GIANT: Experiments

Experiment Environment

- Spark 2.1.1 + Scala 2.11.8

Experiment Environment

- Spark 2.1.1 + Scala 2.11.8

APACHE Spark

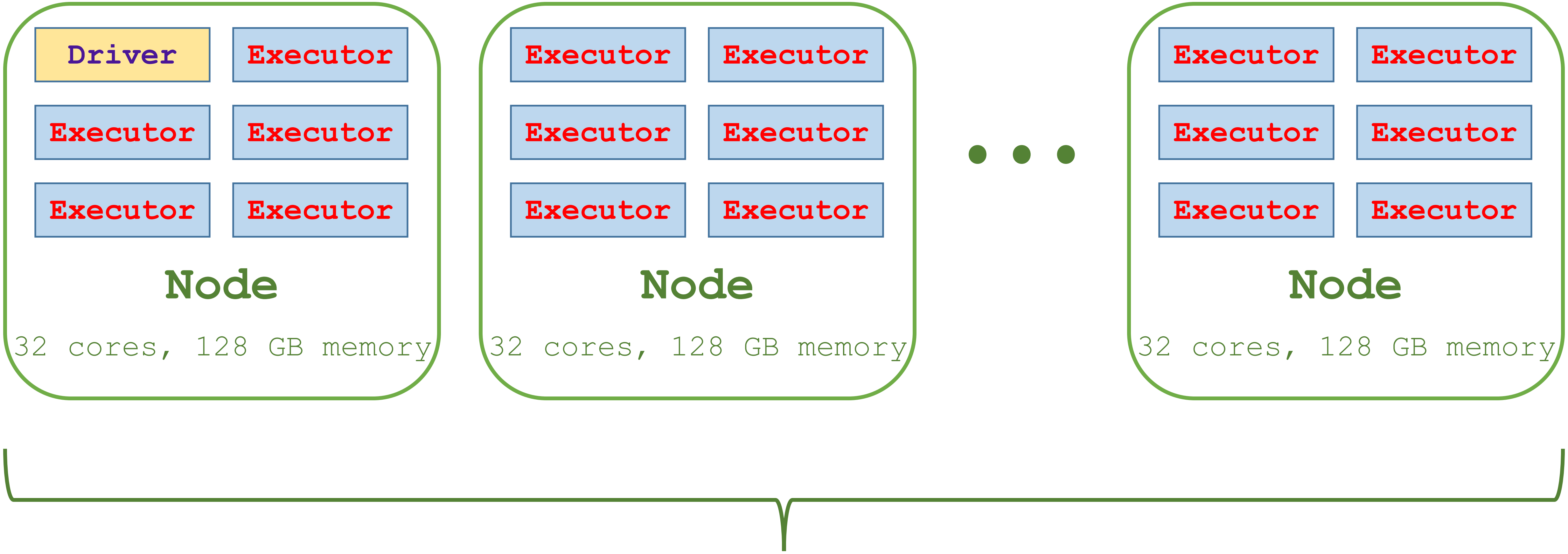
Scala

- Cori Supercomputer

NERSC
National Energy Research
Scientific Computing Center

BERKELEY LAB
Bringing Science Solutions to the World

Experiment Environment



10 nodes, 59 executors

Settings

- Solve the ℓ_2-regularized logistic regression:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \left\{ f(\mathbf{w}) \triangleq \frac{1}{n}\sum_{j=1}^{n} \log\left(1 + e^{-y_j \mathbf{x}_j^T \mathbf{w}}\right) + \frac{\gamma}{2}\|\mathbf{w}\|_2^2 \right\}$$

- Split $\mathbf{X} \in \mathbb{R}^{n\times d}$ (by data) to $m = 59$ parts.
- Local sample size $s = \frac{n}{m}$

Settings

- Solve the ℓ_2-regularized logistic regression:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \left\{ f(\mathbf{w}) \;\triangleq\; \frac{1}{n}\sum_{j=1}^{n}\log\left(1+e^{-y_j\mathbf{x}_j^T\mathbf{w}}\right)+\frac{\gamma}{2}\|\mathbf{w}\|_2^2 \right\}$$

- Split $\mathbf{X}\in\mathbb{R}^{n\times d}$ (by data) to $m=59$ parts.
- Local sample size $s=\frac{n}{m} \quad \gtrsim \quad$ number of features d.

