BULLETIN OF THE POLISH ACADEMY OF SCIENCES
TECHNICAL SCIENCES, Vol. 58, No. 1, 2010
DOI: 10.2478/v10175-010-0018-2

Iteratively reweighted least squares classifier and its ℓ_2- and ℓ_1-regularized Kernel versions

J.M. ŁĘSKI*

Institute of Electronics, Silesian University of Technology, 16 Akademicka St., 44-100 Gliwice, Poland

Abstract. This paper introduces a new classifier design method based on regularized iteratively reweighted least squares criterion function. The proposed method uses various approximations of misclassification error, including: linear, sigmoidal, Huber and logarithmic. Using the represented theorem a kernel version of classifier design method is introduced. The conjugate gradient algorithm is used to minimize the proposed criterion function. Furthermore, ℓ_1-regularized kernel version of the classifier is introduced. In this case, the gradient projection is used to optimize the criterion function. Finally, an extensive experimental analysis on 14 benchmark datasets is given to demonstrate the validity of the introduced methods.

Key words: classifier design, IRLS, conjugate gradient optimization, gradient projection, Kernel matrix.

1. Introduction

The classifier design is a fundamental problem in a pattern recognition. Up to now, a lot of classifier design methods have been proposed. An overview of these methods can be found in [1–5]. Among classifier design methods, Support Vector Machine (SVM) is the most successful one [6]. It maps the input data into high- (even infinite-) dimensional feature space using the so-called kernel function and then it founds a linear separating hyperplane with maximal margin between classes. The learning process of SVM is formulated as a solution of a constrained quadratic optimization problem [7, 8]. The main disadvantage of this state-of-the-art technique is its high computational cost. To overcome this drawback a lot of alternative methods have been proposed, including: Lagrangian support vector machine [9], least squares support vector machine [10] and core vector machine [11–13].

For the last few years, there has been an increasing interest to incorporate the main result of the statistical learning theory, i.e. the generalization ability of a machine depends both on the empirical risk, on a training set and complexity of this machine [7, 8], to the traditional pattern recognition methods, c.f.: kernel Fisher discriminant [14], kernel Foley-Sammon discriminant [15], kernel perceptron [16], kernel fuzzy perceptron [17], kernel dissimilarity-based classifier [18].

In 1965 Y. Ho and R. Kashyap proposed an iterative method to solve a system of linear inequalities and its application to pattern classification problems [19, 20]. Among the traditional methods of classifier design the Ho-Kashyap algorithm is the most powerful one. On-line version of this algorithm has been introduced by M. Hassoun and J. Song [21]. The Ho-Kashyap classifier with generalization control [22] and its kernel version [23] have been also proposed. Furthermore, the above methods have been extended to a matrix pattern [24] and a multiple kernel learning algorithm [25]. However, the most important disadvantage of the Ho-Kashyap algorithm is its high computational effort. The algorithm needs $(N+1)\times(N+1)$ matrix inversion, where N denotes dimensionality of data from the training set.

In contrast to the Minimum Squared Error (MSE) method of a classifier design [1, 2, 5], the Ho-Kashyap procedure focuses on the misclassified data. In each iteration, elements of the so-called target (or margin) vector corresponding to properly classified data are increased. Thus, squared errors corresponding to these data are decreased or even zeroed, so the algorithm in the next iteration pays attention to misclassified data. The above may be viewed as the relaxation procedure realized in "batch" mode. A "force" to the separating hyperplane created by each properly classified datum is relaxed. Another main disadvantage of the Ho-Kashyap procedure is the use of the quadratic loss function that leads to an approximation of the misclassification error which is not the best in this case. The absolute approximation of the misclassification error in the Ho-Kashyap method using Iteratively Reweighted Least Squares (IRLS) procedure has been introduced in [22].

The main goal of this work is to show that the IRLS procedure can be used for a better approximation of misclassification error than the squared one as well as may be used for relaxation. So, the iterative modification of target vector is not needed. The conjugate gradient algorithm is used for minimization of proposed criterion function. Furthermore, ℓ_2- and ℓ_1-regularized kernel version of the classifier is introduced. For ℓ_1-regularized criterion function the gradient projection is used to optimization. The next goal is to investigate the generalization ability of the proposed classifier design methods for synthetic and real-world benchmark data.

This paper focuses on a two-class (binary) classifiers. The proposed method can be easily generalized to a multi-class

*e-mail: jleski@polsl.pl

problem using the "one-against-all" (class-remainder) and the "one-against-one" (class-class) methodology [4, 26].

