Data Science in TalkingData

主讲人： TalkingData首席数据科学家 张夏天

Data in TalkingData

CHINA'S LARGEST INDEPENDENT MOBILE DATA PLATFORM

Established in 2011
Headquarters in Beijing
Three rounds of VC financing

650mln+
Monthly Active Unique Devices

100,000+
Apps with SDK Integrated

30mln
Daily Mobile Ad Clicks: China's Largest Mobile Ad Tracking Platform

200mln+
Monthly Device Panel on App Install & Usage

Challenges in TalkingData

Big Data

- Volume
- Velocity
- Variety
- Variability
- Veracity
- Unreadable Data

Various Applications

- Finance
- Retail
- Real Estate
- …

Data Science in TalkingData

Learning on Big Data

- Fregata
- Myna
- Event Data Mining

Improve Efficiency of Data Science

- Smart Data Lab
- AutoModel

Applications

- Lookalike
- Recommender System
- Demographic Cognition
- Churn Alert
- Context Awareness
- Indoor Positioning
- ……

Open

- Business Partners
- Academic Partners
- Education
- ……

Learning on Big Data

Fregata (Open Source)

- Large scale machine learning library on Spark

Myna (Open Source)

- The framework of context awareness of Andriod

Event Data Mining

- Event data management solution
- Event data & unreadable data mining

9.12 人本数据和智能

Myna：Context Awareness Framework On Smart Devices 09:30 — 10:10

让海量移动数据产生价值 15:30 — 16:10

Fregata: Machine Learning

license Apache 2.0

- Fregata is a light weight, super fast, large scale machine learning library based on Apache Spark, and it provides high-level APIs in Scala.
- More accurate: For various problems, Fregata can achieve higher accuracy compared to MLLib.
- Higher speed: For Generalized Linear Model, Fregata often converges in one data epoch. For a 1 billion X 1 billion data set, Fregata can train a Generalized Linear Model in 1 minute with memory caching or 10 minutes without it. Usually, Fregata is 10-100 times faster than MLLib.
- Parameter Free: Fregata uses GSA SGD optimization, which dosen't require learning rate tuning, because we found a way to calculate appropriate learning rate in the training process. When confronted with super high-dimension problem, Fregata calculates remaining memory dynamically to determine the sparseness of the output, balancing accuracy and efficiency automatically. Both features enable Fregata to be treated as a standard module in data processing for different problems.
- Lighter weight: Fregata just uses Spark's standard API, which allows it to be integrated into most business' data processing flow on Spark quickly and seamlessly.

large scale machine learning

Pull requests Issues Marketplace Explore

Repositories 76 | Code 499K | Commits 288 | Issues 1K | Wikis 3K | Users 35

76 repository results

Sort: Most stars

Tencent/angel — Java — ★ 2.4k

A Flexible and Powerful Parameter Server for *large-scale machine learning*

spark · machine-learning · scala · model

Updated 2 days ago

OryxProject/oryx — Java — ★ 1.2k

Oryx 2: Lambda architecture on Apache Spark, Apache Kafka for real-time *large scale machine learning*

kafka · oryx · lambda-architecture · java

Updated 4 days ago

TalkingData/Fregata — Scala — ★ 530

A light weight, super fast, *large scale machine learning* library on spark .

Updated 5 days ago

The Road To High Performance ML Algorithms: Fregata's Approach

Remove Hype Parameters

- Greedy step averaging optimization method

Low Cost Parallelization Method

- Model averaging method
- Convergence with only one scan of the whole data

Compress Model Sizes

- Expand the model capability on a single node by a factor of 1000

Greedy Step Averaging

Greedy Step Averaging: A parameter-free stochastic optimization method

Xiatian Zhang*, Fan Yao*, and Yongjun Tian*

*TalkingData Technology(Beijing)Co,.Ltd, China,
Email: {xiatian.zhang, fan.yao, yongjun.tian}@tendcloud.com

November 14, 2016

Abstract

In this paper we present the greedy step averaging(GSA) method, a parameter-free stochastic optimization algorithm for a variety of machine learning problems. As a gradient-based optimization method, GSA makes use of the information from the minimizer of a single sample's loss function, and takes average strategy to calculate reasonable learning rate sequence. While most existing gradient-based algorithms introduce an increasing number of hyper parameters or try to make a trade-off between computational cost and convergence rate, GSA avoids the manual tuning of learning rate and brings in no more hyper parameters or extra cost. We perform exhaustive numerical experiments for logistic and softmax regression to compare our method with the other state of the art ones on 16 datasets. Results show that GSA is robust on various scenarios.

Keywords Optimization, algorithm, learning rate, parameter-free, self-adaptive, averaging strategy

