Please refer to the Supplementary Material for the proof.

We conclude this section with the following remarks:

(1) When proving the convergence of IRL1 for solving problem (8) or (12) in (Chen and Zhou ; Lu 2012), they use the Young's inequality which is a special property of the function y^p $(0 < p < 1)$

$$\sum_{i=1}^{n}(|x_i^k|+\epsilon)^p-(|x_i^{k+1}|+\epsilon)^p \geq \sum_{i=1}^{n} w_i^k\left(\left|x_i^k\right|-\left|x_i^{k+1}\right|\right), \tag{28}$$

where $w_i^k = p/(|x_i^k|+\epsilon)^{1-p}$. Eqn (28) is a special case of (23). But (23) is obtained by using the concavity of $f(\mathbf{y})$, which is much more general.

(2) In Eqn (27), $||\mathbf{x}^{k+1}-\mathbf{x}^k||_2$ is used to measure the convergence rate of the algorithm. The reason is that $||\mathbf{x}^{k+1}-\mathbf{x}^k||_2 \to 0$ is a necessary optimality condition as shown in the Theorem 1.

(3) PIRE requires that $\mu > L(h)/2$. But sometimes the Lipschitz constant $L(h)$ is not known, or it is not computable for large scale problems. One may use the backtracking rule to estimate μ in each iteration (Beck and Teboulle 2009). PIRE with multiple splitting shown in the next section also eases this problem.

PIRE with Multiple Splitting

In this section, we will show that PIRE can also solve multi-variable problem as follows

$$\min_{\mathbf{x}_1,\cdots,\mathbf{x}_S} F(\mathbf{x}) = \lambda\sum_{s=1}^{S} f_s(\mathbf{g}_s(\mathbf{x}_s)) + h(\mathbf{x}_1,\cdots,\mathbf{x}_S), \tag{29}$$

where f_s and $\mathbf{g}_s$ holds the same assumptions as f and $\mathbf{g}$ in problem (1), respectively. Problem (29) is similar to problem (1), but splits $\mathbf{x}$ into $\mathbf{x} = [\mathbf{x}_1;\cdots;\mathbf{x}_S] \in \mathbb{R}^n$, where $\mathbf{x}_s \in \mathbb{R}^{n_s}$, and $\sum_{i=1}^{S} n_s = n$.

Based on different assumptions of $h(\mathbf{x}_1,\cdots,\mathbf{x}_S)$, we have two splitting versions of the PIRE algorithm. They use different updating orders of the variables.

PIRE with Parallel Splitting

If we still assume that (**C3**) holds, i.e. $h(\mathbf{x}_1,\cdots,\mathbf{x}_S)$ has a Lipschitz continuous gradient, with Lipschitz constant $L(h)$, PIRE is naturally parallelizable. In each iteration, we parallelly update $\mathbf{x}_s^{k+1}$ by

$$\begin{aligned}\mathbf{x}_s^{k+1} = &\arg\min_{\mathbf{x_s}} \lambda\langle\mathbf{w}_s^k, \mathbf{g}_s(\mathbf{x}_s)\rangle \\ &+\frac{\mu}{2}\left\|\mathbf{x}_s-\left(\mathbf{x}_s^k-\frac{1}{\mu}\nabla_s h\left(\mathbf{x}_1^k,\cdots,\mathbf{x}_S^k\right)\right)\right\|^2,\end{aligned} \tag{30}$$

where the notion $\nabla_s h\left(\mathbf{x}_1,\cdots,\mathbf{x}_S\right)$ denotes the gradient w.r.t $\mathbf{x}_s$, $\mu > L(h)/2$, and $\mathbf{w}_s^k$ is the weight vector corresponding to $\mathbf{g}(\mathbf{x}_s^k)$, which can be computed by

$$\mathbf{w}_s^k \in -\partial\left(-f_s(\mathbf{g}_s(\mathbf{x}_s^k))\right),\ s=1,\cdots,S. \tag{31}$$

When updating $\mathbf{x}_s$ in the $(k+1)$-th iteration, only the variables in the k-th iteration are used. Thus the variables $\mathbf{x}_s^{k+1}$, $s = 1,\cdots,S$, can be updated in parallel. This is known as Jacobi iteration in numerical algebra (Liu, Lin, and Su 2013). This algorithm is named as PIRE with Parallel Splitting (PIRE-PS). Actually the updating rule of PIRE-PS is the same as PIRE, but in parallel. It is easy to check that the proofs in Theorem 1 and 2 also hold for PIRE-PS.

