Fast, Exact Model Selection and Permutation Testing for ℓ_2-Regularized Logistic Regression

Bryan Conroy
Columbia University
New York, NY
bc2468@columbia.edu

Paul Sajda
Columbia University
New York, NY
psajda@columbia.edu

Abstract

Regularized logistic regression is a standard classification method used in statistics and machine learning. Unlike regularized least squares problems such as ridge regression, the parameter estimates cannot be computed in closed-form and instead must be estimated using an iterative technique. This paper addresses the computational problem of regularized logistic regression that is commonly encountered in model selection and classifier statistical significance testing, in which a large number of related logistic regression problems must be solved for. Our proposed approach solves the problems simultaneously through an iterative technique, which also garners computational efficiencies by leveraging the redundancies across the related problems. We demonstrate analytically that our method provides a substantial complexity reduction, which is further validated by our results on real-world datasets.

1 Introduction

Regularized logistic regression is a standard classification technique for predicting a binary label from a set of features. It has been used successfully across a wide array of applications. Since it is capable of not only predicting the class from the data, but also the posterior class probabilities, it has particularly become popular in the medical and life sciences fields (Hosmer and Lemeshow 2000, and references therein).

While the parameter estimates in regularized least squares regression, such as ridge regression, can be computed in closed-form, logistic regression necessitates an iterative solver. The most popular technique is a variant of Newton's method, iteratively reweighted least squares (IRLS), which iteratively minimizes a quadratic approximation to the likelihood. This approach may be slow because it requires computing the inverse of a Hessian matrix. This step scales poorly with problem size unless the Hessian has special structure, such as being sparse or banded.

Many papers have addressed this computational problem. For example, cyclic coordinate descent (Zhang and Oles 2001, Genkin et al. 2007) avoids inverting the Hessian matrix by updating the estimates one coordinate at a time. The technique of Komarek and Moore (2003,2005) also avoids computing the inverse of the Hessian by solving each Newton step by a conjugate gradient method. Bound optimization techniques (Böhning and Lindsay 1988, Krishnapuram et al. 2005) complete a sequence of Newton steps with only a single Hessian matrix inversion.

Although these approaches have been very successful at improving the efficiency of logistic regression, real-world applications rarely consist of solving only a single problem. Instead, an entire machine learning application involves solving hundreds or even thousands of regression problems over the course of model selection and significance testing. For example, model selection strategies based on cross-validation estimate prediction accuracy by re-training the classifier on different splits of the data. Additionally, assessing the statistical significance of a chosen model involves running a nonparametric test that trains the classifier on different permutations of the data. These model fits are computed sequentially in a loop, or in parallel if a computing cluster is available.

For large problems, cross-validation and significance testing can be very expensive endeavors, even with efficient logistic regression solvers. In the context of cross-validation, this has sometimes been addressed by

Appearing in Proceedings of the 15^{th} International Conference on Artificial Intelligence and Statistics (AISTATS) 2012, La Palma, Canary Islands. Volume XX of JMLR: W&CP XX.

deriving upper bounds on generalization error that are fast to compute from the training data (Zhang 2003, Cawley and Talbot 2004). These bounds, however, can be loose and may greatly overestimate prediction error. For regularized least squares (ridge regression), it is well known that leave-one-out error can be computed exactly from one model fit to the training data (Rifkin and Lippert 2007). This special case, unfortunately, does not extend to logistic regression.

With these considerations in mind, this paper addresses the computational problems of logistic regression from a different perspective. Instead of learning each classifier's parameters *tabula rasa*, we solve the multitude of problems simultaneously, and leverage the shared structure to improve computational performance. Our method is simple to implement and is based on the well-established theory of stationary iterative methods for solving linear systems of equations. It can be applied to a large number of applications, including K-fold cross-validation and classifier significance testing through permutation tests.

In Section 2 we introduce the necessary preliminaries and outline the details of our approach, along with a complexity analysis that highlights its merits. We then offer a summary of applications of our method in Section 3. Finally, we validate our approach on real-world datasets in Section 4 and conclude with potentials for future work in Section 5.

2 Methods

2.1 Preliminaries

Let a classification dataset be given by $\{(x_i, y_i)\}_{i=1}^N$, with features $x_i \in \mathbb{R}^D$ and binary labels $y_i \in \{0, +1\}$. Logistic regression attempts to find a separating hyperplane in feature space, parameterized by normal vector $w = (w_1, \ldots, w_D) \in \mathbb{R}^D$, that separates the two classes. The posterior label probability is modeled as:

$$p(y = 1|x, w) = \frac{1}{1 + \exp(-x^T w)} \tag{1}$$

We assume for simplicity that the bias is estimated by incorporating a constant regressor into the feature space so that w_D is the bias term. We will denote by $\mu(x^T w) = p(y = 1|x, w)$. Also, let $X = [x_1, x_2, \ldots, x_N]$ be the data matrix.