https://arxiv.org/abs/1611.03608

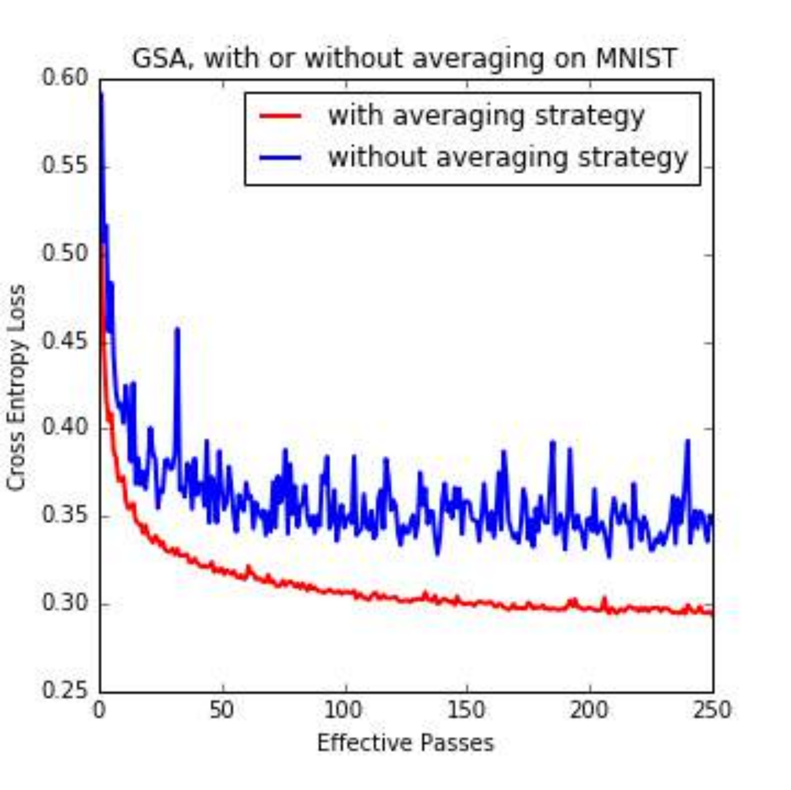
Algorithm 2 GSA algorithm in general

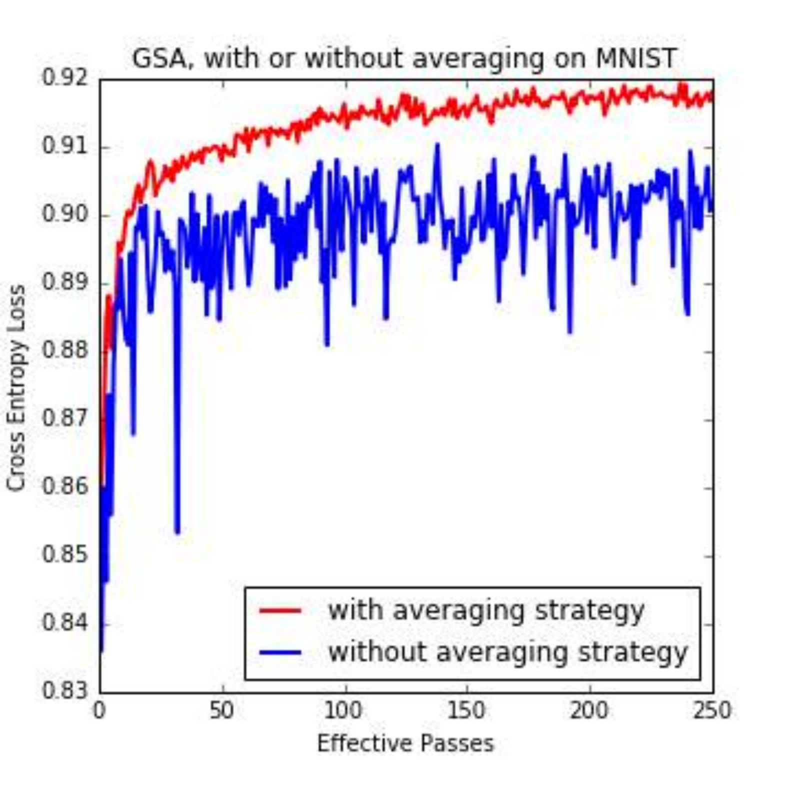
Require: Initial parameter ω_0, loss function $L(\omega) = \sum_{i=1}^{N} l_i(\omega)$

for t in $i \in [0, T]$ **do**
  Take a Training Sample (x_t, y_t);
  Compute Stochastic Gradient $g_t = \frac{\partial l_t}{\partial \omega}$;
  Compute Greedy Step Size η_t by exact line search on ${}_t(\omega_t - \eta g_t)$;
  Compute Averaged Greedy Step Size $\bar{\eta} = mean(\eta_t)$;
  Apply Update $\omega_{t+1} = \omega_t - \bar{\eta} g_t$;
end for

