A majorization-minimization algorithm for (multiple) hyperparameter learning

Chuan-Sheng Foo CSFOO@CS.STANFORD.EDU
Institute for Infocomm Research, 1 Fusionopolis Way, Singapore 138632, Singapore

Chuong B. Do CHUONGDO@CS.STANFORD.EDU
Andrew Y. Ng ANG@CS.STANFORD.EDU
Computer Science Department, Stanford University, Stanford, CA 94305, USA

Abstract

We present a general Bayesian framework for hyperparameter tuning in L_2-regularized supervised learning models. Paradoxically, our algorithm works by first analytically integrating out the hyperparameters from the model. We find a local optimum of the resulting non-convex optimization problem efficiently using a majorization-minimization (MM) algorithm, in which the non-convex problem is reduced to a series of convex L_2-regularized parameter estimation tasks. The principal appeal of our method is its simplicity: the updates for choosing the L_2-regularized subproblems in each step are trivial to implement (or even perform by hand), and each subproblem can be efficiently solved by adapting existing solvers. Empirical results on a variety of supervised learning models show that our algorithm is competitive with both grid-search and gradient-based algorithms, but is more efficient and far easier to implement.

1. Introduction

Regularization helps to prevent overfitting in supervised learning models by restricting model capacity. Penalty-based regularization methods encode a preference for simpler hypotheses directly into the optimization problem used for estimating model parameters. Support vector machines, for example, use an L_2 penalty of the form $\frac{1}{2}C\,||\mathbf{w}||^2$, where $\mathbf{w} \in \mathbb{R}^n$ is the vector of model parameters, $||\cdot||$ is the standard Euclidean norm, and C is a user-specified constant, commonly termed a regularization *hyperparameter*. However, this gives rise to the problem of selecting the appropriate value for the hyperparameter C.

The most common methods for hyperparameter selection use cross-validation to tune the hyperparameters. For example, in the widely used grid-search algorithm, one trains a model using a range of values for C and selects the value that gives the best performance on a holdout set. However, the computational cost of grid-search scales exponentially with the number of hyperparameters in the model, thus limiting its use to models with few hyperparameters.

For models with multiple hyperparameters, more sophisticated gradient-based algorithms exist for optimizing cross-validation loss with respect to the hyperparameters. Gradient-based hyperparameter learning algorithms have been proposed for a variety of supervised learning models, including neural networks (Larsen et al., 1996a; Andersen et al., 1997; Goutte & Larsen, 1998; Larsen et al., 1996b), support vector machines (Glasmachers & Igel, 2005; Keerthi et al., 2007; Chapelle et al., 2002), and more recently, conditional log-linear models (Do et al., 2008). However, these algorithms typically require complicated computations, making them cumbersome to implement. For example, the hyperparameter learning algorithm of Do et al. (2008) for log-linear models requires the computation of products of the Hessian of the training log-likelihood with arbitrary vectors, and performing conjugate gradient optimization for solving linear systems based on these Hessian-vector products. As such, the programmer effort required in getting a gradient-based hyperparameter learning algorithm to work in practice is non-trivial.

For probabilistic supervised learning models, Bayesian methods provide an alternative to cross-validation–based methods for hyperparameter learning; Bayesian methods treat hyperparameters as parameters in the model with prespecified prior distributions. We focus on the "integrate-out" approach proposed by Buntine and Weigend (1991),

Appearing in *Proceedings of the 26^{th} International Conference on Machine Learning*, Montreal, Canada, 2009. Copyright 2009 by the author(s)/owner(s).

and later extended by Williams (1995), Cawley and Talbot (2006), and Cawley, Talbot and Girolami (2007). In this approach, an uninformative prior is placed over the regularization hyperparameter, which is then integrated out, resulting in a modified regularization penalty (and optimization objective) in which the hyperparameters have been *eliminated*.

In this paper, we propose a simple and general framework for hyperparameter learning in L_2-regularized models based on Buntine and Weigend's strategy. Using their Bayesian formulation, our framework expresses the problem of parameter learning in a model where regularization hyperparameters have been eliminated as a non-convex optimization problem. In general, although non-convex optimization problems can be extremely difficult to solve, we propose an effective strategy based on *majorization-minimization*. In this approach, we iteratively replace the non-convex portion of our objective function with a convex upper-bound. Each convex optimization subproblem, in turn, has the form of an L_2-regularized parameter estimation task, which we solve efficiently by adapting existing solvers. Finally, we interpret the regularization coefficients from the sequence of convex subproblems as the hyperparameter choices in our hyperparameter learning algorithm.

The salient characteristic of the majorization-minimization algorithm when applied to our problem is its striking ease of implementation. Like cross-validation–based strategies, our algorithm requires iterated solution of regularized parameter estimation problems for varying choices of hyperparameters. Unlike gradient-based methods, however, the choice of hyperparameter setting in each iteration is a simple closed-form expression which is trivial to evaluate. However, the assumptions made by the Bayesian model are not without consequences, and we do not claim that the hyperparameters chosen by our algorithm will always be optimal for generalization performance. In practice, however, our approach often performs surprisingly well in spite of its simplicity. Moreover, our framework is applicable to a wide variety of problems, including structured classification models (*e.g.*, log-linear models) and standard classification/regression models (*e.g.*, least-squares and logistic regression).

