Sparse Least-Squares Methods in the Parallel Machine Learning (PML) Framework

Ramesh Natarajan
IBM T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY, 10598.
Email: nramesh@us.ibm.com

Vikas Sindhwani
IBM T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY, 10598.
Email: vsindhw@us.ibm.com

Shirish Tatikonda
Department of Computer Science
2015 Neil Avenue
Columbus, OH 43210-1277.
Email: tatikond@cse.ohio-state.edu

***Abstract*—We describe parallel methods for solving large-scale, high-dimensional, sparse least-squares problems that arise in machine learning applications such as document classification. The basic idea is to solve a two-class response problem using a fast regression technique based on minimizing a loss function, which consists of an empirical squared-error term, and one or more regularization terms. We consider the use of Lanczos-based methods for solving these regularized least-squares problems, with the parallel implementation in the Parallel Machine Learning (PML) framework, and performance results on the IBM Blue Gene/P parallel computer.**

***Keywords*-sparse regression; classification; parallel machine learning;**

I. Introduction

This paper considers the parallelization of regularized, sparse, least-squares problems for machine learning applications. Although this topic has been widely studied in the inverse problems and numerical analysis literature, and there are several software implementations of the relevant parallel algorithms, nevertheless, our work has a different focus with two specific objectives, as described below.

The first objective is the implementation of sparse least-squares methods using the Parallel Machine Learning (PML) software framework [27]. This framework provides an abstract parallel computational model with a supporting software API, as described in [26], and any algorithms implemented with this API do not require recoding or reimplementation when targeted to any of the parallel platforms on which PML is supported, which include platforms as disparate as commercial parallel databases and distributed-memory HPC platforms (with support for other parallel platforms also under active consideration). Furthermore, algorithms developers using the PML API do not have to be concerned with the parallel programming details on any of these platforms, which are abstracted away in the software API (the required platform-specific programming details are confined to the control layer in the PML framework, which is the only component in the framework that needs to be reimplemented for each new target platform, with the code for the individual algorithms, as mentioned above, being unchanged). A final advantage of the PML framework is that it provides an efficient, common implementation of the services that are required by all machine learning algorithms, such as data I/O with multiple formats, parallel data partitioning, and parallel task control. However, although many algorithms for clustering, classification and regression have been implemented in PML, the experience with algorithms for sparse training data sets has been somewhat limited. Therefore, the current work addresses this lacuna, and identifies the relevant implementation and performance issues for this class of problems in the PML framework.

The second objective, which is more exploratory in nature, is to consider the parallelization of sparse regression algorithms from a broader perspective which includes the examinaton of various evaluation criteria and parallelization strategies for estimating the optimal value for the regularization parameters in machine learning applications. For example, a reasonable strategy in this context is to solve the regression problem for several values of the regularization parameter, if possible in parallel, and select the optimal value from this set based on some evaluation criterion. However, depending on the form of the regularization term, the sparsity of the data matrix, and the number of values of the regularization parameter being examined, and the form of the evaluation criterion term, it is typically more efficient to explore the regularization parameter space without parallelizing the computations along this particular dimension (e.g., [9], [14]).

With these two objectives in mind, we consider the regression problem for a response variable $y \in \mathbb{R}$ with explanatory covariates $\boldsymbol{x} \in \mathbb{R}^M$, using a training data set of N labelled data points $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$. The regression function is assumed to be linear in the covariates, $f(\boldsymbol{x}; \boldsymbol{w}) = \boldsymbol{w}^T \boldsymbol{x}$, with parameters $\boldsymbol{w} \in \mathbb{R}^M$ (the intercept terms in this linear model are absorbed into a constant-valued feature in the covariate vector $\boldsymbol{x}$). In the specific context of machine learning applications such as document classification, we are often interested in the two-class regression problem, for which the response y_i is binary valued in the training data. We take $y_i \in \{+1, -1\}$ without loss of generality, corresponding to whether or not a given document with feature vector $\boldsymbol{x}_i$, belongs to the specified topic category. The regression function $f(\boldsymbol{x}; \boldsymbol{w})$ can be used a decision rule, $y = \text{sign}(\boldsymbol{w}^T \boldsymbol{x})$ for classifying unlabelled "out-of-sample" documents with feature vectors $\boldsymbol{x}$, and in some applications the magnitude of $f(\boldsymbol{x}; \boldsymbol{w})$ is also used as a score for ranking the relative probability of a document having the

relevant class membership.

