Fast, Provably convergent IRLS Algorithm for p-norm Linear Regression

Deeksha Adil
Department of Computer Science
University of Toronto
deeksha@cs.toronto.edu

Richard Peng
School of Computer Science
Georgia Institute of Technology
rpeng@cc.gatech.edu

Sushant Sachdeva
Department of Computer Science
University of Toronto
sachdeva@cs.toronto.edu

Abstract

Linear regression in ℓ_p-norm is a canonical optimization problem that arises in several applications, including sparse recovery, semi-supervised learning, and signal processing. Generic convex optimization algorithms for solving ℓ_p-regression are slow in practice. Iteratively Reweighted Least Squares (IRLS) is an easy to implement family of algorithms for solving these problems that has been studied for over 50 years. However, these algorithms often diverge for $p > 3$, and since the work of Osborne (1985), it has been an open problem whether there is an IRLS algorithm that is guaranteed to converge rapidly for $p > 3$. We propose p-IRLS, the first IRLS algorithm that provably converges geometrically for any $p \in [2, \infty)$. Our algorithm is simple to implement and is guaranteed to find a high accuracy solution in a sub-linear number of iterations. Our experiments demonstrate that it performs even better than our theoretical bounds, beats the standard Matlab/CVX implementation for solving these problems by 10–50x, and is the fastest among available implementations in the high-accuracy regime.

1   Introduction

We consider the problem of ℓ_p-norm linear regression (henceforth referred to as ℓ_p-regression),

$$\arg \min_{\boldsymbol{x} \in \mathbb{R}^n} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p, \tag{1}$$

where $\boldsymbol{A} \in \mathbb{R}^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m$ are given and $\|\boldsymbol{v}\|_p = \left(\sum_i |\boldsymbol{v}_i|^p\right)^{1/p}$ denotes the ℓ_p-norm. This problem generalizes linear regression and appears in several applications including sparse recovery [CT05], low rank matrix approximation [CGK+17], and graph based semi-supervised learning [AL11].

An important application of ℓ_p-regression with $p \geq 2$ is graph based semi-supervised learning (SSL). Regularization using the standard graph Laplacian (also called a 2-Laplacian) was introduced in the seminal paper of Zhu, Gharamani, and Lafferty [ZGL03], and is a popular approach for graph based SSL, see e.g. [ZBL+04, BMN04, CSZ09, Zhu05]. The 2-Laplacian regularization suffers from degeneracy in the limit of small amounts of labeled data [NSZ09]. Several works have since suggested using the p-Laplacian instead [AL11, BZ13, ZB11] with large p, and have established its consistency and effectiveness for graph based SSL with small amounts of data [ACR+16, Cal17, RCL19, ST17, KRSS15]. Recently, p-Laplacians have also been used for data clustering and learning problems

[ETT15, EDT17, HFE18]. Minimizing the p-Laplacian can be easily seen as an ℓ_p-regression problem.

Though ℓ_p-regression is a convex programming problem, it is very challenging to solve in practice. General convex programming methods such as conic programming using interior-point methods (like those implemented in CVX) are very slow in practice. First order methods do not perform well for these problems with $p > 2$ since the gradient vanishes rapidly close to the optimum.

For applications such graph based SSL with p-Laplacians, it is important that we are able to compute a solution $\boldsymbol{x}$ that approximates the optimal solution $\boldsymbol{x}^\star$ coordinate-wise rather than just achieving an approximately optimal objective value, since these coordinates determine the labels for the vertices. For such applications, we seek a $(1 + \varepsilon)$-approximate solution, an $\boldsymbol{x}$ such that its objective value, $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p$ is at most $(1 + \varepsilon)$ times the optimal value $\|\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{b}\|_p^p$, for some very small ε (10^{-8} or so) in order achieve a reasonable coordinate-wise approximation. Hence, it is very important for the dependence on ε be $\log 1/\varepsilon$ rather than $\mathrm{poly}(1/\varepsilon)$.

IRLS Algorithms. A family of algorithms for solving the ℓ_p-regression problem are the IRLS (Iterated Reweighted Least Squares) algorithms. IRLS algorithms have been discovered multiple times independently and have been studied extensively for over 50 years e.g. [Law61, Ric64, Osb85, GR97] (see [Bur12] for a detailed survey). The main step in an IRLS algorithm is to solve a weighted least squares (ℓ_2-regression) problem to compute the next iterate,

$$\boldsymbol{x}^{(t+1)} = \arg\min_{\boldsymbol{x}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^\top \boldsymbol{R}^{(t)}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}), \tag{2}$$

starting from any initial solution $\boldsymbol{x}^{(0)}$ (usually the least squres solution corresponding to $\boldsymbol{R} = \boldsymbol{I}$). Each iteration can be implemented by solving a linear system $\boldsymbol{x}^{(t+1)} \leftarrow (\boldsymbol{A}^\top \boldsymbol{R}^{(t)}\boldsymbol{A})^{-1}\boldsymbol{A}^\top \boldsymbol{R}^{(t)}\boldsymbol{b}$. Picking $\boldsymbol{R}^{(t)} = \mathrm{diag}\left(|\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{b}|^{p-2}\right)$, gives us an IRLS algorithm where the only fixed point is the minimizer of the regression problem (1) (which is unique for $p \in (1, \infty)$).

The basic version of the above IRLS algorithm converges reliably in practice for $p \in (1.5, 3)$, and diverges often even for moderate p (say $p \geq 3.5$ [RCL19, pg 12]). Osborne [Osb85] proved that the above IRLS algorithm converges in the limit for $p \in [1, 3)$. Karlovitz [Kar70] proved a similar result for an IRLS algorithm with a line search for even $p > 2$. However, both these results only prove convergence in the limit without any quantitative bounds, and assume that you start close enough to the solution. The question of whether a suitable IRLS algorithm converges geometrically to the optimal solution for (1) in a few iterations has been open for over three decades.

Our Contributions. We present p-IRLS, the first IRLS algorithm that provably converges geometrically to the optimal solution for ℓ_p-regression for all $p \in [2, \infty)$. Our algorithm is very similar to the standard IRLS algorithm for ℓ_p regression, and given an $\varepsilon > 0$, returns a feasible solution $\boldsymbol{x}$ for (1) in $O_p(m^{\frac{p-2}{2(p-1)}} \log \frac{m}{\varepsilon}) \leq O_p(\sqrt{m} \log \frac{m}{\varepsilon})$ iterations (Theorem 3.1). Here m is the number of rows in $\boldsymbol{A}$. We emphasize that the dependence on ε is $\log \frac{1}{\varepsilon}$ rather than $\mathrm{poly}(\frac{1}{\varepsilon})$.

Our algorithm p-IRLS is very simple to implement, and our experiments demonstrate that it is much faster than the available implementations for $p \in (2, \infty)$ in the high accuracy regime. We study its performance on random dense instances of ℓ_p-regression, and low dimensional nearest neighbor graphs for p-Laplacian SSL. Our Matlab implementation on a standard desktop machine runs in at most 2–2.5s (60–80 iterations) on matrices for size 1000×850, or graphs with 1000 nodes and around 5000 edges, even with $p = 50$ and $\varepsilon = 10^{-8}$. Our algorithm is at least 10–50x faster than the standard Matlab/CVX solver based on Interior point methods [GB14, GB08], while finding a better solution. We also converge much faster than IRLS-homotopy based algorithms [RCL19] that are not even guaranteed to converge to a good solution. For larger p, say $p > 20$, this difference is even more dramatic, with p-IRLS obtaining solutions with at least 4 orders of magnitude smaller error with the same number of iterations. Our experiments also indicate that p-IRLS scales much better than as indicated by our theoretical bounds, with the iteration count almost unchanged with problem size, and growing very slowly (at most linearly) with p.

1.1 Related Works and Comparison

IRLS algorithms have been used widely for various problems due to their exceptional simplicity and ease of implementation, including compressive sensing [CW08], sparse signal reconstruction [GR97], and Chebyshev approximation in FIR filter design [BB94]. There have been various attempts at analyzing variants of IRLS algorithm for ℓ_p-norm minimization. We point the reader to the survey by Burrus [Bur12] for numerous pointers and a thorough history.

The works of Osborne [Osb85] and Karlovitz [Kar70] mentioned above only prove convergence in the limit without quantitative bounds and under assumptions on p and that we start close enough. Several works show that it is similar to Newton's method (e.g. [Kah72, BBS94]), or that adaptive step sizes help (e.g. [VB99, VB12]) but do not prove any guarantees.

A few notable works prove convergence guarantees for IRLS algorithms for sparse recovery (even $p < 1$ in some cases) [DDFG08, DDFG10, BL18], and for low-rank matrix recovery [FRW11]. Quantitative convergence bounds for IRLS algorithms for ℓ_1 are given by Straszak and Vishnoi [SV16b, SV16c, SV16a], inspired by slime-mold dynamics. Ene and Vladu give IRLS algorithms for ℓ_1 and ℓ_∞ [EV19]. However, both these works have poly($1/\varepsilon$) dependence in the number of iterations, with the best result by [EV19] having a total iteration count roughly $m^{1/3}\varepsilon^{-1}$.

The most relevant theoretical results for ℓ_p-norm minimization are Interior point methods [NN94], the homotopy method of Bubeck *et al* [BCLL18], and the iterative-refinement method of Adil *et al.* [AKPS19]. The convergence bounds we prove on the number of iterations required by p-IRLS (roughly $m^{\frac{p-2}{2p-2}}$) has a better dependence on m than Interior Point methods (roughly $m^{1/2}$), but marginally worse than the dependence in the work of Bubeck *et al.* [BCLL18] (roughly $m^{\frac{p-2}{2p}}$) and Adil *et al.* [AKPS19] (roughly $m^{\frac{p-2}{2p+(p-2)}}$). Also related, but not directly comparable is the work of Bullins [Bul18] (restricted to $p = 4$) and the work of Maddison *et al.* [MPT$^+$18] (first order method with condition number dependence).

More importantly though, our algorithm is far simpler to implement (no implementations are available for any of the above other than Interior Point Methods), and unlike any of these works, our algorithm has a *locally greedy* structure that allows for greedily optimizing the objective using a line search, resulting in much better performance in practice than that guaranteed by our theoretical bounds.

