Convexity, Surrogate Functions and Iterative Optimization in Multi-class Logistic Regression Models

Zhihua Zhang, James T. Kwok, Dit-Yan Yeung, and Gang Wang

Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong
{zhzhang,jamesk,dyyeung,wanggang}@cs.ust.hk

Abstract. In this paper, we propose a family of surrogate maximization (SM) algorithms for multi-class logistic regression models (also called conditional exponential models). An SM algorithm aims at turning an otherwise intractable maximization problem into a tractable one by iterating two steps. The S-step computes a tractable surrogate function to substitute the original objective function, and the M-step seeks to maximize this surrogate function. We apply SM algorithms to logistic regression models, leading to the standard SM, generalized SM, gradient SM, and quadratic SM algorithms. Compared with Newton's method, these SM algorithms dramatically save computational costs when either the dimensionality or number of data samples is huge. Finally, we demonstrate the efficacy of these SM algorithms and compare their empirical performance on text categorization.

1 Introduction

In machine learning and statistics, many problems involve maximization or minimization and hence optimization techniques are very important for solving them. An objective function that has been widely used in the optimization process is the log-likelihood function. Since it is closely related to convex (or concave) functions, convexity often plays a central role. For example, the well-known *expectation maximization* (EM) algorithm [9] can be derived by using either Jensen's inequality or concavity of the logarithm function [15]. Another example is the *optimization transfer* algorithm [15] (also known as the *surrogate maximization* (SM) algorithm [18]), which optimizes a function serving as a surrogate for the original objective function. This surrogate function is often constructed by invoking convexity arguments.

The SM algorithm is very effective because it can make an otherwise intractable or complicated optimization problem tractable or significantly simpler. For example, it can decouple the correlation among parameters so that one can estimate the parameters in parallel, or it can avoid the computational burden of inverting large matrices often associated with Newton's method. Moreover, the SM algorithm enjoys the same local convergence properties as the standard EM algorithm. In this paper, we attempt to demonstrate the power and potential of SM algorithms in machine learning, by using logistic regression models as a specific example.

In particular, we address two major issues in devising SM algorithms, namely, how a surrogate function is defined and how the resultant surrogate function is maximized.

On the first problem, there exist three main approaches, namely, by using Jensen's inequality, first-order Taylor approximation, and the low quadratic bound principle. The first two approaches follow readily from properties of convex functions, while the third uses a quadratic function to approximate the original objective function. On the second problem, in general different maximization methods are required for different surrogate functions. In this paper, this leads to the standard SM, generalized SM, gradient SM, and quadratic SM algorithms as will be detailed in the sequel.

Logistic regression models, also called conditional exponential models in the machine learning community, are closely related to the maximum entropy principle [2, 8]. Recently, they have been successfully applied to data mining and information retrieval problems. An appealing characteristic of logistic regression models is that they can predict the class labels by combining evidences from many correlated input features. To a certain extent, this resembles boosting, which improves the accuracy of a given learning algorithm by combining many weak learners to form a powerful committee or ensemble. In fact, the pioneering work of [10] established a relationship between boosting and logistic regression models, which was then further developed in [16].

Given a training data set, we consider in this paper the problem of finding an optimal logistic regression model using different SM algorithms. One popular solution is the use of generalized iterative scaling (GIS) [7] and its variants, such as improved iterative scaling (IIS) [1] and faster iterative scaling (FIS) algorithms [12]. The basic idea of these methods is to approximate the intractable log-likelihood function of the conditional exponential model by an *auxiliary function*, in which the model parameters are decoupled. Consequently, it becomes tractable to optimize the auxiliary function rather than the log-likelihood function. We can see that surrogate functions play the same role as auxiliary functions in iterative scaling. However, the difference lies in the fact that an auxiliary function is used to approximate the difference of the log-likelihood function values computed based on the parameters in two consecutive iterations. Recently, some optimization methods based on the Bregman distance have been proposed [5, 8, 13]. Bregman distance-based optimization algorithms work with the first-order Taylor expansion of a convex function, the argument of which is itself also a function. Therefore these algorithms share some common principles with SM algorithms.

The rest of this paper is organized as follows. Sections 2 and 3 give brief overviews on the SM algorithms and logistic regression models, respectively. In Section 4, we then devise different SM algorithms for the logistic regression models. Section 5 presents the experimental results and the last section gives some concluding remarks.

2 The Surrogate Maximization Algorithm

We consider the general problem of maximizing an arbitrary function $L(\boldsymbol{\theta})$ w.r.t. some parameter $\boldsymbol{\theta}$. Given an estimate $\boldsymbol{\theta}(t)$ at the tth iteration, the SM algorithm [15, 17] consists of the following two steps:

Surrogate Step (S-Step): Substitute a surrogate function $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}(t))$ for $L(\boldsymbol{\theta})$, such that

$$L(\boldsymbol{\theta}) \geq Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}(t)) \tag{1}$$

for all $\boldsymbol{\theta}$, with equality holding at $\boldsymbol{\theta} = \boldsymbol{\theta}(t)$.

Maximization Step (M-Step): Obtain the next parameter estimate $\boldsymbol{\theta}(t+1)$ by maximizing the surrogate function $Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}(t))$ w.r.t. $\boldsymbol{\theta}$, i.e.,

$$\boldsymbol{\theta}(t+1) = \arg\max_{\boldsymbol{\theta}} Q(\boldsymbol{\theta} \mid \boldsymbol{\theta}(t)). \tag{2}$$

Clearly, construction of the surrogate function is key to the SM algorithm in turning an otherwise intractable optimization problem into a tractable one. On the one hand, the closer is the surrogate function to $L(\boldsymbol{\theta})$, the more efficient is the SM algorithm. On the other hand, a good surrogate function should preferably have a closed-form solution in the M-step. By invoking convexity arguments, a general principle providing guidelines on constructing surrogate functions, as well as some specific examples for special cases, have been discussed in [15] and [18]. Depending upon the context, this often relies on three important techniques, namely, *Jensen's inequality*, *first-order Taylor approximation*, and the *low quadratic bound principle* [4]. Combinations of these methods can also be used [18].