2. The "integrate out" strategy

Consider a general supervised learning problem, in which $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^{m}$ is a training set of m *independently and identically distributed* (IID) examples. The labels $y^{(i)}$ may be real valued, discrete, or structured, depending on the type of problem we are trying to solve. We wish to estimate the parameters $\mathbf{w} \in \mathbb{R}^n$ for a probabilistic model of y given x, $p(y|x;\mathbf{w})$. The usual approach is to formulate the learning problem as regularized maximum likelihood (ML) or maximum *a posteriori* (MAP) estimation,

$$\mathbf{w}_{MAP} = \arg\min_{\mathbf{w}} \left[-\log p(\mathcal{D}|\mathbf{w}) - \log p(\mathbf{w};C)\right], \quad (1)$$

where $p(\mathbf{w};C)$ is a prior distribution over model parameters with the hyperparameter C.[1] For example, when $p(\mathbf{w};C) \propto \exp(-\frac{1}{2}C\,||\mathbf{w}||^2)$ is an isotropic zero-mean Gaussian prior with inverse variance C, then the second term above simplifies to $\frac{1}{2}C\,||\mathbf{w}||^2$, up to additive constants independent of C.

In an ideal Bayesian framework, one regards C as another random variable in the model with some fixed prior distribution $p(C)$, and avoids committing to point estimates of model parameters altogether by making predictions via the integral, $p(y|x,\mathcal{D}) = \int_{\mathbf{w}} p(y|x,\mathbf{w})p(\mathbf{w}|\mathcal{D})\,d\mathbf{w}$, where $p(\mathbf{w}|\mathcal{D}) \propto \int_C p(\mathcal{D}|\mathbf{w})p(\mathbf{w}|C)p(C)\,dC$ follows from Bayes' rule. In general, the combination of integrals required for ideal Bayesian inference is analytically intractable, so one must resort to either sampling strategies or approximations. One possibility is to approximate $p(\mathbf{w}|\mathcal{D})$ using a point mass centered at its mode, $\mathbf{w}^* = \arg\max_{\mathbf{w}} \left[\int_C p(\mathcal{D}|\mathbf{w})p(\mathbf{w}|C)p(C)\,dC\right]$, or equivalently, $\mathbf{w}^*$ is given by

$$\arg\min_{\mathbf{w}} \left[-\log p(\mathcal{D}|\mathbf{w}) - \log \int_C p(\mathbf{w}|C)p(C)dC\right]. \quad (2)$$

In this form, we see that (2) is a close variant of (1); while $p(\mathbf{w};C)$ depends on C, the hyperparameter C has been marginalized out in $p(\mathbf{w}) = \int_C p(\mathbf{w}|C)p(C)dC$. For this reason, (2) is often referred to as the "integrate out" approach for dealing with hyperparameters. The integrate out approach was first introduced by Buntine and Weigend (1991) and subsequently applied by Williams (1995), Cawley and Talbot (2006), and Cawley, Talbot and Girolami (2007). This approach extends in a straightforward manner to the case where multiple hyperparameters or hyperparameter groupings are used. For clarity of exposition we first present the derivations for the single hyperparameter case. We treat the multiple and grouped hyperparameter cases in a later subsection.

Integrating out a single hyperparameter Consider an isotropic Gaussian prior of the form $p(\mathbf{w}|C) \propto \exp(-\frac{1}{2}C\,||\mathbf{w}||^2)$, which corresponds to the product of n independent zero-mean Gaussian priors with common variance $1/C$ for each parameter w_i. If we place a Gamma prior on C, it is possible to compute the integral for $p(\mathbf{w})$ analytically in closed form. Similar derivations for Laplace priors over the parameters w_i and Jeffrey's prior for C were presented in (Williams, 1995; Cawley et al., 2007).

[1] Here, we view C as a *pre-determined* parameter of the distribution $p(\mathbf{w};C)$, and not a random variable. This is why we do not condition over C.

More precisely, suppose $C \sim \text{Gamma}(\alpha, \beta)$, such that $p(C; \alpha, \beta) = \frac{C^{\alpha-1}\beta^{\alpha}\exp(-\beta C)}{\Gamma(\alpha)}$. In this case, the desired integral for $p(\mathbf{w})$ is

$$\int_0^{\infty} \left(\frac{C}{2\pi}\right)^{\frac{n}{2}} e^{-\frac{1}{2}C||\mathbf{w}||^2} \cdot \frac{C^{\alpha-1}\beta^{\alpha}e^{-\beta C}}{\Gamma(\alpha)}\, dC$$
$$= \int_0^{\infty} \frac{\beta^{\alpha}}{(2\pi)^{\frac{n}{2}}\Gamma(\alpha)} C^{\frac{n}{2}+\alpha-1}\, e^{-\left(\frac{1}{2}||\mathbf{w}||^2+\beta\right)C}\, dC$$
$$= \frac{\beta^{\alpha}\Gamma(\frac{n}{2}+\alpha)}{(2\pi)^{\frac{n}{2}}\Gamma(\alpha)} \left(\frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2 + \beta\right)^{-\left(\frac{n}{2}+\alpha\right)} \int_0^{\infty} I(C)\, dC$$

