A Multilevel Acceleration for l_1-regularized Logistic Regression

Javier S. Turek
Intel Labs
2111 NE 25th Ave.
Hillsboro, OR 97124
javier.turek@intel.com

Eran Treister
Earth and Ocean Sciences
University of British Columbia
Vancouver, BC, V6T 1Z2, Canada
eran@cs.technion.ac.il

Abstract

Logistic Regression is a well studied classification technique. In some cases, its l_1-regularized version is used to reduce overfitting to the data and obtain a sparse model. Because this problem is non-differentiable, it requires specialized methods to solve it, and indeed several iterative methods are available in the literature. However, these methods typically introduce many non-zero variables in the model during their first iterations, requiring an extensive number of operations. We develop a multilevel approach to prevent this behavior and accelerate the existing methods. For that we exploit the sparseness of the model and define a hierarchy of reduced problems, which are treated in turn to accelerate the optimization process. Numerical results show an improvement of up to five times in the performance of the accelerated methods compared to the original ones.

1 Introduction

Logistic Regression is a popular classification method in the machine learning literature. Recently, a l_1-regularized version of the Logistic Regression problem was introduced to obtain a sparse model, and was shown to be less prone to overfitting [1]. Given a set of samples $\{\mathbf{x}_i\}_{i=1}^l \in \mathbb{R}^n$ and their respective labels $\{y_i\}_{i=1}^l \in \{-1, +1\}$, the l_1-regularized Logistic Regression classifier is obtained by solving the following optimization problem[1]:

$$\min_{\mathbf{w}\in\mathbb{R}^n} L(\mathbf{w}) + \|\mathbf{w}\|_1 = \min_{\mathbf{w}\in\mathbb{R}^n} C\sum_{i=1}^{l} \log\left(1 + e^{-y_i \mathbf{x}_i^T \mathbf{w}}\right) + \|\mathbf{w}\|_1, \tag{1}$$

where C is a regularization parameter that balances between the sparsity of the model and the loss function $L(\mathbf{w})$. Typically, the l_1-norm regularization term $\|\mathbf{w}\|_1$ prevents model overfitting and leads to a sparse solution $\mathbf{w}$. However, this term is not differentiable and the optimization problem in Equation (1) is non-smooth and convex, and is often treated by specialized iterative methods. Many such methods are available in the literature – some of these are found in [2, 3, 4, 5, 6, 7] and references therein.

In this work, we accelerate the existing iterative solvers for (1) by applying a multilevel approach. This approach was successfully implemented in other domains for accelerating sparse optimization problems such as LASSO solvers [8] and Sparse Inverse Covariance estimation methods [9]. To accelerate the convergence of existing methods, this approach defines a hierarchy of smaller versions of the problem (1). Each of these subproblems is defined by restricting the original problem to

[1] A bias term b can be added to the loss function. Therefore, $\mathbf{x}_i^T\mathbf{w}$ is replaced with $\mathbf{x}_i^T\mathbf{w} + b$ in the loss function and the optimization is held over $\mathbf{w}$ and b.

a subset of the variables, limiting the solution $\mathbf{w}$ to a lower dimensional subspace. The subset of variables is chosen such that the ostensibly irrelevant variables are assigned with a zero value and essentially ignored. Starting from the smallest subproblem, the approach applies iterations of existing methods and transfer the solution to the next subproblem in the hierarchy. These iterations, which are applied on a sequence of gradually higher dimensional problems, aim to activate variables that are part of the minimizer of (1).

