Comput Manag Sci (2011) 8:415–428
DOI 10.1007/s10287-010-0128-1

ORIGINAL PAPER

Kernel logistic regression using truncated Newton method

Maher Maalouf · Theodore B. Trafalis · Indra Adrianto

Received: 2 December 2009 / Accepted: 8 December 2010 / Published online: 25 December 2010

Abstract Kernel logistic regression (KLR) is a powerful nonlinear classifier. The combination of KLR and the truncated-regularized iteratively re-weighted least-squares (TR-IRLS) algorithm, has led to a powerful classification method using small-to-medium size data sets. This method (algorithm), is called truncated-regularized kernel logistic regression (TR-KLR). Compared to support vector machines (SVM) and TR-IRLS on twelve benchmark publicly available data sets, the proposed TR-KLR algorithm is as accurate as, and much faster than, SVM and more accurate than TR-IRLS. The TR-KLR algorithm also has the advantage of providing direct prediction probabilities.

Keywords Classification · Logistic regression · Kernel methods · Truncated Newton method

1 Introduction

Logistic regression (LR) is an essential data mining technique for classifying binary data sets. Recently, there has been a revival of LR importance through the implementation of methods such as the truncated Newton. Truncated Newton methods have been effectively applied to solve large scale optimization problems. Komarek and Moore

M. Maalouf (✉) · T. B. Trafalis · I. Adrianto
School of Industrial Engineering, University of Oklahoma,
202 WestBoyd, Room 124, Norman, OK 73019, USA
e-mail: mcm@ou.edu

T. B. Trafalis
e-mail: ttrafalis@ou.edu

I. Adrianto
e-mail: iadrianto@ou.edu

(2005) were the first to show that the truncated-regularized iteratively re-weighted least squares (TR-IRLS) can be effectively implemented on LR to classify large data sets, and that it can outperform the support vector machines (SVM) algorithm. Later on, trust region Newton method (Lin et al. 2007), which is a type of truncated Newton, and truncated Newton interior-point methods (Koh et al. 2007) were applied for large scale LR problems. SVM (Vapnik 1995) is considered a state-of-the-art algorithm for classifying binary data through its implementation of kernels. Kernel logistic regression (KLR) (Canu and Smola 2006; Jaakkola and Haussler 1999), which is a kernel version of LR has also proven to be a powerful classifier (Zhu and Hastie 2005). Like LR, KLR can naturally provide probabilities and extend to multi-class classification problems (Hastie et al. 2001; Karsmakers et al. 2007).

Each one of aforementioned methods has a limitation. LR linearity may be an obstacle to handling highly nonlinearly separable small-to-medium size data sets (Komarek and Moore 2005). SVM does not naturally extend to multi-class classification and does not provide probability estimates (Zhu and Hastie 2005). The SVM method also requires solving a constrained quadratic optimization problem with a time complexity of $O(N^3)$ where N is the number of training instances. The KLR method is not sparse and requires all of the training instances in its model. Like SVM, KLR has a time complexity of $O(N^3)$. Its computation can be slow due to the density of its matrices (Karsmakers et al. 2007). Roth (2001) was the first to apply the conjugate gradient (CG) algorithm on KLR and presented its efficiency on multi-class data sets. Zhu and Hastie (2005) suggested the import vector machine (IVM) algorithm in order to take advantage of the SVM sparsity, thus reducing the time complexity to $O(N^2q^2)$ for binary classification, and $O(MN^2q^2)$ for the multi-class classification, where q is the number of import points and M is the number of classes. Keerthi et al. (2005) incorporated the popular sequential minimal optimization (SMO) algorithm in KLR and showed the results for binary classification. Karsmakers et al. (2007) offered a fixed-size approach based on the number of support vectors to solve a multi-class KLR problems with the method of alternating descent. However, accuracy was compromised for faster computations.

Our motivation for this study stems from the success and effectiveness of truncated Newton methods for solving large scale LR classification problems. In this study we combine the speed of the TR-IRLS algorithm with the accuracy generated by the use of kernels for solving nonlinear problems. Our kernel version of the TR-IRLS algorithm (TR-KLR) is just as easy to implement and requires solving only an unconstrained regularized optimization problem, thus providing a computationally more efficient alternative algorithm to SVM. The combination of regularization, approximate numerical methods, kernelization and efficient implementation are critical to enabling TR-KLR to be an effective and powerful method for classification. TR-KLR can also be extended to handle multi-class classification problems. To make our evaluation more thorough, we test the performance of TR-KLR on twelve benchmark data sets, six of which are binary-class data sets and six are multi-class data sets. In addition, for the multi-class data sets, the performance of TR-KLR is tested using one-versus-all (OVA) (Vapnik 1995; Rifkin and Klautau 2004), one-versus-one (OVO) Kressel (1999), and decision-directed-acyclic graph (DDAG) (Platt et al. 2000) coding methods. In as much as the use of truncated Newton methods has not been fully

exploited in solving LR models (Lin et al. 2007), our intent is to provide further contribution.