Another line of heuristic algorithms combines IRLS algorithms with a homotopy based approach (e.g. [Kah72]. See [Bur12]). These methods start from a solution for $p = 2$, and slowly increase p multiplicatively, using an IRLS algorithm for each phase and the previous solution as a starting point. These algorithms perform better in practice than usual IRLS algorithms. However, to the best of our knowledge, they are not guaranteed to converge, and no bounds on their performance are known. Rios [Rio19] provides an efficient implementation of such a method based on the work of Rios *et al.* [RCL19], along with detailed experiments. Our experiments show that our algorithm converges much faster than the implementation from Rios (see Section 4).

2 Preliminaries

We first define some terms that we will use in the formal analysis of our algorithm. For our analysis we use a more general form of the ℓ_p-regression problem,

$$\arg\min_{\boldsymbol{x}:\boldsymbol{Cx}=\boldsymbol{d}} \|\boldsymbol{Ax} - \boldsymbol{b}\|_p. \tag{3}$$

Setting $\boldsymbol{C}$ and $\boldsymbol{d}$ to be empty recovers the standard ℓ_p-regression problem.

Definition 2.1 (Residual Problem). *The residual problem of* (3) *at $\boldsymbol{x}$ is defined as,*

$$\max_{\Delta:\boldsymbol{C\Delta}=0} \quad \boldsymbol{g}^\top \boldsymbol{A\Delta} - 2p^2\Delta^\top \boldsymbol{A}^\top \boldsymbol{RA\Delta} - p^p\|\boldsymbol{A\Delta}\|_p^p.$$

Here $\boldsymbol{R} = diag\left(|\boldsymbol{Ax} - \boldsymbol{b}|^{p-2}\right)$ and $\boldsymbol{g} = p\boldsymbol{R}(\boldsymbol{Ax} - \boldsymbol{b})$ is the gradient of the objective at $\boldsymbol{x}$. Define $\gamma(\Delta)$ to denote the objective of the residual problem evaluated at Δ.

Definition 2.2 (Approximation to the Residual Problem). *Let $\kappa \geq 1$ and $\Delta^\star$ be the optimum of the residual problem. A κ-approximate solution to the residual problem is $\widetilde{\Delta}$ such that $\boldsymbol{C}\widetilde{\Delta} = 0$, and $\gamma(\widetilde{\Delta}) \geq \frac{1}{\kappa}\gamma(\Delta^\star)$.*

3 Algorithm and Analysis

Algorithm 1 p-IRLS Algorithm

1: **procedure** p-IRLS$(\boldsymbol{A}, \boldsymbol{b}, \varepsilon, \boldsymbol{C}, \boldsymbol{d})$
2:      $\boldsymbol{x} \leftarrow \arg\min_{\boldsymbol{Cx}=\boldsymbol{d}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$.
3:      $i \leftarrow \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p / 16p$
4:      **while** $\frac{\varepsilon}{16p(1+\varepsilon)} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p < i$ **do**
5:          $\boldsymbol{R} \leftarrow |\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}|^{p-2}$
6:          $\boldsymbol{g} = p\boldsymbol{R}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$
7:          $s \leftarrow \frac{1}{2}i^{(p-2)/p}m^{-(p-2)/p}$
8:          $\widetilde{\Delta} \leftarrow \arg\min_{\boldsymbol{g}^\top \boldsymbol{A}\Delta=i/2, \boldsymbol{C}\Delta=0} \quad \Delta^\top \boldsymbol{A}^\top (\boldsymbol{R} + s\boldsymbol{I})\boldsymbol{A}\Delta$
9:          $\alpha \leftarrow \text{LINESEARCH}(\boldsymbol{A}, \boldsymbol{b}, \boldsymbol{x}^{(t)}, \widetilde{\Delta})$ $\triangleright \alpha = \arg\min_\alpha \|\boldsymbol{A}(\boldsymbol{x} - \alpha\widetilde{\Delta}) - \boldsymbol{b}\|_p^p$
10:        $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} - \alpha\widetilde{\Delta}$
11:        **if** INSUFFICIENTPROGRESSCHECK$(\boldsymbol{A}, \boldsymbol{R} + s\boldsymbol{I}, \widetilde{\Delta}, i)$ **then** $i \leftarrow i/2$
12:      **return** $\boldsymbol{x}$

Algorithm 2 Check Progress

1: **procedure** INSUFFICIENTPROGRESSCHECK$(\boldsymbol{A}, \boldsymbol{R}, \Delta, i)$
2:      $\lambda \leftarrow 16p$
3:      $k \leftarrow \frac{p^p \|\boldsymbol{A}\Delta\|_p^p}{2p^2 \Delta^\top \boldsymbol{A}^\top \boldsymbol{R}\boldsymbol{A}\Delta}$
4:      $\alpha_0 \leftarrow \min\left\{\frac{1}{16\lambda}, \frac{1}{(16\lambda k)^{1/(p-1)}}\right\}$
5:      **if** $\gamma(\alpha_0 \cdot \widetilde{\Delta}) < \frac{\alpha_0}{4}i$ or $\Delta^\top \boldsymbol{A}^\top (\boldsymbol{R} + s\boldsymbol{I})\boldsymbol{A}\Delta > \lambda i/p^2$ **then return** true
6:      **else return** false

Our algorithm p-IRLS, described in Algorithm (1), is the standard IRLS algorithm (equation 2) with few key modifications. The first difference is that at each iteration t, we add a small systematic padding $s^{(t)}\boldsymbol{I}$ to the weights $\boldsymbol{R}^{(t)}$. The second difference is that the next iterate $\boldsymbol{x}^{(t+1)}$ is calculated by performing a line search along the line joining the current iterate $\boldsymbol{x}^{(t)}$ and the standard IRLS iterate $\widetilde{\boldsymbol{x}}^{(t+1)}$ at iteration $t+1$ (with the modified weights) [1]. Both these modifications have been tried in practice, but primarily from practical justifications: padding the weights avoids ill-conditioned matrices, and line-search can only help us converge faster and improves stability [Kar70, VB99, VB99]. Our key contribution is to show that these modifications together allow us to provably make $\Omega_p(m^{-\frac{p-2}{2(p-1)}})$ progress towards the optimum, resulting in a final iteration count of $O_p(m^{\frac{p-2}{2(p-1)}} \log \frac{m}{\varepsilon})$. Finally, at every iteration we check if the objective value decreases sufficiently, and this allows us to adjust $s^{(t)}$ appropriately. We emphasize here that our algorithm always converges. We prove the following theorem:

Theorem 3.1. *Given any $\boldsymbol{A} \in \mathbb{R}^{m \times n}, \boldsymbol{b} \in \mathbb{R}^m, \varepsilon > 0, p \geq 2$ and $\boldsymbol{x}^\star = \arg\min_{\boldsymbol{x}: \boldsymbol{Cx}=\boldsymbol{d}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p$. Algorithm 1 returns $\boldsymbol{x}$ such that $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p \leq (1 + \varepsilon) \|\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{b}\|_p^p$ and $\boldsymbol{Cx} = \boldsymbol{d}$, in at most $O\left(p^{3.5}m^{\frac{p-2}{2(p-1)}} \log\left(\frac{m}{\varepsilon}\right)\right)$ iterations.*

3.1 Convergence Analysis

The analysis, at a high level, is based on iterative refinement techniques for ℓ_p-norms developed in the work of Adil *et al* [AKPS19] and Kyng *et al* [KPSW19]. These techniques allow us to use a crude κ-approximate solver for the *residual* problem (Definition 2.1) $O_p(\kappa \log \frac{m}{\varepsilon})$ number of times to obtain a $(1 + \varepsilon)$ approximate solution for the ℓ_p-regression problem (Lemma 3.2).

---

[1]Note that p-IRLS has been written in a slightly different but equivalent formulation, where it solves for $\widetilde{\Delta} = \boldsymbol{x}^{(t)} - \widetilde{\boldsymbol{x}}^{(t+1)}$.

In our algorithm, if we had solved the standard weighted ℓ_2 problem instead, κ would be unbounded. The padding added to the weights allow us to prove that the solution to weighted ℓ_2 problem gives a bounded approximation to the residual problem provided we have the correct padding, or in other words correct value of i (Lemma 3.3). Lemma 3.4 gives the required condition to adjust the value of i and is used in Algorithm 2. We will show that the number of iterations where we are adjusting the value of i are small. Finally, we require proving our termination condition. Lemma 3.6 shows that when the algorithm terminates, we have an ε-approximate solution to our main problem. The remaining lemma of this section, Lemma 3.5 gives the loop invariant which is used at several places in the proof of Theorem 3.1. Due to space constraints, we only state the main lemmas here and defer the proofs to the supplementary material.

We begin with the lemma that talks about our overall iterative refinement scheme. The iterative refinement scheme in [AKPS19] and [KPSW19] has an exponential dependence on p. We improve this dependence to a small polynomial in p.

Lemma 3.2. *(Iterative Refinement). Let $p \geq 2$, and $\kappa \geq 1$. Starting from $\boldsymbol{x}^{(0)} = \mathrm{argmin}_{\boldsymbol{Cx=d}} \|\boldsymbol{Ax} - \boldsymbol{b}\|_2^2$, and iterating as, $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \Delta$, where Δ is a κ-approximate solution to the residual problem (Definition 2.1), we get an ε-approximate solution to (3) in at most $O\left(p^2 \kappa \log\left(\frac{m}{\varepsilon}\right)\right)$ calls to a κ-approximate solver for the residual problem.*

The next lemma talks about bounding the approximation factor κ, when we have the right value of i.

Lemma 3.3. *(Approximation). Let i be such that $(\|\boldsymbol{Ax}^{(t)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{Ax}^{\star} - \boldsymbol{b}\|_p^p)/16p \in (i/2, i]$ and $\boldsymbol{R}, \boldsymbol{g}, s, \alpha$ be as defined in lines (5), (6), (7) and (9) of Algorithm 1. Let $\widetilde{\Delta}$ be the solution of the following program,*

$$\arg\min_{\Delta} \Delta^{\top} \boldsymbol{A}^{\top} (\boldsymbol{R} + s\boldsymbol{I}) \boldsymbol{A} \Delta \quad s.t. \quad \boldsymbol{g}^{\top} \boldsymbol{A} \Delta = i/2, \; \boldsymbol{C}\Delta = 0. \tag{4}$$

Then, $\alpha \cdot \widetilde{\Delta}$ is an $O\left(p^{1.5} m^{\frac{p-2}{2(p-1)}}\right)$- approximate solution to the residual problem.

The following lemma gives us the condition that determines if we do not have the correct value of i. The condition used in the algorithm is the contrapositive of the lemma statement.