Depending on the surrogate functions obtained, different SM algorithms can be devised accordingly. In the *standard SM algorithm*, a closed-form solution for $\boldsymbol{\theta}(t+1)$ in the M-step exists. Then, as

$$L(\boldsymbol{\theta}(t+1)) \geq Q(\boldsymbol{\theta}(t+1)|\boldsymbol{\theta}(t)) \geq Q(\boldsymbol{\theta}(t)|\boldsymbol{\theta}(t)) = L(\boldsymbol{\theta}(t)),$$

the standard SM algorithm enjoys the same local convergence properties as the standard EM algorithm.

However, it is not always possible to obtain a closed-form solution for $\boldsymbol{\theta}(t+1)$ in the M-step. In the same spirit as the generalized EM algorithm [9], we can devise a *generalized SM algorithm*. That is, instead of maximizing $Q(\boldsymbol{\theta}|\boldsymbol{\theta}(t))$, we only attempt to find a $\boldsymbol{\theta}(t+1)$ such that $Q(\boldsymbol{\theta}(t+1)|\boldsymbol{\theta}(t)) \geq Q(\boldsymbol{\theta}(t)|\boldsymbol{\theta}(t))$. Obviously, the generalized SM algorithm is also locally convergent. Alternatively, in the same spirit as the gradient EM algorithm [14], we may also devise a *gradient SM algorithm*, as

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - (\nabla^2 Q(\boldsymbol{\theta}(t)|\boldsymbol{\theta}(t)))^{-1} \nabla L(\boldsymbol{\theta}(t)).$$

Finally, application of the low quadratic bound principle mentioned above leads to the *quadratic SM algorithm*.

3 Logistic Regression Models

In a logistic regression model, the conditional likelihood $p(y|\mathbf{x})$ is usually given by

$$p(y|\mathbf{x}) = \frac{1}{Z(\mathbf{x})} \exp\Big(\sum_{j=1}^{m} \lambda_j f_j(\mathbf{x}, y)\Big), \tag{3}$$

where $f_j(\mathbf{x}, y)$ stands for the jth feature extracted from input $\mathbf{x}$ and output class label y, λ_j is a real-valued weight[1] measuring the importance of feature $f_j(\mathbf{x}, y)$, and $Z(\mathbf{x})$

[1] Technically, λ_j can be considered as a Lagrange multiplier corresponding to $f_j(\mathbf{x}, y)$ in a certain constrained optimization problem.

is the normalization term that enforces the sum of $p(y|\mathbf{x})$ over different class labels y's to be unity. In other words,

$$Z(\mathbf{x}) = \sum_y \exp\Big(\sum_{j=1}^m \lambda_j f_j(\mathbf{x}, y)\Big).$$

In this paper, we focus on the multi-class problem. Let $\mathcal{T} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$ be a finite set of training examples, where each instance $\mathbf{x}_k$ comes from a domain or instance space $\mathcal{X}$ and $y_k \in \{1, 2, \ldots, c\}$ is the corresponding class label (from one of the c possible labels). We also assume that we are given a set of real-valued feature functions, $f_1, \ldots, f_m$, on $\mathcal{X}$. Along the line of [12], we assume that all classes share the same set of features $\{f_j(\mathbf{x}_k)\}$, leading to the following simplified parametric model

$$p(y_k = i|\mathbf{x}_k, \Lambda) = p(i|\mathbf{x}_k, \Lambda) = \frac{1}{Z(\mathbf{x}_k)} \exp\Big(\sum_{j=1}^m \lambda_{ij} f_j(\mathbf{x}_k)\Big), \tag{4}$$

where $\Lambda = \{\lambda_{ij}\}$. In the sequel, we will also write Λ in matrix form, as $\boldsymbol{\Lambda} = \big[\lambda_{ij}\big]_{c\times m}$.

The training procedure aims at finding a set of weights $\{\lambda_{ij}\}$ that maximizes the log-likelihood of the training data. Given the empirical joint density $\tilde{p}(\mathbf{x}, y)$ estimated from the training data, the *log-likelihood* of $p(y|\mathbf{x})$ w.r.t. $\tilde{p}(\mathbf{x}, y)$ can be expressed as

$$\begin{aligned} L(\Lambda) &= \sum_{k,i} \tilde{p}(\mathbf{x}_k, i) \ln p(i|\mathbf{x}_k, \Lambda) \\ &= \sum_{k,i} \tilde{p}(\mathbf{x}_k, i) \sum_{j=1}^m \lambda_{ij} f_j(\mathbf{x}_k) - \sum_k \tilde{p}(\mathbf{x}_k) \ln\Big(\sum_i e^{\sum_{j=1}^m \lambda_{ij} f_j(\mathbf{x}_k)}\Big) \\ &= \sum_{k,i} \tilde{p}(\mathbf{x}_k)\tilde{p}(i|\mathbf{x}_k)\boldsymbol{\lambda}_i^T \mathbf{f}(\mathbf{x}_k) - \sum_k \tilde{p}(\mathbf{x}_k) \ln\Big(\sum_i e^{\boldsymbol{\lambda}_i^T \mathbf{f}(\mathbf{x}_k)}\Big), \end{aligned} \tag{5}$$

where $\tilde{p}(\mathbf{x})$ is the empirical density for input $\mathbf{x}$, $\boldsymbol{\lambda}_i = (\lambda_{i1}, \ldots, \lambda_{im})^T$, and $\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))^T$. Denote $L_k(\Lambda) = \sum_i \tilde{p}(i|\mathbf{x}_k)\boldsymbol{\lambda}_i^T \mathbf{f}(\mathbf{x}_k) - \ln\Big(\sum_i e^{\boldsymbol{\lambda}_i^T \mathbf{f}(\mathbf{x}_k)}\Big)$. Then $L(\Lambda) = \sum_k \tilde{p}(\mathbf{x}_k) L_k(\Lambda)$.