In sect. 2, we provide a brief description of the LR method. In Sect. 3, we derive the KLR model. Section 4 discusses fitting KLR with IRLS. Section 5 describes how the TR-IRLS algorithm is used to solve KLR models. Numerical results are presented in Sect. 6 and Sect. 7 states the conclusion.

2 Logistic regression

Let $\mathbf{X} \in \mathbb{R}^{N \times d}$ be a data matrix where N is the number of instances (examples) and d is the number of features (parameters or attributes), and $\mathbf{y}$ be a binary outcomes vector. For every instance $\mathbf{x}_i \in \mathbb{R}^d$ (a row vector in $\mathbf{X}$), where $i = 1 \ldots N$, the outcome is either $y_i = 1$ or $y_i = 0$. Let the instances with outcomes of $y_i = 1$ belong to the positive class, and the instances with outcomes $y_i = 0$ belong to the negative class. The goal is to classify the instance $\mathbf{x}_i$ as positive or negative. An instance can be thought of as a Bernoulli trial with an expected value $E(y_i)$ or probability p_i. The logistic function commonly used to model each instance $\mathbf{x}_i$ with its expected outcome is given by the following formula (Hosmer and Lemeshow 2000):

$$E[y_i|\mathbf{x}_i, \boldsymbol{\beta}] = p_i = \frac{e^{\mathbf{x}_i \boldsymbol{\beta}}}{1 + e^{\mathbf{x}_i \boldsymbol{\beta}}}, \tag{1}$$

where $\boldsymbol{\beta}$ is the vector of parameters with the assumption that $x_{i0} = 1$ so that the intercept $\boldsymbol{\beta_0}$ is a constant term. From now on, the assumption is that the intercept is included in the vector $\boldsymbol{\beta}$.

The logistic (logit) transformation is the logarithm of the odds of the positive response, and is defined as

$$\eta_i = \ln\left(\frac{p_i}{1 - p_i}\right) = \mathbf{x}_i \boldsymbol{\beta}. \tag{2}$$

In matrix form, the logit function is expressed as

$$\boldsymbol{\eta} = \mathbf{X}\boldsymbol{\beta}. \tag{3}$$

The regularized log likelihood is defined as

$$\ln \mathbb{L}(\boldsymbol{\beta}) = \sum_{i=1}^{N} (y_i \ln p_i + (1 - y_i) \ln(1 - p_i)) - \frac{\lambda}{2}||\boldsymbol{\beta}||^2, \tag{4}$$

where the regularization (penalty) term $\frac{\lambda}{2}||\boldsymbol{\beta}||^2$ is added to obtain better generalization. For binary outputs, the loss function or the deviance $\mathbb{DEV}$ is the negative log likelihood and is given by the formula (Komarek and Moore 2005; Hosmer and Lemeshow 2000)

$$\mathbb{DEV}(\hat{\boldsymbol{\beta}}) = -2\ln \mathbb{L}(\boldsymbol{\beta}). \tag{5}$$

Minimizing the deviance $\mathbb{DEV}(\hat{\boldsymbol{\beta}})$ given in (5) is equivalent to maximizing the log-likelihood given in (2) (Hosmer and Lemeshow 2000). The deviance function above is nonlinear in β and minimizing it requires numerical methods in order to find the maximum likelihood estimate (MLE) of $\boldsymbol{\beta}$, which is $\hat{\boldsymbol{\beta}}$. Recent studies have shown that the CG method provides better results to estimate $\boldsymbol{\beta}$ than any other numerical method (Minka 2003; Malouf 2002).

3 Kernel logistic regression

Following Maalouf and Trafalis (2011), the vector $\boldsymbol{\beta}$ can be expressed as a linear combination of the input vectors, such that

$$\boldsymbol{\beta} = \mathbf{X}^{\mathrm{T}}\boldsymbol{\alpha} = \sum_{i=1}^{N} \alpha_i \mathbf{x}_i, \tag{6}$$

where the vector $\boldsymbol{\alpha}$ is known as the *dual variable* with dimensions $N \times 1$. Now, the logit vector η can be rewritten as

$$\eta = \mathbf{X}\mathbf{X}^{\mathrm{T}}\boldsymbol{\alpha} \tag{7}$$

$$= \mathbf{K}\boldsymbol{\alpha}, \tag{8}$$

where $\mathbf{K} = \mathbf{X}\mathbf{X}^{\mathrm{T}}$ is symmetric positive semidefinite Gram matrix with $N \times N$ dimensions.