where the integrand $I(C)$ is given by

$$I(C) = \frac{C^{\frac{n}{2}+\alpha-1}\left(\frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2+\beta\right)^{\frac{n}{2}+\alpha} e^{-\left(\frac{1}{2}||\mathbf{w}||^2+\beta\right)C}}{\Gamma(\frac{n}{2}+\alpha)};$$

we have factored out terms independent of C and the additional term $\Gamma(\frac{n}{2}+\alpha)\left(\frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2+\beta\right)^{-\left(\frac{n}{2}+\alpha\right)}$ to form the new integrand $I(C)$. Since $I(C)$ is actually the probability density function for a Gamma distribution (with parameters $\frac{n}{2}+\alpha$ and $\frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2+\beta$), $\int_0^{\infty} I(C)\, dC = 1$. Thus, the integral for $p(\mathbf{w})$ simplifies to

$$\frac{\beta^{\alpha}\Gamma(\frac{n}{2}+\alpha)}{(2\pi)^{\frac{n}{2}}\Gamma(\alpha)}\left(\frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2+\beta\right)^{-\left(\frac{n}{2}+\alpha\right)}. \qquad (3)$$

Hence, up to additive constants independent of $\mathbf{w}$, $\log p(\mathbf{w}) = -\left(\frac{n}{2}+\alpha\right)\log\left(\frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2+\beta\right)$ and we have successfully eliminated the regularization constant C.

Integrating out multiple hyperparameters In some cases, different features in a probabilistic model may be known to have considerably different prior variances. Here, we show how the "integrate out" framework may be extended for multiple hyperparameters by describing a more flexible parameterization of the prior $p(\mathbf{w}|\mathbf{C})$ in terms of a vector of parameters $\mathbf{C} := (C_1, \ldots, C_k)$. This parameterization is adapted from (Do et al., 2008) and is also described briefly in (Williams, 1995).

Consider a setting in which parameters $\mathbf{w} = (w_1, \ldots, w_n)$ have been partitioned into k fixed groups, known as *regularization groups*, such that the parameters in each group are known in advance to have roughly similar variances. Such a setting occurs regularly in structured estimation tasks, where different types of features have varying frequencies of occurrence. For example, in natural language processing tasks, unigram and bigram features might be regularized separately; in RNA secondary structure prediction, features for different types of structural motifs should be treated separately.

Let $\pi : \{1, \ldots, n\} \to \{1, \ldots, k\}$ be a fixed mapping from parameters to their corresponding regularization groups. This mapping should be available to the learning algorithm prior to training. The inverse image of this mapping is defined by $\pi^{-1}(j) := \{i \in \{1, \ldots, n\} \,|\, \pi(i) = j\}$. Finally, let $n_j := \left|\pi^{-1}(j)\right|$ be the size of the jth regularization group, for $j = 1, \ldots, k$. An automatic relevance determination (ARD) (MacKay, 1992) prior is a special case of this framework, where we let each hyperparameter be in its own group.

Suppose $p(\mathbf{w}|\mathbf{C}) \propto \exp(-\frac{1}{2}\sum_i C_{\pi(i)} w_i^2)$. Furthermore, suppose that each C_j is now independently sampled from a Gamma distribution with common parameters α and β. Following the derivation in the previous section, we obtain

$$p(\mathbf{w}) \propto \prod_{j=1}^{k} \Gamma\left(\frac{n_j}{2}+\alpha\right)\left(\beta + \frac{1}{2}\sum_{i\in\pi^{-1}(j)} w_i^2\right)^{-\left(\frac{n_j}{2}+\alpha\right)} \qquad (4)$$

and therefore up to additive constants,

$$\log p(\mathbf{w}) = -\sum_{j=1}^{k}\left(\frac{n_j}{2}+\alpha\right)\log\left(\beta+\frac{1}{2}\sum_{i\in\pi^{-1}(j)} w_i^2\right). \qquad (5)$$

3. A majorization-minimization algorithm for hyperparameter learning

In this section we describe how to perform the minimization in equation (2) using the new prior $p(\mathbf{w})$ computed in the previous section, which results in the optimization problem (for the single hyperparameter case)

$$\min_{\mathbf{w}} \left[-\log p(\mathcal{D}|\mathbf{w}) + \left(\frac{n}{2}+\alpha\right)\log\left(\frac{1}{2}\left|\left|\mathbf{w}\right|\right|^2+\beta\right)\right]. \qquad (6)$$

In the models we consider in this paper, the negative log-likelihood $-\log p(\mathcal{D}|\mathbf{w})$ is convex in the parameters $\mathbf{w}$. However, the second term corresponding to the log marginalized prior is in fact *neither convex nor concave* in $\mathbf{w}$. Because the log prior is differentiable, one could consider applying a generic gradient-based algorithm in order to identify a local minimum of the objective function. However, we found that in our experiments, such a strategy is highly prone to identifying poor local minima, which do not give as good performance in practice as compared to our algorithm.