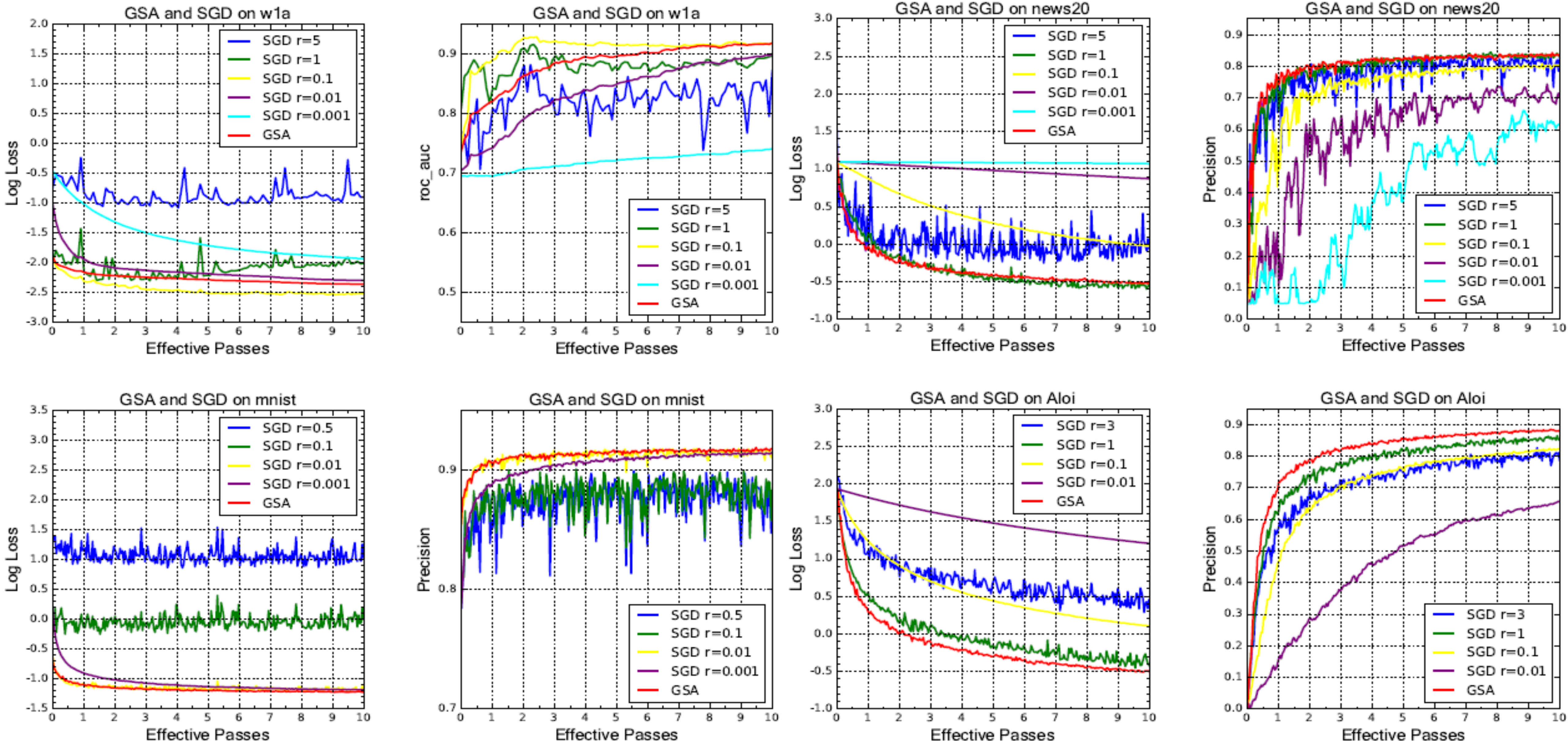
Convergence of GSA

$$E[\eta]_t = \frac{t-1}{t} E[\eta]_{t-1} + \frac{1}{t} \eta_t .$$

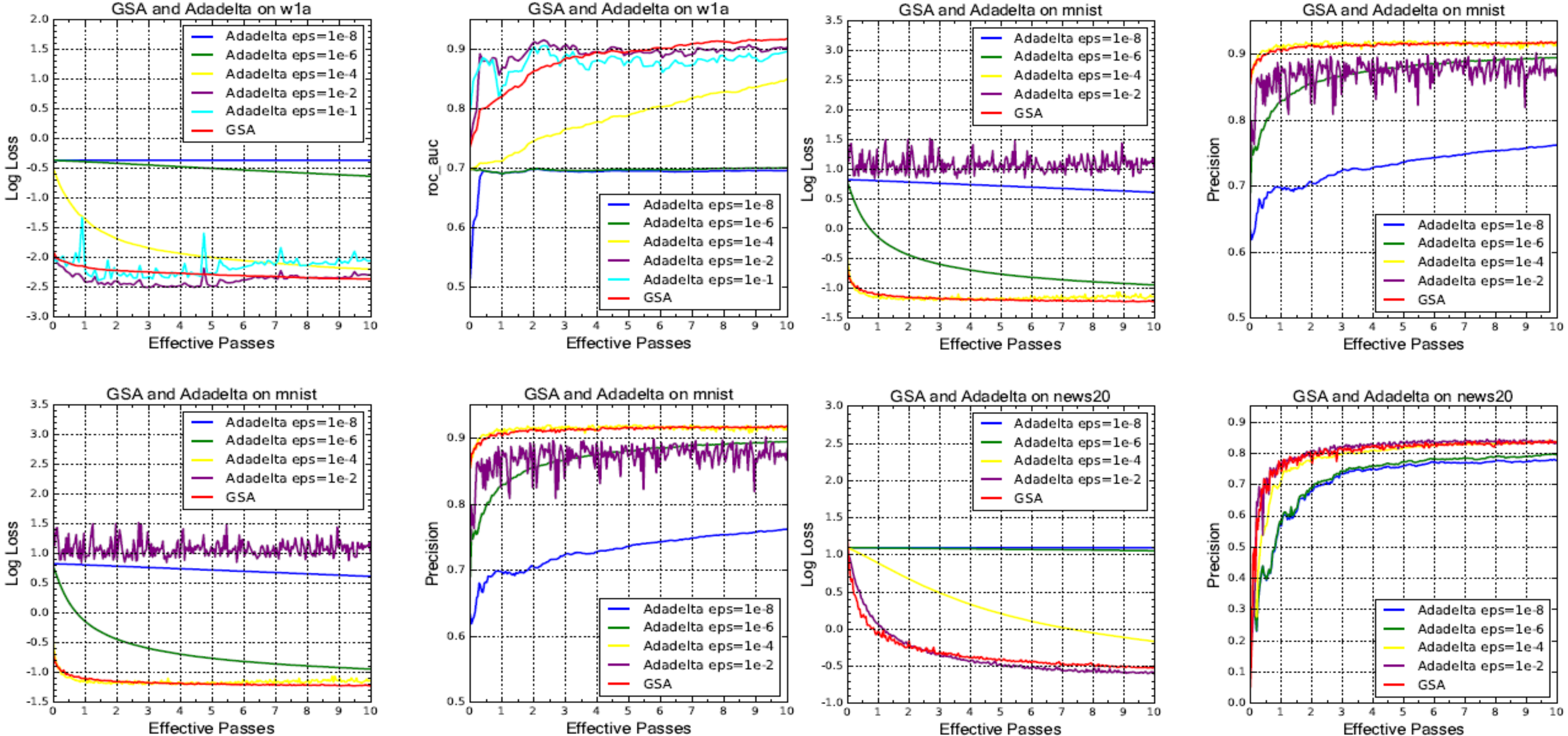




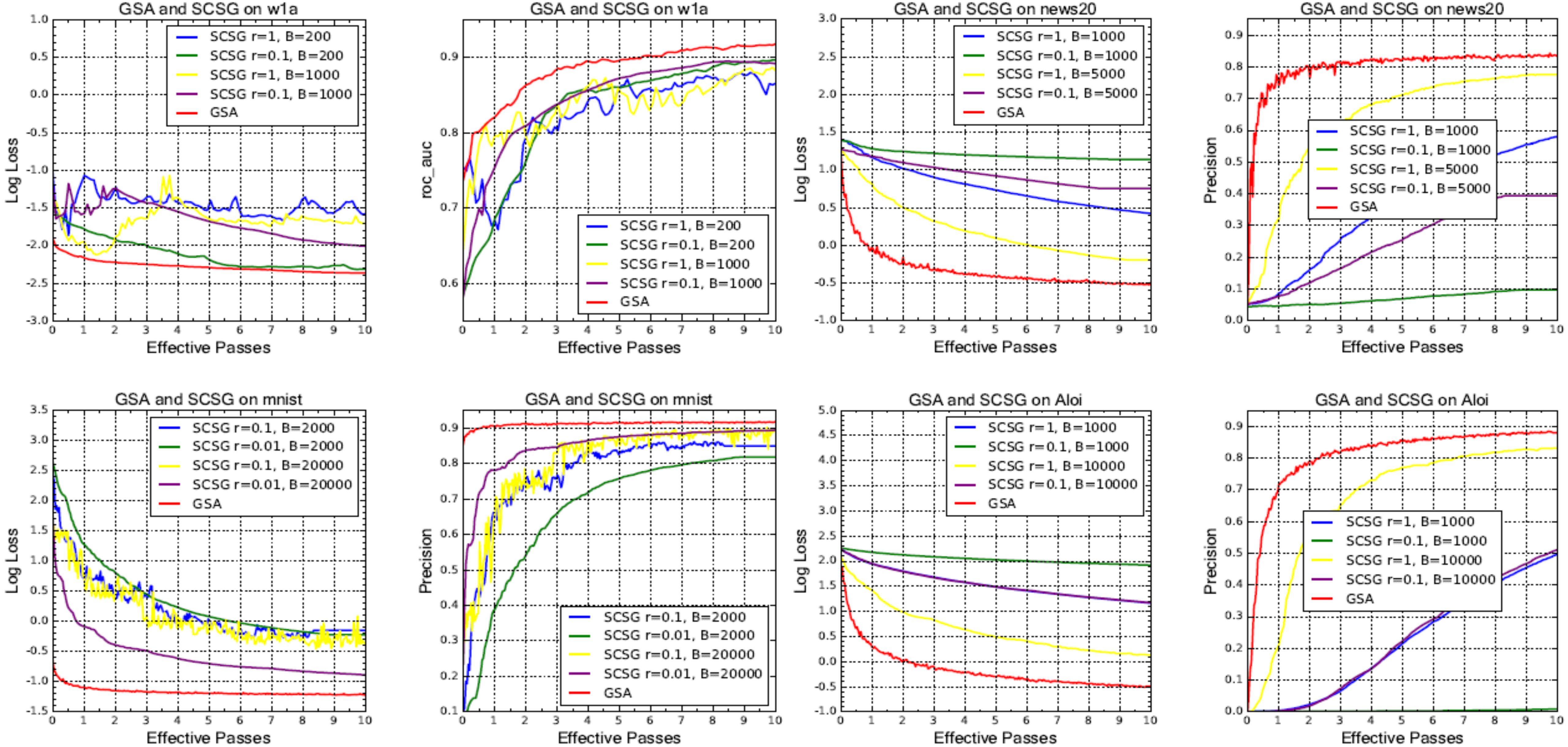
GSA vs SGD


GSA and SGD on w1a
Log Loss
Effective Passes
SGD r=5
SGD r=1
SGD r=0.1
SGD r=0.01
SGD r=0.001
GSA
GSA and SGD on w1a
roc_auc
Effective Passes
SGD r=5
SGD r=1
SGD r=0.1
SGD r=0.01
SGD r=0.001
GSA
GSA and SGD on news20
Log Loss
Effective Passes
SGD r=5
SGD r=1
SGD r=0.1
SGD r=0.01
SGD r=0.001
GSA
GSA and SGD on news20
Precision
Effective Passes
SGD r=5
SGD r=1
SGD r=0.1
SGD r=0.01
SGD r=0.001
GSA
GSA and SGD on mnist
Log Loss
Effective Passes
SGD r=0.5
SGD r=0.1
SGD r=0.01
SGD r=0.001
GSA
GSA and SGD on mnist
Precision
Effective Passes
SGD r=0.5
SGD r=0.1
SGD r=0.01
SGD r=0.001
GSA
GSA and SGD on Aloi
Log Loss
Effective Passes
SGD r=3
SGD r=1
SGD r=0.1
SGD r=0.01
GSA
GSA and SGD on Aloi
Precision
Effective Passes
SGD r=3
SGD r=1
SGD r=0.1
SGD r=0.01
GSA

GSA vs Adadelta



GSA vs SCSG


GSA and SCSG on w1a
Log Loss
Effective Passes
SCSG r=1, B=200
SCSG r=0.1, B=200
SCSG r=1, B=1000
SCSG r=0.1, B=1000
GSA
GSA and SCSG on w1a
roc_auc
Effective Passes
SCSG r=1, B=200
SCSG r=0.1, B=200
SCSG r=1, B=1000
