt{(\boldsymbol{x}_1 - \boldsymbol{x}_2)^\top M(\boldsymbol{x}_1 - \boldsymbol{x}_2)}$$



More accurate result

Remain semantic meaning in original dimension

Supervision to Learn a Distance Function

- Positive / negative pairs:

$$\mathcal{S} = \{(x_i, x_j)\ :\ x_i \text{ and } x_j \text{ should be similar}\}$$
$$\mathcal{D} = \{(x_i, x_j)\ :\ x_i \text{ and } x_j \text{ should be dissimilar}\}$$

- For positive/negative pairs, we need pre-assigned threshold:

$$\mathcal{S} = \{(x_i, x_j) : d_M(x_i, x_j) \leq u\}$$
$$\mathcal{D} = \{(x_i, x_j) : d_M(x_i, x_j) \geq l\}$$

- Relative constraints:

$$\mathcal{R} = \{(x_i, x_j, x_k)\ :\ x_i \text{ should be more similar to } x_j \text{ then to } x_k\}$$

Task-oriented Distance Measurement

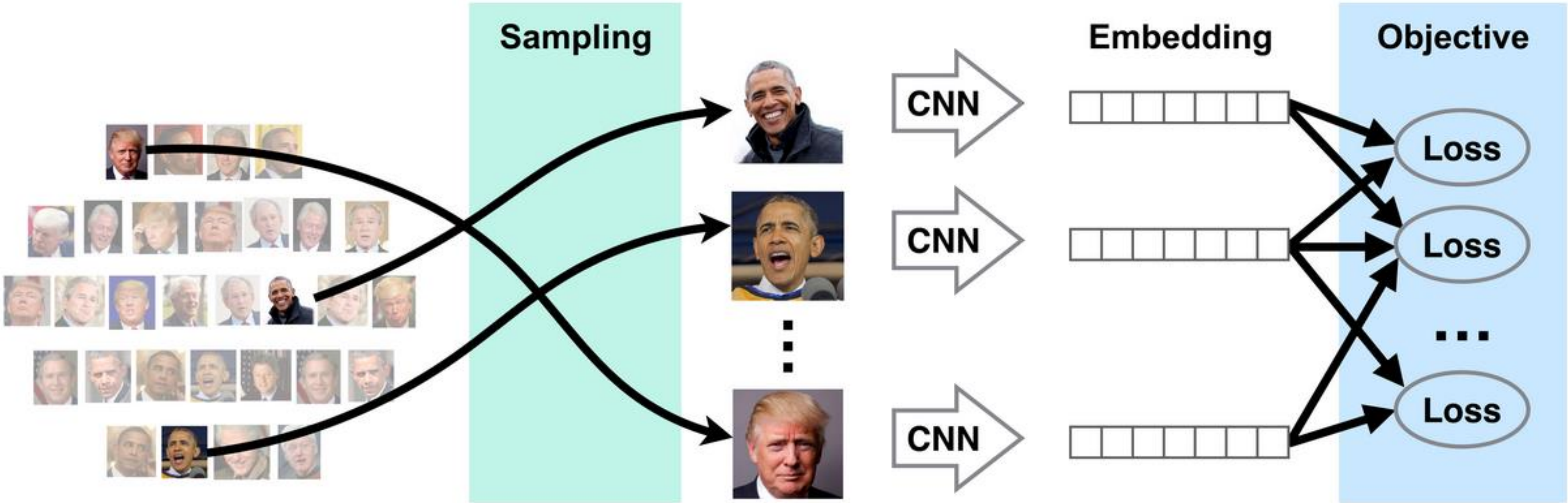
Challenges in Big Data Era

- Handle high-dimensional data

Positive Semidefinite Matrix

$$d_M(x_i, x_j) = (x_i - x_j)^T \times \boxed{M \in \mathbb{R}^{d \times d}} \times (x_i - x_j) \geq 0$$

Challenges: huge number of constraints

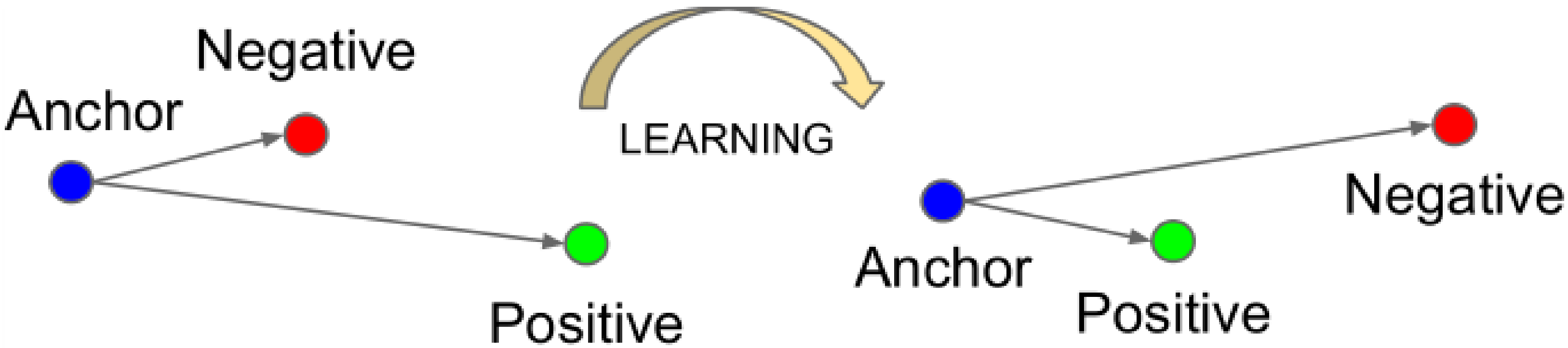


N images → $\mathcal{O}(N^3)$ triplets

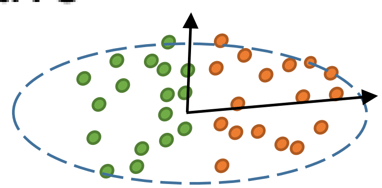
Impossible task for big data set

https://www.cs.utexas.edu/~cywu/projects/sampling_matters

Category of Distance Function

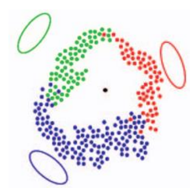


Linear



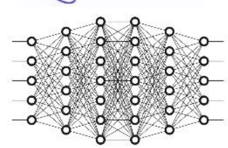
Global distance learning

Non-linear low complexity



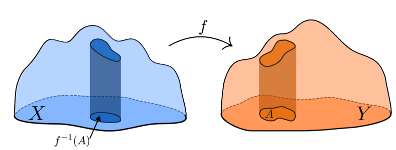
Local distance learning

Non-linear high complexity



Deep distance learning

Probabilistic metric space



Wasserstein distance learning

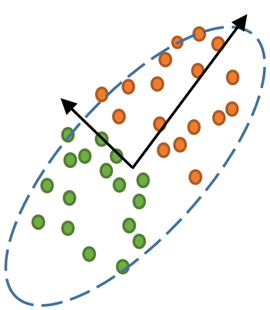
Thesis Contribution in Overall

Employ multiple machines to solve big data problem

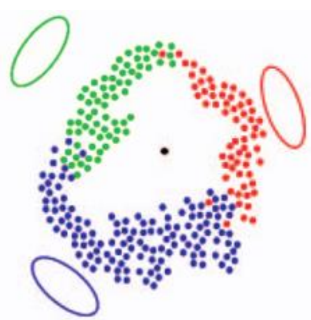
Mathematically sound parallel solution

Communication-efficient Distributed Implementation

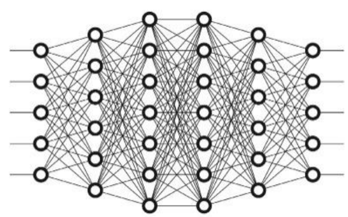
Global



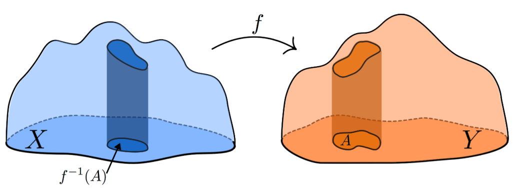
Local



Deep



Probabilistic



Computational Complexity

Thesis Organization

Introduction
(Chapter 1)

Background Review
(Chapter 2)

Global Distance Learning
(Chapter 3) [IJCNN'16]

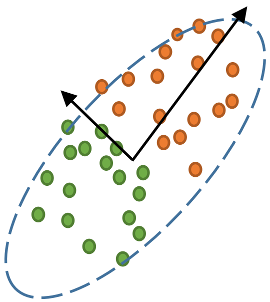
Local Distance Learning
(Chapter 4) [SIGIR'17]

Deep Distance Learning
(Chapter 5) [CIKM'18]

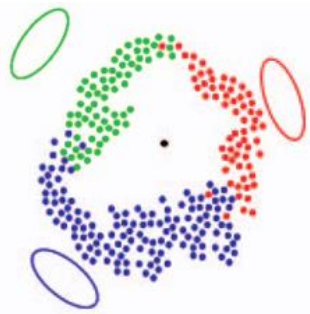
Probabilistic Distance Learning
(Chapter 6) [IJCAI'19]

Conclusion
(Chapter 7)

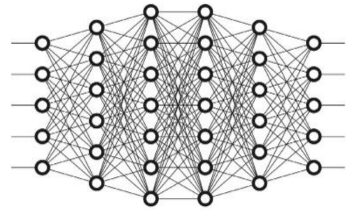
Global



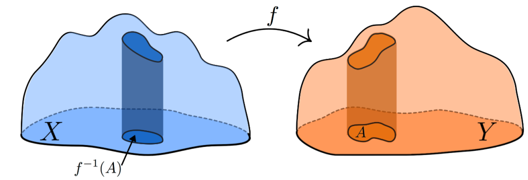
Local



Deep



Probabilistic



Global Distance Learning

Parallel Algorithm

Distributed Implementation

Theoretical Analysis

Experiments

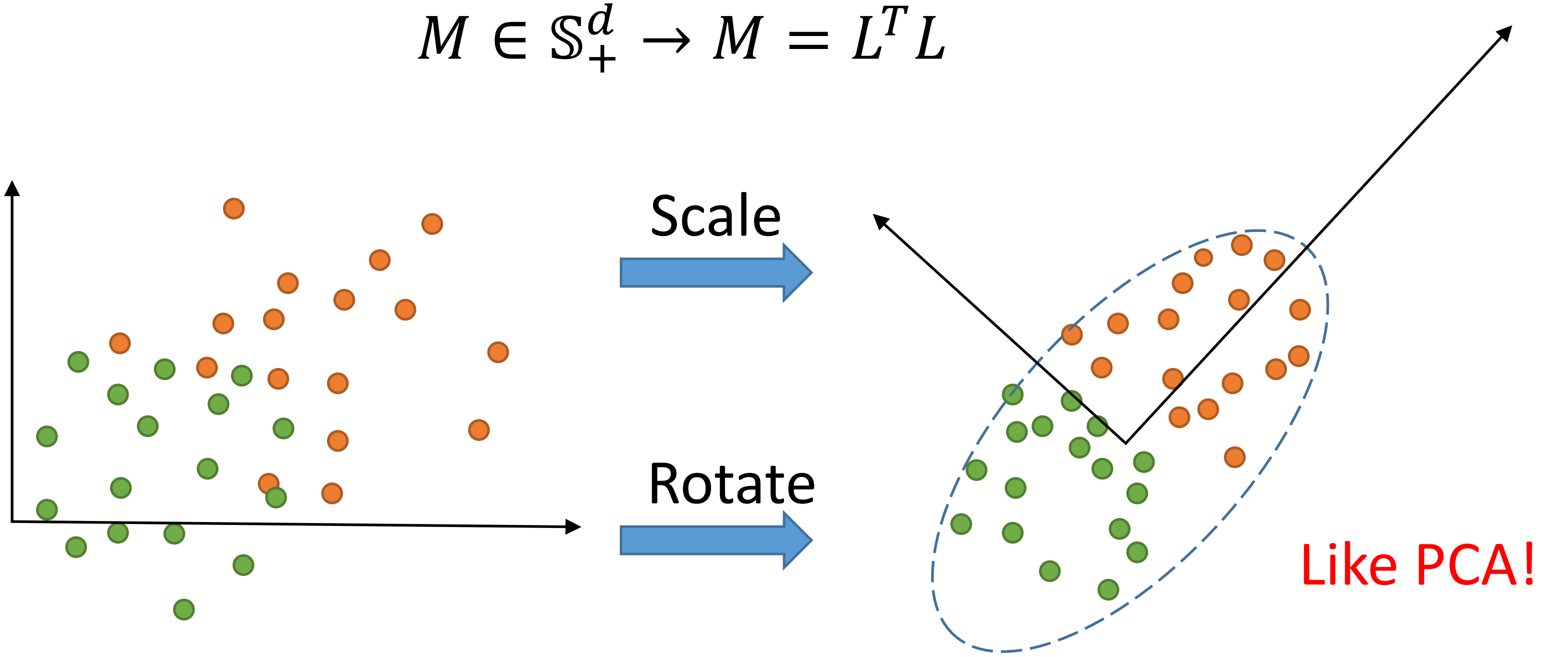
Linear Projection

- Mahalanobis distance:

$$d_M(\boldsymbol{x}_1, \boldsymbol{x}_2) = \sqrt{(\boldsymbol{x}_1 - \boldsymbol{x}_2)^\top M(\boldsymbol{x}_1 - \boldsymbol{x}_2)}$$

- Equivalent to a linear projection on Euclidean metric space

$$M \in \mathbb{S}_+^d \rightarrow M = L^T L$$



Information-Theoretical Metric Learning (ITML)

- Reformulate distance learning problem

$$\min D_\phi(A, A_0)$$
$$\mathrm{tr}(A(x_i - x_j)(x_i - x_j)^T) \le u, \quad (x_i, x_j) \in \mathcal{S}$$
$$\mathrm{tr}(A(x_i - x_j)(x_i - x_j)^T) \ge l, \quad (x_i, x_j) \in \mathcal{D}$$

- The global optimal solution does not rely on eigen-decomposition:

$$A_{t+1} = A_t + \beta A_t(x_i - x_j)(x_i - x_j)^T A_t$$

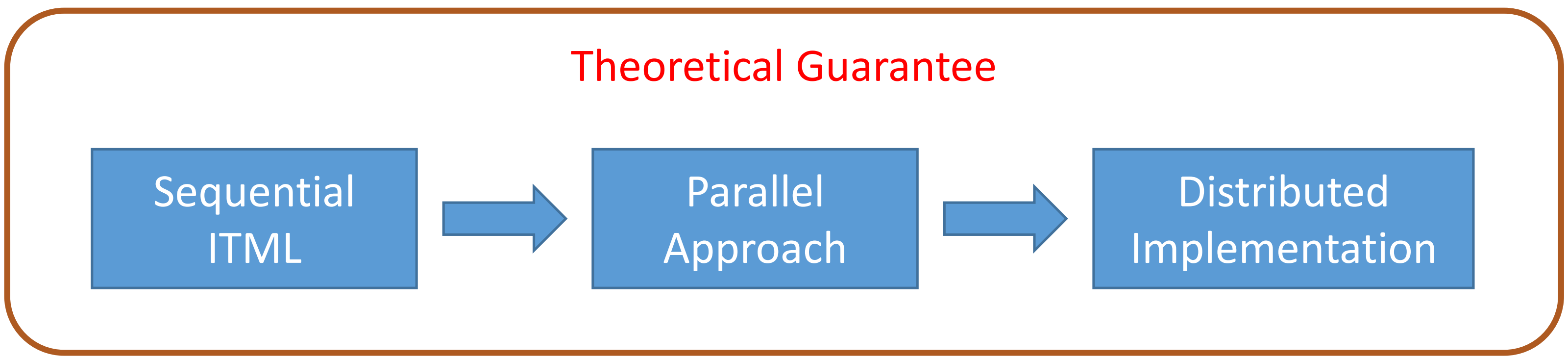
- Mahalanobis distance between data points:

$$p(t) = (x_i - x_j)^T A_t (x_i - x_j)$$

Davis, Jason V., et al. "Information-theoretic metric learning." Proceedings of the 24th international conference on Machine learning. ACM, 2007.

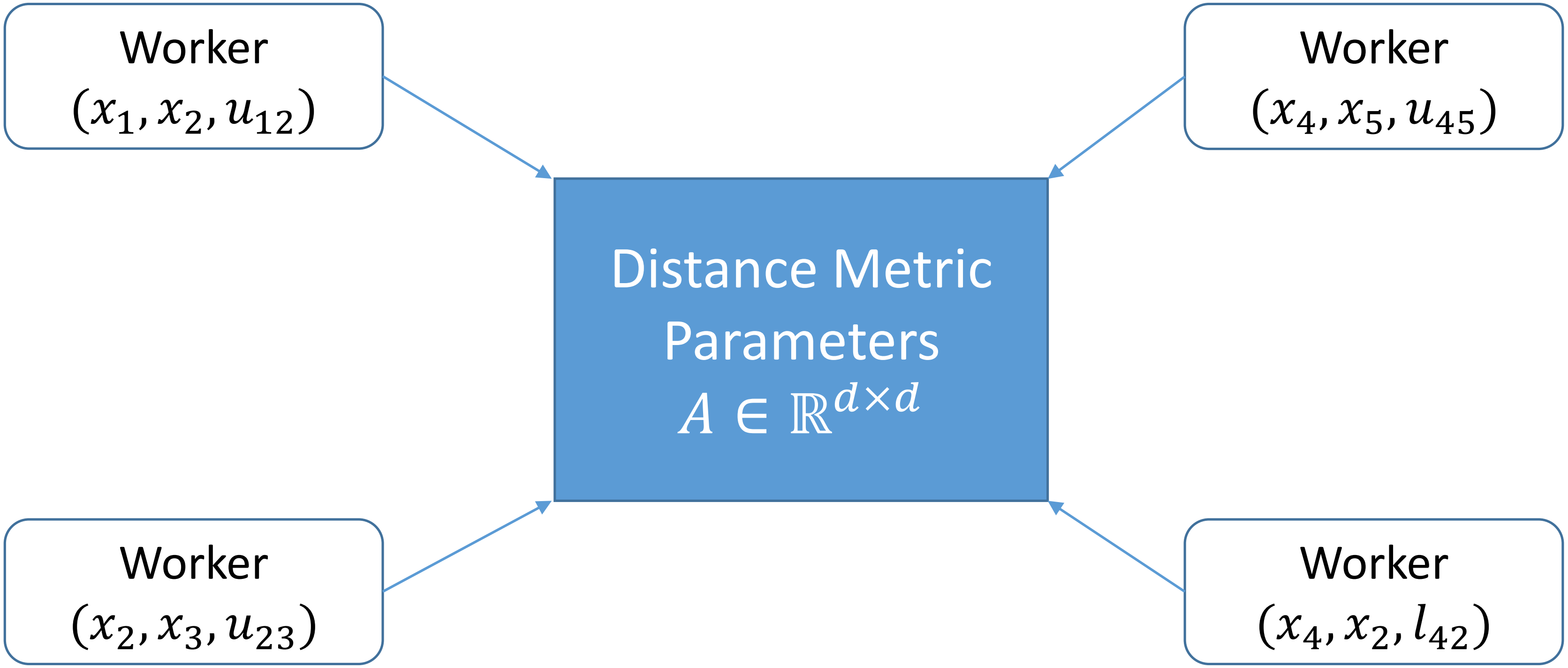
Contributions

- To handle high-dimensional data
  - A parallel Information-Theoretical Metric Learning (ITML) algorithm
  - Theoretical analysis to bound the gap
  - Well-designed approach for a popular distributed platform, Apache Spark



Parallel Computation

- Speed up the algorithm by parallel computation

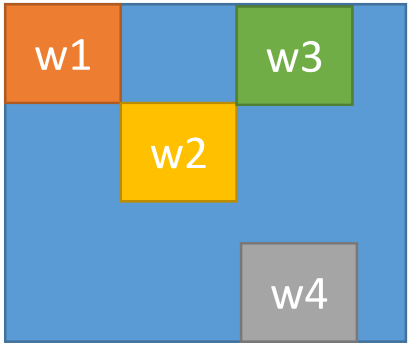


Parallel Updates to PSD Matrix

- Parallel updates will destroy the positive semidefinite property

Positive Semidefinite Matrix

$$d_A(x_i, x_j) = (x_i - x_j)^T \times$$



$$\times (x_i - x_j) \geq 0$$

$$A \in \mathbb{R}^{d \times d}$$

Decomposition of PSD Matrix

- Constraints:

$$\operatorname{rank}(A_i) = 1 \quad \forall i \in [k]$$
$$\operatorname{tr}(A_i) = 1 \quad \forall i \in [k]$$

Positive Semidefinite Matrix

$$= A_1 + A_2 + \cdots + A_k$$

$$A \in \mathbb{R}^{d\times d}$$

Decomposition of Mahalanobis Matrix

- We reformulate Mahalanobis Matrix as

$$A = I + \sum_i \alpha_i z_i z_i^T$$

  - Where $z_i \in \mathbb{R}^d$

- Bregman projection over all constraints:

$$A_{t+1} = I + \sum_{i=1}^{C} \beta_i (t) z_i(t) z_i^T(t)$$

  - C is the number of constraints
  - β is the step size of Bregman projection

Update z_i

- In Bregman projection, $A_t(x_i - x_j) \in \mathbb{R}^d$

$$\begin{aligned} z_k(t+1) &= A_{t+1}(x_i - x_j) \\ &= A_{t+1} c_k \\ &= \left(I + \sum_{i=1}^{C} \beta_i(t) z_i(t) z_i^T(t) \right) c_k \end{aligned}$$

- In the original Bregman projection

$$\beta_k(t) \sim p_k(t)$$

  - where $p_k(t)$ is the Mahalanobis distance between k-th constraint computed with A_t

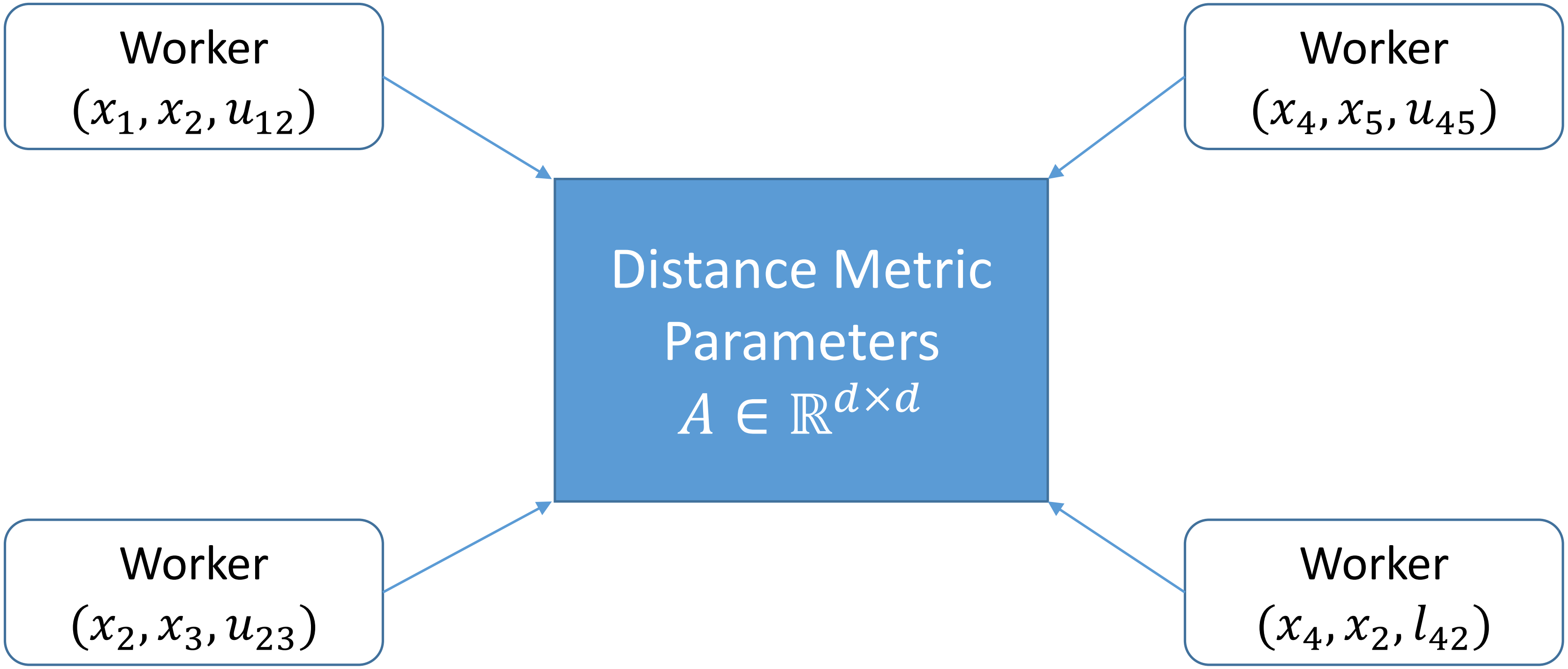
Calculate Mahalanobis distance

$$\begin{aligned} p_k(t) &= c_k^T A_t c_k \\ &= c_k^T \left(I + \sum_{i=1}^{C} \beta_i(t) z_i(t) z_i^T(t) \right) c_k \\ &= c_k^T c_k + \sum_{i=1}^{C} \beta_i(t) c_k^T z_i(t) z_i^T(t) c_k \end{aligned}$$

- The computation procedure is separable and easy to conduct in parallel

Parallel Computation

- For each worker:
  - Compute partial Mahalanobis distance
  - Collect z from other workers
  - Update z and broadcast its newer z to other workers



Algorithm

Input: S : set of similar pairs; D: set of dissimilar pairs; u, l : distance thresholds; γ : slack parameter

Output: A : Mahalanobis matrix

$A = I$, $C = |S| + |D|$
for constraint $(x_p, x_q)_k$, $\quad k \in \{1, 2, \ldots, C\}$ **do**
    $\lambda_k \leftarrow 0$
    $d_k \leftarrow u$ for $(x_p, x_q)_k \in S$ otherwise $d_k \leftarrow l$
    $c_k \leftarrow (x_p - x_q)_k$, $z_k \leftarrow c_k$
end for
while β does not converge **do**
    for all worker $k \in \{1, 2, \ldots, C\}$ **do in parallel**
        $z_k = c_k + \sum_{i=1}^{C} \beta_i z_i z_i^T c_k$
        $p \leftarrow c_k^T z_k$
        if $(x_p, x_q)_k \in S$ **then**

Algorithm (cont'd)

$\alpha \leftarrow \min\left(\lambda_k, \frac{1}{2}\left(\frac{1}{p} - \frac{\gamma}{d_k}\right)\right)$

$\beta \leftarrow \frac{\alpha}{1-\alpha p}$

$d_k \leftarrow \frac{\gamma d_k}{\gamma + \alpha d_k}$

else

$\alpha \leftarrow \min\left(\lambda_k, \frac{1}{2}\left(\frac{\gamma}{d_k} - \frac{1}{p}\right)\right)$

$\beta \leftarrow \frac{-\alpha}{1+\alpha p}$

$d_k \leftarrow \frac{\gamma d_k}{\gamma - \alpha d_k}$

end if

$\lambda_k \leftarrow \lambda_k - \alpha$

$z_k \leftarrow \left(I + \sum_{i=1}^{C} \beta_i z_i z_i^T\right) c_k$

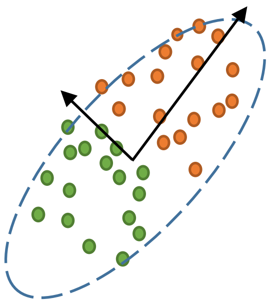
send z_k to other workers.

end for

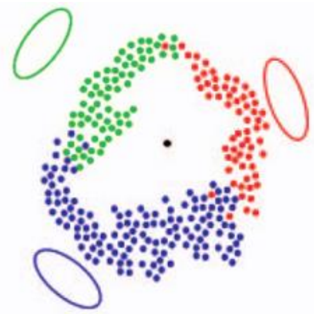
end while

$A = I + \sum_{i=1}^{C} \beta_i z_i z_i^T$

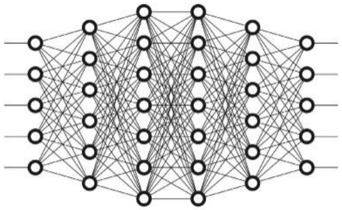
Global



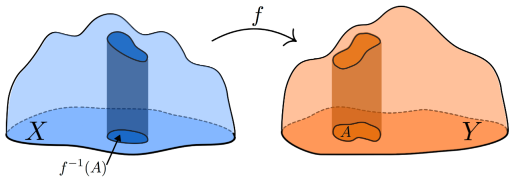
Local



Deep



Probabilistic



Global Distance Learning

Parallel Algorithm

Distributed Implementation

Theoretical Analysis

Experiments

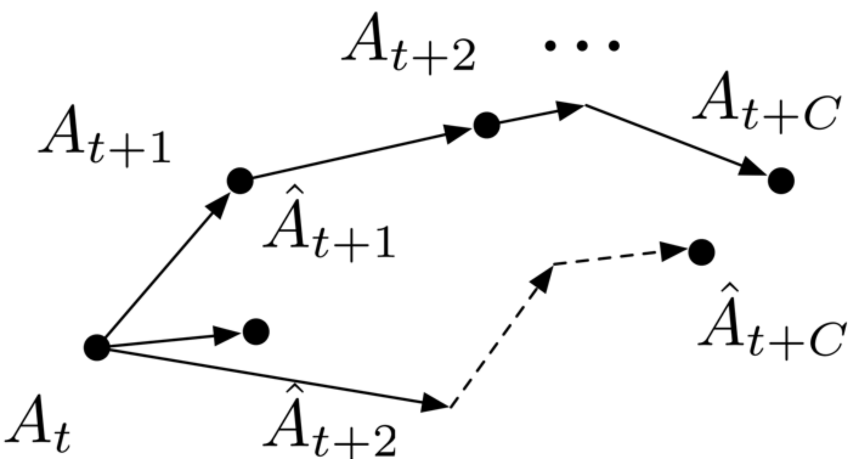
Parallel Update vs Delayed Sequential Update

Sequential Update

$$A_{t+C} = A_t + \sum_{i=1}^{C} \beta_i \, A_{t+i-1} c_i c_i^T A_{t+i-1}^T$$

Parallel Update

$$\hat{A}_{t+C} = A_t + \sum_{i=1}^{C} \beta_i \, A_t c_i c_i^T A_t^T$$



- Parallel update is equivalent to the delayed sequential update

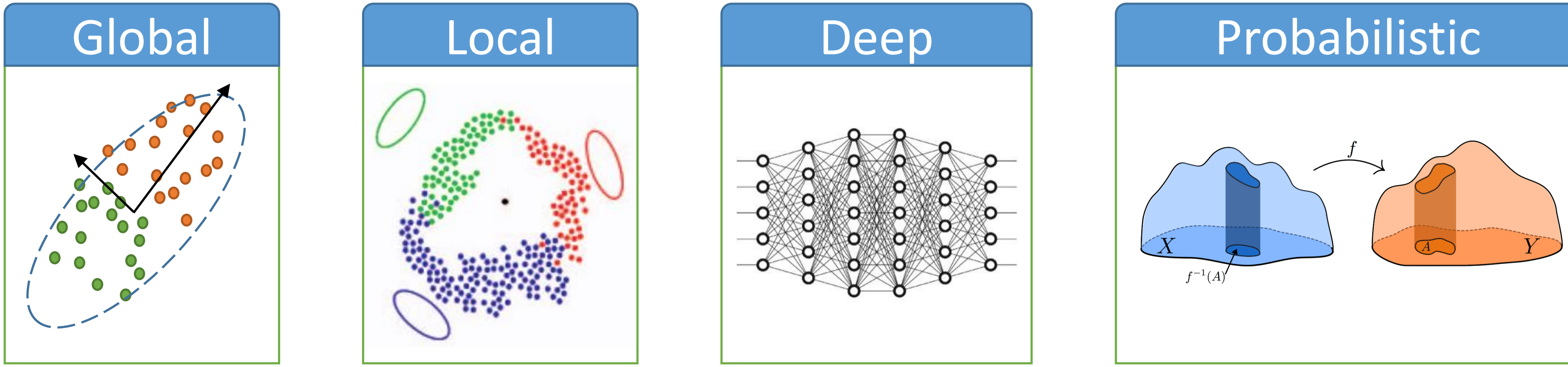
Major Result

Theorem: Delayed update under Bregman divergence

Assume the difference of matrices is measured by the Bregman divergence with respect to LogDet divergence $\phi(X) = -\log\det(X)$. The minimizer of $D_\phi(A_0, A)$ after T iterations is A^*:

$$R[A] \coloneqq \sum_{t=1}^{T} D_\phi(A_t, A^*) \leq \frac{1}{\beta_{\min}} D_\phi(A^*, I) + \frac{1}{2} L\Omega$$

- R is the accumulated loss of learned distance function
- Ω is the length of convergence path.
- L denotes the length of projection path in the original sequential algorithm.