Single hyperparameter learning Because the optimization objective contains a term which is neither convex nor concave, we cannot simply apply standard algorithms for minimizing a difference of two convex functions, such as the Convex-Concave Procedure (CCCP) (Yuille & Rangarajan, 2003). Instead, we employ a bound optimization strategy which is a special instance of the class of

Algorithm 1 Majorization-minimization algorithm for single hyperparameter learning

```
Initialize $t := 0$.
**repeat**
    Compute $C^{(t)} := \begin{cases} 1 & \text{if } t = 0 \\ \frac{n/2+\alpha}{\frac{1}{2}\left|\left|\mathbf{w}^{(t)}\right|\right|^2+\beta} & \text{otherwise.} \end{cases}$
    $\mathbf{w}^{(t+1)} := \arg\min_{\mathbf{w}} \left[ -\log p(\mathcal{D}|\mathbf{w}) + \frac{1}{2} C^{(t)} \left|\left|\mathbf{w}\right|\right|^2 \right].$
**until** convergence
```

majorization-minimization (MM) algorithms (Lange et al., 2000). Our strategy proceeds in an EM-like fashion, alternating between construction of a convex upper bound of the objective function and minimization of that bound.

To see how this is done, recall that the function $\log x$ is concave over the domain $\mathbb{R}_{++} = (0, \infty)$, and hence is upper-bounded by its first-order Taylor expansion. That is, for any $x, y \in \mathbb{R}_{++}$, $\log x \leq \log y + (x - y)/y = \log y + x/y - 1$; furthermore, the inequality is tight if and only if $x = y$. Let $h(\mathbf{w}) = \frac{1}{2} \left|\left|\mathbf{w}\right|\right|^2 + \beta$. Then, given a parameter vector $\mathbf{w}^{(t)}$ on iteration t, it follows that for any $\mathbf{w} \in \mathbb{R}^n$, if $\beta > 0$,

$$\log\left(h(\mathbf{w})\right) \leq \log\left(h(\mathbf{w}^{(t)})\right) + \frac{h(\mathbf{w})}{h(\mathbf{w}^{(t)})} - 1. \qquad (7)$$

Letting $f(\mathbf{w})$ denote the objective function of (6), and defining the function

$$g(\mathbf{w}; \mathbf{w}^{(t)}) := -\log p(\mathcal{D}|\mathbf{w}) \\ + \left(\frac{n}{2} + \alpha\right) \left[\log\left(h(\mathbf{w}^{(t)})\right) + \frac{h(\mathbf{w})}{h(\mathbf{w}^{(t)})} - 1 \right],$$

it follows from (7) that $f(\mathbf{w}) \leq g(\mathbf{w}; \mathbf{w}^{(t)})$ for any parameter vectors $\mathbf{w}$ and $\mathbf{w}^{(t)}$, provided $\beta > 0$.

The MM algorithm proceeds by repeatedly constructing a convex upper bound for the non-convex prior terms using (7), and then minimizing this upper bound until convergence. In particular, it computes a sequence of successive iterates for the parameter vector $\mathbf{w}$ as follows: given a parameter vector $\mathbf{w}^{(t)}$ on iteration t, compute the next iterate in the sequence, $\mathbf{w}^{(t+1)}$, via $\mathbf{w}^{(t+1)} := \arg\min_{\mathbf{w}} g(\mathbf{w}; \mathbf{w}^{(t)})$. More explicitly, $\mathbf{w}^{(t+1)}$ is given by

$$\arg\min_{\mathbf{w}} \left[-\log p(\mathcal{D}|\mathbf{w}) + \left(\frac{n/2 + \alpha}{\frac{1}{2}\left|\left|\mathbf{w}^{(t)}\right|\right|^2 + \beta} \right) \left(\frac{1}{2} \left|\left|\mathbf{w}\right|\right|^2 \right) \right],$$

where we have omitted constant terms from the upper-bounding function that do not affect the optimization.[2] We may then show that our algorithm monotonically decreases the objective, since

$$f(\mathbf{w}^{(t+1)}) \leq g(\mathbf{w}^{(t+1)}; \mathbf{w}^{(t)}) \leq g(\mathbf{w}^{(t)}; \mathbf{w}^{(t)}) = f(\mathbf{w}^{(t)}).$$

The first inequality follows from the argument that $g(\mathbf{w}; \mathbf{w}^{(t)})$ upper bounds $f(\mathbf{w})$ for any $\mathbf{w}$, and the second inequality holds because our algorithm computes $\mathbf{w}^{(t+1)} = \arg\min_{\mathbf{w}} g(\mathbf{w}; \mathbf{w}^{(t)})$.[3] Observe that as a consequence of our upper-bounding procedure, the objective function on each iteration t is that of an L_2-regularized ML estimation problem with regularization constant $C^{(t)} = \frac{n/2+\alpha}{\frac{1}{2}\left|\left|\mathbf{w}^{(t)}\right|\right|^2+\beta}$ (see Algorithm 1). This allows us to use existing solvers for such problems to compute the iterates, making the algorithm easy to implement as a wrapper around any such solver. In addition, improvements in such solvers translate directly into improvements for this algorithm: one may make use of faster solvers for solving these problems when available to speed up the algorithm.

Moreover, the objective functions in each iteration give us a way to interpret what the algorithm is doing. Upon termination, the algorithm will have solved a regularized ML estimation problem with a specific regularization constant $C^{(t)}$, which we can interpret to be the optimal value for the constant. Thus, even though we originally eliminated the regularization hyperparameters from the objective function, the majorization-minimization algorithm provides us a way to learn the optimal hyperparameter C for training a model using standard MAP estimation.