Consider again the logit link function shown in the previous section,

$$\eta_i = \mathbf{x}_i\boldsymbol{\beta} = \beta_0 + x_{i1}\beta_1 \ldots x_{id}\beta_d, \tag{9}$$

where the vector $\mathbf{x}_i$, given by $\mathbf{x}_i = [1, x_{i1}, \ldots x_{id}]$ with $i = 1, \ldots N$. This linear function represents the simplest form of an identity mapping polynomial basis function ϕ of the feature space such that

$$\phi(\mathbf{x}_i) = \phi[(1, x_{i1}, \ldots x_{id})] = \mathbf{x}_i. \tag{10}$$

Thus, the logit link function could be rewritten as

$$\eta_i = \phi(\mathbf{x}_i)\boldsymbol{\beta}. \tag{11}$$

In general, the function $\phi(.)$ maps the data from a lower dimensional space into a higher one (Fig. 1), such that

$$\phi : \mathbf{x} \in \mathbb{R}^d \rightarrow \phi(\mathbf{x}) \in \mathbb{F} \subseteq \mathbb{R}^{\Lambda}. \tag{12}$$

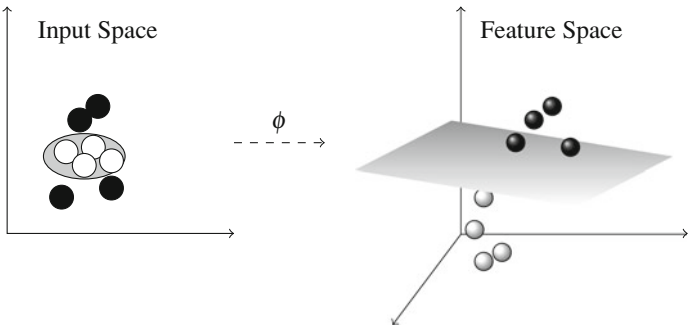

Fig. 1 Mapping of nonlinearly separable data from the input space to the feature space

The goal for choosing the mapping ϕ is to convert nonlinear relations between the response variable and the independent variables into linear relations. However, the transformations $\phi(.)$ are often unknown but the dot product in the feature space can be expressed in terms of the input vectors through the kernel function. In the case of KLR, the logit link function could then be expressed as

$$\eta_i = \sum_{j=1}^{N} \alpha_j \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle \tag{13}$$

$$= \sum_{j=1}^{N} \alpha_j \kappa(\mathbf{x}_i, \mathbf{x}_j) \tag{14}$$

$$= \mathbf{k}_i \boldsymbol{\alpha}, \tag{15}$$

where $\mathbf{k}_i$ is the i-th row in the kernel matrix $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{K}$. The kernel is a transformation function that must satisfy Mercer's necessary and sufficient conditions, which state that a kernel function must be expressed as an inner product and must be positive semidefinite (Shawe-Taylor and Cristianini 2004; Cristianini and Shawe-Taylor 2000).

Now,

$$\eta_i = \mathbf{k}_i \alpha = \ln\left(\frac{p_i}{1 - p_i}\right), \tag{16}$$

which implies that

$$p_i = \frac{e^{\eta_i}}{1 + e^{\eta_i}} = \frac{e^{\mathbf{k}_i \alpha}}{1 + e^{\mathbf{k}_i \alpha}} = \frac{1}{1 + e^{-\mathbf{k}_i \alpha}}, \tag{17}$$

and hence, the regularized log-likelihood can be rewritten with respect to α as

$$\ln \mathbb{L}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} (y_i \ln p_i + (1 - y_i) \ln(1 - p_i)) - \frac{\lambda}{2} \boldsymbol{\alpha}^{\mathrm{T}} \mathbf{K} \boldsymbol{\alpha}, \tag{18}$$

with a deviance

$$\mathbb{DEV}(\boldsymbol{\alpha}) = -2 \ln \mathbb{L}(\boldsymbol{\alpha}). \tag{19}$$

4 Iteratively re-weighted least squares

One of the most popular techniques used to find the MLE of $\boldsymbol{\beta}$ in LR models is the iteratively re-weighted least squares (IRLS) method, which uses Newton–Raphson algorithm to solve the score equations. KLR models can also be fitted using IRLS (Karsmakers et al. 2007). Each iteration finds the *weighted least squares* (WLS) estimates for a given set of weights, which are used to construct a new set of weights (Komarek and Moore 2005; Garthwaite et al. 2002). The gradient and Hessian are obtained by differentiating $\ln \mathbb{L}(\boldsymbol{\alpha})$ with respect to $\boldsymbol{\alpha}$. In matrix form, the gradient is

$$\nabla_{\boldsymbol{\alpha}} \ln \mathbb{L}(\boldsymbol{\alpha}) = \mathbf{K}^{\mathrm{T}}(\mathbf{y} - \mathbf{p}) - \lambda \mathbf{K}\boldsymbol{\alpha} = \mathbf{0}, \tag{20}$$

where $\mathbf{p}$ is the probability vector whose elements are given in (17). The Hessian with respect to $\boldsymbol{\alpha}$ is

$$\nabla^2_{\boldsymbol{\alpha}} \ln \mathbb{L}(\boldsymbol{\alpha}) = -\mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{K} - \lambda \mathbf{K}, \tag{21}$$

where $\mathbf{V}$ is a diagonal matrix with diagonal elements $p_i(1 - p_i)$ for $i = 1 \ldots N$. The Newton–Raphson update with respect to $\boldsymbol{\alpha}$ on the $(c + 1) - th$ iteration is

$$\hat{\boldsymbol{\alpha}}^{(c+1)} = \hat{\boldsymbol{\alpha}}^{(c)} + (\mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{K} + \lambda \mathbf{K})^{-1}(\mathbf{K}^{\mathrm{T}}(\mathbf{y} - \mathbf{p}) - \lambda \mathbf{K}\boldsymbol{\alpha}^{(c)}). \tag{22}$$