Settings

- Solve the ℓ_2-regularized logistic regression:

$$\min_{\mathbf{w}\in\mathbb{R}^d} \quad \left\{ f(\mathbf{w}) \triangleq \frac{1}{n}\sum_{j=1}^{n} \log\left(1 + e^{-y_j \mathbf{x}_j^T \mathbf{w}}\right) + \frac{\gamma}{2}\|\mathbf{w}\|_2^2 \right\}$$

- Split $\mathbf{X} \in \mathbb{R}^{n\times d}$ (by data) to $m = 59$ parts.
- Local sample size $s = \frac{n}{m} \quad \gtrsim \quad$ number of features d.
- We use dense $\mathbf{X}$.

Compared Methods

- Accelerated gradient descent (AGD)
  - choose *step size* from {0.1, 1, 10, 100}
  - choose *momentum* from {0.5, 0.9, 0.95, 0.99, 0.999}

Compared Methods

- Accelerated gradient descent (AGD)
- Limited memory BFGS (a quasi-Newton method)
  - choose *number of history* from {30, 100, 300}
  - line search is used

Compared Methods

- Accelerated gradient descent (AGD)
- Limited memory BFGS
- DANE (another Newton-type method) [Shamir et al. 2014]
  - local solver: SVRG
  - choose *step size of SVRG* from {0.1, 1, 10, 100}
  - choose *max iteration of SVRG* from {30, 100, 300}

Reference:

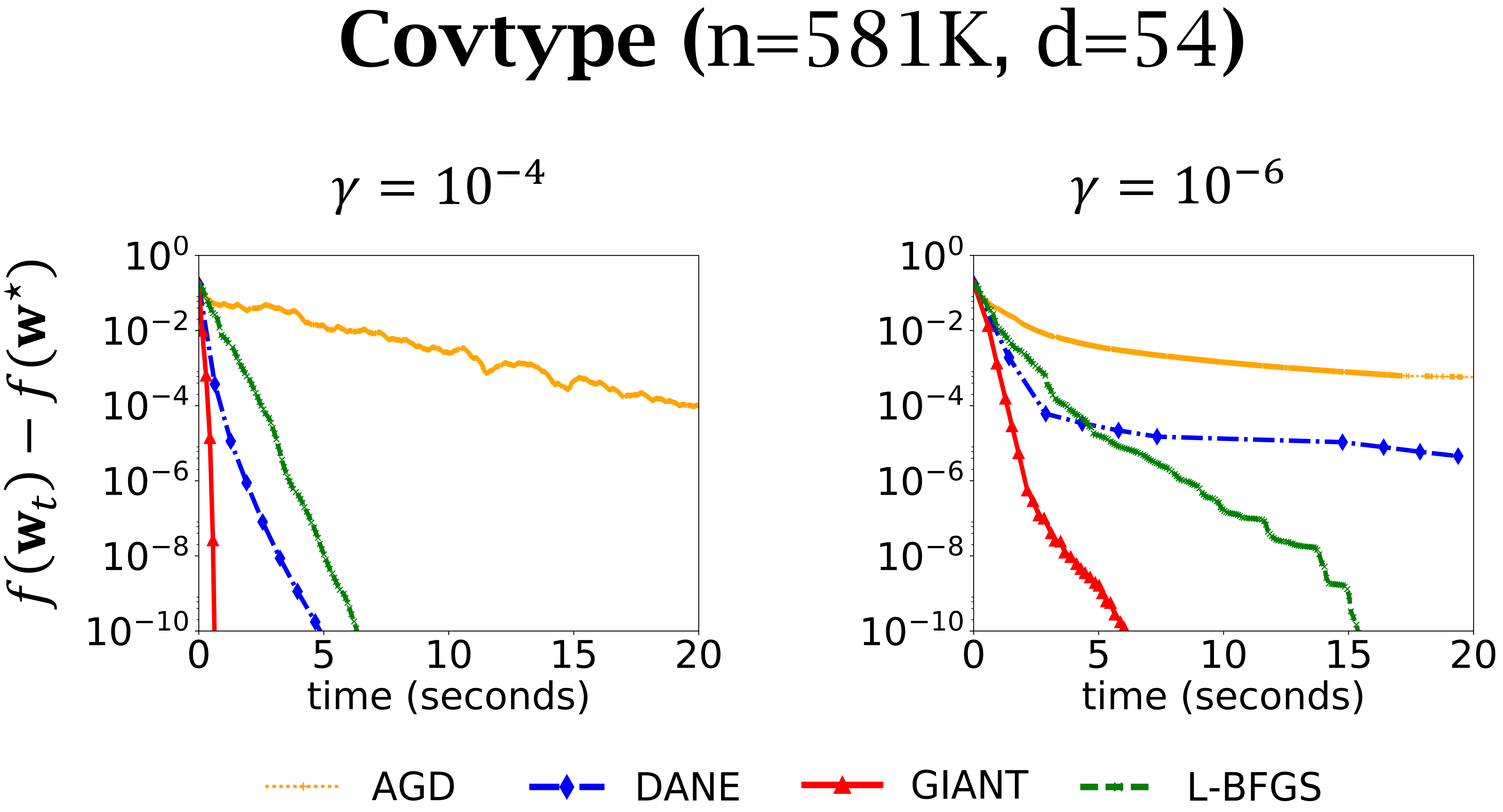
Shamir, Srebro, & Zhang. Communication Efficient Distributed Optimization using an Approximate Newton-type Method. In *ICML*, 2014.

