Multi-class kernel logistic regression: a fixed-size implementation

Peter Karsmakers[1,2], Kristiaan Pelckmans[2], Johan A.K. Suykens[2]

***Abstract*— This research studies a practical iterative algorithm for multi-class kernel logistic regression (KLR). Starting from the negative penalized log likelihood criterium we show that the optimization problem in each iteration can be solved by a weighted version of Least Squares Support Vector Machines (LS-SVMs). In this derivation it turns out that the global regularization term is reflected as a usual regularization in each separate step. In the LS-SVM framework, fixed-size LS-SVM is known to perform well on large data sets. We therefore implement this model to solve large scale multi-class KLR problems with estimation in the primal space. To reduce the size of the Hessian, an alternating descent version of Newton's method is used which has the extra advantage that it can be easily used in a distributed computing environment. It is investigated how a multi-class kernel logistic regression model compares to a one-versus-all coding scheme.**

I. Introduction

Logistic regression (LR) and kernel logistic regression (KLR) have already proven their value in the statistical and machine learning community. Opposed to an empirically risk minimization approach such as employed by Support Vector Machines (SVMs), LR and KLR yield probabilistic outcomes based on a maximum likelihood argument. It seen that this framework provides a natural extension to multi-class classification tasks, which must be contrasted to the commonly used coding approach (see e.g. [3] or [1]).

In this paper we use the LS-SVM framework to solve the KLR problem. In our derivation we see that the minimization of the negative penalized log likelihood criterium is equivalent to solving in each iteration a weighted version of least squares support vector machines (wLS-SVMs) [1] [2]. In this derivation it turns out that the global regularization term is reflected as usual in each step. In [12] a similar iterative weighting of wLS-SVMs, with different weighting factors is reported to converge to an SVM solution.

Unlike SVMs, KLR by its nature is not sparse and needs all training samples in its final model. Different adaptations to the original algorithm were proposed to obtain sparseness such as in [3],[4], [5] and [6]. The second one uses a sequential minimization optimization (SMO) approach and in the last case, the binary KLR problem is reformulated into a geometric programming system which can be efficiently solved by an interior-point algorithm. In the LS-SVM framework, fixed-size LS-SVM has shown its value on large data sets. It approximates the feature map using a spectral decomposition, which leads to a sparse representation of the model when estimating in the primal space. We therefor use this technique as a practical implementation of KLR with estimation in the primal space. To reduce the size of the Hessian, an alternating descent version of Newton's method is used which has the extra advantage that it can be easily used in a distributed computing environment. The proposed algorithm is compared to existing algorithms using small size to large scale benchmark data sets.

The paper is organized as follows. In section II we give an introduction to logistic regression. Section III describes the extension to kernel logistic regression. A fixed-size implementation is given in section IV. Section V reports numerical results on several experiments, and finally we conclude in section VI.

The author is with:[1]K.H. Kempen (Associatie KULeuven), IIBT, Kleinhoefstraat 4,B-2440 Geel, Belgium, [2]K.U.Leuven, ESAT-SCD/SISTA, Kasteelpark Arenberg 10, B-3001, Heverlee,Belgium, (email: name.surname@esat.kuleuven.be).

II. Logistic Regression

A. Multi-class logistic regression

After introducing some notations, we recall the principles of multi-class logistic regression. Suppose we have a multi-class problem with C classes ($C \geq 2$) with a training set $\{(x_i, y_i)\}_{i=1}^N \subset \mathbb{R}^d \times \{1, 2, ..., C\}$ with N samples, where input samples x_i are i.i.d. from an unknown probability distribution over the random vectors $(\mathbf{X}, \mathbf{Y})$. We define the first element of x_i to be 1, so that we can incorporate the intercept term in the parameter vector. The goal is to find a classification rule from the training data, such that when given a new input x_*, we can assign a class label to it. In multi-class penalized logistic regression the conditional class probabilities are estimated via logit stochastic models

$$\begin{cases} Pr(\mathbf{Y}=1 \mid \mathbf{X}=x; w) = \frac{\exp(\beta_1^T x)}{1+\sum_{c=1}^{C-1}\exp(\beta_c^T x)} \\ Pr(\mathbf{Y}=2 \mid \mathbf{X}=x; w) = \frac{\exp(\beta_2^T x)}{1+\sum_{c=1}^{C-1}\exp(\beta_c^T x)} \\ \vdots \\ Pr(\mathbf{Y}=C \mid \mathbf{X}=x; w) = \frac{1}{1+\sum_{c=1}^{C-1}\exp(\beta_c^T x)}, \end{cases} \tag{1}$$

where $w = [\beta_1^T; \beta_2^T; ...; \beta_{C-1}^T], w \in \mathbb{R}^{(C-1)d}$ is a collection of the different parameter vectors of m is equal to $C-1$ linear models. The class membership of a new point x_* can be given by the classification rule which is

$$\underset{c \in \{1,2,...,C\}}{\arg\max} \; Pr(\mathbf{Y}=c|\mathbf{X}=x_*; w). \tag{2}$$

