Least Squares Approximation for a Distributed System

Feng Li

School of Statistics and Mathematics
Central University of Finance and Economics

Collaborators

Xuening Zhu (Fudan)

Hansheng Wang (PKU)

- Xuening Zhu is supported by the National Natural Science Foundation of China (nos. 11901105, U1811461), the Shanghai Sailing Program for Youth Science and Technology Excellence (19YF1402700), and the Fudan-Xinzailing Joint Research Centre for Big Data, School of Data Science, Fudan University.
- Feng Li's research is supported by the National Natural Science Foundation of China (no. 11501587).
- Hansheng Wang's research is partially supported by the National Natural Science Foundation of China (nos. 11831008, 11525101, 71532001, and 71332006). It is also supported in part by China's National Key Research Special Program (no. 2016YFC0207704).

Outline

Distributed statistical modelling: *Literature*

- **One-shot**: *Each worker computes the estimator in parallel. The worker communicates the result to master. The master obtains average global estimator.*
  - Zhang et al. (2013, JMLR) gives the MSE rate for the OS estimator and further used bootstrap for debiasing
  - Liu and Ihler (2014, NIPS) considers a KL-divergence based averaging
  - Lee at al. (2017, JMLR) and Battey et al. (2015) studied the sparse regression estimation using L_1 or non-convex penalties by averaging local debiased estimator
  - Fan et al. (2017) investigated the distributed PCA estimation
- **Iterative algorithms**: *Multiple rounds of communication between the master and workers.*
  - Shamir et al. (2014, ICML) proposed a Newton-type iterative method for distributed optimization.
  - Wang et al. (2017, ICML) studied distributed L_1 regularized loss minimization problem
  - Jordan et al. (2018, JASA) proposed a communication-efcient surrogate likelihood (CSL) framework to solve distributed statistical inference problems
- **Lasso estimation:** Battey et al. (2015), Wang et al. (2017), Lee at al. (2017, JMLR) and Jordan et al. (2018)

Distributed statistical modelling: *Limitations*

- Cannot conduct efficiency estimation using only one round communication
- Mostly focus on the L_1 estimation and no model selection criterion guarantee
- Require data randomly distributed

Key idea for DLSA

(1) First, we estimate the parameter θ on each worker separately by using local data on distributed workers. This can be done efficiently by using standard statistical estimation methods (e.g., maximum likelihood estimation). By assuming that the sample size of each worker is sufficiently large, the resulting estimator and its asymptotic covariance estimate should be consistent but not statistically efficient, as compared with the global estimates.

(2) Each worker passes the local estimator of θ and its asymptotic covariance estimate to the master. Because we do not consider a high-dimensional model setting, the communication cost in this regard should be negligible.

(3) A weighted least squares-type objective function can be constructed. This can be viewed as a local quadratic approximation of the global log-likelihood functions. As one can expect, the resulting estimator shares the same asymptotic covariance with the full-size MLE method (i.e., the global estimator) under appropriate regularity conditions.

DLSA

- Let $\mathcal{L}(\theta; Z)$ be a plausible twice-differentiable loss function. Define the global loss function as $\mathcal{L}(\theta) = N^{-1} \sum_{i=1}^{N} \mathcal{L}(\theta; Z_i)$, whose global minimizer is $\widehat{\theta} = \arg\min \mathcal{L}(\theta)$ and the true value is θ_0.
- We begin by decomposing and approximating the global loss function using Taylor's expansion techniques as follows:

$$\begin{aligned}\mathcal{L}(\theta) &= N^{-1} \sum_{k=1}^{K} \sum_{i \in \mathcal{S}_k} \mathcal{L}(\theta; Z_i) = N^{-1} \sum_{k=1}^{K} \sum_{i \in \mathcal{S}_k} \Big\{ \mathcal{L}(\theta; Z_i) - \mathcal{L}(\widehat{\theta}_k; Z_i) \Big\} + C_1 \\ &\approx N^{-1} \sum_{k=1}^{K} \sum_{i \in \mathcal{S}_k} (\theta - \widehat{\theta}_k)^\top \ddot{\mathcal{L}}(\widehat{\theta}_k; Z_i)(\theta - \widehat{\theta}_k) + C_2, \end{aligned} \tag{1}$$