SCSG r=0.1, B=1000
GSA
GSA and SCSG on news20
Log Loss
Effective Passes
SCSG r=1, B=1000
SCSG r=0.1, B=1000
SCSG r=1, B=5000
SCSG r=0.1, B=5000
GSA
GSA and SCSG on news20
Precision
Effective Passes
SCSG r=1, B=1000
SCSG r=0.1, B=1000
SCSG r=1, B=5000
SCSG r=0.1, B=5000
GSA
GSA and SCSG on mnist
Log Loss
Effective Passes
SCSG r=0.1, B=2000
SCSG r=0.01, B=2000
SCSG r=0.1, B=20000
SCSG r=0.01, B=20000
GSA
GSA and SCSG on mnist
Precision
Effective Passes
SCSG r=0.1, B=2000
SCSG r=0.01, B=2000
SCSG r=0.1, B=20000
SCSG r=0.01, B=20000
GSA
GSA and SCSG on Aloi
Log Loss
Effective Passes
SCSG r=1, B=1000
SCSG r=0.1, B=1000
SCSG r=1, B=10000
SCSG r=0.1, B=10000
GSA
GSA and SCSG on Aloi
Precision
Effective Passes
SCSG r=1, B=1000
SCSG r=0.1, B=1000
SCSG r=1, B=10000
SCSG r=0.1, B=10000
GSA

Parallelization

Gradient Averaging

$$w_t = w_{t-1} - \frac{\eta}{n}\sum_{i=0}^{n} \nabla Q_i(w_{t-1})$$

High cost on training stage

Model Averaging

$$w_t = \frac{1}{n}\sum_{i=0}^{n} w_{t-1,i}$$

Suitable for Spark

Score Averaging

$$y_j = \frac{1}{m}\sum_{k=0}^{m} y_{j,k}$$

High cost on scoring stage

Convergence of Model Averaging

The model averaging method can approach the optimal model for linear problems with a very large amount of training data.