For some special cases of $h(\mathbf{x}_1,\cdots,\mathbf{x}_S)$, we can use different μ_s, usually smaller than μ, for updating $\mathbf{x}_s^{k+1}$. This may lead to faster convergence (Zuo and Lin 2011). If $h(\mathbf{x}_1,\cdots,\mathbf{x}_S) = \frac{1}{2}\left\|\sum_{s=1}^{S}\mathbf{A}_s\mathbf{x}_s-\mathbf{b}\right\|_2^2$, we can update $\mathbf{x}_s^{k+1}$ by

$$\begin{aligned}\mathbf{x}_s^{k+1} = &\arg\min_{\mathbf{x_s}} \lambda\langle\mathbf{w}_s^k, \mathbf{g}_s(\mathbf{x}_s)\rangle \\ &+\frac{\mu_s}{2}\left\|\mathbf{x}_s-\left(\mathbf{x}_s^k-\frac{1}{\mu_s}\mathbf{A}_s^T(\mathbf{A}\mathbf{x}^k-\mathbf{b})\right)\right\|_2^2,\end{aligned} \tag{32}$$

where $\mu_s > L_s(h)/2$ and $L_s(h) = ||\mathbf{A}_s||_2^2$ is the Lipschitz constant of $\nabla_s h(\mathbf{x}_1,\cdots,\mathbf{x}_S)$. If the size of $\mathbf{A}$ is very large, $L(h) = ||\mathbf{A}||_2^2$ may not be computable. We can split it to $\mathbf{A} = [\mathbf{A}_1,\cdots,\mathbf{A}_S]$, and compute each $L_s(h) = ||\mathbf{A}_s||_2^2$ instead. Similar convergence results in Theorem 1 and 2 also hold by updating $\mathbf{x}_s^{k+1}$ in (32). For detailed proofs, please refer to the Supplementary Material. A main difference of the convergence poof is that we use the Pythagoras relation

$$||\mathbf{a}-\mathbf{c}||_2^2-||\mathbf{b}-\mathbf{c}||_2^2 = ||\mathbf{a}-\mathbf{b}||_2^2+2\langle\mathbf{a}-\mathbf{b},\mathbf{b}-\mathbf{c}\rangle, \tag{33}$$

for the squared loss $h(\mathbf{x}_1,\cdots,\mathbf{x}_S)$. This property is much tighter than the property (17) of function with Lipschitz continuous gradient.

The result that using the squared loss leads to smaller Lipschitz constants by PIRE-PS is very interesting and useful. Intuitively, it results to minimize a tighter upper bounded surrogate function. Our experiments show that this will lead to a faster convergence of the PIRE-PS algorithm.

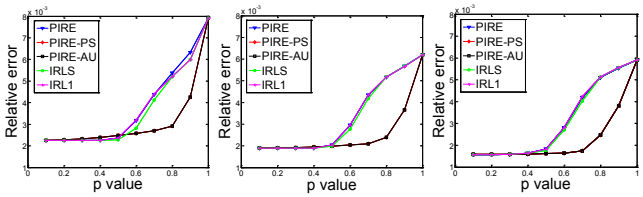
PIRE with Alternative Updating

In this section, we propose another splitting method to solve problem (29) based on the assumption that each $\nabla_s h(\mathbf{x}_1,\cdots,\mathbf{x}_S)$ is Lipschitz continuous with constant $L_s(h)$. Different from PIRE-PS, which updates each $\mathbf{x}_s^{k+1}$ based on $\mathbf{x}_s^k$, $s = 1,\cdots,S$, we instead update $\mathbf{x}_s^{k+1}$ based on all the latest $\mathbf{x}_s$. This is the known Gauss-Sidel iteration in numerical algebra. We name this method as PIRE with Alternative Updating (PIRE-AU).

Since $\nabla_s h(\mathbf{x}_1,\cdots,\mathbf{x}_S)$ is Lipschitz continuous, similar to (17), we have

$$\begin{aligned}&h(\mathbf{x}_1^{k+1},\cdots,\mathbf{x}_{s-1}^{k+1},\mathbf{x}_s,\mathbf{x}_{s+1}^k,\cdots,\mathbf{x}_S^k)\\ \leq &h(\mathbf{x}_1^{k+1},\cdots,\mathbf{x}_{s-1}^{k+1},\mathbf{x}_s^k,\cdots,\mathbf{x}_S^k)+\\ &\langle\nabla_s h(\mathbf{x}_1^{k+1},\cdots,\mathbf{x}_{s-1}^{k+1},\mathbf{x}_s^k,\cdots,\mathbf{x}_S^k),\mathbf{x}_s-\mathbf{x}_s^k\rangle\\ &+\frac{L_s(h)}{2}||\mathbf{x}_s-\mathbf{x}_s^k||_2^2.\end{aligned} \tag{34}$$