A major difficulty in obtaining a fast and scalable regression methodology in these applications, is the large values for N and M in the relevant training data sets. For example, M may be $O(10^5)$ or even larger (e.g., consider a document feature vector $\boldsymbol{x}$ that encodes the relative frequency of occurence of the individual words or word-phrases in a domain dictionary). Similarly, N may be $O(10^6)$ or larger for many applications (e.g., consider the number of patents pending at the U.S. Patent Office as a training corpus of documents). The regression methods for such training data sets must take advantage of the sparsity in these training data sets for computational efficiency (e.g., many documents have only a small percentage of non-zero entries in their feature vector representation). Large-scale, sparse regression problems with variations on this theme also arise in several areas besides document classification, such as bioinformatics, sensor applications, image recognition, collaborative filtering, speech recognition and machine-aided translation.

The outline of this paper is as follows. Section II considers the loss functions that are used for the sparse regression problem, and which we anticipate supporting in the PML framework. Section III considers the specific case of the squared error loss function, and suitable regression methods are described for the parameter estimation with large, sparse data sets. Section V considers the iterative stopping conditions and model selection criteria for the methods in Sections IV and III respectively. Section VI describes some parallel implementation results on the IBM Blue Gene/P computer, and Section VII contains the summary discussion.

II. Mathematical Motivation

The regression parameters $\boldsymbol{w} \in \mathbb{R}^M$ in the linear model $f(\boldsymbol{x};\boldsymbol{w}) = \boldsymbol{w}^T\boldsymbol{x}$ are estimated by minimizing the the regularized empirical loss function over the training data,

$$argmin_{\boldsymbol{w}} \left[\frac{1}{N} \sum_{i=1}^{N} l(y_i, f(\boldsymbol{x}_i;\boldsymbol{w})) + R(\boldsymbol{w};\mu) \right]. \qquad (1)$$

For the loss function term $l(y,f)$ in (1), there are many suitable choices in the two-class regression problem, such as, $|y-f|$ (absolute error), $1/2(y-f)^2$ (squared error), $\max(0,1-yf)$ (SVM), $\max(0,1-yf)^2$ (L2-SVM), $\log(1+\exp(-yf))$ (logistic), $\gamma^{-1}\log(\gamma(1-yf))$ (modified logistic, $\gamma << 1$). The relative merits of these choices, from a computational efficiciency and model quality viewpoint, are discussed in [30].

For the regularization term $R(\boldsymbol{w};\mu)$ in (1), a common choice is Tikhonov regularization based on the l_2 norm of the parameter vector, $1/2\mu||\boldsymbol{w}||_2^2$, and this choice leads to "shrinkage" of the terms in the estimated parameter vector $\boldsymbol{w}$ (if a constant-valued feature is present, its coefficient may often be excluded from this regularization term). Another choice is the l_1 norm regularization $\mu||\boldsymbol{w}||_1$, which is widely used for obtaining sparse feature selection, since many of the terms in the estimated parameter vector are forced to zero. However, note that in that context, the "sparsity" refers to the terms in the estimated parameter vector $\boldsymbol{w}$, whereas in this paper it also refers to the large number of zeros in the data vectors $\boldsymbol{x}_i$ as well. In addition to parameter shrinkage and sparsity, the regularization term can also be used to impose other smoothness constraints over the feature space. For example, consider $1/2\mu(\boldsymbol{w}^T\boldsymbol{L}\boldsymbol{w})$ where $\boldsymbol{L}$ is a $M \times M$ symmetric, positive-semidefinite matrix (so that $R(\boldsymbol{w})$ is a semi-norm regularization), in which the non-zero, off-diagonal terms $\boldsymbol{L}(i,j)$ are negative values whose magnitude encodes the affinity weights between the features i and j, and the diagonal terms $\boldsymbol{L}(i,i) = -\sum_{i \neq j} \boldsymbol{L}(i,j)$ are the sign-reversed, row-sum of the off-diagonal terms. In document classification, one possible choice for $\boldsymbol{L}$ is the sparse graph Laplacian on the nearest-neighbor adjacency graph of the M features, with edge weights proportional to the feature similarity measures, and this regularization term has the effect of shrinking the coefficient parameters of strongly-coupled features to common values (the strongly-coupled features are close to collinear in the semi-norm induced by the regularization term). Another choice is to contruct the graph Laplacian $\boldsymbol{L}_e$ on the nearest-neighbor adjacency graph of the N training data examples, and then set $\boldsymbol{L} = \boldsymbol{X}^T\boldsymbol{L}_e\boldsymbol{X}$, which has the effect of smoothing the regression response so that two data points that are strongly coupled on this affinity graph, will also tend to have similar regression responses. The affinity graphs that are used in this regularization, may be based on extrinsic considerations that are not part of the training data matrix, and for example, one possible affinity-graph structure on the data examples can be generated by the presence or absence of hyperlinks or citations between any two documents, and this affinity measure is of considerable interest in internet-based applications [29]. The construction of such a graph may make use of a large amount of unlabeled data, which is in general more abundantly available when compared to labeled examples. The use of such graph-Laplacian based regularization methods have been widely studied in transductive graph methods [18], and more generally, in manifold regularization [1].

