Adaptive optimization of hyperparameters in L2-regularised logistic regression

Ahmed Abdel-Gawad
ahmedag@stanford.edu

Simon Ratner
sratner@stanford.edu

December 14, 2007

Abstract

We investigate a gradient-based method for adaptive optimization of hyperparameters in logistic regression models. Adaptive optimization of hyperparameters reduces the computational cost of selecting good hyperparameter values, and allows these optimal values to be pinpointed more precisely, as compared to an exhaustive search of the hyperparameter space.

1 Introduction

Supervised learning methods often include many additional hyperparameters besides the parameters being optimized, such as the regularization parameters in regularized methods, and kernel parameters. The effectiveness of a particular learning algorithm is strongly influenced by the choice of such hyperparameters. A common method for selecting good hyperparameter values is via an exhaustive grid search over the hyperparameter space. While this yields acceptable results, it is often computationally expensive, and outright infeasible if the number of hyperparameters is large.

Recently, several methods have been proposed for optimizing hyperparameter selection, including cross-validation optimizations for kernel logistic regression [1], SVM [2] and conditional log-linear models [3]. These methods rely on re-training the model in the inner loop of the algorithm, after each hyperparameter update. Further, gradient-based methods have been successfully used in neural networks to optimize regularization parameters concurrently with the parameters of the model [4, 5]. In this paper, we investigate applying similar gradient-based techniques for adaptive hyperparameter optimization to L2-regularized logistic regression.

2 Preliminaries

We consider L2-regularized logistic regression, a solution to which maximize the log-likelyhood C over the training data set $\mathcal{T}$:

$$C(\theta) = \sum_{i=1}^{|\mathcal{T}|} log\ p(y^{(i)}|x^{(i)};\theta) - \lambda||\theta||^2 \tag{1}$$

Additionally, we use a stretched sigmoid approximation to the step function $s_\theta(x) = 1/(1 + exp(-\sigma_1\theta^T x))$ to define a validation function, V, as a smooth approximation to the generalization error over the hold-out validation set $\mathcal{V}$:

$$\begin{aligned} V(\theta) = &\ \tfrac{1}{|\mathcal{V}|}\textstyle\sum_{i=1}^{|\mathcal{V}|} 1\{s_\theta(x^{(i)}) \neq y^{(i)})\} \\ = &\ \tfrac{1}{|\mathcal{V}|}\textstyle\sum_{i=1}^{|\mathcal{V}|} s_\theta(x^{(i)})(1 - 2y^{(i)}) + y^{(i)} \end{aligned} \tag{2}$$

We use Newton's method for optimizing the model parameters:

$$\theta^{(k+1)} = \theta^{(k)} - H_C^{-1}(\theta^{(k)})\nabla_\theta C(\theta^{(k)}) \tag{3}$$

where $\theta^{(k)}$ is the value of θ after k iterations of the algorithm. Let diagonal matrices $\Lambda = \text{diag}(0, \lambda \dots)$ and $W = \text{diag}(h_{\theta^{(k)}}(x^{(i)})(1 - h_{\theta^{(k)}}(x^{(i)})), i \in \{1 \dots |\mathcal{T}|\})$, and input matrix $X = \left[x^{(1)} \dots x^{(n)}\right]^T, i \in \{1 \dots |\mathcal{T}|\}$. Then the gradient of the objective function can be written as

$$\nabla_\theta C(\theta^{(k)}) = \sum_{i=1}^{|\mathcal{T}|}(y^{(i)} - h_{\theta^{(k)}}(x^{(i)}))x^{(i)} - 2\Lambda\theta^{(k)} \tag{4}$$

and the Hessian matrix is

$$H_C(\theta^{(k)}) = -X^T W X - 2\Lambda \tag{5}$$

Training data set $\mathcal{T}$ and validation data set $\mathcal{V}$ are chosen to be non-overlapping subsets of the available data.