The hand right part of (34) is used as a surrogate function of $h(\mathbf{x}_1^{k+1},\cdots,\mathbf{x}_{s-1}^{k+1},\mathbf{x}_s,\mathbf{x}_{s+1}^k,\cdots,\mathbf{x}_S^k)$, which is tighter



(a) (b) (c)

Figure 1: Recovery performance comparison with a different number of measurement $\mathbf{A} \in \mathbb{R}^{m\times 1000}$: (a) $m = 200$; (b) $m = 300$; and (c) $m = 400$.

than (17) in PIRE. Then we update $\mathbf{x}_s^{k+1}$ by

$$\mathbf{x}_s^{k+1} = \arg\min_{\mathbf{x}_s} \lambda\langle \mathbf{w}_s^k, \mathbf{g}_s(\mathbf{x}_s)\rangle + \frac{\mu_s}{2}||\mathbf{x}_s - \mathbf{x}_s^k||_2^2 + \langle \nabla_s h(\mathbf{x}_1^{k+1}, \cdots, \mathbf{x}_{s-1}^{k+1}, \mathbf{x}_s^k, \cdots, \mathbf{x}_S^k), \mathbf{x}_s - \mathbf{x}_s^k\rangle, \quad (35)$$

where $\mu_s > L_s(h)/2$ and $\mathbf{w}_s^k$ is defined in (31).

The updating rule in PIRE-AU by (35) and (31) also leads to converge. Any accumulation point of $\{\mathbf{x}^k\}$ is a stationary point. See the detailed proofs in the Supplementary Material.

Both PIRE-PS and PIRE-AU can solve the multi-variable problems. The advantage of PIRE-PS is that it is naturally parallelizable, while PIRE-AU may converge with less iterations due to smaller Lipschitz constants. If the squared loss function is used, PIRE-PS use the same small Lipschitz constants as PIRE-AU.

Experiments

We present several numerical experiments to demonstrate the effectiveness of the proposed PIRE algorithm and its splitting versions. All the algorithms are implemented by Matlab, and are tested on a PC with 8 GB of RAM and Intel Core 2 Quad CPU Q9550.

ℓ_p-Minimization

We compare our proposed PIRE, PIRE-PS and PIRE-AU algorithms with IRLS and IRL1 for solving the ℓ_p-minimization problem (7). For fair comparison, we try to use the same settings of all the completed algorithms. We use the solution to the ℓ_1-minimization problem as the initialization. We find that this will accelerate the convergence of the iteratively reweighted algorithms, and also enhance the recovery performance. The choice of ϵ in (8) and (10) plays an important role for sparse signal recovery, but theoretical support has not been carried out so far. Several different decreasing rules have been tested before (Candès, Wakin, and Boyd 2008; Mohan and Fazel 2012; Lai, Xu, and Yin 2013), but none of them dominates others. Since the sparsity of sparse signal is usually unknown, we empirically set $\epsilon^{k+1} = \epsilon^k/\rho$, with $\epsilon^0 = 0.01$, and $\rho = 1.1$ (Mohan and Fazel 2012). The algorithms are stopped when $||\mathbf{x}^k - \mathbf{x}^{k+1}||_2/||\mathbf{x}^k||_2 \leq 10^{-6}$.

IRL1 requires solving (9) as inner loop. FISTA is employed to solve (9) with warm start, i.e. using $\mathbf{x}^k$ as initialization to obtain $\mathbf{x}^{k+1}$. This trick greatly reduces the inner loop iteration, which is the main cost for IRL1. For PIRE-PS and PIRE-AU algorithms, we solve problem (29) by setting $S = 20$.

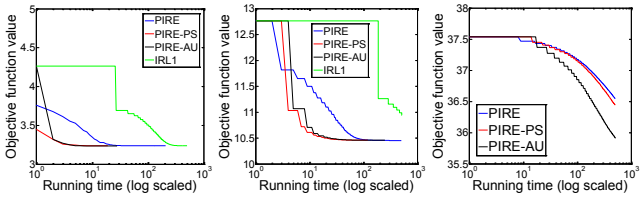
Table 1: Comparison of iteration number, running time (in seconds), objective function value and relative recovery error of different iterative reweighted methods.