The remainder of this paper is organized as follows: Section 2 presents the introduction of an iteratively reweighted least square criterion function to design a binary classifier. Section 3 shows that this approach may be extended to a nonlinear case using the representer theorem. Section 4 shows that ℓ_1-regularized kernel version of the classifier may be presented as an bound-constrained quadratic program. Section 5 presents simulation results and a discussion for the classification of real-world and synthetic benchmark datasets. Finally, conclusions are drawn in Sec. 6.

2. Classifier design method – linear case

The binary classifier is designed on the basis of a training set, $\mathcal{T_R}^{(N)} = \{(\mathbf{x}_1, \theta_1), (\mathbf{x}_2, \theta_2), \cdots, (\mathbf{x}_N, \theta_N)\}$, where N is data cardinality, and each independent datum (pattern) $\mathbf{x}_i \in \mathbb{R}^t$ has a corresponding dependent datum $\theta_i \in \{+1, -1\}$, which indicates its assignment to one of two classes, ω_1 or ω_2: $\theta_i = +1$ iff $\mathbf{x}_i \in \omega_1$ and $\theta_i = -1$ iff $\mathbf{x}_i \in \omega_2$. Defining the augmented pattern vector $\mathbf{x}'_i = \left[\mathbf{x}_i^\top, 1\right]^\top$, we seek a weight vector $\mathbf{w} = \left[\widetilde{\mathbf{w}}^\top, w_0\right]^\top \in \mathbb{R}^{t+1}$, such that

$$d(\mathbf{x}_i) \triangleq \mathbf{w}^\top \mathbf{x}'_i = \widetilde{\mathbf{w}}^\top \mathbf{x}_i + w_0 \begin{cases} \geq 0, & \mathbf{x}_i \in \omega_1, \\ < 0, & \mathbf{x}_i \in \omega_2, \end{cases} \tag{1}$$

where $d(\mathbf{x}_i)$ is called a linear discrimination function.

If we multiply by -1 all patterns of the training set that are members of ω_2 class, then (1) can be rewritten in the form $\theta_i \mathbf{w}^\top \mathbf{x}'_i \geq 0$, for $i = 1, 2, \cdots, N$. If the above conditions are satisfied for all members of the training set, then the data are said to be linearly separable. For overlapping classes it is impossible to find the weight vector $\mathbf{w}$, such that the conditions are satisfied for all data from the training set.

Even in linearly separable case, some data may lie near the separating hyperplane $\mathbf{w}^\top \mathbf{x}' = 0$. Thus, according to the statistical learning theory the safer approach is to seek the vector $\mathbf{w}$, such that

$$\theta_i \mathbf{w}^\top \mathbf{x}'_i \geq \varepsilon; \qquad \varepsilon > 0. \tag{2}$$

Now consider the data for which the equality hold in (2). For ω_1 these data lie on the hyperplane $H_1: \ \mathbf{w}^\top \mathbf{x}' = \varepsilon$, with normal $\widetilde{\mathbf{w}}$ and perpendicular distance from the space origin $|\varepsilon - w_0|/\|\widetilde{\mathbf{w}}\|$. Similarly, for ω_2 these data lie on the hyperplane $H_2: \mathbf{w}^\top \mathbf{x}' = -\varepsilon$, with normal $\widetilde{\mathbf{w}}$ and perpendicular distance from the origin $|-\varepsilon - w_0|/\|\widetilde{\mathbf{w}}\|$. From above we see that the margin of separation between ω_1 and ω_2, i.e. the distance between H_1 and H_2 is $\mathcal{M}(\mathbf{w}, \varepsilon) = 2\varepsilon/\|\widetilde{\mathbf{w}}\|$. The region between H_1 and H_2 is usually called the region of separation. Note that the margin of separation is independent to rescaling (2). If we divide both sides of (2) by ε, then a new weight vector equals $\mathbf{w}^* = \mathbf{w}/\varepsilon$ and the margin of separation is $\mathcal{M}(\mathbf{w}^*, 1) = 2/\left\|\widetilde{\mathbf{w}}^*\right\| = 2\varepsilon/\|\widetilde{\mathbf{w}}\|$. This states that maximizing the margin of separation between classes is equivalent to use $\varepsilon = 1$ and minimizing the Euclidean norm of $\widetilde{\mathbf{w}}$. So, for the linearly separable case we seek $\mathbf{w}$ such that $\theta_i \mathbf{w}^\top \mathbf{x}'_i \geq 1$ for $i = 1, 2, \cdots, N$ and minimizing $\widetilde{\mathbf{w}}^\top \widetilde{\mathbf{w}}$. Let $\mathbf{X}$ be the $N \times (t+1)$ matrix

$$\mathbf{X}^\top \triangleq [\theta_1 \mathbf{x}'_1, \theta_2 \mathbf{x}'_2, \cdots, \theta_N \mathbf{x}'_N]. \tag{3}$$