The common method to infer the parameters of the different models is via the use of a penalized negative log likelihood (PNLL) criterion.

$$\begin{aligned} &\min_{\beta_1,\beta_2,...,\beta_m} \ell(\beta_1, \beta_2, ..., \beta_m) = \\ &-\ln\left(\prod_{i=1}^N Pr(\mathbf{Y}=y_i|\mathbf{X}=x_i; w)\right) + \frac{\nu}{2}\sum_{c=1}^m \beta_c^T \beta_c. \end{aligned} \tag{3}$$

We derive the objective function for penalized logistic regression by combining (1) with (3) which gives

$$\begin{aligned}\ell_{LR}\,(\beta_1,\beta_2,...,\beta_m) = \\ \sum_{i\in\mathcal{D}_1}\left(-\beta_1^T x_i + \ln(1+e^{\beta_1^T x_i}+e^{\beta_2^T x_i}+...+e^{\beta_m^T x_i})\right)+ \\ \sum_{i\in\mathcal{D}_2}\left(-\beta_2^T x_i + \ln(1+e^{\beta_1^T x_i}+e^{\beta_2^T x_i}+...+e^{\beta_m^T x_i})\right)+ \\ ...+ \\ \sum_{i\in\mathcal{D}_C}\left(-1 + \ln(1+e^{\beta_1^T x_i}+e^{\beta_2^T x_i}+...+e^{\beta_m^T x_i})\right)+ \\ \frac{\nu}{2}\sum_{c=1}^{m}\beta_c^T\beta_c,\end{aligned} \tag{4}$$

where $\mathcal{D}=\{(x_i,y_i)\}_{i=1}^N$, $\mathcal{D}=\mathcal{D}_1\cup\mathcal{D}_2\cup...\cup\mathcal{D}_C$, $\mathcal{D}_i\cap\mathcal{D}_j=\varnothing,\forall i\neq j$ and $y_i=c,x_i\in\mathcal{D}_c$. In the sequel we use the shorthand notation

$$p_{c,i}=Pr(\mathbf{Y}=c|\mathbf{X}=x_i;\Theta). \tag{5}$$

where Θ denotes a parameter vector which will be clear from the context. This PNLL criterion for penalized logistic regression is known to possess a number of useful properties such as the fact that it is convex in the parameters w, smooth and has asymptotic optimality properties.

B. Logistic regression algorithm: iteratively re-weighted least squares

Until now we have defined a model and an objective function which has to be optimized to fit the parameters on the observed data. Most often this optimization is performed by a Newton based strategy where the solution can be found by iterating

$$w^{(k)}=w^{(k-1)}+s^{(k)}, \tag{6}$$

over k until convergence. We define $w^{(k)}$ as the vector of all parameters in the k-th iteration. In each iteration the step $s^{(k)}=-{H^{(k)}}^{-1}g^{(k)}$ can be computed where the gradient and the ij-th element of the Hessian are respectively defined as $g^{(k)}=\frac{\partial\ell_{LR}}{\partial w^{(k)}}$ and $H_{ij}^{(k)}=\frac{\partial^2\ell_{LR}}{\partial w_i^{(k)}\partial w_j^{(k)}}$. The gradient and Hessian can be formulated in matrix notation which gives

$$g^{(k)}=\begin{bmatrix}X^T(u_1^{(k)}-v_1^{(k)})+\nu\beta_1^{(k-1)}\\ \vdots\\ X^T(u_m^{(k)}-v_m^{(k)})+\nu\beta_m^{(k-1)}\end{bmatrix}, \tag{7}$$

$$H^{(k)}=\begin{bmatrix}X^TT_{1,1}^{(k)}X+\nu I & X^TT_{1,2}^{(k)}X & ... & X^TT_{1,m}^{(k)}X\\ & & \ddots & \\ X^TT_{m,1}^{(k)}X & X^TT_{m,2}^{(k)}X & ... & X^TT_{m,m}^{(k)}X+\nu I\end{bmatrix}.$$

where $X\in\mathbb{R}^{N\times d}$ is the input matrix with all values x_i for $i=1,...,N$. Next we define the indicator function $\mathcal{I}(y_i=j)$ which is equal to 1 if y_i is equal to j otherwise it is 0. Some other definitions are $u_c^{(k)}=\left[p_{c,1}^{(k)},...,p_{c,N}^{(k)}\right]^T$, $v_c=[\mathcal{I}(y_1=c),...,\mathcal{I}(y_N=c)]^T$, $t_i^{a,b}=p_{a,i}^{(k)}(1-p_{a,i}^{(k)})$ if a is equal to b otherwise it is $t_i^{a,b}=-p_{a,i}^{(k)}p_{b,i}^{(k)}$ and $T_{a,b}^{(k)}=\mathrm{diag}(\left[t_1^{a,b},...,t_N^{a,b}\right])$. The following matrix notation is convenient to reformulate the Newton sequence as an iteratively regularized re-weighted least squares (IRRLS) problem which will be explained shortly. We define $A^T\in\mathbb{R}^{md\times mN}$ as