Global Distance Learning

Parallel Algorithm

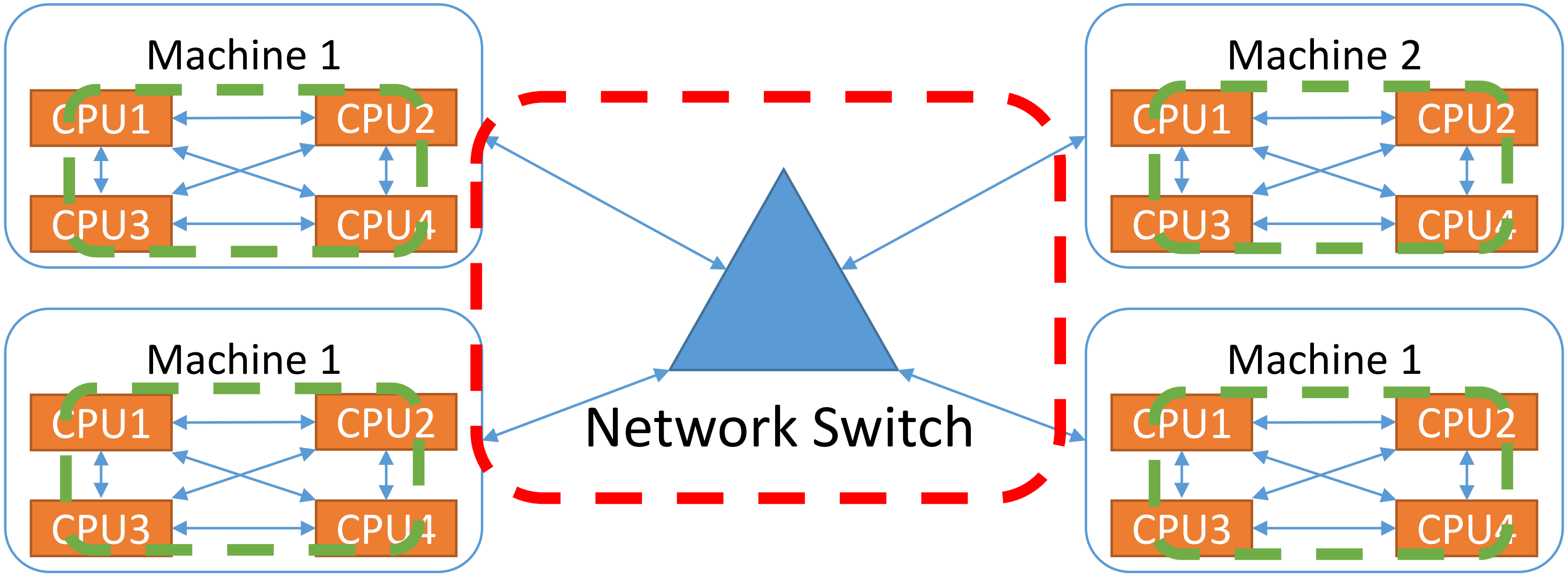
Distributed Implementation

Theoretical Analysis

Experiments

From "Parallel" to "Distributed"

- Parallel computation is not enough because of the limited memory
- One-by-one mapping between logical worker and real machine is not suggested because of
  - Heavy communication between machines
  - Imbalanced workload wastes physical resources

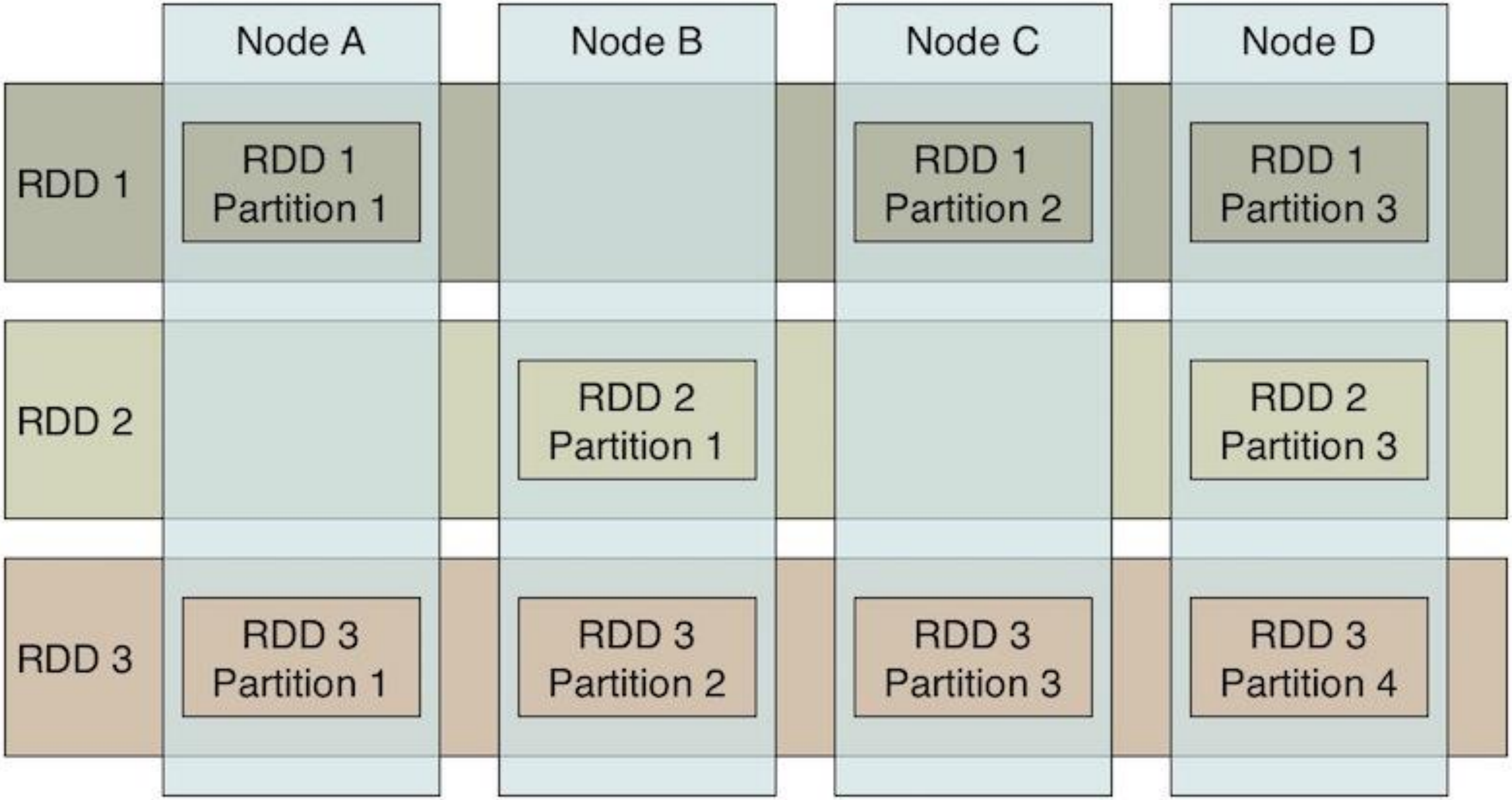


Apache Spark

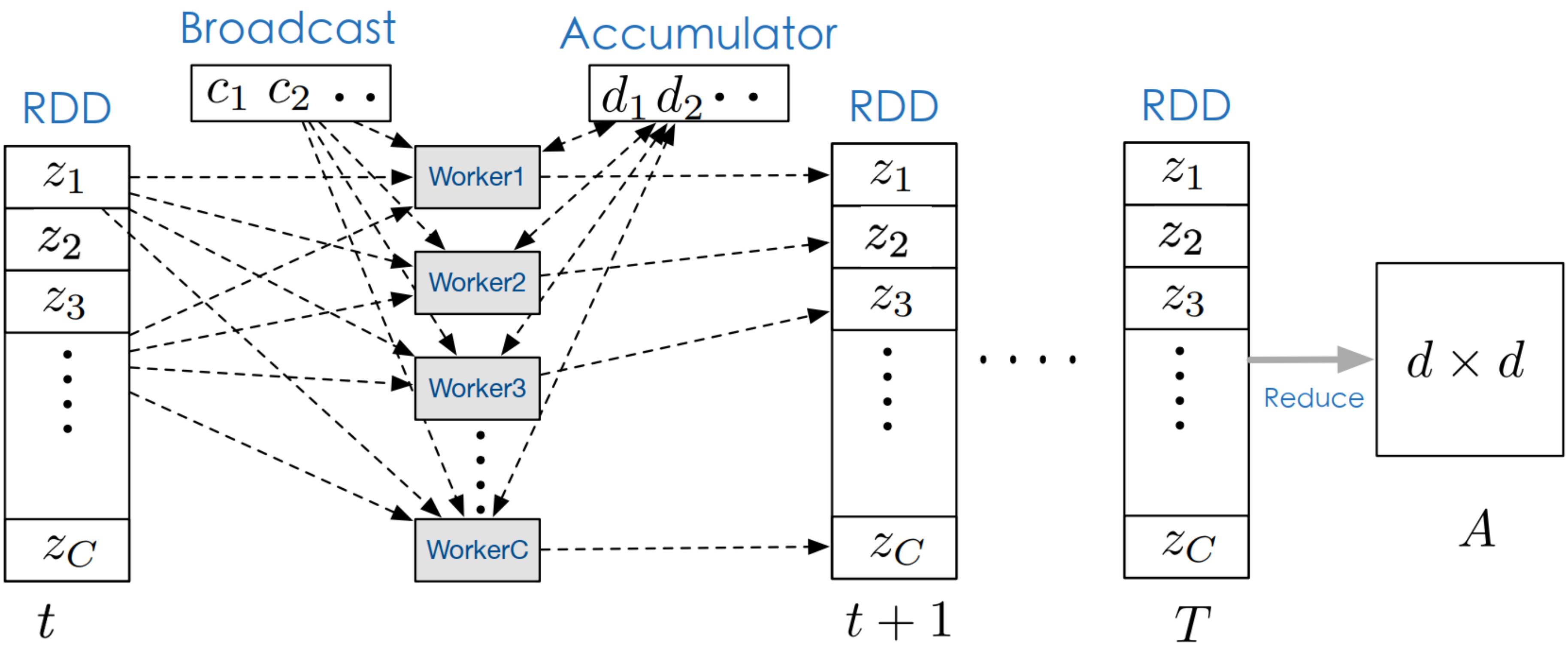
- Apache Spark is the most popular distributed platform for large-scale machine learning task.
- Resilient Distributed Datasets (RDD) in Spark is suitable for our approach.

Spark Lightning-fast cluster computing

RDD Partition

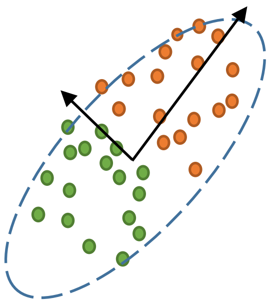


Framework of Parallel Distance Learning on Spark

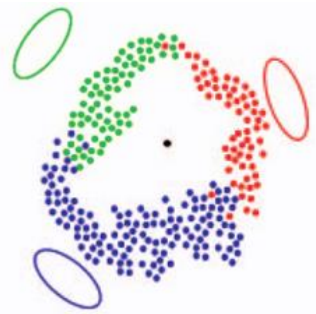


- Broadcast and Accumulator are shared with workers.
- Worker i is responsible for updating z_i

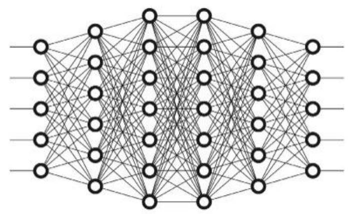
Global



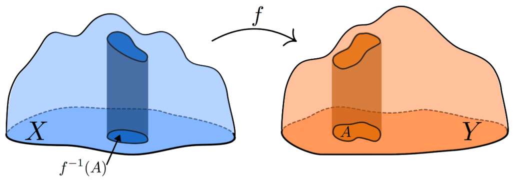
Local



Deep



Probabilistic



Global Distance Learning

Parallel Algorithm

Distributed Implementation

Theoretical Analysis

Experiments

Setup

Environment

- 32 physical machines, 4TB memory, 668 logical cores
- 10Gbps network switch
- Apache Spark 1.6.0 with Scala 2.10
- YARN cluster management

Synthetic Datasets

- Binary classification
- Dimensions range from 10^2 to 10^5
- The number of constraints range from 10 to 10^4

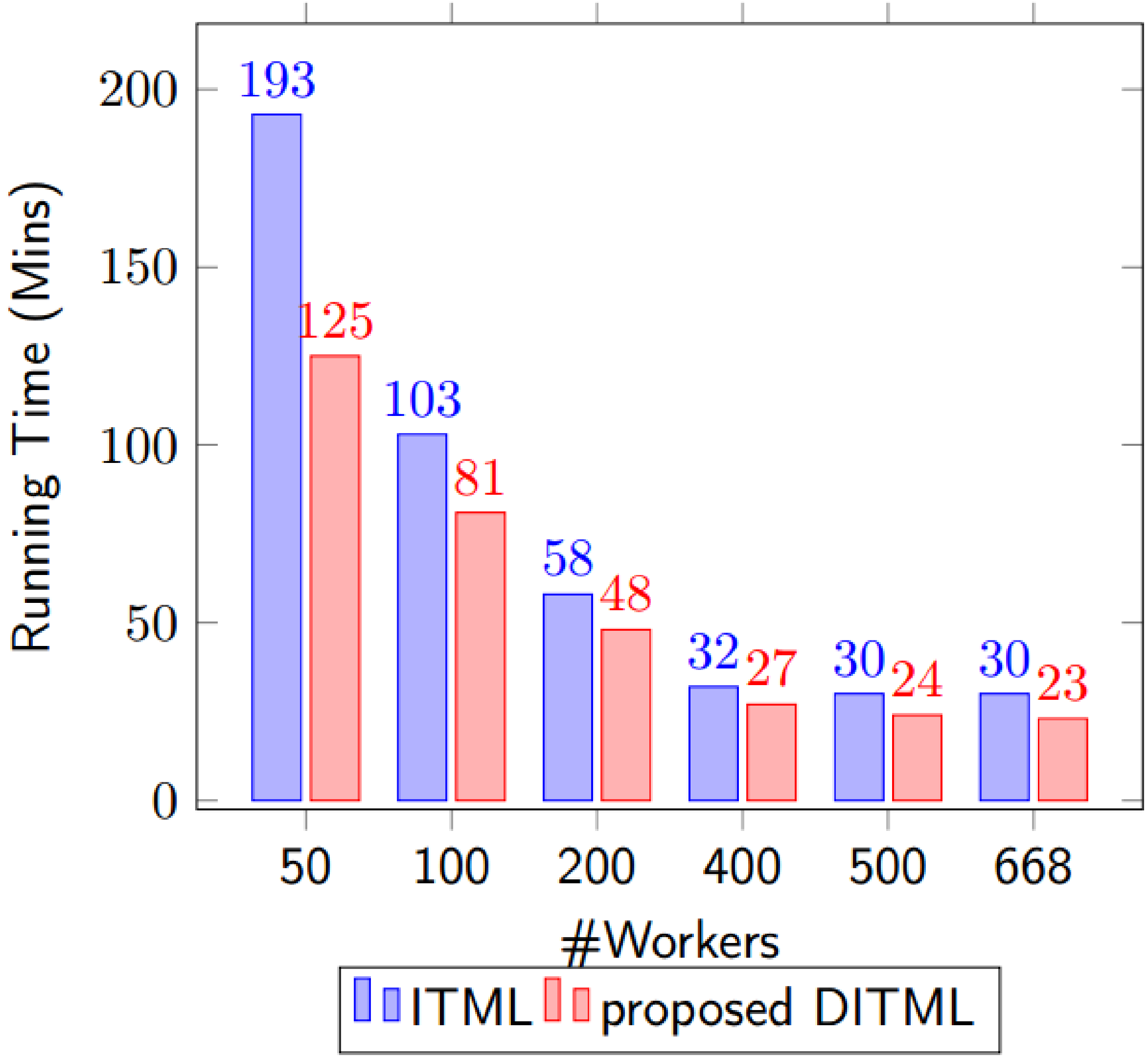
ImageNet Datasets

- DeCAF [1] features with $51{,}456$ dimension
- 50 images with $1{,}225$ constraints

- We also implement the sequential ITML in Apache Spark with distributed matrix multiplication.

[1] Donahue, Jeff, et al. "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition." international conference on machine learning (2014): 647-655.

Scalability

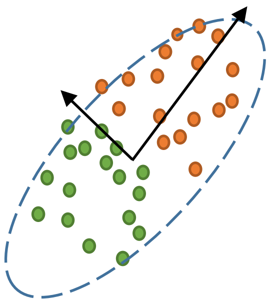


Performance on Accuracy

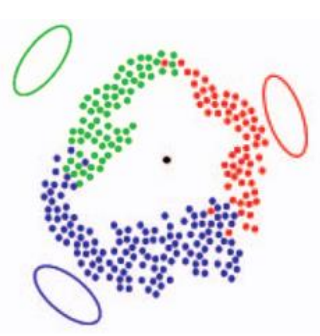
Accuracy of k-NN classification when $k = 4$

Accuracy	k-NN	ITML+k-NN	Proposed DITML+k-NN
Synthetic-10^2	0.900	0.930	0.920
Synthetic-10^3	0.940	0.962	0.957
Synthetic-10^4	0.933	0.938	0.938
Synthetic-10^5	0.812	0.923	0.900
ImageNet	0.682	0.835	0.830

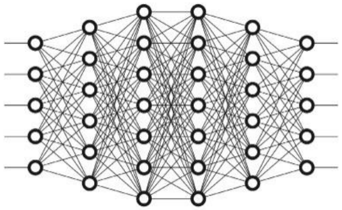
Global



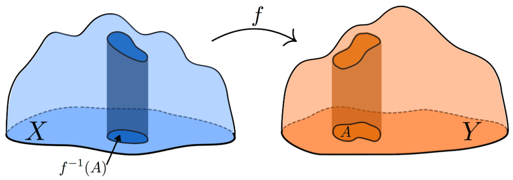
Local



Deep



Probabilistic

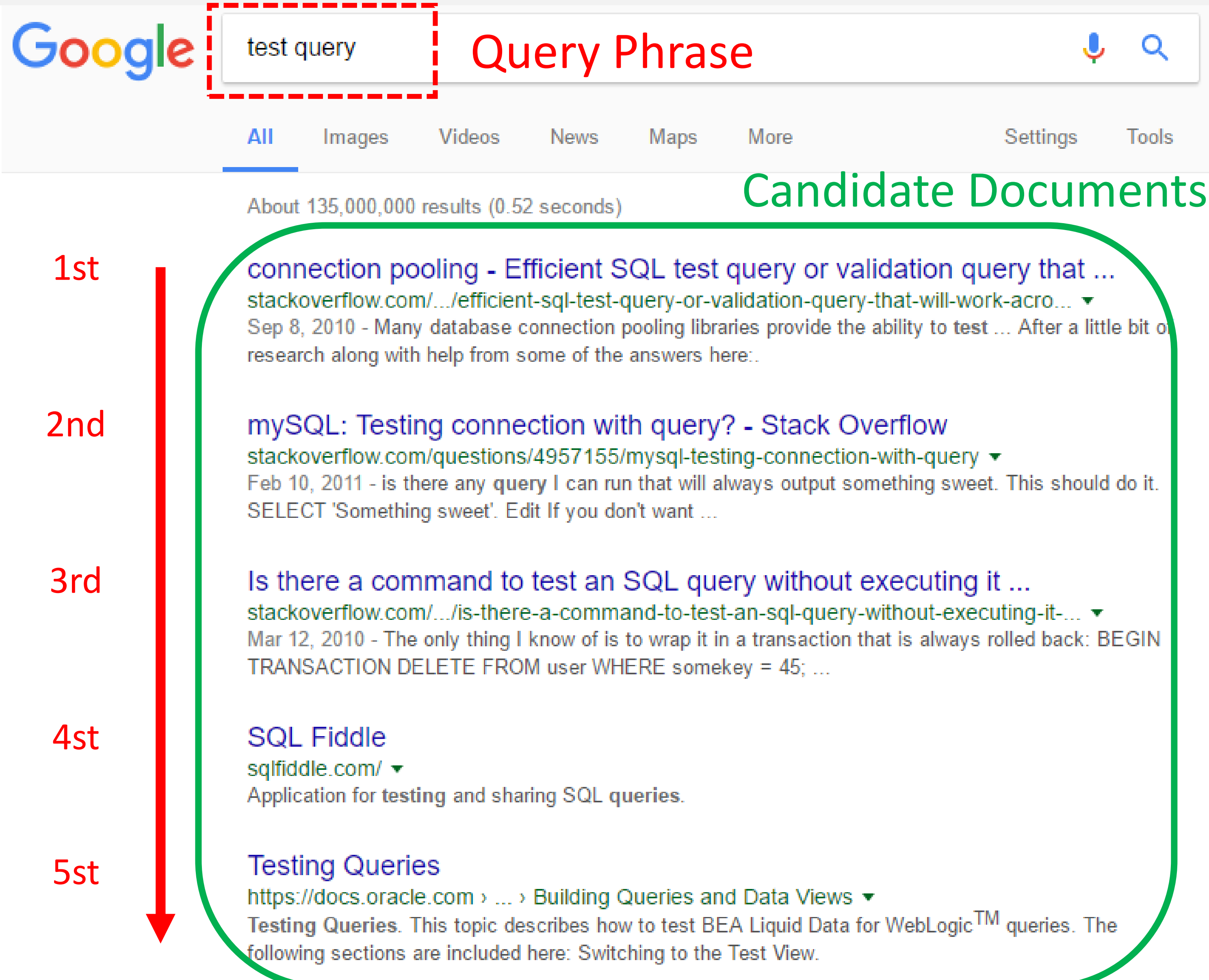


Local Distance Learning

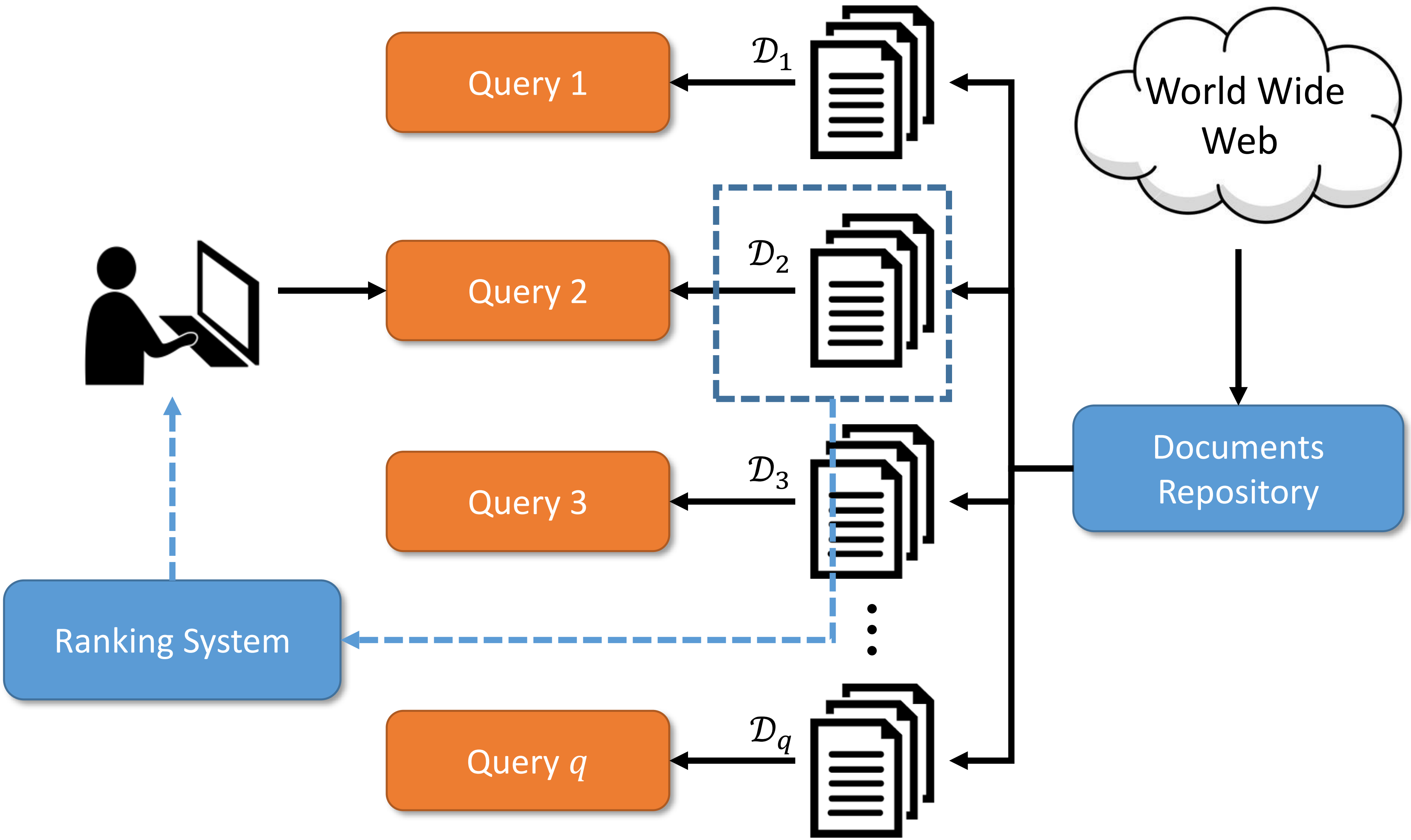
Learning to rank

Experiments

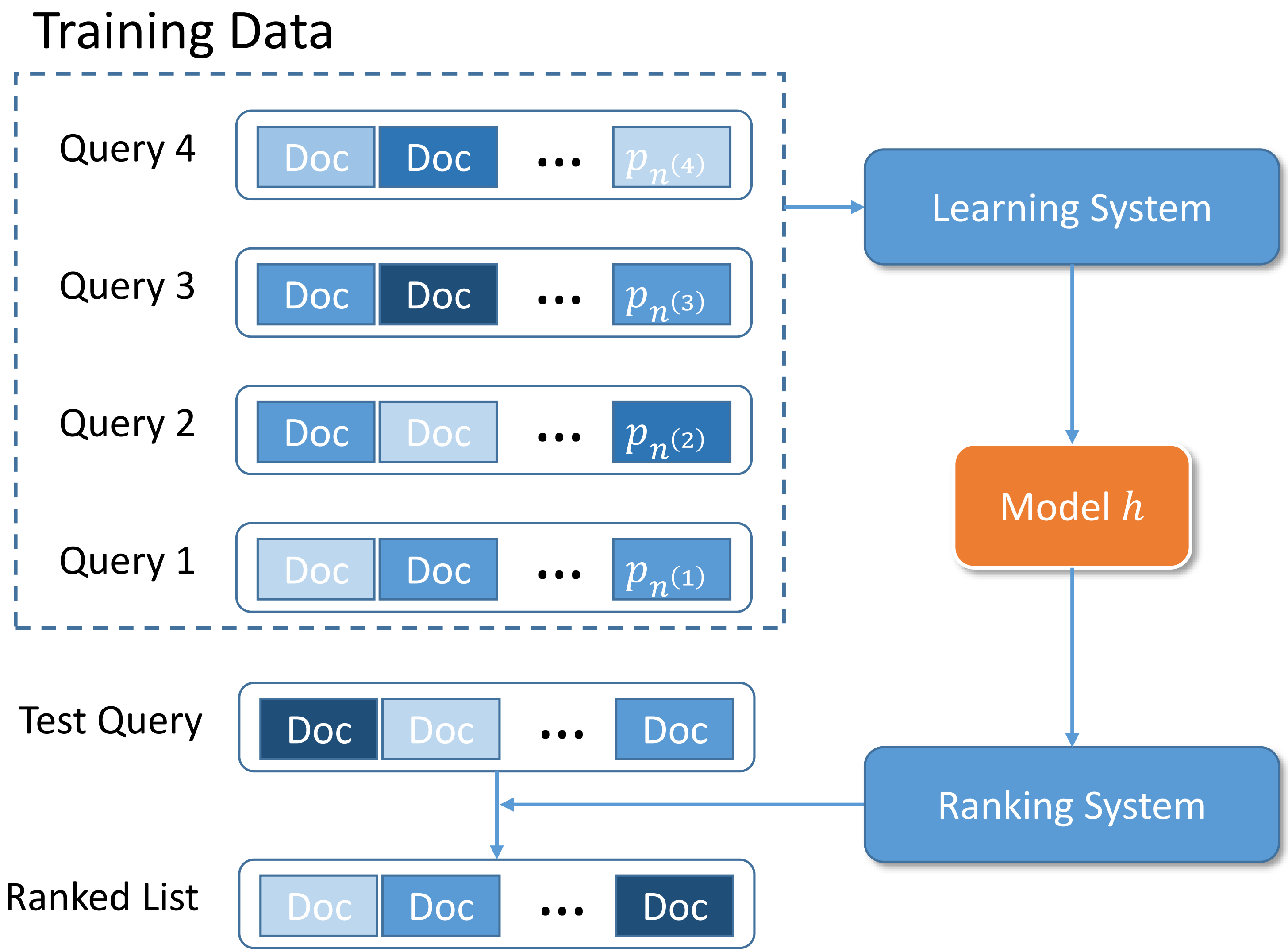
Search Engine



Candidate Documents: Query Independent



Learning to Rank for Query-document Pair



Feature List of Microsoft Learning to Rank Datasets

Feature ID	Description	Stream
3	Covered query term number	Title
4		URL
5		Whole document
71	Sum of TF-IDF	Title
72		URL
73		Whole document
106		Title
107		URL
108		Whole document
128		
129	Out-link number	
130	PageRank	
131	SiteRank	
134	URL click count	