For logistic regression, the negative log-likelihood is:

$$\mathcal{L}(w) = -\sum_{i=1}^{N} y_i \log(\mu(x_i^T w)) + (1 - y_i) \log(1 - \mu(x_i^T w)) \tag{2}$$

In many cases, the maximum-likelihood estimator may overfit to the training data. To reduce overfitting, penalized likelihood methods based on l_2-regularization seek to minimize a version of:

$$J(w) = \mathcal{L}(w) + \lambda w^T L w \tag{3}$$

The matrix L can be any symmetric positive semi-definite (PSD) matrix, in which case $J(w)$ is a convex function. For example, L could be a graph Laplacian matrix (Belkin et al. 2004), or the identity matrix.

Newton-type methods optimize for w by iteratively minimizing a quadratic approximation to $J(w)$. The log-likelihood is approximated locally around a current estimate $w^{(t)}$ by a quadratic function $\mathcal{L}_q(w, w^{(t)})$ given by (ignoring terms independent of w):

$$\mathcal{L}_q(w, w^{(t)}) = \frac{1}{2}(w - w^{(t)})^T H (w - w^{(t)}) + \nabla\mathcal{L}(w^{(t)})^T w \tag{4}$$

where H and $\nabla\mathcal{L}(w^{(t)})$ are the Hessian matrix and gradient of $\mathcal{L}(w)$ evaluated at $w = w^{(t)}$:

$$\begin{aligned} \nabla\mathcal{L}(w^{(t)}) &= -X(y - \mu(X^T w^{(t)})) && (5) \\ H &= XRX^T && (6) \end{aligned}$$

Here, we have overloaded the notation $\mu(X^T w^{(t)})$ to denote a column vector whose i^{th} entry is given by $\mu(x_i^T w^{(t)})$. Also, the Hessian changes on each iteration based on the diagonal matrix R, whose diagonal entries contain the current variances of the model fits:

$$R_{ii} = \mu\left(w^{(t)T} x_i\right)\left(1 - \mu\left(w^{(t)T} x_i\right)\right) \tag{7}$$

Substituting $\mathcal{L}_q(w, w^{(t)})$ into (3) gives us our objective to minimize at each iteration:

$$J_q(w) \quad = \quad \mathcal{L}_q(w, w^{(t)}) + \lambda w^T L w \tag{8}$$

Equating the gradient of $J_q(w)$ to zero, we obtain:

$$w^{(t+1)} \quad = \quad (H + 2\lambda L)^{-1}\left[H w^{(t)} - \nabla\mathcal{L}(w^{(t)})\right] \tag{9}$$

This procedure repeats until convergence.

2.2 Simultaneous Logistic Regression

The update equation in (9) amounts to solving a linear system of equations. In many situations, including cross-validation and permutation testing, this computation is performed repeatedly for a set of P different problems. Despite this, there is frequently a large amount of shared structure across the P problems that is otherwise ignored. We propose an efficient method to compute the update in (9) simultaneously across a set of P related problems. Specifically, our approach simultaneously solves P linear systems of the form:

$$A_p w_p \quad = \quad X m_p,\, p = 1, \ldots, P \tag{10}$$

where the matrix A_p is of the form:

$$A_p \quad = \quad X R_p X^T + C \tag{11}$$

and R_p is a diagonal matrix with nonnegative entries as in (7). Note that (9) can be put into the form of (10) and (11) by taking $C = 2\lambda L$ and $m_p = R_p X^T w_p^{(t)} + y - \mu(X^T w_p^{(t)})$. Here, we use the subscript p on a variable to emphasize that it depends on the problem $p \in \{1, \ldots, P\}$. Without the subscript, the variable is assumed to be shared across the P problems.

There are many ways to effectively solve related systems of linear equations such as (10). For example, given a Cholesky factorization for A_1, which enables the linear system to be solved via back substitution, techniques exist to update the factorization for other A_p, $p > 1$ (Golub and Van Loan 1996). However, these methods are geared for situations in which the matrices are related by very low rank perturbations, such as rank-one changes of the form $A_p = A_1 + u_p u_p^T$ for some column vector u_p. The difficulty with (10) is that since the model fits are in general different across the P problems, the weights in the diagonal entries of R_p are not shared across the problems. As a result, the variability in A_p across the P problems cannot be modeled simply as a low-rank perturbation.

Instead, we focus on stationary iterative methods for solving a linear system $Ax = b$ because of their simplicity to implement, and efficiency when designed appropriately. Throughout, we assume that A is invertible. In this case, stationary methods define a sequence of iterates of the form (Young 1971):

$$x^{(k+1)} = G x^{(k)} + d \tag{12}$$

where G is typically called the iteration matrix.