$$\left[\begin{array}{ccc|ccc|ccc}x_1&0&0&x_2&0&0&x_N&0&0\\0&x_1&0&0&x_2&0&0&x_N&0\\&\ddots&&&\ddots&&&\ddots&\\0&0&x_1&0&0&x_2&0&0&x_N\end{array}\right] \tag{8}$$

where we define $a_i\in\mathbb{R}^{d\times m}$ as a row of A. Next we define the following vector notations

$$r_i=[\mathcal{I}(y_i=1);...;\mathcal{I}(y_i=m)],r=[r_1;...;r_N],$$
$$P^{(k)}=\left[p_{1,1}^{(k)};\ldots;p_{m,1}^{(k)};\ldots;p_{1,N}^{(k)};\ldots;p_{m,N}^{(k)}\right]\in\mathbb{R}^{mN}. \tag{9}$$

The i-th block of a block diagonal weight matrix $W^{(k)}$ can be written as

$$W_i^{(k)}=\begin{bmatrix}t_i^{1,1}&t_i^{1,2}&...&t_i^{1,m}\\t_i^{2,1}&t_i^{2,2}&&t_i^{2,m}\\&&\ddots&\\t_i^{m,1}&t_i^{m,2}&&t_i^{m,m}\end{bmatrix}. \tag{10}$$

This results in the block diagonal weight matrix

$$W^{(k)}=\mathrm{blockdiag}(W_1^{(k)},...,W_N^{(k)}). \tag{11}$$

Now, we can reformulate the resulting gradient in iteration k as

$$g^{(k)}=A^T(P^{(k)}-r)+\nu w^{(k-1)}. \tag{12}$$

The k-th Hessian is given by

$$H^{(k)}=A^TW^{(k)}A+\nu I. \tag{13}$$

With the closed form solutions of the gradient and Hessian we can setup the second order approximation of the objective function used in Newton's method and use this to reformulate the optimization problem to a weighted least squares equivalent. It turns out that the global regularization term is reflected in each step as a usual regularization term, resulting in a robust algorithm when ν is chosen appropriately. The following lemma summarizes results

Lemma 1 (IRRLS) *Logistic regression can be expressed as an iteratively regularized re-weighted least squares method. The weighted regularized least squares minimization problem is defined as*

$$\begin{aligned}\min_{s^{(k)}}\quad & \tfrac{1}{2}||As^{(k)}-z^{(k)}||^2_{W^{(k)}}+\\ & \tfrac{\nu}{2}(s^{(k)}+w^{(k-1)})^T(s^{(k)}+w^{(k-1)}).\end{aligned}$$

where $z^{(k)}=(W^{(k)})^{-1}(r-P^{(k)})$ and $r,P^{(k)},A,W^{(k)}$ are respectively defined as in (9), (11).

Proof: Newton's method computes in each iteration k the optimal step $s^{(k)opt}$ using the Taylor expansion of the objective function ℓ_{LR}. This results in the following local

objective function

$$s^{(k)opt} = \arg\min_{s^{(k)}} \ \ell_{LR}(w^{(k-1)}) + (A^T(P^{(k)} - r) + \nu w^{(k-1)})^T s^{(k)} + \frac{1}{2} s^{(k)^T}(A^T W^{(k)} A + \nu I) s^{(k)}. \tag{14}$$

By trading terms we can proof that (14) can be expressed as a iteratively regularized re-weighted least squares problem (IRRLS) which can be written as

$$\min_{s^{(k)}} \quad \frac{1}{2}||As^{(k)} - z^{(k)}||^2_{W^{(k)}} + \frac{\nu}{2}(s^{(k)} + w^{(k-1)})^T(s^{(k)} + w^{(k-1)}), \tag{15}$$

where

$$z^{(k)} = (W^{(k)})^{-1}(r - P^{(k)}). \tag{16}$$

■

This classical result is described in e.g. [3].

