

Extreme logistic regression

Che Ngufor · Janusz Wojtusiak

Received: 6 February 2014 / Revised: 23 November 2014 / Accepted: 13 December 2014 /
Published online: 31 December 2014
© Springer-Verlag Berlin Heidelberg 2014

Abstract Kernel logistic regression (KLR) is a very powerful algorithm that has been shown to be very competitive with many state-of-the-art machine learning algorithms such as support vector machines (SVM). Unlike SVM, KLR can be easily extended to multi-class problems and produces class posterior probability estimates making it very useful for many real world applications. However, the training of KLR using gradient based methods or iterative re-weighted least squares can be unbearably slow for large datasets. Coupled with poor conditioning and parameter tuning, training KLR can quickly design matrix become infeasible for some real datasets. The goal of this paper is to present simple, fast, scalable, and efficient algorithms for learning KLR. First, based on a simple approximation of the logistic function, a least square algorithm for KLR is derived that avoids the iterative tuning of gradient based methods. Second, inspired by the extreme learning machine (ELM) theory, an explicit feature space is constructed through a generalized single hidden layer feedforward network and used for training iterative re-weighted least squares KLR (IRLS-KLR) and the newly proposed least squares KLR (LS-KLR). Finally, for large-scale and/or poorly conditioned problems, a robust and efficient preconditioned learning technique is proposed for learning the algorithms presented in the paper. Numerical results on a series of artificial and 12 real bench-mark datasets show first that LS-KLR compares favorably with SVM and traditional IRLS-KLR in terms of accuracy and learning speed. Second, the extension of ELM to KLR results in simple, scalable and very fast algo-

C. Ngufor (✉)
School of Physics, Astronomy, and Computational Sciences,
George Mason University, Fairfax, VA 22030, USA
e-mail: nguforche@gmail.com; Ngufor.Che@mayo.edu

J. Wojtusiak
Department of Health Administration and Policy,
George Mason University, Fairfax, VA 22030, USA
e-mail: jwojt@mli.gmu.edu

rithms with comparable generalization performance to their original versions. Finally, the introduced preconditioned learning method can significantly increase the learning speed of IRLS-KLR.

Keywords Kernel logistic regression · Extreme learning machine · Classification · Least squares · Kernel matrix · Preconditioner

Mathematics Subject Classification 62H30 · 62H12

1 Introduction

Logistic regression (LR) is a very popular classification algorithm with a sound statistical background that has found widespread use in many fields including machine learning, data mining, and statistics. The popularity of LR can be attributed to its simplicity and interpretability of model parameters. However, LR is a linear classifier whereas most real world classification problems are non-linear. Applying the “kernel trick” to LR as done for support vector machines (SVM), a robust non-linear version of LR is obtained called kernel logistic regression (KLR) which has been proven to be a very powerful classifier (Zhu and Hastie 2005). Unlike SVM, KLR has the advantage that it directly provides estimates of class posterior probabilities and can be easily extended to multi-class problems. As with LR, the training of KLR is typically done through the Newton–Rahpson or iterative re-weighted least squares (IRLS) algorithm (Hastie et al. 2001; Cawley and Talbot 2004). Although several adaptations have been made on these algorithms to improve their performance, computing solutions for KLR is still very challenging especially for very large data sets.

Training KLR and other kernel learning machines requires choosing a kernel function along with kernel parameters. In addition, the models are often regularized. The setting of these parameters is critical for the quality of the results. In practice, one has to (1) chose the right kernel, (2) tune kernel parameters (if any) and, (3) tune the regularization parameter. This procedure is commonly carried out by cross-validation or trial-and-error. However, this can be computationally very expensive and time consuming.

As a first attempt to solve this problem, this paper proposes a least squares solution of KLR based on a simplifying approximation of the logistic function proposed in Ngufor and Wojtusiak (2013). The proposed least squares solution for KLR, termed LS-KLR in the sequel, is found to compete favorably with SVM and the traditional iterative re-weighted least squares KLR (IRLS-KLR) while having a much faster learning speed and is computationally less expensive to deploy. Though the proposed LS-KLR is relatively faster to train, computing the kernel and selection of optimal parameters can still be time consuming.

To address this problem, this paper further presents a new approach for training KLR inspired by the extreme learning machine (ELM) of Huang et al. (2006b) which practically avoids the expensive kernel matrix computation and most of the parameter tuning steps. ELM builds an explicit feature space through a generalized single-hidden layer feed-forward network (SLFN) whose hidden layer need not be tuned (Huang et al. 2006a, 2012). Specifically, ELM randomly chooses the input weights and biases of the

hidden layer and determines the output weights of the SLFN through a simple linear system. Since the determination of the inputs weights and biases are independent of the training data, no tuning is required (Huang et al. 2006a, 2012). This results in an extremely fast algorithm with excellent generalization performance.

Based on this observation, a number of researchers have derived extreme learning machine methods for the SVM by simply plugging in the resulting ELM hidden layer output weight matrix or *ELM-kernel* for the standard SVM kernel. For example, Fréney and Verleysen (2010) uses the plug-in ELM-kernel for training the standard SVM. The authors obtained comparable generalization performance with SVM, however, standard training algorithms for SVM are known to scale at least quadratic in the number of examples which may quickly become infeasible for large datasets. Liu et al. (2008) on the other hand, uses the ELM-kernel for training a version of the least squares support vector machine producing a significantly faster algorithm.

The extension of ELM technique to KLR is yet to be derived. Extending ELM methods to KLR faces similar problems as with SVM. KLR is commonly trained through the iterative re-weighted least squares algorithm which performs the minimization by solving iteration-dependent linear systems recursively. However, this process can be unbearably slow for moderate to very large data sets, in fact IRLS-KLR is computationally more expensive than SVM (Zhu and Hastie 2002; Komarek 2004; Ramani and Fessler 2010). Furthermore, convergence of the method may be troublesome for poor conditioned systems (Ramani and Fessler 2010). This paper extends the extreme learning technique to IRLS-KLR and LS-KLR producing two new algorithms. While the new algorithms show comparable generalization performance to their original versions, they are however relatively very fast to train. Henceforth, the new extreme learning machine algorithms for KLR or simply “extreme logistic regression” (ELR) will be called iterative-reweighted least squares ELR (IRLS-ELR) and least squares ELR (LS-ELR) respectively.

Even with the introduction of extreme learning machine methods to KLR, convergence of IRLS-ELR for large datasets can still be very slow. A very large dataset poses numerous challenges for the computational complexity of a learning algorithm. At worst, the algorithm should scale linearly with data size to be able to learn in a reasonable time. To address this problem, this paper further proposes a large scale learning method that can be used to train IRLS-KLR, LS-KLR, IRLS-ELR and LS-ELR. The method makes use of an efficient approximate inverse preconditioner in a Krylov subspace method for the solution of the linear systems represented by these algorithms.

Given that the classification performance of SVM and KLR are very similar (Zhu and Hastie 2002), the major benefit of introducing ELM methods to KLR is that it produces an algorithm with little or no parameter optimization while maintaining similar generalization performance to SVM and KLR. Thus ELR is very quick to implement and train. Like ELM, ELR can be readily extended to multi-class problems. However, a major advantage of ELR over SVM and ELM is that ELR produces class posterior probability outcomes based on maximum likelihood criterion, whereas SVM and ELM outputs class decision scores. This can be very useful in many real-world applications such as in medical diagnosis where class probabilities are more important than class decisions. In summary, the following contributions are made in this paper:

1. An approximate least squares solution or LS-KLR for KLR is derived. LS-KLR is very simple, fast, and competes favorably to SVM and traditional IRLS-KLR as shown by the numerical experiments performed in this paper.
2. The paper extends ELM technique to IRLS-KLR and the newly proposed LS-KLR to produce highly scalable and computationally less complex algorithms. Like ELM, ELR is very fast to train requiring little or no parameter optimization, and the implementation of ELR is very straight forward.
3. A preconditioning learning approach is proposed for very large and/or poorly conditioned problems. It is shown that the preconditioner can significantly increase the convergence speed of IRLS-KLR and IRLS-ELR. Further more, the preconditioner can be directly applied to train other linear systems such as least squares SVM (LS-SVM) (Suykens and Vandewalle 1999).

Unless otherwise mentioned, the following notations will be adopted throughout the paper: $\mathcal{D} = \{(x_i, y_i), i = 1, \dots, n\}$ is a list of n training examples with $x_i = (x_{i1}, \dots, x_{id})^\top \in \mathbb{R}^d$ a d -dimensional column vector of feature variables and y_i the class label of x_i . For simplicity, only binary classification tasks will be considered where $y_i \in \{0, 1\}$ for KLR and $y_i \in \{-1, 1\}$ for ELM. These class labels are chosen solely for mathematical convenience. \mathbf{I}_n and $\mathbf{1}_n$ are the $n \times n$ identity matrix and $n \times 1$ column vector of ones respectively. x^\top is the transpose of x , i.e. a row vector. Boldfaced type letters represent either matrices or vectors, the distinction will be clear from the context. Further, it is assumed that the observations in \mathcal{D} are independent and identically distributed.