Multiple hyperparameter learning To generalize the majorization-minimization algorithm to the multiple hyperparameter case, observe that the new prior in this case is the sum of multiple functions of the same essential form as the regularizer in (6). Hence, we can apply the upper bounding argument to each logarithm in the same way as before (see Algorithm 2). Here at each step the algorithm solves a *weighted* L_2 regularized ML estimation problem, with the weights for each regularization group given analogously to how they are given in the single hyperparameter case. As before, we may interpret the weights in the final step of the algorithm, when it converges, to be the optimal hyperparameter settings for a regularized ML estimation problem.

[2] In practice, we can set the initial parameter vector $\mathbf{w}^{(0)}$ to be the solution of a regularized ML estimation problem with the regularization constant set to an arbitrary value, e.g., 1.

[3] In fact, when $\alpha = 0$ and $\beta = 1$, our algorithm for single hyperparameter learning is an instance of the algorithm proposed by (Delaney & Bresler, 1998) for edge-preserving regularized image reconstruction, who proved convergence of the method to a local minimizer of (6) under certain conditions. For convenience, we used $\alpha = 0$, and $\beta = 1$ in our experiments. We performed a sensitivity analysis in our experiments using multinomial logistic regression and found that this is a reasonable choice. For other values of α and $\beta > 0$, it is also possible to prove convergence of our algorithm using a similar proof to that in (Delaney & Bresler, 1998). We omit the proof due to a lack of space.

Algorithm 2 Majorization-minimization algorithm for multiple hyperparameter learning

```
Initialize $t := 0$.
**repeat**
  For $j = 1, \dots, k$, compute
    $C_j^{(t)} := \begin{cases} 1 & \text{if } t = 0 \\ \frac{n_j/2+\alpha}{\frac{1}{2}\sum_{i\in\pi^{-1}(j)}\left(w_i^{(t)}\right)^2+\beta} & \text{otherwise.} \end{cases}$
  $\mathbf{w}^{(t+1)} := \arg\min_{\mathbf{w}} \left[ -\log p(\mathcal{D}|\mathbf{w}) + \frac{1}{2}\sum_{i=1}^{n} C_{\pi(i)}^{(t)} w_i^2 \right].$
**until** convergence
```

4. Experiments

In order to illustrate the generality of our proposed framework, we show that it is competitive with grid-search on linear regression; that it is competitive with grid-search and the gradient-based method in (Do et al., 2008) on logistic and multinomial logistic regression models when evaluated on an array of tasks;[4] and that it is competitive with a gradient-based algorithm in the case of a conditional log-linear model for RNA secondary structure prediction. We also evaluated the approach of directly optimizing the modified objective using L-BFGS. The sensitivity of the algorithm to parameters α and β was also evaluated on the multinomial logistic regression model.

Linear regression Suppose that the target labels $y^{(i)} \in \mathbb{R}$ are related to the input variables $\mathbf{x}^{(i)}$ via $y^{(i)} = \mathbf{w}^T\mathbf{x}^{(i)} + \epsilon^{(i)}$ where $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, C^{-1}\mathbf{I})$ and $\epsilon^{(i)} \sim \mathcal{N}(0, \sigma^2)$ for some unknown variance parameter σ^2. In this case, we may treat σ^2 as a random variable in a similar way to the regularization hyperparameter C. Integrating out σ using Jeffrey's prior, we obtain the optimization problem,

$$\mathbf{w}^* = \arg\min_{w} \left[\frac{m}{2} \log \sum_{i=1}^{m} \left(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)} \right)^2 + \frac{1}{2} C \left|\left|\mathbf{w}\right|\right|^2 \right]$$

In this setting, the regularized ML estimation problems in each iteration of the majorization-minimization algorithm are standard ridge regression problems which can be solved using standard numerical methods. We compared the performance of our algorithm against grid-search, in terms of mean squared-error (MSE). For reference, we also include the MSE of ordinary linear regression (LR).

Table 1. Methods comparison for L_2-regularized linear regression, on test set MSE. -n/a- indicates that the problem was ill-conditioned and no unique solution exists.

Dataset	Test MSE			Best C	
	LR	Grid	MM	Grid	MM
abalone	4.815	4.970	4.821	1	0.109
bodyfat	0.000	0.000	0.000	0.0625	0.0600
cpusmall	93.325	93.506	93.381	0.5	0.160
housing	24.111	24.833	24.491	2	1.04
mg	0.0213	0.0213	0.0213	2	1.630
mpg	10.759	11.223	11.204	1	0.951
pyrim	-n/a-	0.00355	0.00367	8	1.137
space	0.0190	0.0190	0.0190	0.00391	0.00503
triazines	-n/a-	0.0272	0.0247	2	26.579

Table 2. Methods comparison for L_2-regularized logistic regression.

Dataset	Accuracy (%)			
	Grid	Grad	Direct	MM
australian	86.96	86.47	85.51	85.51
breast-cancer	96.08	96.08	96.57	96.57
diabetes	76.52	76.96	76.52	76.09
german-numer	75.33	75.33	75.00	75.00
heart	83.95	87.65	83.95	86.42
ionosphere	83.81	82.86	80.00	82.86
liver-disorders	68.93	67.96	67.96	65.05
mushrooms	100.00	100.00	100.00	100.00
sonar	70.97	75.81	66.13	70.97
splice	73.24	85.47	73.29	85.33
w1a	97.32	97.83	97.20	97.03

As shown in the Table 1, the MSE of all the methods are almost identical. This may be because the datasets we tested the methods on are too easy, since even ordinary linear regression performs well. In fact, it often performs best, except on two ill-conditioned datasets, pyrim and triazines, for which no unique solution exists, causing it to fail. Also, we observe that the optimal regularization hyperparameters found by our approach are qualitatively similar to those found by grid-search. Moreover, these hyperparameters were found after only 4-7 iterations of the MM algorithm.