III. Kernel logistic regression

A. Multi-class kernel logistic regression

In this section the derivation of the kernel version of multi-class logistic regression is given. This result is based on an optimization argument opposed to the use of an appropriate *Representer Theorem* [7]. We show that both steps of the IRRLS algorithm can be easily reformulated in terms of a scheme of iteratively re-weighted LS-SVMs (irLS-SVM). Note that in [3] the relation of KLR to Support Vector Machines (SVM) is stated. The problem statement in Lemma 1 can be advanced with a nonlinear extension to kernel machines where the inputs x are mapped to a high dimensional space. Define $\Phi \in \mathbb{R}^{mN \times md_\varphi}$ as A in (8) where x_i is replaced by $\varphi(x_i)$ and where $\varphi : \mathbb{R}^d \to \mathbb{R}^{d_\varphi}$ denotes the feature map induced by a positive definite kernel. With the application of the Mercer's theorem for the kernel matrix Ω as $\Omega_{ij} = K(a_i, a_j) = \Phi_i^T \Phi_j, i, j = 1, \ldots, mN$ it is not required to compute explicitly the nonlinear mapping $\varphi(\cdot)$ as this is done implicitly through the use of positive kernel functions K. For K there are usually the following choices: $K(a_i, a_j) = a_i^T a_j$ (linear kernel); $K(a_i, a_j) = (a_i^T a_j + h)^b$ (polynomial of degree b, with h a tuning parameter); $K(a_i, a_j) = \exp(-||a_i - a_j||_2^2/\sigma^2)$ (radial basis function, RBF), where σ is a tuning parameter. In the kernel version of LR the m models are defined as

$$\begin{cases} Pr(\mathbf{Y} = 1 \mid \mathbf{X} = x; w) = \frac{\exp(\beta_1^T \varphi(x))}{1 + \sum_{c=1}^m \exp(\beta_c^T \varphi(x))} \\ Pr(\mathbf{Y} = 2 \mid \mathbf{X} = x; w) = \frac{\exp(\beta_2^T \varphi(x))}{1 + \sum_{c=1}^m \exp(\beta_c^T \varphi(x))} \\ \vdots \\ Pr(\mathbf{Y} = C \mid \mathbf{X} = x; w) = \frac{1}{1 + \sum_{c=1}^m \exp(\beta_c^T \varphi(x))} \end{cases}. \tag{17}$$

B. Kernel logistic regression algorithm: iteratively re-weighted least squares support vector machine

Starting from Lemma 1 we include a feature map and introduce the error variable e, this results in

$$\min_{s^{(k)}, e^{(k)}} \frac{1}{2} e^{(k)^T} W^{(k)} e^{(k)} + \frac{\nu}{2}(s^{(k)} + w^{(k-1)})^T(s^{(k)} + w^{(k-1)}) \quad \text{such that } z^{(k)} = \Phi s^{(k)} + e^{(k)}, \tag{18}$$

which in the context of LS-SVMs is called the primal problem. In its dual formulation the solution to this optimization problem can be found by solving a linear system.

Lemma 2 (irLS-SVM) *The solution to the kernel logistic regression problem can be found by iteratively solving the linear system*

$$\left(\frac{1}{\nu}\Omega + W^{(k)^{-1}}\right)\alpha^{(k)} = z^{(k)} + \Omega\alpha^{(k-1)}, \tag{19}$$

where $z^{(k)}$ is defined as in (16). The probabilities of a new point x_* given by m different models can be predicted using (17) where $\beta_c^T \varphi(x_*) = \frac{1}{\nu} \sum_{i=1, i \in \mathcal{D}_c}^N \alpha_{i,c} K(x_i, x_*)$.

Proof: The Lagrangian of the constrained problem as stated in (18) becomes

$$\mathcal{L}(s^{(k)}, e^{(k)}; \alpha^{(k)}) = \frac{1}{2} e^{(k)^T} W^{(k)} e^{(k)} + \frac{\nu}{2}(s^{(k)} + w^{(k-1)})^T(s^{(k)} + w^{(k-1)}) - \alpha^{(k)^T}(\Phi s^{(k)} + e^{(k)} - z^{(k)})$$

with Lagrange multipliers $\alpha^{(k)} \in \mathbb{R}^{Nm}$. The first order conditions for optimality are:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial s^{(k)}} = 0 \to s^{(k)} = \frac{1}{\nu}\Phi^T \alpha^{(k)} - w^{(k-1)} \\ \frac{\partial \mathcal{L}}{\partial e^{(k)}} = 0 \to \alpha^{(k)} = W^{(k)} e^{(k)} \\ \frac{\partial \mathcal{L}}{\partial \alpha^{(k)}} = 0 \to \Phi s^{(k)} + e^{(k)} = z^{(k)}. \end{cases} \tag{20}$$

This results in the following dual solution

$$\left(\frac{1}{\nu}\Omega + W^{(k)^{-1}}\right)\alpha^{(k)} = z^{(k)} + \Omega\alpha^{(k-1)}. \tag{21}$$

Remark that it can be easily shown that the block diagonal weight matrix $W^{(k)}$ is positive definite when the probability of the reference class $p_{C,i} > 0, \forall i = 1, ..., N$. The solution $w^{(L)}$ can be expressed in terms of $\alpha^{(k)}$ computed in the last iteration. This can be seen when combining the formula for $s^{(k)}$ (20) and (6) which gives

$$w^{(L)} = \frac{1}{\nu}\Phi^T \alpha^L. \tag{22}$$

The linear system in (21) can be solved in each iteration by substituting $w^{(k-1)}$ with $\frac{1}{\nu}\Phi^T\alpha^{(k-1)}$. Hence, $Pr(\mathbf{Y} = y_* | \mathbf{X} = x_*; w)$ can be predicted by using (17) where $\beta_c^T \varphi_i(x_*) = \frac{1}{\nu} \sum_{i=1, i \in \mathcal{D}_c}^N \alpha_{i,c} K(x_i, x_*)$. ■