Lemma 3.4. *(Check Progress). Let α_0 be as defined in line (4) of Algorithm 2 and $\widetilde{\Delta}$ the solution of program (4). If $i/2 < \frac{(\|\boldsymbol{Ax}^{(t)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{Ax}^{\star} - \boldsymbol{b}\|_p^p)}{16p} \leq i$, then $\gamma(\alpha_0 \cdot \widetilde{\Delta}) \geq \frac{\alpha_0 i}{4}$.*

We next present the loop invariant followed by the conditions for the termination.

Lemma 3.5. *(Invariant) At every iteration of the while loop, we have $\boldsymbol{Cx}^{(t)} = \boldsymbol{d}$, $\frac{(\|\boldsymbol{Ax}^{(t)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{Ax}^{\star} - \boldsymbol{b}\|_p^p)}{16p} < i$ and $i \geq \frac{\varepsilon}{16p(1+\varepsilon)} \|\boldsymbol{Ax}^{(0)} - \boldsymbol{b}\|_p^p m^{-(p-2)/2}$,*

Lemma 3.6. *(Termination). Let i be such that $(\|\boldsymbol{Ax}^{(t)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{Ax}^{\star} - \boldsymbol{b}\|_p^p)/16p \in (i/2, i]$. Then,*

$$i \leq \frac{\varepsilon}{16p(1+\varepsilon)} \|\boldsymbol{Ax}^{(t)} - \boldsymbol{b}\|_p^p \Rightarrow \|\boldsymbol{Ax}^{(t)} - \boldsymbol{b}\|_p^p \leq (1+\varepsilon)\text{OPT}.$$

and,

$$\|\boldsymbol{Ax}^{(t)} - \boldsymbol{b}\|_p^p \leq (1+\varepsilon)\text{OPT} \Rightarrow i \leq 2\frac{\varepsilon}{16p(1+\varepsilon)} \|\boldsymbol{Ax}^{(t)} - \boldsymbol{b}\|_p^p.$$

We next see how Lemmas 3.2, 3.3, 3.4, and 3.6 together imply our main result, Theorem 3.1.

3.2 Proof of Theorem 3.1

Proof. We first show that at termination, the algorithm returns an ε-approximate solution. We begin by noting that the quantity i can only decrease with every iteration. At iteration t, let i_0 denote the smallest number such that $(\|\boldsymbol{Ax}^{(t)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{Ax}^{\star} - \boldsymbol{b}\|_p^p)/16p \in (i_0/2, i_0]$. Note that i must be at least i_0 (Lemma 3.5). Let us first consider the termination condition of the while loop. When we terminate, $\frac{\varepsilon\|\boldsymbol{Ax}^{(t)} - \boldsymbol{b}\|_p^p}{16p(1+\varepsilon)} \geq i \geq i_0$. Lemma 3.6 now implies that $\left\|\boldsymbol{Ax}^{(t)} - \boldsymbol{b}\right\|_p^p \leq (1+\varepsilon)OPT$.

Lemma 3.5 also shows that at each iteration our solution satisfies $\boldsymbol{Cx}^{(t)} = \boldsymbol{d}$, therefore the solution returned at termination also satisfies the subspace constraints.

5

We next prove the running time bound. Note that the objective is non increasing with every iteration. This is because the LINESEARCH returns a factor that minimizes the objective given a direction $\widetilde{\Delta}$, i.e., $\alpha = \arg\min_\delta \|\boldsymbol{A}(\boldsymbol{x} - \delta\widetilde{\Delta}) - \boldsymbol{b}\|_p^p$, which could also be zero. Consider the following two cases on the values of i. These are the only cases possible due to our invariant Lemma 3.5.

1. $(\|\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}^\star - b\|_p^p)/16p \in (i/2, i]$:
   In this case, Lemma 3.3 says that we get a $O_p(m^{(p-2)/2(p-1)})$-approximate solution to the residual problem.
2. $(\|\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}^\star - b\|_p^p)/16p \leq i/2$:
   In this case, we want the algorithm to reduce i so that it falls under the above case. Lemma 3.4 implies that if $\gamma(\alpha_0\widetilde{\Delta}) < \alpha_0 i/4$, then we do not have $(\|\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}^\star - b\|_p^p)/16p \in (i/2, i]$ and therefore the algorithm reduces i. In the case when $\gamma(\alpha_0\widetilde{\Delta}) \geq \alpha_0 i/4$, the algorithm ensures that $\Delta^\top \boldsymbol{A}^\top (\boldsymbol{R} + s\boldsymbol{I})\boldsymbol{A}\Delta \leq \lambda i/p^2$. As a result we can apply Lemma A.5 (from the Appendix, given in the supplementary material) to bound α_0, which following the proof of Lemma 3.3 again gives a $O(p^{1.5}m^{\frac{p-2}{2(p-1)}})$-approximate solution to the residual problem.

From the above, it is clear that the algorithm either reduces i or returns an $O\left(p^{1.5}m^{\frac{p-2}{2(p-1)}}\right)$-approximate solution to the residual problem. The number of steps in which we reduce i is at most $\log(i_{initial}/i_{min}) = p\log\left(\frac{m}{\varepsilon}\right)$ (Lemma 3.5 gives the value of i_{min}). The number of remaining steps from Lemma 3.2 is $O\left(p^{3.5}m^{\frac{p-2}{2(p-1)}}\log\left(\frac{m}{\varepsilon}\right)\right)$. The total number of iterations required by our algorithm is $O\left(p^{3.5}m^{\frac{p-2}{2(p-1)}}\log\left(\frac{m}{\varepsilon}\right)\right)$, thus concluding the proof of the theorem. $\qquad\square$

4 Experiments

In this section, we detail our results from experiments studying the performance of our algorithm, p-IRLS. We implemented our algorithm in Matlab on a standard desktop machine, and evaluated its performance on two types of instances, random instances for ℓ_p-regression, and graphs for p-Laplacian minimization. We study the scaling behavior of our algorithm as we change p, ε, and the size of the problem. We compare our performance to the Matlab/CVX solver that is guaranteed to find a good solution, and to the IRLS/homotopy based implementation from [RCL19] that is not guaranteed to converge, but runs quite well in practice. We now describe our instances, parameters and experiments in detail.
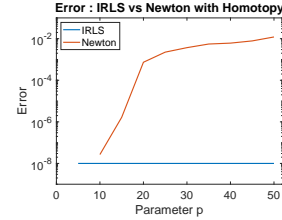

Figure 1: Averaged over 100 random samples. Graph: 1000 nodes (5000-6000 edges). Solver: PCG with Cholesky preconditioner.

Instances and Parameters. We consider two types of instances, random matrices and graphs.

1. **Random Matrices:** We want to solve the problem $\min_x \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p$. In these instances we use random matrices $\boldsymbol{A}$ and $\boldsymbol{b}$, where every entry of the matrix is chosen uniformly at random between 0 and 1.
2. **Graphs:** We use the graphs described in [RCL19]. The set of vertices is generated by choosing vectors in $[0, 1]^{10}$ uniformly at random and the edges are created by connecting the 10 nearest neighbours. Weights of each edge is specified by a gaussian type function (Eq 3.1,[RCL19]). Very few vertices (around 10) have labels which are again chosen uniformly at random between 0 and 1. The problem studied on these instances is to determine the minimizer of the ℓ_p laplacian. We formulate this problem into the form $\min_x \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p$, details of this formulation can be found in the Appendix that is in the supplementary material.

Note that we have 3 different parameters for each problem, the size of the instance i.e., the number of rows of matrix $\boldsymbol{A}$, the norm we solve for, p, and the accuracy to which we want to solve each

(a) Size of A fixed to 1000×850.

(b) Sizes of A: $(50 + 100(k - 1)) \times 100k$. Error $\varepsilon = 10^{-8}$.

(c) Size of A is fixed to 1000×850. Error $\varepsilon = 10^{-8}$.
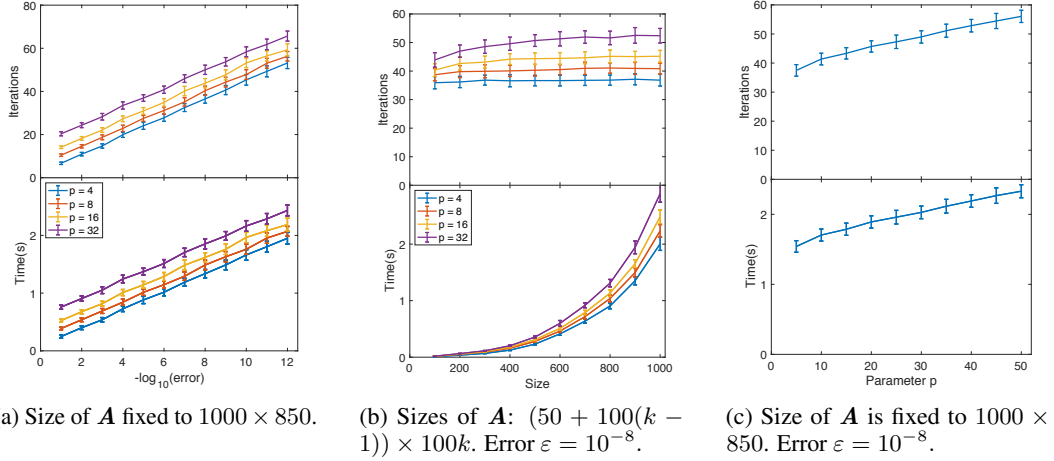
Figure 2: Random Matrix instances. Comparing the number of iterations and time taken by our algorithm with the parameters. Averaged over 100 random samples for A and b. Linear solver used : backslash.

(a) Size of graph fixed to 1000 nodes (around 5000-6000 edges).

(b) Number of nodes: $100k$. Error $\varepsilon = 10^{-8}$.