2 Iterative Proximal Newton Methods for l_1-regularized Logistic Regression

Current state-of-the-art methods [2, 3, 4] apply a second order approximation to (1) to obtain a Newton descent direction. In these methods, only the smooth loss function $L(\mathbf{w})$ is replaced with its second order Taylor approximation, leaving the non-smooth l_1 term intact. This approach is known as "proximal Newton" [10]. For this purpose, the methods require to compute the gradient and the Hessian of $L(\mathbf{w})$:

$$\nabla L(\mathbf{w}) = C \sum_{i=1}^{l} \left(\tau(y_i \mathbf{x}_i^T \mathbf{w}) - 1\right) y_i \mathbf{x}_i, \qquad \nabla^2 L(\mathbf{w}) = CXDX^T, \tag{2}$$

where $\tau(s) = \frac{1}{1+e^{-s}}$ is the derivative of the logistic loss function $\log(1+e^{-s})$, $D \in \mathbb{R}^{l \times l}$ is a diagonal matrix with elements $D_{ii} = \tau(y_i \mathbf{x}_i^T \mathbf{w})\left(1 - \tau(y_i \mathbf{x}_i^T \mathbf{w})\right)$, and $X \in \mathbb{R}^{n \times l}$ is a matrix with all the data samples, i.e., $X = [\mathbf{x}_1, \ldots, \mathbf{x}_l]$. Given the gradient and the Hessian of a current iterate $\mathbf{w}^k$, a Newton direction $\mathbf{d}$ is obtained by solving

$$\min_{\mathbf{d}} L(\mathbf{w}^k) + \nabla L(\mathbf{w}^k)^T \mathbf{d} + \frac{1}{2} \mathbf{d}^T \nabla^2 L(\mathbf{w}^k)^T \mathbf{d} + \|\mathbf{w}^k + \mathbf{d}\|_1. \tag{3}$$

This derived optimization task in (3) is the widely known LASSO problem [11]. To ensure a reduction in the objective of (1), a line-search procedure may be used to find a step-size α that updates the solution in the descend direction $\mathbf{d}$, i.e., $\mathbf{w}^{k+1} \leftarrow \mathbf{w}^k + \alpha \mathbf{d}$.

The GLMNET [3] and its improved version, the newGLMNET [4], are methods that implement the proximal Newton approach described above. To obtain the the Newton direction, the newGLMNET method solves (3) using a Coordinate Descent (CD) method. Each CD iteration over all variables typically requires $\mathcal{O}(nl)$ operations for a dense data matrix X, and $\mathcal{O}(nnz)$ for a sparse data matrix with nnz non-zeros. On the other hand, solving the line search procedure requires $\mathcal{O}(n+l)$ operations. The CDN algorithm [2] is a similar approach that solves (1) using a Coordinate Descent approach. To update each coordinate, CDN solves a one-dimensional proximal Newton problem with a one-dimensional line-search procedure. This method also requires $\mathcal{O}(nl)$ or $\mathcal{O}(nnz)$ operations for a dense or sparse data matrix X, respectively. Although CDN and newGLMNET have the same complexity, CDN usually requires more iterations to converge. In addition, newGLMNET applies less line-search computations, requiring less expensive exponential and logarithm computations.

2.1 Active Set Methods

One way to reduce the number of computations of iterative methods for sparse optimization problems is to temporarily remove variables that may not participate in the solution [12]. This technique restricts the minimization at each Newton iteration to a subset of variables while keeping the remaining entries to zero. The set of variables that remain fixed at zero is called an active set, while the set of remaining variables in the minimization is denoted as a "free set" and is defined as

$$\mathcal{F}(\mathbf{w}) = \{j : \mathbf{w}_j \neq 0 \vee |\nabla_j L(\mathbf{w})| \geq 1\}. \tag{4}$$

Solving (3) only for the elements outside $\mathcal{F}(\mathbf{w})$, leads to a zero solution $\mathbf{d} = \mathbf{0}$. Therefore, one can benefit from restricting the solution of problem (3) to the free set of the current iterate. Computing $\mathcal{F}(\mathbf{w}^k)$ introduces minimal overhead as it requires the computation of the gradient, which is already needed for finding a descent direction for (1). A generic proximal Newton method for solving (1), including a restriction to the free set, is presented in Algorithm 1.