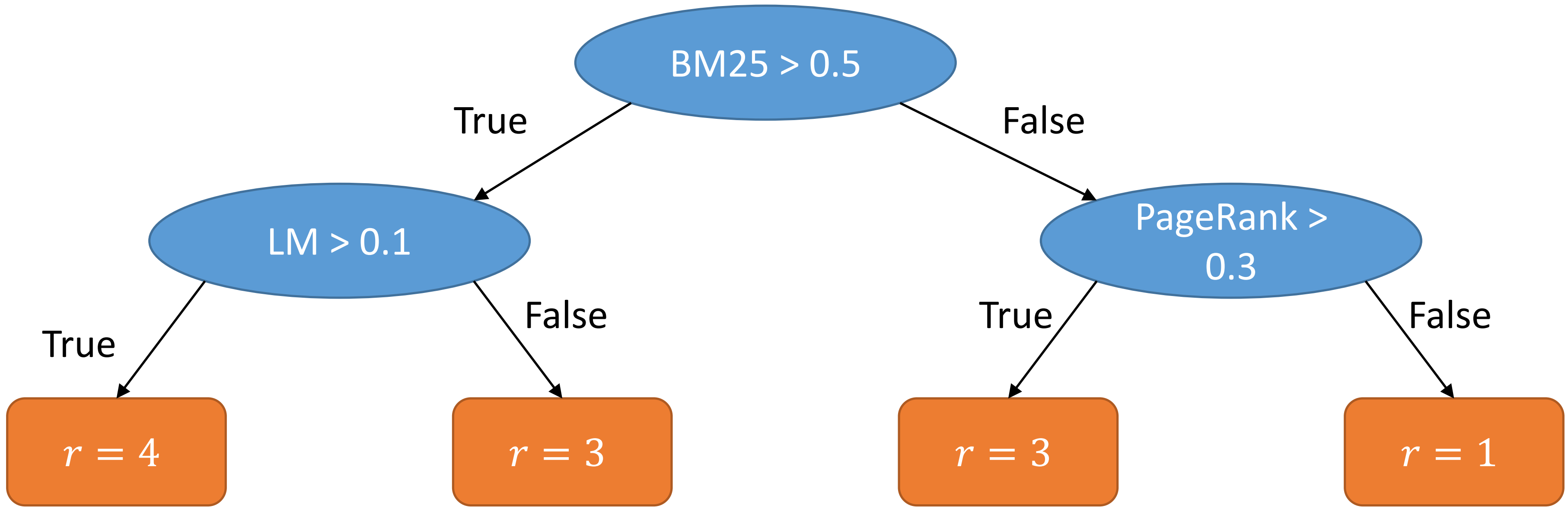
Observation

- Contain local structures in data

https://www.microsoft.com/en-us/research/project/mslr

The State-of-the-art Methods

- Gradient-Boosted Regression Tree (GBRT) [1]
- λ-MART [2]



Drawback of decision tree-like methods

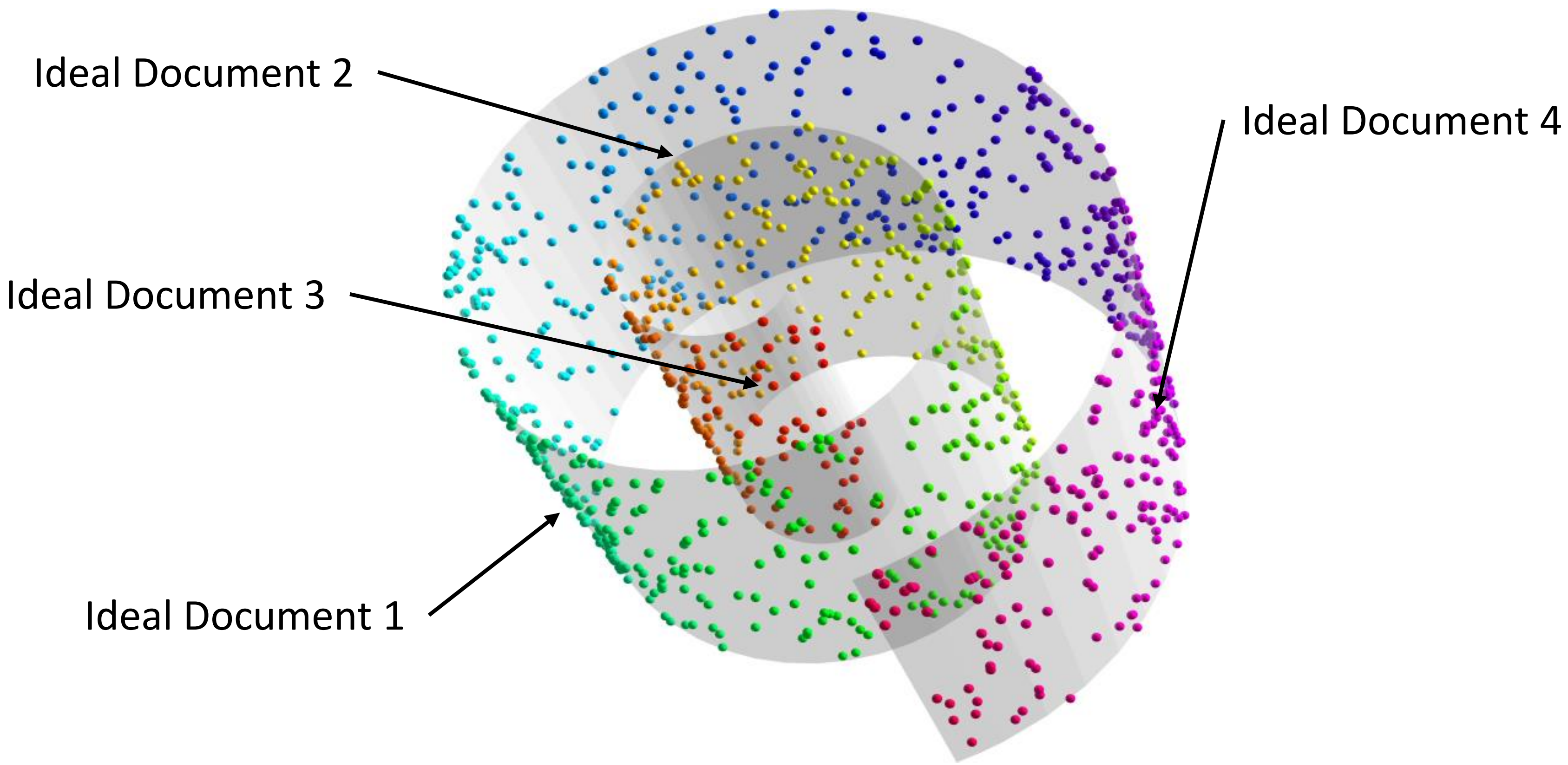
- Sensitive to noise
- No structural information (weak for theoretical analysis)

[1] Friedman, Jerome H. "Greedy function approximation: a gradient boosting machine." *Annals of statistics* (2001): 1189-1232.
[2] Burges, Christopher, et al. "Learning to rank using an ensemble of lambda-gradient models." *Proceedings of the learning to rank Challenge*. 2011.

Ideas from Manifold Learning

Motivation

- Find better similarity measurement on the feature space of query-document pair



Contributions

Method

Distance Learning

Geometric Mean Metric Learning → First → Local Geometric Mean Metric Learning → First → Learning to Rank for Query-document Pairs

Experiments: outperforms in terms of accuracy and computational complexity

- The state-of-the-art query-dependent metric-learning-to-rank algorithms
- The state-of-the-art learning-to-rank methods for query-document pairs

Geometric Mean Metric Learning

- Similar and Dissimilar Matrices:

$$\mathbf{S} = \sum_{(p_i, p_j) \in S} (p_i - p_j)(p_i - p_j)^{\mathsf{T}}$$

$$\mathbf{D} = \sum_{(p_i, p_k) \in D} (p_i - p_k)(p_i - p_k)^{\mathsf{T}}$$

- Compute the metric:

$$\mathbf{M} = \mathbf{S}^{-1/2}\left(\mathbf{S}^{1/2}\mathbf{D}\mathbf{S}^{1/2}\right)^{1/2}\mathbf{S}^{-1/2}$$

- The fastest distance metric learning algorithm

Zadeh, Pourya, Reshad Hosseini, and Suvrit Sra. "Geometric mean metric learning." International conference on machine learning. 2016.

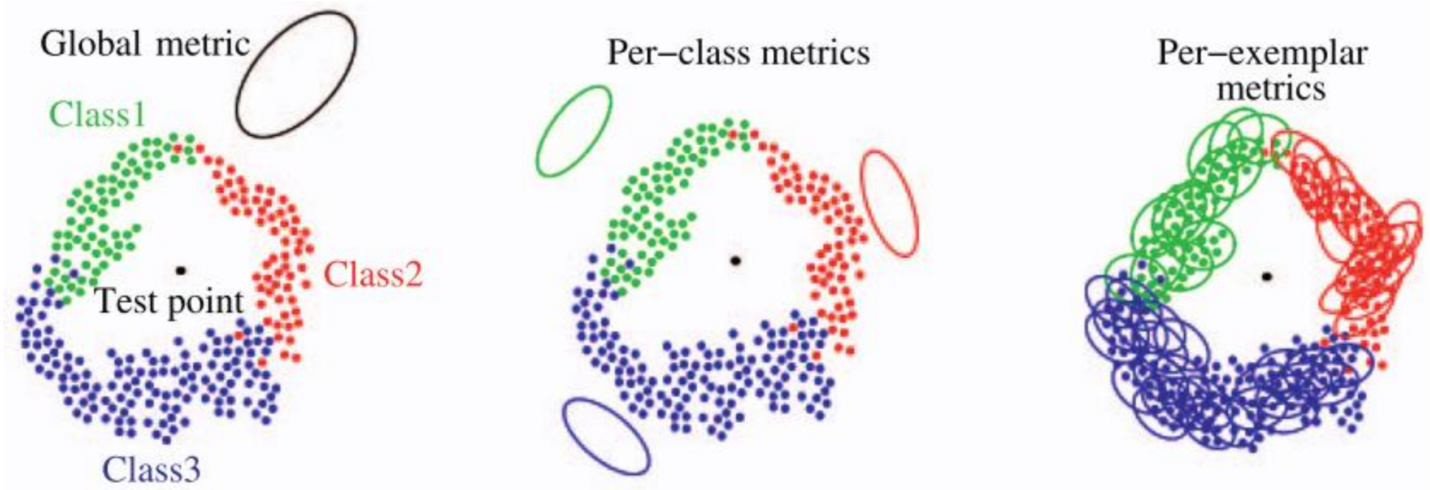
Localized Metric Learning

- A single (global) metric is a linear transformation

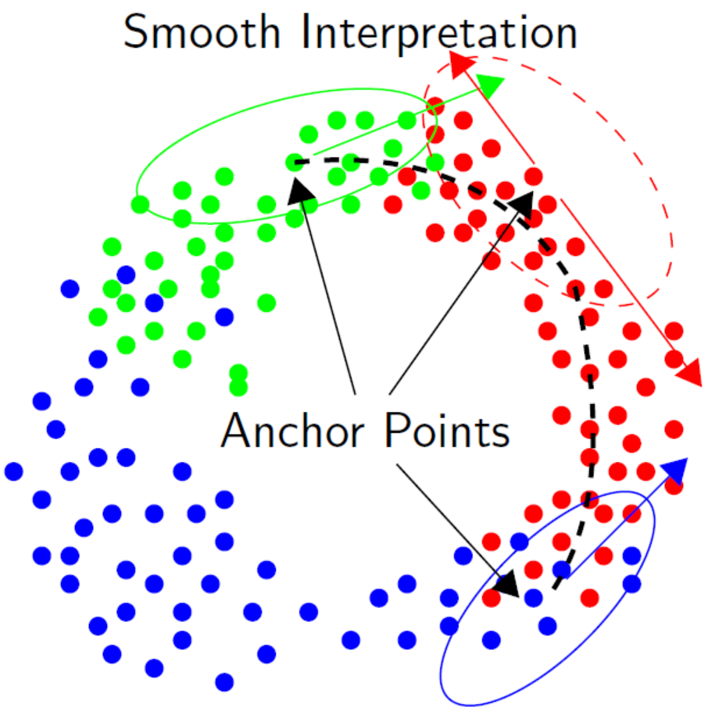
$$d_{\mathbf{M}}(p_1, p_2) = (p_1 - p_2)^\top \mathbf{M}(p_1 - p_2) = \left(\mathbf{L}(p_1 - p_2)\right)^\top \left(\mathbf{L}(p_1 - p_2)\right)$$

- Local metric contains multiple basis metrics:

$$\begin{aligned} d(p_i, p_j) &= d_{\mathbf{M}(p_i)}(p_i, p_j) \\ \mathbf{M}(p_i) &= \sum_{r=1}^{m} w_r(p_i)\mathbf{M}_r \end{aligned}$$



Smooth Interpretation



Theorem

Riemannian metric M_p is a smoothly varying:

$$\langle x_i, x_j \rangle_p = x_i^T \mathbf{M}(p) x_j$$

- For the anchor document p_r, we propose a smoothing weight function:

$$w_r(p) = \exp\left(-\frac{\rho}{2} \| p - p_r \|_{\mathbf{M}_r}\right)$$

- Easy to compute

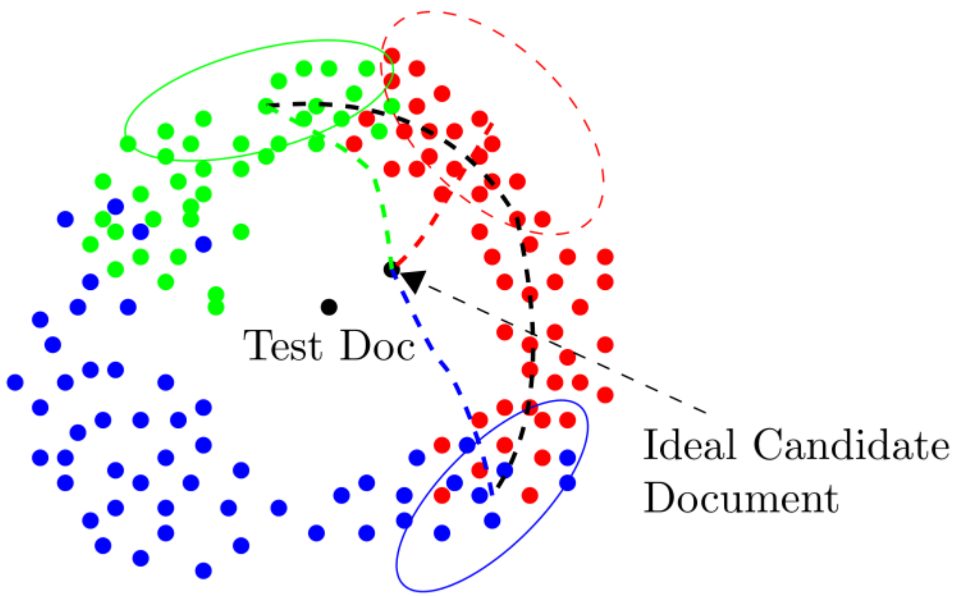
Hauberg, Søren, Oren Freifeld, and Michael J. Black. "A geometric take on metric learning." Advances in Neural Information Processing Systems. 2012.

Ideal Candidate Document

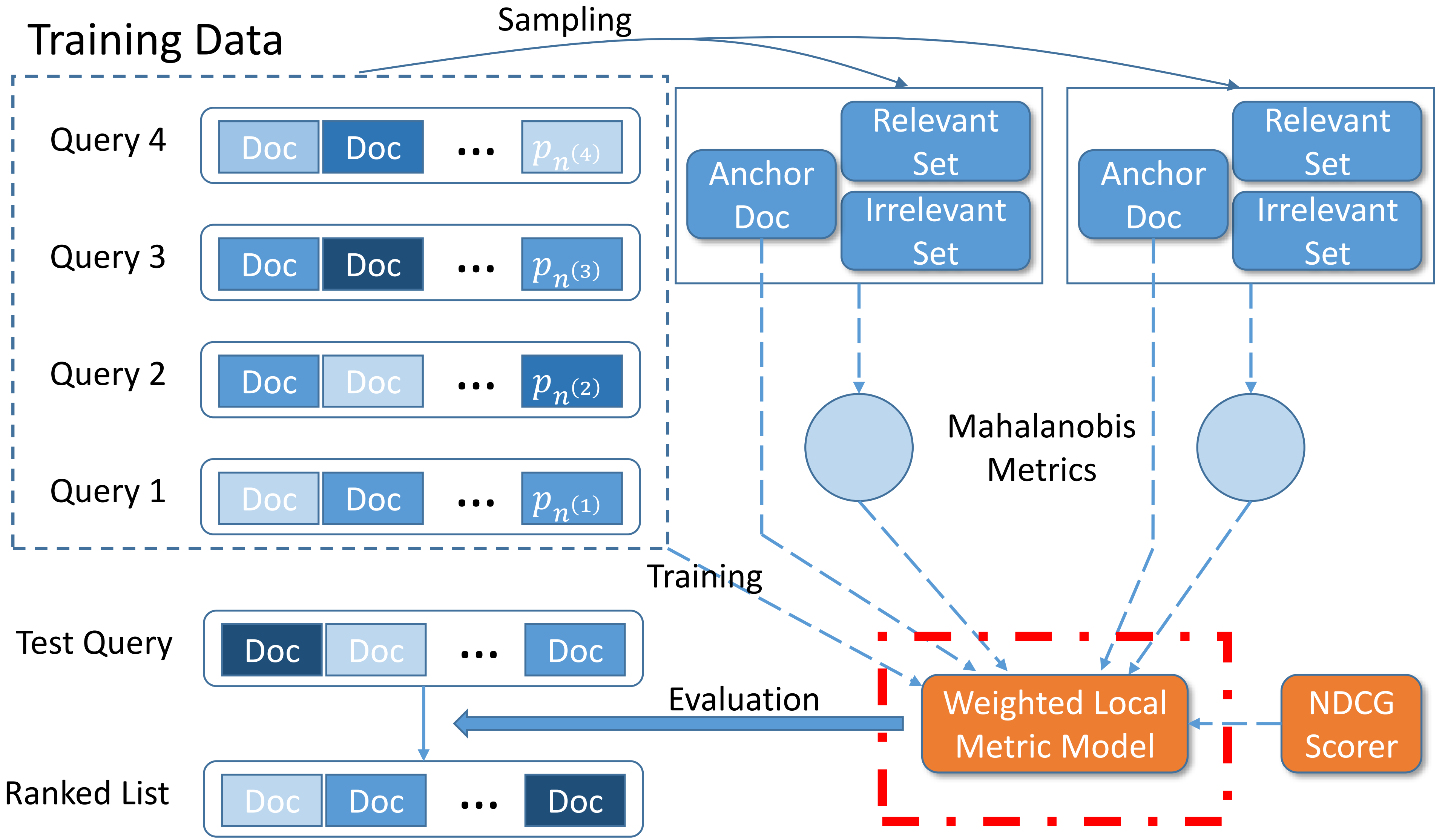
Evaluation Function

$$f_q(p, \Phi_q) = -\sum_{r=1}^{m} \Phi_q^{(r)} \cdot \exp\left(-\|p - p_r\|_{M_r}\right) \cdot \|p - p_r\|_{M_r}$$

- Relevance to q for document p
- Φ_q need to learn
- The combination of p_r is considered as ideal candidate document



Proposed Framework



Weighted Approximate Rank Pairwise

- For a set of candidate document $\mathcal{D}_q$ with query q:

Weighted Approximate Rank Pairwise (WARP) loss

$$\mathcal{L}(q) = \frac{1}{\left|\mathcal{D}_q^+\right|} \sum_{p \in \mathcal{D}_q^+} L\left(v_q(p^+)\right)$$

$v_q(p^+)$ is the number of violators in $\mathcal{D}_q$ for positive p^+

$$v_q(p^+) = \sum_{p^- \in \mathcal{D}_q^-} \mathbf{I}\left[f_q\left(p^-, \Phi_q\right) - f_q\left(p^+, \Phi_q\right)\right]$$

NDCG score

$$L(k) = \sum_{i=1}^{k} \frac{1}{\log_2(i+1)}$$

Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. Machine Learning, 81(1):21–35, 2010.

Update of Φ_q

Stochastic gradient descent to minimize the WARP loss

$$\begin{aligned}&\Phi_q(t+1)\\&=\Phi_q(t)-\mu\frac{\partial l(q,p^+,p^-)}{\partial\Phi_q(t)}\\&=\Phi_q(t)-\mu L\left(\left\lfloor\frac{|\mathcal{D}_q^-|}{N_q}\right\rfloor\right)\cdot\left[\frac{\partial f_q(p^-,\Phi_q(t))}{\partial\Phi_q(t)}-\frac{\partial f_q\left(p^+,\Phi_q(t)\right)}{\partial\Phi_q(t)}\right]\end{aligned}$$

- $\frac{\partial f_q(p,\Phi_q)}{\partial\Phi_q}=\left[\frac{\partial f_q\left(p,\Phi_q^{(r)}\right)}{\partial\Phi_q^{(r)}}\right]_{r=1\ldots m}$
- $\frac{\partial f_q\left(p,\Phi_q^{(r)}\right)}{\partial\Phi_q^{(r)}}=\exp\left(-\|p-p_r\|_{M_r}\right)\cdot\|p-p_r\|_{M_r}$

Algorithm

ALGORITHM 1: L-GMML to Rank

Input: Candidate set for c queries $\{\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_q, \ldots, \mathcal{D}_c\}$, m : number of local metrics, T : number of iteration, μ : step size, ζ: hinge loss margin

Output: $\{(p_1, M_1), (p_2, M_2) \ldots, (p_m, M_m)\}$: set of local metrics and associated anchor points, $p \in \mathbb{R}^d$, $M \in \mathbb{S}_+^d$, $\Phi \in \mathbb{R}^{c \times m}$: weights for local metrics to model the *ideal candidate documents* for each queries

for $q \in [1, c]$ **do**
  Extract $\mathcal{D}_q^+$ and $\mathcal{D}_q^-$ from $\mathcal{D}_q$;
end
for $i \in [1, m]$ **do**
  Sample $\mathcal{D}_i^+$ and $\mathcal{D}_i^-$ from $\{\mathcal{D}_q\}_{q \in [1,c]}$;
  $M_i = \text{GMML}\left(\mathcal{D}_i^+, \mathcal{D}_i^-\right)$;
  for $p \in \mathcal{D}_i^+$ **do**
    $\Gamma_p^{(i)} \leftarrow$Sort $\mathcal{D}_i$ in ascending order by computing $\|p - d\|_{M_i}^2 \ \ \forall d \in \mathcal{D}_q$;
  end
  Find the anchor point p_r with maximum NDCG score of $\Gamma_{p_r}^{(i)}$;
end

Algorithm (cont'd)

for $t = 1$ *to* T **do**

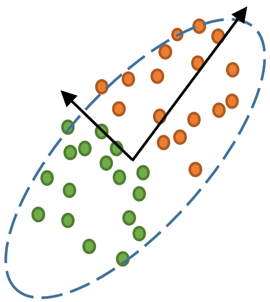
Sample a tuple (q, p^{+}, p^{-}) from $\{\mathcal{D}_q\}_{q \in [1,c]}$ such that $\zeta + f_q\left(p^{+}, \Phi_q(t)\right) > f_q\left(p^{-}, \Phi_q(t)\right)$;

$N_q \leftarrow$ the number of less relevant documents drawn with replacement from $\mathcal{D}_q^{-}$ until p^{-} is found;

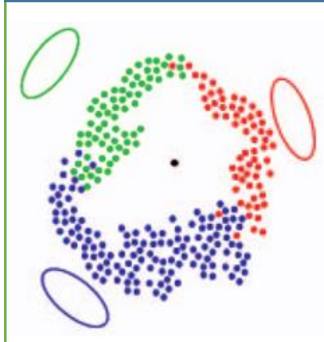
$$\Phi_q(t+1) = \left[\Phi_q(t) - \mu L\left(\left\lfloor \frac{\left|\mathcal{D}_q^{-}\right|}{N_q} \right\rfloor\right) \cdot \left[\frac{\partial f_q(p^{-}, \Phi_q(t))}{\partial \Phi_q(t)} - \frac{\partial f_q\left(p^{+}, \Phi_q(t)\right)}{\partial \Phi_q(t)}\right]\right]_{+};$$

end

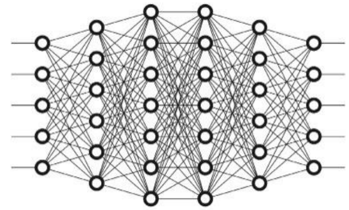
Global



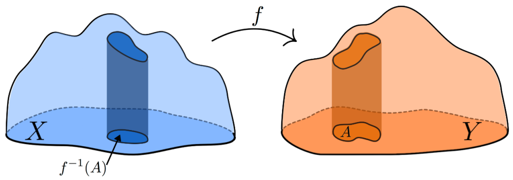
Local



Deep



Probabilistic



Local Distance Learning

Learning to rank

Experiments

Datasets

Query-independent Datasets (Query-document pairs)

Name	# of Queries		# of Doc.		Rel. Levels	# of Features
	Train	Test	Train	Test		
Yahoo! Set I	19,944	6,983	473,134	165,660	5	519
Yahoo! Set II	1,266	3,798	34,815	103,174	5	596
MSLR-WEB10K	6,000	2,000	723,412	235,259	5	136
MSLR-WEB30K	31,531	6,306	3,771k	753k	5	136

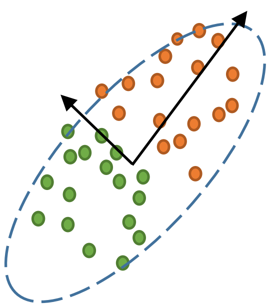
Comparison between GBRT and Proposed L-GMML

Dataset		GBRT		L-GMML	
		Test Set	Time (min.)	Test Set	Time (min.)
Yahoo! Set I	NDCG@5	0.6529	41.2	**0.6698**	28.1
	NDCG@10	0.6824	43.3	0.6715	28.9
	NDCG@20	0.6912	41.5	0.6934	28.8
Yahoo! Set II	NDCG@5	0.6731	37.6	**0.7096**	26.5
	NDCG@10	0.6817	36.8	**0.7264**	26.6
	NDCG@20	0.6954	37.4	**0.7219**	26.4
MSLR-WEB10K	NDCG@5	$0.4019 \pm \mathbf{0.0083}$	49.4 ± 5.2	$\mathbf{0.4771} \pm 0.0951$	19.7 ± 2.1
	NDCG@10	0.4342 ± 0.0219	48.3 ± 2.1	$\mathbf{0.5390} \pm 0.0812$	19 ± 3.1
	NDCG@20	0.4512 ± 0.0279	48.8 ± 3.8	$\mathbf{0.5510} \pm 0.0728$	19 ± 2.8
MSLR-WEB30K	NDCG@5	0.409 ± 0.0312	167 ± 28.6	$\mathbf{0.4837} \pm 0.0715$	71.7 ± 2.7
	NDCG@10	0.4146 ± 0.0327	177 ± 30.1	$\mathbf{0.4976} \pm 0.0619$	71.9 ± 3.9
	NDCG@20	0.421 ± 0.361	167 ± 27.3	$\mathbf{0.5038} \pm 0.0718$	72.5 ± 5.3

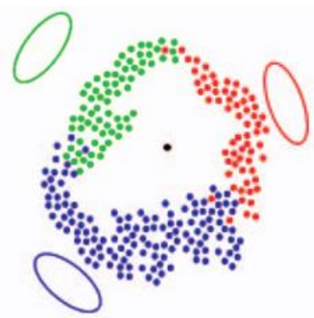
Comparison between λ-MART and Proposed L-GMML

Dataset		λ-MART		L-GMML	
		Test Set	Time (min.)	Test Set	Time (min.)
Yahoo! Set I	NDCG@5	0.6567	46.5	**0.6698**	28.1
	NDCG@10	**0.7060**	48.0	0.6715	28.9
	NDCG@20	**0.7091**	46.9	0.6934	28.8
Yahoo! Set II	NDCG@5	0.6791	43.1	**0.7096**	26.5
	NDCG@10	0.7062	43.3	**0.7264**	26.6
	NDCG@20	0.7087	43.8	**0.7219**	26.4
MSLR-WEB10K	NDCG@5	0.4417 ± 0.0131	58.3 ± 2.8	$\mathbf{0.4771} \pm 0.0951$	19.7 ± 2.1
	NDCG@10	$0.4513 \pm \mathbf{0.0196}$	57.6 ± 3.8	$\mathbf{0.5390} \pm 0.0812$	19 ± 3.1
	NDCG@20	$0.4634 \pm \mathbf{0.0257}$	57.1 ± 5.2	$\mathbf{0.5510} \pm 0.0728$	19 ± 2.8
MSLR-WEB30K	NDCG@5	$0.3812 \pm \mathbf{0.0297}$	182 ± 19.8	$\mathbf{0.4837} \pm 0.0715$	71.7 ± 2.7
	NDCG@10	$0.409 \pm \mathbf{0.0232}$	183 ± 17.9	$\mathbf{0.4976} \pm 0.0619$	71.9 ± 3.9
	NDCG@20	$0.4112 \pm \mathbf{0.0240}$	181 ± 10.7	$\mathbf{0.5038} \pm 0.0718$	72.5 ± 5.3

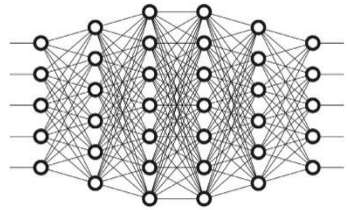
Global



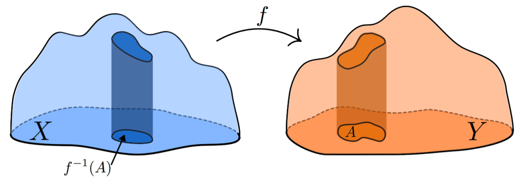
Local



Deep



Probabilistic



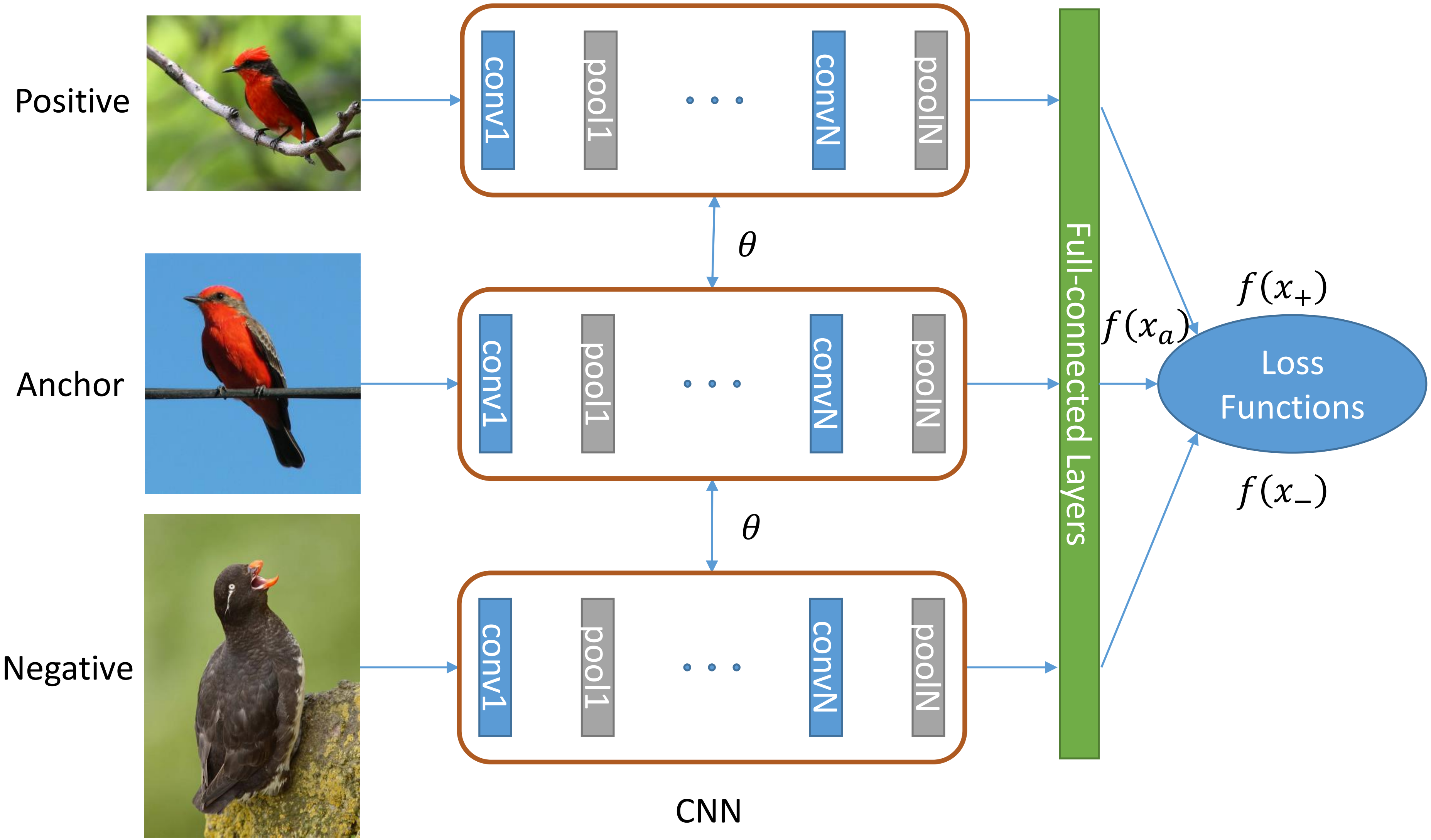
Deep Distance Learning

Parallel Algorithm

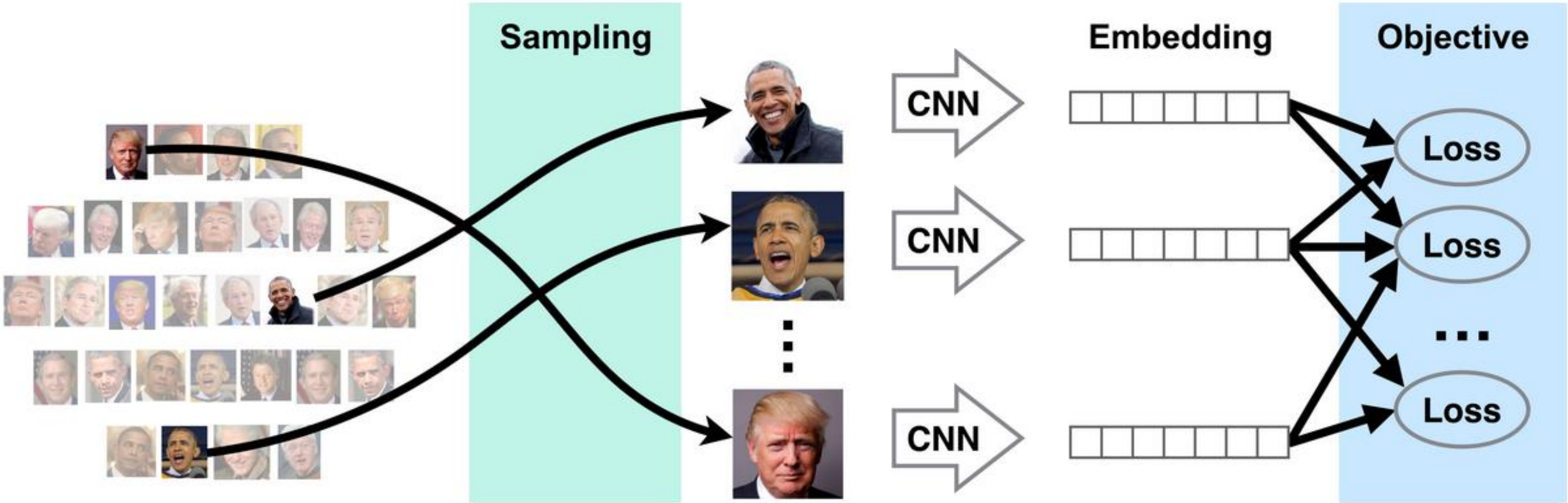
Experiments

Distributed Implementation

Siamese Networks



Most Important Issue: Sampling



N images $\longrightarrow$ $\mathcal{O}(N^3)$ triplets

Impossible task for big data set

https://www.cs.utexas.edu/~cywu/projects/sampling_matters

Semi-hard Negative Mining

- Each mini-batch:

$$\ell(X, y) = \frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \left[D_{i,j}^2 + \alpha - D_{i,k^*}^2\right]_+$$

- Where

$$k^*(i, j) = \underset{k:y[k]\neq y[i]}{\mathrm{argm}in}\ D_{i,k}$$

- Need very large minibatches (1800 images)

Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015.

