FFTLasso: Large-Scale LASSO in the Fourier Domain

Adel Bibi, Hani Itani, Bernard Ghanem

King Abdullah University of Science and Technology (KAUST)

July 21-26 HONOLULU 2017

Abstract

- The LASSO sparse-representation problem has proved to be a powerful tool for applications ranging from signal processing and information theory to computer vision and machine learning.
- Solving large scale LASSOs is often a difficulty due to the incompatibility of solvers to scale efficiently.
- This paper proposes a novel circulant reformulation of the LASSO that lifts the problem to a higher dimension where ADMM can be applied efficiently to its dual form.
- The new formulation updates all the variables with 1D FFTs as the most expensive operator followed by only elementwise operations.
- Neither system linear solvers nor matrix-vector multiplication is required.
- The proposed method can be trivially parallelized over multiple GPUs.

Solvers

- It was shown in previous works [1] that DL-ADMM (applying ADMM to the dual form) be one of the fastest solvers available.

Primal Form: $$\min_{\mathbf{c}} \; \|\mathbf{A}\mathbf{c}-\mathbf{b}\|_2^2+\lambda\|\mathbf{c}\|_1$$

Dual Form: $$\min_{\Psi} \; \frac{1}{4}\|\Psi\|_2^2+\Psi^\top\mathbf{b} \quad \text{s.t.} \quad \|\mathbf{A}^\top\Psi\|_\infty \le \lambda$$

Algorithm 1: PL-ADMM

Input : $\mathbf{b}, \mathbf{A}, \mathbf{c}=\mathbf{0}_n, \mathbf{z}=\mathbf{0}_n, \Psi=\mathbf{0}_m, \lambda, u_1, \gamma>1.$
Output: $\mathbf{c}$
while not converged **do**
- **c update:** solve $(2\mathbf{A}^\top\mathbf{A}+u_k\mathbf{I}_n)\mathbf{c}_{k+1}=2\mathbf{A}^\top\mathbf{b}-\Psi_k+u_k\mathbf{z}_k$
- **z update:** $\mathbf{z}_{k+1}=\text{soft}(\mathbf{c}_{k+1}+\Psi_k/u_k).$
- **Ψ update:** $\Psi_{k+1}=\Psi_k+u_k(\mathbf{c}_{k+1}-\mathbf{z}_{k+1})$
- $u_{k+1}=\gamma u_k$