Logistic regression We applied the framework to L_2-regularized logistic regression, and compared it to grid-search and the gradient-based method. In this model, we have labels $y \in \{-1, +1\}$ distributed according to $p(y|\mathbf{x}; \mathbf{w}) = \frac{1}{1+\exp(-y\mathbf{w}^T\mathbf{x})}$.

As seen from the results in Table 2, our algorithm performs competitively with both grid-search and the gradient-based method. The results also show that directly optimizing the modified objective yields worse results on a number of datasets, most notably on the sonar (5% lower) and splice (12% lower) datasets. We observed that our algorithm is 11 to 34 times as fast as grid-search, and 1.2 to 25 times as fast as the gradient-based method, as measured by running

[4] The datasets used were obtained from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. We used the supplied test sets, or otherwise held out 30% of the data as the test set and used the remaining 70% for training. All algorithms were given the same amount of training data, from which a validation set was constructed if necessary. 5-fold cross-validation was used in the experiments for grid-search and we considered values $C = 2^k$ for integral k such that $-10 \leq k \leq 10$.

time. On 7 of the datasets, our algorithm was at least 20 times as fast as grid-search.

Multinomial Logistic Regression Generalizing logistic regression to the case where we have multiple classes yields the multinomial logistic regression model. Here we have labels $y \in \{1, \ldots, k\}$, distributed as $p(y = k|\mathbf{x}; \mathbf{w}) = \frac{\exp(\mathbf{w}_k^T \mathbf{x})}{\sum_{i=1}^{k} \exp(\mathbf{w}_i^T \mathbf{x})}$, where $\mathbf{w}_i$ is the parameter vector corresponding to class i. We compared our framework against grid search and the gradient-based algorithm.

As can be seen from the results in Table 3(a), our algorithm performs similarly to grid-search and the gradient-based algorithm, even outperforming these competing algorithms on some datasets (dna, glass, iris). However, our algorithm performs worse on the svmguide2 dataset. The results also show that our MM algorithm performs as well or better than direct optimization of the modified objective; direct optimization yielded significantly worse performance on the iris (11% lower) and vowel (7% lower) datasets. Our algorithm was 3.3 to 25 times as fast as grid-search as measured by execution time. As compared to the gradient-based method, our algorithm was 2 to 20 times as fast. Substantial speedups were observed on the larger datasets (connect-4, dna, mnist1, and usps), being 11 to 25 times as fast as grid-search and 2 to 11 times as fast as the gradient-based algorithm.

We also tested for the stability of the learned hyperparameters and model accuracy with respect to α and β using the multinomial logistic regression model. We ran our algorithm for all pairs of α, β where $\alpha, \beta \in \{2^{-7}, 2^{-5}, \ldots, 2^{11}\}$, on each of the datasets listed in Table 3(a). In general, the accuracy of the model did not vary dramatically, except when $\alpha > 2^5$, where the accuracy drops drastically in some cases. On some datasets the learned hyperparameter did not vary much, but on others there was a general increasing trend as α was increased and β was decreased.

Conditional log-linear models Ribonucleic acids (RNAs) are a class of biological macromolecules which play important roles in all living cells. In the problem of RNA secondary structure prediction, we are given an RNA sequence x and asked to predict the pattern of nested base-pairings y that arise when the RNA folds *in vivo*. For this real world task, we applied our hyperparameter learning framework to a probabilistic modeling framework known as conditional log-linear models (CLLMs). In a CLLM, the conditional probability of y given x is modelled as $p(y|x; \mathbf{w}) \propto \exp(\mathbf{w}^T \mathbf{F}(x, y))$ where $\mathbf{F}$ is a mapping of input-output pairs to features.

Here, the features were constructed to mirror the energetic terms found in standard RNA physics-based models, such as hairpin loops, bulge loops and interior loops (Do et al., 2006). Unlike in the previous tasks, the features for the RNA secondary structure problem have very natural groupings based on the types of structural motifs they detect; we took advantage of these groupings in order to test the performance our hyperparameter learning algorithm in the multiple hyperparameter setting.

Table 4. ROC area for the RNA secondary structure prediction task.

Hyperparameter(s)	Gradient	Direct	MM
Single	0.619	0.603	0.603
Grouped	0.638	0.612	0.633

We evaluated our algorithm on a collection of 151 known RNA sequence-structure pairs culled from the Rfam database (Griffiths-Jones et al., 2003) using two-fold cross-validation, and compared it to the gradient-based hyperparameter learning algorithm described in (Do et al., 2008). As shown in Table 3(b), the hyperparameters learned on each fold are practically identical, which reflects the robustness of our approach. Moreover, even after just a single iteration of the iterative linearization algorithm, the learnt hyperparameters were qualitatively close to their final values, further showing the robustness of our approach. In fact, features with small regularization hyperparameters correspond to properties of RNA molecules that are known to contribute strongly to the energetics of RNA secondary structure formation, while most of the features with large regularization hyperparameters correspond to structural properties that are not as well correlated to RNA secondary structure, or may be simply too noisy to be useful.

