Sparse Multinomial Logistic Regression: Fast Algorithms and Generalization Bounds

Balaji Krishnapuram, Lawrence Carin, *Fellow*, *IEEE*, Mário A.T. Figueiredo, *Senior Member*, *IEEE*, and Alexander J. Hartemink

Abstract—Recently developed methods for learning sparse classifiers are among the state-of-the-art in supervised learning. These methods learn classifiers that incorporate weighted sums of basis functions with sparsity-promoting priors encouraging the weight estimates to be either significantly large or exactly zero. From a learning-theoretic perspective, these methods control the capacity of the learned classifier by minimizing the number of basis functions used, resulting in better generalization. This paper presents three contributions related to learning sparse classifiers. First, we introduce a true multiclass formulation based on multinomial logistic regression. Second, by combining a bound optimization approach with a component-wise update procedure, we derive fast exact algorithms for learning sparse multiclass classifiers that scale favorably in both the number of training samples and the feature dimensionality, making them applicable even to large data sets in high-dimensional feature spaces. To the best of our knowledge, these are the first algorithms to perform exact multinomial logistic regression with a sparsity-promoting prior. Third, we show how nontrivial generalization bounds can be derived for our classifier in the binary case. Experimental results on standard benchmark data sets attest to the accuracy, sparsity, and efficiency of the proposed methods.

Index Terms—Supervised learning, classification, sparsity, Bayesian inference, multinomial logistic regression, bound optimization, expectation maximization (EM), learning theory, generalization bounds.

1 Introduction

Recently developed sparse classification algorithms have quickly established themselves among the state-of-the-art in supervised learning. This family of algorithms includes the *relevance vector machine* (RVM) [35], the *sparse probit regression* (SPR) algorithm [10], [11], *sparse online Gaussian processes* [6], the *informative vector machine* (IVM) [22], and the *joint classifier and feature optimization* (JCFO) algorithm [16], [17]. These algorithms learn classifiers that are constructed as weighted linear combinations of basis functions; the weights are estimated in the presence of training data. In many of these algorithms, the set of permitted basis functions is unrestricted; e.g., they may be the original features themselves, some nonlinear transformation of those features, or even kernels centered on the training samples. In this latter case, the learned classifier will be similar in flavor to a *support vector machine* (SVM) [5], [38], although, in contrast to an SVM, the kernel is not required to satisfy the Mercer condition.

• *B. Krishnapuram is with the Computer Aided Diagnosis and Therapy Group, Siemens Medical Solutions USA, Inc., 51 Valley Stream Pkwy, Malvern, PA 19355. E-mail: Balaji.Krishnapuram@siemens.com.*
• *L. Carin is with the Department of Electrical Engineering, Duke University, Durham, NC 27708-0291. E-mail: lcarin@ee.duke.edu.*
• *M.A.T. Figueiredo is with the Institute of Telecommunications, Department of Electrical and Computer Engineering, Instituto Superior Técnico, 1049-001 Lisboa, Portugal. E-mail: mario.figueiredo@lx.it.pt.*
• *A.J. Hartemink is with the Department of Computer Science, Duke University, Durham, NC 27708-0129. E-mail: amink@cs.duke.edu.*

Manuscript received 22 Apr. 2004; revised 7 Dec. 2004; accepted 7 Jan. 2005; published online 14 Apr. 2005.
Recommended for acceptance by L. Kuncheva.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0190-0404.

Since the goal of sparse classification algorithms is to learn as sparse a classifier as possible, the likelihood of the weights in the presence of training data is typically regularized by some prior belief about the weights that promotes their sparsity. The prior is sometimes implicit—as is the case with the *automatic relevance determination* (ARD) framework [28] exploited by the RVM—but is often explicit (as in, e.g., [11], [17], [34]). In the latter case, a common choice of prior in this family of algorithms is the Laplacian, which results in an l_1-penalty, analogous to the LASSO penalty for regression [34]. The sparsity-promoting nature of the Laplacian prior is theoretically well-justified (see [9], [12], [29], as well as references therein) and has been found to be practically and conceptually useful in several research areas [4], [23], [30], [41]. Another interesting property of the Laplacian is that it is the most heavy-tailed density that is still log-concave (though not strictly so); thus, when combined with a concave log-likelihood, it leads to a concave log-posterior with a unique maximum. Since we learn classifiers under a maximum a posteriori criterion in this paper, the existence of a unique maximum proves to be quite advantageous.

From a learning-theoretic perspective, controlling the capacity of the learned classifier is necessary to guarantee good generalization performance [38]. In the context of sparse methods, controlling the capacity of the learned classifier is accomplished by minimizing the number of basis functions used in constructing the decision surface. The sparsity-promoting prior plays a key role in this process by encouraging the weight estimates to be either significantly large or exactly zero, which has the effect of automatically removing irrelevant basis functions from consideration. If the basis functions are chosen to be the original features themselves, this results in automatic feature selection as a side-effect of classifier design, enabling the classifier to effectively overcome the *curse of dimensionality*. If the basis functions are kernels, methods like JCFO can be used to achieve sparsity in the kernel basis functions and automatic feature selection at the same time [16], [17].

In this paper, we address three issues related to learning sparse classifiers. First, we adopt a genuinely multiclass formulation based on multinomial logistic regression introducing the notation for this formulation in the next section; strictly speaking, a multinomial logistic regression formulation for multiclass classification is certainly not new (for example, see [2]), but it is rarely employed in the pattern recognition and machine learning literature. Although not common in the current literature, such an approach can be fruitfully extended to many other sparse classification algorithms, such as the RVM or IVM. We highlight this aspect of our approach in order to contrast it with "1-versus-all" and other similar heuristics that are frequently adopted in current practice.

Second, by combining a bound optimization approach [7], [18], [19] with a component-wise update procedure, we derive in Section 3 a series of new fast algorithms for learning a sparse multiclass classifier that scale favorably in both the number of training samples and the feature dimensionality, making these methods applicable even to large data sets in high-dimensional feature spaces. To the best of our knowledge, these are the first algorithms to perform exact multinomial logistic regression with an explicit sparsity-promoting prior. We call the proposed *sparse multinomial logistic regression* approach SMLR (pronounced "smaller"). Its nonsparse counterpart, which we formulate only for the purposes of comparison, is based on a Gaussian prior (l_2-penalty) and called RMLR (for *ridge multinomial logistic regression*).

Third, in Section 4, we derive generalization bounds for our methods based on recently published learning-theoretic results [26], [33]. Similar in nature to the margin bounds that are frequently used to justify the SVM [38], these bounds can be used to provide theoretical insight into and justification for our algorithm.

Section 5 contains a series of experimental results on benchmark data sets that attest to the accuracy and efficiency of these methods. We conclude, in Section 6, with a critical discussion of our algorithms and learning-theoretical bounds.

2 Multinomial Logistic Regression

The goal of a supervised learning algorithm is to leverage a set of n training samples in order to design a classifier that is capable of distinguishing between m classes on the basis of an input vector of length d. In the context of a linear classifier, the d components of the input vector $\boldsymbol{x} = [x_1, \dots, x_d]^T \in \mathbb{R}^d$ are simply the d observed features. In a more general context, the d components of $\boldsymbol{x}$ are the values of d (possibly nonlinear) basis functions computed from the observed features. In the latter case, while d is the number of basis functions, it need not be the number of originally observed features. For example, in a kernel classifier, the basis functions will be kernels centered at the training samples, meaning that d will equal n regardless of the number of originally observed features [5], [15]. To simplify notation and exposition, we will denote any of these choices simply as $\boldsymbol{x}$ and remind the reader that what follows is equally applicable in the context of linear, nonlinear, or kernel classification.

We adopt the common technique of representing the class labels using a "1-of-m" encoding vector $\boldsymbol{y} = [y^{(1)}, y^{(2)}, \dots, y^{(m)}]^T$ such that $y^{(i)} = 1$ if $\boldsymbol{x}$ corresponds to an example belonging to class i and $y^{(i)} = 0$ otherwise. The n training samples can thus be represented as a set of training data $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{y}_1), \dots, (\boldsymbol{x}_n, \boldsymbol{y}_n)\}$.

Under a multinomial logistic regression model, the probability that $\boldsymbol{x}$ belongs to class i is written as

$$P\left(y^{(i)} = 1 | \boldsymbol{x}, \boldsymbol{w}\right) = \frac{\exp\left(\boldsymbol{w}^{(i)T}\boldsymbol{x}\right)}{\sum_{j=1}^{m} \exp\left(\boldsymbol{w}^{(j)T}\boldsymbol{x}\right)}, \tag{1}$$

for $i \in \{1, \dots, m\}$, where $\boldsymbol{w}^{(i)}$ is the weight vector corresponding to class i and the superscript ^{T} denotes vector/matrix transpose. For binary problems ($m = 2$), this is known as a logistic regression model; for $m > 2$, the usual designation is multinomial logistic regression (or soft-max in the neural networks literature). Because of the normalization condition

$$\sum_{i=1}^{m} P\left(y^{(i)} = 1 | \boldsymbol{x}, \boldsymbol{w}\right) = 1,$$

the weight vector for one of the classes need not be estimated. Without loss of generality, we thus set $\boldsymbol{w}^{(m)} = \boldsymbol{0}$ and the only parameters to be learned are the weight vectors $\boldsymbol{w}^{(i)}$ for $i \in \{1, \dots, m-1\}$. For the remainder of the paper, we use $\boldsymbol{w}$ to denote the $(d(m-1))$-dimensional vector of parameters to be learned.

In a supervised learning context, the components of $\boldsymbol{w}$ are estimated from the training data $\mathcal{D}$. To perform maximum likelihood (ML) estimation of $\boldsymbol{w}$, one simply maximizes the log-likelihood function,

$$\begin{aligned} \ell(\boldsymbol{w}) &= \sum_{j=1}^{n} \log P\left(\boldsymbol{y}_j | \boldsymbol{x}_j, \boldsymbol{w}\right) \\ &= \sum_{j=1}^{n} \left[\sum_{i=1}^{m} y_j^{(i)} \boldsymbol{w}^{(i)T} \boldsymbol{x}_j - \log \sum_{i=1}^{m} \exp\left(\boldsymbol{w}^{(i)T} \boldsymbol{x}_j\right) \right], \end{aligned} \tag{2}$$

which is typically accomplished using Newton's method, also known, in this case, as *iteratively reweighted least squares* (IRLS). Although there are other methods for performing this maximization, none clearly outperforms IRLS [27]. However, when the training data is separable, the function $\ell(\boldsymbol{w})$ can be made arbitrarily large, so a prior on $\boldsymbol{w}$ is crucial. This motivates the adoption of a maximum a posteriori (MAP) estimate (or penalized ML estimate),

$$\widehat{\boldsymbol{w}}_{\text{MAP}} = \arg\max_{\boldsymbol{w}} L(\boldsymbol{w}) = \arg\max_{\boldsymbol{w}} [\ell(\boldsymbol{w}) + \log p(\boldsymbol{w})],$$

with $p(\boldsymbol{w})$ being some prior on the parameters $\boldsymbol{w}$.