Since it is intractable to directly optimize $L(\Lambda)$ w.r.t. Λ, [1] proposed an improved iterative scaling (IIS) algorithm, which is a modified version of the generalized iterative scaling algorithm [7]. More recently, a faster iterative scaling (FIS) algorithm was also proposed for this problem [12]. Let $\Lambda = \{\lambda_{ij}\}$ and $\tilde{\Lambda} = \{\tilde{\lambda}_{ij}\}$ be the parameter values at two consecutive iterations. Both IIS and FIS proceed by replacing $L(\Lambda) - L(\tilde{\Lambda})$ with a lower bound auxiliary function, which is then optimized w.r.t. $\Delta = \{\delta_{ij}\}$, where $\delta_{ij} = \lambda_{ij} - \tilde{\lambda}_{ij}$. Typically, the auxiliary function decouples the correlation among the parameters $\{\lambda_{ij}\}$ and thus makes the optimization problem tractable. While in general there exist many auxiliary functions that can be used, a reasonable criterion that has been used successfully by FIS in choosing the auxiliary function is by measuring its closeness to the original objective function.

4 SM Algorithms for Logistic Regression Models

In this Section, we employ the three surrogate function construction methods discussed in Section 2 to devise SM algorithms for logistic regression models. Without loss of generality[2], we assume that $f_i(\mathbf{x}_k) \geq 0$, $\forall i$ and

$$\sum_{i=1}^{m} f_i(\mathbf{x}_k) \leq 1. \tag{6}$$

4.1 Standard SM Algorithm

In this Section, we use a combination of Jensen's inequality and Taylor approximation to construct the surrogate function. First, using first-order Taylor approximation on the concave function $\ln(\cdot)$, we have

$$\ln\Big(\sum_i e^{\boldsymbol{\lambda}_i^T \mathbf{f}(\mathbf{x}_k)}\Big) \leq \ln\Big(\sum_i e^{\boldsymbol{\lambda}_i^T(t)\mathbf{f}(\mathbf{x}_k)}\Big) + \sum_i p(i|\mathbf{x}_k, \Lambda(t))\Big(e^{(\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_i(t))^T\mathbf{f}(\mathbf{x}_k)} - 1\Big). \tag{7}$$

Next, it follows from Jensen's inequality on the convex function $\exp(\cdot)$ that

$$e^{(\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_i(t))^T\mathbf{f}(\mathbf{x}_k)} \leq \sum_{j=1}^{m} f_j(\mathbf{x}_k) e^{\lambda_{ij} - \lambda_{ij}(t)} + 1 - \sum_{j=1}^{m} f_j(\mathbf{x}_k). \tag{8}$$

Substituting inequalities (7) and (8) back to (5), this leads to the following surrogate function for $L(\Lambda)$:

$$\begin{aligned} Q_M(\Lambda|\Lambda(t)) = &\sum_{k,i} \tilde{p}(\mathbf{x}_k, i) \sum_{j=1}^{m} \lambda_{ij} f_j(\mathbf{x}_k) - \sum_k \tilde{p}(\mathbf{x}_k) \ln\Big(\sum_i e^{\sum_{j=1}^m \lambda_{ij}(t) f_j(\mathbf{x}_k)}\Big) \\ &- \sum_{k,i} \tilde{p}(\mathbf{x}_k) p(i|\mathbf{x}_k, \Lambda(t)) \sum_{j=1}^{m} f_j(\mathbf{x}_k)\Big(e^{\lambda_{ij} - \lambda_{ij}(t)} - 1\Big). \end{aligned} \tag{9}$$

Alternatively, one may apply the Taylor approximation and Jensen's inequality in reverse order. First, it is easy to show that the *log-sum-exp* function $\ln(\sum_i e^{u_i})$ is convex. Using Jensen's inequality on this function, we then have

$$\begin{aligned} &\ln \sum_i e^{\boldsymbol{\lambda}_i^T \mathbf{f}(\mathbf{x}_k)} \\ &= \ln \sum_i e^{\left\{\sum_{j=1}^m f_j(\mathbf{x}_k)[\lambda_{ij} - \lambda_{ij}(t) + \boldsymbol{\lambda}_i^T(t)\mathbf{f}(\mathbf{x}_k)] + (1 - \sum_{j=1}^m f_j(\mathbf{x}_k))\boldsymbol{\lambda}_i^T(t)\mathbf{f}(\mathbf{x}_k)\right\}} \\ &\leq \sum_{j=1}^{m} f_j(\mathbf{x}_k) \ln\Big(\sum_i e^{[\lambda_{ij} - \lambda_{ij}(t) + \boldsymbol{\lambda}_i^T(t)\mathbf{f}(\mathbf{x}_k)]}\Big) \\ &\quad + \Big(1 - \sum_{j=1}^{m} f_j(\mathbf{x}_k)\Big) \ln\Big(\sum_i e^{\boldsymbol{\lambda}_i^T(t)\mathbf{f}(\mathbf{x}_k)}\Big). \end{aligned} \tag{10}$$

[2] Since the features $f_i(\mathbf{x}_k)$'s are prespecified, these conditions can be easily satisfied by the transformation $f_i(\mathbf{x}_k) \leftarrow f_i(\mathbf{x}_k)/\sum_j f_j(\mathbf{x}_k)$.