Two important issues involving these methods are: (a) does the sequence in (12) converge; and (b) consistency – when the sequence does converge, does it converge to the unique solution $A^{-1}b$? Fortunately, these questions can be easily answered when the iteration matrix G takes a special form. Specifically, given an additive splitting $A = M - N$, the sequence (12) is consistent provided the iterates take the form:

$$x^{(k+1)} = M^{-1} N x^{(k)} + M^{-1} b \tag{13}$$

Moreover, the sequence will converge from any initialization $x^{(0)}$ iff the spectral radius of $G = M^{-1}N$ is less than one, $\rho(G) < 1$ (Young 1971). The rate of convergence also depends heavily on the spectral radius. In particular, an estimate, $\tilde{K}$, of the number of iterations K required to reduce the norm of the error by a factor ζ is given by (Hageman and Young 1981):

$$\tilde{K} = \frac{\log(\zeta)}{\log(\rho(G))} \tag{14}$$

Stationary iterative methods are most often used to solve a single linear system efficiently. In these cases, M is taken to have a special structure so that $Mx = y$ can be solved very efficiently. For example, the Jacobi iteration takes M to be the diagonal portion of A, while the Gauss-Seidel iteration takes M to be the lower-triangular portion of A.

Our approach, however, is focused on solving many related problems efficiently, rather than one individual problem. As a result, we are willing to spend some time towards solving $Mx = y$, as long as this solution can then be applied to all of the P problems to be solved. Thus, we produce an additive splitting in which the M matrix is shared across the P problems. Given a template matrix $M = XRX^T + C$, for some diagonal matrix R, we define our splittings as:

$$\begin{aligned} A_p &= M - N_p & (15) \\ N_p &= X(R - R_p)X^T & (16) \end{aligned}$$

and the iterations are defined as:

$$w_p^{(k+1)} = M^{-1} N_p w_p^{(k)} + M^{-1} X m_p \tag{17}$$

Since the iterations in (17) are based on an additive splitting, we are assured of consistency. For convergence, however, we must consider the spectral radius of $G_p = M^{-1}N_p$ for each $p = 1, \ldots, P$. In this regard, we have the freedom to design the diagonal matrix R to ensure convergence for all p.

Since M and N_p are both symmetric matrices, we know that the eigenvalues of $M^{-1}N_p$ are all real. Moreover, the eigenvalues of $M^{-1}N_p$ are equal to the generalized eigenvalues $\lambda Mx = N_p x$. Thus, $\rho(G_p) = \max_x |x^T N_p x|/|x^T M x|$. In the following, we assume that C is positive definite, with $\gamma_{\min} > 0$ its minimum eigenvalue. Also, let $\sigma_{\max} > 0$ be the maximum singular value of X. Then we can bound $\rho(G_p)$ as follows:

$$\begin{aligned} \rho(G_p) &= \max_{y \in \mathbb{R}^D : ||y||=1} \left| \frac{y^T X(R - R_p)X^T y}{y^T (XRX^T + C) y} \right| & (18) \\ &\leq \max_{y \in \mathbb{R}^D : ||y||=1} \left| \frac{y^T X(R - R_p)X^T y}{y^T XRX^T y + \gamma_{\min}} \right| & (19) \\ &\leq \max_{w \in \mathbb{R}^N : ||w||=1} \left| \frac{\sigma_{\max}^2 w^T (R - R_p) w}{\sigma_{\max}^2 w^T R w + \gamma_{\min}} \right| & (20) \\ &= \max_{w \in \mathbb{R}^N : ||w||=1} \left| \frac{w^T (R - R_p) w}{w^T R w + \gamma_{\min}/\sigma_{\max}^2} \right| & (21) \end{aligned}$$

To ensure that $\rho(G_p) < 1$, it is sufficient to design the template weighting matrix R so that, for each p,

$$|w^T(R - R_p)w| < |w^T Rw + \gamma_{\min}/\sigma^2_{\max}| \tag{22}$$

for all unit vectors w. This can be satisfied by taking

$$R = \max(R_1, \ldots, R_P) \tag{23}$$

the element-wise maximum. In this case, both R and $R - R_p$ are PSD for all p, and (22) reduces to:

$$w^T R_p w > -\gamma_{\min}/\sigma^2_{\max} \tag{24}$$

which is true since R_p is a PSD matrix, and the right-hand-side of (24) is strictly negative.

2.3 Connection to Bound Optimization Methods

Bound optimization methods are EM-style algorithms that utilize an upper bound on the Hessian (Böhning and Lindsay 1988), (Krishnapuram et al., 2005). For logistic regression, the bound arises from the fact that the diagonal entries of R in (7) are limited to the range $[0, \frac{1}{4}]$, and so $\tilde{H} = \frac{1}{4}XX^T$ upper bounds the Hessian in (6) in the sense that $\tilde{H} - H$ is PSD (denoted $\tilde{H} \succeq H$). From this, a sequence of quasi-Newton steps are taken, with H replaced by $\tilde{H}$.

Thus, applying this technique simultaneously to P problems amounts to solving a set of P linear systems of equations of the form $Aw_p = Xm_p$, where $A = \tilde{H} + C$. Since A is constant through the iterations, the entire sequence of quasi-Newton steps may be taken with only a single inversion computation.