Size (m, n, t)	Methods	Iter.	Time (second).	Obj. $(\times 10^{-2})$	Recovery error $(\times 10^{-3})$
(100,500,50)	PIRE	116	0.70	**5.238**	2.529
	PIRE-PS	58	**0.48**	5.239	2.632
	PIRE-AU	**56**	0.63	5.239	2.632
	IRLS	168	81.82	5.506	**2.393**
	IRL1	**56**	3.43	5.239	2.546
(200,800,100)	PIRE	119	1.48	16.923	2.246
	PIRE-PS	37	**0.82**	**16.919**	2.192
	PIRE-AU	**36**	0.88	**16.919**	2.192
	IRLS	169	474.19	17.784	**2.142**
	IRL1	81	13.53	16.924	2.248
(300,1000,200)	PIRE	151	4.63	42.840	2.118
	PIRE-PS	29	1.38	**42.815**	1.978
	PIRE-AU	**28**	**1.34**	**42.815**	**1.977**
	IRLS	171	1298.70	44.937	2.015
	IRL1	79	35.59	42.844	2.124
(500,1500,200)	PIRE	159	8.88	64.769	2.010
	PIRE-PS	26	2.27	**64.718**	**1.814**
	PIRE-AU	**25**	**2.20**	**64.718**	**1.814**
	IRLS	171	3451.79	67.996	1.923
	IRL1	89	80.89	64.772	2.013
(800,2000,200)	PIRE	140	14.99	87.616	1.894
	PIRE-PS	33	5.15	**87.533**	**1.648**
	PIRE-AU	**32**	**4.97**	**87.533**	**1.648**
	IRLS	177	7211.2	91.251	1.851
	IRL1	112	173.26	87.617	1.895

Sparse Signal Recovery The first experiment is to examine the recovery performance of sparse signals by using the proposed methods. The setup for each trial is as follows. The dictionary $\mathbf{A} \in \mathbb{R}^{m\times n}$ is a Gaussian random matrix generated by Matlab command `randn`, with the sizes $m = 200, 300, 400$, and $n = 1000$. The sparse signal $\mathbf{x}$ is randomly generated with sparsity $||\mathbf{x}||_0 = 20$. The response $\mathbf{b} = \mathbf{A}\mathbf{x} + 0.01\mathbf{e}$, where $\mathbf{e}$ is Gaussian random vector. Given $\mathbf{A}$ and $\mathbf{b}$, we can recover $\hat{\mathbf{x}}$ by solving the ℓ_p-minimization problem by different methods. The parameter is set to $\lambda = 10^{-4}$. We use the relative recovery error $||\hat{\mathbf{x}}-\mathbf{x}||_2/||\mathbf{x}||_2$ to measure the recovery performance. Based on the above settings and generated data, we find that the recovery performances are stable. We run 20 trials and report the mean relative error for comparison.

Figure 1 plots the relative recovery errors v.s. different p values ($p = 0.1, \cdots, 0.9, 1$) on three data sets with different numbers of measurements. The result for $p = 1$ is obtained by FISTA for ℓ_1-minimization. We can see that all the iteratively reweighted algorithms achieve better recovery performance with $p < 1$ than ℓ_1-minimization. Also a smaller value of p leads to better recovery performance, though the ℓ_p-minimization problem is nonconvex and a globally optimal solution is not available. In most cases, PIRE is comparative with IRLS and IRL1. A surprising result is that PIRE-PS and PIRE-AU outperform the other methods when $0.5 < p < 1$. They use a smaller Lipschitz constant than PIRE, and thus may converge faster. But none of these iteratively reweighted methods is guaranteed to be optimal.

Running Time Comparison The second experiment is to show the advantage in running time of the proposed methods. We implement all the completed methods in matrix



(a) (b) (c)

Figure 2: Running time v.s. objective function value on three synthesis data sets with size (m, n, t): (a) (1000,3000,500); (b) (1000,5000,1000); (c) (1000,10000,1000).