(c) Size of graph fixed to 1000 nodes (around 5000-6000 edges). Error $\varepsilon = 10^{-8}$.
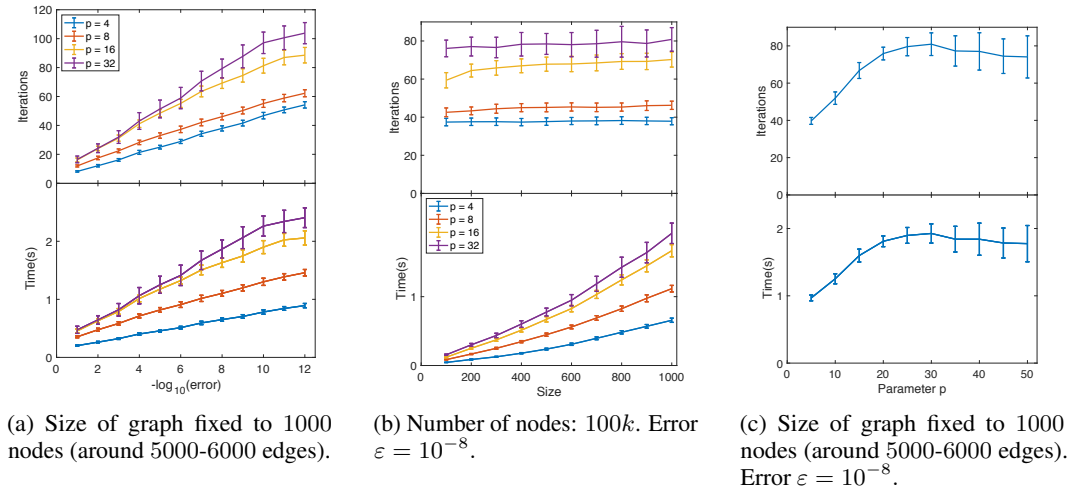
Figure 3: Graph Instances. Comparing the number of iterations and time taken by our algorithm with the parameters. Averaged over 100 graph samples. Linear solver used : backslash.
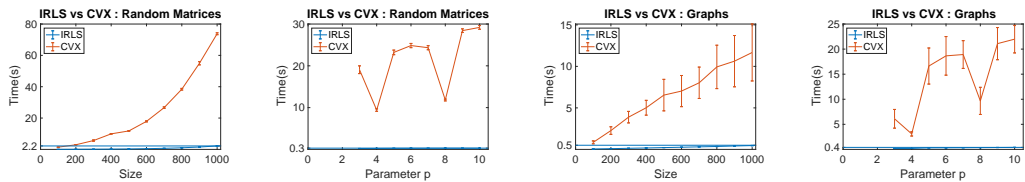

(a) Fixed $p = 8$. Size of matrices: $100k \times (50 + 100(k - 1))$.

(b) Size of matrices fixed to 500×450.

(c) Fixed $p = 8$. The number of nodes : $50k, k = 1, 2, ..., 10$.

(d) Size of graphs fixed to 400 nodes (around 2000 edges).

Figure 4: Averaged over 100 samples. Precision set to $\varepsilon = 10^{-8}$. CVX solver used : SDPT3 for Matrices and Sedumi for Graphs.

problem, ε. We will consider each of these parameters independently and see how our algorithm scales with them for both instances.

Benchmark Comparisons.   We compare the performance of our program with the following:

1. Standard MATLAB optimization package, CVX [GB14, GB08].
2. The most efficient algorithm for ℓ_p-semi supervised learning given in [RCL19] was newton's method with homotopy. We take their hardest problem, and compare the performance of their code with ours by running our algorithm for the same number of iterations as them and showing that we get closer to the optimum, or in other words a smaller error ε, thus showing we converge much faster.

Implementation Details.   We normalize the instances by running our algorithm once and dividing the vector b by the norm of the final objective, so that our norms at the end are around 1. We do this for every instance before we measure the runtime or the iteration count for uniformity and to avoid numerical precision issues. All experiments were performed on MATLAB 2018b on a Desktop ubuntu machine with an Intel Core $i5$-4570 CPU @ $3.20GHz \times 4$ processor and 4GB RAM. For the graph instances, we fix the dimension of the space from which we choose vertices to 10 and the number of labelled vertices to be 10. The graph instances are generated using the code [Rio19] by [RCL19]. Other details specific to the experiment are given in the captions.

4.1   Experimental Results

Dependence on Parameters.   Figure 2 shows the dependence of the number of iterations and runtime on our parameters for random matrices. Similarly for graph instances, Figure 3 shows the dependence of iteration count and runtime with the parameters. As expected from the theoretical guarantees, the number of iterations and runtimes increase linearly with $\log \left(\frac{1}{\varepsilon} \right)$. The dependence on size and p are clearly much better in practice (nearly constant and at most linear respectively) than the theoretical bounds ($m^{1/2}$ and $p^{3.5}$ respectively) for both kinds of instances.

Comparisons with Benchmarks.

- Figure 4 shows the runtime comparison between our IRLS algorithm p-IRLS and CVX. For all instances, we ensured that our final objective was smaller than the objective of the CVX solver. As it is clear for both kinds of instances, our algorithm takes a lot lesser time and also increases more slowly with size and p as compared to CVX. Note that that CVX does a lot better when $p = 2^k$, but it is still at least 30-50 times slower for random matrices and 10-30 times slower for graphs.

- Figure 1 shows the performance of our algorithm when compared to the IRLS/Homotopy method of [RCL19]. We use the same linear solvers for both programs, preconditioned conjugate gradient with an incomplete cholesky preconditioner and run both programs to the same number of iterations. The plots indicate the value ε as described previously. For our IRLS algorithm we indicate our upper bound on ε and for their procedure we indicate a lower bound on ε which is the relative difference in the objectives achieved by the two algorithms. It is clear that our algorithm achieves an error that is orders of magnitudes smaller than the error achieved by their algorithm. This shows that our algorithm has a much faster rate of convergence. Note that there is no guarantee on the convergence of the method used by [RCL19], whereas we prove that our algorithm converges in a small number of iterations.

5   Discussion

To conclude, we present p-IRLS, the first IRLS algorithm that provably converges to a high accuracy solution in a small number of iterations. This settles a problem that has been open for over three decades. Our algorithm is very easy to implement and we demonstrate that it works very well in practice, beating the standard optimization packages by large margins. The theoretical bound on the numbers of iterations has a sub-linear dependence on size and a small polynomial dependence on p, however in practice, we see an almost constant dependence on size and at most linear dependence on p in random instances and graphs. In order to achieve the best theoretical bounds we would require

some form of acceleration. For ℓ_1 and ℓ_∞ regression, it has been shown that it is possible to achieve acceleration, however without geometric convergence. It remains an open problem to give a practical IRLS algorithm which simultaneously has the best possible theoretical convergence bounds.

References

[ACR+16] Ahmed El Alaoui, Xiang Cheng, Aaditya Ramdas, Martin J. Wainwright, and Michael I. Jordan. Asymptotic behavior of ℓ_p-based Laplacian regularization in semi-supervised learning. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *29th Annual Conference on Learning Theory*, volume 49 of *Proceedings of Machine Learning Research*, pages 879–906, Columbia University, New York, New York, USA, 23–26 Jun 2016. PMLR.

[AKPS19] Deeksha Adil, Rasmus Kyng, Richard Peng, and Sushant Sachdeva. Iterative refinement for ℓ_p-norm regression. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1405–1424, 2019.

[AL11] Morteza Alamgir and Ulrike V. Luxburg. Phase transition in the family of p-resistances. In J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. Pereira, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 24*, pages 379–387. Curran Associates, Inc., 2011.

[BB94] Jose Antonio Barreto and C Sidney Burrus. l_p complex approximation using iterative reweighted least squares for fir digital filters. In *Proceedings of ICASSP'94. IEEE International Conference on Acoustics, Speech and Signal Processing*, volume 3, pages III–545. IEEE, 1994.

[BBS94] C.S. Burrus, J.A. Barreto, and I.W. Selesnick. Iterative reweighted least-squares design of fir filters. *Trans. Sig. Proc.*, 42(11):2926–2936, November 1994.

[BCLL18] Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, and Yuanzhi Li. An homotopy method for lp regression provably beyond self-concordance and in input-sparsity time. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 1130–1137, New York, NY, USA, 2018. ACM.

[BL18] Ning Bi and Kaihao Liang. Iteratively reweighted algorithm for signals recovery with coherent tight frame. *Mathematical Methods in the Applied Sciences*, 41(14):5481–5492, 2018.

[BMN04] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In John Shawe-Taylor and Yoram Singer, editors, *Learning Theory*, pages 624–638, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[Bul18] Brian Bullins. Fast minimization of structured convex quartics. *arXiv preprint arXiv:1812.10349*, 2018.

[Bur12] C Sidney Burrus. Iterative reweighted least squares. *OpenStax CNX. Available online: http://cnx. org/contents/92b90377-2b34-49e4-b26f-7fe572db78a1*, 12, 2012.

[BZ13] Nick Bridle and Xiaojin Zhu. p-voltages: Laplacian regularization for semi-supervised learning on high-dimensional data. In *Eleventh Workshop on Mining and Learning with Graphs (MLG2013)*, 2013.

[Cal17] Jeff Calder. Consistency of lipschitz learning with infinite unlabeled data and finite labeled data. *CoRR*, abs/1710.10364, 2017.

[CGK+17] Flavio Chierichetti, Sreenivas Gollapudi, Ravi Kumar, Silvio Lattanzi, Rina Panigrahy, and David P. Woodruff. Algorithms for ℓ_p low-rank approximation. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 806–814, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[CSZ09] O. Chapelle, B. Scholkopf, and A. Zien, Eds. Semi-supervised learning (chapelle, o. et al., eds.; 2006) [book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, March 2009.

[CT05] E. J. Candes and T. Tao. Decoding by linear programming. *IEEE Transactions on Information Theory*, 51(12):4203–4215, Dec 2005.

[CW08] R. Chartrand and Wotao Yin. Iteratively reweighted algorithms for compressive sensing. In *2008 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3869–3872, March 2008.

[DDFG08] I. Daubechies, R. DeVore, M. Fornasier, and S. Gunturk. Iteratively re-weighted least squares minimization: Proof of faster than linear rate for sparse recovery. In *2008 42nd Annual Conference on Information Sciences and Systems*, pages 26–29, March 2008.

[DDFG10] Ingrid Daubechies, Ronald DeVore, Massimo Fornasier, and C. Sinan Gunturk. Iteratively reweighted least squares minimization for sparse recovery. *Communications on Pure and Applied Mathematics*, 63(1):1–38, 2010.

[EDT17] A Elmoataz, X Desquesnes, and M Toutain. On the game p-laplacian on weighted graphs with applications in image processing and data clustering. *European Journal of Applied Mathematics*, 28(6):922–948, 2017.

[ETT15] Abderrahim Elmoataz, Matthieu Toutain, and Daniel Tenbrinck. On the p-laplacian and ∞-laplacian on graphs with applications in image and data processing. *SIAM Journal on Imaging Sciences*, 8(4):2412–2451, 2015.

[EV19] Alina Ene and Adrian Vladu. Improved convergence for ℓ_∞ and ℓ_1 regression via iteratively reweighted least squares. *CoRR*, abs/1902.06391, 2019.