A connection to our approach may be made by the fact that the template matrix computed at each iteration also serves as an upper bound to the Hessian, since taking R as in (23) ensures that $XRX^T \succeq XR_pX^T$ for each p. Thus, our approach may be interpreted as initializing each Newton step from the approximate solution given by the bound optimization, followed by a sequence of iterates in (17) that refines the estimates to the true Newton step solution.

The refinement stage can have a significant improvement in convergence properties: while our system implements the true Newton step, which has quadratic convergence properties in general, the bound optimization method can only guarantee linear convergence (Böhning and Lindsay 1988). Moreover, we show in the next section that the refinement iterates may be computed efficiently.

2.4 Complexity Analysis

In this section, we analyze the computational complexity of each Newton step for our proposed approach and compare it with the brute-force inversion method, which we refer to as the direct method. The key variables in this analysis are the number of features, D, the number of training examples, N, the number of problems to be solved, P, and for our approach, the number of iterations required for (17) to converge, K.

The direct approach requires computing XR_pX^T, for each $p = 1, \ldots, P$, which is $O(DN + D^2N)$. Subsequently, the inversion step of solving $A_pw_p = Xm_p$ is $O(D^3 + 2D^2)$. This results in an overall complexity for the direct method of:

$$O(P(DN + D^2N + D^3 + 2D^2)) \tag{25}$$

Before detailing the complexity of our approach, we first show that the updates in (17) can be computed in one matrix multiplication across all P problems. Let the $D \times N$ matrix Y be the solution to $MY = X$, where M is the template matrix and X is the data matrix. Also, let $W^{(k)}$ be the $D \times P$ matrix whose p^{th} column is equal to the regression weight estimates for problem p at iteration k, $w_p^{(k)}$. Finally, denote $\hat{R}$ as the $N \times P$ matrix whose p^{th} column is the diagonal entries of $R - R_p$, and B the matrix whose p^{th} column is equal to Ym_p. Then $W^{(k+1)}$ can be expressed as:

$$W^{(k+1)} = Y\left[\hat{R} \circ (X^T W^{(k)})\right] + B \tag{26}$$

where $\circ$ denotes the elementwise Hadamard product.

We are now equipped to compute the complexity of our approach. Solving for Y from $MY = X$ requires $O(D^3 + 2D^2N)$ operations, and computing B is $O(DNP)$. The update in (26) requires $O(2PDN + PN)$, and is computed K times. Thus, the overall complexity of our proposed approach is given by:

$$O(D^3 + 2D^2N + P(DN + 2KDN + KN)) \tag{27}$$

As is clear from (14), the number of iterations K for the linear system solver is a balance between the spectral radius of the iteration matrices $M^{-1}N_p$, and the initial error. K does not, however, tend to grow with D or N. Thus, in comparing (25) and (27), we see that while the direct method has two cubic terms D^2N and D^3 that grow with P, our proposed approach scales much better with problem size, having only at most second order terms that grow with P.

Pseudo-code for the algorithm is provided in Algorithm 1. In the following two sections, we discuss two modifications that further improve upon the complexity in (27). The first is based on continuation methods and has the effect of decreasing the number of inner loop iterations K. This is followed by a result in Section 2.6, which is applicable in high-dimensional feature spaces and exploits the discrepancy between the number of features D and the rank of the data matrix. This has the effect of replacing D in (27) with $s = \text{rank}(X)$.

Algorithm 1 Simultaneous Newton Steps

```
Given data matrix $X$ and regularization matrix $C$
Given $R_p, m_p$ for each problem $p = 1, \dots, P$
Initialize $w_p^{(0)}$ and assemble as columns into $W^{(0)}$
Assemble diagonals of $R_p$ as columns into $\hat{R}$
Compute $R$ as row-wise maximum of $\hat{R}$
Compute template matrix $M = XRX^T + C$
Solve $MY = X$ for $Y$
Compute $B$, whose $p^{th}$ column is equal to $Ym_p$
for $k = 1, \dots$ do
  Compute update correction:
      $W^{(k+1)} = Y[\hat{R} \circ (X^T W^{(k)})] + B$
  if $\max_j ||(W^{(k+1)} - W^{(k)})^T e_j||^2 < tol$ then
    $K = k$
    break
  end if
end for
The $p^{th}$ column of $W^{(K)}$ contains the solution to $A_p w_p = Xm_p$, where $A_p = XR_pX^T + C$
```

2.5 Continuation Methods

The regularization matrix C in (11) generally incorporates a parameter λ that trades-off the likelihood fit with the regularization. For example, the ridge penalty takes $C = 2\lambda I$. Since this parameter is not know a-priori, it is often selected by a cross-validation technique that fits the model for a range of λ values.