form for solving the following ℓ_p-minimization problem

$$\min_{\mathbf{X}\in\mathbb{R}^{n\times t}} \lambda||\mathbf{X}||_p^p + \frac{1}{2}||\mathbf{AX}-\mathbf{B}||_F^2, \tag{36}$$

where $\mathbf{A} \in \mathbb{R}^{m\times n}$ and $\mathbf{B} \in \mathbb{R}^{m\times t}$, $||\mathbf{X}||_p^p = \sum_{ij}|X_{ij}|^p$, and p is set to 0.5 in this test. $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{X}$ are generated by the same procedure as the above section, and the same settings of algorithm parameters are followed. Each column of $\mathbf{X}$ is with sparsity $n \times 2\%$. We test on several different sizes of data sets, parameterized as (m, n, t). The iteration number, running time, objective function value and the relative recovery error are tabulated in Table 1. It can be seen that the proposed methods are much more efficient than IRLS and IRL1. PIRE-PS and PIRE-AU converge with less iteration and less running time. In our test, IRL1 is more efficient than IRLS. The reasons lie in: (1) initialization as a sparse solution to ℓ_1-minimization is a good choice for IRL1, but not for IRLS; (2) For each iteration in IRLS, solving t equations (11) in a loop by Matlab is not efficient; (3) IRL1 converges with less inner loop iterations due to warm start.

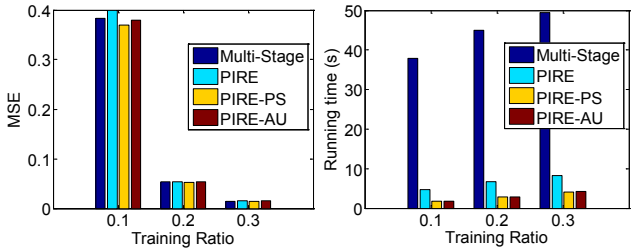
We also plot the running time v.s. objective function value on three larger data sets in Figure 2. The algorithms are stopped within 500 seconds in this test. IRLS costs much more time, and thus it is not plotted. IRL1 is not plotted for the case $n = 10,000$. It can be seen that PIRE-PS and PIRE-AU decreases the objective function value faster than PIRE.

Multi-Task Feature Learning

In this experiment, we use our methods to solve the multi-task learning problem. Assume we are given m learning tasks associated with $\{(\mathbf{X}_1, \mathbf{y}_1), \cdots, (\mathbf{X}_m, \mathbf{y}_m)\}$, where $\mathbf{X}_i \in \mathbb{R}^{n_i\times d}$ is the data matrix of the i-th task with each row a sample, $\mathbf{y}_i \in \mathbb{R}^{n_i}$ is the label of the i-th task, n_i is the number of samples for the i-th task, and d is the data dimension. Our goal is to find a matrix $\mathbf{Z} = [\mathbf{z}_1, \cdots, \mathbf{z}_m] \in \mathbb{R}^{d\times m}$ such that $\mathbf{y}_i \approx \mathbf{X}_i\mathbf{z}_i$. The capped-$\ell_1$ norm is used to regularize $\mathbf{Z}$ (Gong, Ye, and Zhang 2012a)

$$\min_{\mathbf{Z}} \lambda \sum_{j=1}^{d} \min(||\mathbf{z}^j||_1, \theta) + h(\mathbf{Z}), \tag{37}$$

where $h(\mathbf{Z}) = \sum_{i=1}^{m} ||\mathbf{X}_i\mathbf{z}_i - \mathbf{y}_i||_2^2/mn_i$ is the loss function, $\theta > 0$ is the thresholding parameter, and $\mathbf{z}^j$ is the j-th row of $\mathbf{Z}$. The above problem can be solved by our proposed PIRE, PIRE-PS and PIRE-AU algorithms, by letting $f(\mathbf{y}) = \sum_{j=1}^{d} \min(y_i, \theta)$, and $g(\mathbf{Z}) = [||\mathbf{z}^1||_1; \cdots; ||\mathbf{z}^m||_1]$.



(a) (b)

Figure 3: Comparison of (a) mean squared error (MSE) and running time on the Isolet data set for multi-task feature learning.

The Isolet (Bache and Lichman 2013) data set is used in our test. 150 subjects spoke the name of each letter of the alphabet twice. Hence, we have 52 training examples from each speaker. The speakers are grouped into 5 subsets of 30 speakers each. Thus, we have 5 tasks with each task corresponding to a subset. There are 1560, 1560, 1560, 1558, and 1559 samples of 5 tasks, respectively. The data dimension is 617, and the response is the English letter label (1-26). We randomly select the training samples from each task with different training ratios (0.1, 0.2 and 0.3) and use the rest of samples to form the test set. We compare our PIRE, PIRE-PS and PIRE-AU (we set $S = m = 5$ in PIRE-PS and PIRE-AU) with the Multi-Stage algorithm (Zhang 2008). We report the Mean Squared Error (MSE) on the test set and the running time for solving (37) on the training set. The results are averaged over 10 random splittings. As shown in Figure 3, it can be seen that all these methods achieve comparative performance, but our PIRE, PIRE-PS and PIRE-AU are much more efficient than the Multi-Stage algorithm.