Although the IRLS algorithm can be trivially modified to accommodate a Gaussian prior on $\boldsymbol{w}$ (see, for example, [43]), other priors are not so easily handled. In particular, a sparsity-promoting Laplacian prior,

$$p(\boldsymbol{w}) \propto \exp(-\lambda \|\boldsymbol{w}\|_1), \tag{3}$$

where $\|\boldsymbol{w}\|_1 = \sum_l |w_l|$ denotes the l_1 norm, can't be used with IRLS because it is nondifferentiable at the origin. In the next section, we describe a bound optimization algorithm for estimating $\boldsymbol{w}$ that can be used with a Laplacian prior but has the same computational cost as IRLS.

3 Fast Iterative Algorithms

3.1 Bound Optimization Algorithms

Consider $L(\boldsymbol{w}) = \ell(\boldsymbol{w}) + \log p(\boldsymbol{w})$, the objective function to be maximized. In the bound optimization approach [19], $L(\boldsymbol{w})$ is optimized by iteratively maximizing a *surrogate function* Q, thus:

$$\widehat{\boldsymbol{w}}^{(t+1)} = \arg\max_{\boldsymbol{w}} Q(\boldsymbol{w}|\widehat{\boldsymbol{w}}^{(t)}). \tag{4}$$

Such a procedure monotonically increases the value of the objective function at each iteration if the surrogate function satisfies a certain *key condition*, namely, that $L(\boldsymbol{w}) - Q(\boldsymbol{w}|\widehat{\boldsymbol{w}}^{(t)})$ attains its minimum when $\boldsymbol{w} = \widehat{\boldsymbol{w}}^{(t)}$. This can be seen easily:

$$\begin{aligned} L(\widehat{\boldsymbol{w}}^{(t+1)}) &= L(\widehat{\boldsymbol{w}}^{(t+1)}) - Q(\widehat{\boldsymbol{w}}^{(t+1)}|\widehat{\boldsymbol{w}}^{(t)}) + Q(\widehat{\boldsymbol{w}}^{(t+1)}|\widehat{\boldsymbol{w}}^{(t)}) \\ &\geq L(\widehat{\boldsymbol{w}}^{(t)}) - Q(\widehat{\boldsymbol{w}}^{(t)}|\widehat{\boldsymbol{w}}^{(t)}) + Q(\widehat{\boldsymbol{w}}^{(t+1)}|\widehat{\boldsymbol{w}}^{(t)}) \\ &\geq L(\widehat{\boldsymbol{w}}^{(t)}) - Q(\widehat{\boldsymbol{w}}^{(t)}|\widehat{\boldsymbol{w}}^{(t)}) + Q(\widehat{\boldsymbol{w}}^{(t)}|\widehat{\boldsymbol{w}}^{(t)}) \\ &= L(\widehat{\boldsymbol{w}}^{(t)}), \end{aligned} \tag{5}$$

where the first inequality results from the condition that $L(\boldsymbol{w}) - Q(\boldsymbol{w}|\widehat{\boldsymbol{w}}^{(t)})$ attains its minimum when $\boldsymbol{w} = \widehat{\boldsymbol{w}}^{(t)}$, while the second inequality stems from the fact that $Q(\boldsymbol{w}|\widehat{\boldsymbol{w}}^{(t)})$ is maximized when $\boldsymbol{w} = \widehat{\boldsymbol{w}}^{(t+1)}$ (see (4)). The standard *expectation-maximization* (EM) algorithm for ML estimation with missing data is a special case of this approach, with the key condition being a consequence of Jensen's inequality [8], [18], [19].

As suggested in [7], [19], the bound optimization perspective allows us to derive alternative EM-style algorithms without invoking the concept of missing data. In fact, a monotonic algorithm of the form (4) can be obtained from any surrogate function $Q(\boldsymbol{w}|\boldsymbol{w}')$ satisfying the key condition. Finding such a surrogate function can often be achieved by purely analytical methods (e.g., using convenient inequalities), which clearly separates the computational from the probabilistic modeling aspects of the problem [7].

One way to obtain a surrogate function $Q(\boldsymbol{w}|\boldsymbol{w}')$ when $L(\boldsymbol{w})$ is concave is by using a bound on its Hessian (which, if it exists, is negative definite) [3], [18], [19]. If the Hessian $\mathbf{H}$ is lower bounded, i.e., if there exists a negative definite matrix $\mathbf{B}$ such that[1] $\mathbf{H}(\boldsymbol{w}) \succeq \mathbf{B}$ for any $\boldsymbol{w}$, then it is easy to prove that, for any $\boldsymbol{w}'$,

1. For two square matrices of the same size, $\mathbf{M}_1 \succeq \mathbf{M}_2$ denotes that $\mathbf{M}_1 - \mathbf{M}_2$ is positive semidefinite.

$$L(\boldsymbol{w}) \geq L(\boldsymbol{w}') + (\boldsymbol{w} - \boldsymbol{w}')^T \mathbf{g}(\boldsymbol{w}') + \frac{1}{2}(\boldsymbol{w} - \boldsymbol{w}')^T \mathbf{B} (\boldsymbol{w} - \boldsymbol{w}'),$$

where $\mathbf{g}(\boldsymbol{w}')$ denotes the gradient of $L(\boldsymbol{w})$ computed at $\boldsymbol{w}'$. Letting the right-hand side of the previous inequality be denoted as $Q(\boldsymbol{w}|\boldsymbol{w}')$, we have $L(\boldsymbol{w}) - Q(\boldsymbol{w}|\boldsymbol{w}') \geq 0$, with equality if and only if $\boldsymbol{w} = \boldsymbol{w}'$. Thus, $Q(\boldsymbol{w}|\boldsymbol{w}')$ is a valid surrogate function and we obtain a monotonic algorithm by letting

$$Q(\boldsymbol{w}|\widehat{\boldsymbol{w}}^{(t)}) = \boldsymbol{w}^T \left(\mathbf{g}(\widehat{\boldsymbol{w}}^{(t)}) - \mathbf{B}\widehat{\boldsymbol{w}}^{(t)} \right) + \frac{1}{2} \boldsymbol{w}^T \mathbf{B} \boldsymbol{w}, \tag{6}$$

where terms that are irrelevant for the maximization with respect to $\boldsymbol{w}$ have been dropped. Maximization of this surrogate function leads to the simple update equation

$$\widehat{\boldsymbol{w}}^{(t+1)} = \widehat{\boldsymbol{w}}^{(t)} - \mathbf{B}^{-1} \mathbf{g}(\widehat{\boldsymbol{w}}^{(t)}), \tag{7}$$

which is similar to an IRLS iteration, but with $\mathbf{B}$ replacing the Hessian. The crucial advantage is that the inverse $\mathbf{B}^{-1}$ can be precomputed once, whereas in IRLS, a different matrix must be inverted at each iteration [3]. In the following three subsections, we will show how this update equation can be applied to maximum likelihood (ML) multinomial logistic regression as an alternative to IRLS and how simple variations can handle maximum a posteriori (MAP) multinomial logistic regression with either a Gaussian (l_2-penalty) or Laplacian (l_1-penalty) prior on the weights.

3.2 Application to ML Multinomial Logistic Regression

Let $p_j^{(i)}(\boldsymbol{w}) = P(y_j^{(i)} = 1|\boldsymbol{x}_j, \boldsymbol{w})$ and then let us define the vector $\boldsymbol{p}_j(\boldsymbol{w}) = [p_j^{(1)}(\boldsymbol{w}), \ldots, p_j^{(m-1)}(\boldsymbol{w})]^T$ and the diagonal matrix $\boldsymbol{P}_j(\boldsymbol{w}) = \mathrm{diag}\{p_j^{(1)}(\boldsymbol{w}), \ldots, p_j^{(m-1)}(\boldsymbol{w})\}$. For ML multinomial logistic regression, the objective function $L(\boldsymbol{w})$ is just the log-likelihood $\ell(\boldsymbol{w})$, as given in (2). This is a concave function with Hessian (see [2])

$$\mathbf{H}(\boldsymbol{w}) = -\sum_{j=1}^{n} \left(\boldsymbol{P}_j(\boldsymbol{w}) - \boldsymbol{p}_j(\boldsymbol{w})\, \boldsymbol{p}_j^T(\boldsymbol{w}) \right) \otimes \left(\boldsymbol{x}_j \boldsymbol{x}_j^T \right),$$

where $\otimes$ is the Kronecker matrix product. Since the $\boldsymbol{x}_j$ are d-dimensional and the $\boldsymbol{p}_j(\boldsymbol{w})$ are $(m-1)$-dimensional, $\mathbf{H}(\boldsymbol{w})$ is a square matrix of size $d(m-1)$. As shown in [2], the Hessian is, in fact, lower bounded by a negative definite matrix that does not depend on $\boldsymbol{w}$,

$$\mathbf{H}(\boldsymbol{w}) \succeq -\frac{1}{2} \left[\mathbf{I} - \mathbf{1}\mathbf{1}^T/m \right] \otimes \sum_{j=1}^{n} \boldsymbol{x}_j \boldsymbol{x}_j^T \equiv \mathbf{B}, \tag{8}$$

where $\mathbf{1} = [1, 1, \ldots, 1]^T$. The gradient of $\ell(\boldsymbol{w})$ is also simple to obtain in closed form:

$$\mathbf{g}(\boldsymbol{w}) = \sum_{j=1}^{n} (\boldsymbol{y}_j' - \boldsymbol{p}_j(\boldsymbol{w})) \otimes \boldsymbol{x}_j, \tag{9}$$

where $\boldsymbol{y}_j' = [y_j^{(1)}, y_j^{(2)}, \ldots, y_j^{(m-1)}]^T$. Finally, the bound optimization iterations for ML multinomial logistic regression have the form (7), with $\mathbf{g}(\boldsymbol{w})$ given in (9) and $\mathbf{B}$ given in (8).