[FRW11] M. Fornasier, H. Rauhut, and R. Ward. Low-rank matrix recovery via iteratively reweighted least squares minimization. *SIAM Journal on Optimization*, 21(4):1614–1640, 2011.

[GB08] Michael Grant and Stephen Boyd. Graph implementations for nonsmooth convex programs. In V. Blondel, S. Boyd, and H. Kimura, editors, *Recent Advances in Learning and Control*, Lecture Notes in Control and Information Sciences, pages 95–110. Springer-Verlag Limited, 2008. `http://stanford.edu/~boyd/graph_dcp.html`.

[GB14] Michael Grant and Stephen Boyd. CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx`, March 2014.

[GR97] I. F. Gorodnitsky and B. D. Rao. Sparse signal reconstruction from limited data using focuss: a re-weighted minimum norm algorithm. *IEEE Transactions on Signal Processing*, 45(3):600–616, March 1997.

[HFE18] Yosra Hafiene, Jalal Fadili, and Abderrahim Elmoataz. Nonlocal p-laplacian variational problems on graphs. *arXiv preprint arXiv:1810.12817*, 2018.

[Kah72] S. W. Kahng. Best l_p-approximation. *Math. Comput.*, 26(118):505–508, 1972.

[Kar70] L.A Karlovitz. Construction of nearest points in the l_p, p even, and l_∞ norms. i. *Journal of Approximation Theory*, 3(2):123 – 127, 1970.

[KPSW19] Rasmus Kyng, Richard Peng, Sushant Sachdeva, and Di Wang. Flows in almost linear time via adaptive preconditioning. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2019, pages 902–913, New York, NY, USA, 2019. ACM.

[KRSS15] Rasmus Kyng, Anup Rao, Sushant Sachdeva, and Daniel A. Spielman. Algorithms for lipschitz learning on graphs. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015*, pages 1190–1223, 2015.

[Law61] C. L. Lawson. Contribution to the theory of linear least maximum approximation. *Ph.D. dissertation, Univ. Calif.*, 1961.

[MPT+18] Chris J Maddison, Daniel Paulin, Yee Whye Teh, Brendan O'Donoghue, and Arnaud Doucet. Hamiltonian descent methods. *arXiv preprint arXiv:1809.05042*, 2018.

[NN94] Y. Nesterov and A. Nemirovskii. *Interior-Point Polynomial Algorithms in Convex Programming*. Society for Industrial and Applied Mathematics, 1994.

[NSZ09] Boaz Nadler, Nathan Srebro, and Xueyuan Zhou. Statistical analysis of semi-supervised learning: The limit of infinite unlabelled data. 2009.

[Osb85] M. R. Osborne. *Finite Algorithms in Optimization and Data Analysis*. John Wiley & Sons, Inc., New York, NY, USA, 1985.

[RCL19] Mauricio Flores Rios, Jeff Calder, and Gilad Lerman. Algorithms for ℓ_p-based semi-supervised learning on graphs. *CoRR*, abs/1901.05031, 2019.

[Ric64] J.R. Rice. *The Approximation of Functions, By John R. Rice*. Addison-Wesley Series in Computer Science and Information Processing. 1964.

[Rio19] Mauricio Flores Rios. Laplacian_lp_graph_ssl. `https://github.com/mauriciofloresML/Laplacian_Lp_Graph_SSL`, 2019.

[ST17] Dejan Slepcev and Matthew Thorpe. Analysis of p-laplacian regularization in semi-supervised learning. *CoRR*, abs/1707.06213, 2017.

[SV16a] Damian Straszak and Nisheeth K. Vishnoi. IRLS and slime mold: Equivalence and convergence. *CoRR*, abs/1601.02712, 2016.

[SV16b] Damian Straszak and Nisheeth K. Vishnoi. Natural algorithms for flow problems. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1868–1883, 2016.

[SV16c] Damian Straszak and Nisheeth K. Vishnoi. On a natural dynamics for linear programming. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science, Cambridge, MA, USA, January 14-16, 2016*, page 291, 2016.

[VB99] R. A. Vargas and C. S. Burrus. Adaptive iterative reweighted least squares design of l_p fir filters. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No.99CH36258)*, volume 3, pages 1129–1132 vol.3, March 1999.

[VB12] Ricardo A. Vargas and C. Sidney Burrus. Iterative design of l_p digital filters. *CoRR*, abs/1207.4526, 2012.

[ZB11] Xueyuan Zhou and Mikhail Belkin. Semi-supervised learning by higher order regularization. In Geoffrey Gordon, David Dunson, and Miroslav Dudík, editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 892–900, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR.

[ZBL+04] D. Zhou, O. Bousquet, TN. Lal, J. Weston, and B. Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems 16*, pages 321–328, Cambridge, MA, USA, June 2004. Max-Planck-Gesellschaft, MIT Press.

[ZGL03] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[Zhu05] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

Appendix A   Proofs from Section 3

A.1   Proof of Lemma 3.2

Lemma 3.2. *(Iterative Refinement).*   Let $p \geq 2$, and $\kappa \geq 1$.   Starting from $\boldsymbol{x}^{(0)} = \arg\min_{\boldsymbol{C}\boldsymbol{x}=\boldsymbol{d}} \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$, and iterating as, $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \Delta$, where Δ is a κ-approximate solution to the residual problem (Definition 2.1), we get an ε-approximate solution to (3) in at most $O\left(p^2 \kappa \log\left(\frac{m}{\varepsilon}\right)\right)$ calls to a κ-approximate solver for the residual problem.

We first show that we can upper and lower bound the change in objective by a linear term plus a quadratically smoothed function.

Lemma A.1. *For any $\boldsymbol{x}, \Delta$ and $p \geq 2$, we have for $\boldsymbol{r} = |\boldsymbol{x}|^{p-2}$ and $\boldsymbol{g} = p|\boldsymbol{x}|^{p-2}\boldsymbol{x}$,*

$$\frac{p}{8}\sum_e \boldsymbol{r}_e \Delta_e^2 + \frac{1}{2^{p+1}}\|\Delta\|_p^p \leq \|\boldsymbol{x} + \Delta\|_p^p - \|\boldsymbol{x}\|_p^p - \boldsymbol{g}^\top \Delta \leq 2p^2 \sum_e \boldsymbol{r}_e \Delta_e^2 + p^p \|\Delta\|_p^p.$$

The proof of the above lemma is long and hence deferred to the end of this section. Applying the above lemma on our objective we get,

$$\frac{p}{8}(\boldsymbol{A}\Delta)^\top \boldsymbol{R}\boldsymbol{A}\Delta + \frac{1}{2^{p+1}}\|\boldsymbol{A}\Delta\|_p^p \leq \|\boldsymbol{A}(\boldsymbol{x}+\Delta)-\boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}\|_p^p - \boldsymbol{g}^\top \boldsymbol{A}\Delta \leq 2p^2(\boldsymbol{A}\Delta)^\top \boldsymbol{R}\boldsymbol{A}\Delta + p^p\|\boldsymbol{A}\Delta\|_p^p, \tag{5}$$

where $\boldsymbol{R}$ is the diagonal matrix with entries $|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}|^{p-2}$ and $\boldsymbol{g} = p\boldsymbol{R}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})$. We next show the relation between the residual problem defined in the preliminaries and the change in objective value when $\boldsymbol{x}$ is updated by Δ.

Lemma A.2. *For any $\boldsymbol{x}, \Delta$ and $p \geq 2$ and $\lambda = 16p$,*

$$\gamma(\Delta) \leq \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}(\boldsymbol{x} - \Delta) - \boldsymbol{b}\|_p^p,$$

and

$$\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}(\boldsymbol{x} - \lambda\Delta) - \boldsymbol{b}\|_p^p \leq \lambda\gamma(\Delta).$$

Proof. The first inequality directly follows from (5). For the second inequality,

$$\begin{aligned}
\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}(\boldsymbol{x} - \lambda\Delta) - \boldsymbol{b}\|_p^p &\leq \lambda\boldsymbol{g}^\top \Delta - \lambda^2\frac{p}{8}\Delta^\top \boldsymbol{A}^\top \boldsymbol{R}\boldsymbol{A}\Delta - \lambda^p\frac{1}{2^{p+1}}\|\boldsymbol{A}\Delta\|_p^p \\
&= \lambda\left(\boldsymbol{g}^\top \Delta - \lambda\frac{p}{8}\Delta^\top \boldsymbol{A}^\top \boldsymbol{R}\boldsymbol{A}\Delta - \lambda^{p-1}\frac{1}{2^{p+1}}\|\boldsymbol{A}\Delta\|_p^p\right) \\
&\leq \lambda\left(\boldsymbol{g}^\top \boldsymbol{A}\Delta - 2p^2\Delta^\top \boldsymbol{A}^\top \boldsymbol{R}\boldsymbol{A}\Delta - p^p\|\boldsymbol{A}\Delta\|_p^p\right).
\end{aligned}$$

$\square$

A.1.1   Proof of Lemma Iterative Refinement

Proof. Let $\widetilde{\Delta}$ be a κ-approximate solution to the residual problem. Using this fact and Lemma A.2 for $\Delta = \frac{\boldsymbol{x} - \boldsymbol{x}^\star}{\lambda}$, we get,

$$\gamma(\widetilde{\Delta}) \geq \frac{1}{\kappa}\gamma(\Delta^\star) \geq \frac{1}{\kappa}\gamma\left(\frac{\boldsymbol{x} - \boldsymbol{x}^\star}{\lambda}\right) \geq \frac{1}{\lambda\kappa}\left(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p - OPT\right).$$

Also,

$$\begin{aligned}
\left\|\boldsymbol{A}(\boldsymbol{x} - \widetilde{\Delta}) - \boldsymbol{b}\right\|_p^p - OPT &\leq \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p - \gamma(\widetilde{\Delta}) - OPT \\
&\leq \left(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p - OPT\right) - \frac{1}{\lambda\kappa}\left(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p - OPT\right) \\
&= \left(1 - \frac{1}{\lambda\kappa}\right)\left(\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p - OPT\right).
\end{aligned}$$

Now, after t iterations,

$$\left\|\boldsymbol{A}(\boldsymbol{x}^{(t)} - \widetilde{\Delta}) - \boldsymbol{b}\right\|_p^p - OPT \leq \left(1 - \frac{1}{\lambda\kappa}\right)^t\left(\left\|\boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}\right\|_p^p - OPT\right) \leq \left(1 - \frac{1}{\lambda\kappa}\right)^t m^{(p-2)/2}OPT.$$