Then the above inequalities can be rewritten in the matrix form $\mathbf{Xw} \succeq \mathbf{1}$, where $\mathbf{1}$ denotes the vector with all entries equal to 1 and the symbol $\succeq$ stands for componentwise inequality. We define the error vector as $\mathbf{e} = \mathbf{Xw} - \mathbf{1}$. If the pth component of $\mathbf{e}$ is $e_p > 0$, then the pth pattern is on the right side of the separation hyperplane and outside the region of separation. Otherwise, if $e_p \in [-1, 0]$, then the pth pattern is on the right side of the separation hyperplane but inside the region of separation. Moreover, if $e_p < -1$, then the respective pattern is on the wrong side of the separation hyperplane. Thus, the misclassification error on the training set can be written as

$$\sum_{i=1}^{N} \mathcal{S}(-e_i - 1) = \sum_{i=1}^{N} \mathcal{S}\left(-\theta_i \mathbf{w}^\top \mathbf{x}'_i\right), \tag{4}$$

where $\mathcal{S}(\cdot)$ denotes the unit step function, $\mathcal{S}(\zeta) = 1$ for $\zeta > 0$, and zero otherwise. Due to nonconvexity of (4) its minimization is NP-complete problem. There are many approximations of (4) to make this minimization problem mathematically tractable. In minimum squared error procedure of a classifier design the minimized criterion function takes the form [1, 5]

$$J(\mathbf{w}) = \sum_{i=1}^{N} \left(\theta_i \mathbf{w}^\top \mathbf{x}'_i - b_i\right)^2, \tag{5}$$

where b_is form the so-called target or margin vector. Its elements are chosen arbitrarily ($b_i > 0$) – usually all elements are set to one. Above approximation is symmetric. So, its pay attention on misclassified and well classified data. In the Ho-Kashyap modification elements of vector $\mathbf{b}$ are iteratively updated. b_is corresponding to properly classified data are increased (errors are decreased), so impact of these data to the separation hyperplane is also decreased. In support vector machine the linear soft margin is introduced. No penalty occurs for pattern on the right side of the separation hyperplane and outside the region of separation. If pattern lie on the region of separation, even on right side of the separation hyperplane, then penalty is increased linearly to the perpendicular distance from the edge of this region [6]

$$J(\mathbf{w}) = \sum_{i=1}^{N} \max\left(0, 1 - \theta_i \mathbf{w}^\top \mathbf{x}'_i\right). \tag{6}$$

In proposed method of classifier design a various approximation of misclassification error and idea of the soft margin are used. We seek vector $\mathbf{w}$ by the following minimization

$$\min_{\mathbf{w} \in \mathbb{R}^{t+1}} J(\mathbf{w}) \triangleq \sum_{i=1}^{N} \frac{h_i}{2} \mathcal{L}\left(\theta_i \mathbf{w}^\top \mathbf{x}'_i - 1\right) + \frac{\tau}{2} \widetilde{\mathbf{w}}^\top \widetilde{\mathbf{w}}, \tag{7}$$

where $\mathcal{L}(\cdot)$ stands for a loss function used to approximation of misclassification error, h_i is a weight corresponding to the ith pattern (its role is explained later). The second term is related

to the maximization of the margin of separation (minimization complexity of classifier). Parameter $\tau > 0$ controls the trade-off between the complexity of classifier and the amount up to which errors are tolerated.

If we choose the quadratic loss function then in matrix notation (7) takes the form

$$\min_{\mathbf{w}\in\mathbb{R}^{t+1}} J(\mathbf{w}) \triangleq \frac{1}{2}(\mathbf{X}\mathbf{w}-\mathbf{1})^{\top}\mathbf{H}(\mathbf{X}\mathbf{w}-\mathbf{1}) + \frac{\tau}{2}\widetilde{\mathbf{w}}^{\top}\widetilde{\mathbf{w}}, \quad (8)$$

