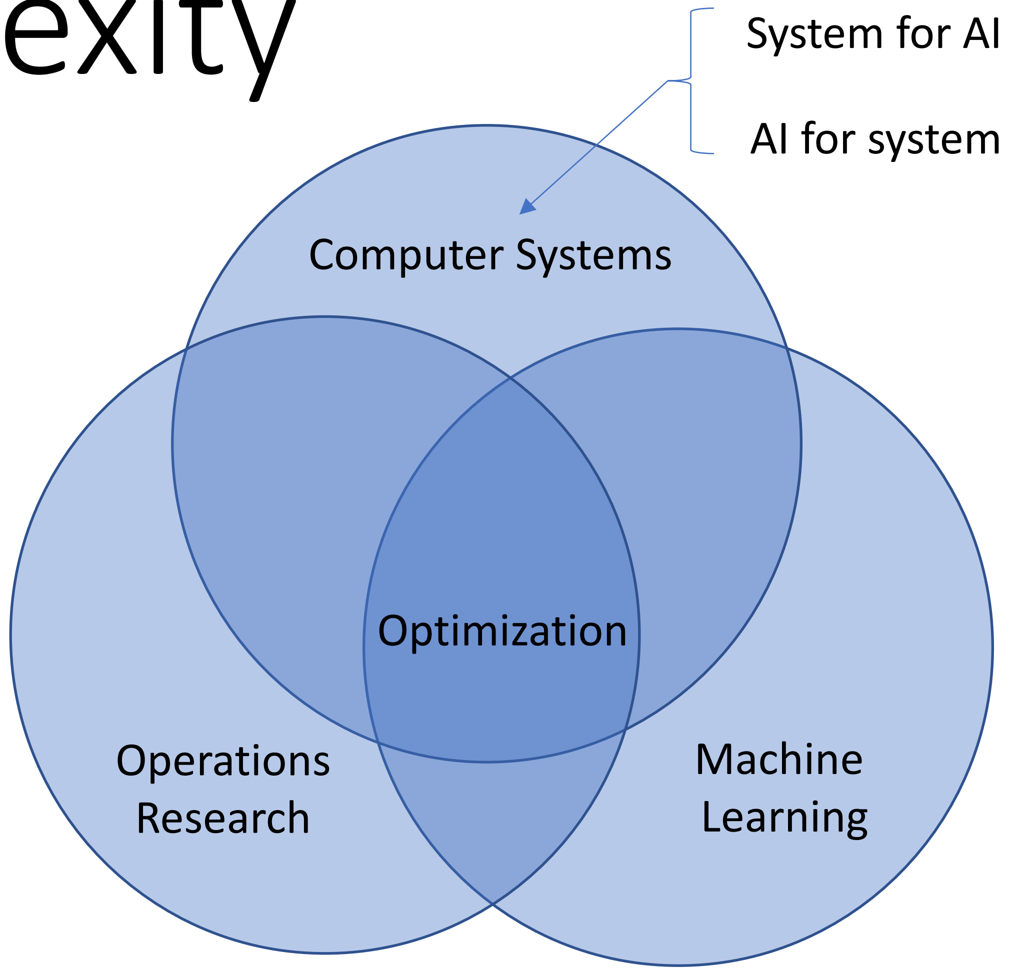
Optimization in Alibaba: Beyond Convexity

Jian Tan
Machine Intelligent Technology
阿里巴巴|达摩院
机器智能技术部 | **智能决策**

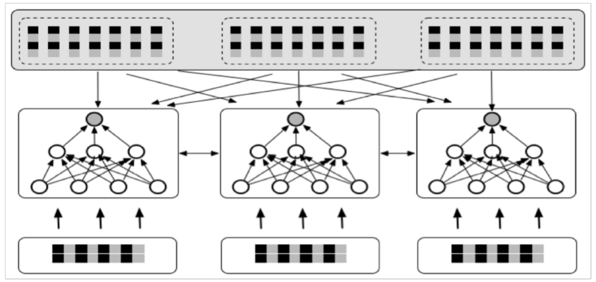


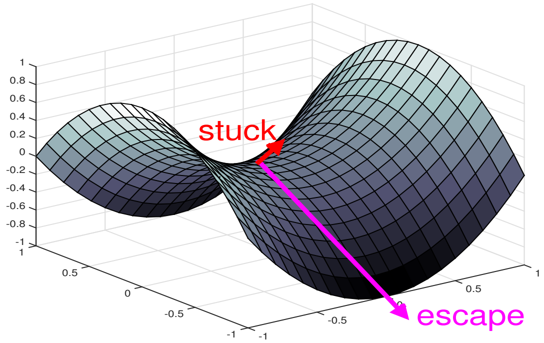
Agenda

➢ Theories on **non-convex** optimization:

Part 1. Parallel restarted SGD: it finds first-order stationary points (why model averaging works for Deep Learning?)

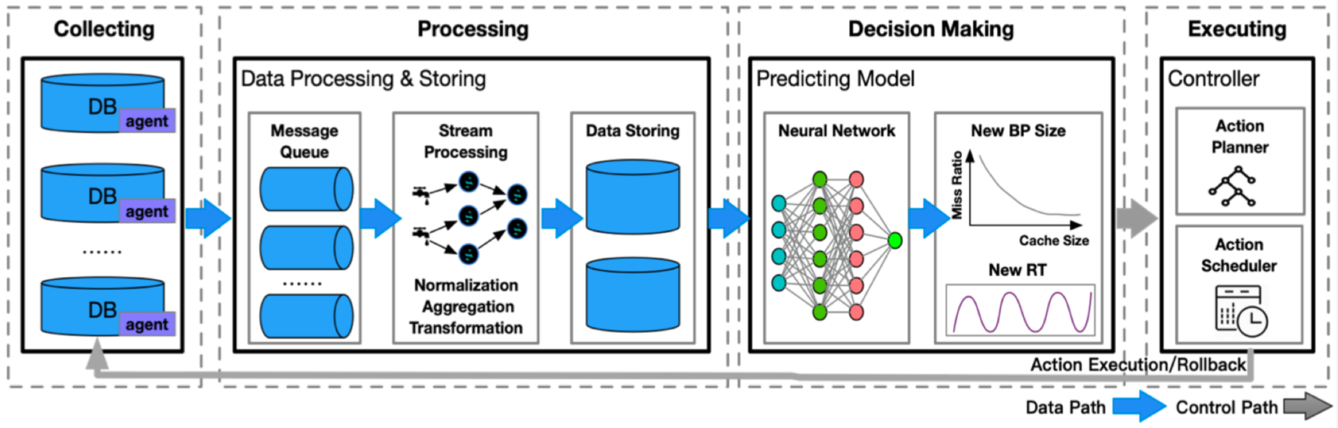
Part 2. Escaping saddle points in non-convex optimization (first-order stochastic algorithms to find second-order stationary points)





➢ System optimization: BPTune for an intelligent database (from OR/ML perspectives)

A real complex system deployment
Combine pairwise DNN, active learning, heavy-tailed randomness …



Part 3. Stochastic (large deviation) analysis for LRU caching

Learning as Optimization

- Stochastic (non-convex) optimization

Loss function

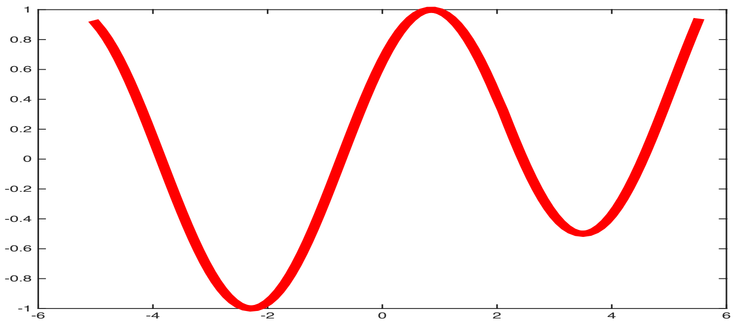
Model

$$\min_{x \in \boldsymbol{R}^d} f(x) = \mathrm{E}[F(x; \xi)]$$

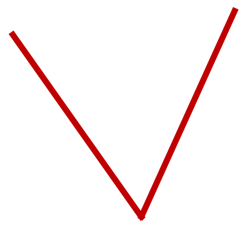
Training samples

- ξ: random training sample
- f(x): has Lipschitz continuous Gradient

$$||\nabla f(x) - \nabla f(y) \leq L||x - y||$$

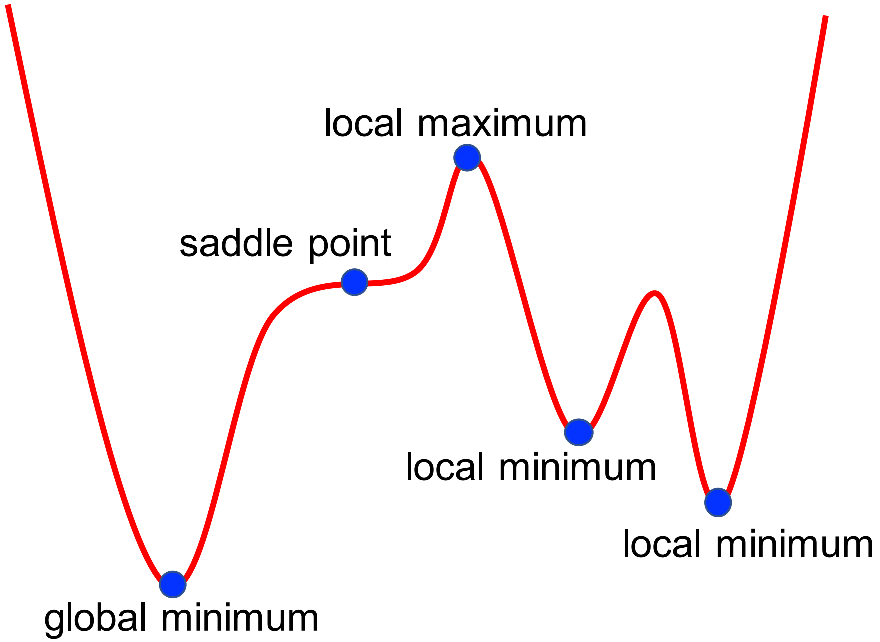


v.s.