The paper is hence organized as follows: Sect. 2 provides a brief review of LR and KLR models. Parameter estimation of KLR is performed by the iterative re-weighted least squares algorithm. Section 3 presents a least squares approximation of IRLS-KLR to improve the scalability and lessen the computational complexity of the method. To understand the extension of ELM to KLR and for comparison purposes, Sect. 4 briefly describes the constrained optimization based version of ELM introduced in Huang et al. (2012). Section 5 then extends ELM technique to KLR to derive ELR. Section 6 presents a new preconditioned learning strategy for KLR suitable for large-scale problems. Experiments on artificial and real datasets are performed in Sect. 7 while Sect. 8 concludes the paper.

2 Logistic regression and kernel logistic regression

This section briefly reviews logistic regression (LR) and its kernelized version (KLR). The iterative re-weighted least square algorithm (IRLS) is then presented to numerically solve the KLR maximum likelihood equations.

2.1 Logistic regression

LR is a popular classification method that makes no assumption about the distribution of the independent variables. The fundamental assumption of the method is that the log-odds or “logit” transformation of the class posterior probability $\pi(x) = \Pr(y = 1|x)$ is a linear combination of the independent variables i.e.

$$\eta(x) = \log\left(\frac{\pi(x)}{1 - \pi(x)}\right) = \beta_0 + \boldsymbol{\beta}^\top x \tag{1}$$

where β_0 and $\boldsymbol{\beta} = (\beta_1, \dots, \beta_d)^\top$ are the unknown parameters of the model. The constant β_0 is the intercept also known as the ‘‘bias’’. Thus LR models the posterior probability of the binary response variable y taking the value $y = 1$ depending on a number of independent variables x by the logistic function

$$\pi(x) = \Pr(y = 1|x; \boldsymbol{\beta}) = 1/(1 + \exp(-\beta_0 - \boldsymbol{\beta}^\top x)).$$

For mathematical convenience, a constant variable 1 will be included in the vector x to account for the bias term and β_0 included in $\boldsymbol{\beta}$. Then the regularized negative log-likelihood function can be written as

$$l(\boldsymbol{\beta}) = \frac{1}{2} \|\boldsymbol{\beta}\|^2 - \frac{\gamma}{2} \sum_{i=1}^n \left[y_i \boldsymbol{\beta}^\top x_i - \log(1 + \exp(\boldsymbol{\beta}^\top x_i)) \right]$$

where the parameter γ reflects the strength of regularization.

Typically, the method of maximum likelihood is used to estimate the model parameters. To solve for $\boldsymbol{\beta}$, the negative log-likelihood equation is differentiated with respect to $\boldsymbol{\beta}$ and set to zero

$$\frac{\partial l(\boldsymbol{\beta})}{\partial \boldsymbol{\beta}} = \boldsymbol{\beta} - \frac{\gamma}{2} \sum_{i=1}^n x_i (y_i - \pi_i(x_i)) = 0. \tag{2}$$

Unfortunately, (2) is nonlinear and there is no closed-form solution. The traditional approach is to approximately solve it using iterative methods.

There are many numerical optimization methods that can provide efficient solutions to (2). The Newton–Raphson method is perhaps the first goto off-the-shelf method to use. The method takes the first degree Taylor series approximation of (2) at a point $\boldsymbol{\beta}^{old}$ (initial guess), set this to zero and solve for a new approximate solution $\boldsymbol{\beta}^{new}$. The update process is repeated until convergence. A full treatment of the Newton–Raphson algorithm and other equivalent maximization techniques can be found in standard statistics text such as [Hastie et al. \(2001\)](#).

One major disadvantage of LR is that it is a linear classifier. It assumes that the outcome or log-odd is a linear function of the independent variables. However, most real world classification problems are non-linear, and so classical logistic regression cannot capture any non-linearity that may exist in the data. The next section describes the kernel version of LR to account for any non-linear relationship that may exist between the outcome and the independent variables.

2.2 Kernel logistic regression

Kernel learning machines has gained a lot of interest since the beginning of the mid 1990’s. Their widespread use can be attributed to their mathematical elegance and

excellent performance. Numerous nonlinear extensions of traditional machine learning classification techniques have been proposed based on the so-called “kernel trick”. The kernelized versions of these algorithms have shown state-of-the-art performances over a wide range of benchmark datasets and real-world problems.

Using the kernel trick, a kernelized version of the LR, i.e., KLR can be easily constructed. A mapping function $\phi(\cdot)$ is chosen to convert the non-linear relationship between the response and the independent variable into a linear relationship in a higher dimensional feature space. Thus the function maps x from a lower dimensional to a higher (and possibly infinite) dimensional feature space

$$\phi : x \in \mathbb{R}^d \rightarrow \phi(x) \in \mathbb{R}^{d_f}, \quad (3)$$

where d_f is the dimension of the feature space. The mapping function is however usually unknown, but dot products in the feature space can be expressed in terms of the input vector through the kernel function

$$K(x_1, x_2) = \phi(x_1)^\top \phi(x_2)$$

for any pair of data points x_1 and x_2 . The kernel function must satisfy Mercer’s condition (Mercer 1909), i.e for it to be expressed as inner product in the feature space, the function must be positive semi-definite. Common choices for the kernel function include: $K(x_1, x_2) = x_1^\top x_2$ (linear kernel), $K(x_1, x_2) = (x_1^\top x_2 + c)^p$ (polynomial kernel of degree p with $c \geq 0$ a tuning parameter), $K(x_1, x_2) = \exp(-\sigma \|x_1 - x_2\|^2)$ (radial basis function kernel with $\sigma > 0$ a tuning parameter).

Using this feature mapping, the logit transformation can be written as

$$\eta(x) = \log \left(\frac{\pi(x)}{1 - \pi(x)} \right) = \beta^\top \phi(x) + b$$

where

$$\pi(x) = \Pr(y = 1|x; \beta) = 1/(1 + \exp(-\beta^\top \phi(x) - b))$$

is the class posterior probability and b is the bias term. With this transformation, the regularized negative log-likelihood function becomes

$$l(\beta) = \frac{1}{2} \|\beta\|^2 - \frac{\gamma}{2} \sum_{i=1}^n \left[y_i (\beta^\top \phi(x_i) + b) - \log (1 + \exp(\beta^\top \phi(x_i) + b)) \right]. \quad (4)$$

As with LR, the maximum likelihood estimates of the parameters of KLR can be obtained through the Newton–Raphson algorithm.

2.3 Iterative re-weighted least squares KLR

Similar to the derivation of the LR model, to facilitate the derivation of the IRLS algorithm for KLR a constant variable 1 is included in the vector $\phi(x)$ and the bias term b is included in β . The minimization of (4) can be carried out by setting the derivative with respect to β to zero and using Newton–Raphson method to iteratively solve the score equation. It can be shown that the Newton–Raphson update formula is equivalent to an iterative re-weighted least squares step [see for example [Zhu and Hastie \(2005\)](#) and [Katz et al. \(2005\)](#)] which can be written as

$$\beta^{new} = \left(\Phi^T W \Phi + \frac{1}{\gamma} I_{d_f} \right)^{-1} \Phi^T W z \tag{5}$$

where W is a $n \times n$ diagonal matrix with i -th element $w_i = \pi(x_i)(1 - \pi(x_i))$, $Y = (y_1, \dots, y_n)^T$, $\pi = (\pi(x_1), \dots, \pi(x_n))^T$, I_{d_f} is a $d_f \times d_f$ identity matrix, Φ is a $n \times (d_f + 1)$ matrix containing $\phi(x_i)^T$ in its i -th row, and

$$z = \Phi \beta^{old} + W^{-1}(Y - \pi).$$

It can be seen that, if z is regarded as a new response variable or adjusted response, then (5) is the solution of the following weighted regularized least squares problem:

$$\min_{\beta} L = \frac{1}{2} \|\beta\|^2 + \frac{\gamma}{2} (z - \Phi \beta)^T W (z - \Phi \beta). \tag{6}$$

This can also be further simplified as

$$\min_{\beta, \varepsilon_i} L = \frac{1}{2} \|\beta\|^2 + \frac{\gamma}{2} \sum_{i=1}^n w_i \varepsilon_i^2 \tag{7}$$

where $\varepsilon_i = z_i - \beta^T \phi(x_i)$.