3.3 Application to MAP Multinomial Logistic Regression with Gaussian Prior

The algorithm described in the previous section is trivially modified in the presence of a Gaussian prior on $\boldsymbol{w}$, as used in ridge regression or the penalized logistic regression method considered in [43]. In this case, the objective function is

$$L(\boldsymbol{w}) = \ell(\boldsymbol{w}) - \frac{\lambda}{2}\,\|\boldsymbol{w}\|_2^2,$$

where $\|\cdot\|_2^2$ denotes a squared Euclidean norm and λ is the inverse prior variance. The only modification of the algorithm is that, in each iteration, we now have to maximize

$$Q(\boldsymbol{w}|\widehat{\boldsymbol{w}}^{(t)}) - \frac{\lambda}{2}\,\|\boldsymbol{w}\|_2^2,$$

leading to the update equation

$$\widehat{\boldsymbol{w}}^{(t+1)} = (\mathbf{B} - \lambda\mathbf{I})^{-1}\Big(\mathbf{B}\widehat{\boldsymbol{w}}^{(t)} - \mathbf{g}(\widehat{\boldsymbol{w}}^{(t)})\Big). \tag{10}$$

Since we can precompute $(\mathbf{B} - \lambda\mathbf{I})^{-1}\mathbf{B}$ and $(\mathbf{B} - \lambda\mathbf{I})^{-1}$, each iteration of this bound optimization method for multinomial logistic regression under a Gaussian prior is still significantly cheaper than each IRLS iteration for ML multinomial logistic regression.

3.4 Application to MAP Multinomial Logistic Regression with Laplacian Prior

However, our main goal is not to use a Gaussian prior, but a Laplacian prior, given in (3). This is nontrivial at first glance because $Q(\boldsymbol{w}|\widehat{\boldsymbol{w}}^{(t)}) - \lambda\|\boldsymbol{w}\|_1$ cannot be maximized in closed form. We address this problem by identifying a lower bound for the log-prior also. To this end, we write $|w_l| = \sqrt{w_l^2}$ and exploit the concavity of the square-root. In particular,

$$\sqrt{u} \le \sqrt{v} + \frac{u-v}{2\sqrt{v}} = \frac{u}{2\sqrt{v}} + \frac{\sqrt{v}}{2},$$

with equality if and only if $u = v$; thus, for any $\boldsymbol{w}'$,

$$-\|\boldsymbol{w}\|_1 \ge -\frac{1}{2}\left(\sum_l \frac{w_l^2}{|w_l'|} + \sum_l |w_l'|\right),$$

with equality if and only if $\boldsymbol{w} = \boldsymbol{w}'$. Using this inequality, we can obtain a surrogate function for $L(\boldsymbol{w}) = \ell(\boldsymbol{w}) - \lambda\|\boldsymbol{w}\|_1$; after dropping all terms that are independent of $\boldsymbol{w}$, we now have to maximize

$$\boldsymbol{w}^T\Big(\mathbf{g}(\widehat{\boldsymbol{w}}^{(t)}) - \mathbf{B}\widehat{\boldsymbol{w}}^{(t)}\Big) + \frac{1}{2}\,\boldsymbol{w}^T(\mathbf{B} - \lambda\boldsymbol{\Lambda}^{(t)})\boldsymbol{w}, \tag{11}$$

where

$$\boldsymbol{\Lambda}^{(t)} = \mathrm{diag}\Big\{\Big|\widehat{w}_1^{(t)}\Big|^{-1}, \ldots, \Big|\widehat{w}_{d(m-1)}^{(t)}\Big|^{-1}\Big\}.$$

The update equation for the algorithm, obtained by maximizing (11) with respect to $\boldsymbol{w}$, is

$$\widehat{\boldsymbol{w}}^{(t+1)} = (\mathbf{B} - \lambda\,\boldsymbol{\Lambda}^{(t)})^{-1}\Big(\mathbf{B}\widehat{\boldsymbol{w}}^{(t)} - \mathbf{g}(\widehat{\boldsymbol{w}}^{(t)})\Big). \tag{12}$$

A numerically more convenient (but equivalent [11]) form is

$$\widehat{\boldsymbol{w}}^{(t+1)} = \boldsymbol{\Upsilon}^{(t)}\Big(\boldsymbol{\Upsilon}^{(t)}\mathbf{B}\boldsymbol{\Upsilon}^{(t)} - \lambda\mathbf{I}\Big)^{-1}\ \boldsymbol{\Upsilon}^{(t)}\Big(\mathbf{B}\widehat{\boldsymbol{w}}^{(t)} - \mathbf{g}(\widehat{\boldsymbol{w}}^{(t)})\Big), \tag{13}$$

where

$$\boldsymbol{\Upsilon}^{(t)} = \mathrm{diag}\Big\{\Big|\widehat{w}_1^{(t)}\Big|^{1/2}, \ldots, \Big|\widehat{w}_{d(m-1)}^{(t)}\Big|^{1/2}\Big\},$$

since it avoids inverse weight estimates, some of which are expected to go zero.

Unfortunately, the matrix to be inverted at each iteration is no longer constant and, thus, the inverse cannot be precomputed. Consequently, we are back to the computational cost of the original IRLS algorithm. Notice, however, that we can now perform exact MAP multinomial logistic regression under a Laplacian prior for the same cost as the original IRLS algorithm for ML estimation. Moreover, unlike with the RVM and SPR approaches, we are dealing with a concave objective function with a unique (global) maximum and, thus, careful initialization is not required.

3.5 Component-Wise Update Procedure

The algorithm derived in the previous section scales unfavorably with the dimension of $\boldsymbol{w}$. The matrix inversion at each iteration requires $O((dm)^3)$ operations and $O((dm)^2)$ storage. This is primarily problematic when d is very large, either because the original number of features is very large (as in many bioinformatics applications) or because we are performing kernel classification with a very large training set. We address this problem by proposing a component-wise update procedure that avoids matrix inversions and, thus, scales much more favorably. The key idea will be to take the surrogate function and maximize it only with respect to one of the components of $\boldsymbol{w}$, while holding the remaining components at their current values. Under the Laplacian prior, it turns out that the component-wise update equation has a closed form solution with no need for bounding the log-prior as we did in the previous section.

For any log-likelihood whose Hessian can be lower bounded by some matrix $\mathbf{B}$ together with a Laplacian prior, the function to be maximized at each iteration is

$$\boldsymbol{w}^T\Big(\mathbf{g}(\widehat{\boldsymbol{w}}^{(t)}) - \mathbf{B}\widehat{\boldsymbol{w}}^{(t)}\Big) + \frac{1}{2}\,\boldsymbol{w}^T\mathbf{B}\boldsymbol{w} - \lambda\|\boldsymbol{w}\|_1. \tag{14}$$

If we maximize (14) with respect to only one of the components of $\boldsymbol{w}$, say w_k, we obtain an update equation in which the value of $\widehat{w}_k^{(t+1)}$ is allowed to change from its value in the previous estimate $\widehat{w}_k^{(t)}$, but, for all $l \ne k$, $\widehat{w}_l^{(t+1)}$ is just the same as $\widehat{w}_l^{(t)}$. This update equation guarantees that the value of the bound function is nondecreasing at each step, which is a weaker condition than (4), but still sufficient to guarantee monotonicity of the resulting algorithm. After some manipulation, we have

$$\widehat{w}_k^{(t+1)} = \mathrm{soft}\left(\widehat{w}_k^{(t)} - \frac{g_k(\widehat{\boldsymbol{w}}^{(t)})}{B_{kk}}; \frac{-\lambda}{B_{kk}}\right), \tag{15}$$

where B_{ij} denotes the (i,j) element of matrix $\mathbf{B}$, $g_k(\widehat{\boldsymbol{w}}^{(t)})$ is the kth element of the gradient vector $\mathbf{g}(\widehat{\boldsymbol{w}}^{(t)})$, and

$$\mathrm{soft}(a;\delta) = \mathrm{sign}(a)\ \max\{0, |a| - \delta\}$$

is the *soft threshold* function, well-known in the wavelets literature [24]. The weight update equation (15) provides an explicit criterion for whether or not to include each basis function in the classifier, in a similar vein to the criterion derived in [36].

In the case of a multinomial logistic likelihood, $\mathbf{B}$ can be precomputed according to (8) so the computational cost of (15) is essentially that of computing one element of the gradient vector, $\mathrm{g}(\widehat{\boldsymbol{w}}^{(t)})$; this can be done with $O(n)$ cost according to (9), given $\boldsymbol{p}$. With careful programming, after changing only one component of $\boldsymbol{w}$, we can again obtain the required components of $\boldsymbol{p}$ for computing $g_k(\widehat{\boldsymbol{w}}^{(t)})$ also with $O(n)$ cost. Thus, the cost of implementing (15) for one basis function remains $O(n)$. A full sweep over all the $d(m-1)$ components of $\boldsymbol{w}$ has $O(ndm)$ cost; for large d, this can be orders of magnitude better than the $O((dm)^3)$ cost of the matrix inversion in (13). Another advantage of this approach is that its memory requirement is $O(d(m+n))$, which, depending on the relative sizes of d and n, could be even less than what is needed for storing the training data $\mathcal{D}$.

Of course, a component-wise update equation can also be derived under a Gaussian prior:

$$\widehat{w}_k^{(t+1)} = \frac{B_{kk}}{B_{kk}-\lambda}\left(\widehat{w}_k^{(t)} - \frac{g_k(\widehat{\boldsymbol{w}}^{(t)})}{B_{kk}}\right). \qquad (16)$$

Note that, in this case, no strong inclusion/exclusion criterion for basis functions remains, which is consistent with the fact that the Gaussian is not a sparsity-promoting prior. A related component-wise update algorithm was proposed in [42], which is only applicable in the case of a Gaussian prior (not a Laplacian one).

One issue that has to be addressed is the choice of component to update at each step. Because of the concavity of our objective function, even a simple-minded visitation schedule is guaranteed to converge to the global maximum, albeit perhaps not as rapidly as more sophisticated ones. In all the experiments reported in this paper, we update the weights in a simple cyclic fashion, with one small modification: When weight $\widehat{w}_k^{(t)}$ is zero, it is updated with a probability that decreases geometrically with the number of iterations, reflecting our decreasing need to reestimate weights for basis functions that are currently excluded.