Compared Methods

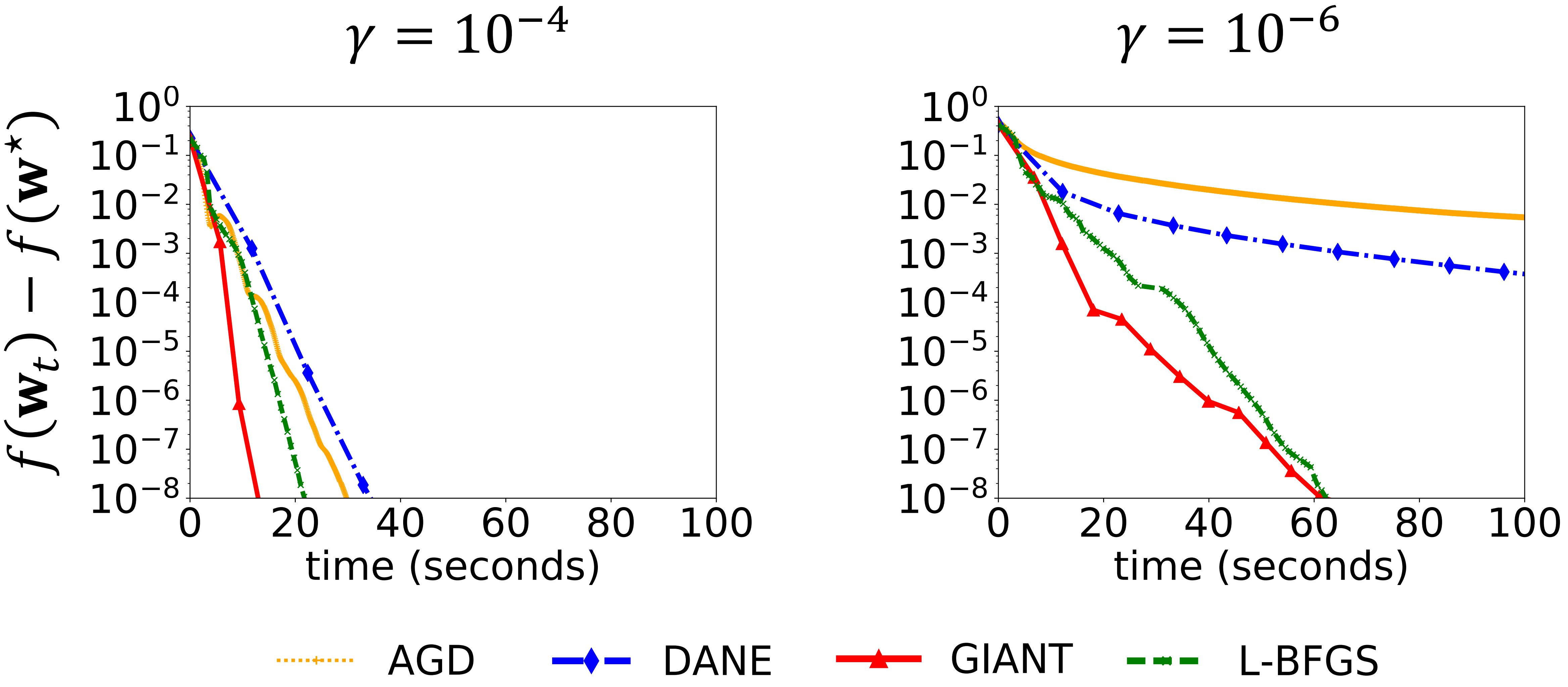
- Accelerated gradient descent (AGD)
- Limited memory BFGS
- DANE (another Newton-type method)
- GIANT
  - local solver: conjugate gradient (CG)
  - choose *max iteration of CG* from {30, 100, 300}