Both CDN and newGLMNET shrink the number of variables in each iteration, hence reducing the number of computations. In CDN, the times that the line-search procedure is computed decreases to $\mathcal{O}(Ml)$, where $M = |\mathcal{F}(\mathbf{w})|$ is the cardinality of the free set. In newGLMNET, the LASSO problem is solved in $\mathcal{O}(Ml)$ (for a dense data matrix), reducing the number of computations for the Hessian, and the line-search is achieved in $\mathcal{O}(M+l)$. However, both methods still require $\mathcal{O}(nl)$ operations to compute the gradient (or $\mathcal{O}(nnz)$ for a sparse X).

Algorithm: $\mathbf{w}^{k+1} \leftarrow$ **ProximalNewtonL1LogReg**$(\mathbf{w}^k)$
Compute the gradient $\nabla L(\mathbf{w}^k)$, and define $\mathcal{F}(\mathbf{w}^k)$.
Calculate the Newton direction $\mathbf{d}$ by solving (3) restricted to $\mathcal{F}(\mathbf{w}^k)$.
Find a step-size α by line-search and define $\mathbf{w}^{k+1} = \mathbf{w}^k + \alpha\mathbf{d}$.

Algorithm 1: Proximal Newton iteration for the l_1-regularized Logistic Regression.

3 A Multilevel Acceleration for l_1-regularized Logistic Regression

In the previous section we described one type of iterative method for solving (1), and the active set heuristic that aims to reduce their number of computations. However, for an iterate $\mathbf{w}^k$ far from the minimizer $\mathbf{w}^*$, the entries of the gradient $\nabla L(\mathbf{w}^k)$ may be very large, yielding a large free set. In particular, this typically happens during the first steps of most methods, resulting in iterates that have significantly more non-zeros than the final sparse solution $\mathbf{w}^*$. When such a method advances through the iterations, many of the gradient entries decrease, reducing the size of the free set and the number of non-zeros in the solution until it converges to the set of the non-zero indices in $\mathbf{w}^*$. This same problem is evident also for other sparse optimization problems, and is treated by multilevel approaches in [8, 9]. In particular, [9] applies it to proximal Newton methods, which include the active set for the sparse inverse covariance estimation.

We now describe our multilevel approach to accelerate methods for solving (1). Our method avoids the explosion of non-zeros in the first iterates, and aims to reduce the number of iterations to converge. For this purpose, we define a hierarchy of levels with *coarse* problems, which are versions of (1) restricted to a subset of variables. In each multilevel iteration, this hierarchy is traversed starting from the coarsest level to the finest one, where all variables are considered (no restriction). On each level we apply one "relaxation", an iteration of a method like those described in the previous section, on the restricted subproblem. We refer to this whole process as a ML-cycle and it is repeated until convergence is achieved. The ML-cycle is summarized in Algorithm 2.

Algorithm: $\mathbf{w}^{k+1} \leftarrow$ **ML-cycle**$(\mathbf{w}^k, Relax(\cdot))$
% Relax(): an iterative method for (1).
Calculate $\nabla L(\mathbf{w}^k)$, and define the hierarchy $\{\mathcal{C}_p\}_{p=0}^{P}$ in (6).
Set $\hat{\mathbf{w}} \leftarrow \mathbf{w}^k$
For $p = P...0$
    Apply $\hat{\mathbf{w}} \leftarrow Relax(\hat{\mathbf{w}})$ iteration for problem (5) with $\mathcal{C}_p$.
Set $\mathbf{w}^{k+1} \leftarrow \hat{\mathbf{w}}$

Algorithm 2: ML-cycle for l_1-regularized Logistic Regression.