Finally, we should point out that a component-wise algorithm has also been recently proposed for the RVM, with very good experimental results [36]. However, since the objective function for learning RVM classifiers is not concave, the component-wise update algorithm for the RVM is not guaranteed to converge to the global maximum and may critically depend on both the initialization and the visitation schedule.

4 Generalization Bounds

The ultimate performance measure of a binary classifier[2] $f : \mathbb{R}^d \rightarrow \{-1,+1\}$ is the true risk, or expected loss, with respect to the true joint probability distribution $P(\boldsymbol{x},y)$,

2. To simplify notation, this section will only focus on binary problems, with the two classes denoted as $\{-1,+1\}$.

$$R_{\mathrm{true}}[f] = E_{P(\boldsymbol{x},y)}[l(y,f(\boldsymbol{x}))], \qquad (17)$$

where $l : \{-1,+1\}^2 \rightarrow \mathbb{R}_0^+$ is the loss function; in particular, $l(y,f(\boldsymbol{x}))$ is the loss incurred by deciding $f(\boldsymbol{x})$ when the true label is y.

Of course, $R_{\mathrm{true}}[f]$ is not directly computable since we do not have access to $P(\boldsymbol{x},y)$, but only to a training set $\mathcal{D} = \{(\boldsymbol{x}_1,y_1),\ldots,(\boldsymbol{x}_n,y_n)\}$, assumed to have been generated in an i.i.d. way following $P(\boldsymbol{x},y)$. In the presence of such a training set, we can compute the empirical risk, which is the sample version of (17),

$$R_{\mathrm{sample}}[f,\mathcal{D}] = \frac{1}{n}\sum_{i=1}^{n} l(y_i, f(\boldsymbol{x}_i)). \qquad (18)$$

One of the key goals of learning theory is to obtain upper bounds on $R_{\mathrm{true}}[f]$ that hold uniformly for any $P(\boldsymbol{x},y)$. In the *probably approximately correct* (PAC) framework [37], such bounds are of the form

$$P(\mathcal{D}:\ R_{\mathrm{true}}[f] \geq \mathrm{bound}(f,\mathcal{D},\delta,n)) \leq \delta, \qquad (19)$$

where the probability is over the random draw of a training set $\mathcal{D}$ consisting of n i.i.d. samples from $P(\boldsymbol{x},y)$. In other words, for any δ chosen by the user, the true risk $R_{\mathrm{true}}[f]$ will be less than the bound term in (19) with probability $(1-\delta)$. However, as the probability δ becomes smaller, the bound becomes less tight; in the limit $\delta \rightarrow 0$, the error bound becomes trivial and uninformative, i.e., $\mathrm{bound}(f,\mathcal{D},\delta,n)$ becomes equal to or larger than one.

Vapnik-Chervonenkis (VC) theory is probably the best known approach to deriving bounds [38]; however, VC theory usually leads to very loose bounds unless one has extremely large training sets. Other more recent approaches include compression bounds [13] and minimum description length (MDL) bounds [38], as well as bounds based on Rademacher complexities [1], [26] and PAC-Bayesian bounds [25], [33]. It has often been noted that compression and MDL bounds are not directly applicable to sparse classification algorithms like the RVM (e.g., [15, p. 178]), so we focus our attention on Rademacher and PAC-Bayesian bounds.

It has been shown that sparsity alone does not guarantee good generalization performance, especially in algorithms that take sparsity to extremes [39], so to rigorously analyze the generalization performance of our SMLR algorithm, we derive two closely related upper bounds on the error rate of binary logistic classifiers with a Laplacian prior. The first of these bounds is based on the Rademacher complexity results of [1], [26], while the second uses the recent PAC-Bayesian formalism [25], [33]. In the next two sections, we introduce some notation and outline the strategy used in deriving the bounds. Subsequently, we shall state our results formally.

4.1 Three Kinds of Classifiers

Consider both a point estimate $\widehat{\boldsymbol{w}}$ of the weight vector $\boldsymbol{w}$ and any distribution $q(\boldsymbol{w})$ over the space of possible weight vectors. Given a data sample $\boldsymbol{x}$, we define three different kinds of classifiers for assigning a label $\widehat{y}$ to $\boldsymbol{x}$:

- *Point Classifier* (PC): This kind of classifier uses only the point estimate $\widehat{\boldsymbol{w}}$, outputing a class label according to the rule: $\widehat{y} = \mathrm{sign}(\widehat{\boldsymbol{w}}^T\boldsymbol{x}) \equiv f_{\mathrm{PC}}(\boldsymbol{x},\widehat{\boldsymbol{w}})$.

- *Bayes Voting Classifier* (BVC): This kind of classifier uses a voting scheme based on $q(\boldsymbol{w})$ and outputs a class label according to the rule:

$$\widehat{y} = \mathrm{sign}\big(E_{q(\boldsymbol{w})}[\mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x})]\big) \equiv f_{\mathrm{BVC}}(\boldsymbol{x}, q). \tag{20}$$

- *Gibbs Classifier* (GC): This kind of classifier draws a $\boldsymbol{w}$ from the distribution $q(\boldsymbol{w})$ and outputs a class label according to the rule: $\widehat{y} = \mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x}) \equiv f_{\mathrm{GC}}(\boldsymbol{x}, \boldsymbol{w})$. Note that, unlike the previous two cases, f_{GC} is a random variable because $\boldsymbol{w}$ here is also a random variable: When presented repeatedly with the same sample $\boldsymbol{x}$, the GC may produce different decisions.

4.2 Strategy for Obtaining the Bounds

The strategy for obtaining the error bounds on our classifiers is as follows:

1. The algorithms presented in this paper yield point classifiers based on MAP point estimates $\widehat{\boldsymbol{w}}$ of the weight vector $\boldsymbol{w}$; we are thus interested in the error rate of a PC. However, the Rademacher complexity formalism bounds the error rate of a BVC and the PAC-Bayesian approach bounds the error rate of a GC. Therefore, our first step is to relate the error rate of a PC to those of a BVC and a GC; this can be done by restricting our attention to symmetric distributions $q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w})$ centered on $\widehat{\boldsymbol{w}}$. While these results are already known (e.g., see [15]), we provide proofs for completeness.
2. We compute an analytical expression for the Kullback-Leibler divergence (KLD) between two Laplacian distributions because we will later derive two bounds, each of which depends on the KLD between our Laplacian prior $p(\boldsymbol{w}|\lambda)$ and any Laplacian distribution $q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w}|\eta)$ centered on $\widehat{\boldsymbol{w}}$.
3. The Rademacher and PAC-Bayesian theorems provide bounds on the error rates of a BVC and a GC, respectively, that depend on the KLD between the prior p and the distribution q. As originally stated, the theorems allow q to be an *arbitrary* distribution that may even depend on the observed data and that can be chosen so as to provide as tight a bound as possible. Because our prior $p(\boldsymbol{w}|\lambda)$ is Laplacian, we restrict our attention to Laplacian distributions $q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w}|\eta)$ so that we can use an analytical expression for the KLD. With this restriction in place, we derive modified statements of the bounds in both the Rademacher and PAC-Bayesian formalisms.
4. Since the error bounds for all distributions from this family hold simultaneously, we choose the tightest error bound from this family. Using the previously derived relationship between the error rate of a PC and the error rates of a BVC and a GC, we obtain two final error bounds for our PC.

4.3 Relationship between Error Rates of PC, BVC, and GC

The following lemma shows that the error rates of a PC $f_{\mathrm{PC}}(\boldsymbol{x}, \widehat{\boldsymbol{w}})$ and a BVC $f_{\mathrm{BVC}}(\boldsymbol{x}, q)$ are identical, provided that the BVC employs a symmetric distribution $q(\boldsymbol{w}) = q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w})$ that is centered on $\widehat{\boldsymbol{w}}$.

Lemma 1. *For any sample $\boldsymbol{x}$, the classification decision of a BVC*

$$f_{\mathrm{BVC}}(\boldsymbol{x}, q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w})) = \mathrm{sign}\left(\int \mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x}) q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w}) d\boldsymbol{w}\right) \tag{21}$$

is identical to that of a PC $f_{\mathrm{PC}}(\boldsymbol{x}, \widehat{\boldsymbol{w}}) = \mathrm{sign}(\widehat{\boldsymbol{w}}^T\boldsymbol{x})$ if $q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w})$ is a symmetric distribution with respect to $\widehat{\boldsymbol{w}}$, that is, if $q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w}) = q_{\widehat{\boldsymbol{w}}}(\widetilde{\boldsymbol{w}})$, where $\widetilde{\boldsymbol{w}} \equiv 2\widehat{\boldsymbol{w}} - \boldsymbol{w}$ is the symmetric reflection of $\boldsymbol{w}$ about $\widehat{\boldsymbol{w}}$.

Proof. For every $\boldsymbol{w}$ in the domain of the integral, $\widetilde{\boldsymbol{w}}$ is also in the domain of the integral. Since $\widetilde{\boldsymbol{w}} = 2\widehat{\boldsymbol{w}} - \boldsymbol{w}$, we have that $\boldsymbol{w}^T\boldsymbol{x} + \widetilde{\boldsymbol{w}}^T\boldsymbol{x} = 2\widehat{\boldsymbol{w}}^T\boldsymbol{x}$. Three cases have to be considered:

1. Case 1: When $\mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x}) = -\mathrm{sign}(\widetilde{\boldsymbol{w}}^T\boldsymbol{x})$ or $\boldsymbol{w}^T\boldsymbol{x} = \widetilde{\boldsymbol{w}}^T\boldsymbol{x} = 0$, the total contribution from $\boldsymbol{w}$ and $\widetilde{\boldsymbol{w}}$ to the integral is 0 since

$$\mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x}) q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w}) + \mathrm{sign}(\widetilde{\boldsymbol{w}}^T\boldsymbol{x}) q_{\widehat{\boldsymbol{w}}}(\widetilde{\boldsymbol{w}}) = 0.$$

2. Case 2: When $\mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x}) = \mathrm{sign}(\widetilde{\boldsymbol{w}}^T\boldsymbol{x})$, since $\boldsymbol{w}^T\boldsymbol{x} + \widetilde{\boldsymbol{w}}^T\boldsymbol{x} = 2\widehat{\boldsymbol{w}}^T\boldsymbol{x}$, it follows that

$$\mathrm{sign}(\widehat{\boldsymbol{w}}^T\boldsymbol{x}) = \mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x}) = \mathrm{sign}(\widetilde{\boldsymbol{w}}^T\boldsymbol{x}).$$