N-pairs Embedding

- Each mini-batch:

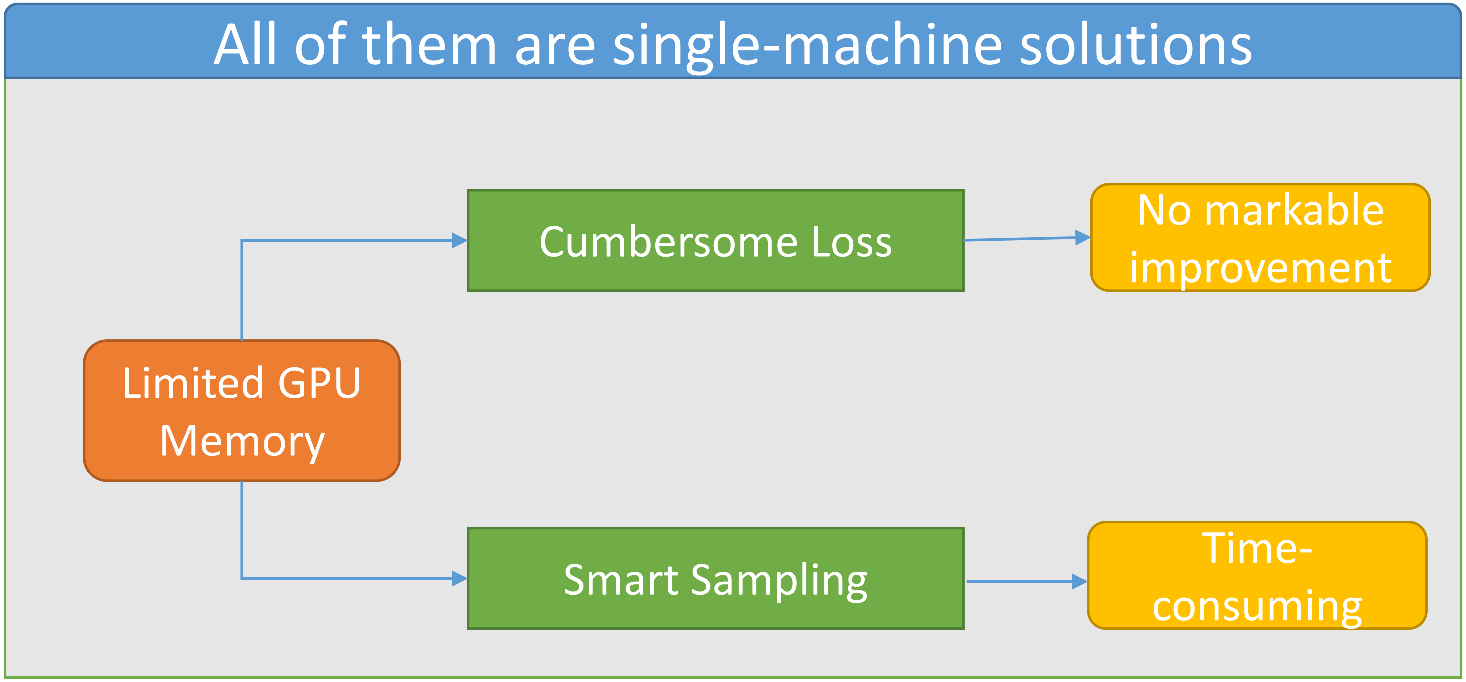
$$\ell(X, y) = -\frac{1}{|\mathcal{P}|} \sum_{(i,j)\in\mathcal{P}} \log \frac{\exp\{S_{i,j}\}}{\exp\{S_{i,j}\} + \sum_{k:y[k]\neq y[i]} \exp\{S_{i,k}\}} + \frac{\lambda}{m}\sum_{i}^{m} \|f(x_i)\|_2$$

- where $S_{i,j} = f(x_i)^T f(x_j)$

- Softmax cross-entropy loss among the pairwise similarity values

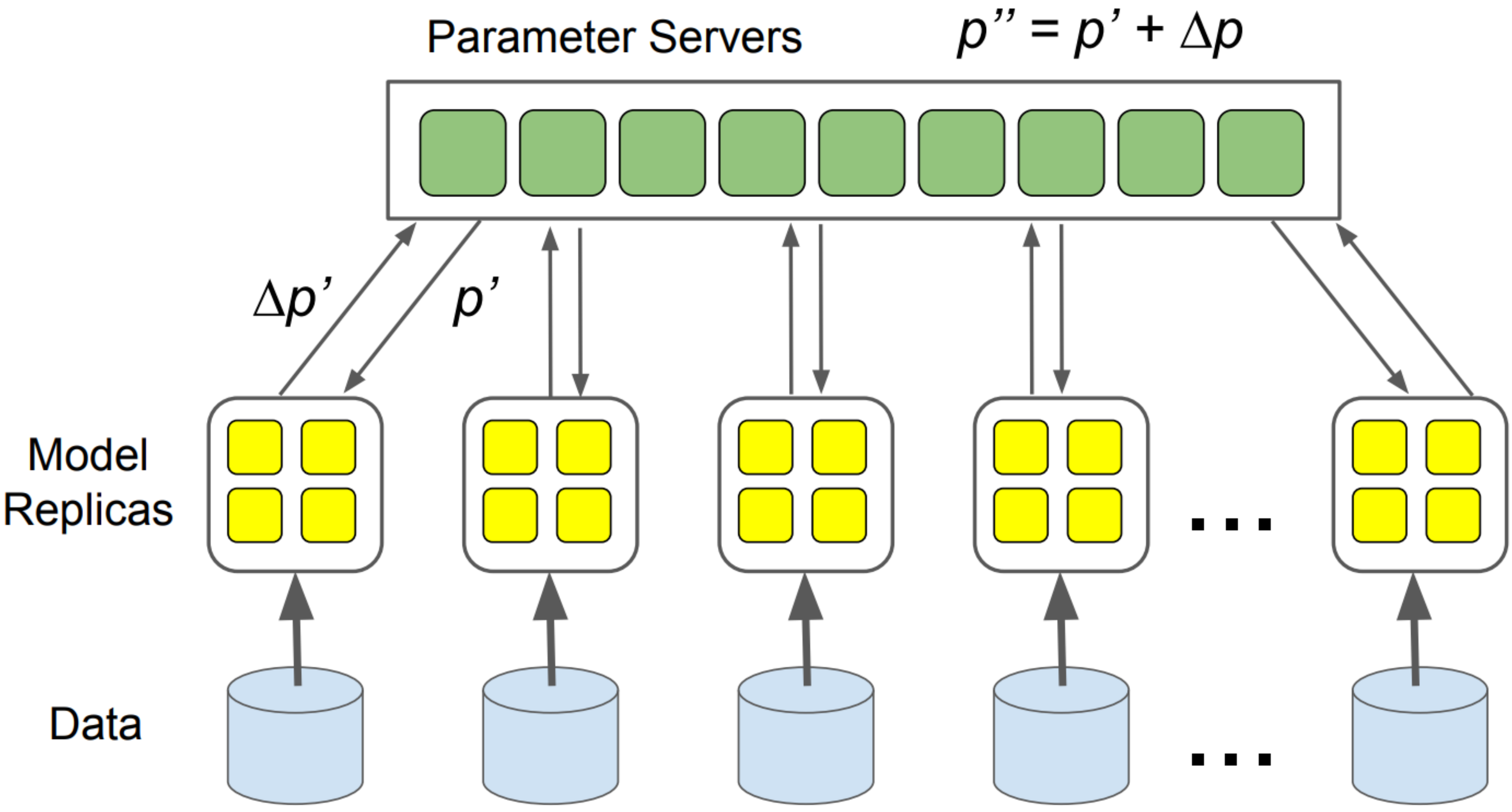
Sohn, Kihyuk. "Improved deep metric learning with multi-class n-pair loss objective." Advances in Neural Information Processing Systems. 2016.

Challenges



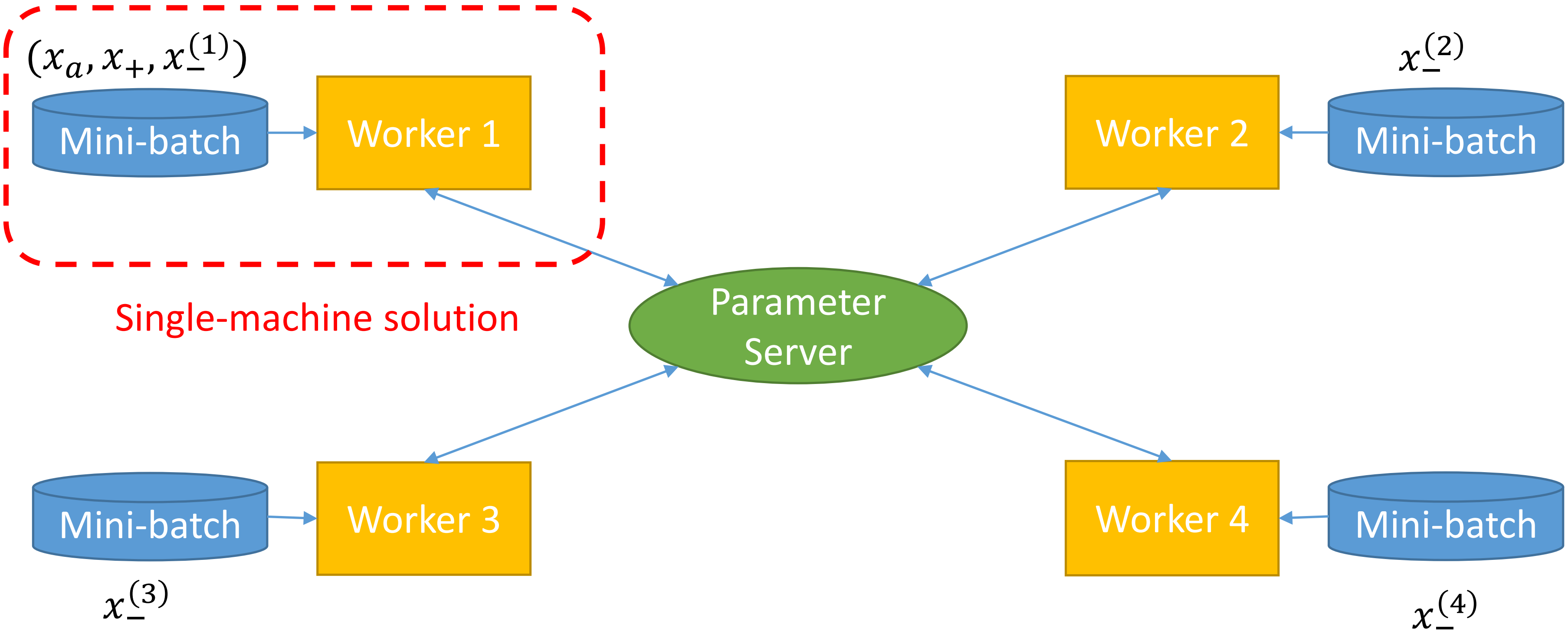
Let's consider multiple machines solution!

Distributed Deep Learning



Distributed Deep Distance Learning

$x_-^{(1)}$ is a local hard-negative sample



Maybe $x_-^{(4)}$ is the global one

Contributions

Synchronization of global hard negative samples

- This is the first distance learning oriented distributed deep distance learning approach

Hybrid synchronization

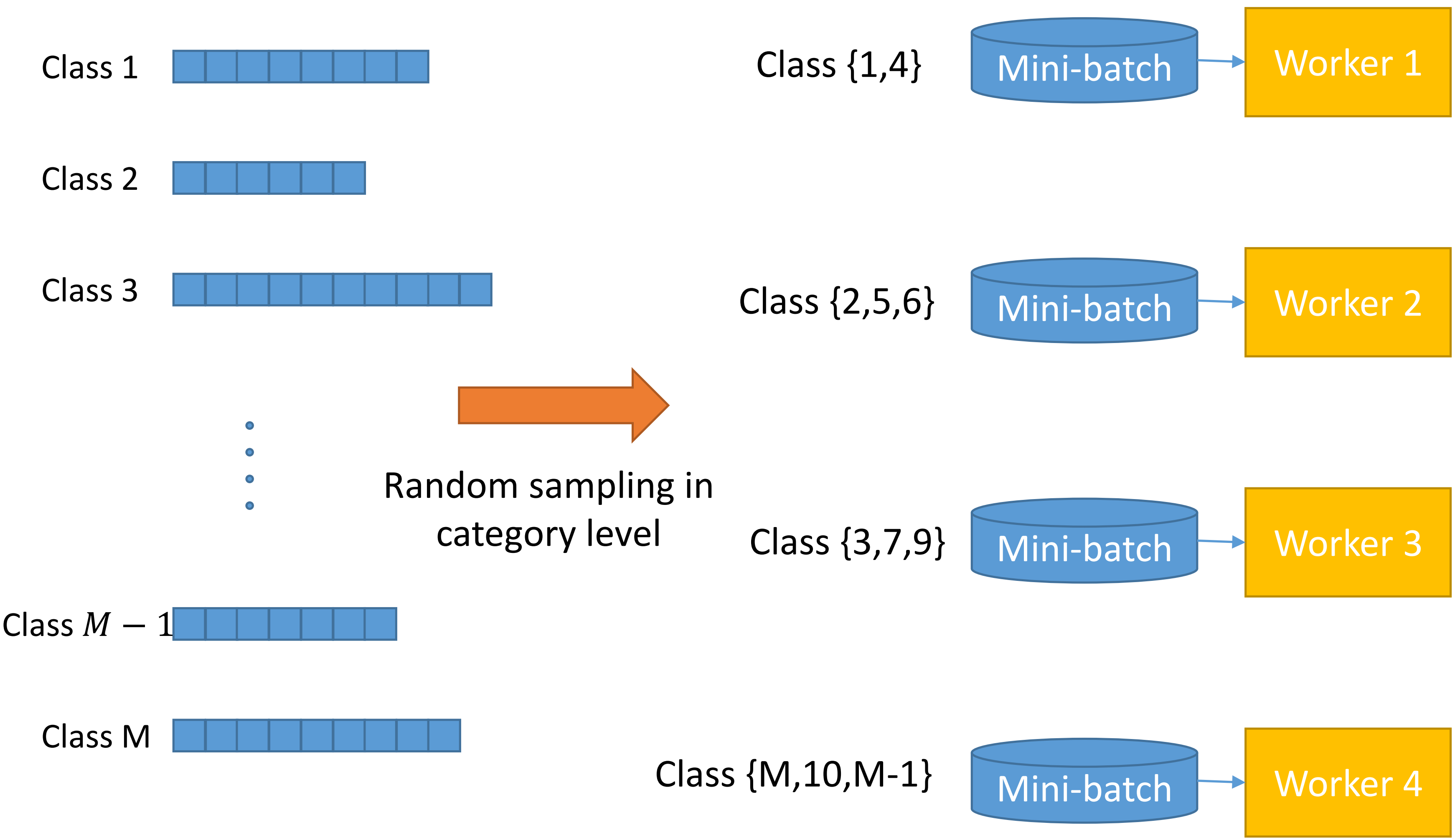
Synchronization of DNN model

- Communication-efficient approach to synchronize DNN model with mixed topology

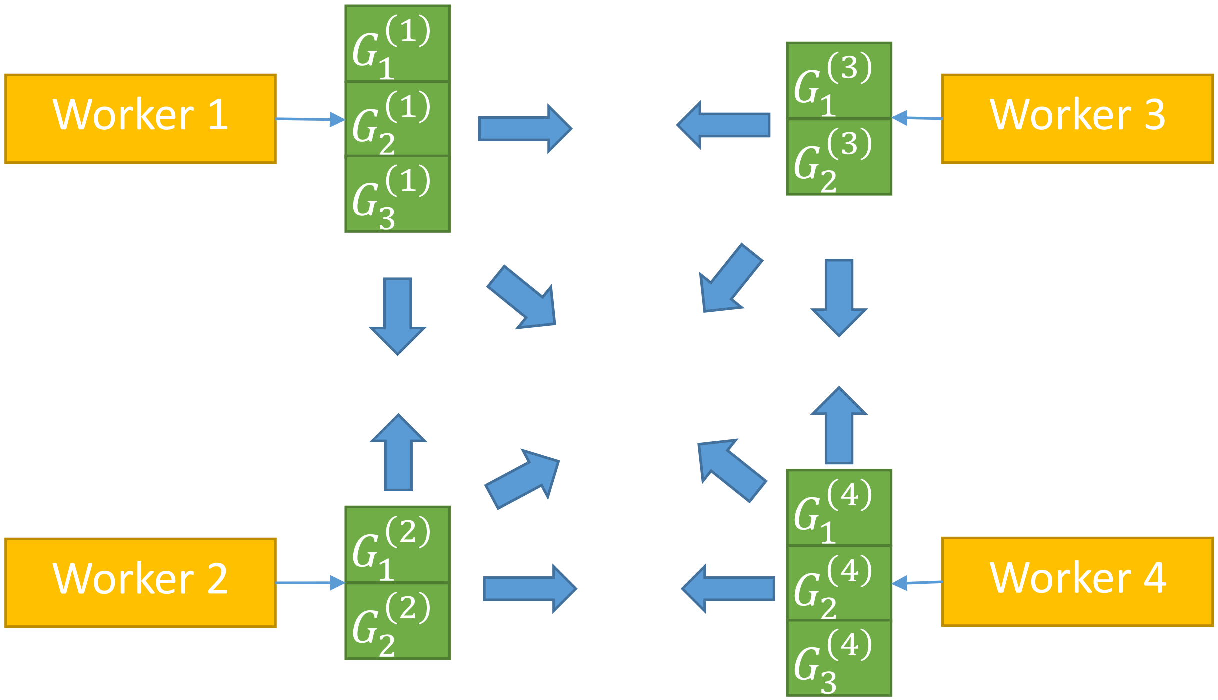
Experimental report

- Huge improvements in terms of accuracy of image retrieval and runtime speedup

Batching with Category Information



Broadcast of Embedded Anchor Points



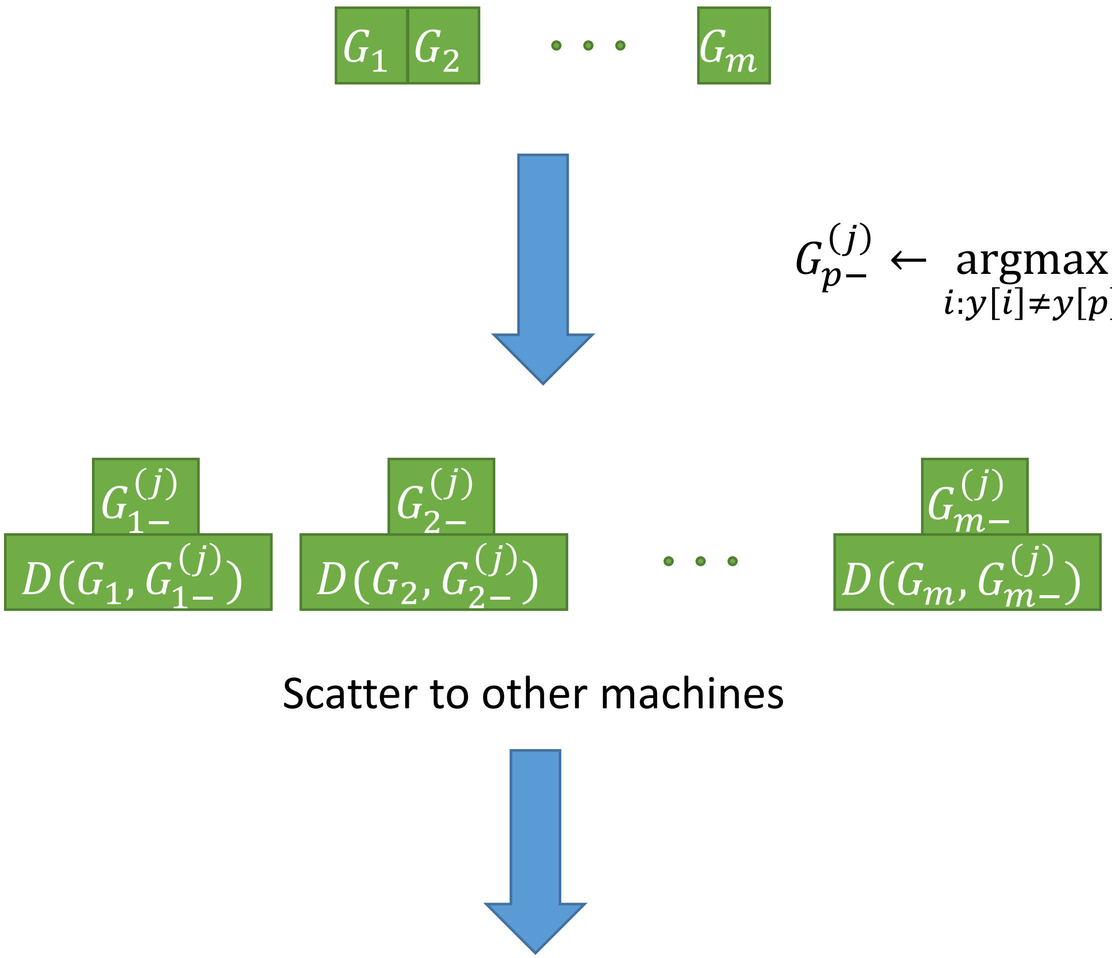
$$G_i^{(j)} = f_j\left(\left[X_a^{(j)}\right]_i\right)$$

$X_a^{(j)}$ is the collection of anchor points in worker j
$f_j(\cdot)$ is the DNN model in worker j

The dimensionality of G is relative small

Scatter the Corresponding Hard Negative Samples

- For the machine j:



$$G_{p-}^{(j)} \leftarrow \underset{i:y[i]\neq y[p]}{\operatorname{argmax}} D(G_i^{(j)}, G_p)$$

$$G_i^{(j)} = f_j(x_i)$$

N-positive Pairs Embedding

- For the machine j:

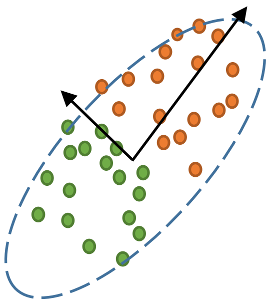
$$\ell(X, y) = -\frac{1}{|\mathcal{P}|} \sum_{(i,k)\in\mathcal{P}} \log \frac{\exp\{S_{i,k}\}}{\exp\{S_{i,k}\} + \exp\{S_{i,\mathbb{I}(i)}\}} + \frac{\lambda}{m} \sum_{i}^{m} \|f(x_i)\|_2$$

- $\mathbb{I}(i) \leftarrow \operatorname*{argmax}_{p} D\left(G_i, G_{i-}^{(p)}\right)$

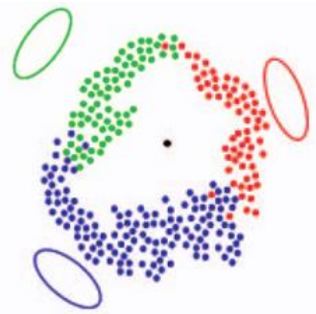
"Global" hard negative sample

- $S_{i,k} = f(x_i)^T f(x_k)$

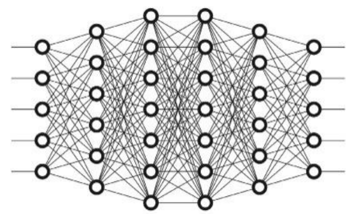
Global



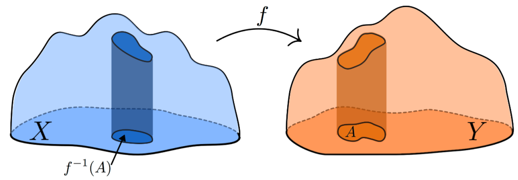
Local



Deep



Probabilistic



Deep Distance Learning

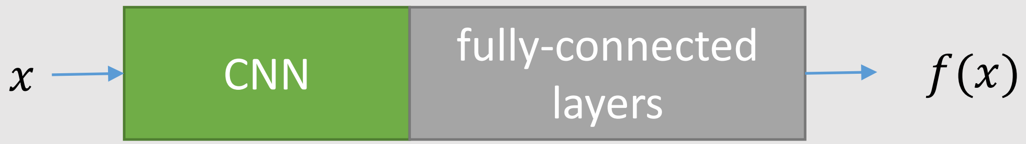
Parallel Algorithm

Experiments

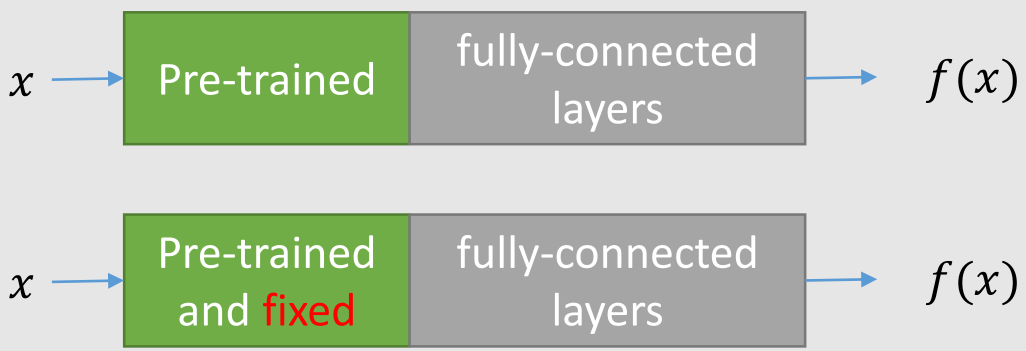
Distributed Implementation

Model Synchronization: Observation I

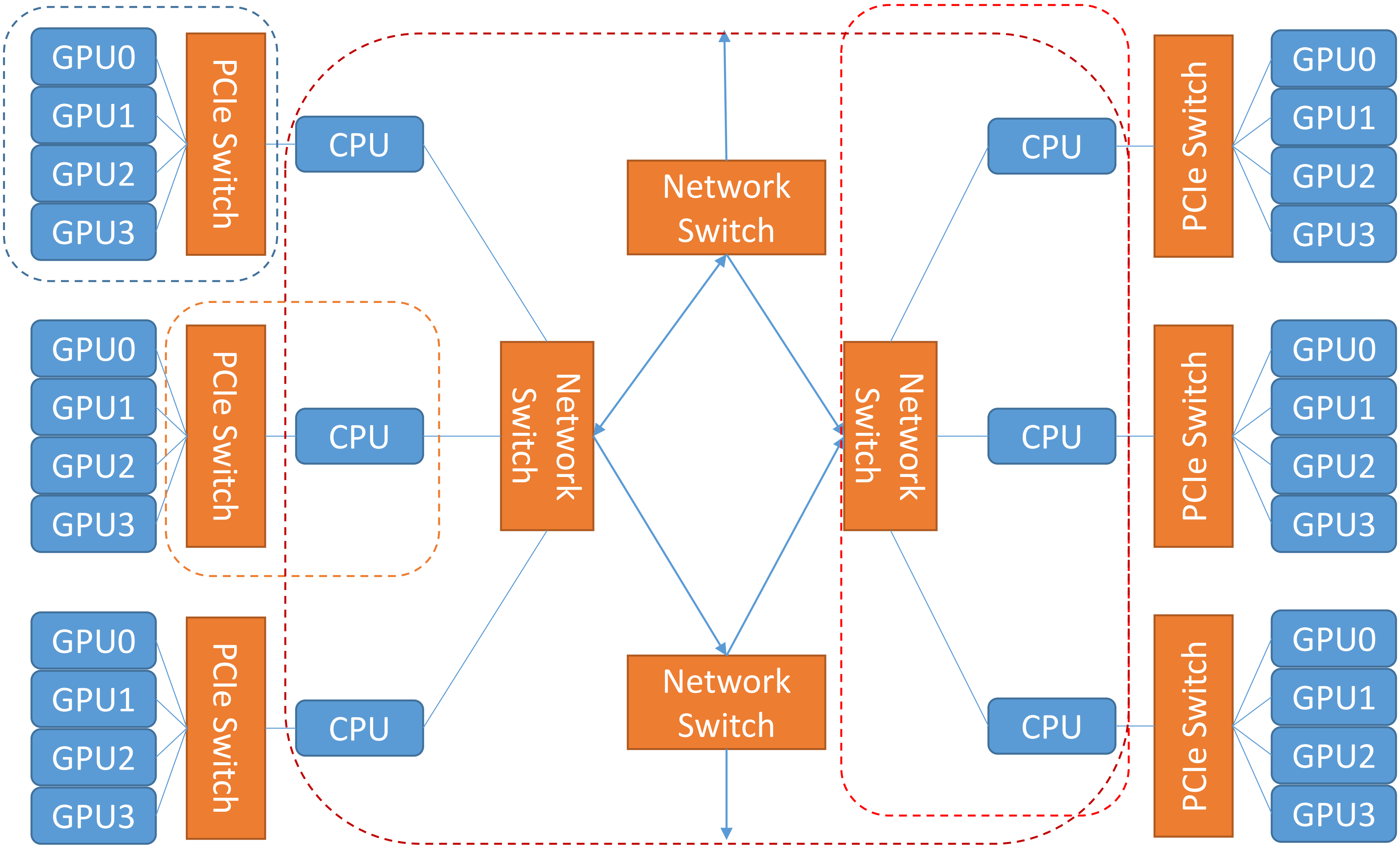
Siamese network has two distinct parts



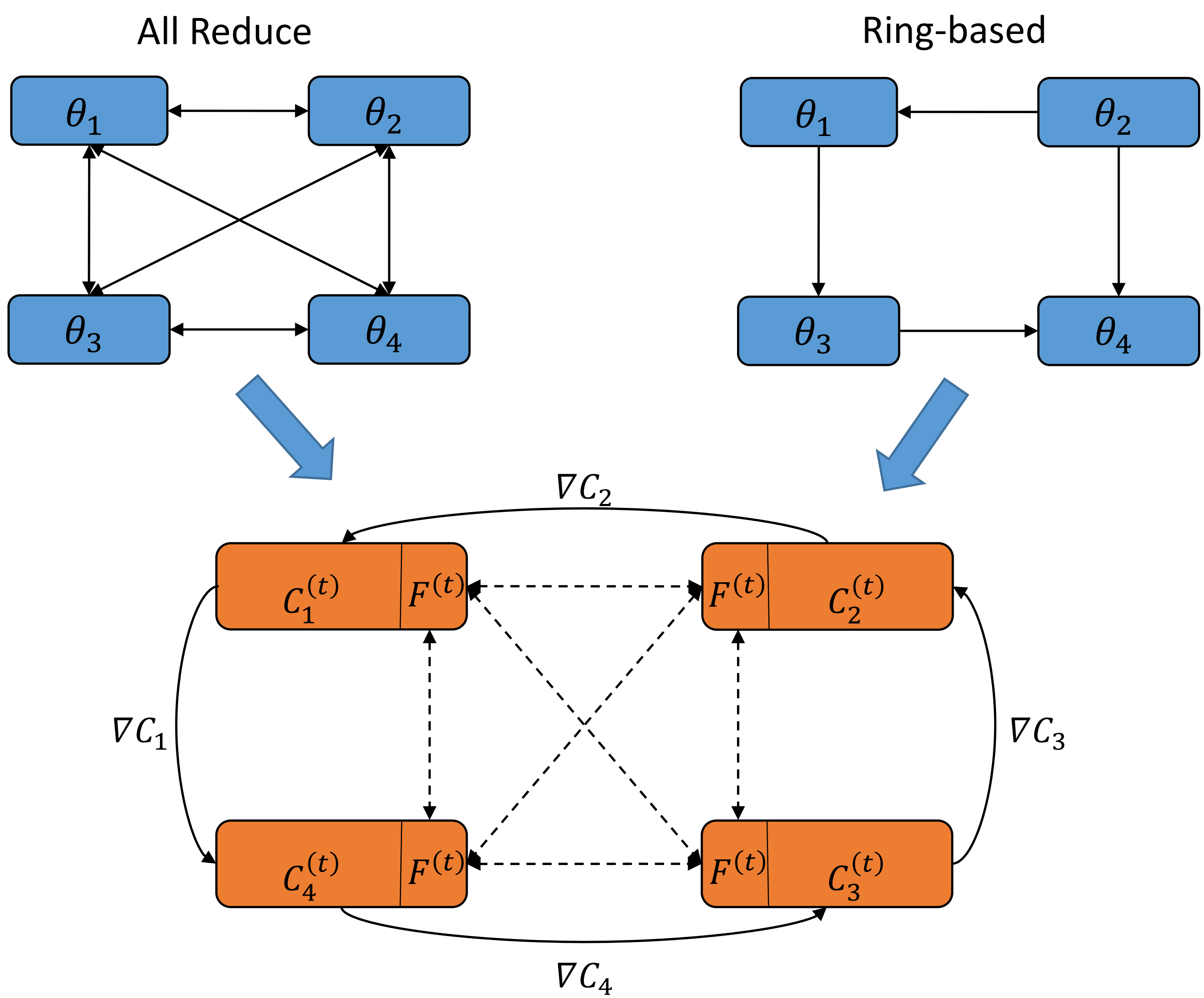
Two settings in all deep distance learning framework