We then apply the first-order Taylor approximation on the concave function $\ln(\cdot)$ and obtain

$$\ln\Big(\sum_i e^{[\lambda_{ij}-\lambda_{ij}(t)+\boldsymbol{\lambda}_i^T(t)\mathbf{f}(\mathbf{x}_k)]}\Big) \leq \ln\Big(\sum_i e^{\boldsymbol{\lambda}_i^T(t)\mathbf{f}(\mathbf{x}_k)}\Big) + \sum_i p(i|\mathbf{x}_k, \Lambda(t))(e^{\lambda_{ij}-\lambda_{ij}(t)}-1). \quad (11)$$

Substituting (10) and (11) back to (5), this leads to the same surrogate function (9).

Next, we consider maximizing $Q_M(\Lambda|\Lambda(t))$ w.r.t. Λ as required in the M-step. On setting the derivative

$$\frac{\partial Q_M(\Lambda|\Lambda(t))}{\partial \lambda_{ij}} = \sum_k \tilde{p}(\mathbf{x}_k, i) f_j(\mathbf{x}_k) - \sum_k \tilde{p}(\mathbf{x}_k) p(i|\mathbf{x}_k, \Lambda(t)) f_j(\mathbf{x}_k) e^{\lambda_{ij}-\lambda_{ij}(t)}$$

to zero, we obtain a closed-form solution and hence the M-step update can be given by

$$\lambda_{ij}(t+1) = \lambda_{ij}(t) + \ln \frac{\sum_k \tilde{p}(\mathbf{x}_k)\tilde{p}(i|\mathbf{x}_k) f_j(\mathbf{x}_k)}{\sum_k \tilde{p}(\mathbf{x}_k) p(i|\mathbf{x}_k, \Lambda(t)) f_j(\mathbf{x}_k)}. \quad (12)$$

4.2 Generalized SM Algorithm

Note that using only either (10) or (11) leads to two other surrogate functions for $L(\Lambda)$, as

$$Q_J(\Lambda|\Lambda(t)) = \sum_{k,i} \tilde{p}(\mathbf{x}_k, i)\boldsymbol{\lambda}_i^T \mathbf{f}(\mathbf{x}_k) - \sum_k \tilde{p}(\mathbf{x}_k)\Big(1 - \sum_{j=1}^m f_j(\mathbf{x}_k)\Big) \ln\Big(\sum_i e^{\boldsymbol{\lambda}_i^T(t)\mathbf{f}(\mathbf{x}_k)}\Big) - \sum_k \tilde{p}(\mathbf{x}_k) \sum_{j=1}^m f_j(\mathbf{x}_k) \ln\Big(\sum_i e^{[\lambda_{ij}-\lambda_{ij}(t)+\boldsymbol{\lambda}_i^T(t)\mathbf{f}(\mathbf{x}_k)]}\Big), \quad (13)$$

and

$$Q_T(\Lambda|\Lambda(t)) = \sum_{k,i} \tilde{p}(\mathbf{x}_k, i)\boldsymbol{\lambda}_i^T \mathbf{f}(\mathbf{x}_k) - \sum_k \tilde{p}(\mathbf{x}_k) \ln\Big(\sum_i e^{\boldsymbol{\lambda}_i^T(t)\mathbf{f}(\mathbf{x}_k)}\Big) + 1 - \sum_{k,i} \tilde{p}(\mathbf{x}_k) p(i|\mathbf{x}_k, \Lambda(t)) e^{(\boldsymbol{\lambda}_i - \boldsymbol{\lambda}_i(t))^T \mathbf{f}(\mathbf{x}_k)}. \quad (14)$$

However, closed-form solutions for $\arg\max_\Lambda Q_J(\Lambda|\Lambda(t))$ and $\arg\max_\Lambda Q_T(\Lambda|\Lambda(t))$ cannot be found. Nevertheless, we have

$$L(\Lambda) \geq Q_J(\Lambda|\Lambda(t)) \geq Q_M(\Lambda|\Lambda(t)),$$

and

$$L(\Lambda) \geq Q_T(\Lambda|\Lambda(t)) \geq Q_M(\Lambda|\Lambda(t)).$$

As a result, it is easy to show that

$$Q_J(\Lambda(t+1)|\Lambda(t)) \geq Q_M(\Lambda(t+1)|\Lambda(t)) \geq Q_M(\Lambda(t)|\Lambda(t)) = L(\Lambda(t)) = Q_J(\Lambda(t)|\Lambda(t)),$$

and

$$Q_T(\Lambda(t+1)|\Lambda(t)) \geq Q_M(\Lambda(t+1)|\Lambda(t)) \geq Q_M(\Lambda(t)|\Lambda(t)) = L(\Lambda(t)) = Q_T(\Lambda(t)|\Lambda(t)).$$

Thus, the iterative algorithm in (12) is a generalized SM algorithm for either $Q_J(\Lambda|\Lambda(t))$ or $Q_T(\Lambda|\Lambda(t))$.

4.3 Gradient SM Algorithms

As mentioned in Section 2, instead of using a generalized SM algorithm for the surrogate functions $Q_T(\Lambda|\Lambda(t))$ and $Q_J(\Lambda|\Lambda(t))$, we can also devise a gradient SM algorithm. As we can see, $Q_M(\Lambda|\Lambda(t))$ completely decouples the correlation among λ_{ij}'s, but $Q_T(\Lambda|\Lambda(t))$ and $Q_J(\Lambda|\Lambda(t))$ only decouple the correlation among λ_{ij}'s in the column and row directions, respectively. Now,

$$\left.\frac{\partial Q_T(\Lambda|\Lambda(t))}{\partial \boldsymbol{\lambda}_i}\right|_{\boldsymbol{\lambda}_i=\boldsymbol{\lambda}_i(t)} = \sum_k \tilde{p}(\mathbf{x}_k)\big(\tilde{p}(i|\mathbf{x}_k) - p(i|\mathbf{x}_k, \Lambda(t))\big)\mathbf{f}(\mathbf{x}_k) = F_i(\Lambda(t)),$$