- Intuitively, the quadratic form in (1) should be a good local approximation of the global loss function (Wang and Leng, 2007). This inspires us to consider the following weighted least squares objective function:

$$\begin{aligned}\widetilde{\mathcal{L}}(\theta) &= N^{-1} \sum_{k} (\theta - \widehat{\theta}_k)^\top \Big\{ \sum_{i \in \mathcal{S}_k} \ddot{\mathcal{L}}(\widehat{\theta}_k; Z_i) \Big\} (\theta - \widehat{\theta}_k), \\ &\stackrel{\text{def}}{=} \sum_{k} (\theta - \widehat{\theta}_k)^\top \alpha_k \widehat{\Sigma}_k^{-1} (\theta - \widehat{\theta}_k),\end{aligned}$$

DLSA

- This leads to a weighted least squares estimator (WLSE), which takes an analytical form as follows:

$$\tilde{\theta} = \arg\min_{\theta} \widetilde{\mathcal{L}}(\theta) = \Big(\sum_k \alpha_k \widehat{\Sigma}_k^{-1} \Big)^{-1} \Big(\sum_k \alpha_k \widehat{\Sigma}_k^{-1} \widehat{\theta}_k \Big).$$

- For simultaneous variable selection and parameter estimation, we follow the idea of Wang and Leng (2007) and consider the adaptive Lasso objective function on the master (Zou, 2006; Zhang and Lu, 2007),

$$Q_\lambda(\theta) = \widetilde{\mathcal{L}}(\theta) + \sum_j \lambda_j |\theta_j|. \tag{2}$$

- Specifically, to consistently recover the sparsity pattern, we consider a distributed Bayesian information criterion (DBIC)-based criterion as follows:

$$\text{DBIC}_\lambda = (\widetilde{\theta}_\lambda - \widetilde{\theta})^\top \widehat{\Sigma}^{-1} (\widetilde{\theta}_\lambda - \widetilde{\theta}) + \log N \times df_\lambda / N, \tag{3}$$

DLSA algorithm

Input: The model function for modelling each partitioned dataset

Output: The weighted least squares estimator $\tilde{\theta}$, covariance matrix $\widehat{\Sigma}$, DBIC DBIC_λ

Steps:

Step (1). Pre-determine the overall cluster available memory as M_{ram}, the total number of CPU cores as C_{cores}, and the total data size to be processed as D_{total};

Step (2) Define the number of batched chunks N_{chunks} to allow for out-of-memory data processing. We recommend that N_{chunks} be at least greater than $3 \times D_{total}/M_{ram}$ in a Spark system.

Step (3). Define the number of partitions $P_{partition} = D_{total}/(N_{chunks} \times C_{cores})$.

Step (4). Define a *model function* whereby the input is an $n \times (p+2)$ Python Pandas DataFrame containing the response variable, covariates and partition id, and the output is a $p \times (p+1)$ Pandas DataFrame whereby the first column is $\tilde{\theta}_k$ and the remaining columns store $\tilde{\Sigma}_k^{-1}$.

Step (5).

for *i in 1:*N_{chunks} **do**

- **(a)**. Transfer the data chunk to Spark's distributed DataFrame if the data are stored in another format.
- **(b)**. Randomly assign an integer partition label from $\{1, \ldots, P_{partition}\}$ to each row of the Spark DataFrame.
- **(c)**. Repartition the DataFrame in the distributed system if the data are not partitioned by the partition label.
- **(d)**. Group the Spark DataFrames by the assigned partition label.
- **(e)**. Apply the model function to each grouped dataset with Spark's *Grouped map Pandas UDFs* API and obtain a $(pP_{partition}) \times (p+1)$ distributed Spark DataFrame R_i.