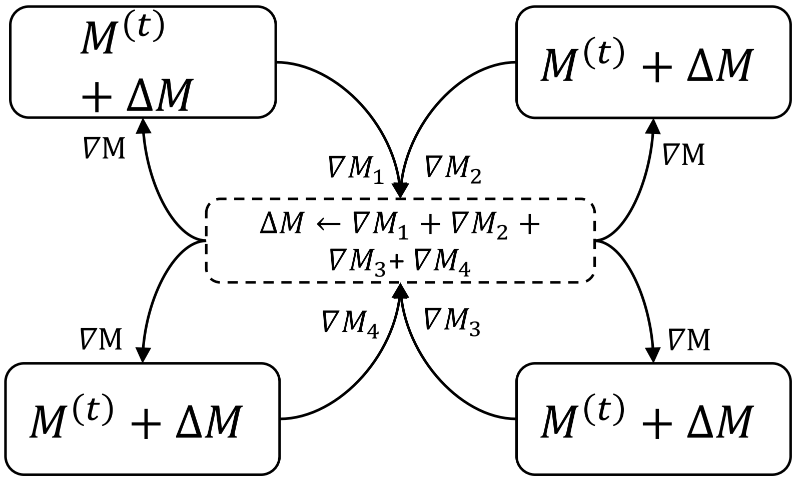
Observation II: Hierarchical Topology

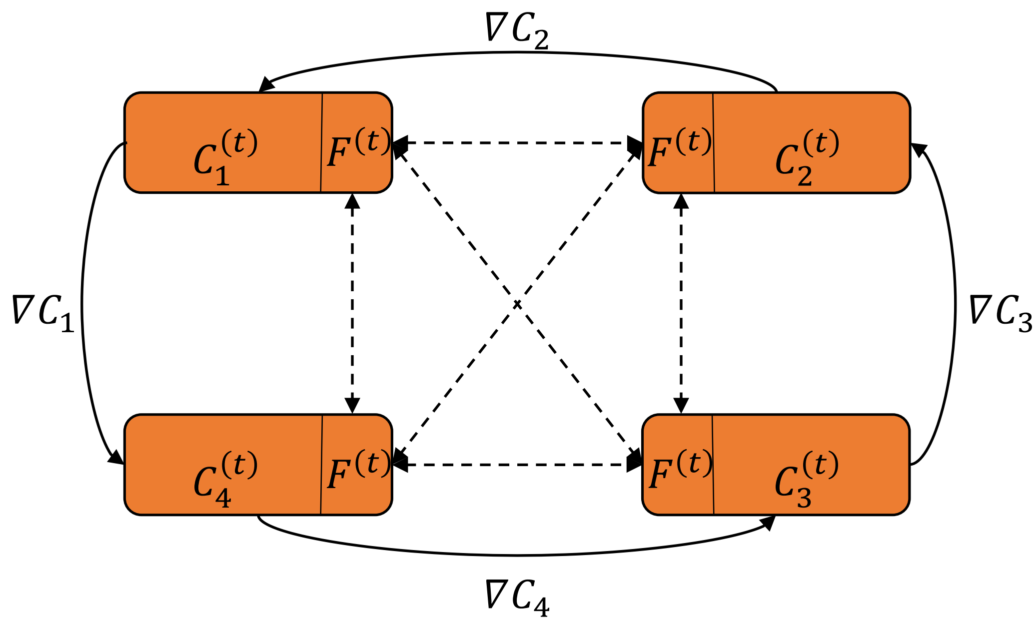


Model Synchronization: Mixed Topology



Parameter Server vs Mixed Topology

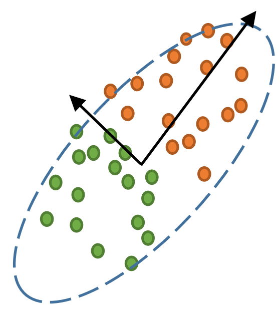




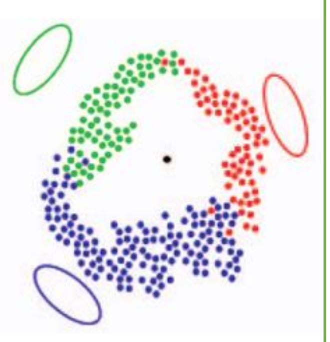
$$\theta_i^{(t+1)} \leftarrow \theta_i^{(t)} + \eta^{(t)} \sum_{j \in P} \nabla f\left(\theta_j^{(t)}\right)$$

$$C_{\mathrm{i}}^{(\mathrm{t+1})} \leftarrow C_i^{(t)} + \eta^{(t)} \left(\beta \nabla f\left(C_i^{(t)}\right) + (1-\beta)\nabla \mathrm{f}\left(C_{\mathrm{Left}(i)}\right)\right)$$

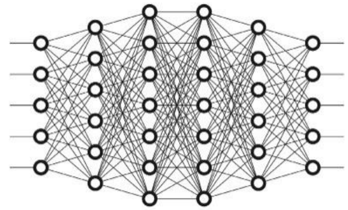
Global



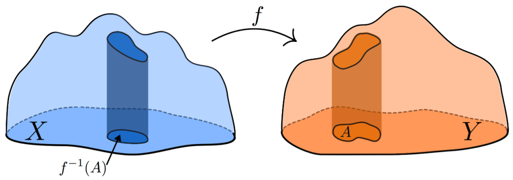
Local



Deep



Probabilistic



Deep Distance Learning

Parallel Algorithm

Experiments

Distributed Implementation

Configuration

Hardware

- 4 servers with 2 NVIDIA GTX 1080 GPUs each
- 10 Gbit/s switch network

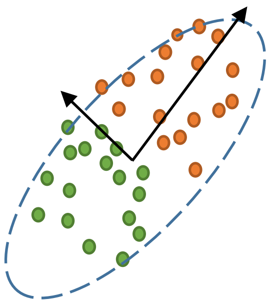
Software

- PyTorch v0.3

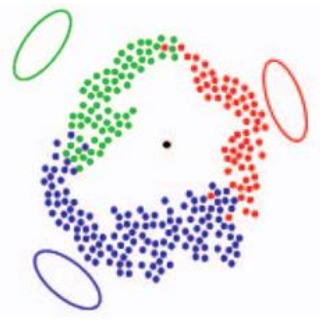
Evaluation on Cars196

Method	Time (s) / epoch	NMI	Recall@1	Recall@5	Recall@10
Triplet semi-hard [CVPR'15]	507	54.09	41.30	75.21	80.32
Lifted struct [CVPR'16]	530	56.90	53.70	75.98	83.30
Histogram [NIPS'16]	512	-	53.67	75.56	81.20
N-pairs [NIPS'16]	489	58.04	54.36	79.03	84.23
NMI-based [CVPR'17]	832	57.27	57.29	79.90	88.24
Spectral [ICML'17]	798	61.08	69.35	81.08	**90.35**
Ours (2 machines)	423	61.24	70.07	**82.13**	89.25
Ours (4 machines)	**298**	**61.30**	**70.23**	81.97	90.10

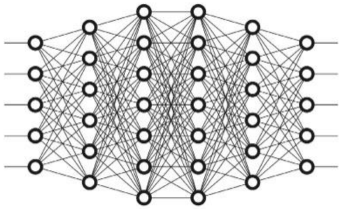
Global



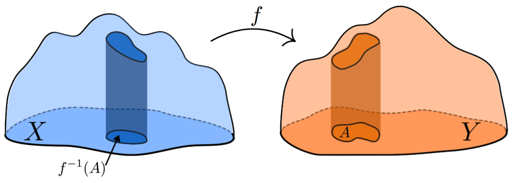
Local



Deep



Probabilistic

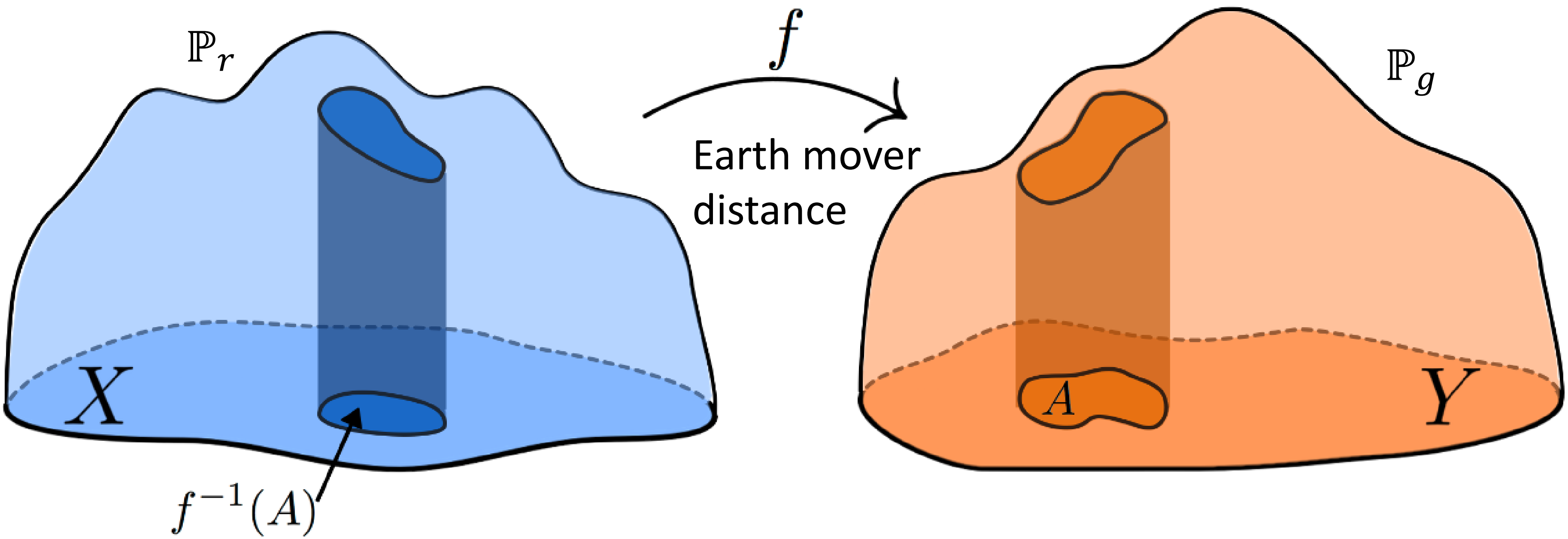


Distance between Probability Distribution

Distributed Primal Form of Wasserstein Distance

Experiments

Probability Distributions and Wasserstein Distance



- Optimal transport problem:

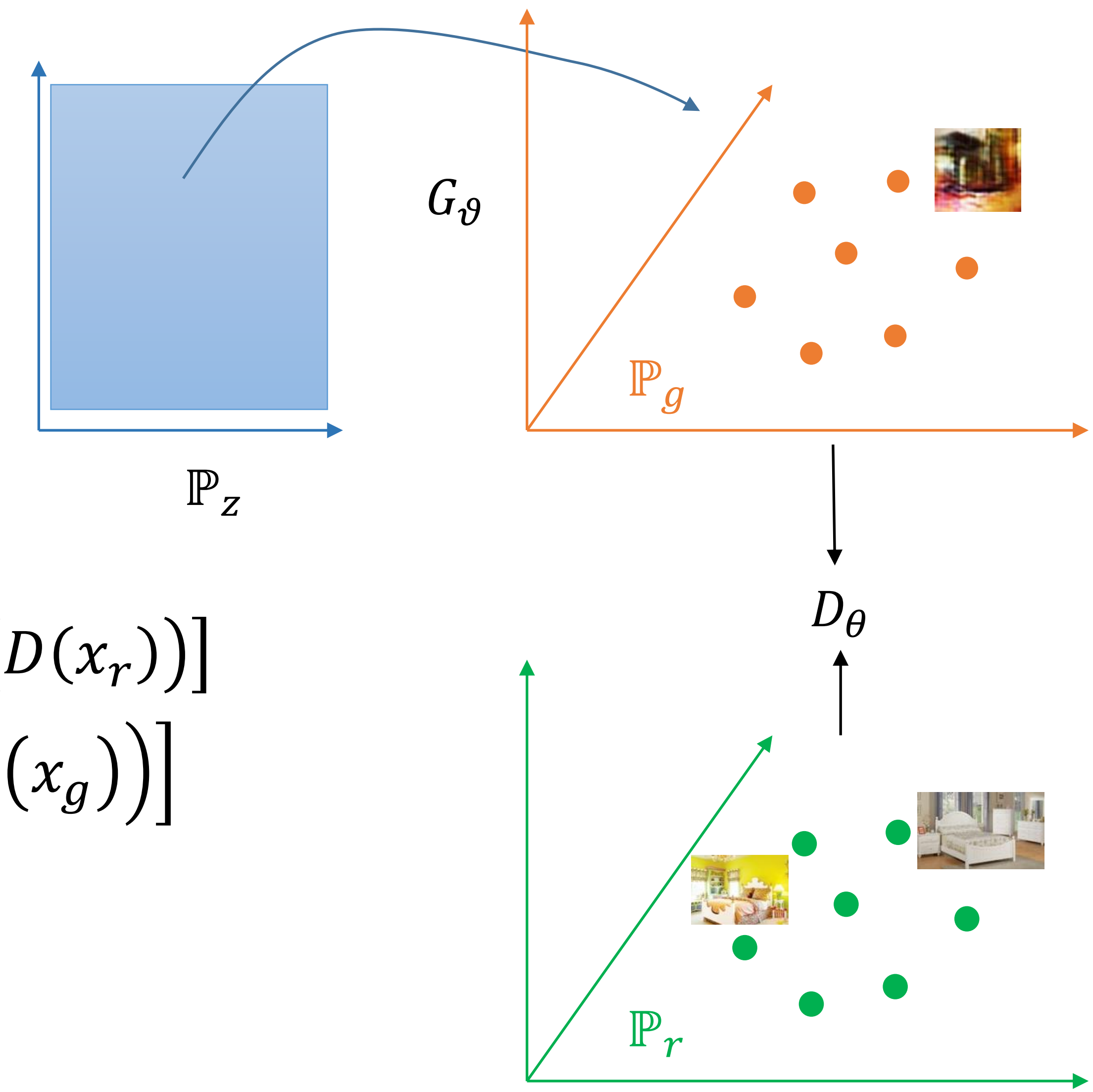
$$W_c(\mathbb{P}_r, \mathbb{P}_g) = \min_{\gamma \in \Pi(\mathbb{P}_r, \mathbb{P}_g)} \int_{X \times Y} c(x_r, x_g) d\gamma(x_r, x_g)$$

- Wasserstein$-p$ distance: $c(\cdot,\cdot) = \|\cdot\|_p$

http://faculty.virginia.edu/rohde/transport/OTCrashCourse.pdf

Generative Adversarial Nets (GANs)

- Generator network:
  - creates plausible data
- Discriminator network
  - distinguish between the generator's fake output and real data sample



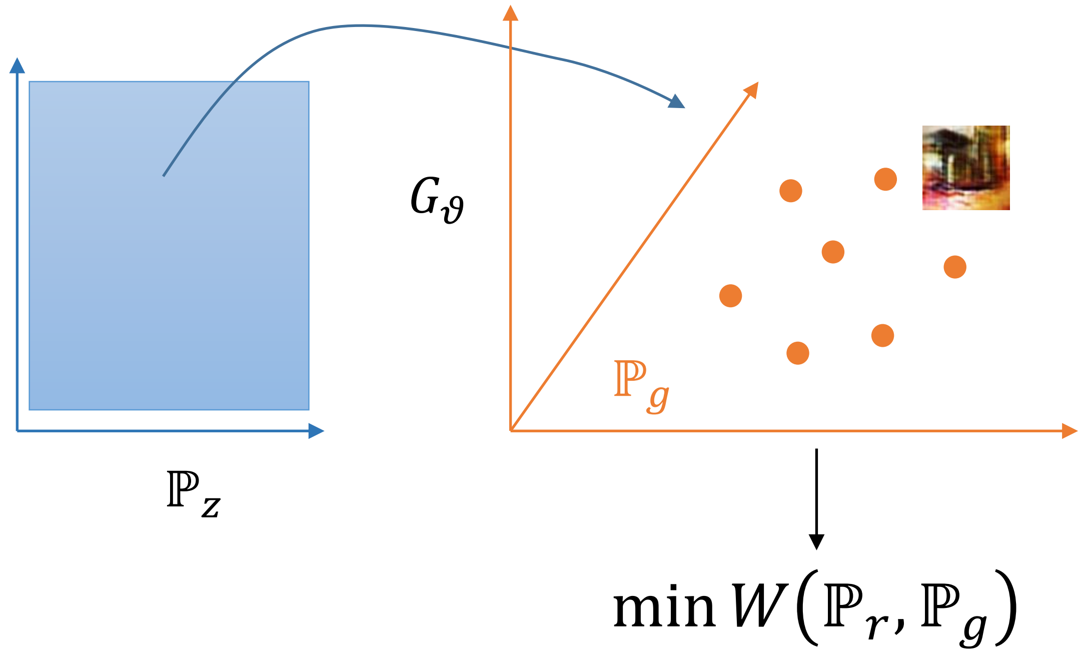
$$\min_G \max_D \mathbb{E}_{x_r \sim \mathbb{P}_r}[\log(D(x_r))] + \mathbb{E}_{x_g \sim \mathbb{P}_g}\left[\log\left(1 - D(x_g)\right)\right]$$

Stability!

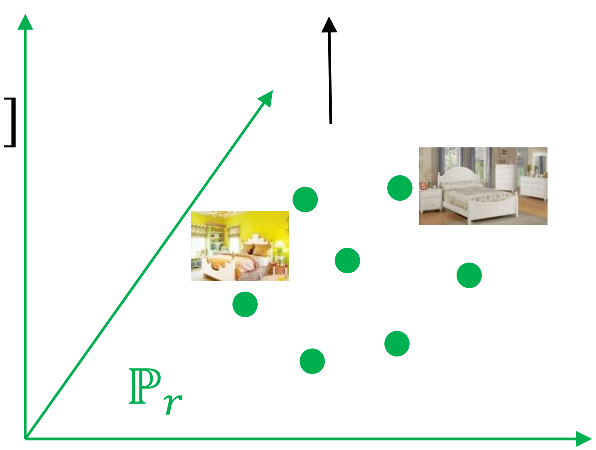
Goodfellow, Ian, et al. "Generative adversarial nets." *Advances in neural information processing systems*. 2014.

Wasserstein Generative Adversarial Nets

- Introduce Wasserstein distance to GAN
- Minimize the distance between real distributions and fake distributions
- Kantorovich-Rubinstein dual form with 1-Lipschitz constraint:



$$W_1(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\boxed{\|D_\theta\|_L < 1}} \mathbb{E}_{x_r \sim \mathbb{P}_r}[D_\theta(x_r)] - \mathbb{E}_{x_g \sim \mathbb{P}_g}[D_\theta(x_g)]$$

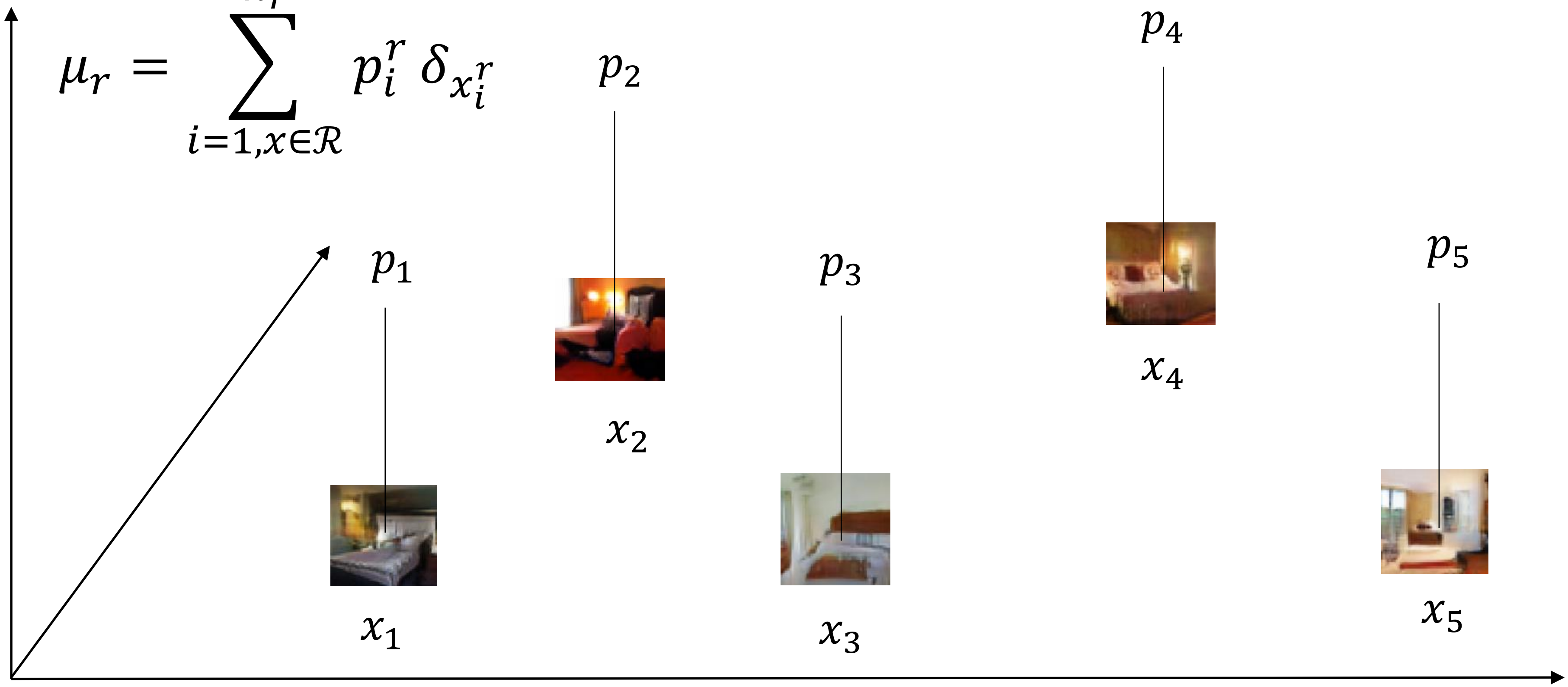


We target primal form!

Arjovsky, Martin, Soumith Chintala, and Léon Bottou. "Wasserstein generative adversarial networks." *International conference on machine learning*. 2017.

Empirical Discrete Distributions

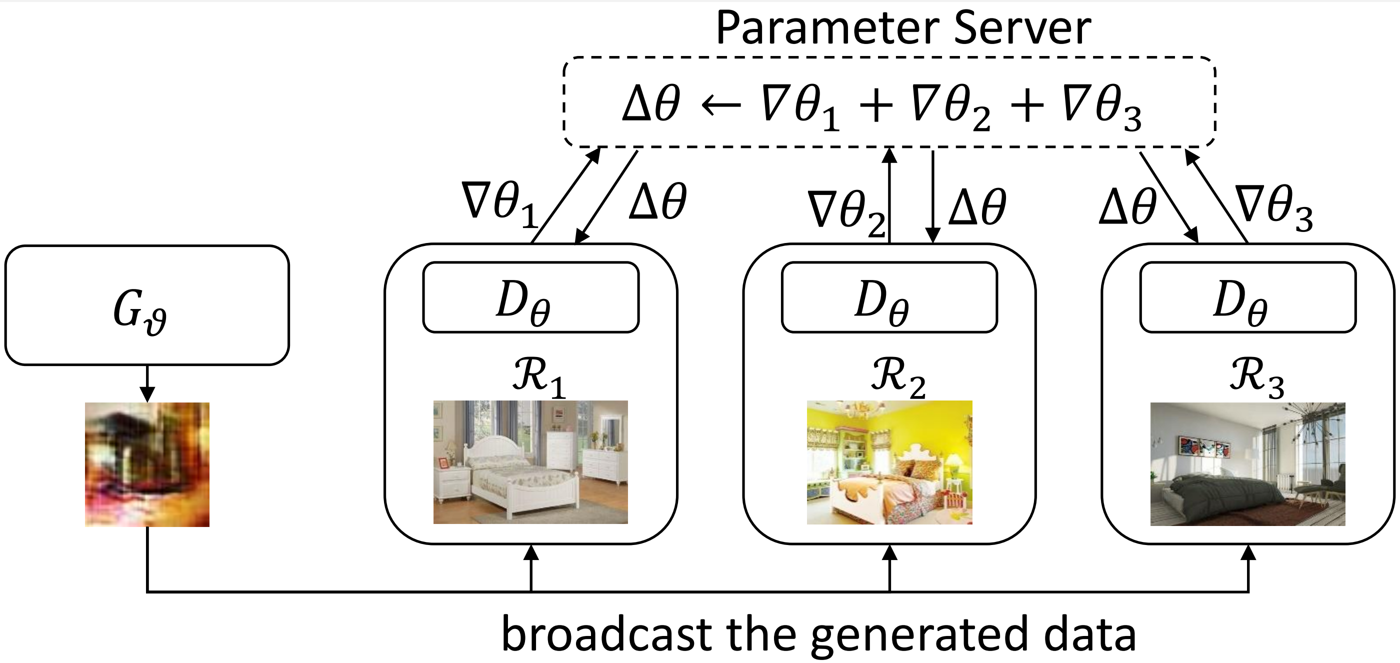
$$\mu_r = \sum_{i=1, x \in \mathcal{R}}^{n_r} p_i^r \, \delta_{x_i^r}$$



Higher accuracy ← More samples ← Large mini-batch

Multiple machines ← Limited GPU memory

Conventional Distributed Approach



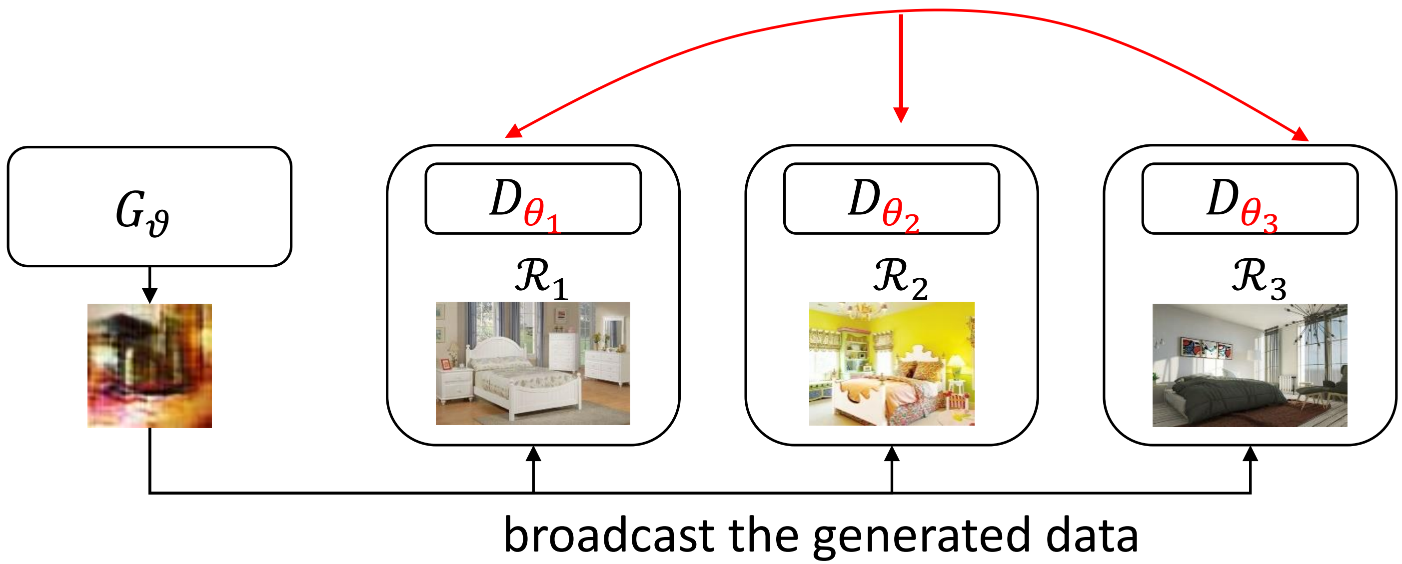
- Distribute supervised real images into multiple machines
- Discriminator is trained collaboratively
- Parameters is synchronized by parameter server

Problems

- Synchronization is expensive
- Synchronization is frequent

Our Solution: Multiple Discriminators

Goal: No heavy communication

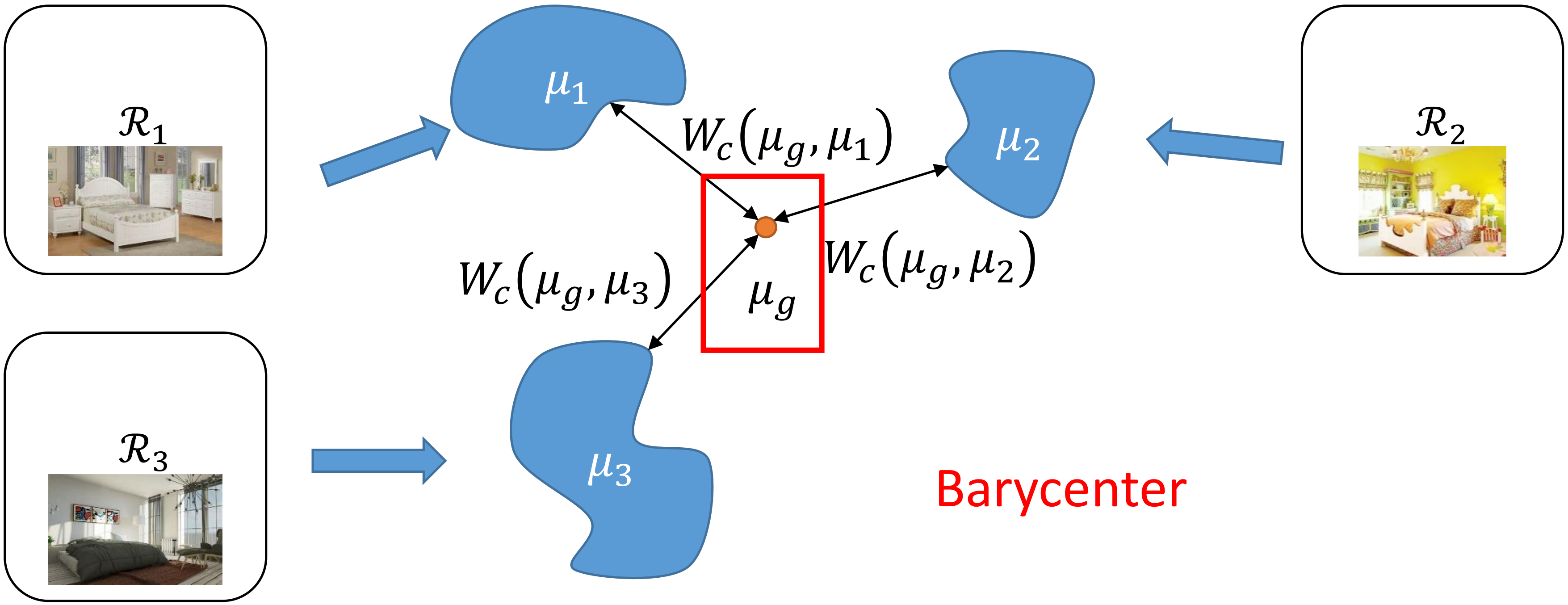


- Generator is more important
- Discriminator could be inconsistent

Challenge

How to provide mathematically sound solution to compute Wasserstein distance?