3 Derivation of hyperparameter update rule

We similarly use Newton's method to update the regularization parameter, λ, with the validation function used in place of the objective. Note that $\lambda^{(k+1)}$ is a function of the updated model parameters $\theta^{(k+1)}$, while $\theta^{(k)}$ is treated as a constant.

$$\lambda^{(k+1)} = \lambda^{(k)} - H_V^{-1}(\theta^{(k+1)}(\lambda^{(k)}))\nabla_\lambda V(\theta^{(k+1)}(\lambda^{(k)})) \tag{6}$$

Derivation of the gradient and Hessian in equation (6) is based on the method in [4].

3.1 Gradient

The gradient of the validation function with respect to λ is given by:

$$\nabla_\lambda V(\theta^{(k+1)}(\lambda^{(k)})) = \frac{\partial}{\partial\lambda^{(k)}}(\theta^{(k+1)})^T.\nabla_\theta V(\theta^{(k+1)}(\lambda^{(k)})) \tag{7}$$

Differentiating (2) with respect to $\theta^{(k+1)}$:

$$\nabla_\theta V(\theta^{(k+1)}(\lambda^{(k)})) = \frac{1}{|\mathcal{V}|}\sum_{i=1}^{|\mathcal{V}|} -\sigma_1 x^{(i)}(1 - 2y^{(i)})s_\theta(x^{(i)})(1 - s_\theta(x^{(i)})) \tag{8}$$

Differentiating (3) with respect to $\lambda^{(k)}$:

$$\frac{\partial}{\partial\lambda^{(k)}}(\theta^{(k+1)}) = -\left[\frac{\partial}{\partial\lambda^{(k)}}H_C^{-1}(\theta^{(k)})\right]\nabla_\theta C(\theta^{(k)}) - H_C^{-1}(\theta^{(k)})\left[\frac{\partial}{\partial\lambda^{(k)}}\nabla_\theta C(\theta^{(k)})\right] \tag{9}$$

To find the derivative $\frac{\partial}{\partial\lambda^{(k)}}H_C^{-1}(\theta^{(k)})$, we use the property of matrix inverse $H_C(\theta^{(k)})H_C^{-1}(\theta^{(k)}) = I$. Differentiating with respect to $\lambda^{(k)}$:

$$\left[\frac{\partial}{\partial\lambda^{(k)}}H_C(\theta^{(k)})\right]H_C^{-1}(\theta^{(k)}) + H_C(\theta^{(k)})\left[\frac{\partial}{\partial\lambda^{(k)}}H_C^{-1}(\theta^{(k)})\right] = 0 \tag{10}$$

Differentiating (5) with respect to $\lambda^{(k)}$:

$$\frac{\partial}{\partial \lambda^{(k)}} H_C(\theta^{(k)}) = 2I \tag{11}$$

From (9), (10) and (11) we now get:

$$\frac{\partial}{\partial \lambda^{(k)}} H_C^{-1}(\theta^{(k)}) = -2\left[H_C^{-1}(\theta^{(k)})\right]^2 \tag{12}$$

$$\frac{\partial}{\partial \lambda^{(k)}} \nabla_\theta C(\theta^{(k)}) = 2\theta^{(k)} \tag{13}$$

$$\begin{aligned} \tfrac{\partial}{\partial \lambda^{(k)}}(\theta^{(k+1)}) &= -2H_C^{-1}(\theta^{(k)})\left[-H_C^{-1}(\theta^{(k)})\nabla_\theta C(\theta^{(k)}) + \theta^{(k)}\right] \\ &= -2H_C^{-1}(\theta^{(k)})\theta^{(k+1)} \end{aligned} \tag{14}$$

Substituting (8) and (14) back into (7), we arrive at the equation for the gradient of the validation function with respect to λ:

$$\nabla_\lambda V(\theta^{(k+1)}(\lambda^{(k)})) = -\frac{2}{|\mathcal{V}|}\left[H_C^{-1}(\theta^{(k)})\theta^{(k+1)}\right]^T \sum_{i=1}^{|\mathcal{V}|} -\sigma_1 x^{(i)}(1-2y^{(i)})s_\theta(x^{(i)})(1-s_\theta(x^{(i)})) \tag{15}$$