Equation (7) can be written as a constrained optimization problem thus

$$\min_{\beta, \varepsilon_i} L = \frac{1}{2} \|\beta\|^2 + \frac{\gamma}{2} \sum_{i=1}^n w_i \varepsilon_i^2 \tag{8}$$

$$\text{subject to : } z_i = \beta^T \phi(x_i) + b + \varepsilon_i, \quad \forall i = 1, \dots, n. \tag{9}$$

where the bias term b has been re-introduced. The Lagrangian function for the optimization problem (8), (9) is given by

$$\mathcal{L}(\beta, b, \alpha, \varepsilon) = \frac{1}{2} \|\beta\|^2 + \frac{\gamma}{2} \sum_{i=1}^n w_i \varepsilon_i^2 - \sum_{i=1}^n \alpha_i (\beta^T \phi(x_i) + b + \varepsilon_i - z_i) \tag{10}$$

where $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n) \in \mathbb{R}^n$ is the Lagrange multipliers. The optimality condition for this problem can be expressed as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = 0 \implies \boldsymbol{\beta} = \sum_{i=1}^n \alpha_i \boldsymbol{\phi}(x_i) \quad (11)$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 \implies \sum_{i=1}^n \alpha_i = 0 \quad (12)$$

$$\frac{\partial \mathcal{L}}{\partial \varepsilon_i} = 0 \implies \alpha_i = \gamma w_i \varepsilon_i, \quad \forall i = 1, \dots, n \quad (13)$$

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \implies \boldsymbol{\beta}^\top \boldsymbol{\phi}(x_i) + b + \varepsilon_i = z_i, \quad \forall i = 1, \dots, n. \quad (14)$$

Using (11) and (13) to eliminate $\boldsymbol{\beta}$ and $\boldsymbol{\varepsilon}$ from (14) gives the linear system

$$\begin{aligned} \sum_{i=1}^n \alpha_i \boldsymbol{\phi}(x_i)^\top \boldsymbol{\phi}(x_j) + b + \frac{1}{\gamma} \frac{\alpha_i}{w_i} &= z_j, \quad \forall j = 1, \dots, n \\ \sum_{i=1}^n \alpha_i &= 0. \end{aligned}$$

This system can be written in matrix form as

$$\begin{pmatrix} \mathbf{K} + \frac{1}{\gamma} \mathbf{W}^{-1} & \mathbf{1}_n \\ \mathbf{1}_n^\top & 0 \end{pmatrix} \begin{pmatrix} [l] \boldsymbol{\alpha} \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{z} \\ 0 \end{pmatrix} \quad (15)$$

where $\mathbf{K} = \boldsymbol{\Phi} \boldsymbol{\Phi}^\top$ is the $n \times n$ kernel matrix and

$$\mathbf{z} = \mathbf{K} \boldsymbol{\alpha} + b \mathbf{1}_n + \mathbf{W}^{-1} (\mathbf{Y} - \boldsymbol{\pi}). \quad (16)$$

A different approach to derive the linear system (15) can be found in [Cawley and Talbot \(2008\)](#). The IRLS-KLR algorithm proceeds iteratively, updating $\boldsymbol{\alpha}$ and b according to (15) and then updating \mathbf{z} according to (16).

The recursive training of IRLS-KLR can be very slow for very large data sets ([Komarek 2004](#); [Ramani and Fessler 2010](#)). The recursion is brought about by the dependence on non-linear terms in the weight matrix \mathbf{W} . Recall that \mathbf{W} depend non-linearly on the parameters $\boldsymbol{\alpha}$ and b through the posterior probabilities $\boldsymbol{\pi} = 1/(1 + \exp(-(\mathbf{K} \boldsymbol{\alpha} + b \mathbf{1}_n)))$. The next section presents an approximation to IRLS-KLR which eliminates the non-linear dependence and hence speeding up the training process.

3 Least squares kernel logistic regression

The training of most kernel machines often scales poorly, with running times that are at least quadratic or cubic in the number of training examples ([Fine and Scheinberg](#)

2002; Kulis et al. 2006). For example, the computational complexity of the traditional training of SVM is $\mathcal{O}(n^2m)$ where n is the number of training points and m the number of support vectors. The situation is even worse for IRLS-KLR which is cubic $\mathcal{O}(n^3)$ in the number of training points (Zhu and Hastie 2002). Furthermore, the computation of the kernel matrix requires $\mathcal{O}(n^2)$ memory overhead, which may be prohibitive for large-scale learning tasks (Kulis et al. 2006).

3.1 Least squares versions of kernel learning machines

Various approaches have been proposed to simplify the implementation and increase training speed of kernel learning machines. For example, Suykens and Vandewalle (1999) proposed the LS-SVM technique for training SVM by replacing the in-equality constraint with an equality constraint leading to a simple linear system. Thus LS-SVM is easier and quicker to implement than SVM (Kuh 2004). Though training LS-SVM can still be impractical for large datasets, a number of schemes have been proposed to solve this problem. For example, Suykens et al. (1999) presented an iterative algorithm based on conjugate gradient, Keerthi and Shevade (2003) proposed a sequential minimal optimization (SMO) algorithm for LS-SVM while Chu et al. (2005) proposed an improved conjugate gradient method. Other methods to speed up training of LS-SVM based on introducing some form of sparseness in the linear system can be found in Suykens et al. (2000, 2002a), De Kruijff and De Vries (2003), Zeng and Chen (2005) and Jiao et al. (2007).

Many tests and comparisons have shown the great performance of LS-SVM on several benchmark classification problems (Gestel et al. 2002) which motivated the extension of LS-SVM formulation to many other methods including kernel principal component analysis, kernel canonical correlation analysis, recurrent networks, and optimal control (Suykens et al. 2002b). No such linear system or LS-SVM type formulation exist for KLR.

KLR is traditionally trained through the Newton–Raphson method as described in Sect. 2. However, each step of the iteration requires the inversion of a $n \times n$ matrix leading to a $\mathcal{O}(n^3)$ computational cost. A few approaches have been proposed to improve the convergence of KLR. For example Keerthi et al. (2005) proposed an SMO type algorithm while Zhu and Hastie (2005) proposed the *import vector machines* which is a sparse method for training KLR. Similar to the support vectors of SVM, the import vectors are a few selected data points that defines the decision hyperplane in the feature space of KLR. As with LS-SVM, these methods can be applied to a least squares version of KLR to obtain similar benefits. The next section describes a derivation of such a least squares solution for KLR.

3.2 Least squares approximation of KLR

The Newton–Raphson method and its variants obtain approximate solution of LR through a first order Taylor series approximation of the maximum likelihood score equations (2). Instead of using the Taylor expansion of the score equation, Ngufor and Wojtusiak (2013) replaced the logistic function by its first order Taylor expansion about the decision boundary $\eta(x) = 0$ in the score equation. This led to a simple least squares

solution for LR which was shown to have excellent generalization and scalability properties compared to popular gradient based methods such as the stochastic gradient descent.

Using this idea, a similar least squares method can be derived for KLR. However, unlike the approach presented in [Ngufor and Wojtusiak \(2013\)](#), the first order Taylor expansion of the logistic function is applied directly to the IRLS-KLR linear system in (15).

Note first that the Taylor series expansion of the logistic function $f(x) = 1/(1 + e^{-x})$ around $z = 0$ is given by

$$f(z) = \frac{1}{2} + \frac{1}{4}z - \frac{1}{48}z^3 + \frac{1}{480}z^5 - \frac{17}{80,640}z^7 + \dots$$

The decision boundary for KLR is given by $\mathbf{K}\boldsymbol{\alpha} + b\mathbf{1}_n = \mathbf{0}$ (or $\boldsymbol{\phi}(x)\boldsymbol{\beta} + b = 0$). Points on the boundary have equal chances to be classified to either class i.e. $\pi(x) = 0.5$, thus they are more informative for learning. The learning of KLR can thus be restricted to learning the characteristic of points on the boundary. Substituting the first order Taylor series expansion of π about $\mathbf{K}\boldsymbol{\alpha} + b\mathbf{1}_n = \mathbf{0}$ in the weight matrix of IRLS-KLR and keeping only first order terms reduces the matrix to identity i.e. $\mathbf{W} = \frac{1}{4}\mathbf{I}_n$. A similar procedure reduces \mathbf{z} in the right hand side of (15) to $\mathbf{z} = 4(\mathbf{Y} - \frac{1}{2}\mathbf{1}_n)$. Thus the IRLS-KLR linear system reduces to a non-iterative least squares system or LS-KLR given by

$$\begin{pmatrix} \mathbf{K} + \frac{4}{\gamma}\mathbf{I}_n & \mathbf{1}_n \\ \mathbf{1}_n^T & 0 \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ b \end{pmatrix} = \begin{pmatrix} 4(\mathbf{Y} - \frac{1}{2}\mathbf{1}_n) \\ 0 \end{pmatrix}. \quad (17)$$

This is a simple linear system similar to LS-SVM that can be solved directly without the iterative steps of IRLS-KLR. Similar to LS-SVM, SMO type and sparse methods can be applied to LS-KLR.

Figure 1 illustrates the superiority of kernelized logistic regression over traditional logistic regression for a non-linear classification problem. It shows how LR can perform very poorly on a dataset with non-linear decision boundaries while KLR accurately identify all non-linear decision boundaries. The decision boundaries are shown by the shaded gray contour lines. The Banana dataset ([Alcalá-Fdez et al. 2009](#)) is a simple two dimensional artificial binary classification dataset where data points belong to several clusters with banana shapes. As seen from Fig. 1a LR is unable to identify any of the clusters correctly. The simple LS-KLR with the Gaussian kernel ($K(x, y) = \exp(-\sigma\|x - y\|^2)$) on the other hand in Fig. 1b correctly captures all clusters with an accuracy of over 90 % (parameters for LS-KLR: $\gamma = 100$, $\sigma = 1.5$).