Conclusions

This paper proposes the PIRE algorithm for solving the general problem (1). PIRE solves a series of problem (2), whose computational cost is usually very cheap. We further propose two splitting versions of PIRE to handle the multi-variable problems. In theory, we prove that PIRE (also its splitting versions) converges and any limit point is a stationary point. We test our methods to solve the ℓ_p-minimization problem and multi-task feature learning problem. Experimental results on both synthesis and real data sets show that our methods are with comparative learning performance, but much more efficient, by comparing with IRLS and IRL1 or multi-stage algorithms. It would be interesting to apply PIRE for structured sparsity optimization, and also the non-convex low rank regularized minimization problems (Lu et al. 2014).

Acknowledgements

This research is supported by the Singapore National Research Foundation under its International Research Centre @Singapore Funding Initiative and administered by the IDM Programme Office. Z. Lin is supported by NSF of China (Grant nos. 61272341, 61231002, and 61121002) and MSRA.

References

Bache, K., and Lichman, M. 2013. UCI machine learning repository. http://archive.ics.uci.edu/ml.

Beck, A., and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences* 2(1):183–202.

Bertsekas, D. P. 1999. *Nonlinear programming*. Athena Scientific (Belmont, Mass.), 2nd edition.

Candès, E.; Wakin, M.; and Boyd, S. 2008. Enhancing sparsity by reweighted ℓ_1 minimization. *Journal of Fourier Analysis and Applications* 14(5):877–905.

Chen, X., and Zhou, W. Convergence of the reweighted ℓ_1 minimization algorithm for $\ell_2 - \ell_p$ minimization. *to appear in Comp. Optim. Appl*.

Fan, J., and Li, R. 2001. Variable selection via nonconcave penalized likelihood and its oracle properties. *Journal of the American Statistical Association* 96(456):1348–1360.

Gong, P.; Zhang, C.; Lu, Z.; Huang, J.; and Ye, J. 2013. A general iterative shrinkage and thresholding algorithm for non-convex regularized optimization problems. *ICML*.

Gong, P.; Ye, J.; and Zhang, C. 2012a. Multi-stage multi-task feature learning. In *NIPS*.

Gong, P.; Ye, J.; and Zhang, C. 2012b. Robust multi-task feature learning. In *ACM SIGKDD*, 895–903. ACM.

Jacob, L.; Obozinski, G.; and Vert, J.-P. 2009. Group Lasso with overlap and graph Lasso. In *ICML*, 433–440. ACM.

Knight, K., and Fu, W. 2000. Asymptotics for Lasso-type estimators. *Annals of Statistics* 1356–1378.

Lai, M.-J.; Xu, Y.; and Yin, W. 2013. Improved iteratively reweighted least squares for unconstrained smoothed ℓ_q minimization. *SIAM Journal on Numerical Analysis* 51(2):927–957.

Liu, R.; Lin, Z.; and Su, Z. 2013. Linearized alternating direction method with parallel splitting and adaptive penalty for separable convex programs in machine learning. In *ACML*.

Lu, C.; Tang, J.; Lin, Z.; and Yan, S. 2014. Generalized nonconvex nonsmooth low-rank minimization. In *CVPR*.

Lu, Z. 2012. Iterative reweighted minimization methods for ℓ_p regularized unconstrained nonlinear programming. *Mathematical Programming*.

Mohan, K., and Fazel, M. 2012. Iterative reweighted algorithms for matrix rank minimization. In *JMLR*, volume 13, 3441–3473.

Wright, J.; Yang, A. Y.; Ganesh, A.; Sastry, S. S.; and Ma, Y. 2009. Robust face recognition via sparse representation. *TPAMI* 31(2):210–227.

Zhang, T. 2008. Multi-stage convex relaxation for learning with sparse regularization. In *NIPS*, 1929–1936.

Zhang, C. 2010. Nearly unbiased variable selection under minimax concave penalty. *The Annals of Statistics*.

Zhou, Z.; Li, X.; Wright, J.; Candès, E.; and Ma, Y. 2010. Stable principal component pursuit. In *IEEE International Symposium on Information Theory Proceedings*, 1518–1522. IEEE.

Zuo, W., and Lin, Z. 2011. A generalized accelerated proximal gradient approach for total-variation-based image restoration. *TIP* 20(10):2748–2759.