$$\left.\frac{\partial^2 Q_T(\Lambda|\Lambda(t))}{\partial \boldsymbol{\lambda}_i \partial \boldsymbol{\lambda}_i^T}\right|_{\boldsymbol{\lambda}_i=\boldsymbol{\lambda}_i(t)} = -\sum_k \tilde{p}(\mathbf{x}_k) p(i|\mathbf{x}_k, \Lambda(t))\mathbf{f}(\mathbf{x}_k)\mathbf{f}^T(\mathbf{x}_k) = H_i(\Lambda(t)),$$

the iterative procedure of the resultant gradient SM algorithm for $Q_T(\Lambda|\Lambda(t))$ is

$$\boldsymbol{\lambda}_i(t+1) = \boldsymbol{\lambda}_i(t) - \big(H_i(\Lambda(t))\big)^{-1} F_i(\Lambda(t)), \quad \text{for } i = 1, \ldots, c. \tag{15}$$

Similarly, on denoting $\boldsymbol{\lambda}_{\cdot j} = (\lambda_{1j}, \ldots, \lambda_{cj})^T$, we obtain

$$\left.\frac{\partial Q_J(\Lambda|\Lambda(t))}{\partial \boldsymbol{\lambda}_{\cdot j}}\right|_{\boldsymbol{\lambda}_{\cdot j}=\boldsymbol{\lambda}_{\cdot j}(t)} = \sum_k \tilde{p}(\mathbf{x}_k) f_j(\mathbf{x}_k)(\tilde{\mathbf{q}}_k - \mathbf{q}_k(t)) = F_{\cdot j}(\Lambda(t)),$$

$$\left.\frac{\partial^2 Q_J(\Lambda|\Lambda(t))}{\partial \boldsymbol{\lambda}_{\cdot j} \partial \boldsymbol{\lambda}_{\cdot j}^T}\right|_{\boldsymbol{\lambda}_{\cdot j}=\boldsymbol{\lambda}_{\cdot j}(t)} = -\sum_k \tilde{p}(\mathbf{x}_k) f_j(\mathbf{x}_k)\big(\text{diag}(\mathbf{q}_k(t)) - \mathbf{q}_k(t)\mathbf{q}_k^T(t)\big) = H_{\cdot j}(\Lambda(t)),$$

where $\tilde{\mathbf{q}}_k = (\tilde{p}(1|\mathbf{x}_k), \ldots, \tilde{p}(c|\mathbf{x}_k))^T$ and $\mathbf{q}_k(t) = \big(p(1|\mathbf{x}_k, \Lambda(t)), \ldots, p(c|\mathbf{x}_k, \Lambda(t))\big)^T$. Then the gradient SM algorithm for $Q_J(\Lambda|\Lambda(t))$ updates the current parameter estimate $\lambda_j(t)$ through

$$\boldsymbol{\lambda}_{\cdot j}(t+1) = \boldsymbol{\lambda}_{\cdot j}(t) - \big(H_{\cdot j}(\Lambda(t))\big)^{-1} F_{\cdot j}(\Lambda(t)), \quad \text{for } j = 1, \ldots, m. \tag{16}$$

Note that the matrices H_i and $H_{\cdot j}$ are of sizes $m \times m$ and $c \times c$, respectively. Therefore, in order to update Λ, the gradient step in (15) needs to invert c $m \times m$ matrices, while that in (16) needs to invert m $c \times c$ matrices.

4.4 Quadratic SM Algorithm

Let $L(\boldsymbol{\theta})$ be the objective function, $\nabla L(\boldsymbol{\theta})$ the Fisher score vector, and $\nabla^2 L(\boldsymbol{\theta})$ the second derivative Hessian matrix at $\boldsymbol{\theta} \in \mathbb{R}^r$. Böhning and Lindsay [4] proposed the quadratic lower bound algorithm under the assumption that a negative definite $r \times r$ matrix $\mathbf{B}$ can be found such that $\nabla^2 L(\boldsymbol{\theta}) \succeq \mathbf{B}$ for all $\boldsymbol{\theta}$.[3] Thus, we can define the

[3] Here $\mathbf{C} \succeq \mathbf{D}$ means $\mathbf{C} - \mathbf{D}$ is positive semi-definite.

surrogate function $Q(\boldsymbol{\theta} \mid \boldsymbol{\phi})$ of $L(\boldsymbol{\theta})$ as

$$Q(\boldsymbol{\theta}|\boldsymbol{\phi}) = L(\boldsymbol{\phi}) + (\boldsymbol{\theta} - \boldsymbol{\phi})^T \nabla L(\boldsymbol{\phi}) + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\phi})^T \mathbf{B}(\boldsymbol{\theta} - \boldsymbol{\phi}).$$

It is clear that $L(\boldsymbol{\theta}) - Q(\boldsymbol{\theta}|\boldsymbol{\phi})$ attains its minimum at $\boldsymbol{\theta} = \boldsymbol{\phi}$. Since $Q(\boldsymbol{\theta}|\boldsymbol{\phi})$ is a quadratic function, its convexity shows that it has only one maximum. The SM algorithm seeks to approximate the maximum of $L(\boldsymbol{\theta})$ with that of $Q(\boldsymbol{\theta}|\boldsymbol{\phi})$ through an iterative procedure. If we let $\boldsymbol{\phi}$ be the tth estimate of $\boldsymbol{\theta}$, denoted $\boldsymbol{\theta}^{(t)}$, then maximizing $Q(\boldsymbol{\theta}|\boldsymbol{\theta}^{(t)})$ w.r.t. $\boldsymbol{\theta}$ yields the $(t+1)$th estimate of $\boldsymbol{\theta}$ as