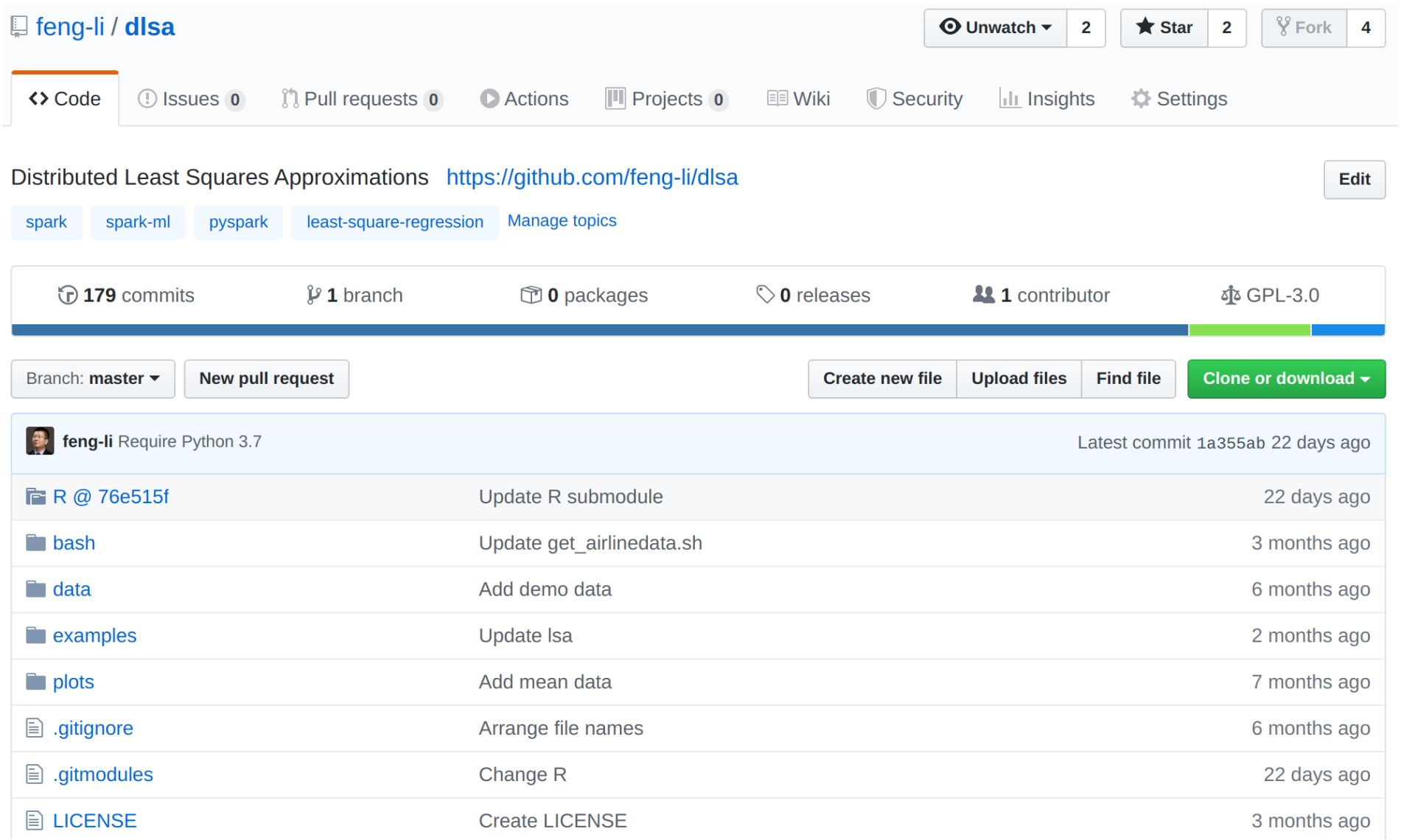
end

Step (6). Aggregate R_i over both partitions and chunks and return the $p \times (p+1)$ matrix R_{final}.

Step (7). Return $\tilde{\theta}$, $\hat{\Sigma}$, and DBIC_λ.

- Because the final step in the DLSA algorithm is carried out on the master node and because data transformation from worker nodes to the master node is required, a special tool called "Apache Arrow" (https://arrow.apache.org/) is plugged-in to our system to allow efficient data transformation between Spark's distributed DataFrame and Python's Pandas DataFrame.

DLSA API for Spark



DLSA contributions

- **Flexible**: able to handle a large class of regression problems (e.g., LM, GLM, Cox's Model)
- **High efficiency**: Match global efficiency & especially useful when data is heterogenously distributed.
- **Easy variable selection**: Oracle property can be obtained. A BIC type criterion can be used on master and no further communication is needed.
- **Smart computation**: Analytical solution can be obtained and an LARS algorithm can be applied on master.
- **New distributed statistical API to Spark**: We improved the Spark's current machine learning API to allow for efficient out-of-memory modelling within Spark.

Airline Data

Variable	Description	Variable used in the model
Delayed	Whether the flight is delayed, 1 for Yes; 0 for No.	Used as the response variable
Year	Year between 1987 and 2008	Used as numerical variable
Month	Which month of the year	Converted to 11 dummies
DayofMonth	Which day of the month	Used as numerical variable
DayofWeek	Which day of the week	Converted to 6 dummies
DepTime	Actual departure time	Used as numerical variable
CRSDepTime	Scheduled departure time	Used as numerical variable
CRSArrTime	Scheduled arrival time	Used as numerical variable
ElapsedTime	Actual elapsed time	Used as numerical variable
Distance	Distance between the origin and destination in miles	Used as numerical variable
Carrier	Flight carrier code for 29 carriers	Top 7 carries converted to 7 dummies
Destination	Destination of the flight (total 348 categories)	Top 75 destination cities converted to 75 dummies
Origin	Departing origin (total 343 categories)	Top 75 origin cities converted to 75 dummies