Thus,

$$\mathrm{sign}\Big[\mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x}) q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w}) + \mathrm{sign}(\widetilde{\boldsymbol{w}}^T\boldsymbol{x}) q_{\widehat{\boldsymbol{w}}}(\widetilde{\boldsymbol{w}})\Big] = \mathrm{sign}(\widehat{\boldsymbol{w}}^T\boldsymbol{x}).$$

3. Case 3: When $\boldsymbol{w}^T\boldsymbol{x} = 0, \widetilde{\boldsymbol{w}}^T\boldsymbol{x} \neq 0$ or $\boldsymbol{w}^T\boldsymbol{x} \neq 0, \widetilde{\boldsymbol{w}}^T\boldsymbol{x} = 0$, we have again from $\boldsymbol{w}^T\boldsymbol{x} + \widetilde{\boldsymbol{w}}^T\boldsymbol{x} = 2\widehat{\boldsymbol{w}}^T\boldsymbol{x}$ that

$$\mathrm{sign}\Big[\mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x}) q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w}) + \mathrm{sign}(\widetilde{\boldsymbol{w}}^T\boldsymbol{x}) q_{\widehat{\boldsymbol{w}}}(\widetilde{\boldsymbol{w}})\Big] = \mathrm{sign}(\widehat{\boldsymbol{w}}^T\boldsymbol{x}).$$

Unless $\widehat{\boldsymbol{w}} = 0$ (in which case the classifier is undefined), Case 2 or 3 will occur for at least some of the $\boldsymbol{w}$ in the domain of the integral. Hence, the lemma is true. □

The next lemma relates the error rates of a BVC and a GC:

Lemma 2. *For any labeled sample $(\boldsymbol{x}, y)$, the error rate of a BVC using any distribution $q(\boldsymbol{w})$ is no more than twice the expected error rate of a GC using the same distribution.*

Proof. Let us first consider the expected probability that the Gibbs classifier will predict the class of the sample $\boldsymbol{x}$ to be $+1$:

$$\begin{aligned} P(f_{\mathrm{GC}} = +1) &= \int \left(\frac{1 + \mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x})}{2}\right) q(\boldsymbol{w}) d\boldsymbol{w} \\ &= \frac{1}{2} + \frac{1}{2}\int \mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x}) q(\boldsymbol{w}) d\boldsymbol{w}. \end{aligned} \tag{22}$$

Clearly, if the BVC prediction for the label is $+1$, we know that $\int \mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x}) q(\boldsymbol{w}) d\boldsymbol{w} > 0$ and it follows that $P(f_{\mathrm{GC}} = +1) > \frac{1}{2}$. Conversely, if the BVC prediction is -1, then $P(f_{\mathrm{GC}} = -1) > \frac{1}{2}$. Therefore, when the BVC makes a mistake in predicting the label y of any sample $\boldsymbol{x}$, it must be true that the expected probability of the GC making a mistake on that sample is at least $\frac{1}{2}$. Hence, for any set of test samples, the error rate of a BVC is no more than twice the expected error rate of a GC. □

By combining Lemma 1 and Lemma 2, we conclude that the error rate of a PC $\widehat{\boldsymbol{w}}$ is upper bounded by twice the expected error rate of a GC based on a symmetric distribution $q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w})$ centered on $\widehat{\boldsymbol{w}}$.

4.4 Kullback-Leibler Divergence between Two Laplacians

A quantity that will be needed below is the Kullback-Leibler divergence (KLD) between two Laplacian densities. Consider our Laplacian prior $p(\boldsymbol{w}|\lambda) \propto \exp(-\lambda\|\boldsymbol{w}\|_1)$ and another distribution $q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w}|\eta) \propto \exp(-\sum_i \eta_i\, |w_i - \widehat{w}_i|)$ centered on $\widehat{\boldsymbol{w}}$. Using the independence of the elements of $\boldsymbol{w}$ in both these densities, it is easy to show that

$$D(q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w}|\boldsymbol{\eta})\|p(\boldsymbol{w}|\lambda)) = \sum_i \left[\ln\frac{\eta_i}{e\,\lambda} + \frac{\lambda\, e^{-\eta_i|\widehat{w}_i|}}{\eta_i} + \lambda|\widehat{w}_i| \right]. \quad (23)$$

As explained below, in order to tighten the bound we will soon derive, we may look for the $\boldsymbol{\eta}$ that minimizes the KLD in (23), given $\widehat{\boldsymbol{w}}$ and λ:

$$D_{\min}(q \parallel p) = \min_{\eta} D(q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w}|\boldsymbol{\eta})\|p(\boldsymbol{w}|\lambda)). \quad (24)$$

A few observations show that this optimization task is simple: First, optimization can be performed separately with respect to each η_i because (23) is a sum of functions, each depending on only one η_i; second, for those $\widehat{w}_i$ that are zero (usually many due to the presence of the sparsity-promoting prior), the solution is simply $\eta_i = \lambda$; finally, for those $\widehat{w}_i$ that are not zero, the solution can be found by numerically solving a one-dimensional optimization problem with a unique minimum with respect to each η_i.

An even simpler approach is to use the looser bound $D_{\min}(q \parallel p) \leq \lambda\|\widehat{\boldsymbol{w}}\|_1$, which is satisfied with equality if and only if $\widehat{\boldsymbol{w}} = 0$. To prove that this bound holds, consider $\eta_i = \lambda$, for all i; in this case,

$$\begin{aligned} D(q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w}|\boldsymbol{\eta} = \lambda\mathbf{1})\|p(\boldsymbol{w}|\lambda)) &= \sum_{i:\widehat{w}_i \neq 0} \left(e^{-\lambda|\widehat{w}_i|} + \lambda|\widehat{w}_i| - 1 \right) \\ &\leq \lambda\|\widehat{\boldsymbol{w}}\|_1, \end{aligned}$$

where the last inequality results from the fact that $\exp\{-\lambda|\widehat{w}_i|\} \leq 1$ since $\lambda|\widehat{w}_i| \geq 0$ and $\sum_{i:\widehat{w}_i \neq 0} |\widehat{w}_i| = \|\widehat{\boldsymbol{w}}\|_1$. Thus, we can easily ensure that $D_{\min}(q \parallel p)$ is no larger than $\lambda\|\widehat{\boldsymbol{w}}\|_1$, although we can often choose $\boldsymbol{\eta}$ to make it even smaller.

4.5 Rademacher Complexity Bound

Let us adopt a loss that depends on the margin $y\boldsymbol{w}^T\boldsymbol{x}$: $l(y, \boldsymbol{w}^T\boldsymbol{x}) = l_s(y\boldsymbol{w}^T\boldsymbol{x})$, where $l_s(a) = \min(1, \max(0, 1 - a/s))$. Function $l_s(\cdot)$ is $(1/s)$-Lipschitz and dominates pointwise the zero-one loss $l_{0-1}(y, \boldsymbol{w}^T\boldsymbol{x}) = h(-y\boldsymbol{w}^T\boldsymbol{x})$, where $h(\cdot)$ is Heavyside's function. Since our loss is bounded above by 1, we quote a slightly refined version of a result of Meir and Zhang [26]. We shall subsequently compute the bound for our algorithm using (23) in conjunction with this result.

Theorem 1. *Let $(\boldsymbol{x}, y) \in \mathbb{R}^d \times \{-1, +1\}$ be a random variable with distribution $P(\boldsymbol{x}, y)$ and let the (random) training set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$, be a set of n samples drawn i.i.d. from $P(\boldsymbol{x}, y)$. Let*

$$f_{\mathrm{SBVC}}(\boldsymbol{x}, q) \equiv E_{q(\boldsymbol{w})}[\mathrm{sign}(\boldsymbol{w}^T\boldsymbol{x})]$$

denote the (real) value computed by a BVC before thresholding (compare with (20)), which will be called the soft Bayes voting classifier (SBVC). Let $g > 0$ and $r > 0$ be arbitrary parameters. Then, with probability at least $(1 - \delta)$ over draws of training sets, the following bound holds:

$$\begin{aligned} P(yf_{\mathrm{SBVC}}(\boldsymbol{x},q) < 0) &\leq R_{\mathrm{sample}}[f_{\mathrm{SBVC}}, \mathcal{D}] \\ &\quad + \frac{2}{s}\sqrt{\frac{2\tilde{g}(q)}{n}} + \sqrt{\frac{\ln\log_r\frac{r\tilde{g}(q)}{g} + \frac{1}{2}\ln\frac{1}{\delta}}{n}}, \end{aligned}$$

where $R_{\mathrm{sample}} f_{\mathrm{SBVC}}, \mathcal{D}$ is the sample mean of the loss of the SBVC,

$$R_{\mathrm{sample}}[f_{\mathrm{SBVC}}, \mathcal{D}] = \frac{1}{n}\sum_{i=1}^{n} l_s(y_i f_{\mathrm{SBVC}}(\mathbf{x}_i, q)),$$

and $\tilde{g}(q) = r\max(D(q \parallel p), g)$.

Notice that $R_{\mathrm{sample}}[f_{\mathrm{SBVC}}, \mathcal{D}]$ is computed on the SBVC rather than on the BVC. Thus, although the BVC and PC agree, the sample losses of the PC and SBVC, as defined above, may differ slightly. For this reason, we compute sample loss $R_{\mathrm{sample}}[f_{\mathrm{SBVC}}, \mathcal{D}]$ on the training set using Monte Carlo approximations of $f_{\mathrm{SBVC}}(\boldsymbol{x}, q)$. Plugging the KLD between two Laplacians given in (23) into the statement of the theorem gives us a bound on the error rate of the point classifier $\widehat{\boldsymbol{w}}$ learned using the prior with parameter λ.