In many applications, multiple regularization terms may be additively combined in (1), but this also entails the need to estimate optimal values for each of these multiple regularization parameters, which further increases the computational difficulty, as discussed below in Section V.

For many combinations of the loss functions and regularization terms, the optimization problem (1) is nonlinear and/or non-differentiable, and specialized techniques have been developed for individual cases (e.g., SVM [17], L2-SVM [5], [19], logistic [21], and modified logistic [30], [28]). Some of these techniques are specifically designed for large, high-dimensional data sets, and only require holding only small sections of the large data matrices in memory (e.g., the coordinate descent methods described in [30], [28], [5], which iteratively improve the solution by cycling over the individual features in the data set). For computational tractability, we have found that in many of these methods, the regularization parameter is either fixed at some reasonable value (e.g., $\mu = 1$), or a few values of μ are examined using a hold-out

or cross-validation loss criterion, so that there is considerable scope for using parallel computation to improve the estimates for the optimal regularization parameters.

The case of the squared-error loss with Tikhonov regularization has been very widely studied, and it is well known that the optimization problem (1) in this case has an explicit solution in terms of the generalized inverse of the normal equations [10], [3]. However, for large N and M, the direct application of this approach is prohibitively expensive, and further, does not take advantage of the sparsity of the data matrix.

Regularized least-squares problems have also been widely studied in the inverse problems literature (for a review, see [12]). However, many of the applications there arise from the discretization of integral or partial differential equations, which typically lead to square data matrices that are either dense, or if sparse, have some definite sparsity structure. In contrast, machine learning applications often lead to rectangular data matrices with a general sparsity structure. Nevertheless, many of the techniques used in the inverse problems literature for obtaining the optimal regularization parameter, can also be used for machine learning applications, as discused further in Section V.

III. Sparse Least Squares Problems

For the squared-error loss with Tikhonov regularization, (1) becomes

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} ||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||_2^2 + \mu||\boldsymbol{w}||_2^2, \tag{2}$$

where $\boldsymbol{y} \in \mathbb{R}^N$ contains the response vector, and $\boldsymbol{X} \in \mathbb{R}^{N\times M}$ is the sparse data matrix whose rows contain the corresponding feature vectors. The solution for (2) is equivalent to

$$\boldsymbol{w}(\mu) = (\boldsymbol{X}^T\boldsymbol{X} + \mu\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}, \tag{3}$$

where $\boldsymbol{I}$ is the conforming $M \times M$ identity matrix, and the notation $\boldsymbol{w}(\mu)$ is used whenever it is necessary to make the dependence of the solution $\boldsymbol{w}$ on μ explicit. However, as mentioned earlier, it is impractical to obtain $\boldsymbol{w}$ from (3) for large M, as it requires the formation and inversion of a dense $M \times M$ matrix, even when the data matrix $\boldsymbol{X}$ is sparse. A suitable alternative is to solve for $\boldsymbol{w}$ using an iterative Krylov subspace method, where after denoting $\boldsymbol{b} = \boldsymbol{X}^T\boldsymbol{y}$, we have in the m'th step of this iteration, that $\boldsymbol{w}_m \in \mathcal{K}_m(\boldsymbol{X}^T\boldsymbol{X}+\mu\boldsymbol{I}, \boldsymbol{b})$, where $\mathcal{K}_m(\boldsymbol{X}^T\boldsymbol{X} + \mu\boldsymbol{I}, \boldsymbol{b})$ denotes the Krylov subspace of order m, is equivalent to

$$\text{Span}\{\boldsymbol{b}, (\boldsymbol{X}^T\boldsymbol{X} + \mu\boldsymbol{I})\boldsymbol{b}, \ldots, (\boldsymbol{X}^T\boldsymbol{X} + \mu\boldsymbol{I})^{m-1}\boldsymbol{b}\}. \tag{4}$$

In particular, this Krylov subspace is shift-invariant for all values of μ for which the Krylov matrix $\boldsymbol{X}^T\boldsymbol{X} + \mu\boldsymbol{I}$ is positive-definite, so that the approximation $\boldsymbol{w}_m(\mu)$ in the m'th step for these values of μ can also be obtained in the same subspace.