So to get an ε-approximate solution, we need $8p^2\kappa \log(m/\varepsilon)$ iterations for our value of λ. $\square$

A.1.2 Proof of Lemma A.1

Proof. To show this, we show that the above holds for all coordinates. For a single coordinate, the above expression is equivalent to proving,

$$\frac{p}{8}|x|^{p-2}\Delta^2 + \frac{1}{2^{p+1}}|\Delta|^p \leq |\boldsymbol{x}+\Delta|^p - |\boldsymbol{x}|^p - p|x|^{p-1}\,sgn(x)\Delta \leq 2p^2|x|^{p-2}\Delta^2 + p^p|\Delta|^p\,.$$

Let $\Delta = \alpha x$. Since the above clearly holds for $x = 0$, it remains to show for all α,

$$\frac{p}{8}\alpha^2 + \frac{1}{2^{p+1}}|\alpha|^p \leq |1+\alpha|^p - 1 - p\alpha \leq 2p^2\alpha^2 + p^p|\alpha|^p\,.$$

1. $\alpha \geq 1$:
   In this case, $1 + \alpha \leq 2\alpha \leq p \cdot \alpha$. So, $|1+\alpha|^p \leq p^p|\alpha|^p$ and the right inequality directly holds. To show the other side, let

   $$h(\alpha) = (1+\alpha)^p - 1 - p\alpha - \frac{p}{8}\alpha^2 - \frac{1}{2^{p+1}}\alpha^p.$$

   We have,

   $$h'(\alpha) = p(1+\alpha)^{p-1} - p - \frac{p}{4}\alpha - \frac{p}{2^{p+1}}\alpha^{p-1}$$

   and

   $$h''(\alpha) = p(p-1)(1+\alpha)^{p-2} - \frac{p}{4} - \frac{p(p-1)}{2^{p+1}}\alpha^{p-2} \geq 0.$$

   Since $h''(\alpha) \geq 0$, $h'(\alpha) \geq h'(1) \geq 0$. So h is an increasing function in α and $h(\alpha) \geq h(1) \geq 0$.

2. $\alpha \leq -1$:
   Now, $|1+\alpha| \leq 1 + |\alpha| \leq p \cdot |\alpha|$, and $2\alpha^2 p^2 - |\alpha|\,p \geq 0$. As a result,

   $$|1+\alpha|^p \leq -|\alpha|\,p + 2\alpha^2 p^2 + p^p \cdot |\alpha|^p$$

   which gives the right inequality. Consider,

   $$h(\alpha) = |1+\alpha|^p - 1 - p\alpha - \frac{p}{8}\alpha^2 - \frac{1}{2^{p+1}}|\alpha|^p.$$

   $$h'(\alpha) = -p|1+\alpha|^{p-1} - p - \frac{p}{4}\alpha + p\frac{1}{2^{p+1}}|\alpha|^{p-1}.$$

   Let $\beta = -\alpha$. The above expression now becomes,

   $$-p(\beta-1)^{p-1} - p + \frac{p}{4}\beta + p\frac{1}{2^{p+1}}\beta^{p-1}.$$

   We know that $\beta \geq 1$. When $\beta \geq 2$, $\frac{\beta}{2} \leq \beta - 1$ and $\frac{\beta}{2} \leq \left(\frac{\beta}{2}\right)^{p-1}$. This gives us,

   $$\frac{p}{4}\beta + p\frac{1}{2^{p+1}}\beta^{p-1} \leq \frac{p}{2}\left(\frac{\beta}{2}\right)^{p-1} + \frac{p}{2}\left(\frac{\beta}{2}\right)^{p-1} \leq p(\beta-1)^{p-1}$$

   giving us $h'(\alpha) \leq 0$ for $\alpha \leq -2$. When $\beta \leq 2$, $\frac{\beta}{2} \geq \left(\frac{\beta}{2}\right)^{p-1}$ and $\frac{\beta}{2} \leq 1$.

   $$\frac{p}{4}\beta + p\frac{1}{2^{p+1}}\beta^{p-1} \leq \frac{p}{2}\cdot\frac{\beta}{2} + \frac{p}{2}\cdot\frac{\beta}{2} \leq p$$

   giving us $h'(\alpha) \leq 0$ for $-2 \leq \alpha \leq -1$. Therefore, $h'(\alpha) \leq 0$ giving us, $h(\alpha) \geq h(-1) \geq 0$, thus giving the left inequality.

3. $|\alpha| \leq 1$:
   Let $s(\alpha) = 1 + p\alpha + 2p^2\alpha^2 + p^p|\alpha|^p - (1+\alpha)^p$. Now,

   $$s'(\alpha) = p + 4p^2\alpha + p^{p+1}|\alpha|^{p-1}\,sgn(\alpha) - p(1+\alpha)^{p-1}.$$

13

When $\alpha \leq 0$, we have,

$$s'(\alpha) = p + 4p^2\alpha - p^{p+1}|\alpha|^{p-1} - p(1+\alpha)^{p-1}.$$

and

$$s''(\alpha) = 4p^2 + p^{p+1}(p-1)|\alpha|^{p-2} - p(p-1)(1+\alpha)^{p-1} \geq 2p^2 + p^{p+1}(p-1)|\alpha|^{p-2} - p(p-1) \geq 0.$$

So s' is an increasing function of α which gives us, $s'(\alpha) \leq s'(0) = 0$. Therefore s is a decreasing function, and the minimum is at 0 which is 0. This gives us our required inequality for $\alpha \leq 0$. When $\alpha \geq \frac{1}{p-1}$, $1 + \alpha \leq p \cdot \alpha$ and $s'(\alpha) \geq 0$. We are left with the range $0 \leq \alpha \leq \frac{1}{p-1}$. Again, we have,

$$s''(\alpha) = 4p^2 + p^{p+1}(p-1)|\alpha|^{p-2} - p(p-1)(1+\alpha)^{p-1}$$

$$\geq 4p^2 + p^{p+1}(p-1)|\alpha|^{p-2} - p(p-1)(1 + \frac{1}{p-1})^{p-1}$$

$$\geq 4p^2 + p^{p+1}(p-1)|\alpha|^{p-2} - p(p-1)e, \text{When } p \text{ gets large the last term approaches } e$$

$$\geq 0.$$

Therefore, s' is an increasing function, $s'(\alpha) \geq s'(0) = 0$. This implies s is an increasing function, giving, $s(\alpha) \geq s(0) = 0$ as required.

To show the other direction,

$$h(\alpha) = (1+\alpha)^p - 1 - p\alpha - \frac{p}{8}\alpha^2 - \frac{1}{2^{p+1}}|\alpha|^p \geq (1+\alpha)^p - 1 - p\alpha - \frac{p}{8}\alpha^2 - \frac{p}{8}\alpha^2 = (1+\alpha)^p - 1 - p\alpha - \frac{p}{4}\alpha^2.$$

Now, since $p \geq 2$,

$$\left((1+\alpha)^{p-2} - 1\right)sgn(\alpha) \geq 0$$

$$\Rightarrow \left((1+\alpha)^{p-1} - 1 - \alpha\right)sgn(\alpha) \geq 0$$

$$\Rightarrow \left(p(1+\alpha)^{p-1} - p - \frac{p}{2}\alpha\right)sgn(\alpha) \geq 0$$

We thus have, $h'(\alpha) \geq 0$ when α is positive and $h'(\alpha) \leq 0$ when α is negative. The minimum of h is at 0 which is 0. This concludes the proof of this case.

$\square$

A.2 Proof of Lemma 3.4

We require bounding the objective of program 4. To do that we first give a bound on a decision version of the residual problem, and then relate this problem with problem 4.

Lemma A.3. *Let i be such that the optimum of the residual problem, $\gamma(\Delta^\star) \in (i/2, \lambda i]$. Then the following problem has optimum at most λi.*

$$\min_{\Delta \in \mathbb{R}^m} \quad 2p^2(A\Delta)^\top RA\Delta + p^p\|A\Delta\|_p^p$$

$$g^\top A\Delta = i/2 \tag{6}$$

$$C\Delta = 0.$$

Proof.

$$\gamma(\Delta^\star) = g^\top A\Delta^\star - 2p^2(A\Delta^\star)^\top RA\Delta^\star + p^p\|A\Delta^\star\|_p^p \in (i/2, \lambda i].$$

The above objective is strictly non-negative since $\Delta = 0$ is a feasible solution and we are maximizing the objective. Since the last 2 terms are strictly non-negative, we must have,

$$g^\top A\Delta^\star \geq i/2.$$

Since $\Delta^\star$ is the optimum and satisfies $C\Delta^\star = 0$,

$$\frac{d}{d\lambda}\left(g^\top \lambda A\Delta^\star - 2p^2\lambda^2(A\Delta^\star)^\top RA\Delta^\star - \lambda^p p^p\|A\Delta^\star\|_p^p\right)_{\lambda=1} = 0.$$

We thus have,

$$\boldsymbol{g}^\top \boldsymbol{A}\Delta^\star - 2p^2(\boldsymbol{A}\Delta^\star)^\top \boldsymbol{R}\boldsymbol{A}\Delta^\star - p^p \|\boldsymbol{A}\Delta^\star\|_p^p = 2p^2(\boldsymbol{A}\Delta^\star)^\top \boldsymbol{R}\boldsymbol{A}\Delta^\star + (p-1)p^p \|\boldsymbol{A}\Delta^\star\|_p^p,$$

which gives the following bound.

$$2p^2(\boldsymbol{A}\Delta^\star)^\top \boldsymbol{R}\boldsymbol{A}\Delta^\star + p^p \|\boldsymbol{A}\Delta^\star\|_p^p \leq \boldsymbol{g}^\top \boldsymbol{A}\Delta^\star - 2p^2(\boldsymbol{A}\Delta^\star)^\top \boldsymbol{R}\boldsymbol{A}\Delta^\star - p^p \|\boldsymbol{A}\Delta^\star\|_p^p \leq \lambda i.$$

For notational convenience, let function $h_p(\boldsymbol{r}, \Delta) = 2p^2(\boldsymbol{A}\Delta)^\top \boldsymbol{R}\boldsymbol{A}\Delta + p^p \|\boldsymbol{A}\Delta\|_p^p$. Now, we know that, $\boldsymbol{g}^\top \boldsymbol{A}\Delta^\star \geq i/2$ and $\boldsymbol{g}^\top \boldsymbol{A}\Delta^\star - h_p(\boldsymbol{r}, \Delta^\star) \leq \lambda i$. This gives,

$$i/2 \leq \boldsymbol{g}^\top \boldsymbol{A}\Delta^\star \leq h_p(\boldsymbol{r}, \Delta^\star) + \lambda i \leq 2\lambda i.$$