end

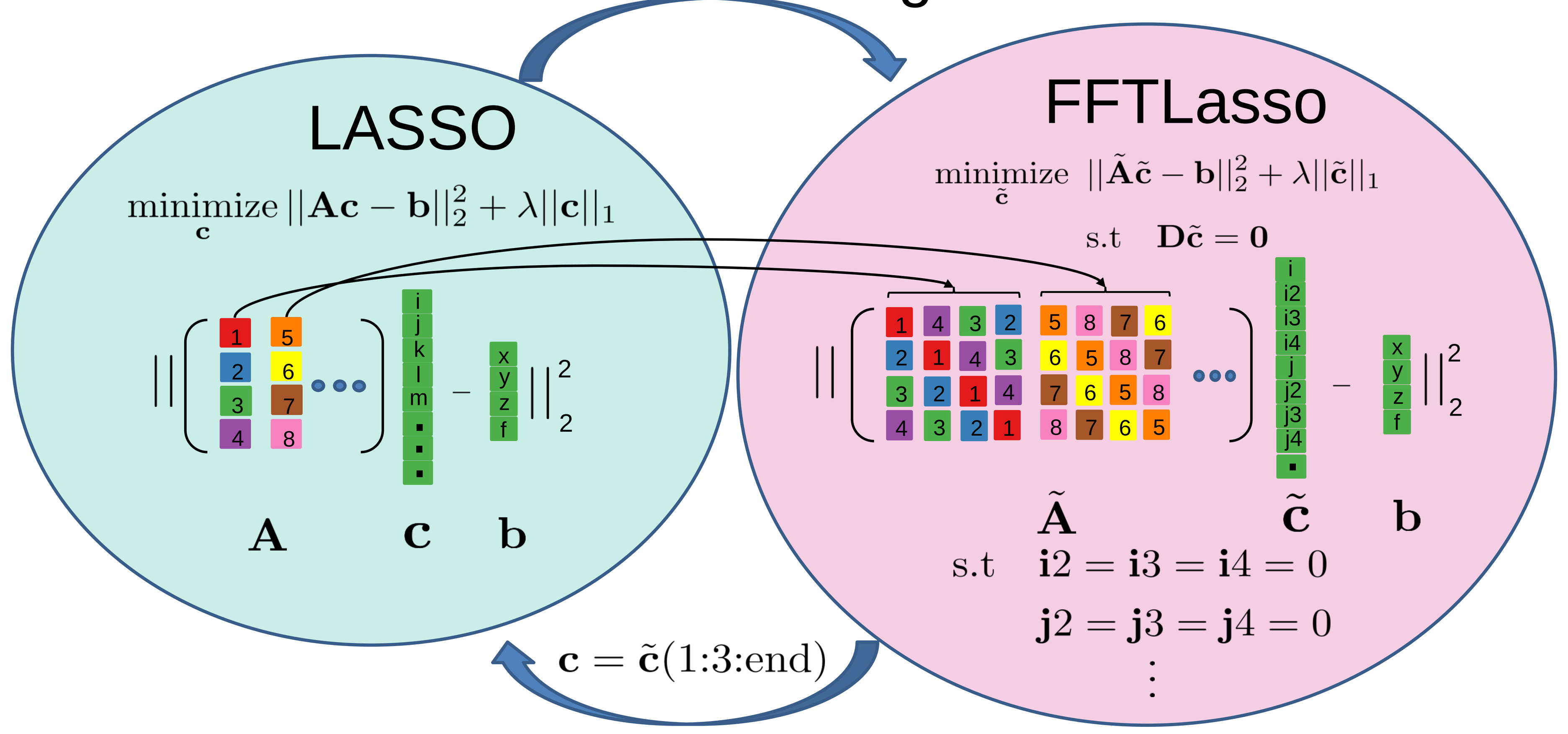
Algorithm 1: DL-ADMM

Input : $\mathbf{b}, \mathbf{A}, \mathbf{c}=\mathbf{0}_n, \zeta=\mathbf{0}_n, \Psi=\mathbf{0}_m, \lambda, \rho_1, \gamma>1$
Output: $\mathbf{c}$
while not converged **do**
- **Ψ update:** solve $(\rho_k\mathbf{A}\mathbf{A}^\top+\frac{1}{2}\mathbf{I}_m)\Psi_{k+1}=\mathbf{A}(\rho_k\zeta-\mathbf{c})-\mathbf{b}.$
- **ζ update:** $\zeta_{k+1}=\text{proj}_{\ell_\infty,\lambda}(\mathbf{A}^\top\Psi_{k+1}+\mathbf{c}_k/\rho_k).$
- **c update:** $\mathbf{c}_{k+1}=\mathbf{c}_k+\rho_k(\mathbf{A}^\top\Psi_{k+1}-\zeta_{k+1})).$
- $\rho_{k+1}=\gamma\rho_k$

end

Problem Reformulation

Problem lifting



Problem down-sampling

Primal Form: $$\min_{\tilde{\mathbf{c}}} \; \|\tilde{\mathbf{A}}\tilde{\mathbf{c}}-\mathbf{b}\|_2^2+\lambda\|\tilde{\mathbf{c}}\|_1 \quad \text{s.t} \quad \mathbf{D}\tilde{\mathbf{c}}=\mathbf{0}$$

Dual Form: $$\min_{\Psi,\theta} \; \frac{1}{4}\|\Psi\|_2^2+\Psi^{\mathbf{H}}\mathbf{b} \quad \text{s.t} \quad \|\tilde{\mathbf{A}}^{\mathbf{H}}\Psi+\mathbf{D}^{\mathbf{H}}\theta\|_\infty \le \lambda$$

- In FFTLasso, the most expensive operation is m-FFTs (n times).
- Nesterov's accelerated gradient is applied to speed up convergence.