Though kernel learning machines such as SVM, LS-SVM, IRLS-KLR and LS-KLR are conceptually appealing and have excellent performance, their training can be very difficult in practice. Computing the kernel can be expensive and considerable amount of time is usually spend on selection of model parameters such as the regularization constant γ and additional kernel parameters. In addition, these methods are known to be sensitive to the combination of training parameters. For example SVM and LS-

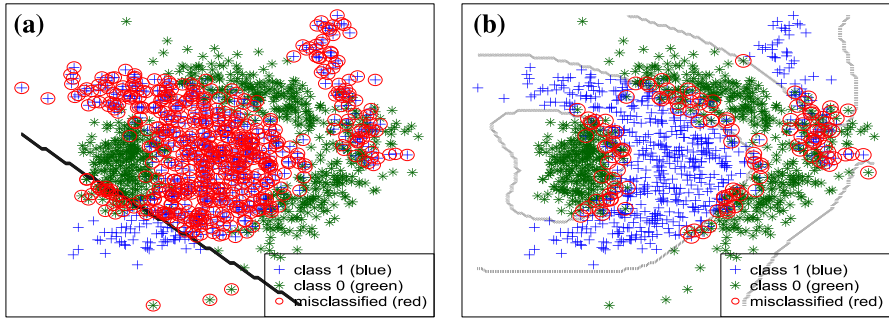


Fig. 1 Decision boundaries of LR and KLR for Banana dataset. The (red) circles represents the missclassified data points by the classifiers. See the electronic version of ADAC for the blue, green, and red color illustrations. **a** LR, **b** LS-KLR (color figure online)

SVM with the Gaussian kernel are sensitive to the combination of (γ, σ) (Huang et al. 2012). The next section briefly describes a recent popular learning method that was proposed to mitigate these problems in the case of SLFNs.

4 Extreme learning machine

The extreme learning machine (ELM) is a relatively recent algorithm that was proposed by Huang et al. (2006b) based on the SLFN. ELM randomly selects the SLFN input weights and biases and computes the output weights through a minimal norm least squares approach. Thus, the overall computational time for training and model selection is reduced by several amounts.

To understand the derivation of the ELR algorithms proposed in this paper, this section briefly describes the basic idea of ELM. However, to simplify comparability with ELR, the constrained-optimization-based version of ELM (Huang et al. 2012) will be presented. For the original idea and theory see Huang et al. (2006b).

4.1 Constrained-optimization-based ELM

Consider a set of n training examples (x_i, y_i) with $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$; the output function of ELM for generalized SLFNs with p hidden neurons is given by

$$f(x_i) = \sum_{j=1}^p \beta_j \phi(x_i) = \beta^T \phi(x_i), \quad i = 1, \dots, n$$

where $\beta = (\beta_1, \dots, \beta_p)^T$ is the vector of output weights and ϕ here represents the activation or output function of the network. The decision function of ELM classifier for the data point x_i is given by $f(x_i) = \text{sgn}(\beta^T \phi(x_i))$.

From Huang et al. (2012), the constrained-optimization problem of regularized ELM is stated as

$$\min_{\beta, \varepsilon_i} L = \frac{1}{2} \|\beta\|^2 + \frac{\gamma}{2} \sum_{i=1}^n \varepsilon_i^2 \tag{18}$$

$$\text{subject to : } \boldsymbol{\beta}^\top \boldsymbol{\phi}(x_i) = y_i - \varepsilon_i, \quad \forall i = 1, \dots, n, \quad (19)$$

where ε_i is the training error of the output node with respect to the training example x_i and $y_i \in \{-1, 1\}$ is the class label of x_i . The Lagrangian for this optimization problem is

$$\mathcal{L}(\boldsymbol{\beta}, \boldsymbol{\alpha}, \boldsymbol{\varepsilon}) = \frac{1}{2} \|\boldsymbol{\beta}\|^2 + \frac{\gamma}{2} \sum_{i=1}^n \varepsilon_i^2 - \sum_{i=1}^n \alpha_i (\boldsymbol{\beta}^\top \boldsymbol{\phi}(x_i) - y_i + \varepsilon_i) \quad (20)$$

and the optimality conditions can be expressed as

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\beta}} = 0 \implies \boldsymbol{\beta} = \sum_{i=1}^n \alpha_i \boldsymbol{\phi}(x_i) = \boldsymbol{\Phi}^\top \boldsymbol{\alpha} \quad (21)$$

$$\frac{\partial \mathcal{L}}{\partial \varepsilon_i} = 0 \implies \alpha_i = \gamma \varepsilon_i, \quad \forall i = 1, \dots, n \quad (22)$$

$$\frac{\partial \mathcal{L}}{\partial \alpha_i} = 0 \implies \boldsymbol{\beta}^\top \boldsymbol{\phi}(x_i) - y_i + \varepsilon_i = 0, \quad \forall i = 1, \dots, n \quad (23)$$

where $\boldsymbol{\Phi}$ is the $n \times p$ hidden layer output matrix with $\boldsymbol{\phi}(x_i)^\top$ in its rows and $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_n)^\top \in \mathbb{R}^n$ is the Lagrange multipliers. Equations (21) and (22) can be substituted in (23) to eliminate $\boldsymbol{\beta}$ and $\boldsymbol{\varepsilon}$ giving

$$\left(\boldsymbol{\Phi} \boldsymbol{\Phi}^\top + \frac{1}{\gamma} \mathbf{I}_n \right) \boldsymbol{\alpha} = \mathbf{Y} \quad (24)$$

Surprisingly, except for the bias term and the kernel matrix, the first linear system of LS-KLR given by (17) is very similar to the constrained ELM given by (24). This similarity suggest ELM can be easily extended to LS-KLR.

5 Extreme logistic regression

In the ELM framework, one can think of the hidden-layer as representing a mapping $\boldsymbol{\phi}$ from the input space $x \in \mathbb{R}^d$ to a feature space $\boldsymbol{\phi}(x) \in \mathbb{R}^p$ where a regularized linear system is solved. Thus the learning process of ELM proceeds in two steps: (1) The input data is mapped to the hidden-layer (a higher dimensional feature space) using any nonlinear piecewise continuous function (Huang et al. 2012), such as the Sigmoid or Gaussian function. The parameters (weights and biases) of the hidden layer map function are randomly generated according to any continuous probability distribution. (2) A regularized least squares solution is obtained through (24), where $\boldsymbol{\Phi}$ is the hidden-layer output matrix. This is very similar to the use of kernels in kernel learning machines except that the map $\boldsymbol{\phi}$ in (3) is unknown whereas it is explicitly computed in ELM. Thus the hidden-layer of ELM can be regarded as a ‘‘ELM-trick’’ for defining a randomized *ELM-kernel* given by $\mathbf{K}_{ELM} = \boldsymbol{\Phi} \boldsymbol{\Phi}^\top$.

Based on these observations, Fréney and Verleysen (2010) and Liu et al. (2008) have derived extreme learning methods for SVM by simply plugging in the corresponding ELM hidden-layer output matrix or the ELM-kernel for the standard SVM kernel. The

authors obtained comparable generalization performance to SVM but at a much lesser computational cost. In Fréney and Verleysen (2010), the ELM-kernel was used to train standard quadratic SVM and only the regularization parameter needed to be turned. The number of hidden-layer neurons p is also a parameter to be tuned, however it has been shown in many works on ELM that setting p to a sufficiently large number such as $p \geq 10^3$ is sufficient to get good generalization (Huang et al. 2010; Fréney and Verleysen 2010; Huang et al. 2012).

To the best of the knowledge of the authors of this paper, extreme learning methods have not yet been derived for KLR. This paper proposes to use the ELM-kernel $\mathbf{K}_{ELM} = \Phi\Phi^T$ to train IRLS-KLR and the new LS-KLR. That is, the random ELM-kernel is substituted for the standard KLR kernel in (15) for learning IRLS-KLR and in (17) for learning LS-KLR. The extension of ELM to KLR is called extreme logistic regression (ELR) and the two algorithms IRLS-KLR and LS-KLR under ELR are called IRLS-ELR and LS-ELR respectively. The new ELR linear systems can be written as

1. IRLS-ELR

$$\begin{pmatrix} \Phi\Phi^T + \frac{1}{\gamma}\mathbf{W}^{-1} & \mathbf{1}_n \\ \mathbf{1}_n^T & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{z} \\ 0 \end{pmatrix} \tag{25}$$

where $\mathbf{z} = \Phi\Phi^T\alpha + b\mathbf{1}_n + \mathbf{W}^{-1}(\mathbf{Y} - \boldsymbol{\pi})$.

2. LS-ELR

$$\begin{pmatrix} \Phi\Phi^T + \frac{4}{\gamma}\mathbf{I}_n & \mathbf{1}_n \\ \mathbf{1}_n^T & 0 \end{pmatrix} \begin{pmatrix} \alpha \\ b \end{pmatrix} = \begin{pmatrix} 4(\mathbf{Y} - \frac{1}{2}\mathbf{1}_n) \\ 0 \end{pmatrix}. \tag{26}$$

Since Φ is explicitly computed in ELR, using Eq. (11) the system of equations above can also be written in terms of β and b . The training of ELR proceeds as in Algorithm 1.