where the matrix $\mathbf{H} = \operatorname{diag}(h_1, h_2, \cdots, h_N)$. Indeed, this loss function is symmetric – the patterns equally distant from the edge of the region of separation, independently they lie on the right or on the wrong side has the same penalty. To obtain asymmetric loss function the weights h_i may be used. If we set $h_i = 0$ for $e_i \geq 0$ and 1 otherwise, then only patterns lie on the region of separation are penalized. However, to check the sign of e_i we must known vector $\mathbf{w}$. Thus, criterion function (8) should be minimized by iteratively reweighting. Let us denote $\mathbf{w}$, $\mathbf{H}$ and $\mathbf{e}$ in kth iteration as $\mathbf{w}^{(k)}$, $\mathbf{H}^{(k)}$ and $\mathbf{e}^{(k)}$, respectively. Criterion function (8) for kth iteration takes the form

$$\min_{\mathbf{w}^{(k)}\in\mathbb{R}^{t+1}} J^{(k)}\left(\mathbf{w}^{(k)}\right) \triangleq \frac{1}{2}\left(\mathbf{X}\mathbf{w}^{(k)}-\mathbf{1}\right)^{\top}\mathbf{H}^{(k)} \left(\mathbf{X}\mathbf{w}^{(k)}-\mathbf{1}\right) + \frac{\tau}{2}\left(\widetilde{\mathbf{w}}^{(k)}\right)^{\top}\widetilde{\mathbf{w}}^{(k)}, \quad (9)$$

where

$$h_i^{(k)} = \begin{cases} 0, & e_i^{(k-1)} \geq 0, \\ 1, & e_i^{(k-1)} < 0, \end{cases} \quad (10)$$

$$\mathbf{e}^{(k-1)} = \mathbf{X}\mathbf{w}^{(k-1)} - \mathbf{1}. \quad (11)$$

Thus, to minimize the criterion function for the kth iteration the weights are obtained using (10), (11) and the result of optimization of the criterion function for the previous iteration. To start this sequential optimizations we set weights in the 0th iteration as $h_i = 1$ for all i. Above minimization may be viewed as Iteratively Reweighted Least Square (IRLS) method with control the complexity of a solution. The loss function obtained in (9), (10) can be called asymmetric quadratic or Asymmetric SQuaRe (ASQR) function. Classifier design methods need to be robust. It is well-known from literature [27], that the squared error loss function does not lead to robustness to noised data and outliers. Also, it is not a good approximation of misclassification error (4). A better is to use the absolute error function. This loss function is easy to obtain by taking

$$h_i^{(k)} = \begin{cases} 0, & e_i^{(k-1)} \geq 0, \\ 1 \Big/ \left|e_i^{(k-1)}\right|, & e_i^{(k-1)} < 0. \end{cases} \quad (12)$$

In this case, reweighting is used for both asymmetrization (relaxation) and changing the loss function. The loss function obtained in (9), (12) can be called asymmetric absolute or Asymmetric LINear (ALIN) function. Many other loss functions may be easily obtained:

- Asymmetric HUBer (AHUB)

$$h_i^{(k)} = \begin{cases} 0, & e_i^{(k-1)} \geq 0, \\ 1, & -1 \leq e_i^{(k-1)} < 0, \\ 1 \Big/ \left|e_i^{(k-1)}\right|, & e_i^{(k-1)} < -1. \end{cases} \quad (13)$$

- SIGmoidal (SIG)

$$h_i^{(k)} = 1 \Big/ \left(\left(e_i^{(k-1)}\right)^2 \left(1 + \exp\left(\alpha\left(e_i^{(k-1)} + 1\right)\right)\right)\right). \quad (14)$$

- Asymmetric SIGmoidal-Linear (ASIGL)

$$h_i^{(k)} = 1 \Big/ \left(\left|e_i^{(k-1)}\right| \left(1 + \exp\left(\alpha\left(e_i^{(k-1)} + 1\right)\right)\right)\right). \quad (15)$$

- Asymmetric LOGarithmic (ALOG)

$$h_i^{(k)} = \begin{cases} 0, & e_i^{(k-1)} \geq 0, \\ \log\left(1 + \left(e_i^{(k-1)}\right)^2\right) \Big/ \left(e_i^{(k-1)}\right)^2, & e_i^{(k-1)} < 0. \end{cases} \quad (16)$$

- Asymmetric LOG-Linear (ALOGL)

$$h_i^{(k)} = \begin{cases} 0, & e_i^{(k-1)} \geq 0, \\ \log\left(1 + \left(e_i^{(k-1)}\right)^2\right) \Big/ \left|e_i^{(k-1)}\right|, & e_i^{(k-1)} < 0. \end{cases} \quad (17)$$

The condition for optimality of (9) for the kth iteration is obtained by its differentiating with respect to $\mathbf{w}$ and setting the result equals to zero

$$\mathbf{w}^{(k)} = \left(\mathbf{X}^{\top}\mathbf{H}^{(k)}\mathbf{X} + \tau\widetilde{\mathbf{I}}\right)^{-1}\mathbf{X}^{\top}\mathbf{H}^{(k)}\mathbf{1}, \quad (18)$$

where $\widetilde{\mathbf{I}}$ is the identity matrix with the last element on the main diagonal set to zero.