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \mathbf{B}^{-1} \nabla L(\boldsymbol{\theta}^{(t)}). \tag{17}$$

This shows that the quadratic lower bound algorithm amounts to maximizing $L(\boldsymbol{\theta})$ by using Newton's method but with the Hessian matrix $\nabla^2 L(\boldsymbol{\theta})$ replaced by $\mathbf{B}$. Let $\boldsymbol{\lambda} = (\lambda_{11}, \ldots, \lambda_{1m}, \ldots, \lambda_{c1}, \ldots, \lambda_{cm})^T$. Since

$$\frac{\partial L_k(\Lambda)}{\partial \lambda_{ij}} = (\tilde{p}(i|\mathbf{x}_k) - p(i|\mathbf{x}_k, \Lambda)) f_j(\mathbf{x}_k),$$

this gives rise to the score vector

$$\nabla L_k(\Lambda) = (\tilde{\mathbf{q}}_k - \mathbf{q}_k) \otimes \mathbf{f}(\mathbf{x}_k),$$

where the symbol $\otimes$ stands for the Kronecker product [11] of two arbitrary matrices. In addition, the second partial derivative of $L_k(\Lambda)$ is computed by

$$\frac{\partial^2 L_k(\Lambda)}{\partial \lambda_{ij} \partial \lambda_{i'j'}} = \begin{cases} -p(i|\mathbf{x}_k, \Lambda)(1 - p(i|\mathbf{x}_k, \Lambda)) f_j(\mathbf{x}_k) f_{j'}(\mathbf{x}_k) & i = i', \\ p(i'|\mathbf{x}_k, \Lambda) p(i|\mathbf{x}_k, \Lambda) f_j(\mathbf{x}_k) f_{j'}(\mathbf{x}_k) & i \neq i', \end{cases}$$

leading to the Hessian matrix

$$\nabla^2 L_k = -(\text{diag}(\mathbf{q}_k) - \mathbf{q}_k \mathbf{q}_k^T) \otimes \mathbf{f}(\mathbf{x}_k) \mathbf{f}^T(\mathbf{x}_k).$$

Thus, we have

$$\nabla^2 L(\Lambda) = \sum_k \tilde{p}(\mathbf{x}_k) \nabla^2 L_k = -\sum_k \tilde{p}(\mathbf{x}_k) [(\text{diag}(\mathbf{q}_k) - \mathbf{q}_k \mathbf{q}_k^T) \otimes \mathbf{f}(\mathbf{x}_k) \mathbf{f}^T(\mathbf{x}_k)].$$

This leads us to Newton's method, as

$$\boldsymbol{\lambda}(t+1) = \boldsymbol{\lambda}(t) - (\nabla^2 L(\boldsymbol{\lambda}(t)))^{-1} \nabla L(\boldsymbol{\lambda}(t)). \tag{18}$$

Now using the following inequality [3, 4]

$$\text{diag}(\mathbf{q}_k) - \mathbf{q}_k \mathbf{q}_k^T \preceq \frac{1}{2} \left[\mathbf{I} - \frac{1}{c} \mathbf{1}\mathbf{1}^T \right],$$

where $\mathbf{1}$ is the $c \times 1$ matrix (i.e., column vector) of ones, we obtain

$$\begin{aligned}\nabla^2 L(\Lambda) &\succeq -\frac{1}{2}\sum_{k=1}^{n}\tilde{p}(\mathbf{x}_k)\left\{\left[\mathbf{I}-\frac{1}{c}\mathbf{1}\mathbf{1}^T\right]\otimes\mathbf{f}(\mathbf{x}_k)\mathbf{f}^T(\mathbf{x}_k)\right\}\\ &= -\frac{1}{2}\left[\mathbf{I}-\frac{1}{c}\mathbf{1}\mathbf{1}^T\right]\otimes\sum_{k=1}^{n}\tilde{p}(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)\mathbf{f}^T(\mathbf{x}_k)\\ &= -\frac{1}{2}\left[\mathbf{I}-\frac{1}{c}\mathbf{1}\mathbf{1}^T\right]\otimes(\mathbf{F}\tilde{\mathbf{D}}\mathbf{F}^T)\triangleq\mathbf{B},\end{aligned}$$

where $\mathbf{F}=\left[f_j(\mathbf{x}_k)\right]_{c\times n}$ and $\tilde{\mathbf{D}}=\mathrm{diag}(\tilde{p}(\mathbf{x}_1),\ldots,\tilde{p}(\mathbf{x}_n))$. We have an iterative procedure of resolving $\boldsymbol{\lambda}$ as

$$\boldsymbol{\lambda}(t+1)=\boldsymbol{\lambda}(t)-\mathbf{B}^{-1}\sum_{k=1}^{n}\tilde{p}(\mathbf{x}_k)\left[(\tilde{\mathbf{q}}_k-\mathbf{q}_k(t))\otimes\mathbf{f}(\mathbf{x}_k)\right]. \tag{19}$$

As we can see, the algorithm described by (19) is just a standard SM algorithm w.r.t. the following surrogate function:

$$Q_Q(\Lambda|\Lambda(t))=L(\boldsymbol{\Lambda}(t))+(\boldsymbol{\lambda}-\boldsymbol{\lambda}(t))^T\nabla L(\boldsymbol{\Lambda}(t))+\frac{1}{2}(\boldsymbol{\lambda}-\boldsymbol{\lambda}(t))^T\mathbf{B}(\boldsymbol{\lambda}-\boldsymbol{\lambda}(t)). \tag{20}$$

Here, we refer to it as the quadratic SM algorithm because its derivation is based on the low quadratic bound principle. Note that for both the quadratic SM and Newton's method, it is not necessary to require $f_i(\mathbf{x}_k)\geq 0, \forall i$ and $\sum_{i=1}^{m} f_i(\mathbf{x}_k)\leq 1$.