Thus, for example, an orthogonal basis for $\mathcal{K}_m(\boldsymbol{X}^T\boldsymbol{X}, \boldsymbol{b})$ can be generated by the well-known Lanczos bidiagonalization procedure [10], [3]. We denote the Lanczos vectors by $\boldsymbol{u}_k, \boldsymbol{v}_k$ for $k = 1, 2, \ldots$, so that

$$\boldsymbol{X}\boldsymbol{V}_k = \boldsymbol{U}_{k+1}\boldsymbol{B}_k \tag{5}$$

$$\boldsymbol{X}^T\boldsymbol{U}_{k+1} = \boldsymbol{V}_k\boldsymbol{B}_k^T + \alpha_{k+1}\boldsymbol{v}_{k+1}\boldsymbol{e}_{k+1}^T \tag{6}$$

where $\boldsymbol{V}_k = [\boldsymbol{v}_1, \ldots, \boldsymbol{v}_k]$ and $\boldsymbol{U}_{k+1} = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_{k+1}]$ are orthogonal matrices with $\boldsymbol{V}_k^T\boldsymbol{V}_k = I_k$ and $\boldsymbol{U}_{k+1}^T\boldsymbol{U}_{k+1} = I_{(k+1)}$, and $\boldsymbol{B}_k$ is a is a $(k+1) \times k$ bidiagonal matrix matrix given by

$$\boldsymbol{B}_k = \begin{bmatrix} \alpha_1 & & & \\ \beta_1 & \ddots & & \\ & \ddots & \ddots & \\ & & \beta_{k-1} & \alpha_k \\ & & & \beta_k \end{bmatrix}, \tag{7}$$

With the starting vectors $\boldsymbol{u}_1 = \boldsymbol{y}/||\boldsymbol{y}||_2$ and $\boldsymbol{v}_1 = \boldsymbol{X}^T\boldsymbol{u}_1/||\boldsymbol{X}^T\boldsymbol{u}_1||_2$, we have for $j \geq 2$,

$$\beta_{j-1}\boldsymbol{u}_j = \boldsymbol{X}\boldsymbol{v}_{j-1} - \alpha_{j-1}\boldsymbol{u}_{j-1}, \quad \alpha_j\boldsymbol{v}_j = \boldsymbol{X}^T\boldsymbol{u}_{j-1} - \beta_{j-1}\boldsymbol{v}_{j-1}, \tag{8}$$

with $\beta_{j-1} = \boldsymbol{u}_j^T\boldsymbol{X}\boldsymbol{v}_{j-1}$ and $\alpha_j = \boldsymbol{u}_j^T\boldsymbol{X}\boldsymbol{v}_j$.

From (8), $\boldsymbol{v}_j$ are the Lanczos vectors for the iteration matrix $\boldsymbol{X}^T\boldsymbol{X}$, since

$$\boldsymbol{X}^T\boldsymbol{X}\boldsymbol{V}_k = \boldsymbol{V}_k\boldsymbol{B}_k^T\boldsymbol{B}_k + \alpha_{k+1}\beta_k\boldsymbol{v}_{k+1}\boldsymbol{e}_k^T, \tag{9}$$

and denoting the symmetric tridiagonal matrix $\boldsymbol{T}_k = \boldsymbol{B}_k^T\boldsymbol{B}_k$, so that

$$\boldsymbol{T}_k = \text{Tridiag}\{\alpha_j\beta_{j-1}, \alpha_j^2 + \beta_j^2, \alpha_{j+1}\beta_j\}, \tag{10}$$

then the Lanczos vectors $\boldsymbol{v}_k$ are identical for the iteration matrix $\boldsymbol{X}^T\boldsymbol{X} + \mu\boldsymbol{I}$, for $\mu > 0$, with $\boldsymbol{T}_k$ being replaced by $\boldsymbol{T}_k + \mu\boldsymbol{I}$.

Similarly, $\boldsymbol{u}_j$ are the Lanczos vectors for the iteration matrix $\boldsymbol{X}\boldsymbol{X}^T$, since

$$\begin{aligned}(\boldsymbol{X}\boldsymbol{X}^T)\boldsymbol{U}_{k+1} =& \boldsymbol{U}_{k+1}(\boldsymbol{B}_k\boldsymbol{B}_k^T + \alpha_{k+1}^2\boldsymbol{e}_k\boldsymbol{e}_k^T) \\ &+ \alpha_{k+1}\beta_{k+1}\boldsymbol{u}_{k+2}\boldsymbol{e}_k^T,\end{aligned} \tag{11}$$

and denoting the symmetric tridiagonal matrix $\tilde{\boldsymbol{T}}_k = \boldsymbol{B}_k^T\boldsymbol{B}_k + \alpha_{k+1}^2\boldsymbol{e}_k\boldsymbol{e}_k^T$, we have

$$\tilde{\boldsymbol{T}}_k = \text{Tridiag}\{\alpha_j\beta_j, \alpha_j^2 + \beta_{j-1}^2, \alpha_{j+1}\beta_{j+1}\}. \tag{12}$$