Compared Methods

- Accelerated gradient descent (AGD) — 2 Tuning Parameters
- Limited memory BFGS — 1 Tuning Parameter
- DANE (another Newton-type method) — 2 Tuning Parameters
- GIANT — 1 Tuning Parameter

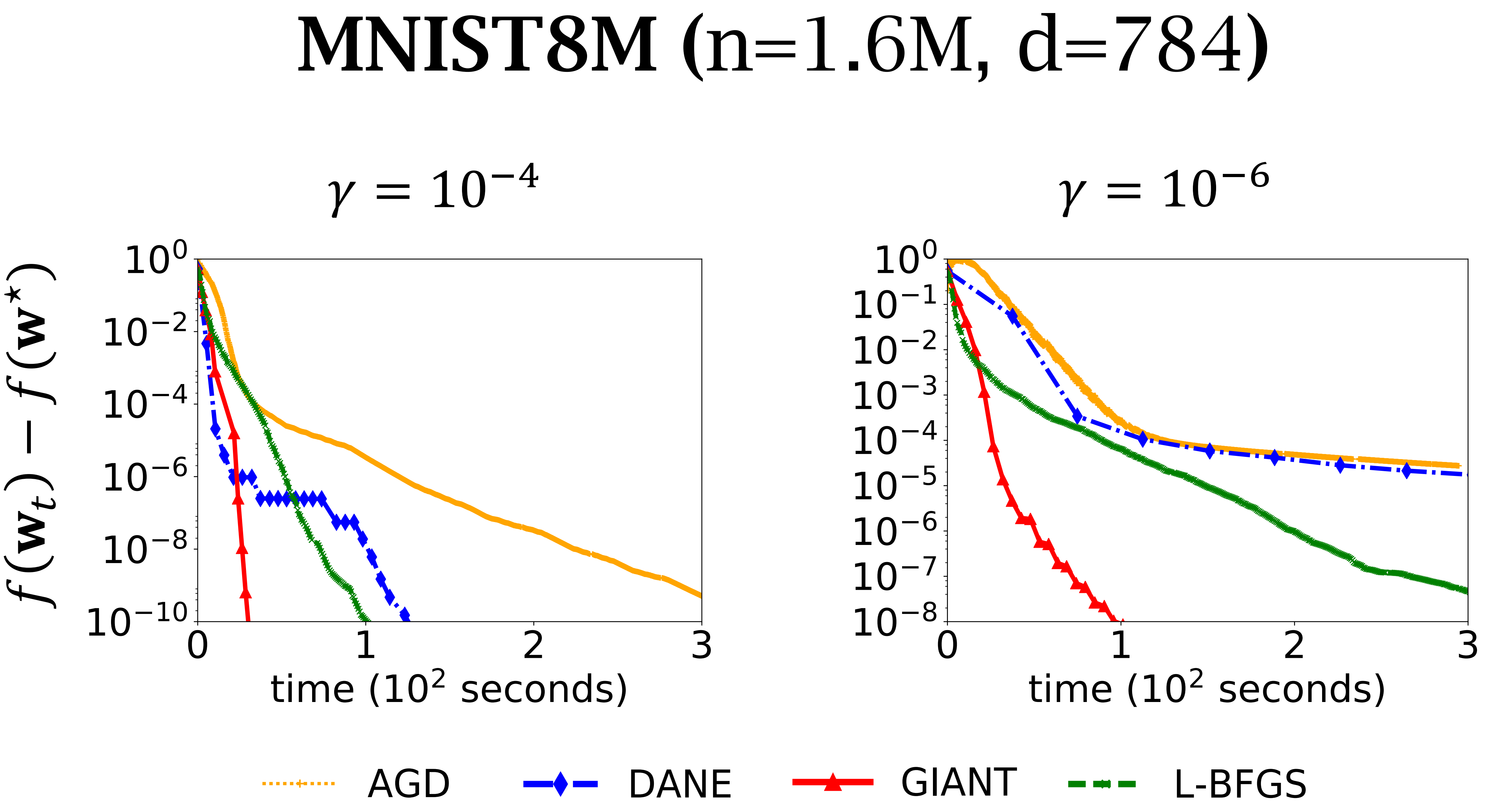

Covtype (n=581K, d=54)
$\gamma = 10^{-4}$
$\gamma = 10^{-6}$
$f(\mathbf{w}_t) - f(\mathbf{w}^\star)$
10^{0}
10^{-2}
10^{-4}
10^{-6}
10^{-8}
10^{-10}
0
5
10
15
20
time (seconds)
AGD
DANE
GIANT
L-BFGS

Epsilon (n=400K, d=2K)



MNIST8M (n=1.6M, d=784)

- Digits “4” versus “9”: 1.6M samples out of the total 8M samples


MNIST8M (n=1.6M, d=784)
γ = 10^−4
γ = 10^−6
f(w_t) − f(w*)
time (10^2 seconds)