Non-Convex Optimization is Challenging

Many local minima & saddle points



For stationary points $\nabla f(x)$=0 (**first-order stationary**)

$\nabla^2 f(x) \succ 0$ ⇨ Local minimum

$\nabla^2 f(x) \prec 0$ ⇨ Local maximum

$\nabla^2 f(x)$ has both +/- eigenvalues ⇨ saddle points

$\nabla^2 f(x)$ has 0/+ eigenvalues

⇨ Degenerate case: could be either local minimum or saddle points

In general, finding global minimum of non-convex optimization is NP-hard

Instead …

- For some applications, e.g., matrix completion, tensor decomposition, dictionary learning, and certain neural networks,

Good news: local minima

- Either all local minima are all global minima
- Or all local minima are close to global minima

Bad news: saddle points

- Poor function value compared with global/local minima
- Possibly many saddle points (even exponential number)

Finding First-order Stationary Points (FSP)

- Stochastic Gradient Descent (SGD):

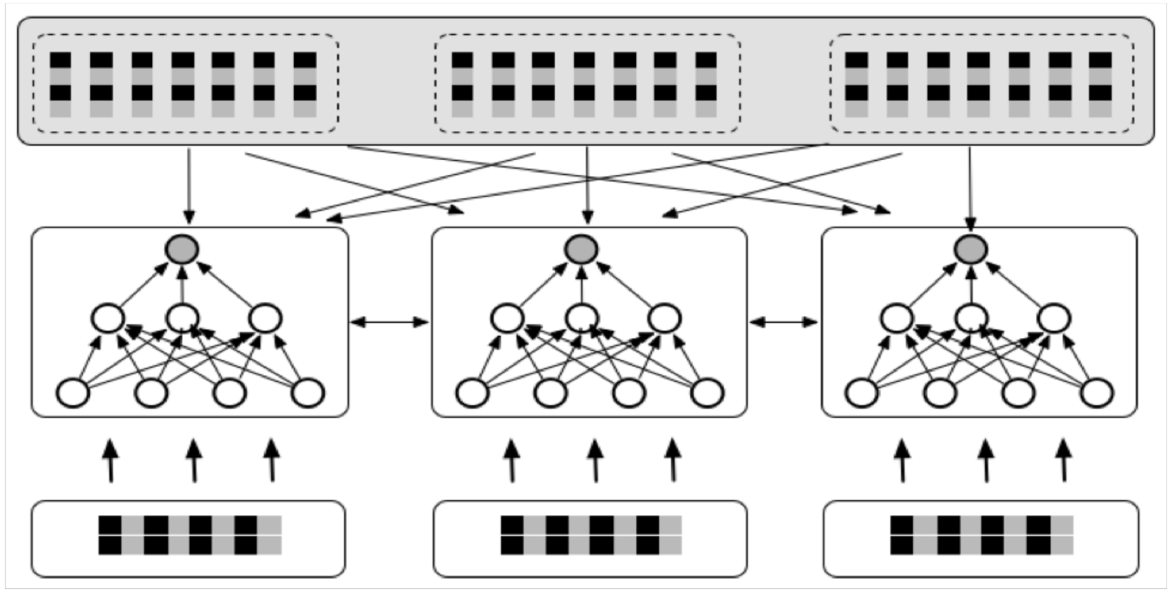
$$x_{t+1} = x_t - \eta \nabla F(x_t; \xi_t)$$

- Complexity of SGD (Ghadimi & Lan, 2013, 2016; Ghadimi et al., 2016; Yang et al., 2016) :
  - ϵ-FSP, $\mathrm{E}[\|\nabla f(x)\|_2^2] \leq \epsilon^2$: Iteration complexity $O(1/\epsilon^4)$
- Improved Iteration complexity based on Variance Reduction:
  - SCSG (Lei et al.,2017): $O(1/\epsilon^{10/3})$
- Workhorse of deep learning

Part 1:

Parallel Restarted SGD with Faster Convergence and Less Communication: Demystifying Why Model Averaging works for Deep Learning

Hao Yu, Sen Yang, Shenghuo Zhu (AAAI 2019)

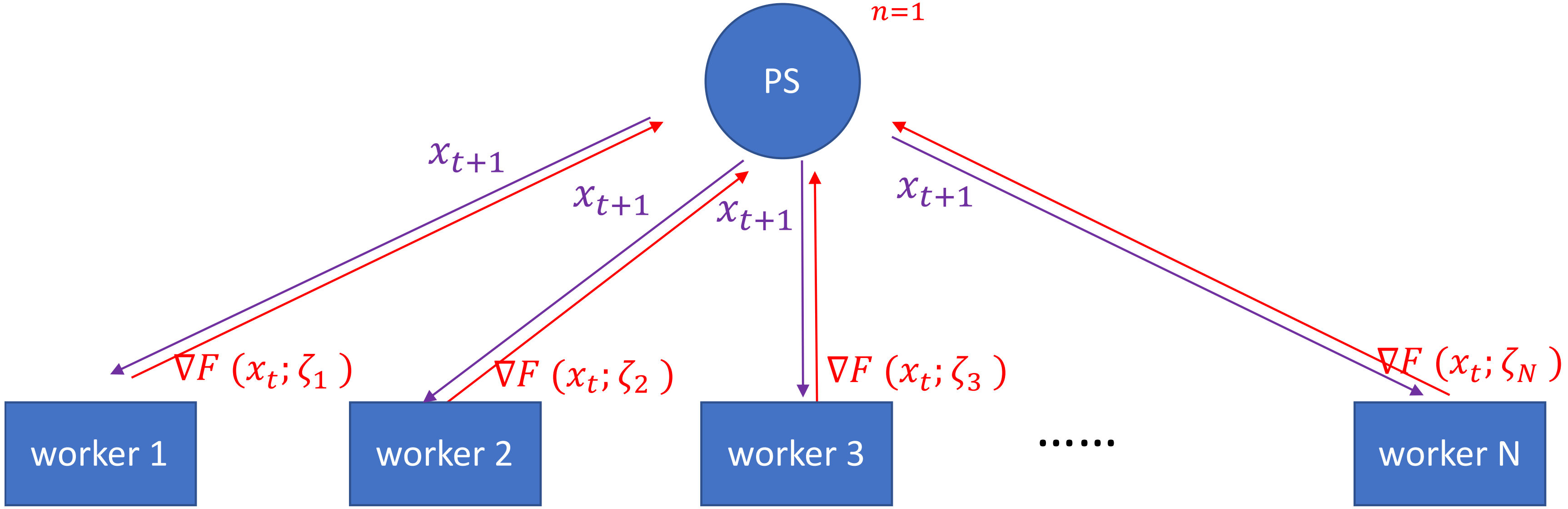


- One server is not enough:
  - too many parameters, e.g., deep neural networks
  - huge number of training samples
  - training time is too long
- Parallel on N servers:
  - With N machines, can we be N times faster? If yes, we have the linear speed-up (w.r.t. # of workers)

Classical Parallel mini-batch SGD

- The classical Parallel mini-batch SGD (PSGD) achieves $O(\frac{1}{\sqrt{NT}})$ convergence with N workers [Dekel et al. 12]. PSGD can attain a linear speed-up.

$$x_{t+1} = x_t - \gamma \frac{1}{N} \sum_{n=1}^{N} \nabla F(x_t; \zeta_n)$$



- Each iteration aggregates gradients from every workers. Communication too high!
- Can we reduce the communication cost? Yes, model averaging.