IV. Kernel logistic regression: a fixed-size implementation

A. Nyström approximation

Suppose one takes a finite dimensional feature map (e.g. a linear kernel). Then one can equally well solve the primal as the dual problem. In fact solving the primal problem is more advantageous for larger data sets because the dimension of the unknowns $w \in \mathbb{R}^{md}$ compared to $\alpha \in \mathbb{R}^{mN}$. In order to work in the primal space using a kernel function other than the linear one, it is required to compute an explicit approximation of the nonlinear mapping φ. This leads to a sparse representation of the model when estimating in primal space. Explicit expressions for φ can be obtained by means of an eigenvalue decomposition of the kernel matrix Ω with entries $K(a_i, a_j)$. Given the integral equation $\int K(a, a_j)\phi_i(a)p(a)da = \lambda_i\phi_i(a_j)$, with solutions λ_i and ϕ_i for a variable a with a probability density $p(a)$, we can write

$$\varphi = [\sqrt{\lambda}_1\phi_1, \sqrt{\lambda}_2\phi_2, \ldots, \sqrt{\lambda}_{d_\varphi}\phi_{d_\varphi}]. \tag{23}$$

Given the data set, it is possible to approximate the integral by a sample average. This will lead to the eigenvalue problem (Nyström approximation [9])

$$\frac{1}{mN}\sum_{l=1}^{mN} K(a_l, a_j)u_i(a_l) = \lambda_i^{(s)}u_i(a_j), \tag{24}$$

where the eigenvalues λ_i and eigenfunctions ϕ_i from the continuous problem can be approximated by the sample eigenvalues $\lambda_i^{(s)}$ and the eigenvectors $u_i \in \mathbb{R}^{Nm}$ as

$$\hat{\lambda}_i = \frac{1}{Nm}\lambda_i^{(s)}, \hat{\phi}_i = \sqrt{Nm}u_i. \tag{25}$$

Based on this approximation, it is possible to compute the eigendecomposition of the kernel matrix Ω and use its eigenvalues and eigenvectors to compute the i-th required component of $\hat{\varphi}(a)$ simply by applying (23) if a is a training point, or for any new point a_* by means of

$$\hat{\varphi}(a_*) = \frac{1}{\sqrt{\lambda}_i^{(s)}}\sum_{j=1}^{Nm} u_{ji}K(a_j, a_*). \tag{26}$$

B. Sparseness and large scale problems

Until now the entire training set is of size Nm. Therefore the approximation of φ will yield at most Nm components, each one of which can be computed by (25) for all a, where a is a row of A. However, if we have a large scale problem, it has been motivated [1] to use a subsample of $M \ll Nm$ data points to compute the $\hat{\varphi}$. In this case, up to M components will be computed. External criteria such as entropy maximization can be applied for an optimal selection of the subsample: given a fixed-size M, the aim is to select the support vectors that maximize the quadratic Renyi entropy [10]

$$H_R = -\ln\int p(a)^2 da, \tag{27}$$

which can be approximated by using $\int \hat{p}(a)^2 da = \frac{1}{M^2}1_M^T\Omega 1_M$. The use of this active selection procedure can be important for large scale problems, as it is related to the underlying density distribution of the sample. In this sense, the optimality of this selection is related to the final accuracy of the model. This finite dimensional approximation $\hat{\varphi}(a)$ can be used in the primal problem (18) to estimate w with a sparse representation [1].

C. Method of alternating descent

The dimensions of the approximate feature map $\hat{\varphi}$ can grow large when the number of subsamples M is large. When the number of classes is also large, the size of the Hessian which is proportional to m and d becomes very large and causes the matrix inversion to be computational intractable. To overcome this problem we resort to an alternating descent version of Newton's method [8] where in each iteration the logistic regression objective function is minimized for each parameter β_c separately. The negative log likelihood criterion following this strategy is given by

$$\min_{\beta_c}\ell_{LR}(w_c(\beta_c)) = -\ln\left(\prod_{i=1}^{N} Pr(\mathbf{Y} = y_i|\mathbf{X} = x_i; w_c(\beta_c))\right) + \frac{\nu}{2}\beta_c^T\beta_c, \tag{28}$$

for $c = 1, \ldots, m$. Here we define $w_c(\beta_c) = [\beta_1; ...; \beta_c; ...; \beta_m]$ where only β_c is adjustable in this optimization problem, the other β-vectors are kept constant. This results in a complexity of $O\big(mM^2\big)$ per update of $w^{(k)}$ instead of $O\big(m^2M^2\big)$ for solving the linear system using conjugated gradient [8]. As a disadvantage the convergence rate is worse. Remark that this formulation can be easily embedded in a distributed computing environment because the m different smaller optimization problems can be handled in parallel for each iteration. Before stating the lemma let us define