Algorithm 1: Extreme Logistic Regression

Input : $\mathcal{D} = \{(x_i, y_i) | x_i \in \mathbb{R}^d, y_i = 0, 1, i = 1, \dots, n\}$,
 hidden-layer map function $\phi(\cdot)$,
 ELR method: IRLS-ELR or LS-ELR

Output: Parameters: (α, b)

- 1 Randomly generate the weights $\{(\mathbf{w}_i, b_{0_i})\}_{i=1}^n$ according to any continuous probability distribution,
 - 2 Compute the ELM-kernel: $\mathbf{K}_{ELM} = \Phi\Phi^T$,
 - 3 Solve the linear systems (29) and (30) iteratively or directly according to the ELR method.
-

Commonly used ELM hidden-layer output map functions include the Sigmoid function $\phi(x) = 1/(1 + \exp(-\mathbf{w}^T x - b_0))$ with $\mathbf{w} \in \mathbb{R}^d, b_0 \in \mathbb{R}$ and the Gaussian

function $\phi(x) = \exp(-b_0\|x - \mathbf{w}\|^2)$ with $b_0 > 0$. The values $\{(\mathbf{w}_j, b_{0j})\}_{j=1}^p$ are randomly generated according to any continuous probability distribution.

The training of IRLS-ELR proceeds iteratively, however the computational cost is now significantly less compared to IRLS-KLR. Since the convergence of IRLS-ELR can still be very slow, optionally, when time and high accuracy requirements are essential, the vary fast solution from LS-ELR can be used to initialize IRLS-ELR. Experiments on the datasets used in this paper showed that IRLS-ELR always converges when initialize with LS-ELR and the convergence time can be reduce in some cases to more than half. However, results from LS-ELR competes rather favorably to the iterative methods.

6 An approximate inverse preconditioner for KLR and ELR

Equations (15), (17), (25) and (26) can be written in a general form as

$$\underbrace{\begin{pmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^\top & 0 \end{pmatrix}}_{\mathcal{A}} \begin{pmatrix} \boldsymbol{\alpha} \\ b \end{pmatrix} = \begin{pmatrix} \mathbf{f} \\ 0 \end{pmatrix}. \tag{27}$$

In its general form, the coefficient matrix \mathcal{A} in (27) arises from the solution of equality constrained quadratic programming or saddle point problems (Benzi et al. 2005). It is known that if the 1×1 block matrix \mathbf{A} is non-singular, then \mathcal{A} is invertible if and only if the Schur complement of \mathbf{A} : $\mathbf{S} = \mathbf{B}^\top \mathbf{A}^{-1} \mathbf{B}$ is non-singular (Benzi et al. 2005). In this case however, the Schur complement \mathbf{S} is a 1×1 matrix i.e a scalar, thus the block inverse of \mathcal{A} can be expressed as

$$\mathcal{A}^{-1} = \begin{pmatrix} \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{B} \mathbf{S}^{-1} \mathbf{B}^\top \mathbf{A}^{-1} & \mathbf{A}^{-1} \mathbf{B} \mathbf{S}^{-1} \\ \mathbf{S}^{-1} \mathbf{B}^\top \mathbf{A}^{-1} & -\mathbf{S}^{-1} \end{pmatrix}. \tag{28}$$

This shows in particular that the solution of (27) is given by

$$\begin{aligned} \boldsymbol{\alpha} &= (\mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{B} \mathbf{S}^{-1} \mathbf{B}^\top \mathbf{A}^{-1}) \mathbf{f} \\ b &= \mathbf{S}^{-1} \mathbf{B}^\top \mathbf{A}^{-1} \mathbf{f} \end{aligned}$$

By defining the vector $\mathbf{z}_0 = \mathbf{A}^{-1} \mathbf{f}$, the solutions above can be further simplified to

$$b = \mathbf{S}^{-1} \mathbf{B}^\top \mathbf{z}_0 \tag{29}$$

$$\boldsymbol{\alpha} = \mathbf{z}_0 - \mathbf{A}^{-1} \mathbf{B} b. \tag{30}$$

This simple transformation of the direct solution greatly reduces the number of matrix-matrix and matrix-vector multiplication and can thus speed up the computation.

Assuming \mathbf{A}^{-1} exist, these two equations are iterated recursively in IRLS-KLR and IRLS-ELR until convergence or a specified maximum number of iterations is reached. For LS-KLR and LS-ELR, these are simply the direct solutions of the system.

The simple direct solutions represented by (29) and (30) has the disadvantage that as the size of the matrix \mathbf{A} becomes large, the computational effort grows in the order of

n^3 for dense systems. Iterative methods such as krylov subspace methods are typically used to solve such linear systems. The amount of reduction in computational time by an iterative method depends on the spectral properties of the coefficient matrix which in turn determines the convergence rate of the method (Benzi 2002). Most real world application problems leading to weighted least squares systems may produce very poor conditioned matrices, thus making the convergence rate of the iterative solver to be unacceptably slow (Ramani and Fessler 2010). Convergence can be accelerated by the use of robust and efficient preconditioners. This section presents an approximate inverse preconditioner for the solution of KLR and ELR for large scale problems based on the generalized inverse \mathcal{A}^{-1} of \mathcal{A} .

6.1 Preconditioned KLR and ELR

Preconditioning amounts to transforming the original system into one having the same solution but with more favorable spectral properties (Benzi 2002). A preconditioner is a matrix that affects such a transformation. Specifically, if \mathcal{P} is a nonsingular matrix which approximates \mathcal{A}^{-1} , then instead of solving $\mathcal{A}\mathbf{x} = \mathbf{b}$, the equivalent but much simpler system $\mathcal{P}\mathcal{A}\mathbf{x} = \mathcal{P}\mathbf{b}$ can be solved. In general a good preconditioner \mathcal{P} is chosen such that the preconditioned system is easy to solve and the preconditioner should be cheap to construct and apply. With a robust and efficient preconditioner, the computing time for the preconditioned system should be sign cantly less than that for the unpreconditioned one.

Based on the generalized inverse of \mathcal{A} given by (28), Le Borne and Ngufor (2010) proposed an implicit approximate inverse preconditioner for the saddle point problem which exploits the 2×2 block structure but does not require any additional information about the matrix or underlying problem. Since very little information may be available about the underlying problem leading up to the system represented by (27), this preconditioning technique is therefore suitable for the system.

Based on this idea, a simple approximate inverse preconditioner for IRLS-KLR, LS-KLR, IRLS-ELR and LS-ELR systems is defined as

$$\mathcal{P} = \begin{pmatrix} \tilde{\mathbf{A}}^{-1} - \tilde{\mathbf{A}}^{-1}\mathbf{B}\tilde{\mathbf{S}}^{-1}\mathbf{B}^\top\tilde{\mathbf{A}}^{-1} & \tilde{\mathbf{A}}^{-1}\mathbf{B}\tilde{\mathbf{S}}^{-1} \\ \tilde{\mathbf{S}}^{-1}\mathbf{B}^\top\tilde{\mathbf{A}}^{-1} & -\tilde{\mathbf{S}}^{-1} \end{pmatrix} \tag{31}$$

where $\tilde{\mathbf{A}}^{-1}$ and $\tilde{\mathbf{S}}^{-1}$ are some good approximations of \mathbf{A}^{-1} and \mathbf{S}^{-1} respectively. However, since \mathbf{S} is just a scalar, the problem reduces to finding a good approximation of \mathbf{A}^{-1} which in turn reduces to finding a suitable approximation of the kernel matrix \mathbf{K} .

6.2 Approximating the kernel matrix

Any positive definite matrix \mathbf{K} can be represented by its Cholesky factorization $\mathbf{K} = \mathbf{Q}\mathbf{Q}^\top$, where \mathbf{Q} is a lower triangular matrix. If \mathbf{K} is positive semi-definite and possibly

singular, it is still possible to compute an “incomplete” Cholesky factorization (ILU) where some columns of \mathbf{Q} are zero. In particular, in the case where \mathbf{K} is a kernel matrix, an attractive factorization suitable for large-scale learning problems is the predictive low-rank incomplete Cholesky decomposition (pILU) (Bach and Jordan 2005). pILU makes use of side information such as the class labels for classification and the continuous outputs for regression to compute the low-rank decomposition. Using these decompositions, an explicit computation of the inverse of \mathbf{K} can be avoided by solving linear systems using approximate inverse preconditioners. The preconditioned linear system can then be solved using any appropriate Krylov subspace method.

The main difficulty in computing the approximate inverse preconditioner \mathcal{P} lies in finding a suitable approximate inverse of the 1×1 block matrix \mathbf{A} which in the case of IRLS-KLR (or IRLS-ELR) is given by $\mathbf{A} = \mathbf{K} + \frac{1}{\gamma}\mathbf{W}$. However, since kernel matrices usually have low ranks, \mathbf{K} can be approximated by its ILU or pILU such that $\mathbf{K} \approx \mathbf{Q}\mathbf{Q}^\top$ where \mathbf{Q} is an $n \times q$ matrix with $q \ll n$. Even if the kernel has full rank, it is still possible to approximate it by a low rank positive semidefinite matrix (Fine and Scheinberg 2002).