4.6 PAC-Bayesian Bound

Our second result uses a PAC-Bayesian bound on the error rate of a GC [25], [33]:

Theorem 2. *Let $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_n, y_n)\}$ be as in Theorem 1. Then, with probability at least $(1 - \delta)$ over draws of training sets, the following bound holds:*

$$D_{\mathrm{Ber}}(R_{\mathrm{true}} \parallel R_{\mathrm{sample}}) \leq \frac{D(q \parallel p) + \ln\left(\frac{n+1}{\delta}\right)}{n}, \quad (25)$$

where R_{true} is the true expected error rate of a GC based on $q(\boldsymbol{w})$,

$$R_{\mathrm{true}} = E_{q(\boldsymbol{w})}\left[E_{P(\boldsymbol{x},y)}[l(y, f_{\mathrm{GC}}(\boldsymbol{x}, \boldsymbol{w}))]\right],$$

R_{sample} is the expected error rate of the same classifier on the training set,

$$R_{\mathrm{sample}} = E_{q(\boldsymbol{w})}\left[\frac{1}{n}\sum_{i=1}^{n} l(y_i, f_{\mathrm{GC}}(\boldsymbol{x}_i, \boldsymbol{w}))\right],$$

$D_{\mathrm{Ber}}(a \parallel b)$ denotes the KLD between two Bernoulli distributions of probabilities a and b,

$$D_{\mathrm{Ber}}(a \parallel b) \equiv a\ln\frac{a}{b} + (1 - a)\ln\frac{(1 - a)}{(1 - b)},$$

and $D(q \parallel p)$ denotes the KLD between the two densities q and p.

For a given R_{sample} value, $D_{\text{Ber}}(R_{\text{true}} \| R_{\text{sample}})$ is a convex function of R_{true}. Thus, (25) implies an upper bound on R_{true} that is easy to obtain numerically. Note that Theorem 2 bounds the expected error rate of a GC. However, we have shown in Lemmas 1 and 2 that the expected error rate of a PC $\widehat{\boldsymbol{w}}$ is less than twice that of a GC based on a symmetric $q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w})$ centered on $\widehat{\boldsymbol{w}}$. So, if we choose q to be Laplacian and, as with the Rademacher bound, plug the KLD between two Laplacians given in (23) into the statement of the theorem, we get a bound on the error rate of the point classifier $\widehat{\boldsymbol{w}}$ learned using the prior with parameter λ. In practice, the error rate of a PC has empirically been observed to be no larger than that of the corresponding GC (and is often much smaller) so the factor of two is probably not required (though this has not yet been proven).

To compute R_{sample} on the training set, we could again use Monte Carlo. However, since $\boldsymbol{w}^T\boldsymbol{x}$ is the weighted sum of a number of Laplacian random variables w_i, as the dimensionality of $\boldsymbol{w}$ (i.e., $d(m-1)$) increases, the distribution of $\boldsymbol{w}^T\boldsymbol{x}$ approaches a Gaussian, allowing analytical computations. We have observed experimentally that even $d(m-1) \geq 10$ is sufficient for the Gaussian approximation to be indistinguishable from the Monte Carlo results.

4.7 Tightening the Bounds

Having chosen a Laplacian distribution centered on our MAP classifier, we can easily evaluate either of these bounds. Since the bounds are simultaneously applicable for any choice of Laplacian distribution $q_{\widehat{\boldsymbol{w}}}(\boldsymbol{w}|\boldsymbol{\eta})$ centered on $\widehat{\boldsymbol{w}}$, we choose $\boldsymbol{\eta}$ to minimize the KLD to the prior $p(\boldsymbol{w}|\lambda)$, as shown in (24), so as to obtain the tightest possible bound.

4.8 Discussion

Since $D_{\min}(q \| p) \leq \lambda \|\widehat{\boldsymbol{w}}\|_1$, these bounds show analytically that the generalization capacity of the classifier is a function of both the regularization parameter λ and the sparsity of $\widehat{\boldsymbol{w}}$. This matches our intuition regarding the nature of the bounds: If we make a strong bet on sparsity by choosing a large λ and, in return, obtain a nonsparse $\widehat{\boldsymbol{w}}$, the bounds will be loose; in contrast, if we make only a mild bet on sparsity and, in return, obtain a sparse $\widehat{\boldsymbol{w}}$, the bound will be tight.

Unfortunately, the expected error rate of a GC on the training set can be much higher than that of a PC. Thus, the bound in Theorem 2 is made much weaker by its use of a GC.[3] Nevertheless, since the bound is on the KLD of Bernoulli random variables, it *always* yields a nontrivial (less than 1) bound on the expected error rate. This is in contrast with bounds based on VC theory which tend to be trivial except when the data sets are enormous.

4.9 Relationship to Previous PAC-Bayesian Error Bounds

The PAC-Bayesian formalism has been utilized to provide upper bounds on the generalization error of other classification algorithms. Margin bounds due to Langford and Shawe-Taylor [20] have been used to justify the SVM classifier; the application of the PAC-Bayesian theorem in our paper is similar to their derivation. In another closely related paper, Seeger [33] provides PAC-Bayesian generalization bounds for Gaussian process classification and mentions that his approach can be extended to provide error bounds on the RVM. While our derivation is similar to these results (especially [20]), we specifically focus on bounds for our own SMLR algorithm, both to analyze its generalization performance and to justify our choice of a Laplacian prior.

3. This limitation is not present in our first bound, which does not depend on GC error rates.

5 Experimental Results

We used a collection of benchmark data sets to evaluate the accuracy, sparsity, and computational efficiency of our classifier design algorithms. Being especially interested in classification problems with large numbers of features or training samples, the data sets were selected to vary widely in the number of originally observed features, the training set size n, and the number of classes m. We implemented our own MATLAB code for the RVM (based on the original block-wise algorithm of [35]), as well as for the proposed algorithms.[4] For the SVM, we adopted the widely used SVM-light program.[5]

Table 1 describes the data sets[6] and the methods used to assess classification performance (either k-fold cross-validation or fixed training/test sets); in each case, the method is chosen to be consistent with what has already been reported in the literature so as to facilitate comparison. The *Crabs*, *Iris*, and *Forensic Glass* data sets are well-known. *AML/ALL* and *Colon* are standard gene expression data sets for cancer classification and provide a good test of accuracy, sparsity, and efficiency in the context of high feature dimensionality. With *Colon*, most authors have randomly split the 62 samples into a training set of size 50 and a test set of size 12; to enable comparison with previous results, we adopt the same procedure and repeat this procedure 30 times. *Yeast* is another gene expression data set with 208 samples of 79 features in five classes; we adopt the 8-fold cross-validation method used by other authors with this data set. *Mines* contains 22,933 samples of sonar data for detecting underwater mines, only 119 of which are positive samples; we use *Mines* to test the tightness of our generalization bounds for large n.

We evaluated four different classifiers on six of the seven benchmark data sets in terms of accuracy, sparsity, and efficiency. The first three algorithms were the SVM, the RVM, and our proposed *sparse multinomial logistic regression* (SMLR), which uses a sparsity-promoting Laplacian prior (l_1-penalty). To tease out the effect of this prior, we also evaluated the *ridge multinomial logistic regression* (RMLR), which uses a Gaussian prior (l_2-penalty), as in ridge regression, thus not promoting sparsity.

4. The MATLAB code for RVM, SMLR, and RMLR may be obtained for noncommercial use from the first author.

5. Available from svmlight.joachims.org.

6. Data set sources: Crabs and Forensic Glass from www.stats.ox.ac.uk/pub/PRNN/. Iris from www.ics.uci.edu/~mlearn/MLSummary.html. AML/ALL from www-genome.wi.mit.edu/mpr/table_AML_ALL_samples.rtf. Colon from microarray.princeton.edu/oncology/affydata/index.html. Yeast from www.cse.ucsc.edu/research/compbio/genex/genex.html. Mines is not publicly available due to its sensitive nature; for this reason, we use it only to illustrate the tightness of our error bound on a large data set.

TABLE 1
Characteristics of the Several Benchmark Data Sets Used in Our Experiments

	Crabs	Iris	FGlass	AML/ALL	Colon	Yeast	Mines
Number of classes (m)	2	3	6	2	2	5	2
Number of features	5	4	9	7129	2000	79	106
Total number of samples ($n+N$)	200	150	214	72	62	208	22933
Number of training samples (n)	80	135	193	38	50	182	11467
Number of test samples (N)	120	15	21	34	12	26	11466
k-fold cross-validation	—	10	10	—	30†	8	—

†In this case, we mean not 30-fold cross-validation, but 30 different random 50/12 training/test splits.

TABLE 2
Total Number of Classification Errors and Average Number of Retained Kernel Basis Functions (in Parentheses, Rounded to the Nearest Integer) for Various Classifiers on Six of the Seven Benchmark Data Sets Described in Table 1

	Crabs		Iris		FGlass		AML/ALL		Colon		Yeast	
	RBF		RBF		RBF		linear		linear		linear	
SVM	2	(53)	5	(124)	62	(365)	3	(31)	75	(32)	7	(155)
RVM	0	(4)	10	(37)	61	(74)	5	(3)	84	(6)	12	(49)
SMLR (l_1)	0	(10)	1	(136)	50	(901)	2	(10)	75	(15)	3	(83)
RMLR (l_2)	1	(80)	8	(270)	59	(965)	2	(38)	84	(50)	9	(728)
out of	120	(80)	150	(270)	214	(965)	34	(38)	360	(50)	208	(728)

The second row indicates the kernel that was used in each test; the last row indicates the maximum possible value attainable in each column.

Because the SVM is intrinsically limited to be a kernel classifier, we restricted our attention to kernel basis functions with the RVM, SMLR, and RMLR; the specific form of the kernel in each case was chosen to coincide with previously chosen kernels, so as to allow comparison with other published results. Kernel parameters were selected by cross-validation for the SVM; the best parameters for the SVM were then used for all the other methods. The regularization parameter for each method was chosen by cross-validation on the training samples.

Since the standard SVM and RVM are formulated for binary problems, the "1-versus-all" approach was used for multiclass problems with these algorithms. Table 2 indicates which type of kernel was used for each problem and summarizes the results obtained by these four methods.

In terms of accuracy, we see that SMLR outperforms the other three methods on all six data sets (for *Crabs*, the RVM exhibits the same accuracy; for *Colon*, the SVM exhibits the same accuracy). The difference is largest on data sets that are difficult to separate, like *Forensic Glass*.

In terms of sparsity, we first observe that SMLR yields much sparser classifiers than RMLR, although the difference is negligible with *Forensic Glass*. In particular, RMLR never prunes any of the kernel basis functions. We also notice that SMLR classifiers are usually sparser than SVMs, although *Forensic Glass* again represents an exception. We believe this is due to the fact that our method encourages sparsity only when it helps the classifier; in cases like *Forensic Glass*, in which the classification performance can be improved by retaining more basis functions, our method does so. Note that the included basis functions are still penalized, so this is not a case of simple overfitting. Finally, we see that the RVM consistently keeps the fewest number of basis functions, even when this is to its detriment; this seems to demonstrate a systematic under-fitting.