AGD
DANE
GIANT
L-BFGS

How about Larger d?

- Split $\mathbf{X} \in \mathbb{R}^{n \times d}$ (by data) to $m = 59$ parts.
- Previously, local sample size $s = \frac{n}{m} \gg$ number of features d.
- Does GIANT work if $s \approx d$?

Random Feature Maps (RFM)

- Generate 10K *random Fourier features* *[Rahimi & Recht, 07]* of the *RBF kernel*

$$k(\mathbf{x}_i, \mathbf{x}_j) \;=\; \exp\Big(-\frac{1}{2\sigma}\|\mathbf{x}_i - \mathbf{x}_j\|_2^2\Big)$$

- Setting of RBF *kernel width parameter* σ:

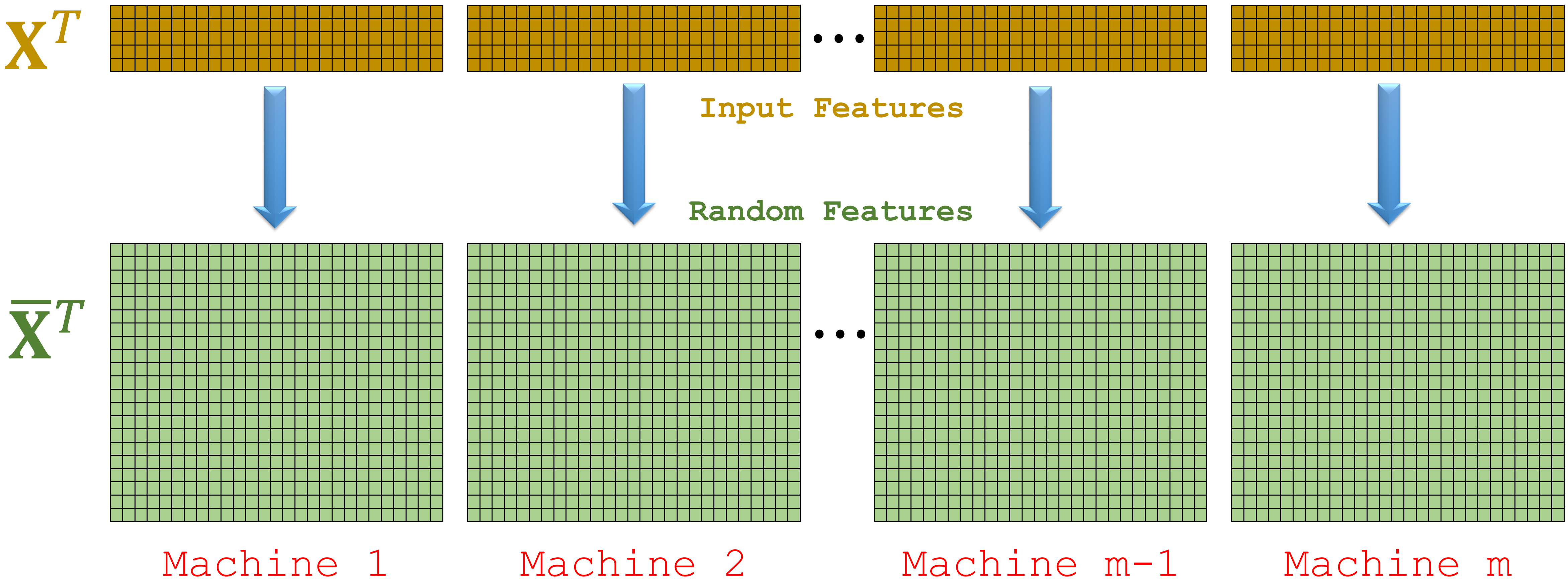
$$\sigma \;=\; \frac{1}{n^2}\sum_{i=1}^{n}\sum_{j=1}^{n}\|\mathbf{x}_i - \mathbf{x}_j\|_2^2$$