From (9), the approximation $\boldsymbol{w}_k \in \mathcal{K}_m(\boldsymbol{X}^T\boldsymbol{X}, \boldsymbol{X}^T\boldsymbol{y})$ to (2) satisfies the equation

$$\boldsymbol{w}_k = \alpha_1||y||_2\boldsymbol{V}_k(\boldsymbol{B}_k^T\boldsymbol{B}_k)^{-1}\boldsymbol{e}_1. \tag{13}$$

The tridiagonal matrices $\boldsymbol{T}_k$ or $\tilde{\boldsymbol{T}}_k$ do not have to be explicitly constructed or inverted, and in fact, $\boldsymbol{w}_k$ (13) can be obtained using simple update formulas that are similar to those for generating the Lanczos vectors. Specific examples of these update formulas include the CGLS method, which is based on the implicit Choleski factorization of $\boldsymbol{T}_k$ and the LSQR method, which is based on the more stable implicit-QR factorization of $\boldsymbol{T}_k$ ([3], Chapter 7).

Furthermore, by virtue of the shift-invariance of the Krylov basis (4), the solution $\boldsymbol{w}_k(\mu)$ for multiple values can also be

obtained with a single Lanczos iteration (6), which is clearly preferable to solving (2) anew for each value of μ.

Our parallel implementation in PML is therefore based on the CGLS algorithm for the multi-shift case, as described in [9] and further evaluated in [24]. The most expensive operations algorithm, both in terms of the computational and I/O costs, are the matrix-vector multiplications involving $\boldsymbol{X}$ and $\boldsymbol{X}^T$ in each iteration of the Lanczos procedure, which are parallelized by partitioning the rows of $\boldsymbol{X}$ among the processors, and computing the required matrix-vector products in a distributed fashion, with the final result being collected in the designated master processor.

IV. Iterative Stopping Condition

The stopping criterion for iterative convergence in the Lanczos procedure requires monitoring the two terms that appear in (2), and for any fixed μ, each of these terms will tend to constant values, as the solution $\boldsymbol{w}_k(\mu)$ converges in any iterative procedure. Therefore, it is important to be able to estimate these two terms in a computationally-efficient way, and to distinguish between any intermediate plateauing of these quantities and their final converged values, during the iterative process.

In particular, for the square-error loss with Tikhonov regularization, the relevant terms $||\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}||_2^2$ and $||\boldsymbol{w}||_2^2$ appearing in (2), which correspond to the norm of the residual and solution vectors respectively, can be evaluated as part of the Lanczos iteration, and furthermore, monotonic upper and lower bounds for these quantities can also be estimated, and a suitable stopping criteria for iterative convergence, as mentioned above, can be achieved by monitoring the gap between the upper and lower bounds for the two terms in (2). This is in fact a special case of the more general problem of computing bounds for the estimates of matrix moments of the form $\boldsymbol{g}^T(\boldsymbol{X}^T\boldsymbol{X} + \mu\boldsymbol{I})^p\boldsymbol{g}$ for negative integer p and certain vectors $\boldsymbol{g}$, which can be obtained from a single Lanczos iteration (6) as described in [11], [4]. This approach, whose details we omit for brevity, not only provides a valuable computational adjunct for the iterative stopping criterion, but is also useful for evaluation the criterion used for obtaining the optimal regularization parameter as described below.

V. Optimal Tikhonov Parameter Estimation

The estimation of the optimal regularization parameter using an exhaustive search requires solving (1) for many values of μ, and selecting the optimum value from this set based on a suitable evaluation criterion, such as the hold-out or the cross-validation estimates of the loss function (2). This exhaustive-search approach is computationally expensive, particularly in the case when the search space involves multiple regularization parameters. The use of gradient-based optimization methods to guide the optimal parameter estimation has been considered in [2], [20], [6], [7], although this still requires solving (2) for many points in the joint parameter space and dealing with the possibly non-convex optimization geometry of the evaluation criterion. However, this suggests that parallelism can be used in order to independently solve (2) for several values of the regularization parameter (or several sets of values in the multiple parameter case), even possibly as part of a line-search optimization step in the gradient-based methods.