For cross-validation performance, we used ROC area as a measure of accuracy, as described in (Do et al., 2006). The results are summarized in Table 4. Our algorithm gives slightly worse but nonetheless competitive results as compared to the gradient-based method in terms of the ROC area. Considering the complexity involved in the implementation of the gradient-based method, the slight reduction in performance may be a small price to pay for ease of implementation. We also tried directly optimizing the non-convex objective function using standard gradient methods; while this gave comparable performance in the single hyperparameter case, in the multiple hyperparameter setting, it performed significantly worse, indicating convergence to a poor local optimum.

5. Related work and discussion

We have presented a Bayesian approach for hyperparameter learning based on placing a Gamma prior over the hyperparameters and integrating them out of the model. A

Table 3. (a) Methods comparison for L_2-regularized multinomial logistic regression.. (b) Grouped hyperparameters learned for RNA secondary structure prediction task.

Dataset	k	Accuracy (%)			
		Grid	Grad	Direct	MM
connect-4	3	75.62	75.62	75.59	75.60
dna	3	94.94	95.03	95.11	95.11
glass	6	64.06	65.62	67.19	67.19
iris	3	84.44	88.89	82.22	93.33
letter	26	75.62	77.34	75.60	77.30
mnist1	10	91.44	91.64	89.68	89.86
satimage	6	83.25	83.65	83.80	83.50
segment	7	95.24	95.09	93.51	93.51
svmguide2	3	84.62	84.62	60.68	60.68
usps	10	92.08	91.98	90.33	90.33
vehicle	4	82.21	83.00	82.21	83.00
vowel	11	40.48	48.70	40.91	48.27
wine	3	98.11	98.11	98.11	98.11

(a)

Hyperparameter group	After 1 iteration		Final values	
	Fold A	Fold B	Fold A	Fold B
base pairings	1.647	1.884	1.047	1.129
terminal mismatch interactions	13.473	10.442	126.280	124.953
hairpin loop lengths	0.771	0.875	0.886	0.953
explicit internal loop sizes	4.436	3.505	3.891	3.478
bulge loop lengths	2.961	3.310	5.903	6.200
internal loop lengths	4.663	3.072	9.626	8.792
symmetric internal loop lengths	5.528	5.469	6.617	6.511
internal loop asymmetry	5.710	6.869	9.798	10.967
single base bulge nucleotides	1.882	1.943	1.261	1.241
1×1 internal loop nucleotides	2.140	3.209	6.236	6.571
helix stacking interactions	20.738	21.066	67.288	65.758
helix closing interactions	0.683	0.788	0.302	0.380
multi-branch loop lengths	0.454	0.443	0.329	0.312
single base pair stacking interactions	12.377	13.262	61.294	62.128
external loop lengths	0.691	0.787	0.833	0.779

(b)

similar approach has also been adopted for L_2-regularized models by Figueiredo (2003) for the purpose of learning sparse models. However, in his approach, an uninformative Jeffrey's prior is placed over the variance, as opposed to the inverse variance, as in our formulation.[5] Nonetheless, the resultant distributions over model parameters are equivalent in the case where each parameter is separately regularized. In the other settings where we have single or grouped hyperparameters, the integrals involved when we proceed using Figueiredo's approach become analytically intractable, and it is unclear how to proceed further. In contrast, the integrals remain tractable in our framework, and we can efficiently find a local minimum for the resultant optimization problem using the majorization-minimization algorithm that we propose in this paper. Williams (1995) and Cawley *et al.* (2007) have also adopted the approach of integrating out the hyperparameters in the case when L_1-regularization is used. Williams applied this approach to neural networks while Cawley *et al.* applied it to multinomial logistic regression. A similar optimization algorithm is described in (Fazel et al., 2003) for the purpose of matrix rank minimization.

Another possible approach would be to integrate out the parameters to obtain the marginal likelihood (or evidence), and optimize the latter over the hyperparameters. This is known as the empirical Bayes (i.e., ML-II) strategy, and it is used in Automatic Relevance Determination (MacKay, 1992) and the Relevance Vector Machine (RVM) (Tipping, 2001). In these two methods, the posterior distribution over model parameters is approximated using a Gaussian distribution about their most probable values. However, in order to compute the required "Occam factor," one has to compute the determinant of the Hessian matrix, which may be computationally expensive, for example, in large log-linear models. More recently, a fast marginal likelihood optimization algorithm has been proposed for the RVM (Tipping & Faul, 2003). A third approach is the fully Bayesian strategy, in which both parameters and hyperparameters are integrated out to obtain a posterior distribution over all possible outputs given the input data. However, the integrals are typically analytically intractable, and Monte Carlo integration techniques (Neal, 1996) may be used to approximate them. Such techniques are computationally expensive and may be slow to converge.

As mentioned, numerous other approaches to hyperparameter learning exist. While our approach may not always give the best performance as compared to all these other methods, its performance is quite competitive; we do note that in a number of situations, possibly due to model misspecification, our method performs worse than grid-search and the gradient-based algorithm. Nonetheless, there are typically efficiency gains as compared to grid-search and the gradient-based algorithm, especially on larger datasets. Our algorithm thus appears to be trading off a little accuracy for modest efficiency gains. One issue that needs to be further explored is the stability of the model with regards to the α and β parameters. While our simple choice of $\alpha = 0, \beta = 1$ has worked well on the array of tasks that we described in this paper, further investigation is needed to see if this choice works well in general.