Similar to (Friedman et al. 2010), requiring fits for multiple λ values can be used to our advantage by a warm-starting procedure, in which results from larger λ values are used to initialize fits for successively smaller λ values. The intuition is that the higher regularization further constrains the problem, making it easier to solve. The solution can then be used to better initialize the more challenging (smaller λ) problems.

In general, the larger λ problems are easier to solve in that they require fewer Newton steps. In the context of the proposed algorithm though, there is a secondary advantage in that for larger λ values, each Newton step is more efficient because the iterative solver (17) tends to require fewer iterations to converge. The reasoning is twofold. First, the spectral radius of the iteration matrix decreases for higher regularization. To see this, let $C^{(1)}, C^{(2)}, \dots, C^{(M)}$ denote the regularization matrices in (11) for a decreasing sequence of regularization parameters $\lambda_1 > \lambda_2 > \cdots > \lambda_M \geq 0$. Note that since $C^{(k)} = \lambda_k C$, for some base PSD matrix C, we know that the eigenvalues of $C^{(k)}$ grow proportionally with λ. From this, it is easy to see that the spectral radius of the iteration matrix $G^{(k)} = (XRX^T + C^{(k)})^{-1}N_p$ tends to decrease as k increases. The second reason fis that higher regularization acts to shrink the estimates, so that there is decreased variability in the model fits. As a result, there tends to be less variability in the entries of the R_p matrices, in which case the template matrix M serves as a better model for the Hessian of all P problems. These intuitions are confirmed in the results.

Additionally, we see from (14) that the number of iterations K also depends on the accuracy of our initialization. Thus, warm-starting can significantly improve convergence for smaller λ values by decreasing the initial approximation error even though the spectral radius of the iteration matrix is larger.

2.6 Exploiting low-rank data matrices

Further computational improvements can be made if there is a large disparity between the number of features in the dataset, D, and the rank of the data matrix, $s = \text{rank}(X)$, (e.g., when $N < D$). In effect, (10) can be transformed from systems of D equations into systems of s equations. For this, we require a low-rank factorization of the data matrix, $X = QZ$, with $Q \in \mathbb{R}^{D \times s}$ having orthonormal columns and $Z \in \mathbb{R}^{s \times N}$. The result follows from the next 2 lemmas, whose proofs are given in supplementary material.

Lemma 1. *The solution w_p to $A_p w_p = Xm_p$ satisfies $w_p \in range(C^{-1}Q)$.*

Lemma 2. *Let $\begin{bmatrix} Q & W \end{bmatrix}$ be a matrix with orthonormal columns that is a basis for $range([Q, C^{-1}Q])$. Note that W has at most s columns. Then the solution w_p to $A_p w_p = Xm_p$ can be computed by first solving a linear system of s equations:*

$$\tilde{A}_p \tilde{w}_p = Zm_p \tag{28}$$

where

$$\tilde{A}_p = ZR_pZ^T + \tilde{C} \tag{29}$$

$$\tilde{C} = Q^T C(Q - WF) \tag{30}$$

$$F = (W^T CW)^{-1} W^T CQ \tag{31}$$

Then w_p can be obtained from $\tilde{w}_p$ by:

$$w_p = (Q - WF)\,\tilde{w}_p \tag{32}$$

Since (28) takes the same form as (10), we may implement our proposed algorithm to solve (28). This yields significant computational savings when $s < D$. Specifically, in the complexity equation of our approach in (27), all occurrences of D are replaced with s.

3 Applications

3.1 Permutation Testing

Permutation tests are a popular nonparametric technique for estimating classifier significance (Ojala and Garriga 2010). Here, a hypothesis test is formulated to see whether or not a meaningful relationship between the features and the class labels has been identified by the classifier. A distribution of classifier accuracy under the null hypothesis that no relationship exists is computed by re-training the classifier on new datasets in which the labels have been permuted across instances. This allows the derived classifier accuracy to be assigned a p-value of statistical significance.

Permutation tests fit naturally into our method. Since only the binary labels y change in each permutation problem $p = 1, \ldots, P$, the data matrix X is shared across the P problems. As a result, each permutation p can be written in the form (10) for some R_p and m_p.

3.2 Cross-validation

For K-fold cross-validation, the logistic regression model is trained on K different subsets of the training data. At first glance, it appears that the data matrix is not shared across problems, because even though there will be a large amount of overlap in training data across folds, each fold still contains a different subset.

This technicality is overcome by a small modification. For a particular fold p of the data, let h_p index the training instances that are excluded from the training data. Expressing this fold's Newton step in the form of (10) simply involves zeroing out the h_p diagonals of R_p, as well as the h_p indices of m_p. With this modification, the full data matrix X can be shared across all K folds.

Thus, K-fold cross-validation fits into our approach given the slight modification mentioned above, with the problem size P equal to the number of cross-validation folds. For leave-one-out cross-validation with $P = N$, our approach yields significant savings when there are a large number of trials. However, the true value of our approach is not really apparent if the number of folds is small, as in 10-fold cross-validation. As noted in Kohavi (1995), the cross-validation error depends on the random split of the data and may be highly biased for small K. To minimize this bias, it is best to average cross-validation error over multiple random K-fold splits of the data. In this case, our approach is again valuable, even if K is small.