If let $\mathbf{H}=\left[\mathbf{I}-\frac{1}{c}\mathbf{1}\mathbf{1}^T\right]\otimes(\mathbf{F}\tilde{\mathbf{D}}\mathbf{F}^T)$, then $\mathbf{H}^{-1}=\left[\mathbf{I}-\frac{1}{c}\mathbf{1}\mathbf{1}^T\right]^{+}\otimes(\mathbf{F}\tilde{\mathbf{D}}\mathbf{F}^T)^{-1}$, where the superscript $^{+}$ stands for the Moore-Penrose inverse. On the other hand, it is easy to obtain that

$$\begin{aligned}\sum_{k=1}^{n}\tilde{p}(\mathbf{x}_k)\left[(\tilde{\mathbf{q}}_k-\mathbf{q}_k(t))\otimes\mathbf{f}(\mathbf{x}_k)\right] &= \mathrm{vec}\Big(\sum_{k=1}^{n}\tilde{p}(\mathbf{x}_k)\mathbf{f}(\mathbf{x}_k)(\tilde{\mathbf{q}}_k-\mathbf{q}_k(t))^T\Big)\\ &= \mathrm{vec}\Big(\mathbf{F}\tilde{\mathbf{D}}(\tilde{\mathbf{Q}}-\mathbf{Q}_t)\Big),\end{aligned}$$

where the notation $\mathrm{vec}(\cdot)$ denotes the $st\times 1$ vector formed by stacking the columns of an $s\times t$ matrix [11], $\tilde{\mathbf{Q}}=\left[\tilde{p}(i|\mathbf{x}_k)\right]_{n\times c}$ and $\mathbf{Q}_t=\left[p(i|\mathbf{x}_k,\Lambda(t))\right]_{n\times c}$. Using properties of the vec operation [11], we have $\mathrm{vec}(\boldsymbol{\Lambda})=\boldsymbol{\lambda}$ and

$$\begin{aligned}&\left\{\left[\mathbf{I}-\frac{1}{c}\mathbf{1}\mathbf{1}^T\right]^{+}\otimes(\mathbf{F}\tilde{\mathbf{D}}\mathbf{F}^T)^{-1}\right\}\mathrm{vec}\Big(\mathbf{F}\tilde{\mathbf{D}}(\tilde{\mathbf{Q}}-\mathbf{Q}_t)\Big)\\ &=\mathrm{vec}\left((\mathbf{F}\tilde{\mathbf{D}}\mathbf{F}^T)^{-1}(\mathbf{F}\tilde{\mathbf{D}}(\tilde{\mathbf{Q}}-\mathbf{Q}_t))\left[\mathbf{I}-\frac{1}{c}\mathbf{1}\mathbf{1}^T\right]^{+}\right).\end{aligned}$$

Thus, we can rewrite the iterative procedure in (19) in matrix form as

$$\boldsymbol{\Lambda}(t+1)=\boldsymbol{\Lambda}(t)+2(\mathbf{F}\tilde{\mathbf{D}}\mathbf{F}^T)^{-1}\mathbf{F}\tilde{\mathbf{D}}(\tilde{\mathbf{Q}}-\mathbf{Q}_t)\left[\mathbf{I}-\frac{1}{c}\mathbf{1}\mathbf{1}^T\right]^{+}. \tag{21}$$

Note that if (19) is used, then, because $\mathbf{B}$ is $cm \times cm$, inversion of a $cm \times cm$ matrix is required at all iterations. On the other hand, if (21) is used, then only the inverses of an $m \times m$ matrix and a $c \times c$ matrix are required. Clearly, the latter is more efficient, especially when cm is large. Moreover, note that for Newton's method defined in (18), an efficient iterative equation like (21) cannot be derived.

5 Discussions and Experimental Results

We can see that the surrogate function for an objective function is not unique. As in Section 4, by using the three different approaches outlined in Section 2, different SM algorithms corresponding to different surrogate functions can be devised. While each of these approaches can be used separately, using them together is sometimes also useful. On the other hand, we also see that a standard SM algorithm for one surrogate function can at the same time be a generalized SM algorithm for another surrogate function. We are thus interested in criteria that guide us in the design of good surrogate functions. Intuitively, one criterion that could be used is the closeness of a surrogate function to the original objective function. For example, the closer is the surrogate function to the objective function, the better it will be. Another possible criterion is the tractability of the M-step. For example, a closed-form update equation will be most desirable. In other words, we want the surrogate function to be both *efficient* and *effective*. In practice, however, there has to be a tradeoff between these two criteria.

Now we demonstrate this tradeoff of the SM algorithms on a text categorization task using the WebKB data set [6]. This data set contains web pages gathered from computer science departments in several universities. The pages can be divided into seven categories. In the experiments, we only use the four most populous entity-representing categories, namely, student, faculty, course, and project, resulting in a total of 4199 pages. Based on the information gain, 300 features are selected. To satisfy the condition in (6), we define a feature as