The most important characteristic of our method, however, is its simplicity in implementation. Generally, the majorization-minimization algorithm reduces to writing a wrapper around existing solvers. Often, the updates can even be easily computed by hand, making this method ideal for quick-and-dirty machine learning "proof-of-concept" applications. If additional performance is desired, one may then turn to model-specific methods which may require greater effort in implementation and more computation.

[5] The uninformative Jeffrey's prior may be viewed as the limit of Gamma distributions whose parameters tend to zero.

Acknowledgments

We would like to thank the reviewers for their comments. This work was done while CSF was a student at Stanford University. CSF was supported by an A*STAR National Science Scholarship. CBD was supported by an NSF Graduate Research Fellowship.

References

Andersen, L. N., Larsen, J., Hansen, L. K., & Hintz-Madsen, M. (1997). Adaptive regularization of neural classifiers. *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing VII, Amelia Island, FL, USA* (pp. 24–33).

Buntine, W. L., & Weigend, A. S. (1991). Bayesian backpropagation. *Complex Systems*, *5*, 603–643.

Cawley, G. C., Talbot, N. L., & Girolami, M. (2007). Sparse multinomial logistic regression via Bayesian L1 regularisation. In B. Schölkopf, J. Platt and T. Hoffman (Eds.), *Advances in Neural Information Processing Systems 19*, 209–216. Cambridge, MA: MIT Press.

Cawley, G. C., & Talbot, N. L. C. (2006). Gene selection in cancer classification using sparse logistic regression with Bayesian regularization. *Bioinformatics*, *22*, 2348–2355.

Chapelle, O., Vapnik, V., Bousquet, O., & Mukherjee, S. (2002). Choosing multiple parameters for support vector machines. *Machine Learning*, *46*, 131–159.

Delaney, A. H., & Bresler, Y. (1998). Globally convergent edge-preserving regularized reconstruction: an application to limited-angle tomography. *IEEE Transactions on Image Processing*, *7*, 204–221.

Do, C. B., Foo, C.-S., & Ng, A. Y. (2008). Efficient multiple hyperparameter learning for log-linear models. In J. Platt, D. Koller, Y. Singer and S. Roweis (Eds.), *Advances in Neural Information Processing Systems 20*, 377–384. Cambridge, MA: MIT Press.

Do, C. B., Woods, D. A., & Batzoglou, S. (2006). CONTRAfold: RNA secondary structure prediction without physics-based models. *Bioinformatics*, *22*, e90–e98.

Fazel, M., Hindi, H., & Boyd, S. P. (2003). Log-det heuristic for matrix rank minimization with applications to Hankel and Euclidean distance matrices. *Proceedings of the 2003 American Control Conference* (pp. 2156–2162 vol. 3).

Figueiredo, M. A. T. (2003). Adaptive sparseness for supervised learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *25*, 1150–1159.

Glasmachers, T., & Igel, C. (2005). Gradient-based adaptation of general Gaussian kernels. *Neural Computation*, *17*, 2099–2105.

Goutte, C., & Larsen, J. (1998). Adaptive regularization of neural networks using conjugate gradient. *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech and Signal Processing, Seattle, WA, USA* (pp. 1201–1204 vol.2).

Griffiths-Jones, S., Bateman, A., Marshall, M., Khanna, A., & Eddy., S. R. (2003). Rfam: an RNA family database. *Nucleic Acids Research*, *31*, 439–441.

Keerthi, S. S., Sindhwani, V., & Chapelle, O. (2007). An efficient method for gradient-based adaptation of hyperparameters in SVM models. In B. Schölkopf, J. Platt and T. Hoffman (Eds.), *Advances in Neural Information Processing Systems 19*, 673–680. Cambridge, MA: MIT Press.

Lange, K., Hunter, D. R., & Y., I. (2000). Optimization transfer using surrogate objective functions. *Journal of Computational and Graphical Statistics*, *9*, 1–59.

Larsen, J., Hansen, L. K., Svarer, C., & Ohlsson, M. (1996a). Design and regularization of neural networks: the optimal use of a validation set. *Proceedings of the IEEE Workshop on Neural Networks for Signal Processing VI, Kyoto, Japan* (pp. 62–71).

Larsen, J., Svarer, C., Andersen, L. N., & Hansen, L. K. (1996b). Adaptive regularization in neural network modeling. *Neural Networks: Tricks of the Trade* (pp. 113–132).

MacKay, D. J. C. (1992). Bayesian interpolation. *Neural Computation*, *4*, 415–447.

Neal, R. M. (1996). *Bayesian Learning for Neural Networks*. Springer.

Tipping, M., & Faul, A. (2003). Fast marginal likelihood maximisation for sparse Bayesian models. In C. M. Bishop and B. J. Frey (Eds.), *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics, Key West, FL, USA*.

Tipping, M. E. (2001). Sparse Bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, *1*, 211–244.

Williams, P. M. (1995). Bayesian regularization and pruning using a Laplace prior. *Neural Computation*, *7*, 117–143.

Yuille, A. L., & Rangarajan, A. (2003). The concave-convex procedure. *Neural Computation*, *15*, 915–936.