Computational details

- Ultimately, a total of 181 variables are used in the model. The total sample size is **113.9 million** observations.
- Raw dataset **12 GB** on a hard drive. After the dummy transformation, the overall in-memory size is over **52 GB**, even if all the dummies are stored in a sparse matrix format. Thus, this dataset can hardly be handled by a single computer.
- Much more memory (typically > 128 GB with a double-precision floating-point format) is needed for operating on matrices of such a huge size.
- The builtin distributed SGD algorithm (implemented in the `spark.ml.classification.LogisticRegression` module) simply fails with the aforementioned cluster due to the well-known out-of-memory problem in Spark.

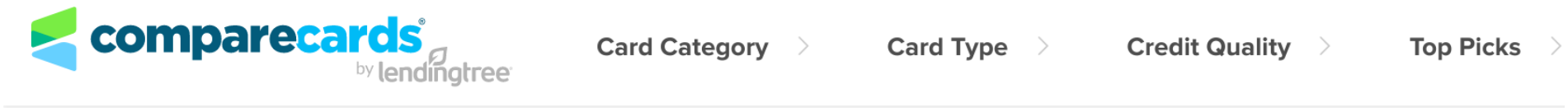
Efficiency and cost effectiveness

- A standard industrial-level architecture Spark-on-YARN cluster on the Alibaba cloud server consists of one master node and two worker nodes. Each node contains 64 virtual cores, 64 GB of RAM and two 80 GB SSD local hard drives. (cost **300 RMB per day**)
- Finally, we find that 26.2 minutes are needed for DLSA, and 25.3 minutes are needed for the OS method.
- We remark that the computing time for the MLE includes the data shuffling time with our DLSA and OS algorithms. The corresponding log-likelihood values are -1.62×10^8 and -1.65×10^8, respectively.
- Comparing these results with that of the traditional MLE, we find that the traditional MLE is extremely difficult to compute. It takes more that 15 hours and obtains an inferior result (i.e., smaller log-likelihood value) (cost **187 RMB**). In contrast, the log-likelihood value of the DLSA is the best.
- That means we have saved 97% computational power. (cost **only 6 RMB)**.

Logistic Regression

	Intercept	Year	CRSArrTime	Distance	CRSDepTime	DayofMonth	ElapsedTime	DepTime			
	−0.30	6.12	−0.13	−5.68	−0.12	−0.06	0.08	0.15			
Month	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
	−0.13	−0.01	0.03	**6.03**	−0.28	−0.09	−0.02	−0.07	0.8	0.4	0.19
Day of Week	Tue	Wed	Thu	Fri	Sat	Sun					
	0.6	0.85	−0.57	0.39	0.22	0.26					
Carrier	AA	CO	DL	NW	UA	US	WN				
	0.49	−0.16	0.18	0.39	0.7	−0.43	−0.6				
A	ABQ	ANC	ATL	AUS	BDL	BHM	BNA	BOS	BUF	BUR	BWI
O	−0.46	−0.14	−0.02	0.39	0.38	0.13	−0.17	0.13	−0.55	−0.01	0.54
D	−0.61	0.78	−0.84	−0.73	−0.74	−0.46	−0.28	0.51	−0.50	−0.90	0.69
A	CLE	CLT	CMH	CVG	DAL	DAY	DCA	DEN	DFW	DTW	ELP
O	0.49	−0.87	−0.80	0.29	−0.08	0.40	−0.07	−0.22	0.53	0.32	−0.56
D	0.14	−0.46	0.64	−1.60	−0.72	−0.79	1.07	0.37	−1.04	−0.19	−0.54
A	EWR	FLL	GSO	HNL	HOU	IAD	IAH	IND	JAX	JFK	LAS
O	0.21	−0.99	0.11	0.19	−0.51	−0.29	0.82	0.46	−0.71	0.07	−0.94
D	−0.44	−1.70	0.77	0.79	−1.62	0.00	−0.54	−0.27	−0.12	−0.70	−0.29
A	LAX	LGA	MCI	MCO	MDW	MEM	MIA	MKE	MSP	MSY	OAK
O	0.38	0.87	0.61	0.52	−0.48	0.71	0.39	−0.64	−0.10	0.21	−1.04
D	−0.20	0.13	1.00	0.30	0.10	1.16	−0.11	−0.83	−0.61	0.97	−1.47
A	OKC	OMA	ONT	ORD	ORF	PBI	PDX	PHL	PHX	PIT	PVD
O	−0.71	−0.85	1.58	−0.41	−0.67	−1.12	−0.13	0.94	0.37	0.18	−0.42
D	0.69	−0.74	−1.49	−0.72	−0.79	0.99	−0.64	−0.54	0.83	0.33	−0.40
A	RDU	RIC	RNO	ROC	RSW	SAN	SAT	SDF	SEA	SFO	SJC
O	−5.80	0.02	−0.06	0.20	0.05	0.00	−0.04	0.00	0.24	0.24	0.14
D	0.97	0.83	0.71	−1.07	0.37	1.15	0.78	−0.35	−0.79	−0.51	0.21
A	SJU	SLC	SMF	SNA	STL	SYR	TPA	TUL	TUS		
O	−0.14	0.59	1.59	−0.15	0.14	−1.21	−0.28	−0.36	−1.51		
D	0.84	0.09	−0.03	0.02	0.38	−0.06	0.31	0.43	0.03		