On the optimality of averaging in distributed statistical learning

JONATHAN D. ROSENBLATT†
Department of Industrial Engineering and Management, Ben Gurion University of the Negev, Be'er-Sheva, Israel
†Corresponding author. Email: johnros@bgu.ac.il

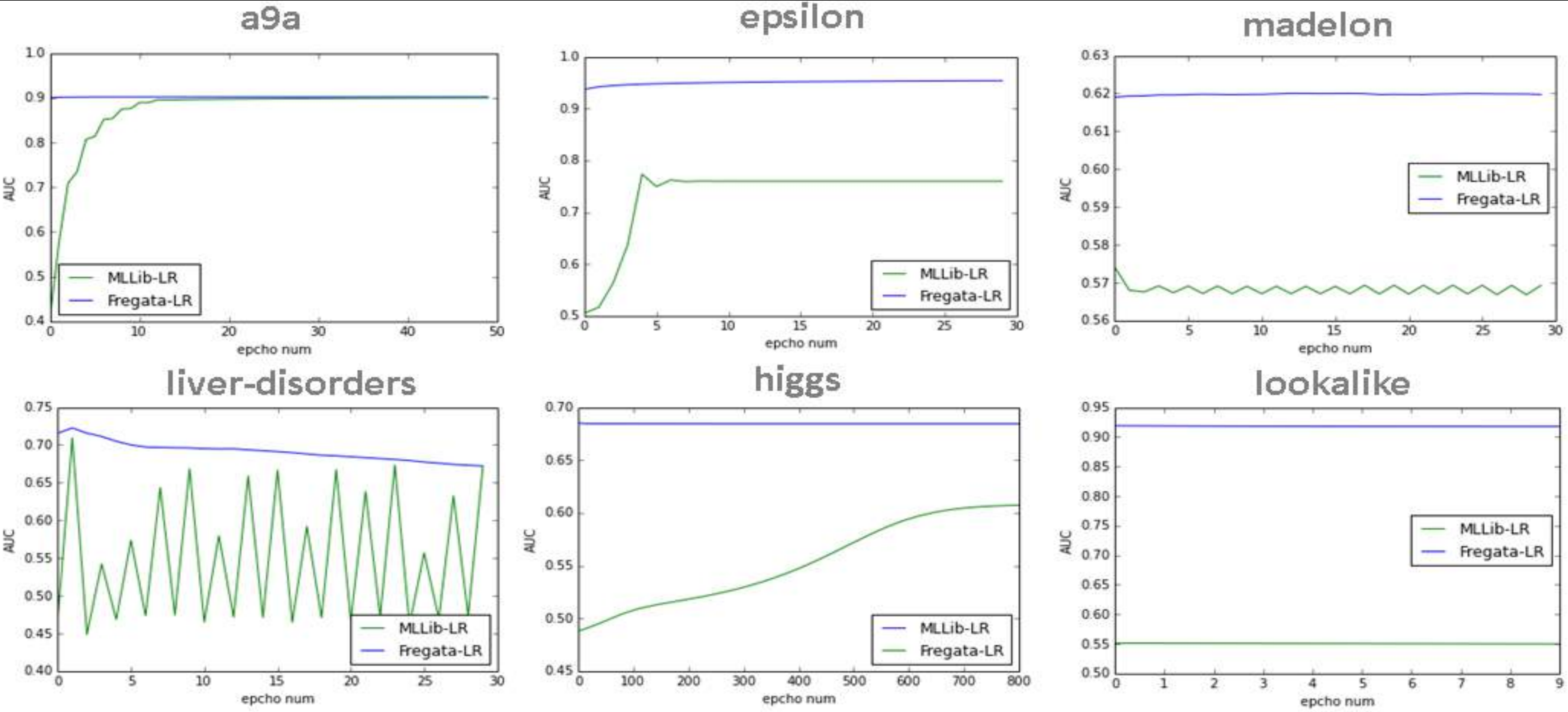
AND

BOAZ NADLER
Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel
Email: boaz.nadler@weizmann.ac.il

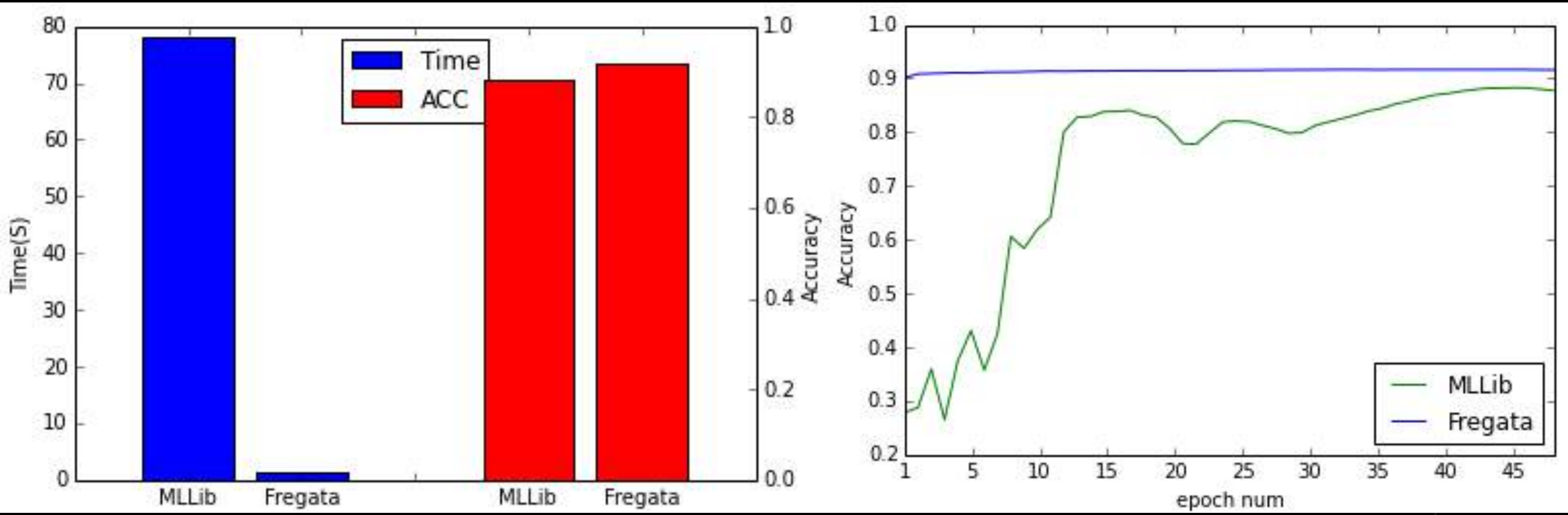
[Received on 10 June 2015; revised on 7 February 2016; accepted on 5 April 2016]

A common approach to statistical learning with Big-data is to randomly split it among m machines and learn the parameter of interest by averaging the m individual estimates. In this paper, focusing on empirical risk minimization or equivalently M-estimation, we study the statistical error incurred by this strategy. We consider two large-sample settings: first, a classical setting where the number of parameters p is fixed, and the number of samples per machine $n \to \infty$. Second, a high-dimensional regime where both $p, n \to \infty$ with $p/n \to \kappa \in (0, 1)$. For both regimes and under suitable assumptions, we present *asymptotically exact* expressions for this estimation error. In the fixed-p setting, we prove that to leading order averaging is *as accurate as* the centralized solution. We also derive the second-order error terms, and show that these can be non-negligible, notably for nonlinear models. The high-dimensional setting, in contrast, exhibits a qualitatively different behavior: data splitting incurs a first-order accuracy loss, which increases linearly with the number of machines. The dependence of our error approximations on the number of machines traces an interesting accuracy-complexity tradeoff, allowing the practitioner an informed choice on the number of machines to deploy. Finally, we confirm our theoretical analysis with several simulations.