With this approximations, the inverse (approximate) of \mathbf{A} can be obtained using the Woodbury matrix identity (Hager 1989),

$$\begin{aligned}\tilde{\mathbf{A}}^{-1} &= (\mathbf{Q}\mathbf{Q}^\top + \frac{1}{\gamma}\mathbf{W})^{-1} \\ &= \gamma\mathbf{W}^{-1} - \gamma^2\mathbf{W}^{-1}\mathbf{Q}\underbrace{(\mathbf{I}_q + \gamma\mathbf{Q}^\top\mathbf{W}^{-1}\mathbf{Q})^{-1}}_{q \times q}\mathbf{Q}^\top\mathbf{W}^{-1}.\end{aligned}$$

Thus the inversion of an $n \times n$ matrix has been reduced to the inversion of a $q \times q$ matrix, and the speed up in computation time can be substantial. Depending on the application and computational complexity of pILU, q can be set to as small as desired. To complete the derivation of the preconditioner, a suitable approximation to \mathbf{A}^{-1} can be taken as $\tilde{\mathbf{A}}^{-1}$ with $\mathbf{W} = \mathbf{I}_n$.

Note that the use of pILU to compute the preconditioner can be seen as a learning procedure where the data is used to learn the approximate inverse preconditioner. Note also that incomplete Cholesky factorization of the ELM-kernel is not necessary since a decomposed form of the matrix is already available.

6.3 A preconditioned MINRES solver for KLR and ELR

With the approximate inverse preconditioner, a preconditioned minimal residual (MINRES) method can be used to solve the linear systems presented in this paper. Other Krylov subspace methods such as the well-known conjugate gradient (CG) can equally be used. However, the coefficient matrix \mathcal{A} in (27) is not positive definite (Cawley and Talbot 2008), some transformations are required in order to use the CG method (Suykens et al. 1999). The MINRES method on the other hand can be applied to both symmetric positive definite and indefinite systems. Research has indicated that MINRES can converge faster than CG on symmetric positive definite systems (Hogben 2006).

In the preconditioned step of the MINRES algorithm, one can take advantage of the reduced form of the direct solution given by (29) and (30) to reduce the number of matrix-matrix or matrix-vector multiplications. In some of the experiments reported in Sect. 7, the preconditioned MINRES algorithm as presented in Algorithm 2 converges in as little as 5–10 steps with a relative norm residual less than or equal to 1×10^{-5} . Further details of the MINRES algorithm can be found in standard linear algebra text such as Hogben (2006).

Algorithm 2: Preconditioned MINRES solver for KLR and ELR

Input : Coefficient matrix \mathcal{A} and right hand side vector b of equation $\mathcal{A}x = b$ and preconditioner \mathcal{P} .

Output: Solution x .

```

1 Choose initial guess  $x_0$  ;
2 Set  $r_0 = \mathbf{b} - \mathcal{A}x_0$ ,  $p_0 = z_0 = \mathcal{P}r_0$ ,  $q_0 = \mathcal{A}p_0$ ,  $t_0 = q_0$ ;
3 for  $k = 0, 1, \dots$ , to convergence or max-iter do
4      $v_k = \mathcal{P}q_k$  ; // Apply (29) and (30)
5      $\delta_k = z_k^T q_k / q_k^T v_k$  ;
6      $x_{k+1} = x_k + \delta_k p_k$ ;
7      $r_{k+1} = r_k - \delta_k q_k$ ;
8      $z_{k+1} = z_k - \delta_k v_k$ ;
9      $t_{k+1} = \mathcal{A}z_{k+1}$  ;
10     $\eta_{k+1} = z_{k+1}^T t_{k+1} / z_k^T t_k$  ;
11     $p_{k+1} = z_{k+1} + \eta_{k+1} p_k$  ;
12     $q_{k+1} = t_{k+1} + \eta_{k+1} q_k$  ;
13 end

```

7 Experiments

This section evaluates and compares the performances of the algorithms presented in this paper on a series of artificial datasets and on 12 real benchmark classification datasets from the UCI machine learning repository (Bache and Lichman 2013).

7.1 Artificial datasets

A simple two population multivariate normal distribution with equal covariance matrix $\mathcal{N}_d(\mu_i, \Sigma)$ was used to generate a series of datasets with sample sizes varying from 100 to 10,000. The mean vectors was set to $\mu_1 = (0, 0, \dots, 0)$ and $\mu_2 = \mu_1 + 0.2$ and $\Sigma = \text{diag}(1.5, \dots, 1.5)$. The dimension of each dataset was fixed at $d = 50$ and random noise from a zero mean normal distribution with variance 0.03 was added to each dimension to further increase the overlap between the two populations.

The artificial datasets are used in a first experiment to demonstrate the time scalability of LS-KLR, LS-ELR, IRLS-ELR and the performance of the approximate inverse preconditioner when used to train IRLS-KLR or simply pIRLS-KLR.

7.2 Benchmark datasets

The 12 benchmark datasets consists of Pima Indians diabetes (Diabetes), Breast Cancer (Breast), South African Hearth Disease (SAheart) [data from [Hastie et al. \(2001\)](#)], Mammographic Mass (Mam), Sonar, Mines vs. Rocks (Sonar), Car Evaluation (Car), Australian Credit Approval (Credit), Ozone Level Detection (Ozone), Spambase (Spam), Abalone, Wine Quality (Wine), and Waveform Database Generator (Waveform). Each dataset was randomly split into 75 % training and 25 % testing as shown in Table 1.

Most of the datasets represents binary classification problems except for the Car, Abalone and Wine datasets. The Car dataset has four classes representing car evaluations. These were aggregated into two classes: class 1 = (“unacceptable”, “acceptable”) and class 0 = (“good”, “very good”). The Abalone dataset represents the problem of predicting the age of abalone from physical measurements. The age is determined by the number of rings in the Abalone’s cone. The number of rings varied from 1 to 29. The negative class was taken to represent rings in the range 1–10 while the rest made up the positive class. Finally, the outcome variable in the wine dataset is wine quality ranging from 0 (very bad) to 10 (very excellent). The binary classification problem constructed out of this dataset was to distinguish poor or normal wine (0–6) from good or excellent (7–10) wine.

The algorithms compared are grouped into two categories (a) Kernel-Methods comprising SVM, IRLS-KLR and LS-KLR (2) ELM-Methods including ELM, IRLS-ELR and LS-ELR. All computations were performed using the R statistical programming environment ([R Core Team 2012](#)) version 3.0.1 running on a 8 core, 1.4 GHz and

Table 1 Binary classification datasets

Data	Attributes	Sample size	
		Train	Test
Diabetes	8	576	192
Breast	9	207	70
SAheart	9	346	116
Mam	5	622	208
Sonar	60	156	52
Car	6	1,296	432
Credit	14	517	173
Ozone	72	1,385	462
Spam	57	3,447	1,150
Abalone	8	3,132	1,045
Wine	11	3,673	1,225
Waveform	40	3,750	1,250

16 GB RAM machine. The computation of kernel matrices and the pILU of kernels is facilitated by the kernlab add on package (Karatzoglou et al. 2004).

For the experiments, the Kernel-Methods: LS-KLR and IRLS-KLR are trained using the systems (15) and (17) while ELM-Methods: IRLS-ELR and LS-ELR are trained using (25) and (26). All the methods can be directly solved through the simple direct solution given by (29) and (30). Due to the slow convergence or non-convergence of IRLS-KLR on some datasets, the preconditioning approach described in Sect. 6 was used for this algorithm. Results for preconditioning is indicated by the asterisk (*) symbol. The LIBSVM library was used for training SVM and is conveniently implemented in the kernlab package through a function *ksvm*. The function can internally compute the kernel matrix from the data or can accept a kernel matrix from the user. The later option was chosen so that the same method for computing the kernel matrix is used for all the kernel-methods.

7.3 Model selection

The Gaussian kernel function was chosen for the kernel-methods while the Sigmoid function was selected as the ELM hidden-layer map function. The weights and biases (\mathbf{w}_i, b_{0_i}) of the Sigmoid function were uniformly generated from the interval $(-1/\sqrt{d}, 1/\sqrt{d})$ where d is the dimension of the training data. The regularization and kernel parameters (γ, σ) were selected by a tenfold cross-validation using the 75 % training data. In the tenfold cross-validation step, each training data was randomly split into ten non-overlapping parts or folds: onefold is used for training and the rest used as a validation set for parameter selection. Though the number of hidden layer neurons could be conveniently set to a large value such as $p \geq 10^3$ with very little effect on performance as Fig. 2 shows, cross-validation was however used to select an optimal value.

Through a grid search, the parameters with the highest performance on the validation set were selected and used to construct new models using the full training data. The following values were chosen for the parameters in the grid search:

$$\begin{aligned}\gamma &\in \{0.001, 0.01, 0.05, 0.1, 0.5, 1.5, 5, 10, 20, 50, 100, 1,000\}, \\ \sigma &\in \{0.0001, 0.001, 0.01, 0.5, 0.8, 1.01, 1.5, 5.5, 10, 20\}, \\ p &\in \{10, 20, 40, 50, 80, 100, 150, 200, 250, 300, 500, 1,000\}.\end{aligned}$$

Because the ELM-kernel is randomly generated, the output from ELM-methods are non-deterministic, i.e. slightly different results are obtained on each run. Thus in the experiments, the final training and testing was repeated ten times and the results averaged.

7.4 Results

Experimental results showing the accuracy, computational time and optimal parameters are presented in this section.