A restricted version of problem (1) in each level is obtained by constraining the non-zero entries in $\mathbf{w}$ to a subset $\mathcal{C}$ by

$$\min_{\mathbf{w},\ \mathrm{supp}(\mathbf{w})\subseteq\mathcal{C}} L(\mathbf{w}) + \|\mathbf{w}\|_1, \tag{5}$$

where $\mathrm{supp}(\mathbf{w})$ is the support of a vector $\mathbf{w}$ (i.e., the set of non-zero entries). For example, to solve the subproblem (5) with a method of the form of Algorithm 1, the variables can be restricted to $\mathcal{F}(\mathbf{w}^k) \cap \mathcal{C}$. We define the subsets in the hierarchy as

$$\mathrm{supp}(\mathbf{w}^k) \subseteq \mathcal{C}_P \subseteq \mathcal{C}_{P-1} \subseteq \cdots \subseteq \mathcal{C}_1 \subseteq \mathcal{C}_0 = \{1, ..., n\}. \tag{6}$$

The coarsest level, $\mathcal{C}_P$, is lower bounded by the support of the last iterate so $\mathbf{w}^k$ is a feasible point for the restricted problem (5). The finest level $\mathcal{C}_0$ introduces no restriction on the variables. Generally, the size of each level is defined as $|\mathcal{C}_{p+1}| = \lceil \frac{1}{2} |\mathcal{C}_p| \rceil$. However, whenever a free set is used, $\mathcal{C}_1$ may be defined to be of size $\lceil \mathcal{F}(\mathbf{w}^k) \rceil$ [9]. We adopt this choice in this work.

This way, assuming that the cost of a relaxation is proportional to the size of $\mathcal{C}$p, then the overall cost for a ML-cycle is similar to two relaxations of the complete problem. Starting from level P, each level p in a ML-cycle undergoes a relaxation that treats the restricted problem (5) with $\mathcal{C}_p$. Because the subsets are nested, the variables in the coarser levels are treated with more relaxations.

Table 1: performance results for the original methods and their multilevel accelerated versions.

Dataset	n	l	C	$\|\mathbf{w}^*\|_0$	nGLM	ML-nGLM	CDN	ML-CDN
news20	1355191	15997	64	2792	3.52 (13)	**1.87 (7)**	16.31(182)	7.57 (9)
rcv1	47236	541920	4	10893	37.89 (13)	**36.43 (14)**	167.54 (86)	90.72 (19)
webspam	16609143	280000	64	7914	122.2 (8)	**87.87 (1)**	2228.4 (51)	532.0 (1)
epsilon	2000	400000	0.5	1106	196.0 (13)	**162.0 (13)**	2933.6 (139)	1501.1 (35)
gisette	5000	6000	0.25	554	1.44 (10)	**0.92 (4)**	19.89 (91)	3.76 (7)

Therefore, we choose for the intermediate levels the variables with the largest gradient magnitude $\left|\nabla_j L(\mathbf{w}^k)\right|$ as they can reduce the objective function in (1) the most.

4 Numerical Results

In this section we compare between the performances of state-of-the-art methods and their accelerated versions. We consider the CDN [2] and the newGLMNET [4] methods, and their multilevel accelerations: ML-CDN and ML-newGLMNET . The stopping criterion of all methods, suggested by [4], is $\|\nabla^S L(\mathbf{w}^k)\|_1 \leq \varepsilon \frac{\min(\#pos,\#neg)}{l} \|\nabla^S L(\mathbf{w}^1)\|_1$, where $\#pos$ and $\#neg$ are the number of positive and negative labels in the samples, and $\nabla^S L(\mathbf{w})$ is the minimum norm subgradient. All the methods were implemented in C++ based on the implementation in LIBLINEAR [13]. All the experiments were executed on a machine with 2 Intel Xeon[2] E5-2699V3 2.30GHz processors with 36 cores, 128GB RAM, and Linux Cent-OS.