Fregata vs. MLLib: Logistic Regression



Fregata vs. MLLib: Softmax on MNIST



TalkingData

Model Compression

Discretize parameter values by K-Means

- Typically, discretize parameter values to 128 buckets.
- Then we can use 7 bits to encode a bucket, and build a mapping index to discretize parameter values.

Compress the resulting model bitmap by Roaring Bitmaps

Model Compression: Accuracy

Data Set	Original Model		Compressed Model (128 buckets)	
	Accuracy	AUC	Accuracy	AUC
a9a	0.848	0.897	0.843	0.894
rcv1	0.947	0.987	0.938	0.983
lookalike	0.952	0.985	0.950	0.982

Model Compression: Efficiency

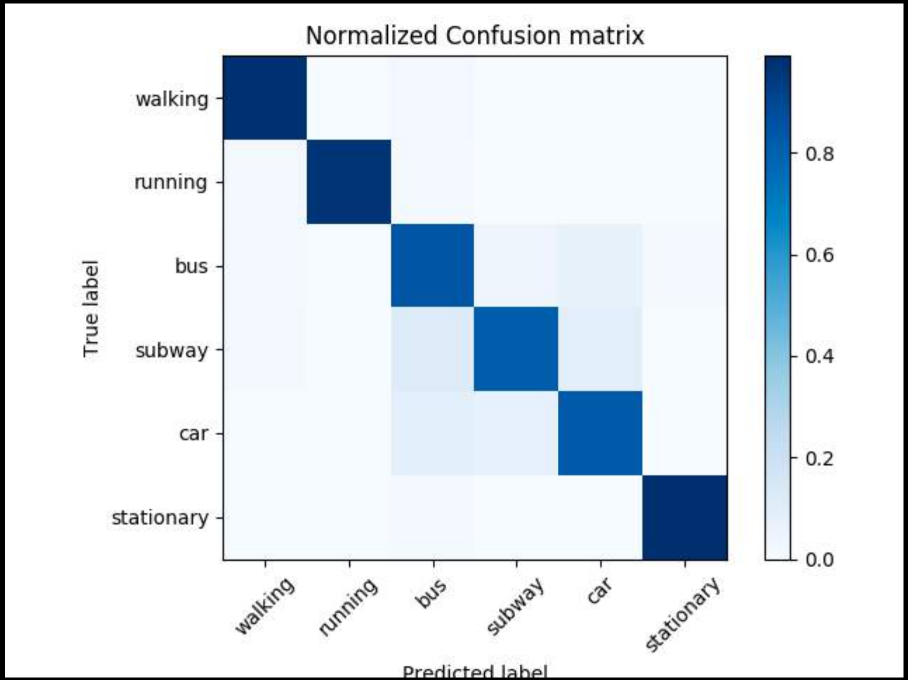
Model Size	Spark Conf（2.0）	Training Time (S)
20 Millions	48 Executors, 1 Core/Executor, 1G/Executor&Driver	469
400 Millions	48 Executors, 1 Core/Executor, 1G/Executor&Driver	455
800 Millions	48 Executors, 1 Core/Executor, 1G/Executor&Driver	449
1 Billions	48 Executors, 1 Core/Executor, 1G/Executor&Driver	487
2 Billions	48 Executors, 1 Core/Executor, 1G/Executor&Driver	449
10 Billions	48 Executors, 1 Core/Executor, 1G/Executor&Driver	473
100 Billions	48 Executors, 1 Core/Executor, 2G/Executor&Driver	481
1000 Billions	48 Executors, 1 Core/Executor, 8G/Executor&Driver	814