Since $\hat{\boldsymbol{\alpha}}^{(c)} = (\mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{K} + \lambda \mathbf{K})^{-1}(\mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{K} + \lambda \mathbf{K})\hat{\boldsymbol{\alpha}}^{(c)}$, then equation (11) can be rewritten as

$$\hat{\boldsymbol{\alpha}}^{(c+1)} = (\mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{K} + \lambda \mathbf{K})^{-1}\mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{z}^{(c)}, \tag{23}$$

where $\mathbf{z}^{(c)} = \mathbf{K}\hat{\boldsymbol{\alpha}}^{(c)} + \mathbf{V}^{-1}(\mathbf{y} - \mathbf{p})$ is the adjusted dependent variable or the adjusted response.

5 TR-KLR algorithm

The KLR WLS subproblem,

$$(\mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{K} + \lambda \mathbf{K})\hat{\boldsymbol{\alpha}}^{(c+1)} = \mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{z}^{(c)}, \tag{24}$$

is a systems of linear equations with a kernel matrix $\mathbf{K}$, vector of adjusted responses $\mathbf{z}$, and a weight matrix $\mathbf{V}$. Both the weights and the adjusted response vector are dependent on $\hat{\boldsymbol{\alpha}}^{(c)}$, which is the current estimate of the parameter vector. Therefore, specifying an initial estimate $\hat{\boldsymbol{\alpha}}^{(0)}$ for $\hat{\boldsymbol{\alpha}}$ can be solved iteratively, giving a sequence of estimates that converges to the MLE of $\hat{\boldsymbol{\alpha}}$. That can be solved using the linear CG

method, which is equivalent to minimizing the quadratic problem,

$$\frac{1}{2}\hat{\boldsymbol{\alpha}}^{(c+1)}(\mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{K}+\lambda\mathbf{K})\hat{\boldsymbol{\alpha}}^{(c+1)}-\hat{\boldsymbol{\alpha}}^{(c+1)}(\mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{z}^{(c)}). \tag{25}$$

When applied to KLR, the CG method has a time complexity of $O(N^3)$ in the worst case, as it converges in at most N steps. To avoid the long computations that the CG may suffer from, a limit to the number of CG iterations can be placed, thus creating an approximate, or truncated Newton direction.

Algorithm 1 represents the main (outer) loop of TR-KLR and it summarizes the IRLS for KLR and is terminated when the relative difference of deviance between two consecutive iterations is no greater than a specified threshold ε_1. As with TR-IRLS, the main problem to solve is the WLS in (24), which is a linear system of N equations and N variables. This is done through Algorithm 2, which represents the inner loop that

Algorithm 1: KLR MLE using IRLS

Data: $\mathbf{K}, \mathbf{y}, \hat{\alpha}^{(0)}$
Result: $\hat{\alpha}$
begin
  $c = 0$
  while $|\frac{DEV^{(c)}-DEV^{(c+1)}}{DEV^{(c+1)}}| > \varepsilon_1$ **and** $c \leq$ *Max IRLS Iterations* **do**
    for $i \leftarrow 1$ **to** N **do**
      $\hat{p}_i = \frac{1}{1+e^{-\mathbf{x}_i\hat{\alpha}}}$; /* Compute probabilities */
      $v_i = \hat{p}_i(1-\hat{p}_i)$; /* Compute weights */
      $z_i = \mathbf{k}_i\hat{\alpha}^{(c)} + \frac{(y_i-\hat{p}_i)}{\hat{p}_i(1-\hat{p}_i)}$; /* Compute adjusted response */
    $\mathbf{V} = diag(v_1, ..., v_N)$
    $(\mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{K}+\lambda\mathbf{K})\hat{\alpha}^{(c+1)} = \mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{z}^{(c)}$; /* Compute $\hat{\alpha}$ via WLS */
    $c = c+1$
end

Algorithm 2: Linear CG. $\mathbf{A} = \mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{K}+\lambda\mathbf{K}$, $\mathbf{b} = \mathbf{K}^{\mathrm{T}}\mathbf{V}\mathbf{z}$