However, as mentioned above, in the case of the squared-error loss with Tikhonov regularization, the solutions of (2) for multiple values of μ can be obtained, with only a little extra work, from a single Lanczos iteration itself. Similarly, the estimates to well-known GCV or L-curve evaluation criteria for multiple values of μ can be obtained from a single Lanczos iteration, from which the optimal value of μ can be evaluated. For instance, the GCV criterion [25], [8], which must be optimized with respect to μ, is given by the ratio

$$\frac{\|(\boldsymbol{X}^T\boldsymbol{X} + \mu\boldsymbol{I})^{-1}\boldsymbol{X}^T\boldsymbol{y}\|_2}{trace(\boldsymbol{X}^T\boldsymbol{X} + \mu\boldsymbol{I})^{-1}}, \tag{14}$$

which is the closed-form expression for the leave-one-out error in regularized least squares models. For large-scale, sparse problems, the matrix inverse in the denominator is impractical to evaluate, and an approximation to this denominator is obtained by the use using stochastic trace estimators. Similarly, the L-curve criterion is based on the highest curvature point while plotting the solution norm $\|\boldsymbol{w}\|^2$ against the residual norm $\|\boldsymbol{y} - \boldsymbol{X}\boldsymbol{w}\|^2$ on a log-log scale [13]. We note this L-curve criterion requires the estimation of the two terms that appear in (2), as discussed previously for the iterative stopping condition above.

However, apart from the GCV and L-curve criteria, a cross-validation criterion may also be used to select the optimal value of the regularization parameter. In this case, the model computations for each independent cross-validation fold can be carried out simultaneously along with the main Lanczos iteration involving the full $\boldsymbol{X}$ data matrix. In the parallel implementation of the Lanczos-based iterative method, which uses a data-parallel row partitioning of the $\boldsymbol{X}$, the matrix-vector operations for each fold will involve various row subsets of the $\boldsymbol{X}$ matrix, which can all be evaluated in a single data scan of $\boldsymbol{X}$, so that the entire set of matrix-vector operations for training data and all cross-validation folds can be performed with minimum disk and memory access costs. This implementation, whose details are omitted for brevity, takes advantage of the generic support for cross-validation in the PML framework, which ensures that the data in the cross-validation folds, and hence the consistency of the optimal regularization parameter, is invariant with the number of parallel processing nodes used in the computations.

VI. Computational Results

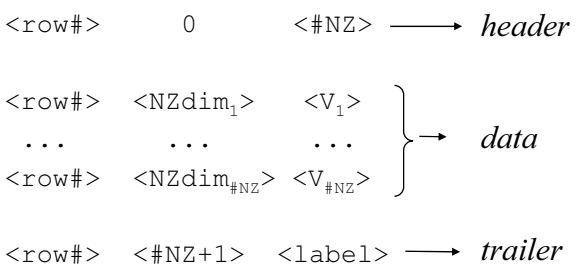
Table (I) describes the sparse datasets, which are merely chosen here to illustrate the performance analysis issues. The source references for the data sets can be obtained from the download site http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets.

Figure 1 shows the format for sparse input data set that is used by PML, along with a schematic row partitioning of the data set, that ensures that each processor gets roughly the same

number of non-zero entries to process, upto the nearest row boundary, for good load balance in the parallel computations.

	No. of Examples	*No. of Features*	*Percent Sparsity*
a7a	16100	123	11.27
rcv1-train	67399	47236	0.16

TABLE I
DATASETS



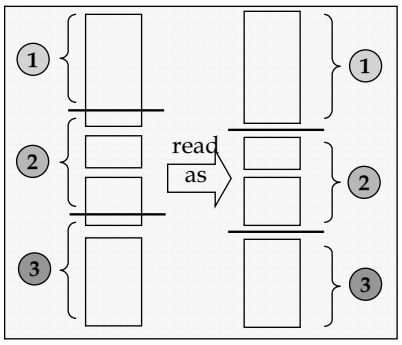


Fig. 1. Sparse data row format and row-partitioning schematic

Figure 2 shows the classification accuracy for regression models with rcv1-train in Table (I) evaluated on a hold-out data set rcv1-test (not shown in Table (I)). These results point to the importance of choosing the regularization parameter μ to obtain the best model accuracy (note that the rcv1-train and rcv1-test data sets from the original source have been interchanged, with the larger of the two being used as the training data set here for illustrative purposes).