$$F_c^{(k)} = \text{diag}\left(\left[t_1^{c,c}; t_2^{c,c}; \ldots; t_N^{c,c}\right]\right), \Psi = [\hat{\varphi}(x_1); \ldots; \hat{\varphi}(x_N)], \tag{29}$$
$$E_c^{(k)} = \left[p_{c,1}^{(k)} - \mathcal{I}(y_1 = c); \ldots; p_{c,N}^{(k)} - \mathcal{I}(y_N = c)\right].$$

Lemma 3 (alternating descent IRRLS) *Kernel logistic regression can be expressed in terms of an iterative alternating descent method in which each iteration consists of m reweighted least squares optimization problems*

$$\min_{s_c^{(k)}}\frac{1}{2}||\Psi s_c^{(k)} - z_c^{(k)}||^2_{F_c^{(k)}} + \frac{\nu}{2}(s_c^{(k)} + \beta_c^{(k-1)})^T(s_c^{(k)} + \beta_c^{(k-1)}),$$

where $z_c^{(k)} = -{F_c^{(k)}}^{-1}E_c^{(k)}$ for $c = 1, ..., m$.

Proof: By substituting (17) in the criterion as defined in (28) we obtain the alternating descent KLR objective function. Given fixed $\beta_1, ..., \beta_{c-1}, \beta_{c+1}, ..., \beta_m$ we consider

$$\min_{\beta_c} f(\beta_c, \mathcal{D}_1) + ... + f(\beta_c, \mathcal{D}_C) + \frac{\nu}{2}\beta_c^T\beta_c, \tag{30}$$

for $c = 1, ..., m$.
Where

$$f(\beta_c, \mathcal{D}_j) = \begin{cases} \sum_{i \in \mathcal{D}_j} -\beta_c^T \varphi(x_i) + \ln(1 + e^{\beta_c^T \varphi(x_i)} + \kappa) & c = j \\ \sum_{i \in \mathcal{D}_j} \ln(1 + e^{\beta_c^T \varphi(x_i)} + \kappa) & c \neq j, \end{cases}$$

and κ denotes a constant. Again we use a Newton based strategy to infer the parameter vectors β_c for $c = 1, ..., m$. This results in minimizing m Newton updates per iteration

$$\beta_c^{(k)} = \beta_c^{(k-1)} - s_c^{(k)}, \tag{31}$$

$$s_c^{(k)} = \left(\Psi^T F_c^{(k)} \Psi + \nu I\right)^{-1} \left(\Psi^T E_c^{(k)} + \nu \beta_c^{(k-1)}\right). \tag{32}$$

using an analogous reasoning as in (16), the previous Newton procedure can be reformulated to m IRRLS schemes,

$$\min_{s_c^{(k)}} \frac{1}{2} ||\Psi s_c^{(k)} - z_c^{(k)}||^2_{F_c^{(k)}} + \frac{\nu}{2} (s_c^{(k)} + \beta_c^{(k-1)})^T (s_c^{(k)} + \beta_c^{(k-1)}),$$

where

$$z_c^{(k)} = -{F_c^{(k)}}^{-1} E_c^{(k)}, \tag{33}$$

for $c = 1, \ldots, m$. ■

The resulting alternating descent fixed-size algorithm for KLR is presented in algorithm 1.

Algorithm 1 Alternated descent Fixed-Size KLR

Input: training data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$
Parameters: $w^{(k)}$
Output: probabilities $Pr(\mathbf{X} = x_i | \mathbf{Y} = y_i; w^{opt}), i = 1, ..., N$ and w^{opt} is the converged parameter vector
Initialize: $\beta_c^{(0)} := 0$ for $c = 1, ..., m, k := 0$
Define: $F_c^{(k)}, z_c^{(k)}$ according to resp. (29), (33)
$w^{(0)} = [\beta_1^{(0)}; ...; \beta_m^{(0)}]$
support vector selection according to (27)
compute features Ψ as in (29)
repeat
    $k := k + 1$
    for $c = 1..m$ **do**
        compute $Pr(\mathbf{X} = x_i | \mathbf{Y} = y_i; w^{(k-1)}), i = 1, ..., N$
        construct $F_c^{(k)}, z_c^{(k)}$
        $\min_{s_c^{(k)}} \frac{1}{2} ||\Psi s_c^{(k)} - z_c^{(k)}||^2_{F_c^{(k)}} +$
            $\frac{\nu}{2} (s_c^{(k)} + \beta_c^{(k-1)})^T (s_c^{(k)} + \beta_c^{(k-1)})$
        $\beta_c^{(k)} = \beta_c^{(k-1)} + s_c^{(k)}$
    end for
    $w^{(k)} = [\beta_1^{(k)}; ...; \beta_m^{(k)}]$
until convergence

V. Experiments

All (K)LR experiments in this section are carried out in MATLAB. For the SVM experiments we used LIBSVM [14]. To benchmark the KLR algorithm according to (21) we did some experiments on several small data sets[1] and compared with SVM. For each experiment we used an RBF kernel. The hyperparameters ν and σ were tuned by a 10-fold crossvalidation procedure. For each data set we used the provided realizations. In table 1 it is seen that the error rates of KLR are comparable with those achieved with SVM.