We use the data sets news20, gisette, webspam, rcv1, and epsilon with values for the regularizer C as reported in [4]. The ε value for news20 dataset is 1e-4, and for the other data sets it is 1e-3. For ML-newGLMNET, the number of coordinate descend iterations for the finest and mid-levels are selected small (one and two respectively) because those relaxations are used for inclusion or correction of non-zeros in the support. On the coarsest level, a relatively small problem is considered and there we allow more iterations (up to five). This level aims to determine the values of the non-zeros in the support [8].

Performance results are presented in Table1. For each method we present the timing results in seconds and the number of iterations in parenthesis. The number of iterations for newGLMNET is for each proximal Newton update, for CDN accounts for updating all the variables (n one-dimensional proximal Newton problems), and for the accelerated methods accounts for the number of ML-cycles. To save space in Table 1, newGLMNET and ML-newGLMNET are denoted by nGLM and ML-nGLM, respectively.

The multilevel approach shows the best performance and runtime improvement for both methods in almost all cases. In some cases ML-CDN achieves a runtime reduction of factor 4 or 5 compared to CDN. This improvement comes from saving several iterations until it achieves a support size of about the support size of the true solution. This fact is reflected in the number of iterations of ML-CDN comparing to those of CDN. The reductions in runtime for ML-newGLMNET are more limited, as newGLMNET usually converges in a few iterations. In particular, in the dataset rcv1, the support of the solution concentrates 88% of the non-zeros in the matrix X, and the multilevel acceleration is unable to save computations. Still, the number of iterations is reduced or is similar, while the runtime decreases by up to 45% in the best case.

5 Conclusions

In this work, we develop a multilevel approach for accelerating iterative solvers for the l_1-regularized Logistic Regression problem. We define a hierarchy of restricted problems using a handful of subsets of the problem variables. We solve the problem traversing the hierarchy and gradually building the solution, while avoiding the explosion of non-zeros. Numerical results demonstrate the performance improvement of our acceleration over the standard versions of existing methods.

[2]Intel and Xeon are trademarks of Intel Corporation in the U.S. and/or other countries.

References

[1] Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 78–, New York, NY, USA, 2004. ACM.

[2] Guo-Xun Yuan, Kai-Wei Chang, Cho-Jui Hsieh, and Chih-Jen Lin. A comparison of optimization methods and software for large-scale l1-regularized linear classification. *Journal of Machine Learning Research*, 11:3183–3234, 2010.

[3] Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010.

[4] Guo-Xun Yuan, Chia-Hua Ho, and Chih-Jen Lin. An improved glmnet for l1-regularized logistic regression. *Journal of Machine Learning Research*, 13:1999–2030, 2012.

[5] Kwangmoo Koh, Seung-Jean Kim, and Stephen Boyd. An interior-point method for large-scale l1-regularized logistic regression. *J. Mach. Learn. Res.*, 8:1519–1555, December 2007.

[6] Alexander Genkin, David D Lewis, and David Madigan. Large-scale bayesian logistic regression for text categorization. *Technometrics*, 49(3):291–304, 2007.

[7] Shai Shalev-Shwartz and Ambuj Tewari. Stochastic methods for l1 regularized loss minimization. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 929–936, New York, NY, USA, 2009. ACM.

[8] E. Treister and I. Yavneh. A multilevel iterated-shrinkage approach to l_1 penalized least-squares minimization. *Signal Processing, IEEE Transactions on*, 60(12):6319–6329, 2012.

[9] E. Treister, J.S. Turek, and I. Yavneh. A multilevel framework for sparse optimization. *Submitted to SIAM Scientific Computing*, 2015.

[10] Paul Tseng and Sangwoon Yun. A coordinate gradient descent method for nonsmooth separable minimization. *Mathematical Programming*, 117(1-2):387–423, 2009.

[11] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[12] Simon Perkins, Kevin Lacker, and James Theiler. Grafting: Fast, incremental feature selection by gradient descent in function space. *J. Mach. Learn. Res.*, 3:1333–1356, March 2003.

[13] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.