$$f_j(\mathbf{x}_k) = \frac{N_j(\mathbf{x}_k)}{N(\mathbf{x}_k)},$$

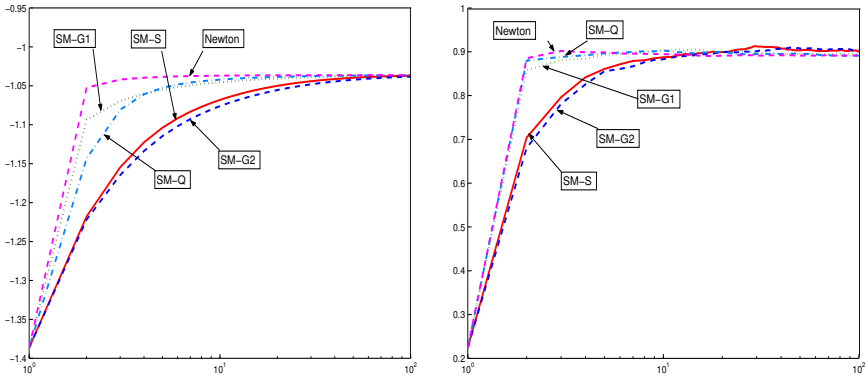
where $N_j(\mathbf{x}_k)$ is the number of occurrences of feature j in document $\mathbf{x}_k$ and $N(\mathbf{x}_k)$ is the total number of occurrences of all features in document $\mathbf{x}_k$. In the experiments, 70% of the data are randomly sampled for training while the remaining 30% are used for testing. For each training example $\mathbf{x}_k$, $\tilde{p}(i|\mathbf{x}_k)$ is set to 0.7 if $y_k = i$, and to 0.1 otherwise. The initial values of λ_{ij}'s are set to zero. The following four SM algorithms

SM-S: standard SM algorithm in (12),
SM-G1: gradient SM algorithm in (15),
SM-G2: gradient SM algorithm in (16),
SM-Q: quadratic SM algorithm in (19), and

Newton's method in (18) are compared.

All implementations are in MATLAB, and the experiments are run on a 8 x Sun Microsystems Ultra-SPARC III 900Mhz CPU each with 8MB E-Cache and 8GB RAM. Figure 1 shows the results of our four algorithms and Newton's method. Note that the

x-axis is in log scale for both plots. From Figure 1(a), we can see that SM-G1, SM-Q and Newton's method take only around five iterations to converge to the maximum log-likelihood value, while SM-S and SM-G2 need tens of iterations. Results on the test set classification accuracy also show similar trends (Figure 1(b)). Table 1 gives the CPU time of these five methods. Recall that for SM-Q, we are required to invert a 300×300 matrix and a 4×4 matrix in all 100 iterations. On our workstation, it takes almost zero time to invert the 4×4 matrix but it takes 6.79 seconds to invert the 300×300 matrix. For Newton's method, it needs to invert a 1200×1200 matrix at each iteration. So its computational cost is very huge. Moreover, it also needs a lot of memory for this 1200×1200 matrix. So Newton's method is ineffective for this problem. As we see, both SM-G1 and SM-G2 reduce the computational and storage demands of Newton's method.



(a) (b)

Fig. 1. (a) Log-likelihood loss vs. number of iterations; (b) Testing accuracy vs. number of iterations.

Table 1. CPU time (in seconds) required for running 100 iterations.

SM-S	SM-G1	SM-G2	SM-Q	Newton's
0.03×100	27.31×100	26.61×100	$6.79 + 0.06 \times 100$	976.46×100

6 Concluding Remarks

In this paper, we use multi-class logistic regression models as a specific example to illustrate some general principles for devising SM algorithms, which include construction of the surrogate function and its iterative maximization. Recall that SM-G1 (or SM-G2) is essentially Newton's method that works on the surrogate function $Q_T(\Lambda|\Lambda(t))$

(or $Q_J(\Lambda|\Lambda(t))$) instead of $L(\Lambda)$. For problems with a large amount of data and/or data with high dimensionality (such as gene expression arrays, which typically have 50 to 100 samples and $5,000$ to $20,000$ genes), Newton's method on $L(\Lambda)$ becomes intractable because the inverse of a high-dimensional Hessian matrix is required during its each iteration. However, to a certain extent, both SM-G1 and SM-G2 can be used to deal with this problem.

References

1. A. Berger. The improved iterative scaling algorithm: a gentle introduction. Technical report, 1997. Available from http://www.cs.cmu.edu/∼aberger/www/ps/scaling.ps.
2. A. Berger, V. Della Pietra, and S. Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22:39–71, 1996.
3. D. Böhning. Multinomial logistic regression algorithm. *Annals of the Institute of Statistical Mathematics*, 44(1):197–200, 1992.
4. D. Böhning and B. G. Lindsay. Monotonicity of quadratic-approximation algorithms. *Annals of the Institute of Statistical Mathematics*, 40(4):641–663, 1988.
5. M. Collins, R. E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 47(2/3):253–285, 2002.
6. M. Craven, D. Dopasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Web Wide. In *The Fifteenrh National Conference on Artificial Intelligence*, 1998.
7. J. N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43(5):1470–1480, 1972.
8. S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, 1997.
9. A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society Series B*, 39(1):1–38, 1977.
10. J. H. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: A statistical view of boosting. *Annals of Statistics*, 28(2):337–374, 2000.
11. R. A. Horn and C. R. Johnson. *Topics in Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1991.
12. R. Jin, R. Yan, J. Zhang, and A. G. Hauptmann. A faster iterative scaling algorithm for conditional exponential model. In *The 20th International Conference on Machine Learning*, 2003.
13. J. Lafferty. Additive models, boosting and inference for generalized divergences. In *The Twelfth Annual Conference on Computational Learning Theory*, pages 125–133, 1999.
14. K. Lange. A gradient algorithm locally equivalent to the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 57(2):425–437, 1995.
15. K. Lange, D. R. Hunter, and I. Yang. Optimization transfer using surrogate objective functions (with discussion). *Journal of Computational and Graphical Statistics*, 9(1):1–59, 2000.
16. G. Lebanon and J. Lafferty. Boosting and maximum likelihood for exponential models. In *Advances in Neural Information Processing Systems 14*, 2001.
17. X.-L. Meng. Discussion on "Optimization transfer using surrogate objective functions". *Journal of Computational and Graphical Statistics*, 9(1):35–43, 2000.
18. Z. Zhang, J. K. Kwok, and D. Y. Yeung. Surrogate maximization/minimization algorithms for adaboost and the logistic regression model. In *The 21th International Conference on Machine Learning*, 2004.