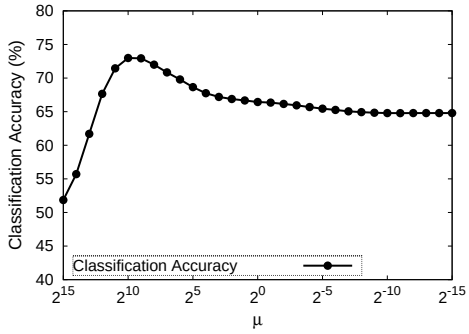


Fig. 2. Accuracy for the rcv1-train/rcv1-test data sets

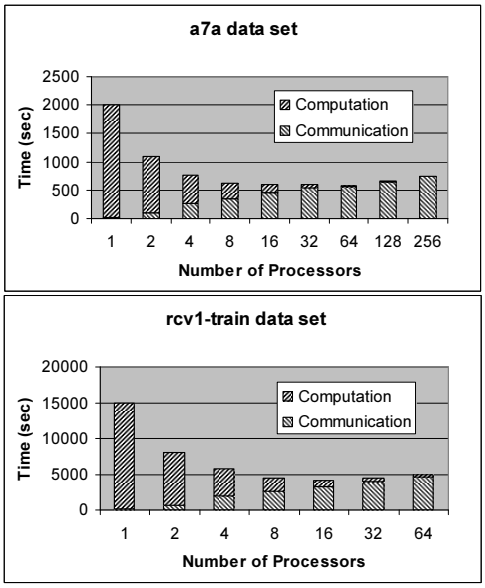


Fig. 3. Parallel performance for a7a and rcv1-train data sets

Figure 3 shows the parallel performance for these data sets on the IBM Blue Gene/P [16], and for clarity of exposition, these results were obtained by fixing the number of Lanczos iterations ($m = 50$), fixing the relative convergence criterion for the residual norm $\varepsilon = 10^{-3}$, and evaluating solutions for a fixed number of values of $\mu_1, \ldots, \mu_l$ ($l = 21$). Note that changing m and ε will change the overall program run time, but not the run time on a per-iteration basis. Similarly, any early-convergence of the iterations for a subset of the values of μ has little or no impact on the per-iteration program run time. Finally, increasing l changes the the amount of serial work in the computation, but again typically with negligible impact on the per-iteration program run time.

The major limitation in achieving a good parallel speedup is the overhead of the collective communication costs. Although these costs grow only as $O(\log P)$ for large P, the constant factor depends on the following two aspects. The first aspect is the size of the data being communicated, which on a per-iteration basis for the sparse matrix-vector operations is $O(\max(M, N))$ (in contrast, the computational granularity is $O(\rho NM/P)$, where ρ is the average row sparsity in the data matrix), and as a result, the high-dimensional case with large M, N is especially unfavorable for large P or small α. The second aspect is the implementation of the collective communications interface in PML, which is based on object serialization and materialization, and although this interface shields the algorithm developer from the details of the parallel programming, there is a small but extra overhead of memory allocation and memory copy operations in its implementation, which may be particularly noticeable in the case of sparse data sets which tend to have a lower ratio of computation to communication. We are actively pursuing performance-tuning optimizations to lower these communication costs, which will

improve the parallel performance closer to that for the "native" implementation, while retaining all the benefits of the PML approach.

The parallel speedup results in Figure 3 do not include the results for cross-validation computations, for which the details of the PML implementation and performance will be discussed elsewhere.

VII. Summary Remarks

Our preliminary benchmark results point to the usefulness of the PML framework for sparse matrix algorithms, although further work is required in terms of fine-tuning the performance and scalability. We note that the IBM Blue Gene/P implementation is easily capable of handling much larger data sets than those considered in this paper, but our goal here has been to consider the implementation and performance issues for sparse data sets in PML, rather than provide a specific hardware capability demonstration. In future work, we plan to further clarify the usefulness of these algorithms in the PML implementation by comparing the model quality and model training time with other state-of-the-art serial and parallel algorithms that have been used for machine learning applications.

The requirements for machine learning algorithms with sparse data sets, indicates the potential for exploiting the on-chip parallelism in multi-core processor architectures. For instance, in document classification applications, the training data set size is often constrained by the cost of manual labelling of the examples, and furthermore, regression models of acceptable quality may be obtained without using the largest possible training data sets. Therefore, particularly for highly-sparse data sets, the computation time, rather than the memory requirement, is the primary driver for the parallel implementation. The PML computational model, which in this paper was considered only for distributed-memory computers, can also be expected to provide a multi-threaded, multi-core implementation with good data locality and low synchronization overheads. In addition, the use of shared memory will considerably reduce the communication overheads that are a factor in the in the parallel speedup results described in this paper.

References

[1] M. Belkin, P. Niyogi and V. Sindhwani, *Manifold Regularization: A Geometric Framework for Learning from Labeled and Unlabeled Examples*, Journal of Machine Learning Research, Vol. 7, pp. 2399-2434, 2006.

[2] Y. Bengio, *Gradient-based optimization of hyperparameters*, Neural Computation, vol 12, pp. 1889-1900, 2000.

[3] A. Bjorck, Numerical Methods for Least Squares Problems, SIAM, Philadelphia, 1996.

[4] D. Calvetti, G. H. Golub, L. Reichel and Ax B, *Estimation of the L-Curve via Lanczos Bidiagonalization*, BIT, 39, pp. 603-619, 1997.

[5] K. W. Chang, C. J. Hsieh and C. J. Lin, *Coordinate Descent Method for Large-scale L2-loss Linear Support Vector Machines*, Journal of Machine Learning Research, Vol. 9, pp. 1369-1398, 2008.