- Replace the original features by the higher-dim random features.

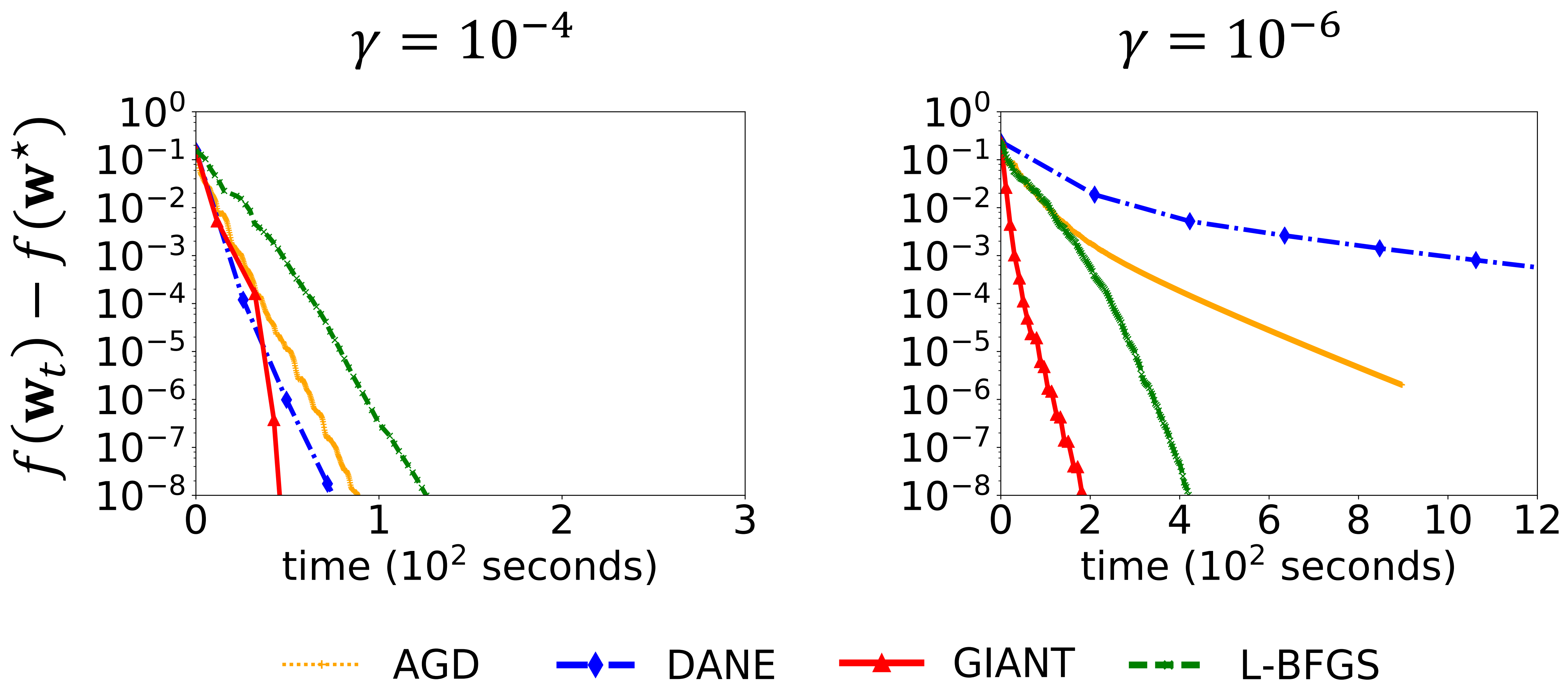
Reference:

Rahimi & Recht. Random Features for Large-Scale Kernel Machines. In *NIPS,* 2007.