Let $\Delta = \delta \Delta^\star$, where $\delta = \frac{i}{2\boldsymbol{g}^\top \boldsymbol{A}\Delta^\star}$. Note that $\delta \in [1/4\lambda, 1]$. Now, $\boldsymbol{g}^\top \boldsymbol{A}\Delta = i/2$ and,

$$h_p(\boldsymbol{r}, \Delta) \leq \max\{\delta^2, \delta^p\} h_p(\boldsymbol{r}, \Delta^\star) \leq \lambda i.$$

Note that this Δ satisfies the constraints of program (6) and has an optimum at most λi. So the optimum of the program must have an objective at most λi. $\qquad \square$

Claim A.4. *If the optimal objective of program (6) is at most Z, then the optimum objective of program (4) is at most $\frac{Z}{2p^2} + \frac{i^{(p-2)/p}Z^{2/p}}{2p^2}$.*

Proof. Let $\Delta^\star$ denote the optimizer of (6) and $\widetilde{\Delta}$ be the optimizer of (4). Since the optimum objective of (6) is at most Z, we have we have $\|\boldsymbol{A}\Delta^\star\|_p^p \leq \frac{Z}{p^p}$. This implies that $\Delta^{\star\top}\boldsymbol{A}^\top \boldsymbol{A}\Delta^\star \leq \frac{Z^{2/p}}{p^2}m^{(p-2)/p}$. Since $\Delta^\star$ is a feasible solution of (4), we have for our value of s,

$$\widetilde{\Delta}^\top \boldsymbol{A}^\top (\boldsymbol{R} + s^{(t)}\boldsymbol{I})\boldsymbol{A}\widetilde{\Delta} \leq \Delta^{\star\top}\boldsymbol{A}^\top (\boldsymbol{R} + s^{(t)}\boldsymbol{I})\boldsymbol{A}\Delta^\star \leq \frac{Z}{2p^2} + \frac{i^{(p-2)/p}Z^{2/p}}{2p^2}.$$

$\qquad \square$

Lemma 3.4. *(Check Progress). Let α_0 be as defined in line (4) of Algorithm 2 and $\widetilde{\Delta}$ the solution of program (4). If $i/2 < \frac{(\|\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}^\star - b\|_p^p)}{16p} \leq i$, then $\gamma(\alpha_0 \cdot \widetilde{\Delta}) \geq \frac{\alpha_0 i}{4}$.*

Proof. Since,

$$i/2 < \frac{(\|\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}^\star - b\|_p^p)}{16p} \leq i,$$

we know that the optimum of the residual problem lies between $(i/2, \lambda i]$. From Claim A.4, we know that $(\boldsymbol{A}\widetilde{\Delta})^\top (\boldsymbol{R} + s\boldsymbol{I})(\boldsymbol{A}\widetilde{\Delta}) \leq \lambda i/p^2$. Also, note that $\alpha_0 + k\alpha_0^{p-1} \leq \frac{1}{8\lambda}$. Consider the following,

$$\begin{aligned}
\gamma(\alpha_0 \cdot \widetilde{\Delta}) &= \alpha_0 \boldsymbol{g}^\top \boldsymbol{A}\widetilde{\Delta} - \alpha_0^2 2p^2(\boldsymbol{A}\widetilde{\Delta})^\top \boldsymbol{R}(\boldsymbol{A}\widetilde{\Delta}) - \alpha_0^p p^p \left\|\boldsymbol{A}\widetilde{\Delta}\right\|_p^p \\
&\geq \alpha_0 \boldsymbol{g}^\top \boldsymbol{A}\widetilde{\Delta} - (\alpha_0^2 + k\alpha_0^p)2p^2(\boldsymbol{A}\widetilde{\Delta})^\top (\boldsymbol{R} + s\boldsymbol{I})(\boldsymbol{A}\widetilde{\Delta}) \\
&= \alpha_0 \left(\boldsymbol{g}^\top \boldsymbol{A}\widetilde{\Delta} - (\alpha_0 + k\alpha_0^{p-1})2p^2(\boldsymbol{A}\widetilde{\Delta})^\top (\boldsymbol{R} + s\boldsymbol{I})(\boldsymbol{A}\widetilde{\Delta})\right) \\
&\geq \alpha_0 \left(\boldsymbol{g}^\top \boldsymbol{A}\widetilde{\Delta} - \frac{1}{8\lambda}2p^2(\boldsymbol{A}\widetilde{\Delta})^\top (\boldsymbol{R} + s\boldsymbol{I})(\boldsymbol{A}\widetilde{\Delta})\right) \\
&\geq \alpha_0 \left(\frac{i}{2} - \frac{1}{8\lambda}2\lambda i\right) \\
&\geq \frac{\alpha_0}{4}i
\end{aligned}$$

$\qquad \square$

A.3 Proof of Lemma 3.5

Lemma 3.5. *(Invariant)* *At every iteration of the while loop, we have* $\boldsymbol{C}\boldsymbol{x}^{(t)} = \boldsymbol{d}$, $\frac{(\|\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{b}\|_p^p)}{16p} < i$ *and* $i \geq \frac{\varepsilon}{16p(1+\varepsilon)}\|\boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}\|_p^p m^{-(p-2)/2}$,

Proof. We use induction to show this. Initially we set, $i = \|\boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}\|_p^p / 16p$. When the optimum is not 0, this is greater than $(\|\boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{b}\|_p^p)/16p$. When the optimum is 0, the initial solution (2-norm minimizer) will also give zero and we can stop our procedure. Therefore, the claim holds for $t = 1$. Suppose at iteration t the claim holds. Since the objective is non-increasing, we know that,

$$\|\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{b}\|_p^p \geq \|\boldsymbol{A}\boldsymbol{x}^{(t+1)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{b}\|_p^p.$$

Let $\widetilde{\Delta}$ denote the solution returned in iteration $t+1$. At iteration $t+1$, if $(\|\boldsymbol{A}\boldsymbol{x}^{(t+1)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{b}\|_p^p)/16p \in (i/2, i]$, from Lemma 3.4, we will always have $\gamma(\alpha_0\widetilde{\Delta}) \geq \alpha_0 i/4$. Since we have the bound on the objective at $t + 1$ minus the optimum, Lemma A.2 implies that $\gamma(\Delta^\star) \in (i/2, \lambda i]$. Now from Lemma A.3 followed by Claim A.4, we always have $(\boldsymbol{A}\widetilde{\Delta})^\top (\boldsymbol{R} + s\boldsymbol{I})\boldsymbol{A}\widetilde{\Delta} \leq \lambda i/p^2$. So the algorithm does not reduce i and as a result our claim holds for $t + 1$. Otherwise, we know that $(\|\boldsymbol{A}\boldsymbol{x}^{(t+1)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{b}\|_p^p)/16p \leq i/2$ and the algorithm might reduce i by half if either of the two conditions are true. However, the claim still holds. Therefore, i is always at least $(\|\boldsymbol{A}\boldsymbol{x}^{(t+1)} - \boldsymbol{b}\|_p^p - \|\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{b}\|_p^p)/16p$.

We start with a solution $\boldsymbol{x}^{(0)}$ that minimizes the ℓ_2 norm. Therefore, the following holds,

$$\|\boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}\|_p^p \leq \|\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{b}\|_p^p m^{(p-2)/2}.$$

The value of i is the minimum at termination. Therefore, it is sufficient to prove the above bound for the termination condition. Our condition gives us the following at termination (the left inequality holds because otherwise, we would have terminated in the previous iteration).

$$i \geq \frac{\varepsilon}{16p(1+\varepsilon)}\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p \geq \frac{i}{2}.$$

This implies,

$$i \geq \frac{\varepsilon}{16p(1+\varepsilon)}\|\boldsymbol{A}\boldsymbol{x}^\star - \boldsymbol{b}\|_p^p \geq \frac{\varepsilon}{16p(1+\varepsilon)}\|\boldsymbol{A}\boldsymbol{x}^{(0)} - \boldsymbol{b}\|_p^p m^{-(p-2)/2}.$$

We next prove the second claim. Initially our solution satisfies $\boldsymbol{C}\boldsymbol{x}^{(0)} = \boldsymbol{d}$. Assuming the condition holds at iteration t, we show it for $t + 1$. At every iteration we solve for $\widetilde{\Delta}$ under the constraint $\boldsymbol{C}\widetilde{\Delta} = 0$. Our update rule, $\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \alpha\widetilde{\Delta}$ gives us,

$$\boldsymbol{C}\boldsymbol{x}^{(t+1)} = \boldsymbol{C}\boldsymbol{x}^{(t)} - \alpha\boldsymbol{C}\widetilde{\Delta} = \boldsymbol{d} - \alpha \cdot 0 = \boldsymbol{d}.$$

$\square$

A.4 Proof of Lemma 3.3

Our approximation depends on the quantity α_0 which is defined in the algorithm. This depends on the value of k, the ratio of the p-norm term to the square term. Therefore, in order to bound the approximation, we first give a bound on k.

Lemma A.5. *Let* $\widetilde{\Delta}$ *the optimum of* (4) *and let* $k = \frac{p^p\|\boldsymbol{A}\widetilde{\Delta}\|_p^p}{2p^2\widetilde{\Delta}^\top \boldsymbol{A}^\top(\boldsymbol{R}+s\boldsymbol{I})\boldsymbol{A}\widetilde{\Delta}}$. *If the optimum of* (4) *is at most* $\lambda i/p^2$, *then* k *is at most* $(32pm)^{(p-2)/2}$ *for* $\lambda = 16p$.