Data: $\mathbf{A}, \mathbf{b}, \hat{\alpha}^{(0)}$
Result: $\hat{\alpha}$ such that $\mathbf{A}\hat{\alpha} = \mathbf{b}$
begin
  $\mathbf{r}^{(0)} = \mathbf{b} - \mathbf{A}\hat{\alpha}^{(0)}$; /* Initialize the residual */
  $c = 0$
  while $||\mathbf{r}^{(c+1)}||^2 > \varepsilon_2$ **and** $c \leq$ *Max CG Iterations* **do**
    if $c = 0$ **then**
      $\zeta^{(c)} = 0$
    else
      $\zeta^{(c)} = \frac{\mathbf{r}^{\mathrm{T}(c+1)}\mathbf{r}^{(c+1)}}{\mathbf{r}^{\mathrm{T}(c+1)}\mathbf{r}^{(c)}}$; /* Update **A**-Conjugacy enforcer */
    $\mathbf{d}^{(c+1)} = \mathbf{r}^{(c+1)} + \zeta^{(c)}\mathbf{d}^{(c)}$; /* Update the search direction */
    $s^{(c)} = \frac{\mathbf{r}^{\mathrm{T}(c)}\mathbf{r}^{(c)}}{\mathbf{d}^{\mathrm{T}(c)}\mathbf{A}\mathbf{d}^{(c)}}$; /* Compute the optimal step length */
    $\hat{\alpha}^{(c+1)} = \hat{\alpha}^{(c)} + \zeta^{(c)}\mathbf{d}^{(c+1)}$; /* Obtain approximate solution */
    $\mathbf{r}^{(c+1)} = \mathbf{r}^{(c)} - s^{(c)}\mathbf{A}\mathbf{d}^{(c+1)}$; /* Update the residual */
    $c = c+1$
end

approximates the Newton direction through the CG method. Algorithm 2 is the linear CG algorithm and is terminated when the CG residual is no greater than a threshold ε_2.

With exception to the value of ε_1, the default parameter values suggested by Komarek and Moore (2005) are used for both algorithms and are shown to provide adequate accuracy. For Algorithm 1, the maximum number of iterations is set to 30 and for the relative difference of deviance threshold, ε_1, the value 2.5 is sufficient to reach the desired accuracy and at the same time maintain good convergence speed. By choosing the value 2.5 as a threshold, computational speed is improved while not affecting accuracy. Should the algorithm reach a certain desired accuracy with a low threshold, then by slightly modifying the parameters values (e.g. σ, λ) with a larger threshold, the same accuracy can be reached, hence the robustness of the algorithm. However, in some cases it may be advisable to make this threshold smaller to obtain better accuracy. As for Algorithm 2, the maximum number of iterations for the CG is set to 200 iterations. In addition, the CG convergence threshold, ε_2, is set to 0.005 and no more than three non-improving iterations are allowed on the CG algorithm.

6 Computational results and discussion

The performance of the TR-KLR algorithm was examined using twelve benchmark datasets (see Table 1) found on the UC Irvine website (Asuncion and Newman 2007), six of which are binary classification datasets, and the other six are multi-class classification datasets. The algorithm performance was then compared to that of both SVM and TR-IRLS. The binary datasets used are Wisconsin Breast Cancer Diagnostic (WBCD), Ionosphere, Bupa Liver Disorders, Haberman Survival, Pima Indians Diabetes, and Sonar datasets. The multi-class datasets consist of Wine, Glass, Iris, Dermatology, Thyroid, and Ecoli datasets. In addition, the multi-class classification performance was assessed using three methods; one versus all, one versus. one, and DDAG. The Gaussian radial basis function (RBF) kernel

Table 1 Data sets

Data set	Instances	Features	Classes
WBCD	569	30	2
Ionosphere	351	34	2
Liver	345	6	2
Survival	306	3	2
Sonar	208	60	2
Diabetes	768	8	2
Wine	178	13	3
Glass	214	10	6
Iris	150	4	3
Dermatology	358	34	6
Thyroid	215	5	3
Ecoli	336	7	8

Table 2 Optimal parameter values found for the binary-class data sets

	TR-KLR		SVM		TR-IRLS
	σ	λ	σ	C	λ
WBCD	5.4	0.1	5.0	10.0	30.0
Ionosphere	3.5	0.009	1.9	100.0	30.0
Liver	7.0	0.0009	5.0	10.0	0.05
Survival	5.0	0.01	1.8	1.0	10.0
Sonar	3.2	0.05	7.1	10.0	50.0
Diabetes	5.0	0.07	7.5	1.0	1.0

Table 3 Comparison of tenfold CV accuracy (%) with 95% confidence level

	TR-KLR	SVM	TR-IRLS
WBCD	98.1 ± 1.1	98.2 ± 1.1	98.1 ± 1.1
Ionosphere	93.7 ± 2.5	90.6 ± 3.1	88.0 ± 3.4
Liver	70.1 ± 4.8	70.1 ± 4.8	65.3 ± 5.0
Survival	75.4 ± 4.8	75.1 ± 4.9	73.8 ± 4.9
Sonar	89.0 ± 4.3	87.9 ± 4.5	79.2 ± 5.5
Diabetes	78.0 ± 2.9	77.2 ± 3.0	78.0 ± 2.9

$$K(\mathbf{x}_i, \mathbf{x}_j) = e^{(-\frac{1}{2\sigma^2}||\mathbf{x}_i - \mathbf{x}_j||)^2} = e^{(-\gamma||\mathbf{x}_i - \mathbf{x}_j||)^2} \tag{26}$$

was used for both the TR-KLR and SVM methods, where σ is the kernel parameter. The values of these parameters that give the best generalization are usually chosen from a range of different values (generally user-defined), and tuned using tenfold cross-validation (CV). The datasets were preprocessed using normalization of a mean of zero and standard deviation of one. All of the computations for TR-IRLS and TR-KLR were carried out using MATLAB version 2007a on a 512 MiB RAM computer for the binary datasets and on a 1.5 GiB RAM computer for the multi-class datasets. As for the SVM method, MATLAB SVM Toolbox (Gunn 1998) was used for the binary datasets and MATLAB LIBSVM toolbox (Chang and Lin 2001) for the multi-class datasets.