Interesting findings



Key takeaways

- **Big Apple, big delays:** Newark-Liberty remains America's most delay-ridden airport in the summer months. One out of three planes landed late between 2009 and 2018. Summer visitors to New York won't get much relief by booking flights to other airports. The summer on-time arrival rate at LaGuardia (69%, second worst in the nation) and JFK (about 72%, fourth worst) weren't much better. (San Francisco came in third at about 70%.)
- **Airports in western cities dominate the top of the list:** 87.2% of vacationers to Honolulu can expect to land on-time, making Hawaii that much more of an ideal destination. Salt Lake City and Orange County, Calif., came in second and third, with 10-year average on-time arrivals of about 86% and 83%, respectively. Phoenix, Seattle and Portland, Oregon were next on the list.
- **Historically, June is the worst month for travel:** It's a hugely popular time for travel, but that also brings problems. The biggest arrival delays of the year happen in June for 30 out of 50 largest airports in the U.S.
- **The rest of the summer isn't great, either:** It's not just June. More than two-thirds of the 50 airports we reviewed have the worst 10-year average on-time arrival percentage of the year between June and August.
- **Most airports are getting (a little) worse:** More than half (27 of 50) of the airports we reviewed saw more delays from 2017 to 2018. Much of the movement we saw was small, however, with 17 of the 50 airports' rates moving up or down by 1 percentage point or less.

Working in progress

- Distributed modeling with time series data.
- Distributed variational inference with least square as proposals.
- Bayesian updating on a distributed system with streaming data flow.

References

Battey, H., Fan, J., Liu, H., Lu, J., and Zhu, Z. (2015), "Distributed estimation and inference with statistical guarantees," *arXiv preprint arXiv:1509.05457*.

Fan, J., Wang, D., Wang, K., and Zhu, Z. (2017), "Distributed estimation of principal eigenspaces," *arXiv preprint arXiv:1702.06488*.

Jordan, M. I., Lee, J. D., and Yang, Y. (2018), "Communication-efficient distributed statistical inference," *Journal of the American Statistical Association*, 1–14.

Liu, Q. and Ihler, A. T. (2014), "Distributed estimation, information loss and exponential families," in *Advances in neural information processing systems*, pp. 1098–1106.

Shamir, O., Srebro, N., and Zhang, T. (2014), "Communication-efficient distributed optimization using an approximate newton-type method," in *International conference on machine learning*, pp. 1000–1008.

Wang, H. and Leng, C. (2007), "Unified LASSO estimation by least squares approximation," *Journal of the American Statistical Association*, 102, 1039–1048.

Wang, J., Wang, W., and Srebro, N. (2017), "Memory and communication efficient distributed stochastic optimization with minibatch-prox," *arXiv preprint arXiv:1702.06269*.

Zhang, H. H. and Lu, W. (2007), "Adaptive Lasso for Cox's proportional hazards model," *Biometrika*, 94, 691–703.

Zhang, Y., Duchi, J. C., and Wainwright, M. J. (2013), "Communication-efficient algorithms for statistical optimization," *The Journal of Machine Learning Research*, 14, 3321–3363.

Zou, H. (2006), "The adaptive lasso and its oracle properties," *Journal of the American Statistical Association*, 101, 1418–1429.

Zou, H. and Li, R. (2008), "One-step sparse estimates in nonconcave penalized likelihood models," *Annals of statistics*, 36, 1509.

Thank you!

feng.li@cufe.edu.cn

http://feng.li/