In terms of running time, it is difficult to provide precise values because RVM, SMLR, and RMLR were implemented using nonoptimized MATLAB code, while the SVM-light implementation of the SVM uses highly optimized C code. Nevertheless, we will make three observations. First, despite the implementation difference, the running time of SMLR on these data sets was generally around twice that of the optimized SVM, with two notable exceptions: 1) On problems that involve repeated reestimation (like *Colon*, with 30 random training/test splits), the convergence of our method is sped up by initialization with the previous MAP estimate,[7] for example, our method was twice as fast as SVM-light on the *Colon* data set, and 2) on multiclass problems, the two running times are not directly comparable because our method produces a true multiclass classifier rather than a series of "1-versus-all" binary classifiers.

Second, while we do not present results here, we also tested general purpose optimization algorithms—specifically, conjugate-gradient (CG) and Broyden-Fletcher-Goldfarb-Shanno (BFGS) [18]—for maximizing our SMLR objective function. Since our objective function includes a

7. For kernel classifiers, it may be noted that the basis functions change during cross-validation. In this case, we initialize weights associated with the basis functions that are common to the previous iteration to the earlier MAP estimate and initialize the remaining (new) weights to zero. Since our objective function is concave, the MAP classifier learned by our algorithm is independent of initialization. Thus, this choice of initialization does not affect the reported statistics concerning the accuracy of our algorithm.

nondifferentiable term $\|\widehat{\boldsymbol{w}}\|_1$, we used the following smooth approximation:

$$|w_l| \approx \sqrt{w_l^2 + 10^{-10}}.$$

Even with this smooth approximation, these methods exhibited running times orders of magnitude longer than those we observe with the proposed algorithms. While preconditioning might improve the convergence rate of CG, it is not the principal focus of the paper, so we have not pursued this further.

Third, whether we choose to use component-wise updating or not depends somewhat on the specific values of d, m, and n. In particular, if $n > (dm)^2$, it may not be worth using the component-wise update algorithm; this was true for *Crabs* and *Mines*, so we used the non-component-wise bound optimization algorithm of Section 3.4 and adopted an adaptive over-relaxation technique [32]. The over-relaxation approach effectively speeds up the block update algorithm and makes it comparable to quasi-Newton methods.

In terms of feature selection, because our approach can be formulated directly on the features themselves (without a kernel), we can use the sparsity-promoting nature of SMLR to perform automatic feature selection. This is especially useful when the number of features is large, as is the case with the *AML/ALL* and *Colon* data sets. For instance, using SMLR directly on the features of *AML/ALL*, we obtain only 1 error on the independent test set, using only 81 of the 7,129 original features. In contrast, when we repeat the test with RMLR, we get three errors and retain all 7,129 features. Using SMLR directly on the features of *Colon*, we obtain an average of 2.5 errors on the 30 test sets, using an average of 15 of the 2,000 original features.

In terms of generalization bounds, let us recall that the bounds we presented are always less than 1, in contrast with VC-based bounds, which are typically much greater than 1 unless n is extremely large. While nontrivial, our bounds are not tight for small n; for instance, the generalization bound on the linear classifier (no kernels) for the *AML/ALL* data set is $26/34$. However, they become tight as n grows; the plot in Fig. 1 suggests that the bound for the large *Mines* data set may even be tight enough to select the value of the prior parameter λ in place of cross-validation.

6 Discussion

6.1 Generalization Bounds

As shown in Fig. 1, for large samples, the generalization bounds we present may be helpful in choosing the regularization parameter for our algorithms. Nevertheless, we would like to emphasize that these bounds (just like most other learning-theoretical bounds) should be considered with caution. Similar in nature to the margin bounds that are frequently used to justify the SVM, the bounds we derive show that, if a small training set error is obtained, then a small $\|\widehat{\boldsymbol{w}}\|_1$ is *sufficient* to provide good generalization guarantees (via an upper bound on the error rate). However, if $\|\widehat{\boldsymbol{w}}\|_1$ is large, the bounds derived in this paper become loose and, hence, no longer provide reliable estimates of generalization performance. Thus, just like

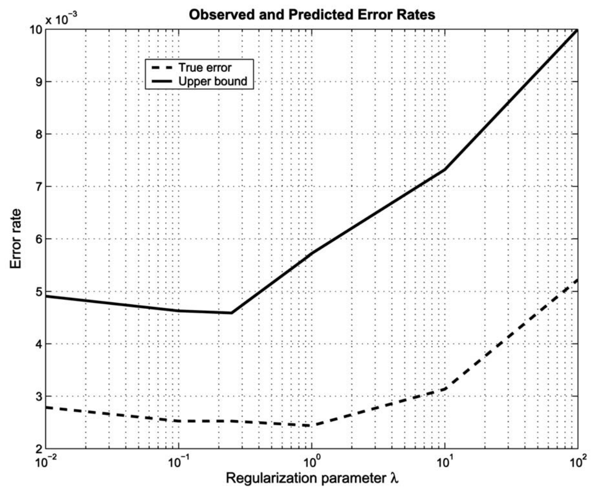

Fig. 1. Plot of the test error rate and generalization error bound for the *Mines* data set as a function of the prior parameter λ.

the margin bounds, the results we derive do not provide *necessary* conditions for good generalization: Even if $\|\widehat{\boldsymbol{w}}\|_1$ is large (or, in the case of the SVM, if the margin separating the classes is small), good generalization may be possible. In the past, margin bounds have been criticized on these grounds (see [14] for an elegant example); similar criticisms remain valid about our bounds also.

While they are theoretically interesting, our bounds provide only partial justification to our algorithm; we have chosen to present the bounds because they share the same properties and drawbacks as the bounds used to justify the SVM but are significantly tighter. One important, but difficult, direction for future research is to develop bounds that remain tight even when $\|\widehat{\boldsymbol{w}}\|_1$ becomes large.

6.2 How Much Sparsity? Cross-Validation versus ARD

It may be noted from the experimental results in Section 5 that we choose our regularization parameter λ based on a frequentist cross-validation approach. While Bayesian *automatic relevance determination* (ARD) methods have been advocated for sparse classification algorithms (most notably, for the RVM), in our experience, this often results in systematic under-fitting. More precisely, using ARD methods, too few basis functions are retained in the learned classifier, even when this proves detrimental to classification accuracy. For instance, in our experimental results, the RVM consistently chose the fewest number of basis functions, but, because SMLR tuned its level of sparsity by cross-validation, it usually achieved somewhat superior generalization. In recent work, this observation about the RVM has been studied systematically and addressed rather elegantly via approximate cross-validation methods [31].

6.3 MAP versus Fully Bayesian Classification

Our algorithm learns the maximum a posteriori classifier, which is only a point estimate from the posterior. Thus, in comparison to fully Bayesian classification, our algorithms might (in principle) provide less accurate estimates of the posterior class membership probabilities for new test samples; further, like other MAP classifiers, our algorithm does not provide error bars on the classification probabilities.

Exact fully Bayesian classification, although simple in principle, cannot be performed analytically because it involves an intractable expectation. This expectation can be approximated by resorting to Markov chain Monte Carlo sampling algorithms. However, in practice, one often prefers approximate strategies, such as the type-II maximum likelihood approach adopted in the RVM [35] or other approximate methods such as those adopted in Gaussian process classifiers [6] or the IVM [22]. Although these methods provide error bars (which our method doesn't and which may be important in some situations), the accuracy of our method is superior to the SVM and RVM (though perhaps not significantly so). Indeed, the widely used SVM algorithm also yields a point classifier and turns out to be competitive with these approximate Bayesian methods.

6.4 Visitation Schedule for Component-Wise Updating

We have not made a large effort to optimize the visitation schedule in our component-wise update procedure and smarter choices may strongly improve its convergence rate. As one preliminary modification, we have developed an adaptive visitation schedule that maximizes the increase in log-likelihood resulting from reestimating a single weight estimate. While computing the increase in log-likelihood is efficient, this computation must be carried out for *each* basis function in order to select a *single* basis function to be updated. This ends up being of comparable cost to the simple approach of cycling through the whole basis set presented earlier. Hence, we do not report results on the adaptive visitation schedule except to mention that it may prove to be advantageous when we have extremely large basis sets (of the order of tens of thousands of basis functions). Such a situation may prevent us from running our algorithm until convergence and we may be interested in stopping the algorithms after a fixed number of iterations. We believe that the development of good heuristics for the visitation schedule is an important direction for future research.

6.5 Final Points

First, we note that the objective function we optimize while learning an SMLR classifier is concave. This is not the case for the RVM or methods using Jeffreys priors [10], [11]. This concavity has significant benefits for identification of unique maxima, for efficient computational implementation, and for derivation of useful generalization bounds.

Second, many other sparse classification algorithms (including the RVM and Gaussian processes) can also be formulated for multiclass problems, but we expect our computational cost to scale more favorably.

Third, the component-wise update procedure we described provides a natural mechanism for determining the inclusion and exclusion of basis functions; in the context of SMLR, the intuition behind traditional forward-backward feature selection heuristics is placed on a rigorous theoretical footing.

Fourth, although we restricted our attention to kernel basis functions and selected our kernel parameters to optimize the performance of the SVM, we observe that neither the SVM nor the RVM outperformed SMLR on any of the benchmark data sets we examined; the statistical significance of any difference remains uncertain, however.

Acknowledgments

This work was partially funded by a US Defense Advanced Research Projects Agency multidisciplinary university research initiative (MURI) on adaptive multimodality sensing. A. Hartemink would like to gratefully acknowledge the support of the Alfred P. Sloan Foundation and the US National Science Foundation under its CAREER award program. M. Figueiredo acknowledges support of the Portuguese *Fundação para a Ciência e Tecnologia*. The authors also wish to thank the anonymous reviewers for several helpful comments and suggestions.