[1]The data sets can be found on the webpage http://ida.first.fraunhofer.de /projects/bench/benchmarks.htm

TABLE I

The table shows the mean and standard deviation of the error rates on different realizations of test and trainingset of different data sets using KLR and SVM with RBF kernel

	KLR	SVM
banana	$\mathbf{10.39 \pm 0.47}$	11.53 ± 0.66
breast-cancer	26.86 ± 0.467	$\mathbf{26.04 \pm 0.66}$
diabetes	$\mathbf{23.18 \pm 1.74}$	23.53 ± 1.73
flare-solar	33.40 ± 1.60	$\mathbf{32.43 \pm 1.82}$
german	23.73 ± 2.15	$\mathbf{23.61 \pm 2.07}$
heart	17.38 ± 3.00	$\mathbf{15.95 \pm 3.26}$
image	3.16 ± 0.52	$\mathbf{2.96 \pm 0.60}$
ringnorm	2.33 ± 0.15	$\mathbf{1.66 \pm 0.12}$
splice	$\mathbf{11.43 \pm 0.70}$	10.88 ± 0.66
thyriod	$\mathbf{4.53 \pm 2.25}$	4.80 ± 2.19
titanic	22.88 ± 1.21	$\mathbf{22.42 \pm 1.02}$
twonorm	$\mathbf{2.39 \pm 0.13}$	2.96 ± 0.23
waveform	$\mathbf{9.68 \pm 0.48}$	9.88 ± 0.43

In Fig. 3 we plot the log likelihoods of test data produced by models inferred with two multi-class versions of LR, a model trained with LDA and a naïve baseline in function of the number of classes. The first multi-class model, which we here will refer to as *LRM*, is as in (1), the second is build using binary subproblems coupled via a one-versus-all encoding scheme [3] which we call *LROneVsAll*. The baseline returns a likelihood which is inverse proportional to the number of classes, independent of the input. For this experiment we used a toy data set which consists of 600 data points in each of the K classes. The data in each class is generated by a mixture of 2 dimensional gaussians. Each time we add a class, ν is tuned using a 10-fold cross validation and the log likelihood averaged over 20 runs is plotted. It can be seen that the KLR multi-class approach results in more accurate likelihood estimates on the test set compared to the alternatives.

To compare the convergence rate of KLR and its alternated descent version we used the same toy data set as before with 6 classes. The resulting curves are plotted in Fig. 1. As expected the convergence rate of the alternated descent algorithm is less than the original formulation of the algorithm. But the cost of each alternated descent iteration is less and therefore gives an acceptable total amount of cpu time. While KLR converges after 18s, alternated descent KLR reaches the stopping criterion after 24s. SVM converges after 13s. The probability landscape of the first out of 6 classes modeled by KLR with RBF kernel is plotted in Fig. 2.

Next we compared the fixed-size KLR implementation with the SMO implementation of LIBSVM on the UCI Adult data set [13]. In this data set one is asked to predict whether an household has an income greater than $50,000$ dollars. It consists of $48,842$ data points and has 14 input variables. Fig. 4 shows the percentage of correctly classified test examples as a function of M, the number of support vectors, together with the CPU time to train the fixed-size KLR model. For SVM we achieved a test set accuracy of

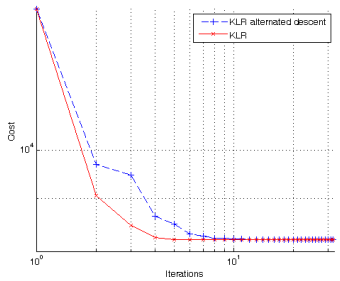
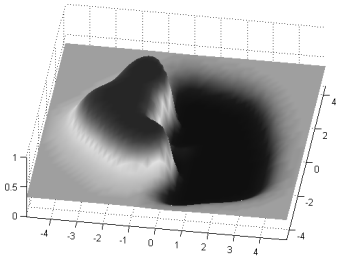


Fig. 1. Convergence plot of multi-class KLR and its alternating descent version.



(a) Class I

Fig. 2. Probability landscape produced by KLR using an RBF kernel on one of the 6 classes from the gaussian mixture data.

85.1% which is comparable with the results shown in Fig. 4.