4 Results

We applied our algorithm to a number of real-world datasets and benchmarked its speed against the direct method of inverting the Hessian matrix independently for each problem p. To enable a fair comparison, the convergence tolerance used in our algorithm was set very conservatively at $tol = 1 \times 10^{-11}$. In all analyses, the regularization matrix was taken to be a scaled identity matrix, with the last diagonal entry set to zero. Zeroing out the last entry prevents shrinkage of the bias term. The amount of regularization is controlled by the parameter λ.

The MNIST handwritten digits database (LeCun et al. 1998), is a well established machine learning dataset. The data consist of $6,000$ grey-level 28×28 images for each digit 0 through 9. Two binary classification datasets were derived from this database – discriminating digit 0 from digit 1, and the more challenging case of distinguishing digits 4 and 9. Features corresponded to pixel intensities, with $D = 28^2 + 1 = 785$.

Leave-one-out classification was performed on subsampled datasets of size $1,000$ up to $10,000$, in increments of $1,000$. For each training set size, we compared the computation times of our approach and the direct method by computing the number of leave-one-out runs that could be computed by the direct method in the same amount of time that was required by our method to complete the full leave-one-out cross-validation set. We call this quantity the effective problem size of our method, and the ratio of P (training set size) to effective problem size gives the overall speedup factor. Plots of effective problem size are given in Figure 1. It is clear that the computational speedup is consistently around 100× for both MNIST datasets and values of the regularization parameter λ.

We also benchmarked our algorithm on the MNIST data for the case of 10-fold cross-validation repeated over 100 random splits, so that $P = 1000$ total problems. Similar to the leave-one-out analysis, we again plot effective problem size against the training set size for various values of the regularization parameter. In this case, P is constant over training set size, so the speedup factor is the ratio of 1000 to effective problem size. Plots are given in Figure 2. In this case, speedup ranges from 30× to 100×. A contributing factor for the reduced speedup, as compared to the leave-one-out analysis, is due to the fact that there is more variability in the model fits for 10-fold cross-validation than leave-one-out cross-validation.

We also applied our technique to classifier significance testing problems on electroencephalography (EEG) and functional MRI (fMRI) datasets collected during an auditory oddball detection task (Goldman et al. 2009). Permutation tests were used to evaluate performance of oddball vs. standard prediction performance of classifiers derived separately from the EEG and fMRI data. There were a total of $N = 374$ examples, with the EEG feature space corresponding to the measured voltages from 43 bipolar electrodes averaged over a 50ms window ($D = 44$), while for the fMRI classifier the feature space was the BOLD response of 300 voxels in auditory cortex ($D = 301$).

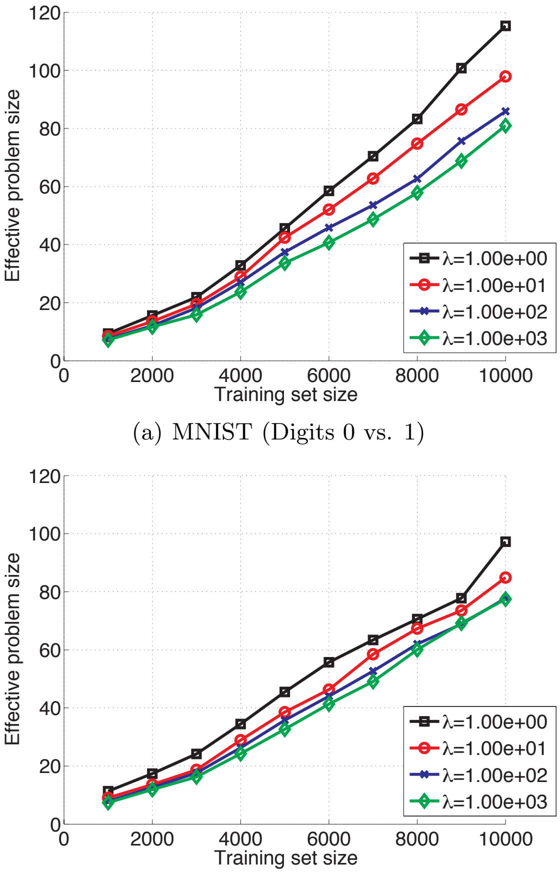
Figure 3 plots the effective problem size against the number of permutation tests P for both the EEG classifier and the fMRI classifier and various regularization values λ. For the EEG classifier, the computational improvement is $\sim 10\times$, while for the fMRI classifier, the improvement is $\sim 100\times$. The more dramatic improvement for the fMRI case is due to the higher feature space dimension ($D = 301$ vs. $D = 44$).