References

[1] P. Bartlett and S. Mendelson, "Rademacher and Gaussian Complexities: Risk Bounds and Structural Results," *J. Machine Learning Research,* vol. 3, pp. 463-482, 2002.
[2] D. Böhning, "Multinomial Logistic Regression Algorithm," *Annals of the Inst. of Statistical Math.,* vol. 44, pp. 197-200, 1992.
[3] D. Böhning and B. Lindsay, "Monotonicity of Quadratic-Approximation Algorithms," *Annals of the Inst. of Statistical Math.,* vol. 40, pp. 641-663, 1988.
[4] S. Chen, D. Donoho, and M. Saunders, "Atomic Decomposition by Basis Pursuit," *SIAM J. Scientific Computation,* vol. 20, pp. 33-61, 1998.
[5] M. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines.* Cambridge, U.K.: Cambridge Univ. Press, 2000.
[6] L. Csato and M. Opper, "Sparse Online Gaussian Processes," *Neural Computation,* vol. 14, no. 3, pp. 641-668, 2002.
[7] J. de Leeuw and G. Michailides, "Block Relaxation Methods in Statistics," technical report, Dept. of Statistics, Univ. of California at Los Angeles, 1993.
[8] A. Dempster, N. Laird, and D. Rubin, "Maximum Likelihood Estimation from Incomplete Data via the EM Algorithm," *J. Royal Statistical Soc. B,* vol. 39, pp. 1-38, 1977.
[9] D. Donoho and M. Elad, "Optimally Sparse Representations in General Nonorthogonal Dictionaries by l_1 Minimization," *Proc. Nat'l Academy of Science,* vol. 100, no. 5, pp. 2197-2202, 2003.
[10] M. Figueiredo and A. Jain, "Bayesian Learning of Sparse Classifiers," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* vol. 1, pp. 35-41, 2001.
[11] M. Figueiredo, "Adaptive Sparseness for Supervised Learning," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, pp. 1150-1159, 2003.
[12] J. Friedman, T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu, "Discussion of Boosting Papers," *The Annals of Statistics,* vol. 32, no. 1, pp. 102-107, 2004.
[13] T. Graepel, R. Herbrich, and J. Shawe-Taylor, "Generalisation Error Bounds for Sparse Linear Classifiers," *Proc. Conf. Computational Learning Theory,* pp. 298-303, 2000.
[14] T. Graepel, R. Herbrich, and R.C. Williamson, "From Margin to Sparsity," *Proc. Neural Information Processing Systems (NIPS) 13,* pp. 210-216, 2001.
[15] R. Herbrich, *Learning Kernel Classifiers: Theory and Algorithms.* Cambridge, Mass.: MIT Press, 2002.
[16] B. Krishnapuram, L. Carin, and A. Hartemink, "Joint Classifier and Feature Optimization for Cancer Diagnosis Using Gene Expression Data," *Proc. Int'l Conf. Research in Computational Molecular Biology,* 2003.
[17] B. Krishnapuram, A. Hartemink, L. Carin, and M. Figueiredo, "A Bayesian Approach to Joint Feature Selection and Classifier Design," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, pp. 1105-1111, 2004.
[18] K. Lange, *Optimization.* New York: Springer Verlag, 2004.

[19] K. Lange, D. Hunter, and I. Yang, "Optimization Transfer Using Surrogate Objective Functions," *J. Computational and Graphical Statistics,* vol. 9, pp. 1-59, 2000.
[20] J. Langford and J. Shawe-Taylor, "PAC-Bayes and Margins," *Advances in Neural Information Processing Systems 15,* S. Becker, S. Thrun, and K. Obermayer, eds., pp. 423-430, Cambridge, Mass.: MIT Press, 2003.
[21] J. Langford, "Practical Prediction Theory for Classification," *Proc. Int'l Conf. Machine Learning,* T. Fawcett and N. Mishra, eds., 2003.
[22] N.D. Lawrence, M. Seeger, and R. Herbrich, "Fast Sparse Gaussian Process Methods: The Informative Vector Machine," *Advances in Neural Information Processing Systems 15,* S. Becker, S. Thrun and K. Obermayer, eds., pp. 609-616, Cambridge, Mass.: MIT Press, 2003.
[23] M. Lewicki and T. Sejnowski, "Learning Overcomplete Representations," *Neural Computation,* vol. 12, pp. 337-365, 2000.
[24] S. Mallat, *A Wavelet Tour of Signal Processing.* San Diego, Calif.: Academic Press, 1998.
[25] D. McAllester, "Some PAC-Bayesian Theorems," *Machine Learning,* vol. 37, pp. 355-363, 1999.
[26] R. Meir and T. Zhang, "Generalization Error Bounds for Bayesian Mixture Algorithms," *J. Machine Learning Research,* vol. 4, pp. 839-860, 2003.
[27] T. Minka, "A Comparison of Numerical Optimizers for Logistic Regression," technical report, Dept. of Statistics, Carnegie Mellon Univ., 2003.
[28] R. Neal, *Bayesian Learning for Neural Networks.* New York: Springer Verlag, 1996.
[29] A.Y. Ng, "Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance," *Proc. Int'l Conf. Machine Learning,* 2004.
[30] B. Olshausen and D. Field, "Emergence of Simple-Cell Receptive Field Properties by Learning a Sparse Code for Natural Images," *Nature,* vol. 381, pp. 607-609, 1996.
[31] Y. Qi, T.P. Minka, R.W. Picard, and Z. Ghahramani, "Predictive Automatic Relevance Determination by Expectation Propagation," *Proc. Int'l Conf. Machine Learning,* 2004.
[32] R. Salakhutdinov and S. Roweis, "Adaptive Overrelaxed Bound Optimization Methods," *Proc. Int'l Conf. Machine Learning,* pp. 664-671, 2003.
[33] M. Seeger, "PAC-Bayesian Generalization Error Bounds for Gaussian Process Classification," *J. Machine Learning Research,* vol. 3, pp. 233-269, 2002.
[34] R. Tibshirani, "Regression Shrinkage and Selection via the LASSO," *J. Royal Statistical Soc. B,* vol. 58, no. 1, pp. 267-288, 1996.
[35] M. Tipping, "Sparse Bayesian Learning and the Relevance Vector Machine," *J. Machine Learning Research,* vol. 1, pp. 211-244, 2001.
[36] M. Tipping and A. Faul, "Fast Marginal Likelihood Maximisation for Sparse Bayesian Models," *Proc. Ninth Int'l Workshop Artificial Intelligence and Statistics,* C. Bishop and B. Frey, eds., 2003.
[37] L. Valiant, "A Theory of the Learnable," *Comm. ACM,* vol. 27, pp. 1134-1142, 1984.
[38] V. Vapnik, *Statistical Learning Theory.* New York: John Wiley, 1998.
[39] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping, "Use of the Zero-Norm with Linear Models and Kernel Methods," *J. Machine Learning Research,* vol. 3, pp. 1439-1461, 2003.
[40] C. Williams and D. Barber, "Bayesian Classification with Gaussian Priors," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 12, pp. 1342-1351, Dec. 1998.
[41] P. Williams, "Bayesian Regularization and Pruning Using a Laplace Prior," *Neural Computation,* vol. 7, pp. 117-143, 1995.
[42] T. Zhang and F. Oles, "Regularized Linear Classification Methods," *Information Retrieval,* vol. 4, pp. 5-31, 2001.
[43] J. Zhu and T. Hastie, "Kernel Logistic Regression and the Import Vector Machine," *Advances in Neural Information Processing Systems 14,* T. Dietterich, S. Becker, and Z. Ghahramani, eds., pp. 1081-1088, Cambridge, Mass.: MIT Press, 2002.

Balaji Krishnapuram received the BTech degree from the Indian Institute of Technology (IIT) Kharagpur in 1999 and the PhD degree from Duke University in 2004, both in electrical engineering. He works as a scientist with Siemens Medical Solutions, USA. His research interests include statistical pattern recognition, Bayesian inference, and computational learning theory. He is also interested in applications in computer-aided medical diagnosis, signal processing, computer vision, and bioinformatics.

Lawrence Carin received the BS, MS, and PhD degrees in electrical engineering from the University of Maryland, College Park, in 1985, 1986, and 1989, respectively. In 1989, he joined the Electrical Engineering Department at Polytechnic University (Brooklyn) as an assistant professor and became an associate professor there in 1994. In 1995, he joined the Electrical Engineering Department at Duke University, where he is now the William H. Younger Professor of Engineering. He was the principal investigator (PI) on a Multidisciplinary University Research Initiative (MURI) on demining (1996-2001), and he is currently the PI of a MURI dedicated to multimodal inversion. He is an associate editor of the *IEEE Transactions on Antennas and Propagation.* His current research interests include short-pulse scattering, subsurface sensing, and wave-based signal processing. He is a member of the Tau Beta Pi and Eta Kappa Nu honor societies and a fellow of the IEEE.

Mário A.T. Figueiredo received the EE, MSc, and PhD degrees in electrical and computer engineering from the Instituto Superior Técnico (IST), the engineering school of the Technical University of Lisbon, Portugal, in 1985, 1990, and 1994, respectively. Since 1994, he has been with the Department of Electrical and Computer Engineering, IST. He is also a researcher and area coordinator at the Institute of Telecommunications, Lisbon. In 1998, he held a visiting position with the Department of Computer Science and Engineering, Michigan State University. His scientific interests include image processing and analysis, computer vision, statistical pattern recognition, and statistical learning. He received the Portuguese IBM Scientific Prize in 1995 for work on unsupervised image restoration. He is an associate editor of the *IEEE Transactions on Image Processing*, *IEEE Transactions on Mobile Computing*, and *Pattern Recognition Letters*. He was guest coeditor of special issues of the *IEEE Transactions on Pattern Analysis and Machine Intelligence* and *IEEE Transactions on Signal Processing*. He was a cochair of the 2001 and 2003 Workshops on Energy Minimization Methods in Computer Vision and Pattern Recognition. He has been a member of program committees of several international conferences, including CVPR, EECV, ICASSP, ICIP, ICPR. He is a senior member of the IEEE and the IEEE Computer Society.

Alexander J. Hartemink received the SM degree in 1997 and the PhD degree in 2001 from the Department of Electrical Engineering and Computer Science at the Massachusetts Institute of Technology (MIT) as a US National Science Foundation Graduate Research Fellow. Before that, he received the MPhil degree in economics in 1996 from Oxford University as a Rhodes Scholar and the BS degree in mathematics, the BS degree in physics, and the AB degree in economics in 1994 from Duke University as an Angier B. Duke Scholar. He is currently an assistant professor of computer science at Duke and is affiliated with the Center for Bioinformatics and Computational Biology, one of five centers in Duke's Institute for Genome Sciences and Policy. His research centers on the development and application of principled computational and statistical methods for understanding complex biological systems, with a particular focus on the use of techniques from statistical and machine learning to address problems in functional genomics and systems neurobiology. He is the recipient of an Alfred P. Sloan fellowship and a Faculty Early Career Development (CAREER) Award from the US National Science Foundation.