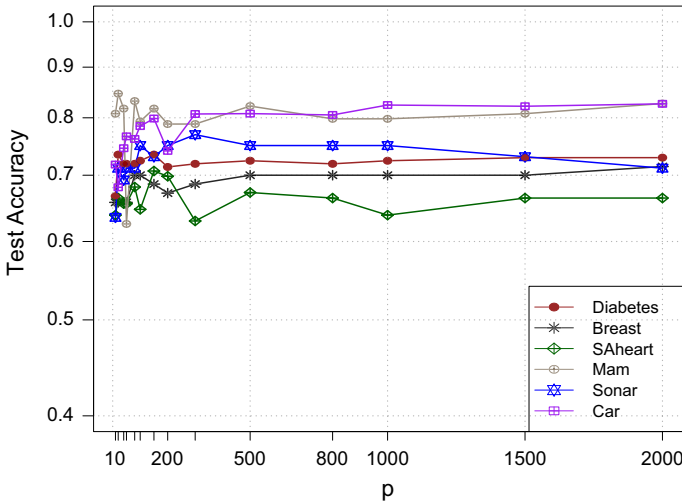


Fig. 2 Test accuracy of LS-ELR with respect to number of hidden-layer neurons (p) for the first six benchmark datasets in Table 1. The optimal regularization parameter γ used in training was taken from Table 3. Good generalization performance can be obtained by setting p large enough say $p \geq 10^3$ (color figure online)

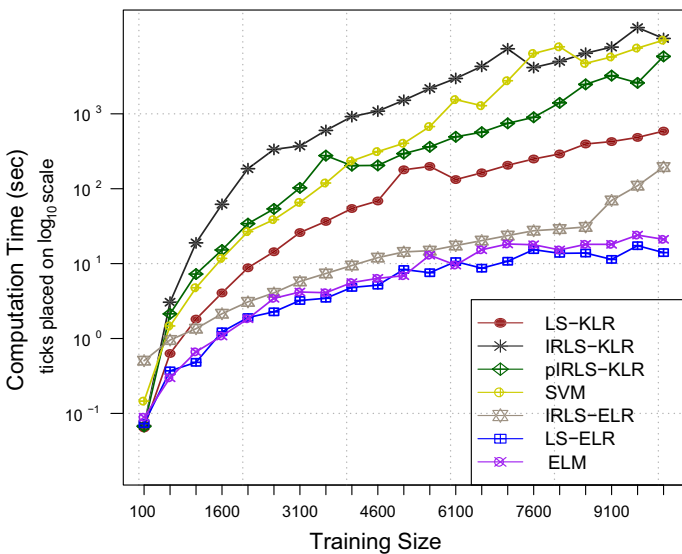


Fig. 3 Time scalability of the algorithms (color figure online)

In a first experiment, the artificial datasets are used to demonstrate time scalability of the algorithms as the sample size is gradually increased. Figure 3 shows the performance of the seven algorithms: LS-KLR, IRLS-KLR, pIRLS-KLR, SVM, IRLS-ELR, LS-ELR, and ELM on the series of artificial datasets. The regularization parameter was arbitrary set to $\gamma = 100$ for all algorithms while the kernel parameter in Kernel-Methods was set to $\sigma = 0.5$ and the number of hidden-layer neurons in ELM-

Methods set to $p = 100$. As Fig. 3 shows, the training time in seconds on the log scale of LS-KLR, pIRLS-KLR, SVM, and IRLS-KLR increases sharply in that order as the training size increases. Clearly, LS-KLR outperforms all the other kernel-methods with IRLS-KLR having the worst performance. On the other hand, the training time for ELM-methods remains very low as the sample size increases. An exception can be found for IRLS-ELR, where its computational time rises sharply for larger sample sizes. A very interesting result illustrated on the figure is the performance of the preconditioner (pIRLS-KLR). The preconditioner significantly reduces the convergence time of IRLS-KLR. Further, for small training sizes, SVM scales slightly better than pIRLS-KLR, however, as the training size increases pIRLS-KLR scales better than SVM.

It should be noted that, the performance of kernel-methods especially SVM with respect to time scalability (and accuracy) depends greatly on the choice and combination of the learning parameters γ and σ . The arbitrary choice of these parameters used in this experiment may not be optimal for some of the algorithms. However, this is one of the goals of this paper: to introduce simple algorithms that are less or non tunable.

The second experiment demonstrates the performance of the algorithms on the 12 benchmark datasets. Results showing validation (VAL) and testing (TEST) accuracy and computation time (in seconds) are shown in Tables 2 and 3. The validation accuracy is the average accuracy over the tenfold cross-validation corresponding to the parameters with the best performance. The computation time is reported only for the final stage of the modeling i.e the time for training and testing using optimal parameters selected at the validation step.

Table 2 shows the performance of the Kernel-Methods: SVM, IRLS-KLR, and LS-KLR while Table 3 shows the performance of the ELM-Methods: ELM, IRLS-ELR, and LS-ELR. For each dataset, the highest test accuracy is boldfaced. If two or more algorithms have the same accuracy, then the accuracy with the smallest time is boldfaced. In general the results show that all ELM-methods have much faster learning speed than kernel-methods.

The following observations can be made from the results presented in the two tables:

1. On average, the performance of LS-KLR is superior to all other methods.
2. LS-KLR is significantly faster than SVM and IRLS-KLR. This can be illustrated more clearly by considering learning speeds on the Abalone, Wine and Waveform datasets:
 - (a) Abalone: LS-KLR runs 3.6 and 22 times faster than SVM and IRLS-KLR respectively
 - (b) Wine: LS-KLR runs 7.1 and 6.3 times faster than SVM and IRLS-KLR respectively.
 - (c) Waveform: LS-KLR runs 2.2 and 5.7 times faster than SVM and IRLS-KLR respectively.

Note that for the Wine and Waveform datasets IRLS-KLR was trained using the preconditioned technique. Without the preconditioner, it took **1,767.21** and **2,027** s to train and test IRLS-KLR on Wine and Waveform datasets respectively. This means without the preconditioner, LS-KLR runs about 28 and 22 times faster than IRLS-KLR on the Wine and Waveform datasets respectively.

Table 2 Performance of kernel-methods

Datasets	SVM				IRLS-KLR				LS-KLR						
	Accuracy		Time	Parameter	Accuracy		Time	Parameter	Accuracy		Time	Parameter			
	TEST	VAL	γ	σ	TEST	VAL	γ	σ	TEST	VAL	γ	σ			
Diabetes	0.74	0.77	3.13	1.5	0.01	0.77	0.78	11.09	50	0.001	0.76	0.78	0.85	20	0.001
Breast	0.73	0.76	0.16	1,000	0.001	0.73	0.78	0.36	5	0.01	0.73	0.78	0.08	50	0.001
SAheart	0.71	0.74	0.50	100	0.001	0.70	0.75	0.86	100	0.001	0.70	0.75	0.13	100	0.001
Mam	0.79	0.85	2.82	20	0.50	0.79	0.85	5.17	1,000	0.001	0.81	0.85	0.27	100	0.01
Sonar	0.83	0.80	0.18	20	0.01	0.85	0.79	0.19	1.5	0.01	0.88	0.79	0.12	5	0.01
Car	0.94	0.89	15.12	1.5	0.80	0.80	0.88	61.83	1,000	0.01	0.94	0.89	7.61	5	0.50
Credit	0.90	0.87	1.57	5	0.01	0.89	0.88	11.91	50	0.01	0.90	0.88	0.83	20	0.01
Ozone	0.91	0.94	16.22	1,000	0.80	0.93	0.94	126.87	20	0.001	0.93	0.94	9.24	5	0.01
Spam	0.86	0.92	138.44	5	0.01	0.88	0.92	854.93	5	0.01	0.93	0.93	78.13	10	0.01
Abalone	0.77	0.77	168.05	5	0.01	0.79	0.78	1,037.36	1,000	0.01	0.80	0.79	47	1,000	0.01
Wine	0.85	0.82	446.01	1.5	1.01	0.80*	0.82*	396.85	100	0.01	0.84	0.83	63.16	0.50	0.80
Waveform	0.85	0.88	200.44	0.5	0.01	0.91*	0.89*	520.85	5	0.01	0.90	0.89	91.23	5	0.01

Table 3 Performance of ELM-methods

Datasets	ELM				IRLS-ELR				LS-ELR						
	Accuracy		Time	Parameter	Accuracy		Time	Parameter	Accuracy		Time	Parameter			
	TEST	VAL		γ	p	TEST	VAL		γ	p	TEST	VAL	γ	p	
Diabetes	0.76	0.79	0.07	0.50	20	0.73	0.79	1.73	1.50	10	0.76	0.79	0.30	0.5	40
Breast	0.79	0.75	0.03	0.10	40	0.79	0.74	0.24	0.01	20	0.77	0.75	0.60	0.1	250
SAheart	0.71	0.75	0.18	0.05	50	0.71	0.74	1.40	50	150	0.73	0.75	0.14	50	10
Mam	0.78	0.86	0.06	0.10	10	0.83	0.86	2.44	100	50	0.80	0.86	0.61	100	50
Sonar	0.77	0.82	0.66	0.05	300	0.77	0.83	1.94	0.10	300	0.79	0.85	0.59	1.5	250
Car	0.89	0.84	4.18	100	300	0.94*	0.85*	31.6	1,000	1,000	0.87	0.79	2.52	100	300
Credit	0.87	0.88	0.83	0.05	250	0.88	0.89	1.36	5	20	0.87	0.89	0.40	1.5	50
Ozone	0.92	0.94	3.77	1.5	300	0.91	0.94	4.04	0.001	250	0.92	0.94	1.47	50	50
Spambase	0.93	0.91	44.10	20	300	0.94	0.93	206.20	50	300	0.91	0.91	13.02	100	300
Abalone	0.78	0.79	19.14	20	250	0.79	0.78	465.04	100	300	0.79	0.79	12.99	20	300
Wine	0.83	0.82	12.12	10	150	0.77*	0.79*	164.74	0.001	40	0.83	0.82	14.34	10	300
Waveform	0.86	0.86	36.72	5	250	0.87	0.86	133.64	100	200	0.84	0.86	9.27	50	200