3.2 Hessian

The Hessian of the validation function with respect to λ can be written as:

$$\begin{aligned} H_V(\theta^{(k+1)}(\lambda^{(k)})) = \; & -\tfrac{2}{|\mathcal{V}|}\left[\tfrac{\partial}{\partial \lambda^{(k)}}\left[H_C^{-1}(\theta^{(k)})\theta^{(k+1)}\right]^T\right] \textstyle\sum_{i=1}^{|\mathcal{V}|} -\sigma_1 x^{(i)}(1-2y^{(i)})s_\theta(x^{(i)})(1-s_\theta(x^{(i)})) \\ & -\tfrac{2}{|\mathcal{V}|}\left[\tfrac{\partial}{\partial \lambda^{(k)}} \textstyle\sum_{i=1}^{|\mathcal{V}|} -\sigma_1 x^{(i)}(1-2y^{(i)})s_\theta(x^{(i)})(1-s_\theta(x^{(i)}))\right] H_C^{-1}(\theta^{(k)})\theta^{(k+1)} \end{aligned} \tag{16}$$

Using (12) and (14):

$$\begin{aligned} \tfrac{\partial}{\partial \lambda^{(k)}}\left[H_C^{-1}(\theta^{(k)})\theta^{(k+1)}\right]^T &= \left[\left[\tfrac{\partial}{\partial \lambda^{(k)}} H_C^{-1}(\theta^{(k)})\right]\theta^{(k+1)} + H_C^{-1}(\theta^{(k)})\tfrac{\partial}{\partial \lambda^{(k)}}\theta^{(k+1)}\right]^T \\ &= -4\left[\left[H_C^{-1}(\theta^{(k)})\right]^2\theta^{(k+1)}\right]^T \end{aligned} \tag{17}$$

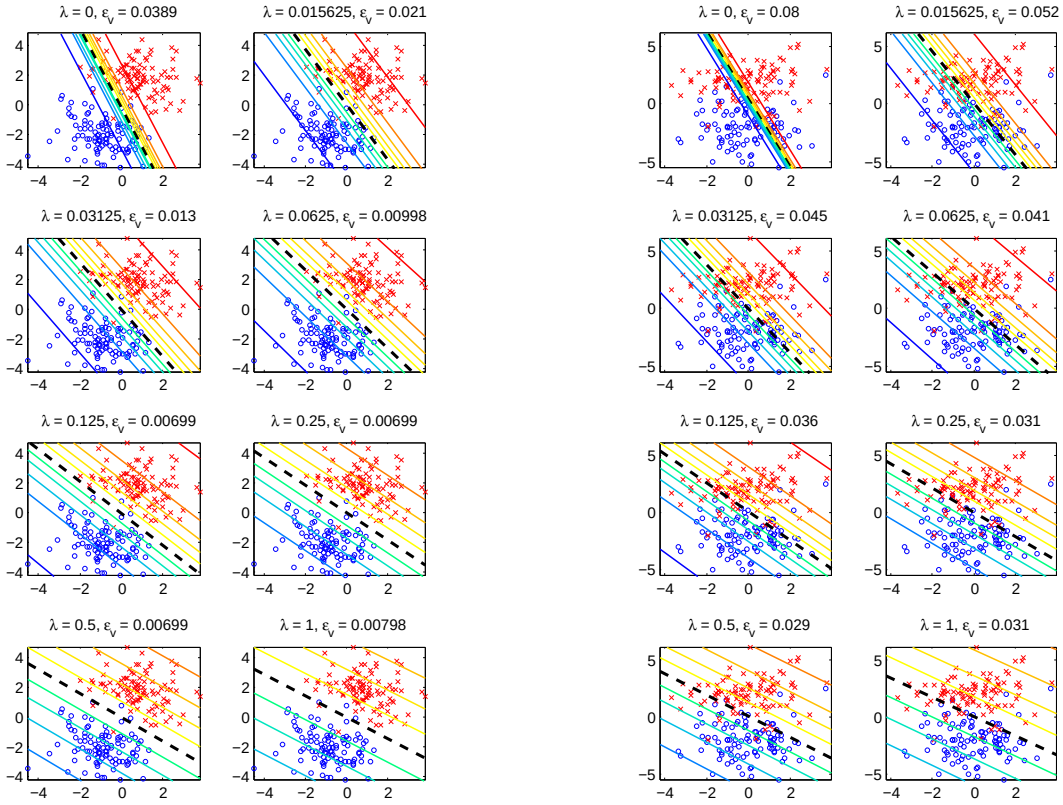
Returning to equation (16), the remaining differentiation can be obtained through the chain rule:

$$\begin{aligned} &\tfrac{\partial}{\partial \lambda^{(k)}} \textstyle\sum_{i=1}^{|\mathcal{V}|} -\sigma_1 x^{(i)}(1-2y^{(i)})s_\theta(x^{(i)})(1-s_\theta(x^{(i)})) \\ &\quad = \tfrac{\partial}{\partial \lambda^{(k)}}(\theta^{(k+1)})^T \tfrac{\partial}{\partial \theta^{(k+1)}} \textstyle\sum_{i=1}^{|\mathcal{V}|} -\sigma_1 x^{(i)}(1-2y^{(i)})s_\theta(x^{(i)})(1-s_\theta(x^{(i)})) \\ &\tfrac{\partial}{\partial \theta^{(k+1)}} \textstyle\sum_{i=1}^{|\mathcal{V}|} -\sigma_1 x^{(i)}(1-2y^{(i)})s_\theta(x^{(i)})(1-s_\theta(x^{(i)})) \\ &\quad = \textstyle\sum_{i=1}^{|\mathcal{V}|} -\sigma_1^2 x^{(i)T}x^{(i)}(1-2y^{(i)})s_\theta(x^{(i)})(1-s_\theta(x^{(i)}))(1-2s_\theta(x^{(i)})) \end{aligned} \tag{18}$$

Substituting (18) along with (17) into (16), we get the final form of the Hessian of the validation function with respect to λ:

$$\begin{aligned} H_V(\theta^{(k+1)}(\lambda^{(k)})) = \; & \tfrac{8}{|\mathcal{V}|}\left[H_C^{-1}(\theta^{(k)})^2\theta^{(k+1)}\right]^T \textstyle\sum_{i=1}^{|\mathcal{V}|} -\sigma_1 x^{(i)}(1-2y^{(i)})s_\theta(x^{(i)})(1-s_\theta(x^{(i)})) \\ + \; & \tfrac{4}{|\mathcal{V}|}\left[H_C^{-1}(\theta^{(k)})\theta^{(k+1)}\right]^T \\ & \cdot \left[\textstyle\sum_{i=1}^{|\mathcal{V}|} \sigma_1^2 x^{(i)T}x^{(i)}(1-2y^{(i)})s_\theta(x^{(i)})(1-s_\theta(x^{(i)}))(1-2s_\theta(x^{(i)}))\right]\left[H_C^{-1}(\theta^{(k)})\theta^{(k+1)}\right] \end{aligned} \tag{19}$$

The quantity $H_C^{-1}(\theta^{(k)})$ is already available from the parameter update step, so this calculation can be performed efficiently.



(a) 2-d, min error at $\lambda \in (0.125, 0.5)$

(b) 5-d (2 plotted), min error at $\lambda \in (0.25, 1.0)$

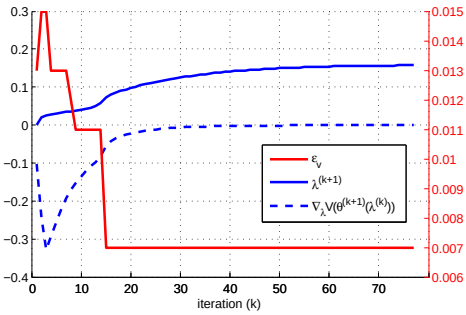
Figure 1: Grid search over the hyperparameter space in (a) two, and (b) five-dimensional data set.