Model Averaging (Parallel Restarted SGD)

Algorithm 1 Parallel Restarted SGD

Input: Initialize $\mathbf{x}_i^0 = \overline{\mathbf{y}} \in \mathbb{R}^m$. Set learning rate $\gamma > 0$ and node synchronization interval (integer) $I > 0$

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

  Each node i observes an unbiased stochastic gradient $\mathbf{G}_i^t$ of $f_i(\cdot)$ at point $\mathbf{x}_i^{t-1}$

  if t is a multiple of I, i.e., $t\%I = 0$, **then**

    Calculate node average $\overline{\mathbf{y}} \triangleq \frac{1}{N}\sum_{i=1}^{N} \mathbf{x}_i^{t-1}$

    Each node i in parallel updates its local solution

$$\mathbf{x}_i^t = \overline{\mathbf{y}} - \gamma \mathbf{G}_i^t, \quad \forall i \tag{2}$$

  else

    Each node i in parallel updates its local solution

$$\mathbf{x}_i^t = \mathbf{x}_i^{t-1} - \gamma \mathbf{G}_i^t, \quad \forall i \tag{3}$$

  end if

end for

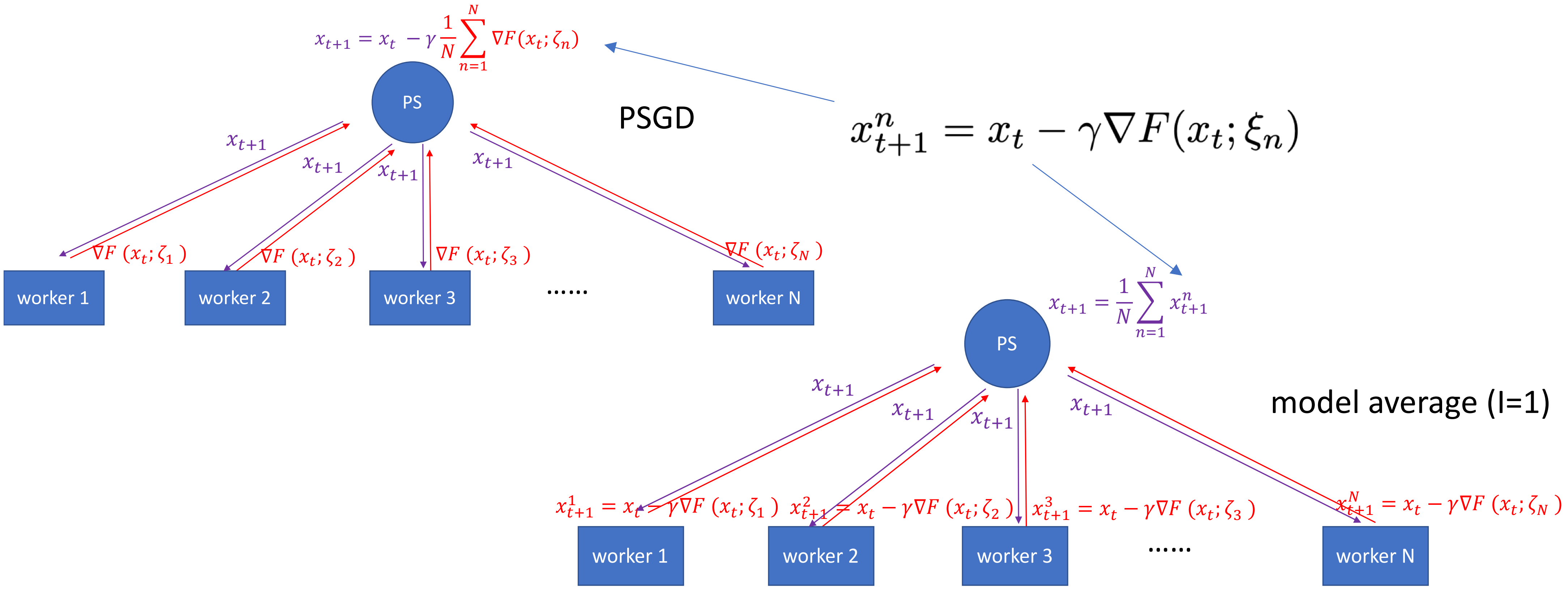
Model Averaging

- Each worker train its local model + (periodically) average on all workers
  - **One-shot averaging**: [Zindevich et al. 2010, McDonalt et al. 2010] propose to average only once at the end.
  - [Zhang et al. 2016] shows averaging once can leads to poor solutions for non-convex opt and suggest more frequent averaging.

- If averaging every I iterations, how large is I ?
  - One-shot averaging: I=T
  - PSGT: I=1

Why l=1 works?

- If we average models each iteration (l=1), then it is equivalent to PSGD.



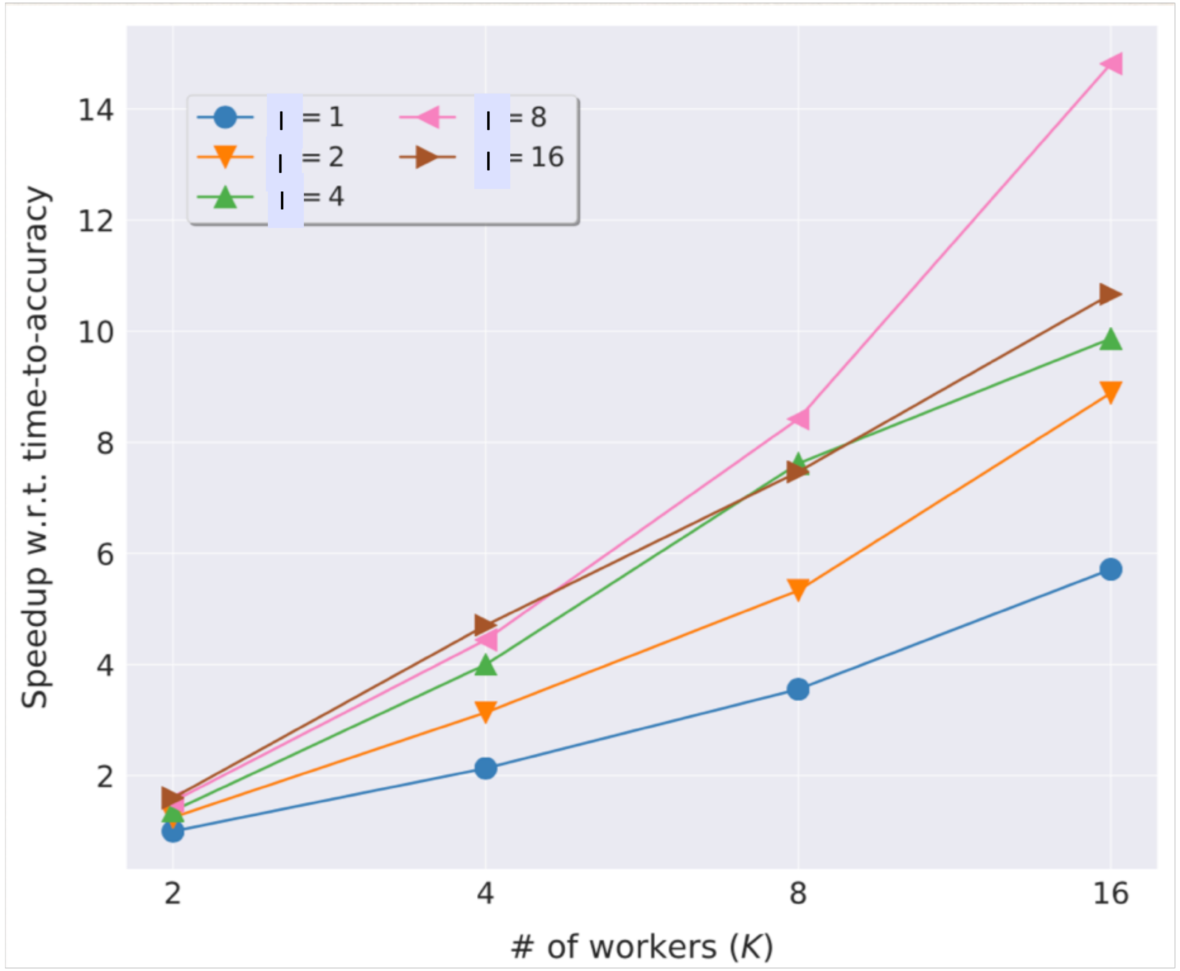
- What if we average after multiple iterations periodically (l>1)?

  Converge or not? Convergence rate? Linear speed-up or not?

Empirical work

- There has been a long line of empirical works …
  - [Zhang et al. 2016]: CNN for MNIST
  - [Chen and Huo 2016] [Su, Chen, and Xu 2018] : DNN-GMM for speech recognition
  - [McMahan et al. 2017] :CNN for MNIST and Cifar10; LSTM for language modeling
  - [Kaamp et al. 2018] :CNN for MNIST
  - [Lin, Stich, and Jaggi 2018]: Res20 for Cifar10/100; Res50 for ImageNet
- These empirical works show that "model averaging" = PSGD with significantly less communication overhead!
- Recall PSGD = linear speed-up

Model Averaging: almost linear speed-up in practice



- Good speed up (measured in wall time used to achieve target accuracy)
- I: averaging intervals (I=4 means "average every 4 iterations")
- Resnet20 over CIFAR10
- Figure 7(a) from "Tao Lin, Sebastian U. Stich, and Martin Jaggi 2018, Don't use large mini-batches, use local SGD"

Alibaba.com Global trade starts here.™

Related work

- For strongly convex opt, [Stich 2018] shows the convergence (with linear speed-up w.r.t. # of workers) is maintained as long as the averaging interval I < $O(\sqrt{T}/\sqrt{N})$.

- Why model averaging achieves almost linear speed-up for deep learning (non-convex) in practice for I>1?

Main result

- Prove "model averaging " (communication reduction) has the same convergence rate as PSGD for non-convex opt under certain conditions

If the averaging interval $I = O(T^{\frac{1}{4}}/N^{\frac{3}{4}})$, then model averaging has the convergence rate $O(\frac{1}{\sqrt{NT}})$.

- "Model averaging" works for deep learning. It is as fast as PSGD with significantly less communication.