Finally, we ran an additional analysis on the MNIST dataset to test the effectiveness of applying a continuation method to our approach, as detailed in Section 2.5. Here we compared the overall time required to perform leave-one-out cross-validation for regularization values $\lambda \in \{1, 10, \ldots, 1e10\}$ between initializing each regularization from $w = 0$, and warm starting from the results of the next highest regularization value. This comparison was performed across a variety of training set sizes. The continuation method provided an additional improvement of $\sim 1.5\times$, and this was consistent across training set sizes.

5 Conclusion and Future Work

This paper addresses the computational problem of l_2-regularized logistic regression that is commonly encountered in model selection and assessment, where the solution to a large number of related problems must be computed. We derive a principled iterative algorithm that is simple to implement and efficiently solves all of the problems simultaneously. We show that this results in a significant improvement in complexity over the direct approach, and we demonstrate this empirically on real-world datasets.

There are a number of avenues of future research related to this algorithm. The first involves extending this algorithm to fitting other regression problems. Logistic regression is one example of a generalized linear model (GLM), which allows for very flexible modeling assumptions. Since all GLM's can be fit almost completely analogously to the IRLS procedure of logistic regression, generalizing the algorithm to all GLM's and testing its validity is a natural next step.



(a) MNIST (Digits 0 vs. 1)

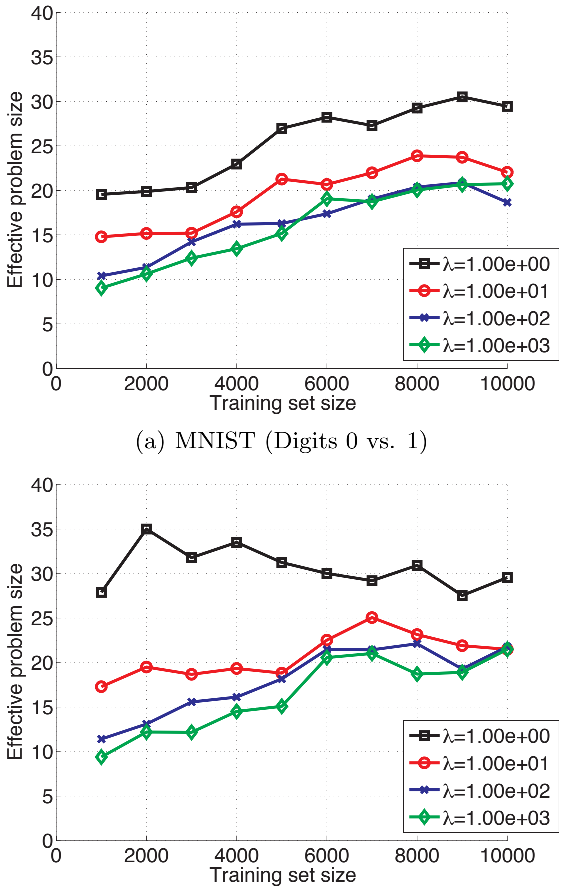
(b) MNIST (Digits 4 vs. 9)

Figure 1: Leave-one-out cross-validation (P = training set size) using the proposed method compared against the direct IRLS method for the MNIST datasets and various training set sizes. The effective problem size represents the number of problems that can be solved by the direct method in the same amount of time as the entire LOO set for the proposed method. For example, LOO cross-validation by our proposed approach on a training set size of $P = 2000$ may be computed in the same time as approximately $P = 20$ folds by the direct method, resulting in a speedup of 100×.

Finally, the current version of the algorithm is limited to l_2-regularized problems, which excludes the sparsity-inducing LASSO regularizers. Extending this algorithm to l_1 and mixed l_1-l_2 regularizers, such as the elastic net, is another important open question.

Acknowledgements

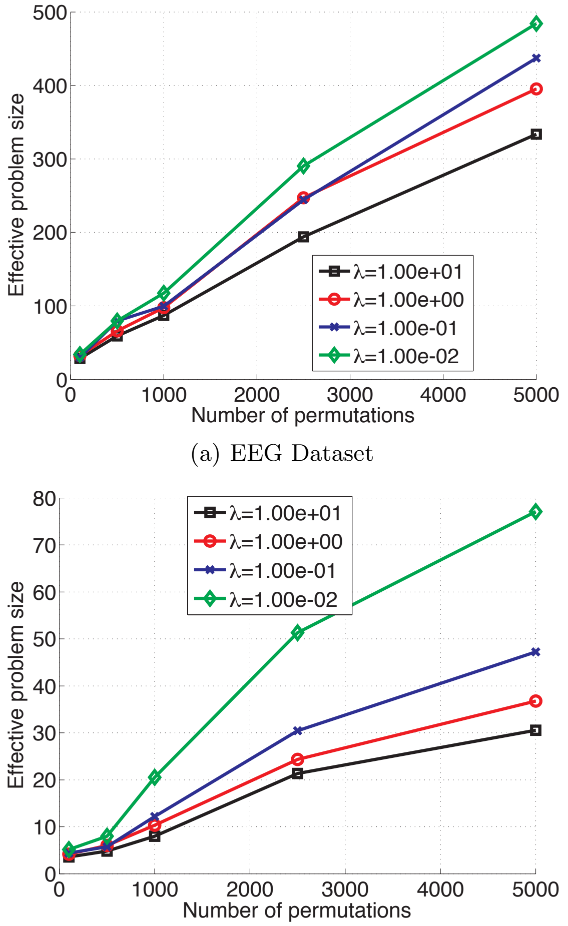
The authors acknowledge Jennifer Walz for providing the EEG and fMRI datasets used in the results. This work was supported by National Institutes of Health grant R01-MH085092.