6.1 Binary classification

For the binary datasets, Tables 2 and 3 summarize the computation results for these three methods with their optimal parameters (the parameters that provided the best generalization) and accuracy, respectively. Table 4 shows a comparison of the total execution time with ten-fold CV using the three methods. Table 3 shows that the TR-KLR method scored as well as or slightly better than SVM on most of the datasets. In addition, both TR-KLR and SVM performed better than TR-IRLS on four out of the six datasets, namely, Ionosphere, Liver, and Sonar. As with all kernel methods, parameter tuning is unavoidable as it involves two parameters, the regularization parameter (λ for TR-KLR and C for SVM), and the kernel parameter (σ or γ for both

Table 4 Comparison of tenfold CV time (in seconds)

	TR-KLR	SVM	TR-IRLS
WBCD	10.6	3305	5.8
Ionosphere	3.6	248	2.3
Liver	3.0	209	2.2
Survival	2.1	158	1.5
Sonar	1.3	58	1.5
Diabetes	17.0	7374	4.8

Table 5 Maximum number of iterations reached by IRLS and CG during the tenfold CV on the binary-class data sets

	IRLS	CG
WBCD	2	31
Ionosphere	2	32
Liver	1	19
Survival	1	14
Sonar	2	46
Diabetes	1	25

TR-KLR and SVM). For TR-IRLS, the only parameter that requires tuning is λ. While Komarek and Moore (2005) observed that the value of λ did not affect the accuracy for large datasets, and hence they were able to use default values, the same cannot be said for smaller datasets. TR-IRLS challenged both TR-KLR and SVM on three datasets, WBCD, Survival, and Diabetes. This is probably because these datasets are more linearly separable than the others, on which TR-IRLS performed worse.

With regard to the execution time, Table 4 shows that while TR-IRLS is the fastest, TR-KLR is still significantly faster than SVM and yet similar to it with regard to accuracy while at the same time more accurate than the regular TR-IRLS, as shown in Table 3.

As mentioned earlier, the maximum number of iterations for IRLS was set to 30 while that of the CG method was set to 200. However, as Komarek and Moore (2005) correctly stated, these numbers should never be reached. This observation applies also to the TR-KLR algorithm, as shown in the empirical results in Table 5, where the maximum number of iterations reached by both algorithms during the ten-fold cross validation is displayed for the binary-class datasets. The maximum iterations reached on the binary-class datasets by IRLS in Algorithm 1 was 2 while the maximum CG iterations in Algorithm 2 was 46. Therefore, the number of iterations is relatively small compared to the size of the datasets.

6.2 Multi-class classification

As mentioned earlier, for multi-class classification, the performance of TR-KLR was evaluated against both SVM and TR-IRLS using three methods: OVA, OVO, and DDAG. While LR and KLR can naturally extend to multi-class classification

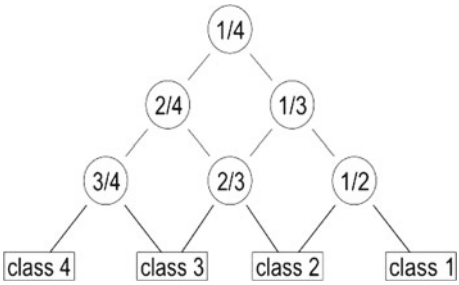

Fig. 2 Illustration of DDAG for classification with four classes

(Karsmakers et al. 2007), the aforementioned multi-class coding schemes were applied in this study to make the comparison with SVM fair. The OVA approach (Vapnik 1995; Rifkin and Klautau 2004) constructs M classifiers for M classes. Classifier f_m is trained to discriminate between class m and all other classes. Then, the class of instance $\mathbf{x}_i$ corresponds to the maximal value of the function $f_m(\mathbf{x}_i)$ such that $C_m = \max f_m(\mathbf{x}_i)$ for $m = 1, \ldots, M$, where C_m is the class of $\mathbf{x}_i$. The OVO approach Kressel (1999), constructs $\frac{M(M-1)}{2}$ binary discriminant functions, one for every pair of classes and proceeds as the OVA approach. As for the DDAG approach (Platt et al. 2000), it also constructs $\frac{M(M-1)}{2}$ classifiers (nodes) in the training phase. In the testing phase, it utilizes decision-directed-acyclic graph (DDAG), whereby at each node, a binary classifier is constructed, and the next node visited depends upon the results of this evaluation. If the value of the binary decision function is zero, the node exits from the left, otherwise, if the value is one, then the node exits from the right. The final answer is the class assigned by the leaf node visited at the final step as illustrated by Fig. 2. The root node can be assigned randomly. The advantages of DDAG are fast computational time especially for large-scale problems.