Control bias-variance after I iterations

- Focus on

$$\bar{x}^t = \frac{1}{N}\sum_{i=1}^{N} x_i^t$$

average of local solution over all N workers

- Note…

$$\bar{x}^t = \bar{x}^{t-1} - \gamma \frac{1}{N}\sum_{i=1}^{N} G_i^t$$

G_i^t : independent gradients sampled at different points x_i^{t-1}

- PSGD has i.i.d. gradients at $\bar{x}^{t-1}$, which are unavailable at local workers without communication

Technical analysis

- Bound the difference between $\bar{x}^t$ and x_i^t

Our Algorithm ensures $E[||\bar{x}^t - x_i^t||^2] \leq 4\gamma^2 I^2 G^2, \forall i, \forall t$

- The rest part uses the smoothness and shows

$$\frac{1}{T}\sum_{t=1}^{T}\mathbb{E}\left[\|\nabla f(\overline{\mathbf{x}}^{t-1})\|^2\right] \leq \frac{2}{\gamma T}\left(f(\overline{\mathbf{x}}^0) - f^*\right) + 4\gamma^2 I^2 G^2 L + \frac{2}{N}\gamma\sigma^2$$

Assume:

$$\mathbb{E}_{\zeta_i \sim \mathcal{D}_i}\|\nabla F_i(\mathbf{x};\zeta_i) - \nabla f_i(\mathbf{x})\|^2 \leq \sigma^2$$

$$\mathbb{E}_{\zeta_i \sim \mathcal{D}_i}\|\nabla F_i(\mathbf{x};\zeta_i\|^2 \leq G^2$$

Proof. Fix $t \geq 1$. By the smoothness of f , we have

$$\mathbb{E}[f(\overline{\mathbf{x}}^t)] \leq \mathbb{E}[f(\overline{\mathbf{x}}^{t-1})] + \mathbb{E}[\langle\nabla f(\overline{\mathbf{x}}^{t-1}), \overline{\mathbf{x}}^t - \overline{\mathbf{x}}^{t-1}\rangle] + \frac{L}{2}\mathbb{E}[\|\overline{\mathbf{x}}^t - \overline{\mathbf{x}}^{t-1}\|^2]$$

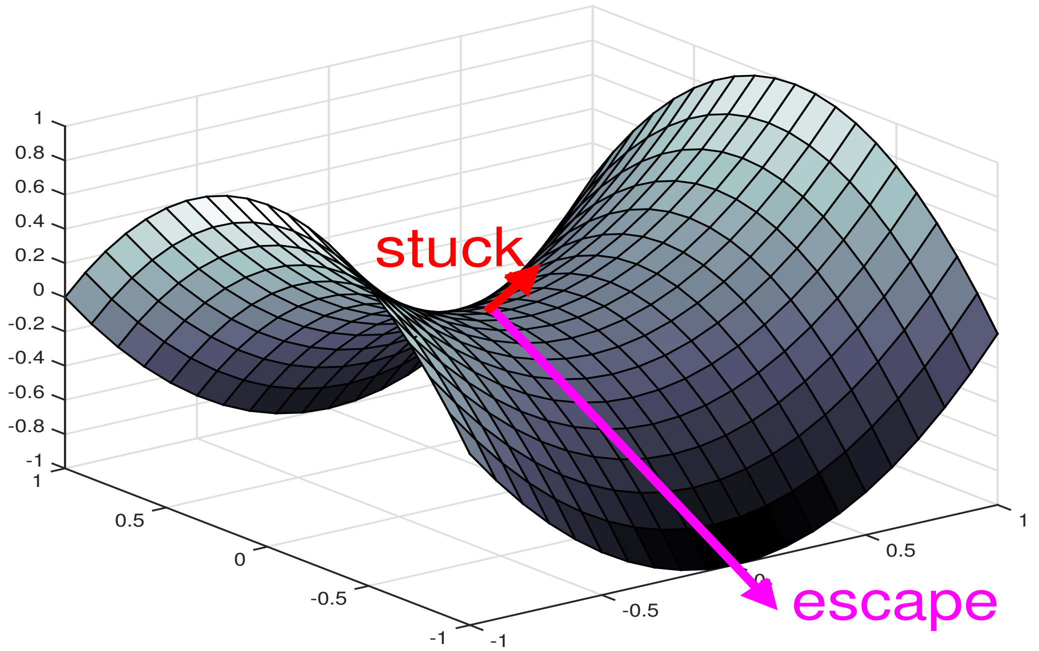
Note that

$$\mathbb{E}[\|\overline{\mathbf{x}}^t - \overline{\mathbf{x}}^{t-1}\|^2] \overset{(a)}{=} \gamma^2\mathbb{E}[\|\frac{1}{N}\sum_{i=1}^{N}\mathbf{G}_i^t\|^2]$$

⋮

Part 2:
Escaping Saddle points in non-convex optimization

Yi Xu*, Rong Jin, Tianbao Yang*

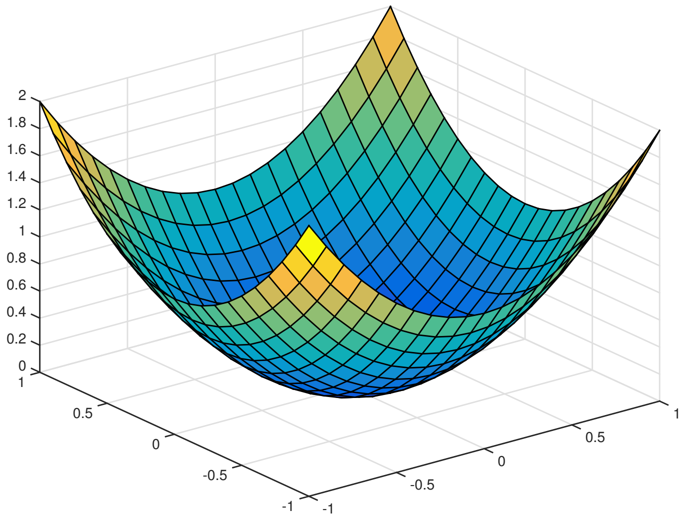


First-order Stochastic Algorithms for Escaping From Saddle Points in Almost Linear Time, NIPS 2018.

* Xu and Yang are with Iowa State University

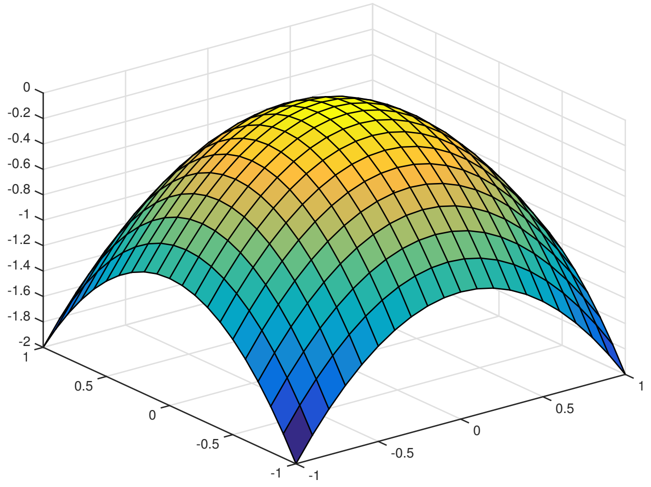
(First-order) Stationary Points (FSP) $\|\nabla F(x)\|_2 = 0$

Local minimum



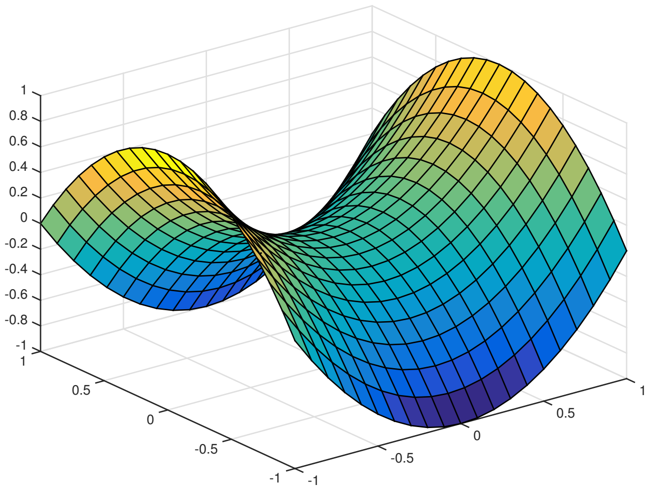
$\nabla^2 f(x) \succ 0$

Local maximum



$\nabla^2 f(x) \prec 0$

Saddle point



$\lambda_{min}(\nabla^2 f(x)) < 0$

Second-order Stationary Points (SSP)

$\|\nabla f(x)\|_2 = 0, \lambda_{min}(\nabla^2 f(x)) \geq 0$

SSP is Local Minimum for non-degenerate saddle point

$\nabla^2 f(x)$ has both +/- eigenvalues ⇨ saddle points, which can be bad!

$\nabla^2 f(x)$ has both 0/+ eigenvalues ⇨ degenerate case: local minimum/saddle points

The Problem

- Finding an approximate local minimum by using first-order methods

$$\epsilon\text{–SSP: } \|\nabla f(x)\|_2 \leq \epsilon, \lambda_{min}\left(\nabla^2 f(x)\right) \geq -\gamma$$

- Choice of γ : small enough, e.g., $\gamma = \sqrt{\epsilon}$ (Nesterov & Polyak 2006)

Nesterov, Yurii, and Polyak, Boris T. "Cubic regularization of Newton method and its global performance." *Mathematical Programming* 108.1 (2006): 177-205.