Wasserstein Barycenter



We attempt to minimize:

$$\mathcal{L}(\mu_g, \{\mu_k^r\}_{k=1}^K) = \frac{1}{K}\sum_{k=1}^{K} W_c(\mu_g, \mu_k^r)$$

Primal Problem of Wasserstein Distance

- Semi-discrete Kantorovich's formulations of the primal problem:

$$W_c(\mu_g, \mu_r) = \max_{\varphi \in \mathbb{R}^{n_g}} \left\{ \sum_{j=1}^{n_g} \varphi_j p_j^g + \int_\Omega \varphi^c(x_r) d\mu_r(x_r) \right\}$$

- $\varphi^c(x_r) = c\left(x_r, x_j^g\right) - \varphi_j$

- Minimize:

$$\mathcal{L}\left(\mu_g, \{\mu_k^r\}_{k=1}^K\right)$$
$$= \max_{\varphi \in \mathbb{R}^{n_g}} \left\{ F(\varphi) = \sum_{j=1}^{n_g} \varphi_j p_j^g + \frac{1}{K} \sum_{k=1}^{K} \int_\Omega \varphi^c(x_r) d\mu_k^r(x_r) \right\}$$

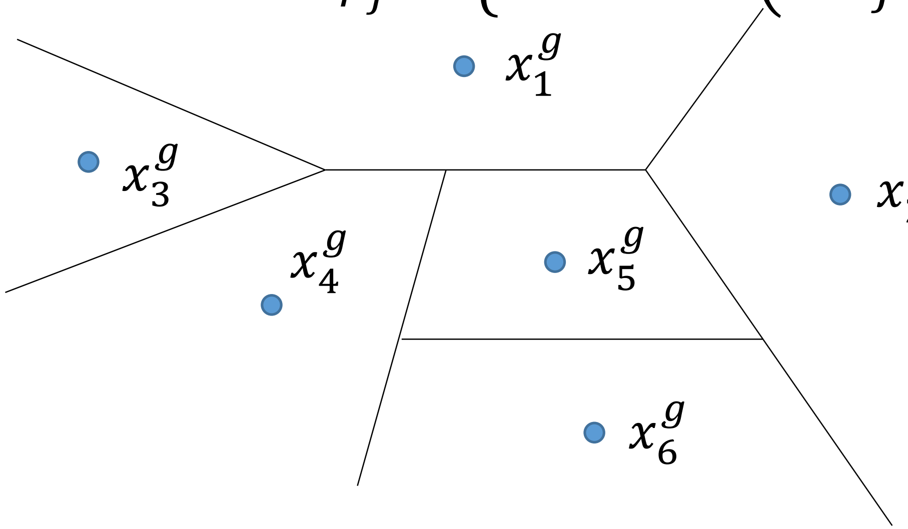
Voronoi Cells

- $F(\varphi)$ is concave and derivative:

$$\frac{\partial F}{\partial \varphi_j} = p_j^g - \frac{1}{K}\sum_{k=1}^{K}\int_{\mathrm{Vor}_{\varphi_j}^k} \mathrm{d}\,\mu_k^r(x_r)$$

- Voronoi cells:

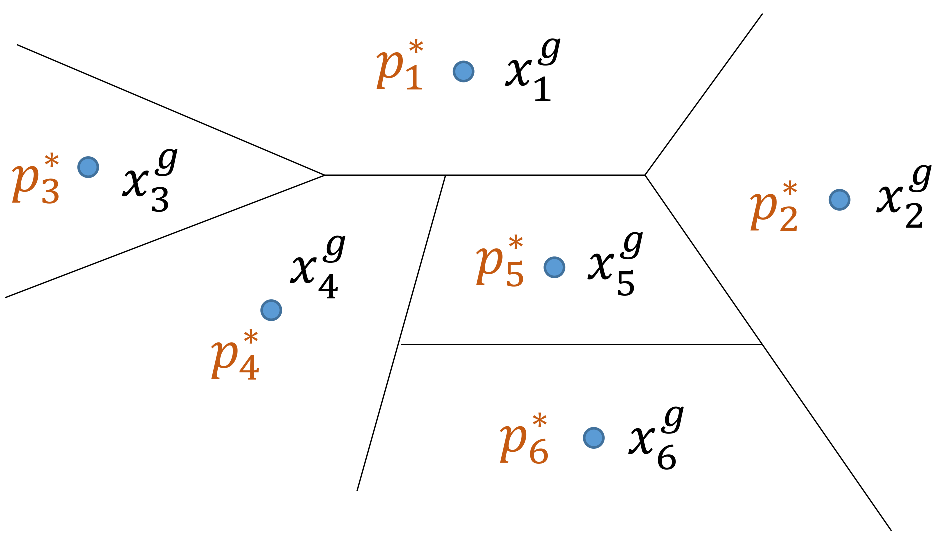
$$\mathrm{Vor}_{\varphi_j} = \left\{x \in \mathcal{R} :\ c\left(x, x_j^g\right) - \varphi_j \leq c\left(x, x_{j'}^g\right) - \varphi_{j'} \forall j'\right\}$$



Santambrogio, Filippo. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Vol. 87. Birkhäuser, 2015.

Optimization

Like Voting!



- When $\frac{\partial F}{\partial \varphi_j} = 0$ and the optimal p_j^* is calculated as:

$$p_j^* = \frac{1}{K}\sum_{k=1}^{K}\int_{\mathrm{Vor}_{\varphi_j}^k} \mathrm{d}\,\mu_k^r(x_r)$$

$$= \frac{1}{K}\sum_{k=1}^{K}\mathbb{E}_{x^r\sim\mu_k^r}\left[I_{x^r\in\mathrm{Vor}_{\varphi_j}^k}\right]$$

$$\approx \frac{1}{K}\sum_{k=1}^{K}\sum_{i=1}^{n_k^r} I_{x_i^k\in\mathrm{Vor}_{\varphi_j}^k}$$

Overall Algorithm: Generator

Algorithm 1: Generator as Master Unit

Input: K: number of worker unit, n_g: batch size in generator, n_r: batch size in worker, T: number of iteration for the estimation of the probability Dirac mass.

Output: G_ϑ: the generator model.

while *ϑ has not converged* **do**

  Sample Gaussian noise $z_1, \ldots, z_{n_g}$ from $\mathcal{N}(0, 1)$

  Generate fake images $\left\{x_j^g; x_j^g = G_\vartheta\left(z_j\right)\right\}_{j=1}^{n_g}$

  $\varphi_j \leftarrow 0\ \forall j \in [1, n_r]$

  $\boxed{\textbf{Broadcast } \{x_j^g\}_{j=1}^{n_g} \text{ to workers}}$

  for *t=1,...,T* **do**

    $\varphi_j \leftarrow \frac{1}{K}\sum_{k=1}^{K} \varphi_j^k\ \forall j \in [1, n_g]$

    Broadcast $\{\varphi_j, p_j^g\}_{j=1}^{n_g}$ to workers

    Receive $\{\varphi_j^k, N_j^k\}_{j=1}^{n_g}$ from workers

    $p_j^g \leftarrow \frac{1}{K}\sum_{k=1}^{K} N_j^k\ \forall j \in [1, n_g]$

  end

  Conduct back-propagation with the loss: $\text{Loss}(\{D_\theta^g(G_\vartheta(z_j))\}_{j=1}^{n_g}, \{p_j^g\})$

end

Overall Algorithm: Discriminator

Algorithm 2: Discriminator as Worker Unit

Input: K: number of worker unit, n_r: batch size in worker unit, η : learning rate for Kantorovich potential, T: number of iterations for the Dirac mass estimation.

for all *worker* $k \in [1, K]$ **do in parallel**

  Sample $\{x_i\}_{i=1}^{n_r}$ a batch from the real dataset $\mathcal{R}_k$

  Receive fake images $\left\{x_j^g\right\}_{j=1}^{n_g}$ from the master unit

  Evaluate cost $c_{ij} = \left\|D_{\theta_k}\left(x_i^r\right) - D_{\theta_k}\left(x_j^g\right)\right\| \ \forall i \in [1, n_r], j \in [1, n_g]$

  for $t = 1, \ldots, T$ **do**

    Receive $\{\varphi_j, p_j^g\}_{j=1}^{n_g}$ from master

    for $j = 1, \ldots, n_g$ **do**

      Compute Vor_{φ_j} from Eq. (10)

      $N_j^k \leftarrow \sum_{i=1}^{n_k^r} I_{x_i^k \in \mathrm{Vor}_{\varphi_j}^k}$

      $\varphi_j^k \leftarrow \varphi_j - \eta N_j^k$

    end

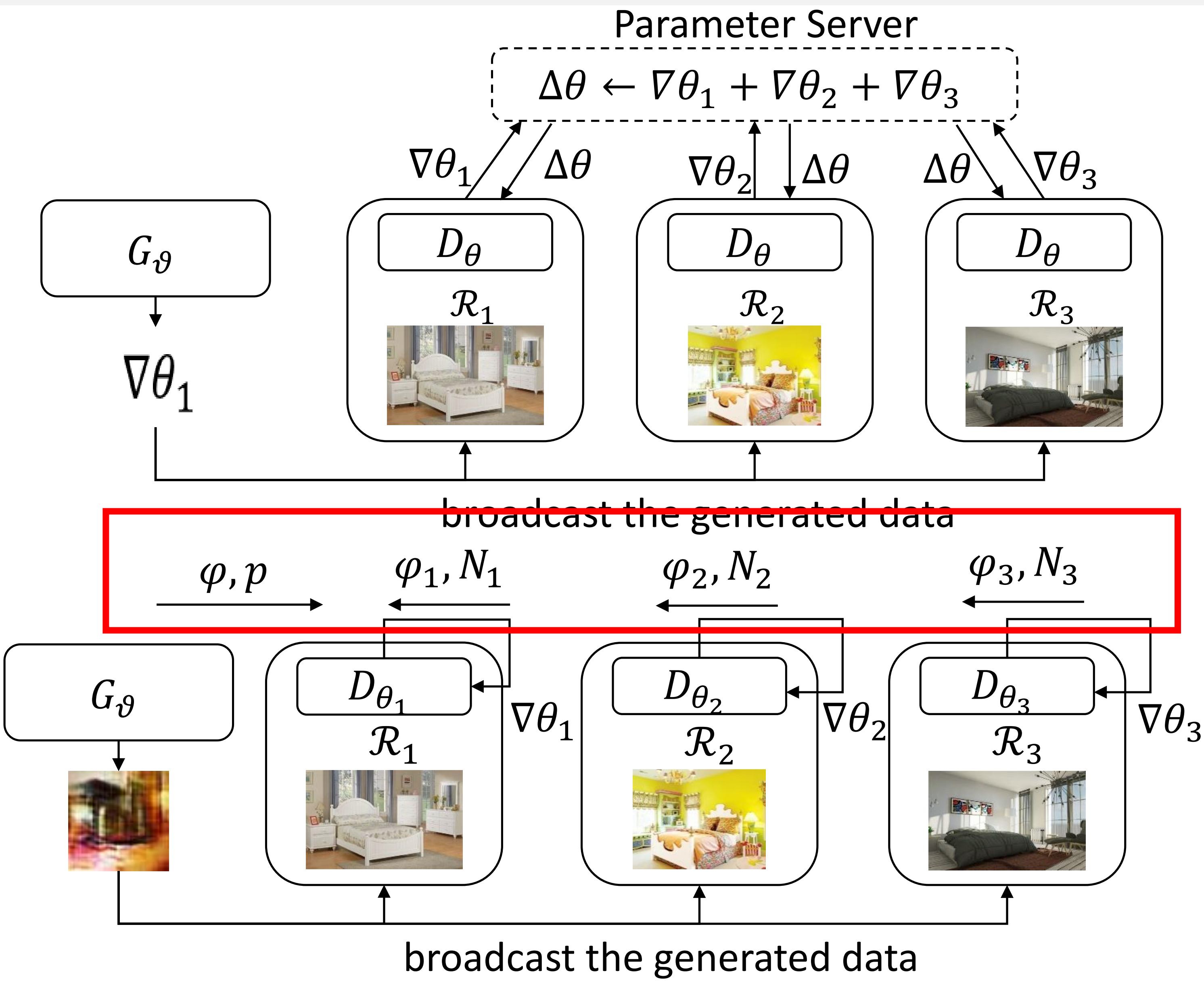
    Send $\{\varphi_j^k, N_j^k\}_{j=1}^{n_g}$ to master

  end

  Conduct back-propagation with the loss: $\mathrm{Loss}(\{D_\theta^k(x_j^g)\}_{j=1}^{n_g}, \{N_j^k\})$

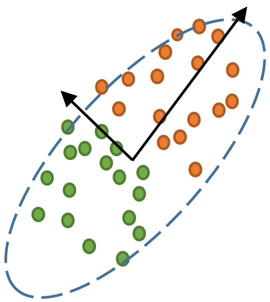
end

Summary

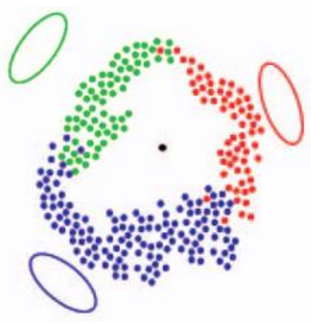


Parameter Server
$\Delta\theta \leftarrow \nabla\theta_1 + \nabla\theta_2 + \nabla\theta_3$
$\nabla\theta_1$
$\Delta\theta$
$\nabla\theta_2$
$\Delta\theta$
$\Delta\theta$
$\nabla\theta_3$
G_ϑ
$\nabla\theta_1$
D_θ
$\mathcal{R}_1$
D_θ
$\mathcal{R}_2$
D_θ
$\mathcal{R}_3$
broadcast the generated data
φ, p
φ_1, N_1
φ_2, N_2
φ_3, N_3
G_ϑ
D_{θ_1}
$\mathcal{R}_1$
$\nabla\theta_1$
D_{θ_2}
$\mathcal{R}_2$
$\nabla\theta_2$
D_{θ_3}
$\mathcal{R}_3$
$\nabla\theta_3$
broadcast the generated data

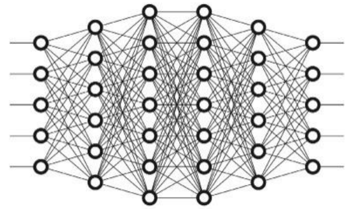
Global



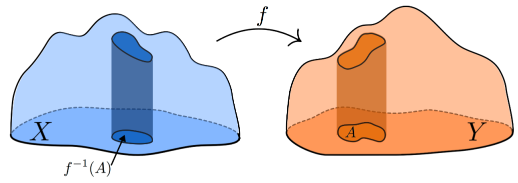
Local



Deep



Probabilistic



Distance between Probability Distribution

Distributed Primal Form of Wasserstein Distance

Experiments

Experimental Setting

- Datasets:

	Image size	Number of images
LSUN with bedrooms	128×128	3,000,000
CIFAR10	32×32	60,000

- Machines:
  - 4 machines with 2 NVIDIA GTX 1080 each
- Network:
  - 101-layer ResNet block for generator and discriminators

Quantitative evaluations

Method	LSUN with bedroom (FID)	CIFAR10 (IS)
WGAN-GP (8 GPUs)	27.3	7.73
Ours (2 GPUs)	23.2	7.12
Ours (4 GPUs)	21.9	7.68
Ours (8 GPUs)	**21.0**	**7.81**

- Smaller Fréchet Inception Distance (FID) is better, larger Inception scores (IS) is better

Comparisons on Bedroom dataset

Wasserstein GAN with gradient penalty

Ours (2 GPUs)

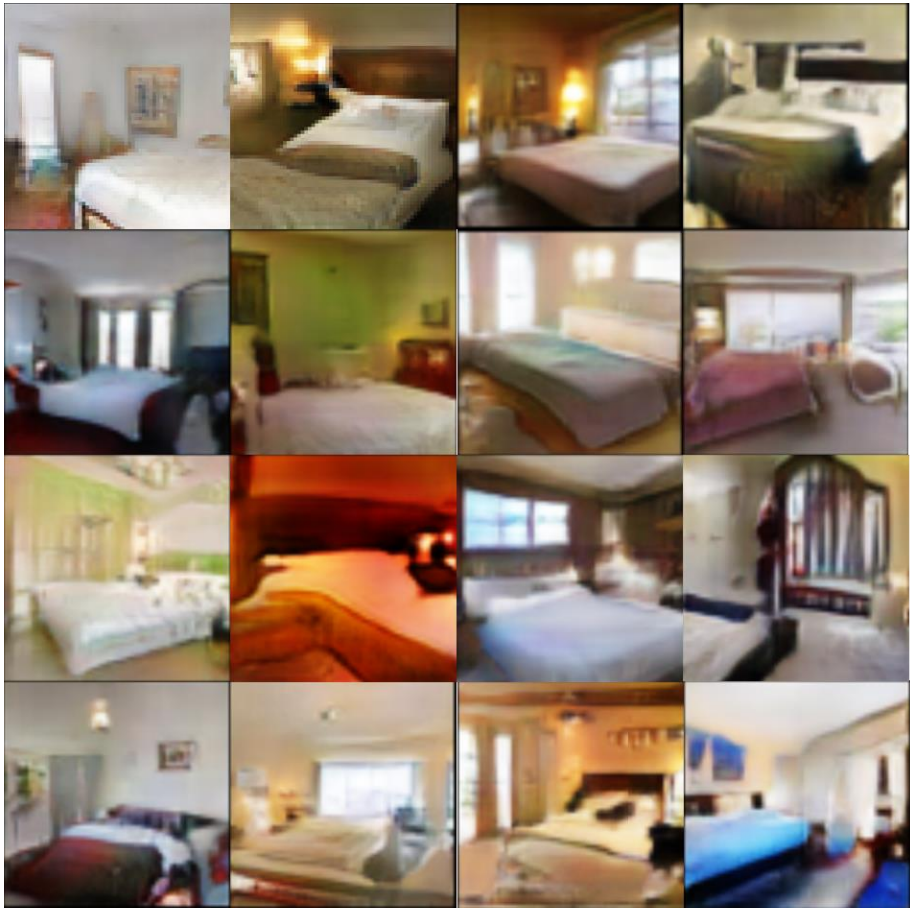
Comparisons on Bedroom dataset

Wasserstein GAN with gradient penalty

Ours (4 GPUs)

Comparisons on Bedroom dataset

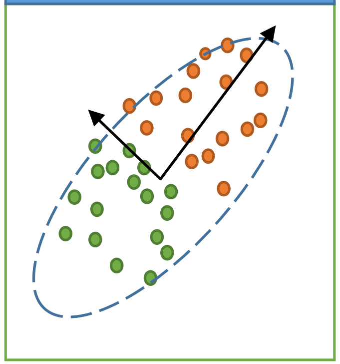
Wasserstein GAN with gradient penalty



Ours (8 GPUs)

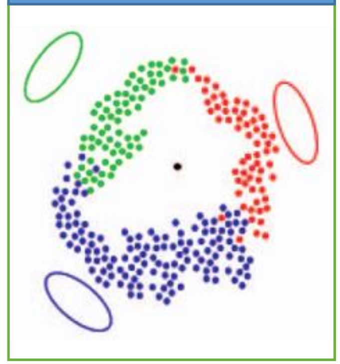
Conclusions

Global



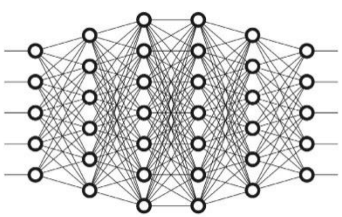
- Semi-synchronous parallel approach
- Theoretical bound
- Efficient distributed implementation

Local



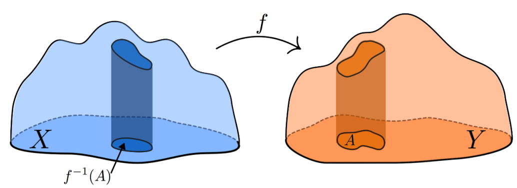
- localized GMML algorithm for the query-independent ranking framework

Deep



- Distributed approach for smart sampling
- Hybrid communication

Probabilistic



- Primal form
- Master-slave distributed solution

Published and In-progress Papers

- In-progress:
  1. Scalable Gromov-Wasserstein Distance for Large-scale Graph Convolutional Networks.
  2. **Yuxin Su**, S. Zhao, X. Chen, X. Shen, Irwin King, Michael Lyu. Powering Graph Convolutional Networks with Wasserstein Barycenter Aggregator on Distributional Node Representation.
  3. X. Chen, **Yuxin Su**, S. Zhao, X. Shen, Michael Lyu, Irwin King. A Simple Orthogonality Regularization to Improve the Training of Deep CNNs.
- Published:
  1. **Yuxin Su**, S. Zhao, X. Chen, Irwin King and Michael Lyu. Parallel Wasserstein Generative Adversarial Nets with Multiple Discriminators. IJCAI 2019
  2. **Yuxin Su**, Michael Lyu, and Irwin King. Communication Efficient Distributed Deep Metric Learning with Hybrid Synchronization. CIKM 2018
  3. **Yuxin Su**, Irwin King, and Michael Lyu. Learning to Rank Using Localized Geometric Mean Metrics. SIGIR 2017
  4. **Yuxin Su**, H. Yang, Irwin King, and Michael Lyu. Distributed Information Theoretic Metric Learning in Apache Spark. IJCNN 2016
  5. **Yuxin Su**, H. Yang, Michael Lyu, Irwin King. Distributed Non-negative Matrix Factorization with Loose Synchronization. WSDM workshop, 2015
  6. J. Hu, H. Yang, **Yuxin Su**, Michael Lyu, Irwin King. Accelerated Information-Theoretic Metric Learning. WSDM workshop 2015
  7. H. Yang, G. Ling, **Yuxin Su**, Michael R. Lyu, and Irwin King. Boosting response aware model-based collaborative filtering. TKDE 2015

Thank You

Appendix

Optimization for Distance Learning

$$\min_{M} L(M, \mathcal{S}, \mathcal{D}, \mathcal{R}) + \lambda \cdot \mathrm{Reg}\,(M)$$

- L is a loss function associated with training constraints
- λ is a regularization parameter
- $\mathrm{Reg}(M)$ is some regularizer term

Global Distance Learning

Appendix

Multivariate Gaussian Distribution

- Probability density function:

$$p(x; A) = \frac{1}{Z}\exp(-\frac{1}{2}d_A(x, \mu))$$

- Minimizing difference relative entropy:

$$\begin{aligned} \min \quad & KL(p(x; A_0)||p(x; A)) \\ s.t. \quad & d_A(x_i, x_j) \leq u \quad (i, j) \in S \\ & d_A(x_i, x_j) \geq l \quad (i, j) \in D \end{aligned}$$

Where

$$KL(p(x|\mu_0, A_0)||p(x|\mu, A)) = \int p(x|\mu_0, A_0)\log\frac{p(x|\mu_0, A_0)}{p(x|\mu, A)}dx$$

Bregman Divergence

- KL divergence:

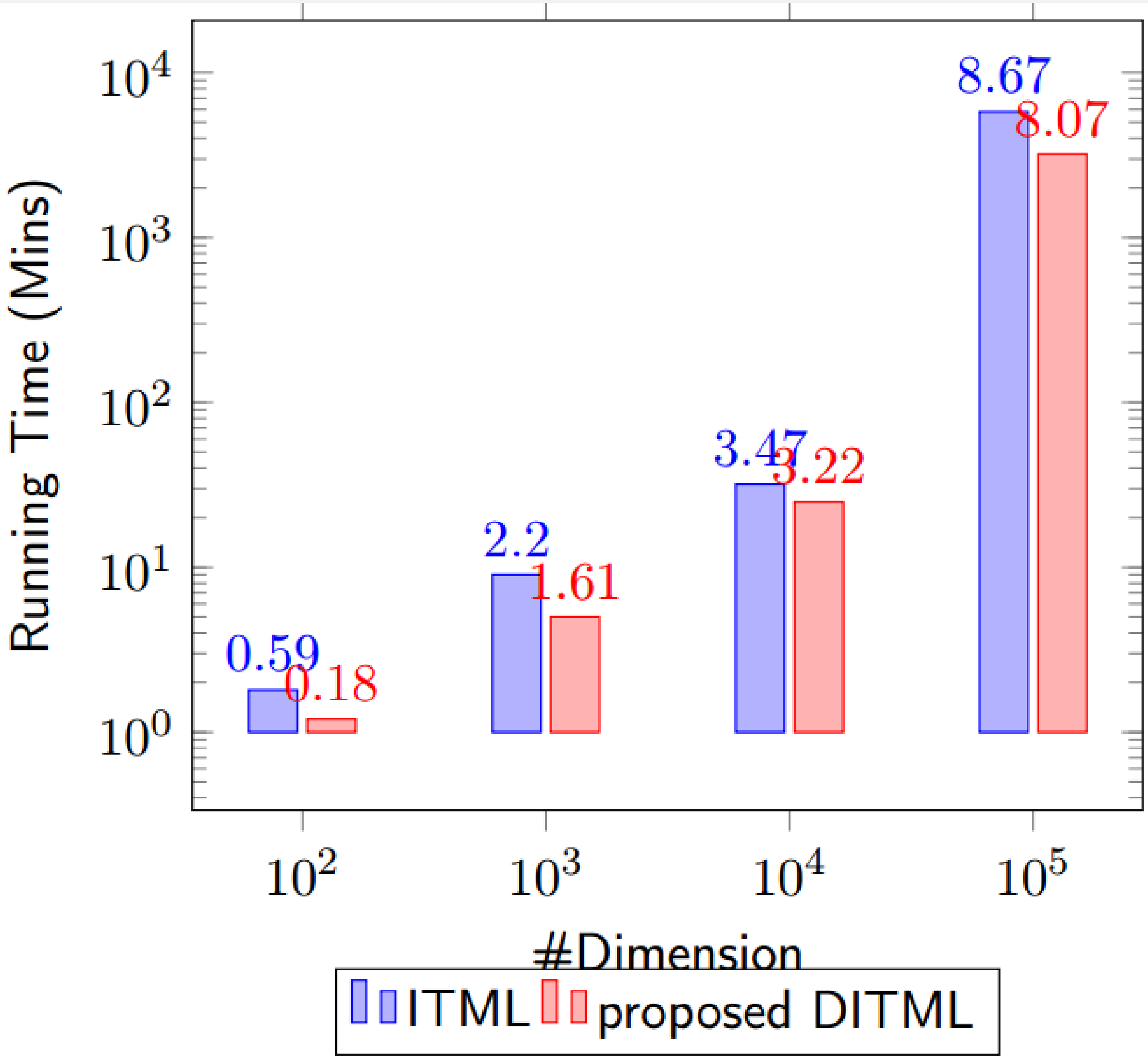
$$KL(p(x|\mu_0, A_0)||p(x|\mu, A) = \frac{1}{2}D_\phi(A, A_0) + \frac{1}{2}d_{\Sigma^{-1}}(\mu_0, \mu)$$

- Bregman divergence:

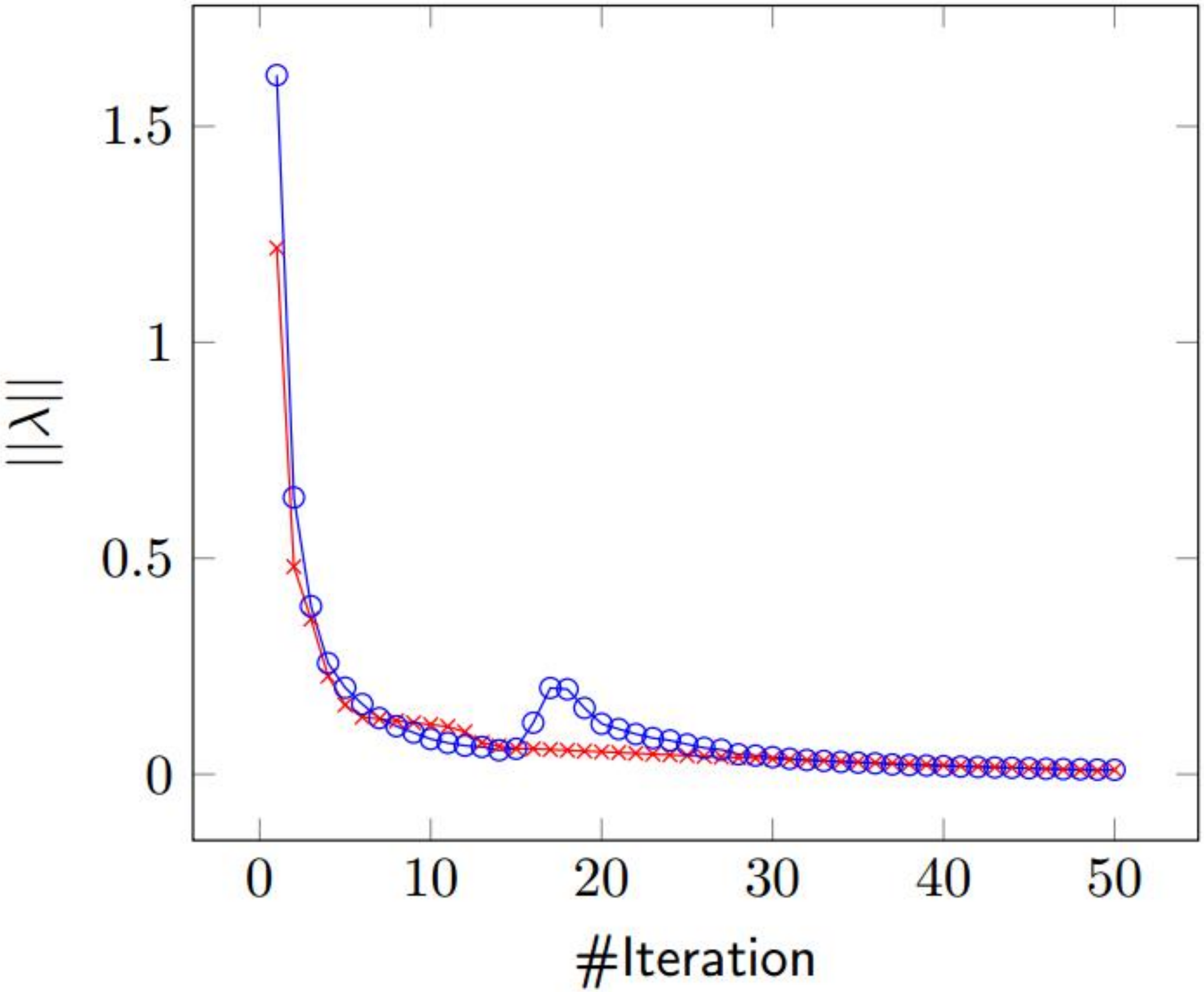
$$D_\phi(A, A_0) = \phi(A) - \phi(A_0) - \mathrm{tr}\big(\nabla\phi(A_0)^T(A - A_0)\big)$$

where $\phi(A) = -\mathrm{logdet}(A)$

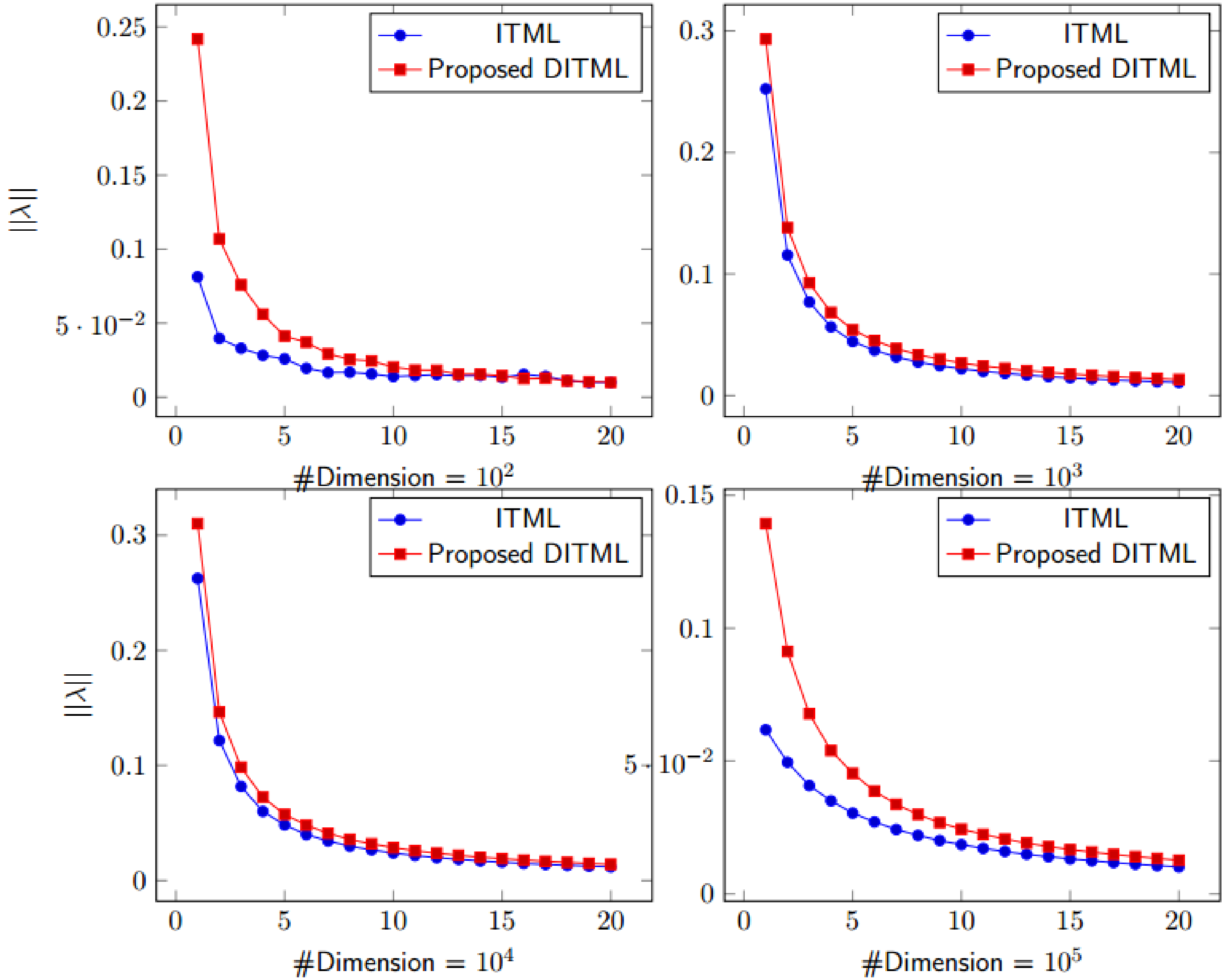
Performance on Running Time



Performance on Accuracy



Performance on Accuracy



Local Distance Learning

Appendix

Experiments: Questions

Correctness	• Is the proposed method a correct extension to the global GMML?
Accuracy	• Does our solution outperform any state-of-the-art LtR algorithm on accuracy?
Scalability	• Does our solution enjoy high computational efficiency and good scalability for scaled datasets?

Datasets

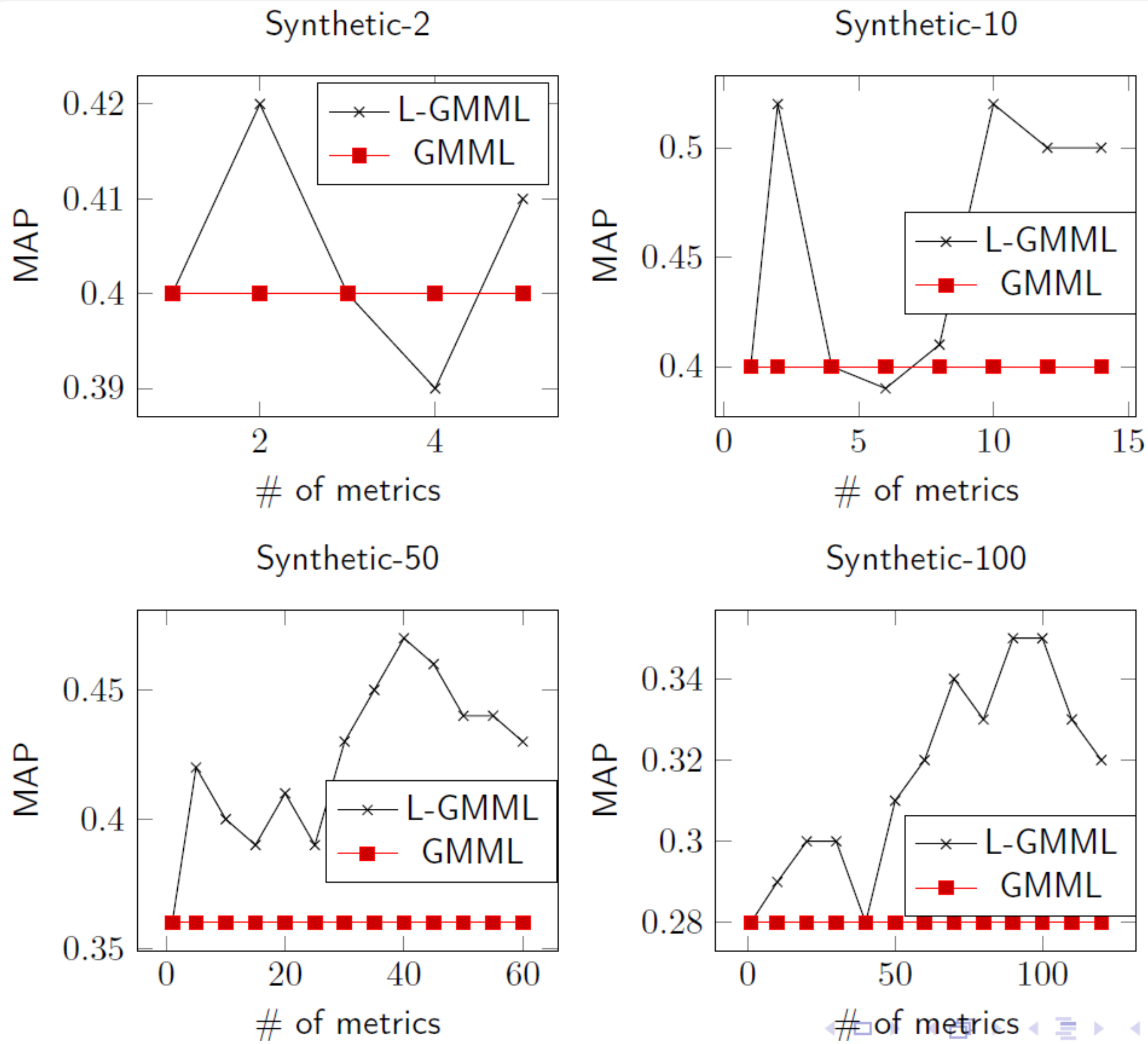
Query-dependent Dataset: CAL10K

	# of Features	# of Songs
Audio	1,024	5,419
Lyrics-128	128	2000
Lyrics-256	256	2000

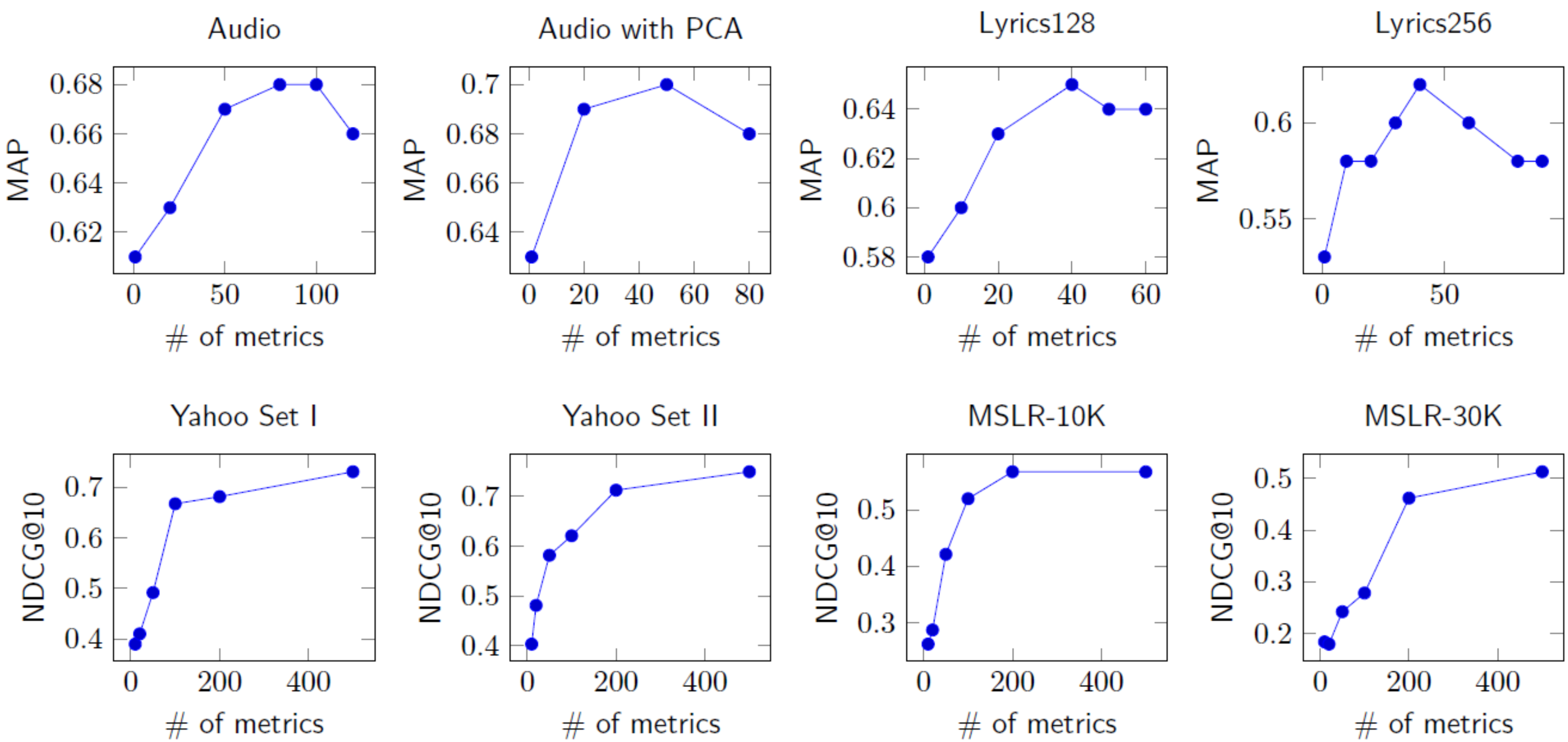
Query-independent Datasets (Query-document pairs)

Name	# of Queries		# of Doc.		Rel. Levels	# of Features
	Train	Test	Train	Test		
Yahoo! Set I	19,944	6,983	473,134	165,660	5	519
Yahoo! Set II	1,266	3,798	34,815	103,174	5	596
MSLR-WEB10K	6,000	2,000	723,412	235,259	5	136
MSLR-WEB30K	31,531	6,306	3,771k	753k	5	136

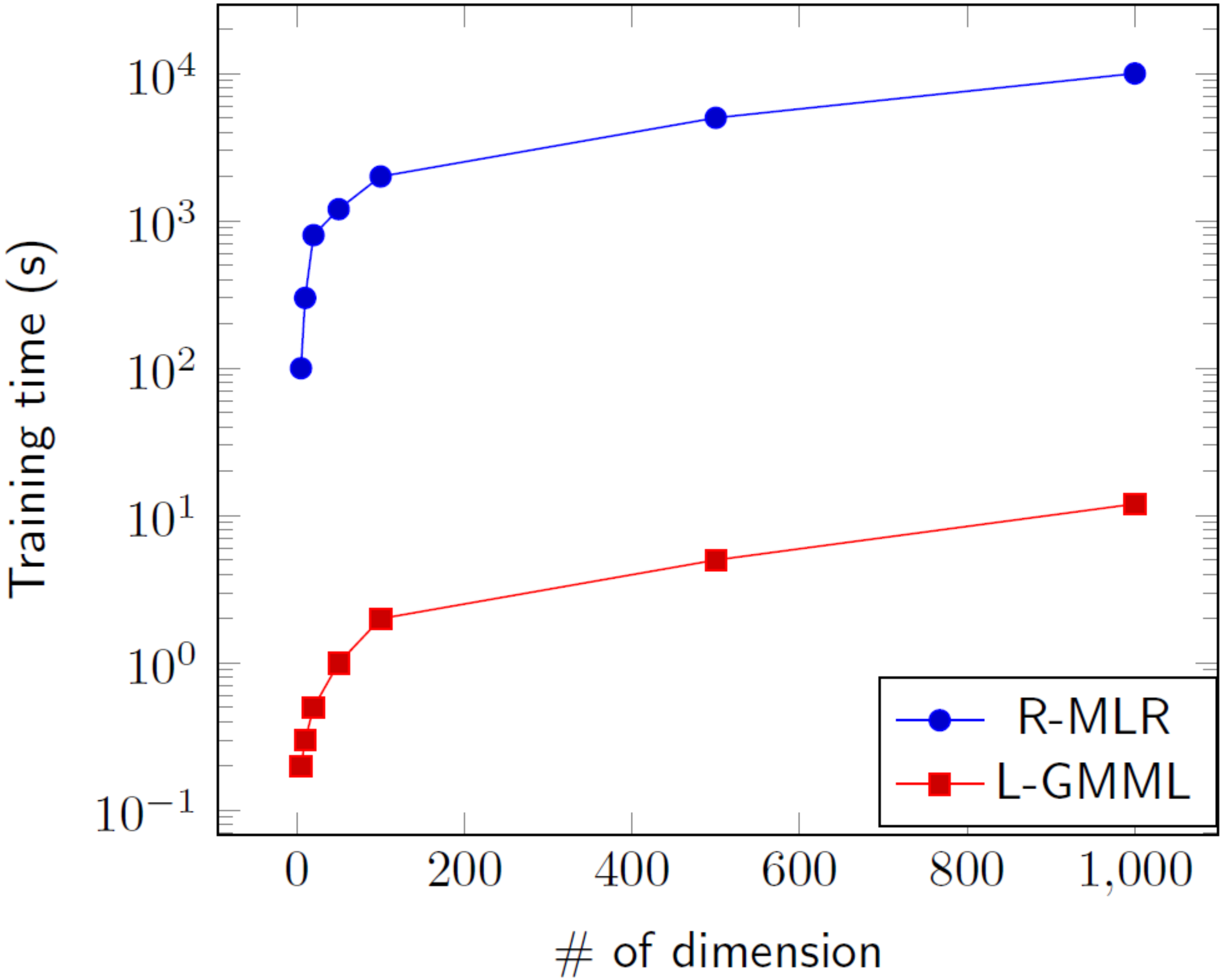
Global GMML vs Local GMML



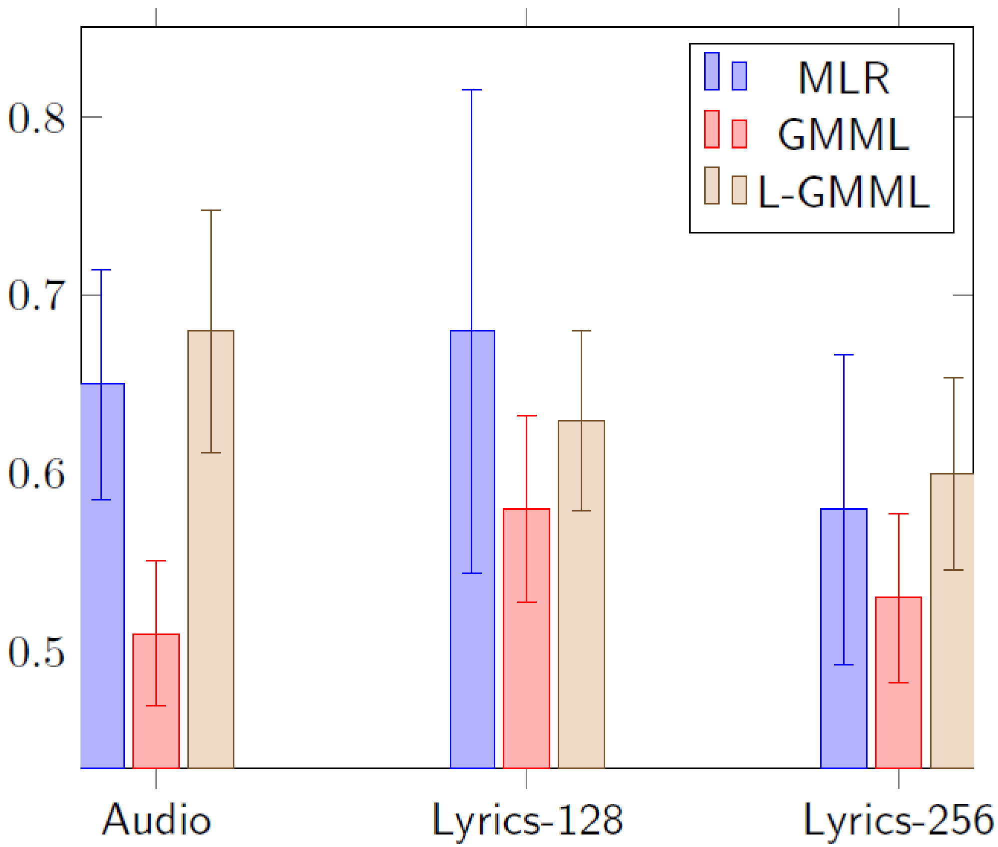
Number of Local Metrics



Scalability: Training Time of L-GMML on Different Scaled Synthetic Datasets



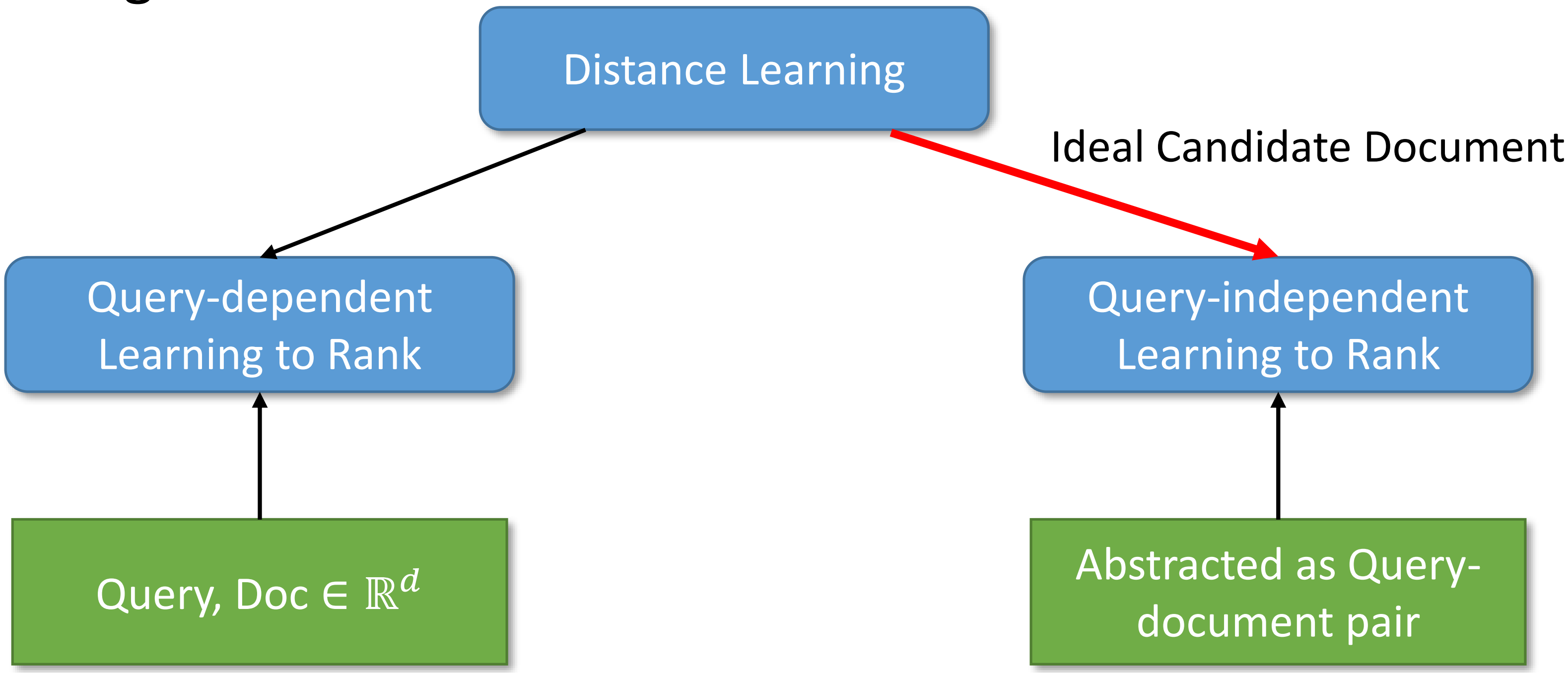
Comparison with R-MLR [Lim et al., ICML13]



Time (s)	R-MLR	L-GMML
Audio	N/A	38
Audio with PCA	607	4.7
Lyrics-128	302	2.6
Lyrics-256	1241	7.8

Conclusions

- Developed a localized GMML algorithm for the query-independent ranking framework

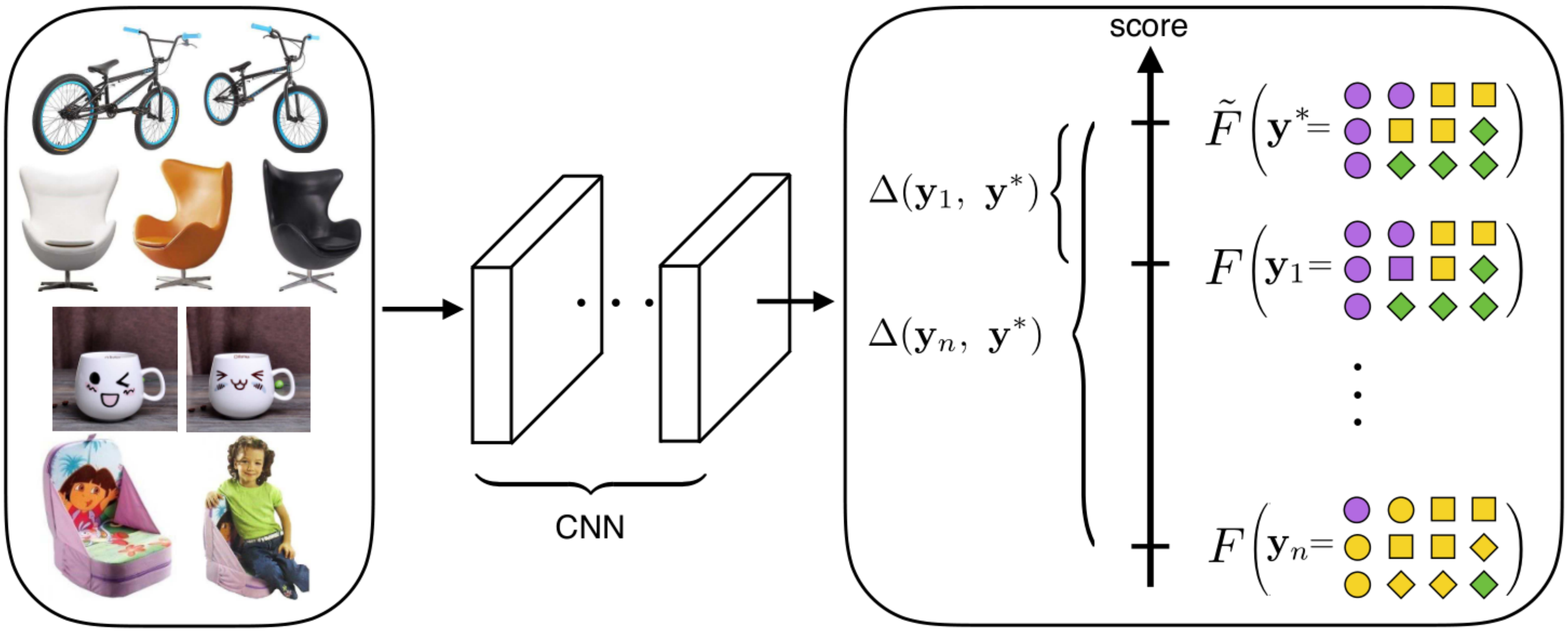


- Experiments show proposed method outperforms the state-of-the-art

Deep Distance Learning

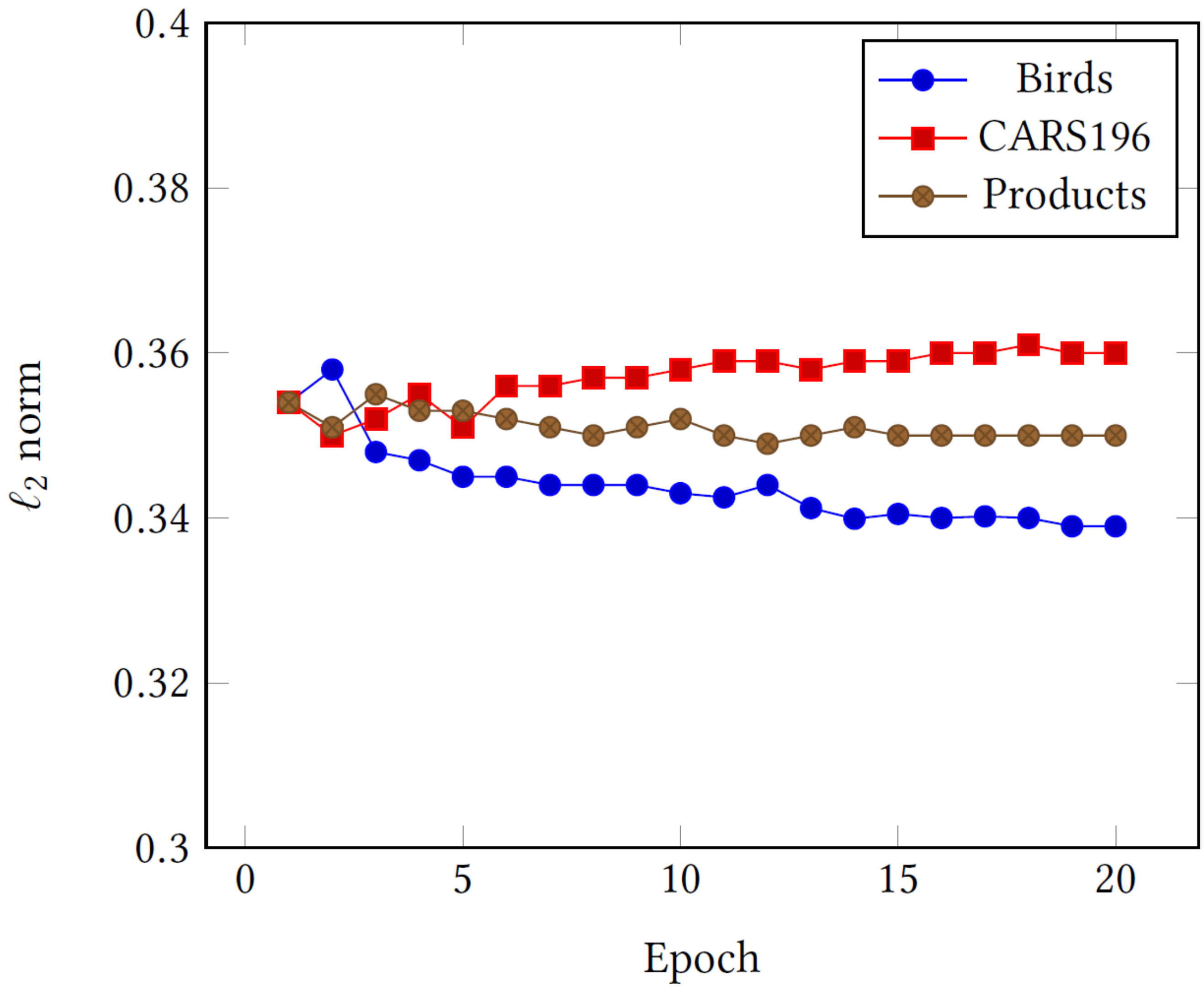
Appendix

Facility Location (NMI-based) [Song et al, CVPR'17]