Table 6 lists the optimal parameters used to reach the desired accuracies, which are shown in Table 7. Table 6 shows that the parameters used to reach the desired accuracies are similar for all algorithms using both OVA and DDAG methods. In addition, it appears that the kernel parameter values (σ or γ) and the regularization parameter (λ) are more stable using OVO and DDAG as they do not vary much from one data set to another.

Table 7 shows the ten-fold CV accuracy with 95% confidence level reached by all algorithms using the three multi-class classification methods. With regard to OVA, TR-KLR performed better than both SVM and TR-IRLS on all datasets. As for the OVO and DDAG methods, accuracies were similar on both methods. TR-KLR generally performed best using OVO and DDAG except for the Ecoli data, on which OVA performed slightly better. The performance of SVM appears consistent using all methods. On the other hand, TR-IRLS accuracies varied depending on the multi-class classification method used. TR-IRLS performed poorer using OVA than using OVO and DDAG, especially on Iris and Thyroid datasets. On the Glass data set, TR-IRLS performs poorest with a difference of almost 10 percentage point in accuracy compared to TR-KLR, and 7 percentage point compared to SVM.

Table 6 Optimal parameter values for the multi-class data sets

	TR-KLR		SVM	TR-IRLS	
	σ	λ	γ	C	λ
OVA					
Wine	6.0	0.005	0.1	1.0	0.5
Glass	1.0	0.0005	0.1	100.0	0.09
Iris	3.0	0.001	0.02	10.0	0.01
Dermatology	5.0	0.1	0.01	1.0	0.3
Thyroid	1.6	0.004	7.1	0.09	0.005
Ecoli	6.0	0.01	7.5	0.01	0.5
OVO					
Wine	4.0	0.01	0.01	1.0	0.5
Glass	1.0	0.01	0.1	1.0	0.5
Iris	5.0	0.01	0.03	10.0	0.004
Dermatology	5.0	0.02	0.01	1.0	0.3
Thyroid	1.2	0.004	0.1	10.0	0.005
Ecoli	5.0	0.05	0.3	1.0	0.5
DDAG					
Wine	4.0	0.01	0.01	1.0	0.5
Glass	1.0	0.01	0.1	1.0	0.5
Iris	5.0	0.01	0.03	10.0	0.004
Dermatology	5.0	0.02	0.01	1.0	0.3
Thyroid	1.2	0.004	0.1	10.0	0.005
Ecoli	5.0	0.05	0.3	1.0	0.5

C is the SVM regularization parameter, σ is the parameter (width) of the RBF kernel, and λ is the regularization parameter for both TR-IRLS and TR-KLR

Similar to the binary-class case, the maximum number of iterations reached by both algorithms during the ten-fold cross validation is displayed in Table 8 for the multi class datasets. As can be observed, the maximum iterations reached by IRLS in Algorithm 1 is 2 while the maximum CG iterations in Algorithm 2 was 59, indicating that the number of iterations is also relatively small compared to the size of the datasets.

7 Conclusion

We have thus presented the TR-KLR algorithm and have shown that it is relatively easy to implement and is as effective as SVM on small-to-medium size data sets. We have further demonstrated that the TR-KLR algorithm takes advantage of the speed of the TR-IRLS and the power of the kernel methods, particularly when the data sets are neither large nor linearly separable. Another benefit to using TR-KLR is that it uses unconstrained optimization methods whose algorithms are less complex than those with constrained optimization methods, such as SVM. Future studies may compare these methods using different kernels and test the algorithm on more datasets. They may also try to improve the speed and sparsity of the algorithm. Methods such as the trust region Newton might also be utilized because of their stability and rigorous mathematical derivation.

Table 7 Comparison of tenfold CV accuracy (%) with 95 % confidence level

	TR-KLR	SVM	TR-IRLS
OVA			
Wine	99.47 ± 1.06	99.44 ± 1.09	98.36 ± 1.86
Glass	74.98 ± 5.80	72.09 ± 6.01	65.35 ± 6.42
Iris	98.00 ± 2.24	97.33 ± 2.58	96.00 ± 3.14
Dermatology	97.45 ± 1.63	97.45 ± 1.72	97.19 ± 1.71
Thyroid	97.64 ± 2.03	96.73 ± 2.38	95.32 ± 2.82
Ecoli	88.90 ± 3.36	88.08 ± 3.47	88.11 ± 3.46
OVO			
Wine	100.0 ± 0.00	99.44 ± 1.09	98.36 ± 1.86
Glass	75.53 ± 5.76	72.09 ± 6.01	65.05 ± 6.39
Iris	98.00 ± 2.24	98.00 ± 2.24	98.00 ± 2.24
Dermatology	98.00 ± 1.35	97.45 ± 1.63	97.73 ± 1.45
Thyroid	97.66 ± 2.02	97.64 ± 2.03	97.64 ± 2.03
Ecoli	88.46 ± 3.42	87.86 ± 3.49	88.02 ± 3.47
DDAG			
Wine	100.0 ± 0.00	99.44 ± 1.09	98.36 ± 1.86
Glass	75.53 ± 5.76	72.09 ± 6.01	65.05 ± 6.39
Iris	98.00 ± 2.24	98.00 ± 2.24	98.00 ± 2.24
Dermatology	98.00 ± 1.35	97.45 ± 1.63	97.73 ± 1.45
Thyroid	97.66 ± 2.02	97.64 ± 2.03	97.64 ± 2.03
Ecoli	88.46 ± 3.42	87.86 ± 3.49	88.02 ± 3.47