The iteratively reweighted least square error minimization procedure for classifier design can be summarized in the following steps:

1. Fix $\tau > 0$ and $\mathbf{H}^{(0)} = \mathbf{I}$. Set the iteration index $k = 0$.
2. $\mathbf{w}^{(k)} = \left(\mathbf{X}^{\top}\mathbf{H}^{(k)}\mathbf{X} + \tau\widetilde{\mathbf{I}}\right)^{-1}\mathbf{X}^{\top}\mathbf{H}^{(k)}\mathbf{1}$.
3. $\mathbf{e}^{(k)} = \mathbf{X}\mathbf{w}^{(k)} - \mathbf{1}$.
4. $\mathbf{H}^{(k+1)} = \operatorname{diag}\left(h_1^{(k+1)}, h_2^{(k+1)}, \cdots, h_N^{(k+1)}\right)$, where $h_i^{(k+1)} = f\left(e_i^{(k)}\right)$, for $i = 1, 2, \cdots, N$, and $f(\cdot)$ stands for selected loss function (10), (12)–(17).
5. if $k > 1$ and $\left\|\mathbf{w}^{(k)} - \mathbf{w}^{(k-1)}\right\| < \xi$, then stop else $k \leftarrow k + 1$, go to (2).

Remarks. The iterations were stopped as soon as the Euclidean norm in a successive pair of $\mathbf{w}$ vectors is less than ξ. The quantity ξ is a pre-set small positive value. In all experiments $\xi = 10^{-3}$ is used. The above algorithm requires the inversion of an $(t+1)\times(t+1)$ matrix that lead to a running time of $\mathcal{O}\left((t+1)^3\right)$, where t stands for the dimensionality of input data. So, this algorithm is computationally infeasible for large data dimensionality.

In above algorithm, to solve unconstrained quadratic optimization problem (9) the well-known conjugate gradient approach can be used. In contrast to solution (18) this algorithm produce a minimizing sequence $\mathbf{w}^{(k),[n]}$, where $n = 0, 1, \cdots$. For the sake of simplicity superscript (k) will be omitted

$$\mathbf{w}^{[n+1]} = \mathbf{w}^{[n]} + \nu^{[n]}\mathbf{d}^{[n]}, \tag{19}$$

where $\nu^{[n]}$ denotes the step size, $\mathbf{d}^{[n]}$ stands for the search direction, both for the nth iteration. Let us assume that the search direction is known, then the step size is chosen to minimize $J\left(\mathbf{w}^{[n+1]}\right) = J\left(\mathbf{w}^{[n]} + \nu^{[n]}\mathbf{d}^{[n]}\right)$. Differentiating above with respect to $\nu^{[n]}$ and setting the result equals to zero, we have

$$\nu^{[n]} = -\frac{\tau\left(\mathbf{d}^{[n]}\right)^{\top}\widetilde{\mathbf{I}}\mathbf{w}^{[n]} + \left(\mathbf{d}^{[n]}\right)^{\top}\mathbf{X}^{\top}\mathbf{H}\,\mathbf{e}^{[n]}}{\left(\mathbf{d}^{[n]}\right)^{\top}\mathbf{G}\mathbf{d}^{[n]}}, \tag{20}$$

where $\mathbf{G} = \mathbf{X}^{\top}\mathbf{H}\,\mathbf{X} + \tau\widetilde{\mathbf{I}}$ and $\mathbf{e}^{[n]} = \mathbf{X}\mathbf{w}^{[n]} - \mathbf{1}$.