(a) MNIST (Digits 0 vs. 1)

(b) MNIST (Digits 4 vs. 9)

Figure 2: 10-fold cross-validation using the proposed method for the MNIST datasets and various training set sizes. Each cross-validation run was repeated over 100 random splits, resulting in $P = 1000$ total problems. The effective problem size represents the number of problems that can be solved by the direct method in the same amount of time as the P problems for the proposed method. Here, the ratio of 1000 to the effective problem size gives the computational speedup, which varies between 30× to 100×.



(a) EEG Dataset

(b) fMRI Dataset

Figure 3: Permutation testing using the proposed method compared against the direct IRLS method for the EEG and fMRI datasets and various numbers of permutations P. The effective problem size represents the number of permutations that can be solved by the direct method in the same time as the entire set of P permutations by our proposed approach. Here, the ratio of number of permutations P to the effective problem size gives the computational speedup.

References

M. Belkin, I. Matveeva and P. Niyogi (2004). Regularization and Semi-supervised Learning on Large Graphs. In J. Shawe-Taylor, Y. Singer (eds.), *Learning Theory, 17th Annual Conference on Learning Theory, COLT*. Banff, Canada.

D. Böhning and B.G. Lindsay (1988). Monotonicity of Quadratic Approximation Algorithms. *Annals Institute of Statistical Mathematics* **40**(4): 641-663.

G.C. Cawley and N.L.C. Talbot (2004). Efficient model selection for kernel logistic regression. In Proceedings of the 17th International Conference on Pattern Recognition (ICPR). Cambridge, UK.

J. Friedman, T. Hastie and R. Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. *Journal of Statistical Software* **33**(1):1-22.

A. Genkin, D.D. Lewis and D. Madigan (2007). Large-Scale Bayesian Logistic Regression for Text Categorization. *Technometrics* **49**(3): 291-304.

R.I. Goldman, C.Y. Wei, M.G. Philiastides, A.D. Gerson, D. Friedman, T.R. Brown and P. Sajda (2009). Single-trial discrimination for integrating simultaneous EEG and fMRI: Identifying cortical areas contributing to trial-to-trial variability in the auditory oddball task. *NeuroImage* **47**(1) 136-147.

G.H. Golub and C.F. Van Loan (1996). *Matrix Computations*. Baltimore, MD: Johns Hopkins University Press.

L.A. Hageman and D.M. Young (1981). *Applied Iterative Methods*. Mineola, NY: Dover Publications.

R.A. Horn and C.R. Johnson (1990). *Matrix Analysis*. New York, NY: Cambridge University Press.

D.W. Hosmer and S. Lemeshow (2000). *Applied Logistic Regression*. USA: John Wiley & Sons, Inc.

P.R. Komarek and A.W. Moore (2003). Fast Robust Logistic Regression for Large Sparse Datasets with Binary Outputs. In C.M. Bishop, B.J. Frey (eds.), *Proceedings of the 9th International Workshop on Artificial Intelligence and Statistics (AISTATS)*. Key West, Florida.

P. Komarek and A. Moore (2005). Making Logistic Regression A Core Data Mining Tool: A Practical Investigation of Accuracy, Speed, and Simplicity. *CMU Tech. Report CMU-RI-TR-05-27*, Robotics Institute, Carnegie Mellon University.

R. Kohavi (1995). A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*. Montreal, Quebec.

B. Krishnapuram, L. Carin, M.A.T. Figueiredo and A.J. Hartemink (2005). Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **27**(6):957-968.

Y. LeCun, L. Bottou, Y. Bengio and P. Haffner (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE* **86**(11):2278-2324.

M. Ojala and G. C. Garriga (2010). Permutation Tests for Studying Classifier Performance. *Journal of Machine Learning Research* **11**:1833-1863.

R.M. Rifkin and R.A. Lippert (2007). Notes on Regularized Least Squares. *MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) Technical Report.*

D.M. Young (1971). *Iterative solution of large linear systems*. Mineola, NY: Dover Publications.

T. Zhang and F. Oles (2001). Text Categorization Based on Regularized Linear Classifiers. *Information Retrieval* **4**:5-31.

T. Zhang (2003). Leave-One-Out Bounds for Kernel Methods. *Neural Computation* **15**(6):1397-1437.