Algorithm 1: FFTLasso

Input : $\mathbf{b}, \mathbf{A}, \tilde{\mathbf{c}}_1=\tilde{\mathbf{y}}_1=\tilde{\mathbf{r}}_1=\mathbf{e}_1=\mathbf{t}_1=\zeta_1=\mathbf{D}^{\mathbf{H}}\theta_1=\mathbf{0}_{mn}, \Psi=\mathbf{0}_m, \lambda, \rho_1, \gamma>1, q.$
Output: $\mathbf{c}$
while not converged **do**
- **compute:** $\mathbf{e}_{k+1}=\rho_k\zeta_k-\rho_k\mathbf{D}^{\mathbf{H}}\theta_k-\tilde{\mathbf{c}}_k$
- **$\hat{\Psi}^*$ update:** $\hat{\Psi}^*_{k+1}=\frac{\sum_i^N \hat{\mathbf{a}}_i^*\odot\hat{\mathbf{e}}^*_{ik+1}-\hat{\mathbf{b}}^*}{\rho_k\sum_i^N \hat{\mathbf{a}}_i\odot\hat{\mathbf{a}}_i^*+\frac{1}{2}}$
- **compute:** $\tilde{\mathbf{A}}^{\mathbf{H}}\Psi_{k+1},$
- **$\mathbf{D}^{\mathbf{H}}\theta$ update:** $(\mathbf{D}^{\mathbf{H}}\theta_{k+1})=(\zeta_k-\frac{1}{\rho_k}\tilde{\mathbf{c}}_k-\tilde{\mathbf{A}}^{\mathbf{H}}\Psi_{k+1})$
- $(\mathbf{D}^{\mathbf{H}}\theta_{k+1})_{1:\text{m}:\text{end}}=\mathbf{0}_n$
- **compute:** $\mathbf{t}_{k+1}=\tilde{\mathbf{A}}^{\mathbf{H}}\Psi_{k+1}+\mathbf{D}^{\mathbf{H}}\theta_{k+1}+\tilde{\mathbf{c}}_k/\rho_k$
- **ζ update:** $\zeta_{k+1}=\text{sign}(\mathbf{t}_{k+1})\odot\min(|\mathbf{t}_{k+1}|,\lambda)$
- **$\tilde{\mathbf{c}}$ update:** $\tilde{\mathbf{c}}_{k+1}=\tilde{\mathbf{y}}_k+\rho_k(\tilde{\mathbf{A}}^{\mathbf{H}}\Psi_{k+1}+\mathbf{D}^{\mathbf{H}}\theta_{k+1}-\zeta_{k+1})$
- **compute:** $\tilde{\mathbf{y}}_{k+1}=(1+q)\tilde{\mathbf{c}}_{k+1}-q\tilde{\mathbf{c}}_k$
- $\rho_{k+1}=\gamma\rho_k$