After some simple algebra criterion function (9) may be presented as

$$J(\mathbf{w}) = \frac{1}{2}\mathbf{w}^{\top}\mathbf{G}\mathbf{w} - \mathbf{b}_1^{\top} + c, \tag{21}$$

where $\mathbf{b}_1 = \mathbf{X}^{\top}\mathbf{H}\mathbf{1}$ and $c = \frac{1}{2}\mathbf{1}^{\top}\mathbf{H}\mathbf{1}$. In the conjugate gradient method the current search direction should be $\mathbf{G}$-conjugate to the previously chosen directions, i.e. $\left(\mathbf{d}^{[n_1]}\right)^{\top}\mathbf{G}\mathbf{d}^{[n_2]} = 0$ for all $n_1 \neq n_2$. A new search direction is obtained as a combination of the previous one and the current gradient vector $\mathbf{g}^{[n]}$

$$\mathbf{d}^{[n]} = \mathbf{g}^{[n]} + \beta^{[n]}\mathbf{d}^{[n-1]}, \tag{22}$$

where $\beta^{[n]}$ is chosen to obtain $\mathbf{G}$-conjugacy with the previous direction. Thus, $\left(\mathbf{d}^{[n-1]}\right)^{\top}\mathbf{G}\left(\mathbf{g}^{[n]} + \beta^{[n]}\mathbf{d}^{[n-1]}\right) = 0$. After some simple algebra, we have

$$\beta^{[n]} = -\frac{\left(\mathbf{d}^{[n-1]}\right)^{\top}\mathbf{G}\mathbf{g}^{[n]}}{\left(\mathbf{d}^{[n-1]}\right)^{\top}\mathbf{G}\mathbf{d}^{[n-1]}}. \tag{23}$$

The gradient vector is obtained using (9)

$$\mathbf{g}^{[n]} = \left.\frac{\partial J(\mathbf{w})}{\partial \mathbf{w}}\right|_{\mathbf{w}=\mathbf{w}^{[n]}} = \tau\widetilde{\mathbf{I}}\mathbf{w}^{[n]} + \mathbf{X}^{\top}\mathbf{H}\left(\mathbf{X}\mathbf{w}^{[n]} - \mathbf{1}\right). \tag{24}$$

Comparing (20) and (24) a simpler form of the step size is obtained

$$\nu^{[n]} = -\frac{\left(\mathbf{d}^{[n]}\right)^{\top}\mathbf{g}^{[n]}}{\left(\mathbf{d}^{[n]}\right)^{\top}\mathbf{G}\mathbf{d}^{[n]}}. \tag{25}$$

The minimization of (9) using the conjugate gradient method may be summarized in the following steps

1. Set the iteration index $n = 0$ and $\mathbf{w}^{[0]} = \mathbf{0}$.
2. Calculate gradient vector $\mathbf{g}^{[n]}$ using (24).
3. if $n = 0$, then $\beta^{[0]} = 0$, else calculate $\beta^{[n]}$ using (23).
4. Calculate search direction $\mathbf{d}^{[n]}$ using (22).
5. Calculate step size $\nu^{[n]}$ using (25).
6. Update $\mathbf{w}$ using (19).
7. if $\left\|\mathbf{w}^{[n+1]} - \mathbf{w}^{[n]}\right\| < \zeta$, then stop, else $n \leftarrow n + 1$, go to (2).

Remarks. The iterations were stopped as soon as the Euclidean norm in a successive pair of $\mathbf{w}$ vectors is less than ζ, where ζ is a pre-set small positive value. In all experiments $\zeta = 10^{-3}$ is used. The conjugate gradient algorithm converges theoretically in $p + 1$ steps, where p is rank of matrix $\mathbf{G}$. If $\mathbf{G}$ is full rank then algorithm converges in $t + 1$ steps, where t denotes the dimensionality of input data. This algorithm replaces step (2) in the previous one. The quantity $\left(\mathbf{d}^{[n]}\right)^{\top}\mathbf{G}\mathbf{d}^{[n]}$ calculated in step (5) may be saved and used in step (3) of the next iteration.

3. Classifier design method – nonlinear case

Among nonlinear classifiers the so-called kernel-based based methods are of special interest in last years [6]. The methods may be motivated by the Cover's theorem on separability of patterns. This theorem states that nonlinearly separated patterns in the input space can be linearly separable in a new space obtained by nonlinear mapping of the original one, if the dimensionality of new space is high enough [28]. So, we seek a function that leads to small classification error and has small norm in Reproducing Kernel Hilbert Space (RHKS) $\mathcal{H}$:

$$\min_{d\in\mathcal{H}} J(d) \triangleq \sum_{i=1}^{N}\mathcal{L}\left(\theta_i d(\mathbf{x}_i) - 1\right) + \frac{\tau}{2}\|d\|_K^2, \tag{26}$$

where $\mathcal{L}$ denotes loss function used to approximation of misclassification error and $\|d\|_K^2$ is the norm in reproducing kernel Hilbert space $\mathcal{H}$ induced by positive definite kernel function K. As previously, τ stands for a regularization parameter. Taking into account (26) and the representer theorem [6], function d can be written as

$$d(\mathbf{x}) = \sum_{j=1}^{N}\alpha_j K(\mathbf{x}, \mathbf{x}_j). \tag{27}$$

In the SVM an additional unregularized bias term is used. According to the representer theorem this term is not needed. However, in the remaining part of this paper the bias term γ_0 is added and its usefulness will be investigated in the experimental part:

$$d(\mathbf{x}) = \sum_{j=1}^{N}\alpha_j K(\mathbf{x}, \mathbf{x}_j) + \gamma_0. \tag{28}$$