3. ELM-methods achieve comparable performance as kernel-methods but with much faster learning speed.
4. The gain in speed resulting from the introduction of extreme learning technique to KLR was very substantial. For example
 - (a) LS-ELR runs 4.5 and 27.7 times faster than LS-KLR and IRLS-KLR respectively on the Wine dataset. While the corresponding gain was about 10 and 56 times on the Waveform dataset.
 - (b) IRLS-ELR gained 2.4 times in speed over IRLS-KLR on the wine data and about 4 times for the Waveform dataset. The results would have been respectively about 11 and 15 times if IRLS-KLR had been trained without the preconditioner.
5. LS-ELR has comparable test accuracy with IRLS-ELR while the two methods have a slight advantage over ELM.
6. ELM and LS-ELR have comparable learning speed.

8 Conclusions

This paper proposed new algorithms for learning KLR. First, a simple approximation of the logistic function is used to convert iterative re-weighted least squares KLR into a non-iterative least squares KLR. Second, extreme learning machine method is extended to KLR by replacing the standard kernel with a new kernel constructed using the first layer of a SLFN. And finally, for large and/or poorly conditioned linear systems, an approximate inverse preconditioner is proposed for learning KLR and ELR.

Experiments show that: (1) LS-KLR is as accurate as SVM and IRLS-KLR and may be better while having the distinct advantage of a much faster learning speed. (2) The extension of ELM to KLR significantly reduces computational cost and improves learning speed while maintaining a comparable good generalization performance. (3) The approximate preconditioner can also increase the learning speed of KLR and ELR.

References

- Alcalá-Fdez J, Sánchez L, García S, Del Jesus MJ, Ventura S, Garrell JM, Otero J, Romero C, Bacardit J, Rivas VM et al (2009) Keel: a software tool to assess evolutionary algorithms for data mining problems. *Soft Comput* 13(3):307–318
- Bach FR, Jordan MI (2005) Predictive low-rank decomposition for kernel methods. In: *Proceedings of the 22nd international conference on machine learning*. ACM, pp 33–40
- Bache K, Lichman M (2013) UCI machine learning repository [<http://archive.ics.uci.edu/ml>]. Irvine, CA: University of California, School of Information and Computer Science
- Benzi M (2002) Preconditioning techniques for large linear systems: a survey. *J Comput Phys* 182(2):418–477
- Benzi M, Golub GH, Liesen J (2005) Numerical solution of saddle point problems. *Acta Numer* 14(1):1–137
- Cawley GC, Talbot NLC (2004) Efficient model selection for kernel logistic regression. In: *IEEE pattern recognition, 2004. ICPR 2004. Proceedings of the 17th international conference, vol 2*, pp 439–442
- Cawley GC, Talbot NLC (2008) Efficient approximate leave-one-out cross-validation for kernel logistic regression. *Mach Learn* 71(2–3):243–264
- Chu W, Ong CJ, Keerthi SS (2005) An improved conjugate gradient scheme to the solution of least squares svm. *IEEE Trans Neural Netw* 16(2):498–501

- De Kruif BJ, De Vries TJA (2003) Pruning error minimization in least squares support vector machines. *IEEE Trans Neural Netw* 14(3):696–702
- Fine S, Scheinberg K (2002) Efficient svm training using low-rank kernel representations. *J Mach Learn Res* 2:243–264
- Frénay B, Verleysen M (2010) Using svms with randomised feature spaces: an extreme learning approach. In: *ESANN*
- Gestel T, Suykens J, Lanckriet G, Lambrechts A, Moor B, Vandewalle J (2002) Bayesian framework for least-squares support vector machine classifiers, gaussian processes, and kernel fisher discriminant analysis. *Neural Comput* 14(5):1115–1147
- Hager WW (1989) Updating the inverse of a matrix. *SIAM Rev* 31(2):221–239
- Hastie T, Tibshirani R, Friedman JH (2001) *The elements of statistical learning: data mining, inference, and prediction: with 200 full-color illustrations*. Springer, New York
- Hogben L (2006) *Handbook of linear algebra*. CRC Press, Boca Raton
- Huang G-B, Chen L, Siew C-K (2006a) Universal approximation using incremental constructive feedforward networks with random hidden nodes. *IEEE Trans Neural Netw* 17(4):879–892
- Huang G-B, Zhu Q-Y, Siew C-K (2006b) Extreme learning machine: theory and applications. *Neurocomputing* 70(1):489–501
- Huang G-B, Ding X, Zhou H (2010) Optimization method based extreme learning machine for classification. *Neurocomputing* 74(1):155–163
- Huang G-B, Zhou H, Ding X, Zhang R (2012) Extreme learning machine for regression and multiclass classification. *IEEE Trans Syst Man Cybern Part B Cybern* 42(2):513–529
- Jiao L, Bo L, Wang L (2007) Fast sparse approximation for least squares support vector machine. *IEEE Trans Neural Netw* 18(3):685–697
- Karatzoglou A, Smola A, Hornik K, Zeileis A (2004) kernlab—an S4 package for kernel methods in R. *J Stat Softw* 11(9):1–20. <http://www.jstatsoft.org/v11/i09/>. Accessed 21 Dec 2014
- Katz M, Schaffner M, Andelic E, Krüger S, Wendemuth A (2005) Sparse kernel logistic regression for phoneme classification. In: *Proceedings of 10th international conference on speech and computer (SPECOM)*, Citeseer, vol 2, pp 523–526
- Keerthi SS, Shevade SK (2003) Smo algorithm for least-squares svm formulations. *Neural Comput* 15(2):487–507
- Keerthi SS, Duan KB, Shevade SK, Poo AN (2005) A fast dual algorithm for kernel logistic regression. *Mach Learn* 61(1–3):151–165
- Komarek P (2004) *Logistic regression for data mining and high-dimensional classification*. Robotics Institute, p 222
- Kuh A (2004) Least squares kernel methods and applications. In: *Soft computing in communications*. Springer, Berlin Heidelberg, pp 365–387
- Kulis B, Sustik M, Dhillon I (2006) Learning low-rank kernel matrices. In: *Proceedings of the 23rd international conference on machine learning*. ACM, pp 505–512
- Le Borne S, Ngufor C (2010) An implicit approximate inverse preconditioner for saddle point problems. *Electron Trans Numer Anal* 37:173–188
- Liu Q, He Q, Shi Z (2008) Extreme support vector machine classifier. In: *Advances in knowledge discovery and data mining*. Springer, pp 222–233
- Mercer J (1909) Functions of positive and negative type, and their connection with the theory of integral equations. In: *Philosophical transactions of the Royal Society of London. Series A, containing papers of a mathematical or physical character*, vol 209, pp 415–446
- Ngufor C, Wojtusiak J (2013) Learning from large-scale distributed health data: an approximate logistic regression approach. *ICML 13: role of machine learning in transforming healthcare*
- R Core Team (2012) *R: a language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna. <http://www.R-project.org/ISBN3-900051-07-0>
- Ramani S, Fessler JA (2010) An accelerated iterative reweighted least squares algorithm for compressed sensing mri. In: *2010 IEEE international symposium, IEEE biomedical imaging: from nano to macro*, pp 257–260
- Suykens JAK, Lukas L, Van Dooren P, De Moor B, Vandewalle J (1999) Least squares support vector machine classifiers: a large scale algorithm. In: *European conference on circuit theory and design, ECCTD*, Citeseer, vol 99, pp 839–842
- Suykens JAK, Vandewalle J (1999) Least squares support vector machine classifiers. *Neural Process Lett* 9(3):293–300

- Suykens JAK, Lukas L, Vandewalle J (2000) Sparse approximation using least squares support vector machines. In: The 2000 IEEE international symposium on circuits and systems, 2000. IEEE Proceedings. ISCAS 2000 Geneva, vol 2, pp 757–760
- Suykens JAK, De Brabanter J, Lukas L, Vandewalle J (2002a) Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing* 48(1):85–105
- Suykens JAK, Van Gestel T, De Brabanter J, De Moor B, Vandewalle J, Suykens JAK, Van Gestel T (2002b) Least squares support vector machines, vol 4. World Scientific, Singapore
- Zeng X, Chen X-W (2005) Smo-based pruning methods for sparse least squares support vector machines. *IEEE Trans Neural Netw* 16(6):1541–1546
- Zhu J, Hastie T (2002) Support vector machines, kernel logistic regression and boosting. In: Multiple classifier systems. Springer, pp 16–26
- Zhu J, Hastie T (2005) Kernel logistic regression and the import vector machine. *J Comput Graph Stat* 14(1):185–205