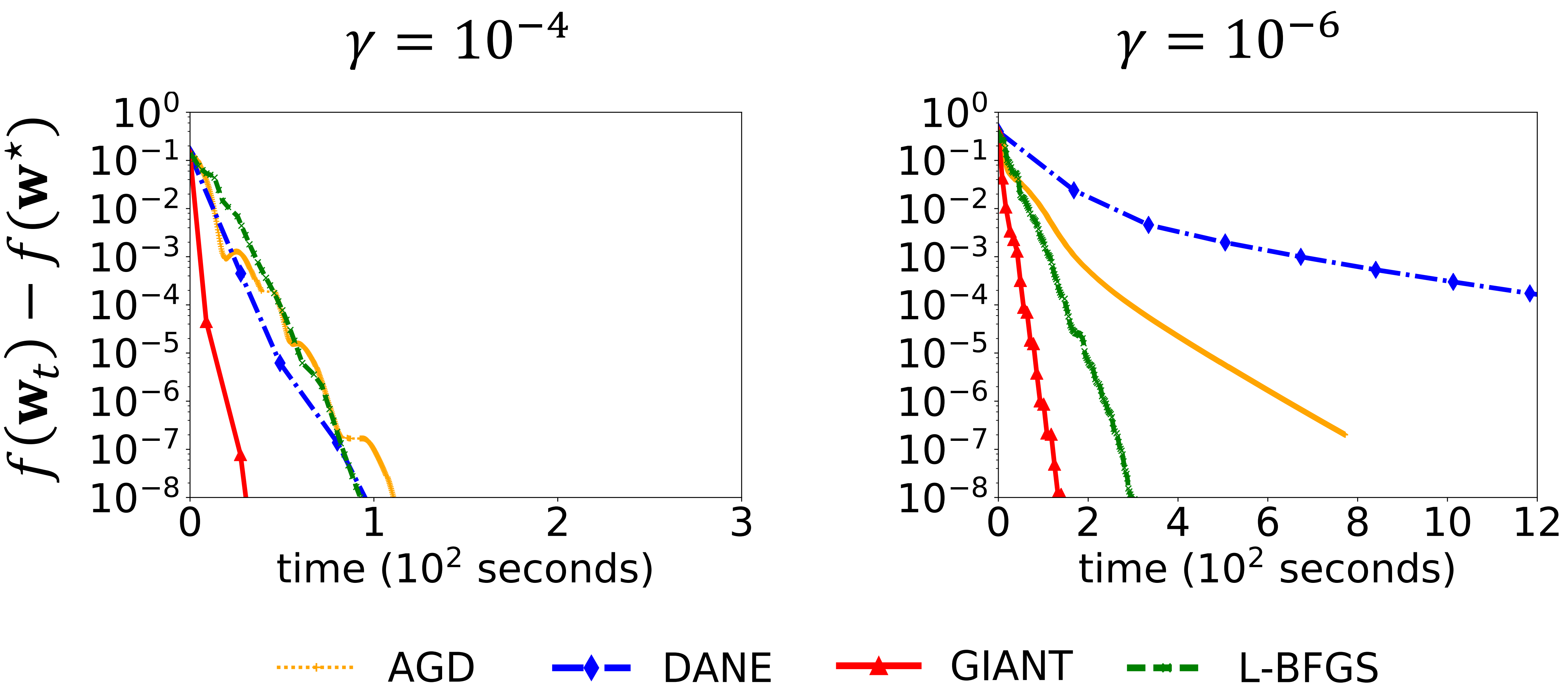
Random Feature Maps (RFM)



Covtype with RFM (n=581K, d=10K)



Epsilon with RFM (n=400K, d=10K)



MNIST8M with RFM (n=1.6M, d=10K)