Related Work

- Adding Isotropic Noise: Noisy SGD (Ge et al., 2015), SGLD (Zhang et al., 2017)

$$x_{t+1} = x_t - \eta(\nabla F(x_t; \xi_t) + n_t)$$

  - n_t is an isotropic noise vector (e.g., Gaussian)
  - Iteration complexity: $\tilde{O}(d^p/\epsilon^4)$, where $p \geq 4$, d is dimension
  - Noisy SGD is the first work on finding local minimum by first order methods
  - For high-dimensional optimization problems, d is large

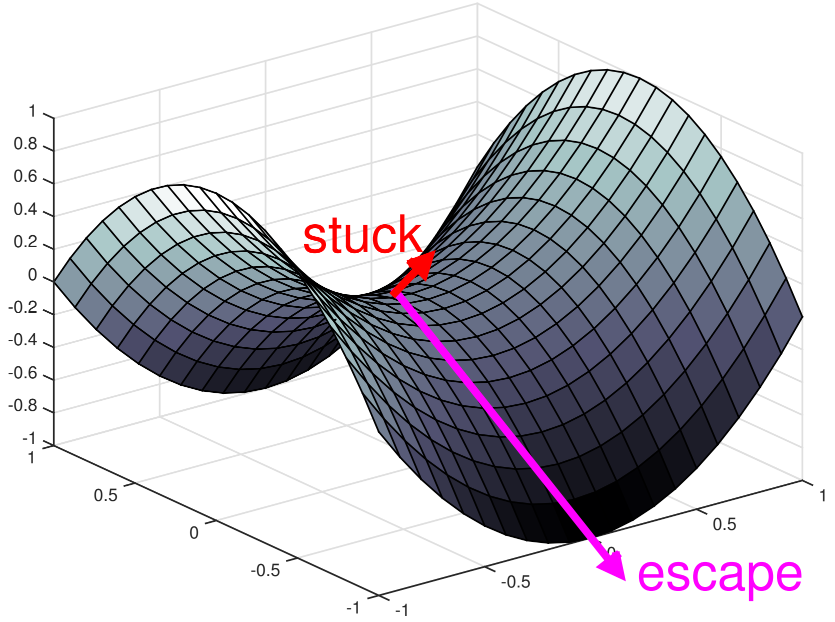
- Assume F$(x; \xi)$ has Lipschitz continuous Gradient and Hessian

More Related Work

- Using Full Gradient (FG) and Isotropic Noise: Perturbed GD (Jin et al., 2017)
  - Add Perturbation Around a Saddle Point $\widetilde{x_t} = x_t + n_t$
  - Take Gradient Descent from $\widetilde{x_t}$
  - Iteration Complexity:$\tilde{O}\ (1/\epsilon^4)$, which hides the team $(\log d)^p$
- Using Hessian-vector product (HVP): (Allen-Zhu, 2017)[Natasha2]
  - Iteration Complexity: $\tilde{O}\ (1/\epsilon^{3.5})$
  - The cost of computing HVP per-iteration could be as high as $O(d^2)$
- Using both FG and HVP (Carmon et al., 2016; Agarwal et al., 2017)

Issue: FG and HVP could be more expensive than SG

Motivation: How to Escape from Saddles?



$$f(x+\Delta) \approx f(x) + \Delta^T \nabla f(x) + \boxed{\frac{L}{2}\Delta^T \nabla^2 f(x) \Delta} < F(x)$$

- Saddle points have zero gradient, i.e., $\nabla f(x) = 0$
- Non-degenerate Hessian, i.e. $\lambda_{min}(\nabla^2 f(x)) < 0$
- Negative eigenvector is a direction of escaping

Negative Curvature

Suppose $\lambda_{min}(\nabla^2 f(x)) \leq -\gamma$, a direction $v \in R^d$ is called negative curvature (NC) direction if it satisfies ($c > 0$ is a constant)

$$v^T \nabla^2 f(x) v \leq -c\gamma \text{ and } \|v\| = 1$$

- Find a NC direction v, update solution by $x_{t+1} = x_t - \eta v$
- Escape Saddles: we show $f(x_t) - f(x_{t+1}) \geq \Omega(\gamma^3)$

How to Find NC?

- Second-order Methods: Power Method and Lanczos method

v_0 =n // isotropic noise

Iterate:

$$v_{t+1} = (I - \eta \nabla^2 F(x))\, v_t$$

How to find NC without using HVP and Full Gradient?

Propose **NEON**: **NE**gative curvature **O**riginated from **N**oise

Alibaba.com Global trade starts here.™

NEON: A New Perspective of Noise Perturbation

- Adding Noise is for Extracting NC
  - x: around a saddle point
  - Inspired by Perturbed Gradient Descent (PGD):
    - $x_0 = x + e$, noise e is from sphere of a Euclidean ball
    - $x_t = x_{t-1} - \eta \nabla F(x_{t-1}), t = 1, \cdots,$
- An Equivalent Sequence: let $u_t = x_t - x$
  - $u_t = u_{t-1} - \eta \nabla F(u_{t-1} + x)$

$$\approx u_{t-1} - \eta [\nabla F(u_{t-1} + x) - \nabla F(x)]$$

$$\approx u_{t-1} - \eta \nabla^2 F(x) u_{t-1} = [I - \eta \nabla^2 F(x)] u_{t-1}$$

$\nabla F(x) \approx 0$

Lipschitz continuous Hessian when $\|u_{t-1}\|$ is small: $\nabla F(u_{t-1} + x) - \nabla F(x)] \approx \nabla^2 F(x) u_{t-1}$

- Around Saddle Point: PGD $\approx$ Power Method

NEON Update: Starting with a random noise u_0, the recurrence:

$$u_{t+1} = u_t - \eta(\nabla F(x + u_t) - \nabla F(x))$$

iteration complexity $= \tilde{O}\left(\frac{1}{\gamma}\right)$

Alibaba.com Global trade starts here.™

NEON+: Another Perspective

- Recall the update of NEON: $u_{t+1} = u_t - \eta(\nabla F(x+u_t) - \nabla F(x))$
- NEON is essentially an application of GD to decrease $F_x(u)$:

$$F_x(u) = F(x+u) - F(x) - \nabla F(x)^T u$$

Use Nesterov's Accelerated Gradient to decrease $F_x(u)$:

$$y_{t+1} = u_t - \eta\nabla F_x(u_t),\ u_{t+1} = y_{t+1} + \zeta(y_{t+1} - y_t)$$

$For\ \zeta = 1 - \sqrt{\eta\gamma}$, # iteration can be reduced to $t = \tilde{O}\left(\frac{1}{\sqrt{\gamma}}\right)$

Applications of NEON: Finding Local Minimum

Given a first-order alg. $\mathcal{A}$ (it can find a FSP)

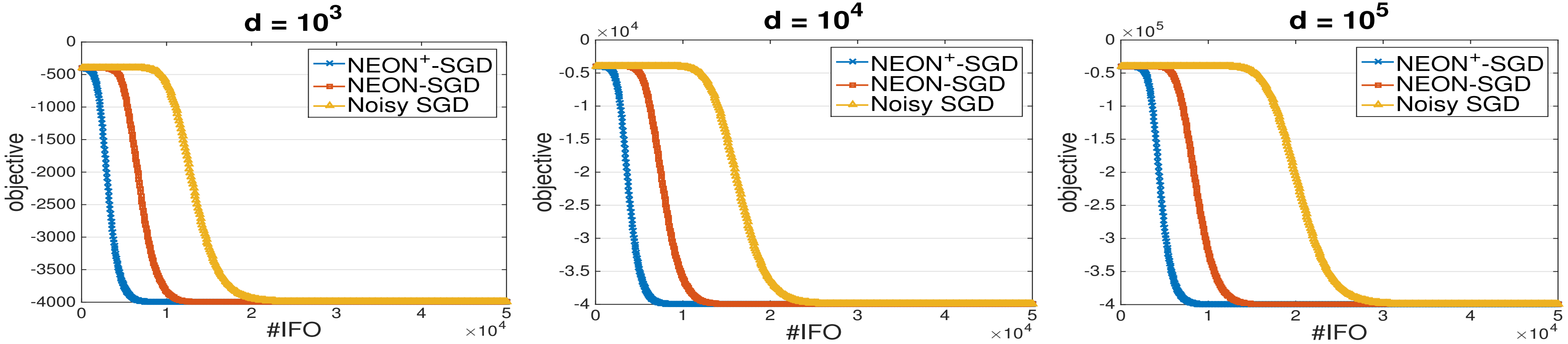
- SGD, Stochastic Heavy-ball, Stochastic Nesterov's Accelerated Method
- Variance reduction methods, e.g., SCSG, SVRG

NEON + $\mathcal{A}$ -> find a SSP point

- e.g., **NEON-SCSG** enjoy iteration complexity of $\tilde{O}\ (1/\epsilon^{3.5})$ for finding $(\epsilon, \sqrt{\epsilon})$-SSP only using first-order information

Example: finding local minimum

$f(x) = \sum_{i=1}^{d} \xi_i (x_i^4 - 4x_i^2)$ ξ_i : a normal random variables with mean of 1