4 Results

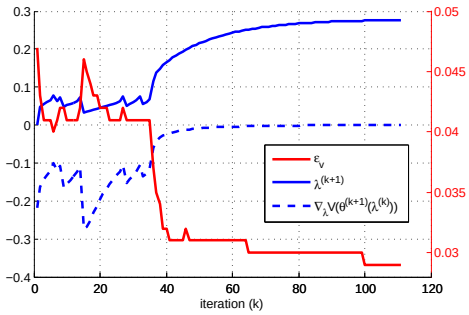
We compare the accuracy and performance of our algorithm to a grid search over the logarithmic space of hyperparamer values. Such a grid search over λ is shown in Figure 1, where each graph corresponds to training using a certain constant value of λ. The data sets we used for these results are generated from multi-variate Gaussian distributions with two label classes. We added a small amount of noise to the data to make the use of regularization beneficial. Further, the five-dimensional data set contains irrelevant features. In these data sets, there appears to be an optimal value of λ which minimises the generalization error.

Figure 2(a) shows the adaptation of λ during the training phase of our algorithm on a two-dimensional data set. The minimum value of generalization error reached is 0.00699 at a $\lambda = 0.1575$. Figure 2(b) shows the algorithm running on a five-dimensional data set, converging on an error of 0.029 at $\lambda = 0.2767$. These results match the approximate values obtained through grid search.

Comparing both methods in terms of speed, the grid search method took 54 and 57 iterations to converge for the two- and five-dimensional cases, respectively. On the other hand, the adaptation method took 63 and 111 iterations, respectively. We comment here that the grid search was done for just 8 values of λ where a larger number might be needed to get more accuracy. Additionally, this comparison is for a single hyperparameter. We expect that applying the same method to adapt multiple hyperparameters (such as the kernel parameters) would maintain a comparable level of performance, while the cost of grid search grows exponentially with the number of hyperparameters searched.



(a) 2-d, optimal $\lambda = 0.1575$ after 63 iterations



(b) 5-d, optimal $\lambda = 0.2767$ after 111 iterations

Figure 2: Learning λ for (a) two, and (b) five-dimensional data set.

5 Conclusion and further work

Gradient-based method of adapting hyperparameters during the learning phase was previously used successfully in neural networks. This work aimed to study the ability to extend the method to other supervised learning methods. Since the results of adapting the regularization parameter in L2-regularized logistic regression are promising, we think this method can be further extended to adapt multiple hyperparameters, such as the kernel parameters in kernel logistic regression.

It is not possible to apply this technique directly to kernel SVM, as SVM is usually solved in its dual form, where the optimization problem is constrained and not smooth with respect to the hyperparameters. Recently, however, there has been work on solving SVM in its primal form while maintaining the convenience of the kernel method [6]; it may be possible to apply the gradient technique in that setting.

References

[1] M. Seeger, "Cross-validation optimization for large scale hierarchical classification kernel methods," in *NIPS*, 2006.

[2] S. S. Keerthi, V. Sindhwani, and O. Chapelle, "An efficient method for gradient-based adaptation of hyperparameters in svm models," in *NIPS*, pp. 673–680, 2006.

[3] C. Do, C.-S. Foo, and A. Ng, "Efficient multiple hyperparameter learning for log-linear models," in *NIPS*, 2007.

[4] D. Chen and M. T. Hagan, "Optimal use of regularization and cross-validation in neural network modeling," *Neural Networks, 1999. IJCNN '99. International Joint Conference on*, vol. 2, pp. 1275–1280, Jul 1999.

[5] R. Eigenmann and J. Nossek, "Gradient based adaptive regularization," *Neural Networks for Signal Processing IX, 1999. Proceedings of the 1999 IEEE Signal Processing Society Workshop*, pp. 87–94, Aug 1999.

[6] O. Chapelle, "Training a support vector machine in the primal," *Neural Computation*, vol. 19, pp. 1155–1178, Mar 2007.