Table 8 Maximum number of iterations reached by IRLS and CG during the tenfold CV on the multi-class data sets

	OVA		OVO		DDAG	
	IRLS	CG	IRLS	CG	IRLS	CG
Wine	2	30	2	37	2	37
Glass	2	44	2	39	2	39
Iris	1	23	2	11	2	11
Dermatology	1	59	2	44	2	44
Thyroid	2	30	2	35	2	35
Ecoli	2	29	2	21	2	21

Acknowledgments The authors would like to thank Dr. S. Lakshmivarahan, Dr. Hillel Kumin, and Dr. Robin C. Gilbert, of the University of Oklahoma for their valuable input, comments and suggestions.

References

Asuncion A, Newman DJ (2007) UCI machine learning repository. University of california, irvine, School of information and computer sciences. http://www.ics.uci.edu/~mlearn/MLRepository.html

Canu S, Smola A (2006) Kernel methods and the exponential family. Neurocomputing 69(7–9):714–720

Chang CC, Lin CJ (2001) LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm

Cristianini N, Shawe-Taylor J (2000) An introduction to support vector machines and other kernel-based learning methods. Cambridge University Press, London

Garthwaite P, Jolliffe I, Jones B (2002) Statistical inference. Oxford University Press, London

Gunn SR (1998) MATLAB support vector machine toolbox. Software available at http://www.isis.ecs.soton.ac.uk/isystems/kernel/

Hastie T, Tibshirani R, Friedman J (2001) The elements of statistical learning. Springer, Berlin

Hosmer DW, Lemeshow S (2000) Applied logistic regression, 2nd edn. Wiley, London

Jaakkola TS, Haussler D (1999) Probabilistic kernel regression models. In: Proceedings of the 1999 conference on AI and statistics. Morgan Kaufmann, Cambridge

Karsmakers P, Pelckmans K, Suykens JAK (2007) Multi-class kernel logistic regression: a fixed-size implementation. In: IJCNN: Proceedings of the international joint conference on neural networks, IEEE, pp 1756–1761

Keerthi SS, Duan KB, Shevade SK, Poo AN (2005) A fast dual algorithm for kernel logistic regression. J Mach Learning 61(1–3):151–165. doi:10.1007/s10994-005-0768-5

Kressel UHG (1999) Pairwise classification and support vector machines. In: Advances in kernel methods: support vector learning. MIT Press, Cambridge, pp 255–268

Komarek P, Moore A (2005) Making LR a core data mining tool with TR-IRLS. In: ICDM: proceedings of the fifth IEEE international conference on data mining, IEEE Computer Society, Washington, USA, pp 685–688

Koh K, Kim S, Boyd S (2007) An interior-point method for large-scale ℓ_1-regularized logistic regression. J Mach Learn Res 8:1519–1555

Komarek P, Moore A (2005) Making logistic regression a core data mining tool: a practical investigation of accuracy, speed, and simplicity. Tech. Rep. CMU-RI-TR-05-27, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA

Lin CJ, Weng RC, Keerthi SS (2007) Trust region newton methods for large-scale logistic regression. In: ICML '07 proceedings of the 24th international conference on machine learning, ACM, New York, pp 561–568

Malouf R (2002) A comparison of algorithms for maximum entropy parameter estimation. In: COLING-02 proceeding of the 6th conference on natural language learning. Association for Computational Linguistics, Morristown, USA, pp 1–7. doi:10.3115/1118853.1118871

Maalouf M, Trafalis TB (2011) Robust weighted kernel logistic regression in imbalanced and rare events data. Comput Stat Data Anal 55(1):168–183

Minka TP (2003) A comparison of numerical optimizers for logistic regression. Tech rep, Deptartment of Statistics, Carnegie Mellon University

Platt JC, Cristianini N, Shawe-taylor J (2000) Large margin DAGs for multiclass classification. In: Advances in neural information processing systems, MIT Press, Cambridge, pp 547–553

Rifkin R, Klautau A (2004) In defense of one-vs-all classification. J Mach Learning Res 5:101–141

Roth V (2001) Probabilistic discriminative kernel classifiers for multi-class problems. In: Proceedings of the 23rd DAGM-symposium on pattern recognition. Springer, London, pp 246–253

Shawe-Taylor J, Cristianini N (2004) Kernel methods for pattern analysis. Cambridge University Press, London

Vapnik V (1995) The Nature of Statistical Learning. Springer, New York

Zhu J, Hastie T (2005) Kernel logistic regression and the import vector machine. J Comput Graphic Stat 14:185–205