Alibaba.com® Global trade starts here.™

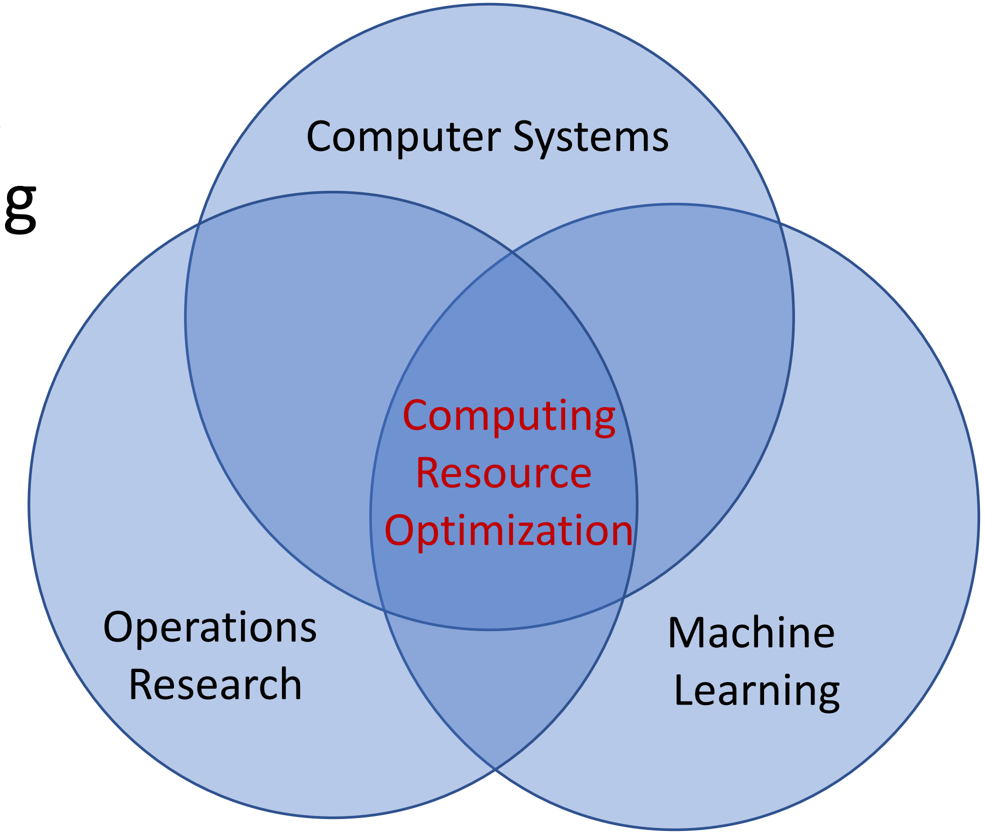
Part 3: BPTune: Optimizing Buffer Pool Management for Large-Scale OLTP Database Clusters

J. Tan, T. Zhang, F. Li, J. Chen, Q. Zheng,
P. Zhang, H. Qiao, Y. Shi, W. Cao, R. Zhang

A real system deployed for Alibaba database clusters

Algorithm: large deviation, deep neural networks, active learning

Large deviation on LRU: joint work with Quan, Ji and Shroff from The Ohio State University



“Personalization” for > 10,000 database instances

Measurements can NOT help much:

1. real BP usage ≈ configured size
2. (miss ratio, response time) ←?→ BP size

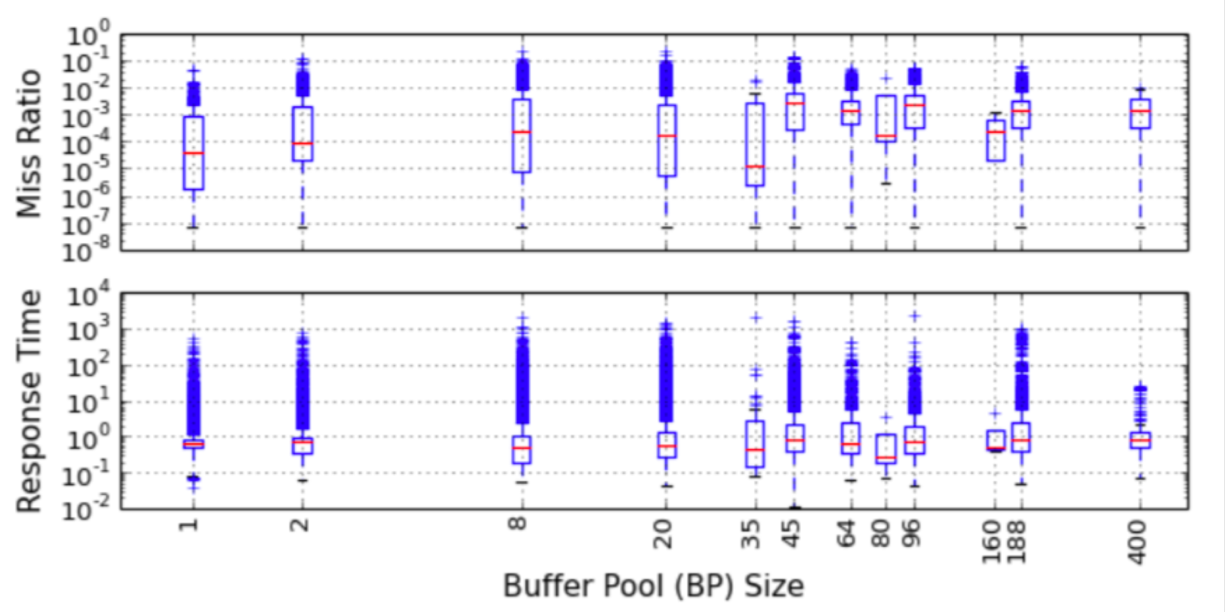
Current practice:

1. Overprovision (e.g., double BP size)
2. Use only a few BP sizes

Challenges:

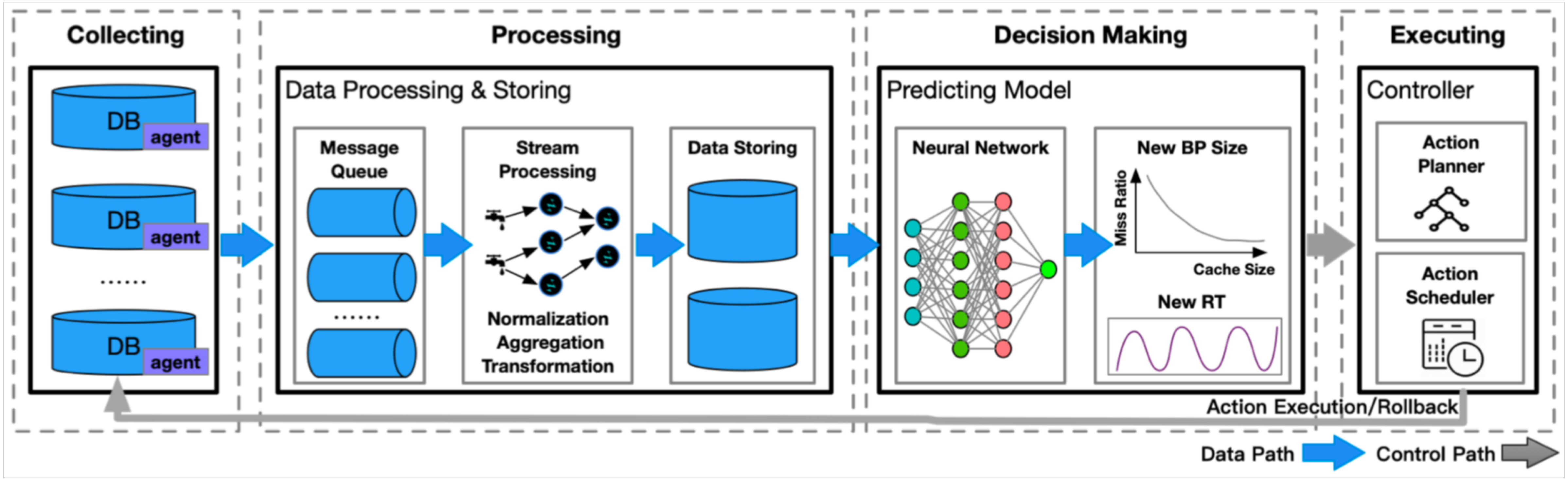
1. “Personalization” - find the “best” BP size for each instance; manual optimization is not scalable.
2. Prediction - estimate the response time for queries on each instance after changing its BP size?

BP = memory = fast access



Measurements on 10,000 database instances
an instance = a database working unit
Use only 11 different BP sizes
by manual configurations