Taking into account that $\theta_j \in \{-1, +1\}$ the α_j's can be written as $\alpha_j = \theta_j\gamma_j$. Thus, we can write

$$\theta_i d(\mathbf{x}_i) = \sum_{j=1}^{N}\gamma_j\theta_i\theta_j K(\mathbf{x}_i, \mathbf{x}_j) + \theta_i\gamma_0. \tag{29}$$

From RHKS theory it is well-known that

$$\begin{aligned}\|d\|_K^2 &= \sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_i\alpha_j K\left(\mathbf{x}_i,\mathbf{x}_j\right)\\ &= \sum_{i=1}^{N}\sum_{j=1}^{N}\gamma_i\gamma_j\theta_i\theta_j K\left(\mathbf{x}_i,\mathbf{x}_j\right).\end{aligned} \tag{30}$$

Defining $(N\times N)$-dimensional kernel matrix

$$\mathcal{K} = \left[\theta_i\theta_j\ K(\mathbf{x}_i,\mathbf{x}_j)\right]_{i,j=1}^{N}, \tag{31}$$

and vectors $\mathbf{\Gamma}^\top = [\gamma_1,\gamma_2,\cdots,\gamma_N]^\top$, $\mathbf{\Theta}^\top = [\theta_1,\theta_2,\cdots,\theta_N]^\top$ criterion function (26) for weighted (using $\mathbf{H}^{(k)}$) least square loss takes the form

$$\begin{aligned}\min_{\substack{\Gamma^{(k)}\in\mathbb{R}^N\\ \gamma_0^{(k)}\in\mathbb{R}}} J^{(k)}\left(\mathbf{\Gamma}^{(k)},\gamma_0^{(k)}\right) &= \frac{\tau}{2}\left(\mathbf{\Gamma}^{(k)}\right)^\top\mathcal{K}\mathbf{\Gamma}^{(k)}\\ &+\frac{1}{2}\left(\mathcal{K}\mathbf{\Gamma}^{(k)}+\gamma_0^{(k)}\mathbf{\Theta}-\mathbf{1}\right)^\top\mathbf{H}^{(k)}\left(\mathcal{K}\mathbf{\Gamma}^{(k)}+\gamma_0^{(k)}\mathbf{\Theta}-\mathbf{1}\right).\end{aligned} \tag{32}$$

Let $\Phi:\mathbf{x}\in\mathbb{R}^t\longmapsto\Phi(\mathbf{x})\in\mathcal{F}$ be a nonlinear transformation of the input vectors $\mathbf{x}$ into a feature space $\mathcal{F}$, which may be high- or even infinite-dimensional. Taking into account that K may be decomposed $K(\mathbf{x}_i,\mathbf{x}_j)=\Phi(\mathbf{x}_i)^\top\Phi(\mathbf{x}_j)$ minimization of (32) means that we design a linear IRLS classifier in the feature space $\mathcal{F}$, i.e. a nonlinear one in the original input space. Commonly used kernel functions includes polynomial and Gaussian:

$$K\left(\mathbf{x},\mathbf{x}_i\right) = \left(1+\beta\mathbf{x}^\top\mathbf{x}_i\right)^\delta;\quad \beta\in\mathbb{R}_+,\delta\in\mathbb{N}, \tag{33}$$

$$K\left(\mathbf{x},\mathbf{x}_i\right) = \exp\left(-\chi\left\|\mathbf{x}-\mathbf{x}_i\right\|^2\right);\quad \chi\in\mathbb{R}_+, \tag{34}$$

where β, δ and χ denote parameters.