end
$\mathbf{c}\leftarrow\tilde{\mathbf{c}}(1{:}m{:}\text{end})$

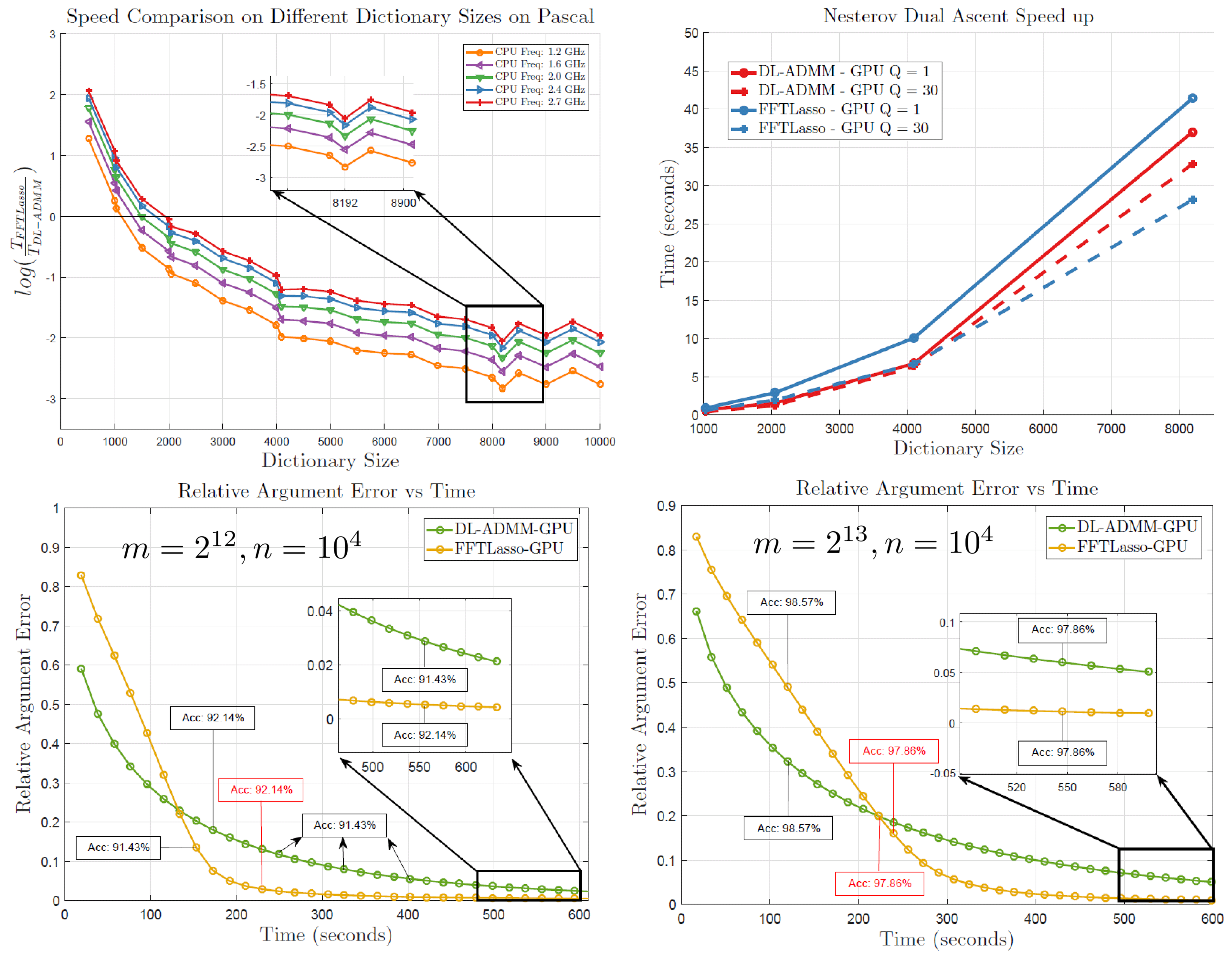
Experiments

- Memory Efficiency. Table [1] demonstrates that FFTLasso is much more memory efficient that DL-ADMM.

Table 1: Memory efficiency comparison between DL-ADMM and FFTLasso on a TitanX GPU with 12GB memory. Dimensions marked with ✓(*) are reached using simple dictionary splitting that is only possible for FFTLasso. Refer to the text for details.

Dictionary Size	2^{12}	6500	2^{13}	8500	9000	9500
DL-ADMM	✓	✓	✗	✗	✗	✗
FFTLasso	✓	✓	✓	✓	✓(*)	✓(*)

- Speed Comparison. We compare FFTLasso against DL-ADMM on both synthetic and Face Recognition datasets.
- Synthetic experiments for DL-ADMM are conducted on both multi core CPUs and on a GPU too.
- Face Recognition experiments are conducted on 2 size-varying problems.



References: [1] A. Yang et al. "A Review for fast l1-Minimization Algorithms for Robust Face Recognition"

Acknowledgements. Thanks to Congli Wang for his help in the cuda implementación. Research reported was supported by competitive funding from King Abdullah University of Science and Technology.