Learn to rank the clustering score

The Average of the ℓ_2 Norm of All Parameters in CNN

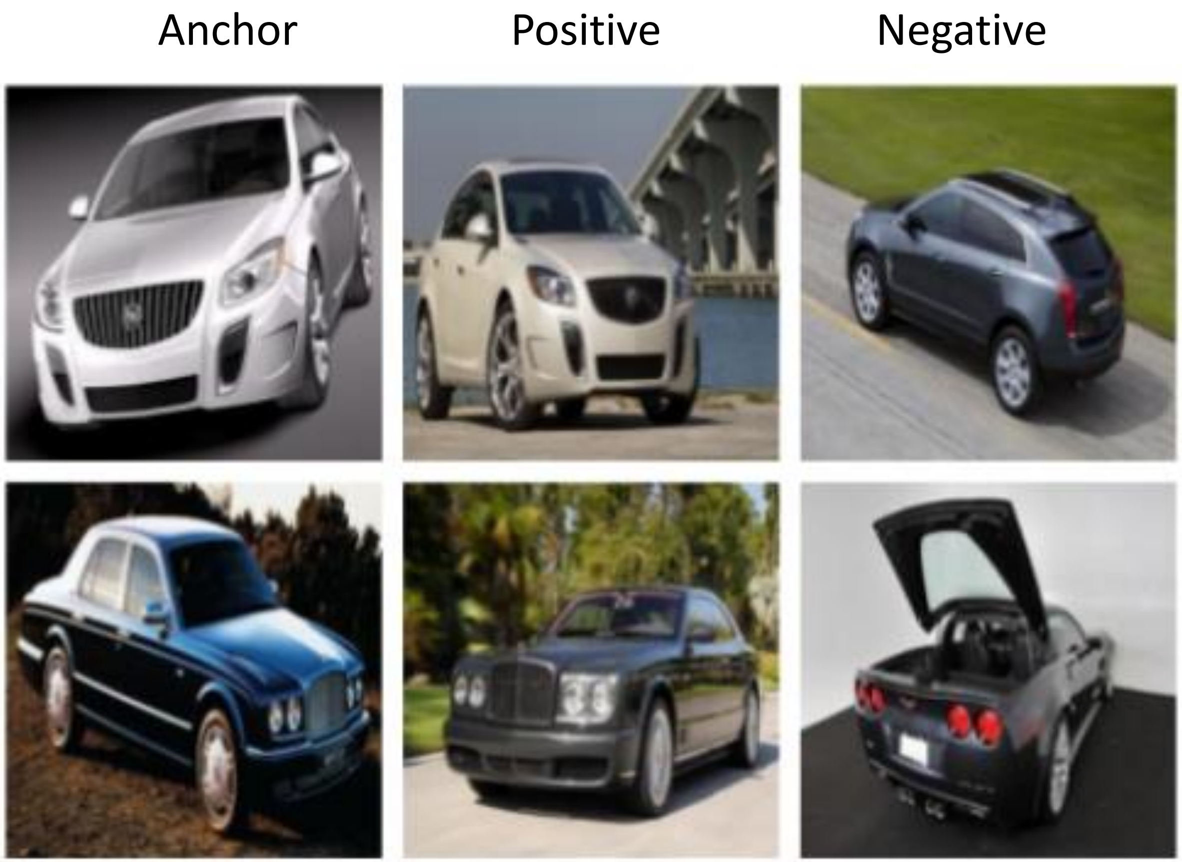


Datasets: CUB-200-2011

Anchor Positive Negative

11,788 images of birds from 200 different categories

Datasets: Cars196 [Krause et al, CVPR'13]



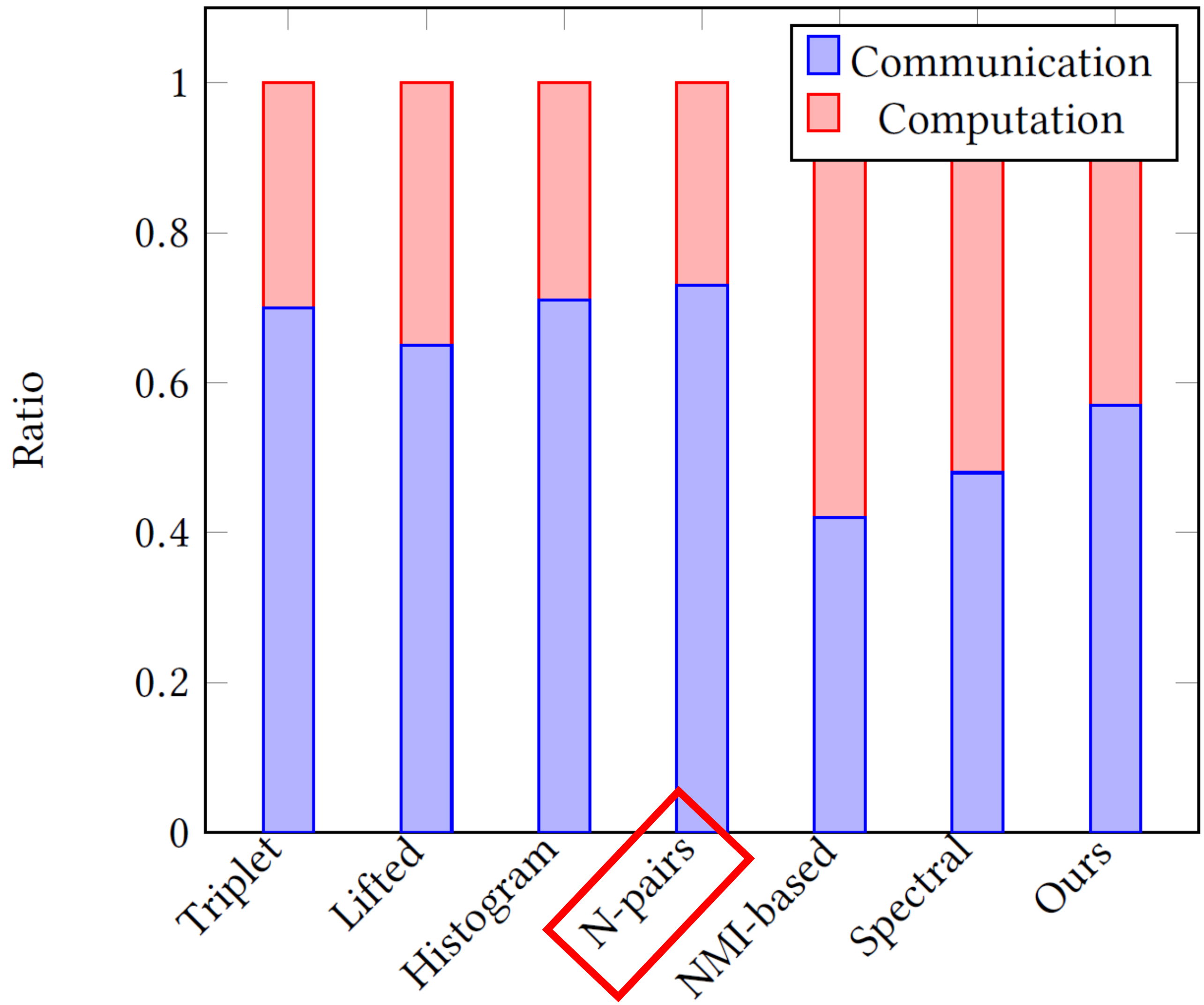
16,185 images of cars from 196 categories

Datasets: Stanford Online Products [Song et al, CVPR'16]

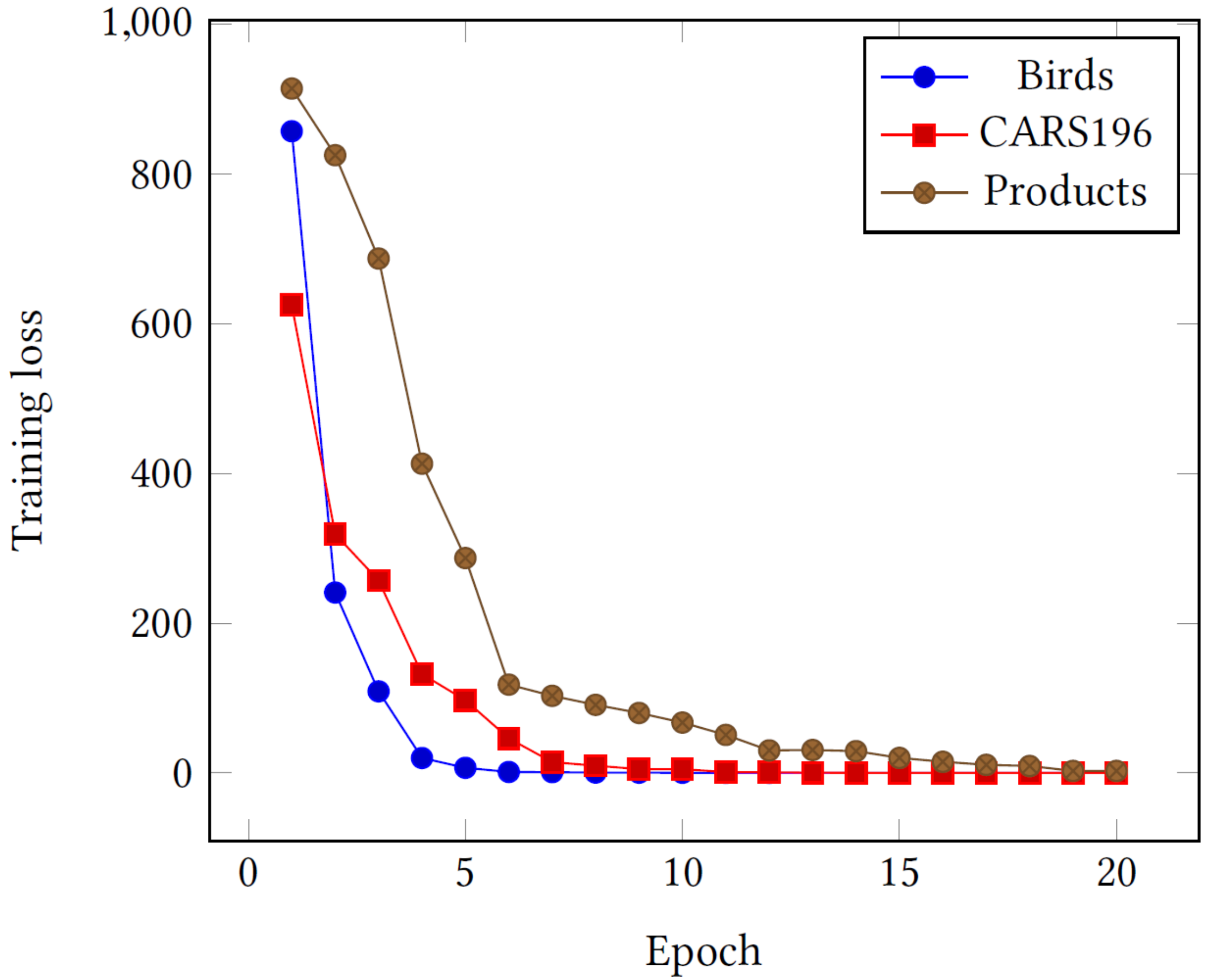
Anchor Positive Negative

120,253 images from 22,634 categories

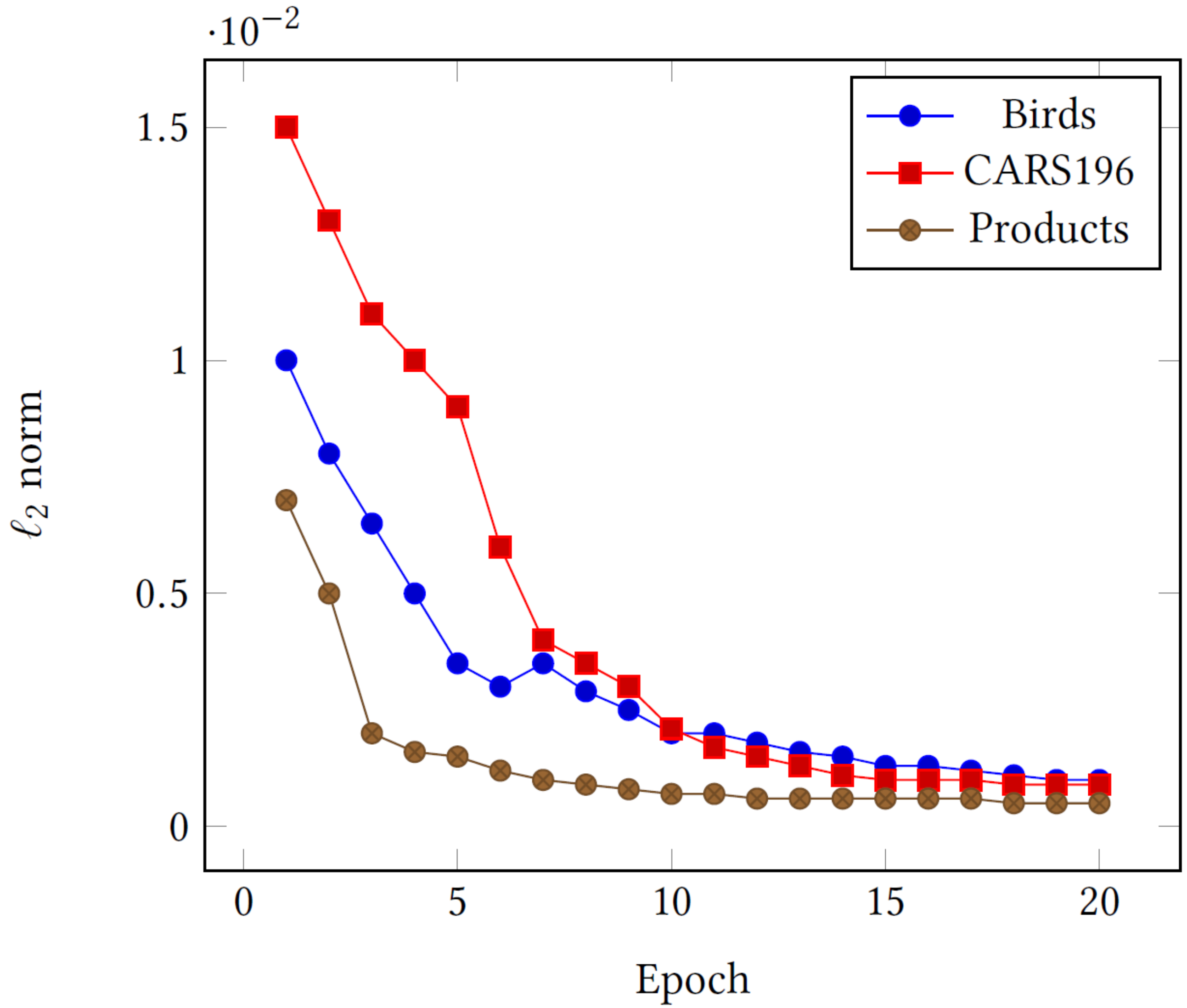
Computation Time vs Communication Time



The Training Loss of the Averaged DML model in proposed framework with 4 machines



The Average of $\left\|C_i - C_{\text{Right(i)}}\right\|_2$



Evaluation on CUB-200-2011

Method	Time (s) / epoch	NMI	Recall@1	Recall@5	Recall@10
Triplet semi-hard [CVPR'15]	424	56.12	40.46	58.15	69.28
Lifted struct [CVPR'16]	536	56.30	43.24	66.73	79.61
Histogram [NIPS'16]	520	-	49.34	68.61	80.58
N-pairs [NIPS'16]	413	58.87	44.29	67.26	79.18
NMI-based [CVPR'17]	617	60.19	48.38	72.47	82.25
Spectral [ICML'17]	702	58.13	50.28	76.80	**85.79**
Ours (2 machines)	378	61.19	**52.45**	77.08	84.24
Ours (4 machines)	**234**	**61.56**	52.40	**77.19**	85.07

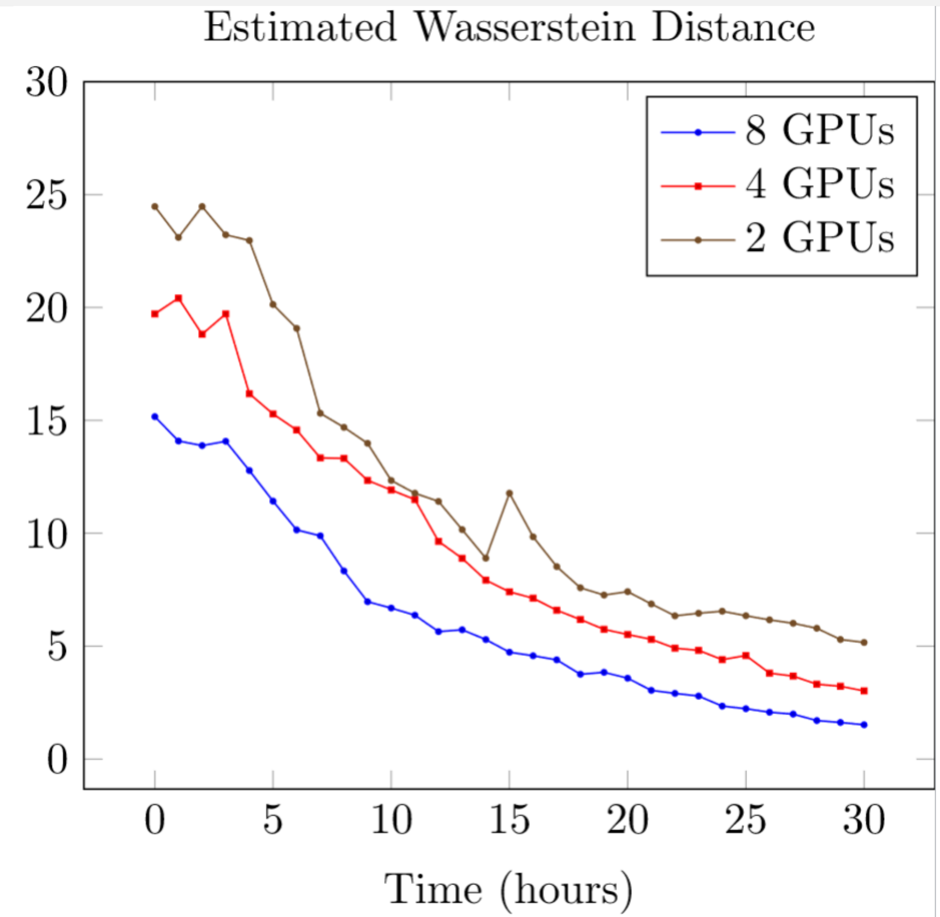
Evaluation on Stanford Online Products

Method	Time (s) / epoch	NMI	Recall@1	Recall@5	Recall@10
Triplet semi-hard [CVPR'15]	4,097	88.38	67.12	77.97	82.28
Lifted struct [CVPR'16]	3,814	88.19	65.50	78.23	81.08
Histogram [NIPS'16]	3,856	-	65.55	78.73	81.56
N-pairs [NIPS'16]	3,701	89.01	67.12	**79.15**	84.09
NMI-based [CVPR'17]	6,238	**90.27**	66.98	77.06	82.15
Spectral [ICML'17]	5,713	87.38	66.09	78.98	83.12
Ours (2 machines)	3,028	89.77	**68.02**	78.34	**85.45**
Ours (4 machines)	**2,356**	89.56	67.79	78.49	84.73

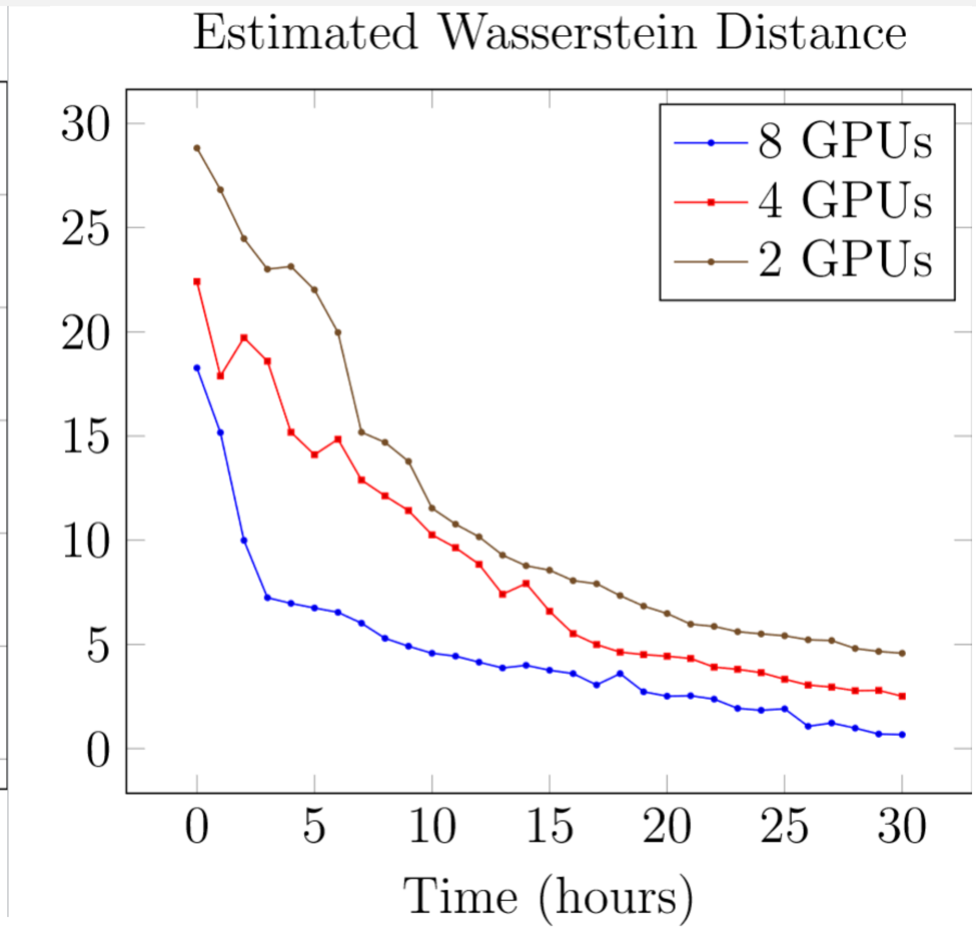
Distance between Probability Distribution

Appendix

Convergence Speed

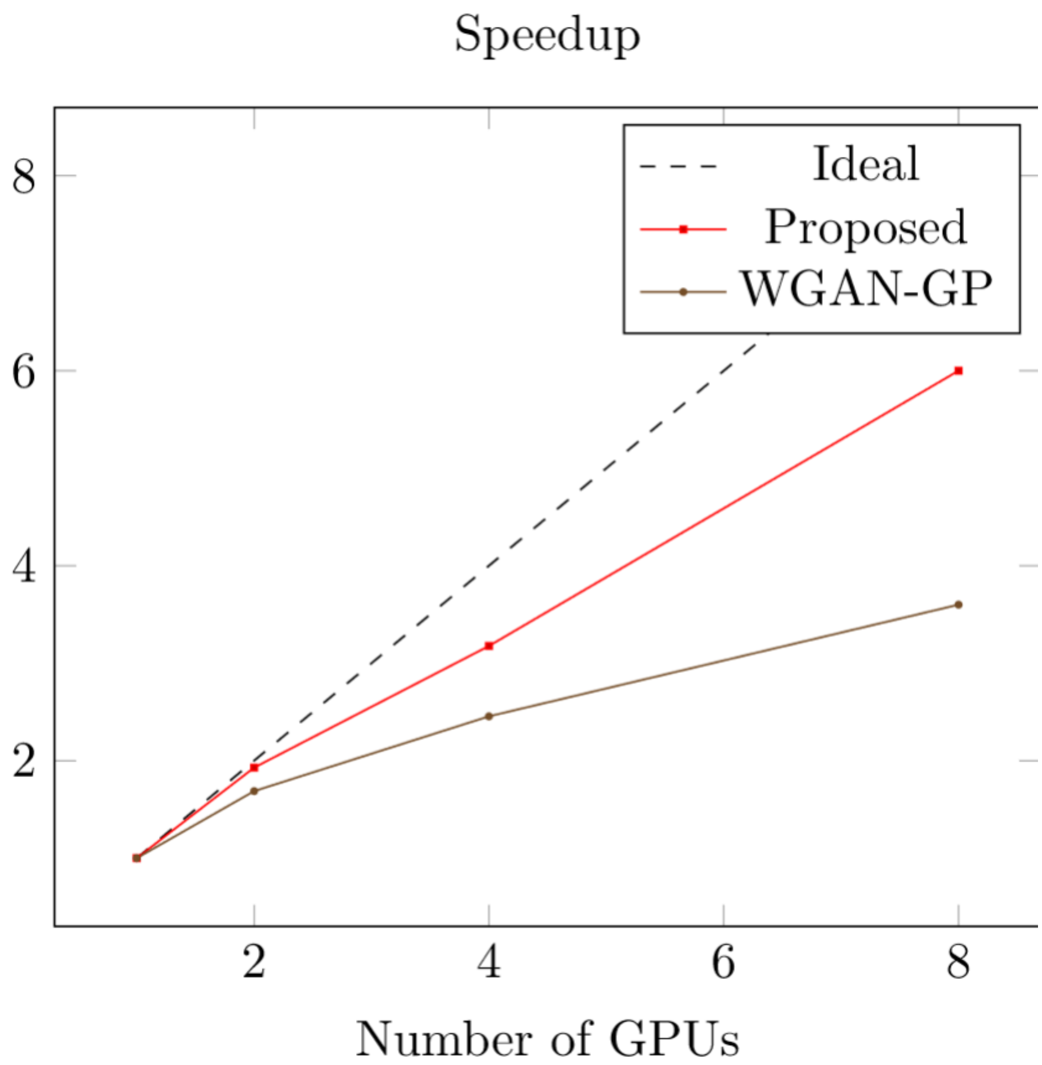


Wasserstein GAN with gradient penalty



Ours

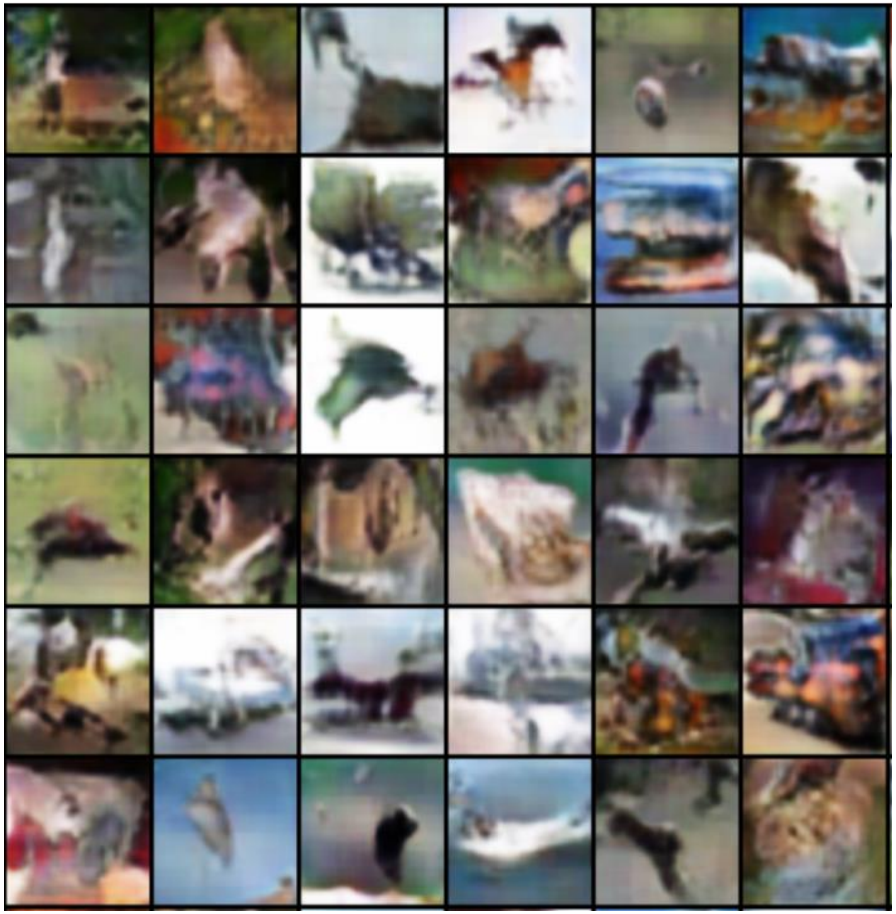
Speedup



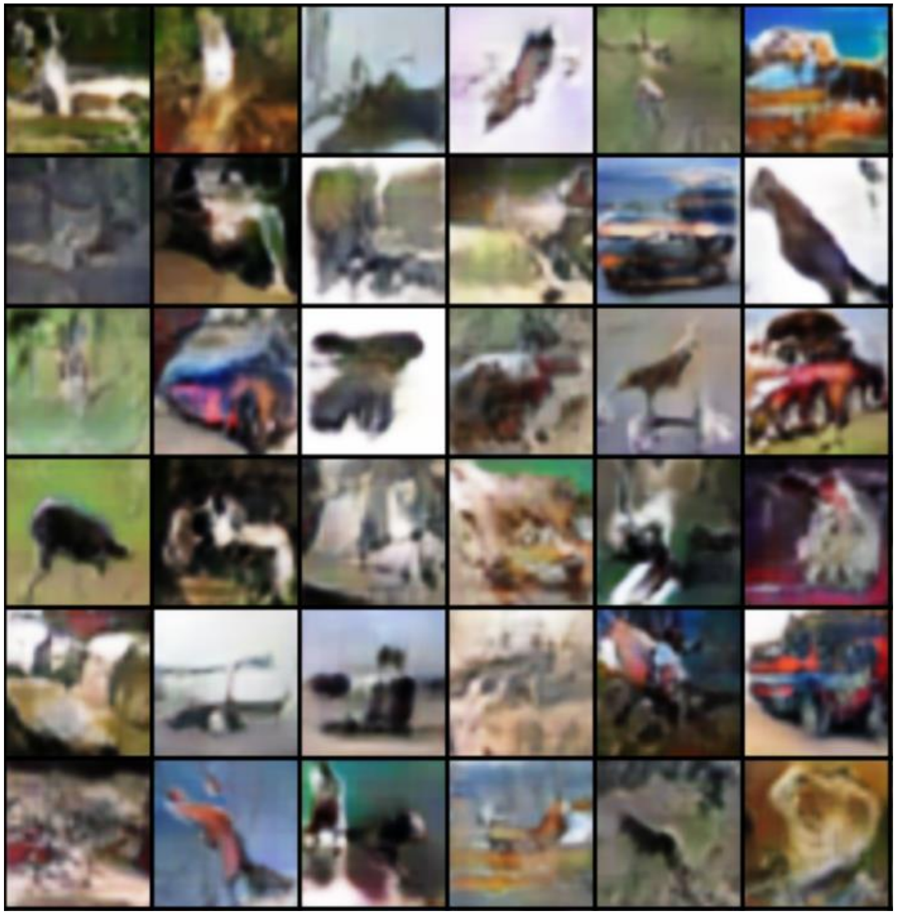
$$\text{Speedup}(N) = \frac{\text{The execution time of one unit}}{\text{The execution time of } N \text{ units}}$$

- To the first checkpoint where Wasserstein distance is less than 5.

Comparisons on CIFAR10 dataset



Wasserstein GAN with gradient penalty



Ours (8 GPUs)

Quantitative evaluations

Method	LSUN with bedroom (FID)	CIFAR10 (IS)
WGAN-GP (8 GPUs)	27.3	7.73
Ours (2 GPUs)	23.2	7.12
Ours (4 GPUs)	21.9	7.68
Ours (8 GPUs)	**21.0**	**7.81**

- Smaller FID is better, larger IS is better

Conclusion

- We introduce a novel parallel architecture to speedup Wasserstein GAN.
- We develop an efficient stochastic algorithm to approximate the Wasserstein distance with higher accuracy.