[6] O. Chapelle, V. Vapnik, O. Bousquet and S. Mukherjee, *Machine Learning*, Vol. 46, pp. 131-159, 2002.

[7] C. B. Do, C. S. Foo and A. Y. Ng, *Efficient multiple hyperparameter learning for log-linear models*, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007. MIT Press 2008.

[8] L. Elden, *A note on the computation of the generalized cross-validation function for ill-conditioned least squares problems*, BIT, Vol. 24, pp. 467-472 (1984).

[9] A. Frommer and P. Maass, *Fast CG-based methods for Tikhonov-Philips regularizations*, SIAM J. Scientific Computing, Vol. 20, pp. 1831-1850 (1999)

[10] G. Golub and C. Van Loan, Matrix Computations, Johns Hopkins University Press, Baltimore MD, Third Edition, 1996.

[11] G. H. Golub and U. von Matt, *Tikhonov regularization for large scale problems*, in Workshop on Scientific Computing, (eds. G. H. Golub, S. H. Lui, F. Luk, R. J. Plemmons), pp. 3-26, Springer, New York, 1997.

[12] M. Hanke and P. C. *Hansen, Regularization Methods for Large-Scale Problems*, Surveys on Mathematics for Industry, Vol. 3, pp. 253-315 (1993).

[13] P. C. Hansen and D. P. O'Leary, *The use of the L-curve in the regularization of discrete ill-posed problems*, SIAM J. of Sci Computing, Vol. 14, 1487-1503, 1993.

[14] T. Hastie, S. Rosset, R. Tibshirani, J. Zhu and N. Christianini, *The Entire Regularization Path for the Support Vector Machine*, Journal of Machine Learning Research, Vol. 5, pp. 1391 - 1415, (2004).

[15] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A Berkeley view of cloud computing. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[16] IBM Blue Gene Team, *Overview of the IBM Blue Gene/P Project*, Special Issue on Applications of Massively Parallel Systems, IBM Journal of Research and Development, pp. 199-220, Vol. 52(1/2) (2008).

[17] T. Joachims, *Training linear SVMs in linear time*, Proceedings of the ACM Conference on Knowledge Discovery and Data Mining, 2006.

[18] R. Johnson and T. Zhang, *Graph-based Semi-supervised Learning and Spectral Kernel Design*, IEEE Transactions on Information Theory, Vol. 54(1), pp. 275-288, 2008

[19] S. S. Keerthi and D. DeCoste, *A modified Newton method for fast solution of large scale linear SVMs*, Journal of Machine Learning Research, Vol. 5, pp. 361-397, 2004.

[20] S. S. Keerthi, V. Sindhwani and O. Chapelle, *An Efficient Method for Gradient-based Adaptation of Hyperparameters in SVM Models*, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007. MIT Press 2008.

[21] C. J. Lin, R. C. Weng and S. S. Keerthi, *Trust region Newton method for large-scale logistic regression*, Journal of Machine Learning Research, vol. 9, pp. 627-650, 2008.

[22] MPICH2, http://www.mcs.anl.gov/research/projects/mpich2/.

[23] A. N. Tikhonov and V. B. Glasko, *Use of the regularization method for non-linear problems*, USSR Comput. Math. Math. Phys. vol. 5, pp. 93-107 (1965).

[24] J. van den Eshof and G. L. J. Sliejpen, *Accurate conjugate gradient methods for families of shifted systems*, Appl. Numer. Math, Vol 49, No. 1, pp. 17-37, 2004.

[25] G. Wahba, Spline Models for Observational Data, SIAM Philadelphia 1991.

[26] E. Yom-Tov, U. Aharoni, A. Ghoting, E. Pednault, D. Pelleg, H. Toledano, and R. Natarajan, *An Introduction to the IBM Parallel Mining Toolkit*, http://www.ibm.com/developerworks/grid/library/gr-ipmlt/index.html (2007).

[27] E. Yom-Tov, E. Pednault, R. Natarajan, D. Pelleg, H. Toledano, E. Aharoni, Y. Ben-haim, *Parallel Machine Learning Toolbox: User Guide*, http://www.alphaworks.ibm.com/tech/pml.

[28] T. Zhang and V. S. Iyengar, *Recommendation Systems Using Linear Classifiers*, Journal of Machine Learning Research, Vol. 2, No. 1, pp. 313-334, 2002.

[29] T. Zhang, A. Popescul and B. Dom, *Linear Prediction Models with Graph Regularization for Web-page Categorization*, Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadephia PA, pp. 821-826, 2006.

[30] T. Zhang and F. J. Oles, *Text Categorization based on regularized linear classification methods*, Information Retrieval, Vol. 4, No. 1, pp. 5-31, 2001.