Proof. Since, $s\boldsymbol{I} \preceq \boldsymbol{R} + s\boldsymbol{I}$,

$$\left\|\boldsymbol{A}\widetilde{\Delta}\right\|_2^2 = \widetilde{\Delta}^\top \boldsymbol{A}^\top \boldsymbol{A}\widetilde{\Delta} \leq \frac{1}{s}\widetilde{\Delta}^\top \boldsymbol{A}^\top(\boldsymbol{R} + s\boldsymbol{I})\boldsymbol{A}\widetilde{\Delta}$$

and,

$$\left\|\boldsymbol{A}\widetilde{\Delta}\right\|_p^p \leq \left\|\boldsymbol{A}\widetilde{\Delta}\right\|_2^p \leq \frac{1}{s}\left(\widetilde{\Delta}^\top \boldsymbol{A}^\top \boldsymbol{A}\widetilde{\Delta}\right)^{(p-2)/2}\widetilde{\Delta}^\top \boldsymbol{A}^\top(\boldsymbol{R} + s\boldsymbol{I})\boldsymbol{A}\widetilde{\Delta}.$$

We also have, $\widetilde{\Delta}^\top A^\top A \widetilde{\Delta} \le \frac{2\lambda}{p^2} m^{(p-2)/p} i^{2/p}$. Combining these,

$$\frac{p^p \left\| A\widetilde{\Delta} \right\|_p^p}{2p^2 \widetilde{\Delta}^\top A^\top (R+sI) A\widetilde{\Delta}} \le \frac{p^p}{2p^2} \frac{2m^{(p-2)/p}}{i^{(p-2)/p}} \left(\frac{2\lambda}{p^2} m^{(p-2)/p} i^{2/p} \right)^{(p-2)/2}$$

$$\le (4\sqrt{2})^{p-2} p^{(p-2)/2} m^{(p-2)/p} m^{(p-2)^2/2p}$$

$$= (32pm)^{(p-2)/2}$$

$\square$

Lemma 3.3. *(Approximation). Let i be such that $(\|Ax^{(t)} - b\|_p^p - \|Ax^\star - b\|_p^p)/16p \in (i/2, i]$ and R, g, s, α be as defined in lines (5), (6), (7) and (9) of Algorithm 1. Let $\widetilde{\Delta}$ be the solution of the following program,*

$$\arg\min_\Delta \Delta^\top A^\top (R+sI) A\Delta \quad s.t. \quad g^\top A\Delta = i/2, \, C\Delta = 0. \tag{4}$$

Then, $\alpha \cdot \widetilde{\Delta}$ is an $O\left(p^{1.5} m^{\frac{p-2}{2(p-1)}} \right)$- approximate solution to the residual problem.

Proof. In the algorithm we choose α such that given $\widetilde{\Delta}$, $\alpha = argmin_\delta \|A(x - \delta\widetilde{\Delta}) - b\|_p^p$. Let α_0 be as defined in Algorithm 2. From Lemma 3.4, we also know that $\gamma(\alpha_0\widetilde{\Delta}) \ge \frac{\alpha_0}{4} i$. Now, since the residual function is a convex function with value zero at the zero vector, we know that $\gamma(\widetilde{\Delta}/\lambda) \le \frac{1}{\lambda} \gamma(\widetilde{\Delta})$ (for our value of $\lambda = 16p$).

$$\gamma(\alpha\widetilde{\Delta}) \ge \lambda\gamma(\alpha\widetilde{\Delta}/\lambda)$$

$$\ge \|Ax - b\|_p^p - \|A(x - \alpha\widetilde{\Delta}) - b\|_p^p$$

$$\ge \|Ax - b\|_p^p - \|A(x - \alpha_0\widetilde{\Delta}) - b\|_p^p$$

$$\ge \gamma(\alpha_0 \cdot \widetilde{\Delta})$$

$$\ge \frac{\alpha_0}{4} i$$

$$\ge \frac{\alpha_0}{4\lambda} OPT.$$

From Lemma6 and Claim A.4 we know that the objective of (4) is at most $\lambda i/p^2$. The lemma now follows from Lemma A.5, which bounds the factor $O(\lambda/\alpha_0) \le O(p^{1+\frac{p-2}{2(p-1)}} m^{\frac{p-2}{2(p-1)}}) \le O(p^{1.5} m^{\frac{p-2}{2(p-1)}})$. $\square$

A.5 Proof of Lemma 3.6

Lemma 3.6. *(Termination). Let i be such that $(\|Ax^{(t)} - b\|_p^p - \|Ax^\star - b\|_p^p)/16p \in (i/2, i]$. Then,*

$$i \le \frac{\varepsilon}{16p(1+\varepsilon)} \|Ax^{(t)} - b\|_p^p \Rightarrow \|Ax^{(t)} - b\|_p^p \le (1+\varepsilon)\text{OPT}.$$

and,

$$\|Ax^{(t)} - b\|_p^p \le (1+\varepsilon)\text{OPT} \Rightarrow i \le 2\frac{\varepsilon}{16p(1+\varepsilon)} \|Ax^{(t)} - b\|_p^p.$$

Proof. We first show the forward implication. We know that,

$$16pi/2 \le \left\| Ax^{(t)} - b \right\|_p^p - OPT \le 16pi.$$

This implies,

$$\frac{\left\| Ax^{(t)} - b \right\|_p^p - OPT}{16p} \le \frac{\varepsilon}{16p(1+\varepsilon)} \left\| Ax^{(t)} - b \right\|_p^p$$

$$\Rightarrow \left\| Ax^{(t)} - b \right\|_p^p \frac{1}{1+\varepsilon} \le OPT$$

$$\Rightarrow \left\| Ax^{(t)} - b \right\|_p^p \le (1+\varepsilon)OPT.$$

For the other direction we have,

$$\frac{\left\|\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{b}\right\|_p^p}{1 + \varepsilon} \leq OPT.$$

$$i \leq 2\frac{\left\|\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{b}\right\|_p^p - OPT}{16p} \leq 2\frac{\left\|\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{b}\right\|_p^p}{16p}\left(1 - \frac{1}{1 + \varepsilon}\right) \leq \frac{2\varepsilon}{1 + \varepsilon}\frac{\left\|\boldsymbol{A}\boldsymbol{x}^{(t)} - \boldsymbol{b}\right\|_p^p}{16p}.$$

$\square$

Appendix B   Converting ℓ_p-Laplacian Minimization to Regression Form

Define the following terms:

- n denote the number of vertices.
- l denote the number of labels.
- $\boldsymbol{B}$ denote the edge-vertex adjacency matrix.
- $\boldsymbol{g}$ denote the vector of labels for the l labelled vertices.
- $\boldsymbol{W}$ denote the diagonal matrix with weights of the edges.

Set $\boldsymbol{A} = \boldsymbol{W}^{1/p}\boldsymbol{B}$ and $\boldsymbol{b} = -\boldsymbol{B}[:, n : n + l]\boldsymbol{g}$. Now $\|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_p^p$ is equal to the ℓ_p laplacian and we can use our IRLS algorithm to find the $\boldsymbol{x}$ that minimizes this.

Appendix C   Solving ℓ_2 Problems under Subspace Constraints

C.1   Finding the Initial Solution

We want to solve:

$$\min_{\boldsymbol{x}} \quad \|\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}\|_2^2$$
$$\boldsymbol{C}\boldsymbol{x} = \boldsymbol{d}.$$

Using Lagrangian duality and noting that strong duality holds, we can write the above as,

$$L(\boldsymbol{x}, \boldsymbol{v}) = \min_{\boldsymbol{x}}\max_{\boldsymbol{v}} \quad (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) + \boldsymbol{v}^\top(\boldsymbol{d} - \boldsymbol{C}\boldsymbol{x})$$
$$= \max_{\boldsymbol{v}}\min_{\boldsymbol{x}} \quad (\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b})^\top(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}) + \boldsymbol{v}^\top(\boldsymbol{d} - \boldsymbol{C}\boldsymbol{x}).$$

We first find $\boldsymbol{x}^\star$ that minimizes the above objective by setting the gradient with respect to $\boldsymbol{x}$ to 0. We thus have,

$$\boldsymbol{x}^\star = (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\left(\frac{2\boldsymbol{A}^\top\boldsymbol{b} + \boldsymbol{C}^\top\boldsymbol{v}}{2}\right).$$

Using this value of $\boldsymbol{x}$ we arrive at the following dual program.

$$L(\boldsymbol{v}) = \max_{\boldsymbol{v}} \quad -\frac{1}{4}\boldsymbol{v}^\top\boldsymbol{C}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{C}^\top\boldsymbol{v} - \boldsymbol{b}^\top\boldsymbol{A}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{b} - \boldsymbol{v}^\top\boldsymbol{C}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{b} + \boldsymbol{b}^\top\boldsymbol{b} + \boldsymbol{v}^\top\boldsymbol{d},$$

which is optimized at,

$$\boldsymbol{v}^\star = 2\left(\boldsymbol{C}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{C}^\top\right)^{-1}\left(\boldsymbol{d} - \boldsymbol{C}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{b}\right).$$

Strong duality also implies that $L(\boldsymbol{x}, \boldsymbol{v}^\star)$ is optimized at $\boldsymbol{x}^\star$, which gives us,

$$\boldsymbol{x}^\star = (\boldsymbol{A}^\top\boldsymbol{A})^{-1}\left(\boldsymbol{A}^\top\boldsymbol{b} + \boldsymbol{C}^\top\left(\boldsymbol{C}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{C}^\top\right)^{-1}\left(\boldsymbol{d} - \boldsymbol{C}(\boldsymbol{A}^\top\boldsymbol{A})^{-1}\boldsymbol{A}^\top\boldsymbol{b}\right)\right).$$

C.2 Solving (4)

At every iteration of the algorithm, we want to solve the following problem,

$$\min_{\Delta} \quad \Delta^\top \boldsymbol{A}^\top (\boldsymbol{R} + s\boldsymbol{I})\boldsymbol{A}\Delta$$

$$\boldsymbol{g}^\top \boldsymbol{A}\Delta = i/2$$

$$\boldsymbol{C}\Delta = 0.$$

The constraints can be combined and rewritten as, $\boldsymbol{C}'\Delta = \boldsymbol{d}'$ where,

$$\boldsymbol{C}' = \begin{bmatrix} \boldsymbol{C} \\ \boldsymbol{g}^\top \boldsymbol{A} \end{bmatrix}, \quad \boldsymbol{d}' = \begin{bmatrix} 0 \\ i/2 \end{bmatrix}.$$

Let $\boldsymbol{R}' = \boldsymbol{R} + s\boldsymbol{I}$. We now want to solve,

$$\min_{\Delta} \quad \|\boldsymbol{R}'^{1/2}\boldsymbol{A}\Delta\|_2^2$$

$$\boldsymbol{C}'\Delta = \boldsymbol{d}'.$$

Using a procedure similar as in the previous section, we get,

$$\Delta^\star = (\boldsymbol{A}^\top \boldsymbol{R}'\boldsymbol{A})^{-1}\boldsymbol{C}'^\top \left(\boldsymbol{C}'(\boldsymbol{A}^\top \boldsymbol{R}'\boldsymbol{A})^{-1}\boldsymbol{C}'^\top\right)^{-1}\boldsymbol{d}'$$