Myna

TalkingData Myna：
Context Awareness Framework of Andriod

Activity Types：

- Still
- Walk
- Run
- On bus
- On subway
- On car



Myna provides two sets of API：

- App developers’ API
- Data scientists’ API

Event Data Mining

Event Data Management

- Trace a device from birth to death
- More efficient store method

Event data & unreadable data mining

- Based on NLP technology

Improve Efficiency of Data Science

Smart Data Lab

- The workbench of data scientists
- Data sandbox

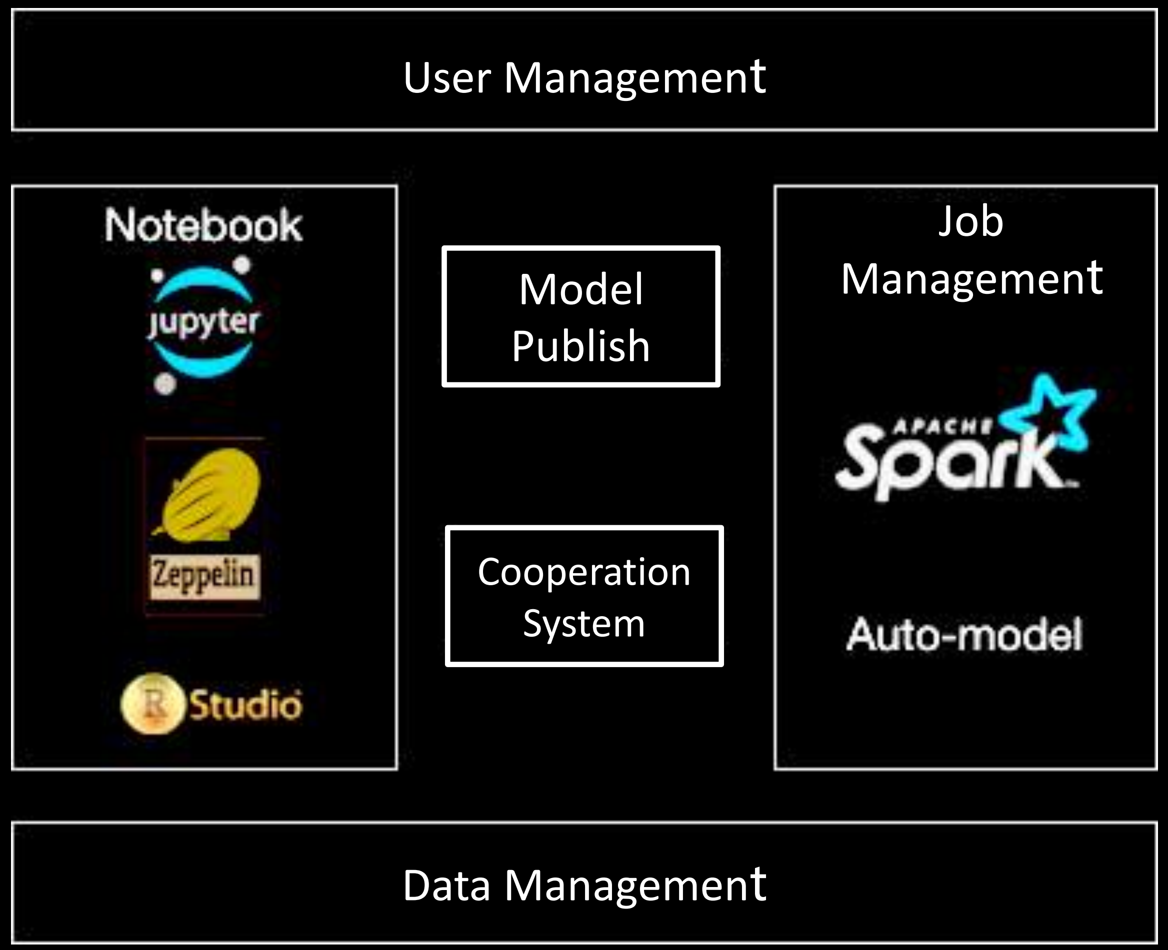
AutoModel

- Training automation tool for machine learning

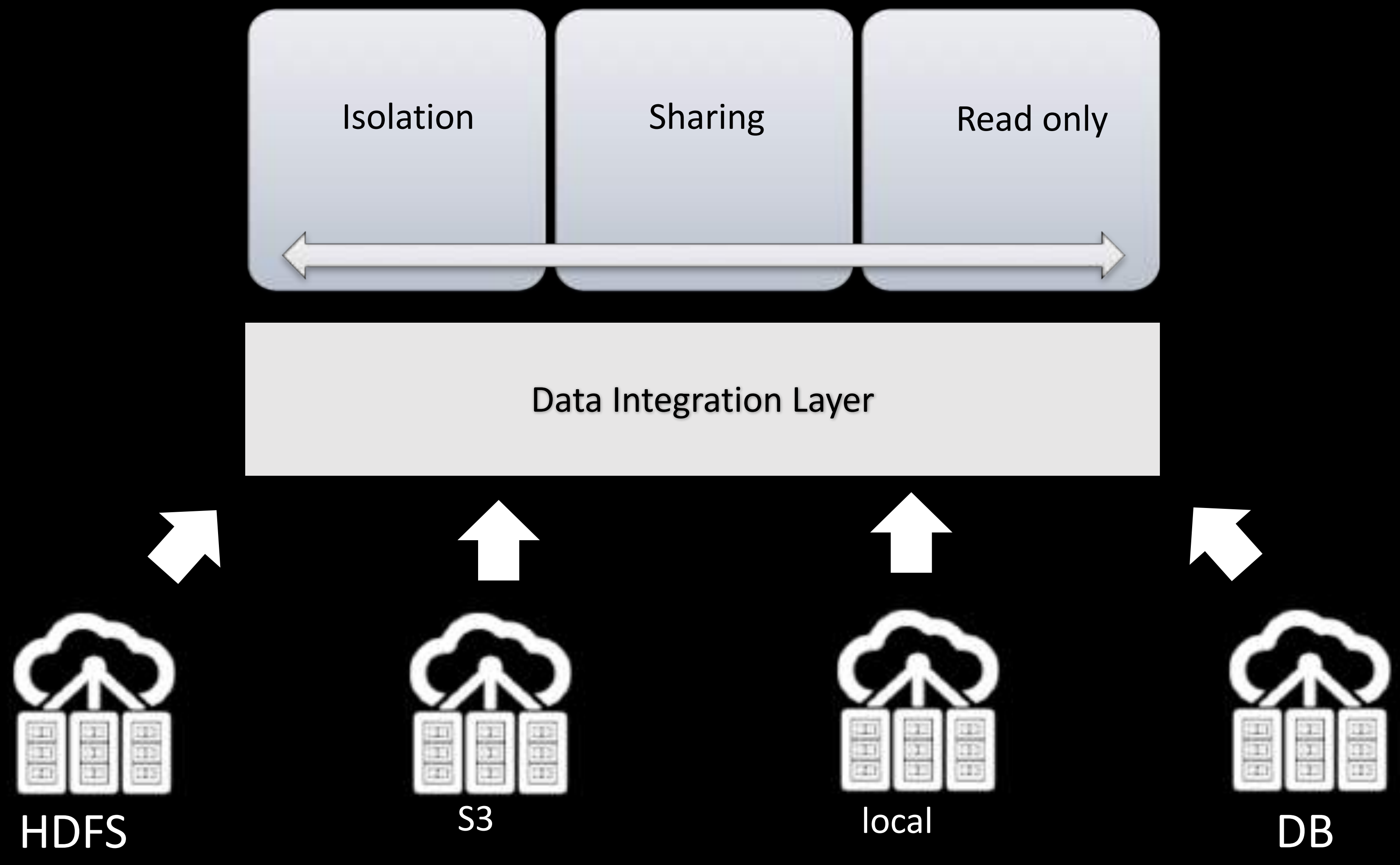
9.12 人本数据和智能

Smart Data Lab——数据科学基础设施搭建的探索与实践 16:50 — 17:30

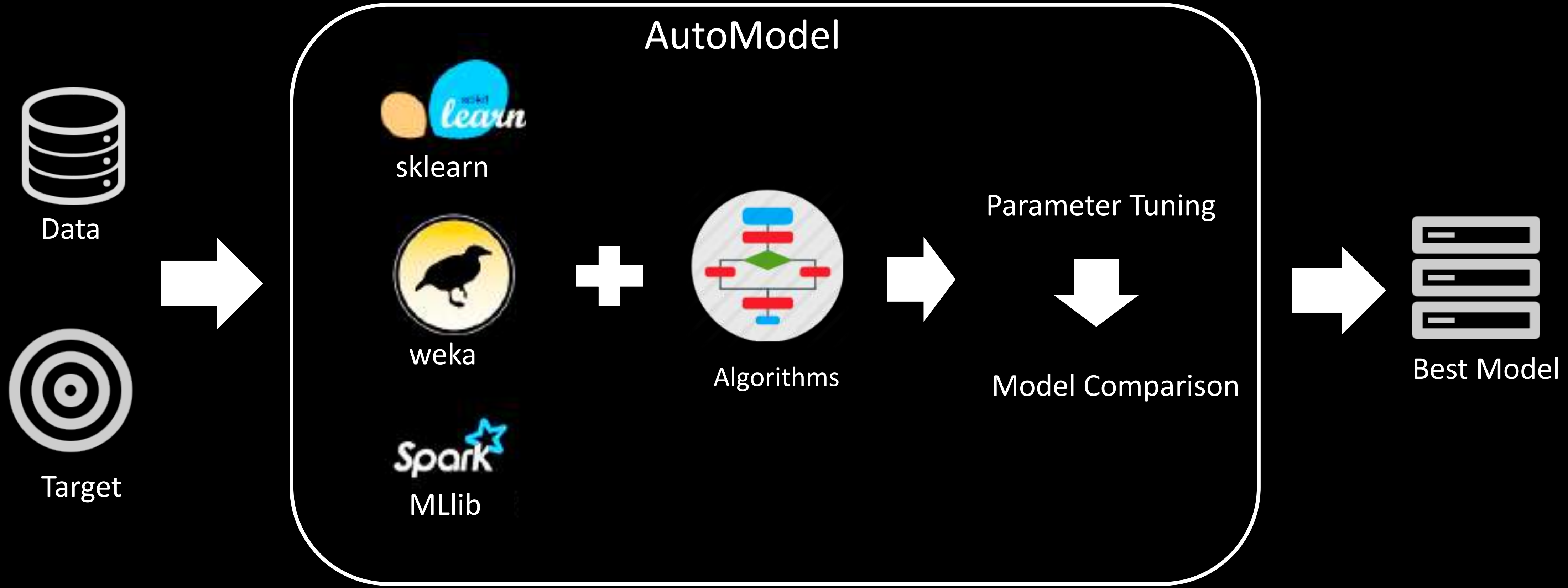
Smart Data Lab

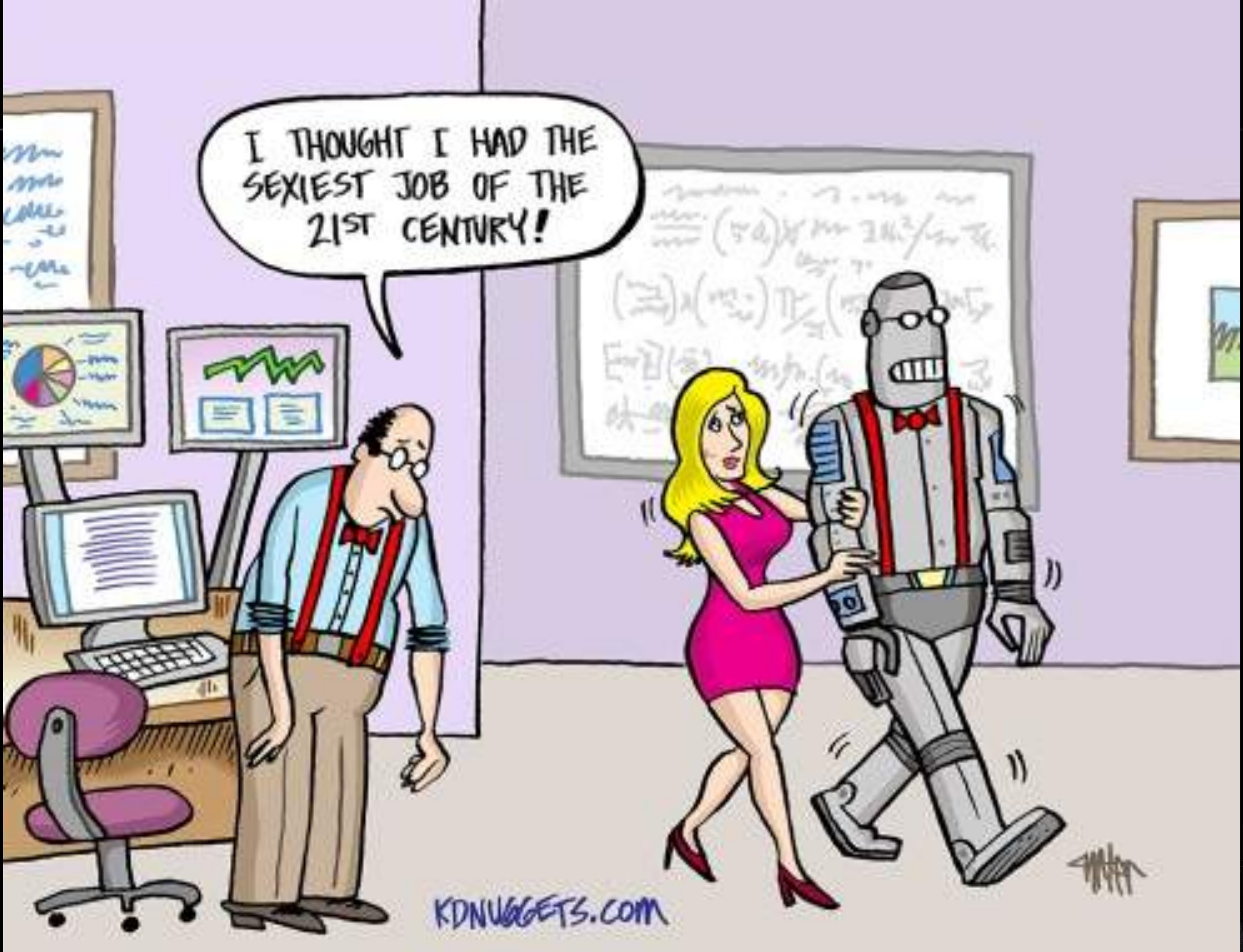


Data Sandbox



AutoModel


Data
Target
AutoModel
sklearn
weka
MLlib
Algorithms
Parameter Tuning
Model Comparison
Best Model



TalkingData

Summary – Our Mission

THANKS