Finally we used the isolet task [13] which contains 26 spoken English alphabet letters who are characterized by 617 spectral components to compare the multi-class fixed-size KLR algorithm with SVM binary subproblems coupled via a one-versus-one coding scheme. In total the data set contains $6,240$ training examples and $1,560$ test instances. Again we used 10-fold crossvalidation to tune the hyperparameters. With fixed-size KLR and SVM we obtained respectively an accuracy on the test set of 96.41% and 96.86% while the former gives additionally probabilistic outcomes which are useful in the context of speech.

VI. Conclusions

In this paper we presented a fixed-size algorithm to compute a multi-class KLR model which is scalable to large data sets. We showed that the performance in terms of correct classifications is comparable to that of SVM, but with the advantage that KLR gives straightforward probabilistic outcomes which is desirable in several applications. Experiments show the advantage of using a multi-class KLR model compared to the use of a coding scheme.

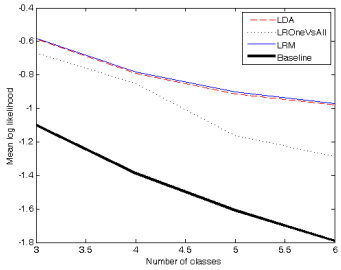


Fig. 3. Mean log likelihood in function of the number of classes in the learning problem.

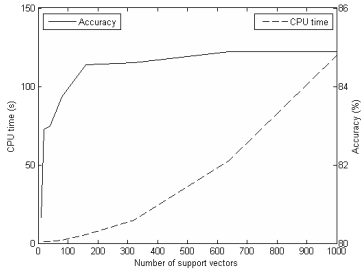


Fig. 4. CPU time and accuracy in function of the number of support vectors when using the fixed-size KLR algorithm.

Acknowledgments. Research supported by GOA AMBioRICS, CoE EF/05/006; (Flemish Government): (FWO): PhD/postdoc grants, projects, G.0407.02, G.0197.02, G.0141.03, G.0491.03, G.0120.03, G.0452.04, G.0499.04, G.0211.05, G.0226.06, G.0321.06, G.0553.06, G.0302.07. (ICCoS, ANMMM, MLDM); (IWT): PhD Grants,GBOU (McKnow), Eureka-Flite2 - Belgian Federal Science Policy Office: IUAP P5/22,PODO-II,- EU: FP5-Quprodis; ERNSI; - Contract Research/agreements: ISMC/IPCOS, Data4s, TML, Elia, LMS, Mastercard. JS is a professor and BDM is a full professor at K.U.Leuven Belgium. This publication only reflects the authors' views.

References

[1] J.A.K. Suykens, T. Van Gestel, J. De Brabanter, B. De Moor and J. Vandewalle, *Least Squares Support Vector Machines*, World Scientific, Singapore, 2002.

[2] J.A.K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers", *Neural Processing Letters*,9(3):293-300, 1999.

[3] J. Zhu, T. Hastie, "Kernel logistic regression and the import vector machine", *Advances in Neural Information Processing Systems*, vol. 14, 2001.

[4] S.S. Keerthi, K. Duan, S.K. Shevade and A.N. Poo "A Fast Dual Algorithm for Kernel Logisic Regression ", *International Conference on Machine Learning*, 2002.

[5] J. Zhu and T. Hastie, "Classification of gene microarrays by penalized logistic regression", *Biostatistics*, vol. 5, pp. 427444, 2004.

[6] K. Koh, S.-J. Kim and S. Boyd "An Interior-Point Method for Large-Scale l_1-Regularized Logistic Regression", *Internal report*, july, 2006.

[7] G. Kimeldorf, G. Wahba, "Some results on Tchebycheffian spline functions", *Journal of Mathematics Analysis and Applications*,vol. 33, pp. 82-95, 1971.

[8] J. Nocedal, S. J. Wright, *Numerical Optimization*, Springer, 1999.

[9] C.K.I Williams, M. Seeger "Using the Nyström Method to Speed Up Kernel Machines", *Proceedings Neural Information Processing Systems*, vol 13., MIT press, 2000.

[10] M. Girolami "Orthogonal Series Density Estimation and the Kernel Eigenvalue Problem", *Neural Computation*, vol. 14(3), 669-688, 2003.

[11] J.A.K. Suykens, J. De Brabanter, L. Lukas, J. Vandewalle, "Weighted least squares support vector machines : robustness and sparse approximation", *Neurocomputing*, vol. 48, no. 1-4, pp. 85-105, 2002.

[12] F. Pérez-Cruz and C. Bousoño-Calzón and A. Artés-Rodríguez, "Convergence of the IRWLS Procedure to the Support Vector Machine Solution", *Neural Computation*, vol. 17, p. 7-18, 2005.

[13] C.J. Merz, P.M. Murphy, "UCI repository of machine learning databases", *http://www.ics.uci.edu/ mlearn/MLRepository.html*, 1998.

[14] C.C. Chang, C.J. Lin, "LIBSVM : a library for support vector machines", *Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm*, 2001.