BPTune architecture



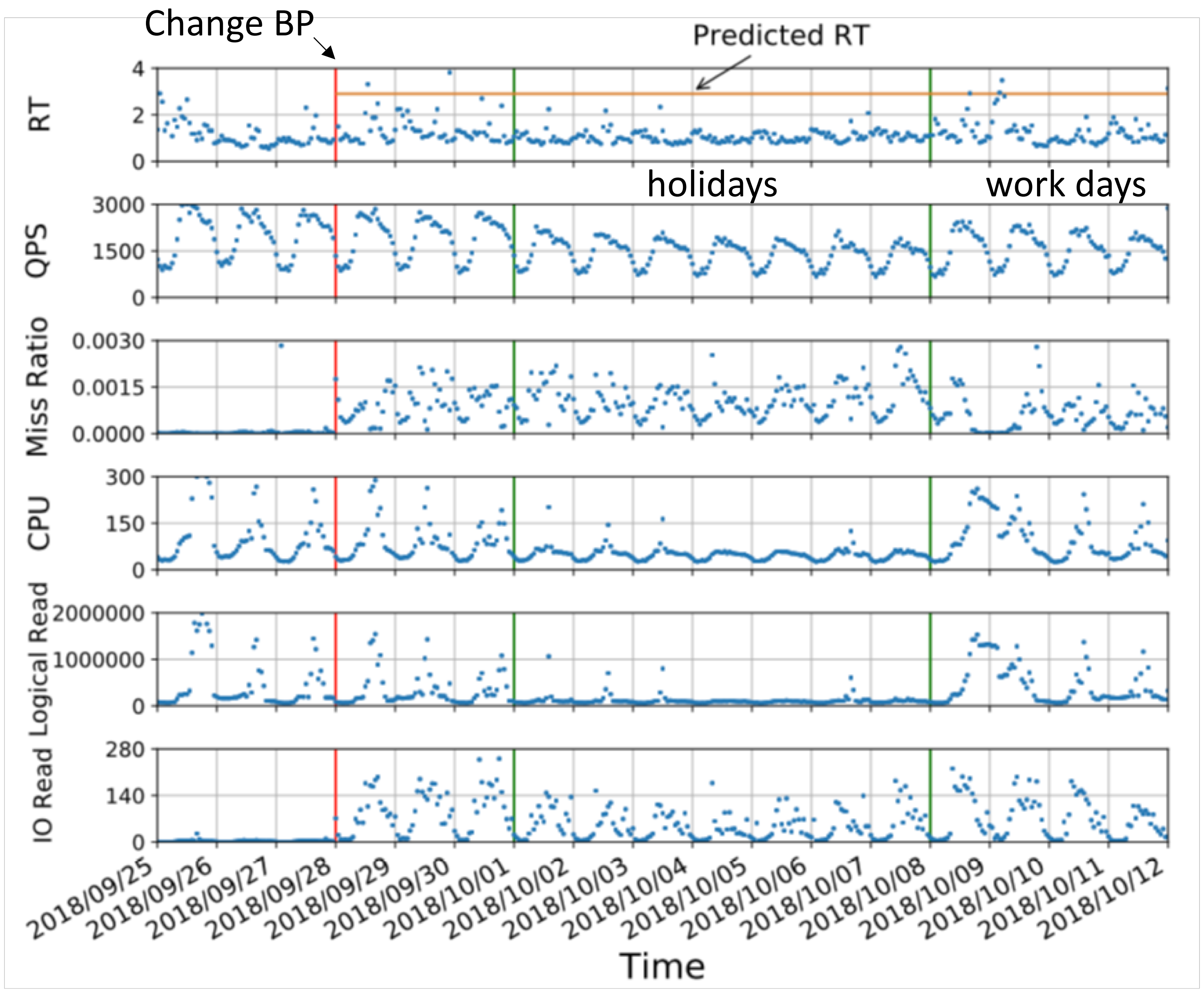
Reduce > 20% BP memory, compared with manual configurations

A bin-packing analysis shows BP is the bottleneck resource

Real experiment on an instance

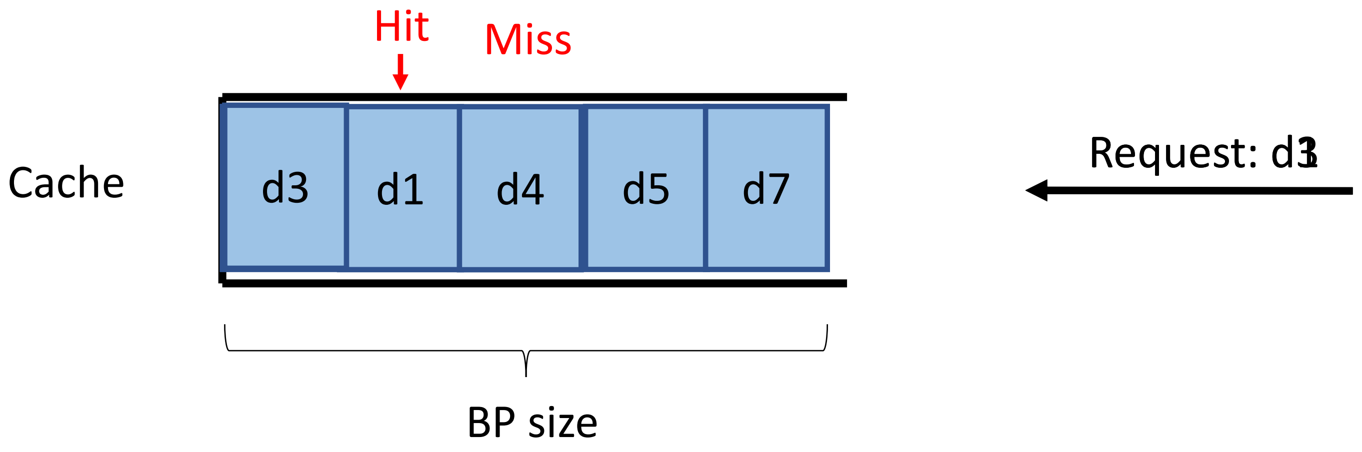
Response Time: processing time of queries

Miss Ratio: fraction of queried Data not in memory



Today focus on LRU Caching algorithm

- Least recently used (LRU) algorithm (widely used: Memcached, Redis)
  - Store the most recently used data in the cache.
  - Easy to implement, adaptive to time-varying popularities
  - Q: What is the miss ratio of LRU?



Goal & challenges

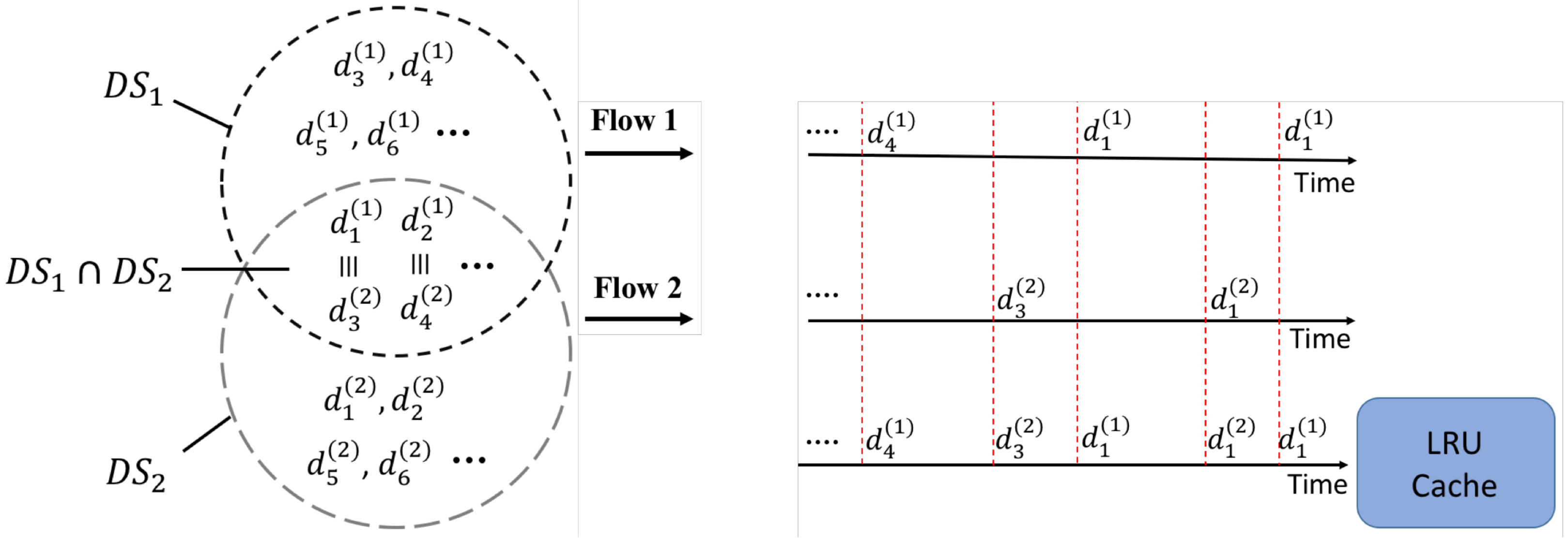
- Goal: characterize BP size = F (miss ratio)
  - Accurately and explicitly compute LRU miss ratio
  - A unified analysis solving all challenges below

- Challenges
  - Different data sizes
  - Time correlations
  - Multiple query flows on a single BP
  - Overlapped data across different flows
  - Long tailed data access probabilities
    e.g., Zipf’s distribution, Weibull distribution

Model

- K sets of data: $DS_1, DS_2, \dots, DS_K$, $DS_k = \{d_i^{(k)}, 1 \le i \le N_k\}$
- K data flows sharing a LRU cache:

  Data flow k: a sequence of requests on the data set DS_k
- Time correlation
  - $\{\Pi_t\}_{t \in \mathbb{R}}$: a stationary and ergodic modulating process with finite states $\{1,2,\dots,M\}$ and the stationary distribution $(\pi_1, \pi_2, \dots, \pi_M)$.
  - Request rates, data popularities vary in different states.
- Goal: $\mathbb{P}[\text{Miss}]$.



New functional representation

- Define the (conditional) popularities

$$p_i^{(k)} \triangleq \sum_{m=1}^{M} \pi_m \mathbb{P}[\text{request data } d_i^{(k)} | \text{in state } m] = \sum_{m=1}^{M} \pi_m p_i^{(k,m)},$$

$$q_i^{(k)} \triangleq \sum_{m=1}^{M} \pi_m \mathbb{P}[\text{request data } d_i^{(k)} | \text{the request is from flow } k, \text{in state } m]$$

$$= \pi_m q_i^{(k,m)}.$$

$p_i^{(k)}$ and $q_i^{(k)}$ can be very different.

- Functional relationship $\Psi_k(\cdot)$ & finite support impacting $\Theta_k(\cdot)$:

For each flow k, for $\forall \lambda > 1$, let the size of the data set $N_k \sim \lambda y$. Find two eventually decreasing functions $\Psi_k(\cdot)$ and $\Theta_k(\cdot)$ that satisfy, as $y \to \infty$,

$$\sum_{i=y}^{N_k} q_i^{(k)} \sim \Psi_k\left(\left(p_y^{(k)}\right)^{-1}\right) + \Theta_k(N_k)$$

where $f(x) \sim g(x) \Leftrightarrow \lim_{x\to\infty} f(x)/g(x) = 1.$

New functional representation

- Example: If $p_i^{(k)} = q_i^{(k)} = c_k / i^{\alpha_k},\ 1 \le i \le N, k = 1$, we have

  for flow 1:

$$\sum_{i=y}^{N} q_i^{(1)} \sim \int_y^N \frac{\pi_1 c_1}{x^{\alpha_1}} dx = \frac{\pi_1 c_1}{(\alpha_1 - 1) y^{\alpha_1 - 1}} - \frac{\pi_1 c_1}{(\alpha_1 - 1) N^{\alpha_1 - 1}}$$

$$= \frac{\pi_1 c_1}{\alpha_1 - 1} \left(\pi_1 c_1 \nu_{1,1} / p_i^{(1)} \right)^{1/\alpha_1 - 1} - \frac{\pi_1 c_1}{(\alpha_1 - 1) N^{\alpha_1 - 1}}$$

$$\Psi_1(x) = \frac{(\pi_1 c_1)^{1/\alpha_1} \nu_{1,1}{}^{1/\alpha_1 - 1}}{\alpha_1 - 1} x^{1/\alpha_1 - 1} \qquad \Theta_1(x) = -\frac{\pi_1 c_1}{\alpha_1 - 1} x^{-\alpha_1 + 1}$$

Alibaba.com Global trade starts here.™

Main result

Theorem *[Tan, Quan, Ji, Shroff]:* Consider K flows without overlapped data that are modulated by the stationary and ergodic process $\{\Pi_t\}_{t\in\mathbb{R}}$. For flow k, if $\Psi_k(x) \sim x^{\beta} l(x)$, then under mild conditions, we have, as the cache size $x \to \infty$, for $\forall\lambda > 0$, $N_k = \lambda m^{\leftarrow}(x)$,

$$\mathbb{P}[\text{Miss}|\text{the request is from flow } k] \sim \beta\Gamma\left(\beta, m^{\leftarrow}(x) p_{N_k}^{(k)}\right) \Psi_k(m^{\leftarrow}(x)),$$

where $m^{\leftarrow}(x)$ is the inverse function of

$$m(x) = \sum_{k=1}^{K} \sum_{i=1}^{N_k} s_i^{(k)} \left(1 - \exp\left(-\sum_{m=1}^{M} \pi_m \nu_{k,m} q_i^{(k,m)} x\right)\right).$$

Note:

- $l(x)$ is any slowly varying function satisfying $\lim_{x\to\infty} l(\lambda x)/l(x) = 1$ for any $\lambda > 0$. (e.g., $\log(x)$, c, etc.)
- $\Gamma(\beta, s) = \int_s^{\infty} x^{\beta-1} e^{-x} dx$ is the incomplete gamma function.
- Quan, Ji and Shroff are with The Ohio State University

Main result

Corollary: Consider one flow of unit-sized data. Assume $q_i^{(1)} \sim c/i^{\alpha}$, $1 \le i \le N$.
For $\forall \lambda > 0$, $N = \lambda m^{\leftarrow}(x)$, we have, as the cache size $x \longrightarrow \infty$,

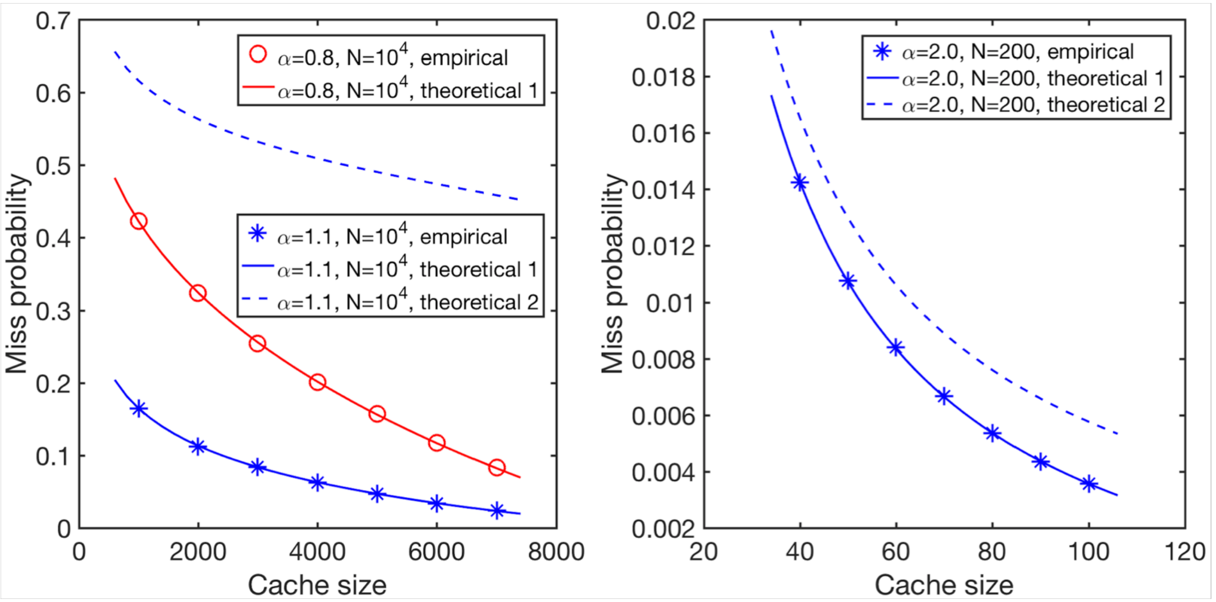
$$\mathbb{P}[\text{Miss}] \sim \frac{c^{1/\alpha}}{\alpha}\Gamma\left(1-\frac{1}{\alpha}, \frac{c m^{\leftarrow}(x)}{N^{\alpha}}\right) m^{\leftarrow}(x)^{-1+1/\alpha},$$

where, $m^{\leftarrow}(x)$ is the inverse function of

$$m(x) = \Gamma\left(1-\frac{1}{\alpha}, \frac{cx}{N^{\alpha}}\right)(cx)^{1/\alpha} + N\left(1-\exp\left(-\frac{cx}{N^{\alpha}}\right)\right).$$

Our result (labeled as 'theoretical 1')

Previous result (labeled as 'theoretical 2')



Alibaba.com Global trade starts here.™

Conclusion

- **System for AI**

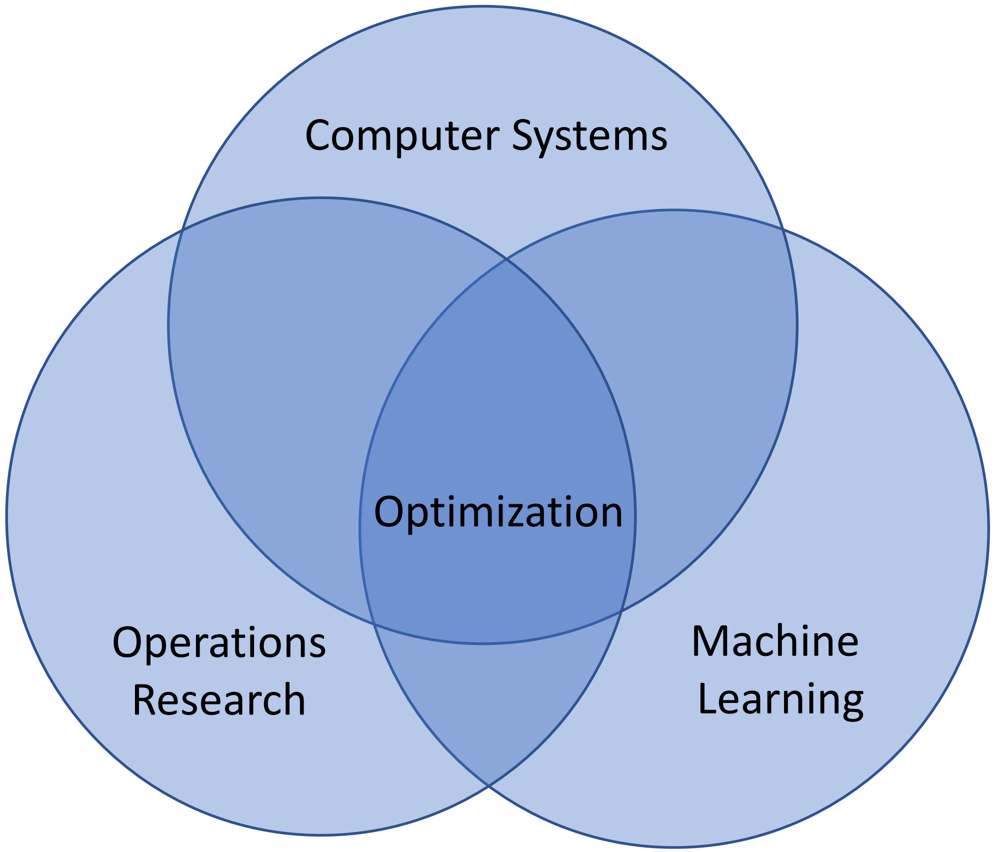
  Part 1. Parallel restarted SGD
  (why model averaging works for Deep Learning?)

  Part 2. Escaping saddle points in non-convex optimization
  (first-order stochastic algorithms to find second-order stationary points)

- **AI for system**

  BPTune: intelligent database
  A real complex system deployment
  Combine OR/ML, e.g., pairwise DNN, active learning, heavy-tailed randomness …

  Part 3. Stochastic (large deviation) analysis for LRU caching



Thank You! Questions?