The decision function of the classifier for input pattern $\mathbf{x}$ can be represented as

$$d\left(\mathbf{x}\right) = \sum_{i=1}^{N}\theta_i\gamma_i\ K(\mathbf{x},\mathbf{x}_i)+\gamma_0, \tag{35}$$

where $\{\gamma_i\}_{i=0}^N$ are parameters of the classifier obtained in the process of training. Before we use the conjugate gradient optimization of (32), it is rewritten in more compact form. Defining $\mathbf{\Gamma}_e^\top = [\gamma_0,\gamma_1,\cdots,\gamma_N]^\top = \left[\gamma_0\ \mathbf{\Gamma}^\top\right]^\top$ and $\widetilde{\mathcal{K}} = [\mathbf{\Theta}\ \mathcal{K}]$, (32) takes the form

$$\begin{aligned}\min_{\mathbf{\Gamma}_e^{(k)}\in\mathbb{R}^{N+1}} J^{(k)}\left(\mathbf{\Gamma}_e^{(k)}\right) &= \frac{\tau}{2}\left(\mathbf{\Gamma}_e^{(k)}\right)^\top\widetilde{\mathcal{I}}^\top\mathcal{K}\widetilde{\mathcal{I}}\,\mathbf{\Gamma}_e^{(k)}\\ &+\frac{1}{2}\left(\widetilde{\mathcal{K}}\mathbf{\Gamma}_e^{(k)}-\mathbf{1}\right)^\top\mathbf{H}^{(k)}\left(\widetilde{\mathcal{K}}\mathbf{\Gamma}_e^{(k)}-\mathbf{1}\right),\end{aligned} \tag{36}$$

where matrix $\widetilde{\mathcal{I}} = [\mathbf{0}\ \mathbf{I}]$ is used to unregularize the bias term.

For minimization of (36) with respect to $\mathbf{\Gamma}_e^{(k)}$ the conjugate gradient method can be used. For the sake of simplicity in minimizing sequence $\mathbf{\Gamma}_e^{(k),[n]}$, where $n=0,1,\cdots$ the superscript (k) will be omitted

$$\mathbf{\Gamma}_e^{[n+1]} = \mathbf{\Gamma}_e^{[n]}+\nu^{[n]}\mathbf{d}^{[n]}, \tag{37}$$

where as previously $\nu^{[n]}$ denotes the step size and $\mathbf{d}^{[n]}$ stands for the search direction. Repeating calculations from the linear case, it is easy obtain that

$$\begin{aligned}\nu^{[n]} &= -\frac{\tau\left(\mathbf{d}^{[n]}\right)^\top\widetilde{\mathcal{I}}^\top\mathcal{K}\widetilde{\mathcal{I}}\,\mathbf{\Gamma}^{[n]}+\left(\mathbf{d}^{[n]}\right)^\top\widetilde{\mathcal{K}}^\top\mathbf{H}\,\mathbf{e}^{[n]}}{\left(\mathbf{d}^{[n]}\right)^\top\mathbf{G}\,\mathbf{d}^{[n]}}\\ &= -\frac{\left(\mathbf{d}^{[n]}\right)^\top\mathbf{g}^{[n]}}{\left(\mathbf{d}^{[n]}\right)^\top\mathbf{G}\,\mathbf{d}^{[n]}},\end{aligned} \tag{38}$$

$$\beta^{[n]} = -\frac{\left(\mathbf{d}^{[n-1]}\right)^\top\mathbf{G}\,\mathbf{g}^{[n]}}{\left(\mathbf{d}^{[n-1]}\right)^\top\mathbf{G}\,\mathbf{d}^{[n-1]}}, \tag{39}$$

where in this case $\mathbf{G} = \tau\widetilde{\mathcal{I}}^\top\mathcal{K}\widetilde{\mathcal{I}}+\widetilde{\mathcal{K}}^\top\mathbf{H}\widetilde{\mathcal{K}}$, $\mathbf{e}^{[n]} = \widetilde{\mathcal{K}}\mathbf{\Gamma}_e^{[n]}-\mathbf{1}$ and

$$\mathbf{g}^{[n]} = \left.\frac{\partial J\left(\mathbf{\Gamma}_e\right)}{\partial\mathbf{\Gamma}_e}\right|_{\mathbf{\Gamma}_e=\mathbf{\Gamma}_e^{[n]}} = \tau\widetilde{\mathcal{I}}^\top\mathcal{K}\widetilde{\mathcal{I}}\,\mathbf{\Gamma}_e^{[n]}+\widetilde{\mathcal{K}}^\top\mathbf{H}\,\mathbf{e}^{[n]}. \tag{40}$$

The algorithm for design a nonlinear IRLS classifier can be summarized in the following steps:

1. Fix $\tau>0$, $\mathbf{H}^{(0)}=\mathbf{I}$ and calculate the kernel matrix $\widetilde{\mathcal{K}}$. Set the iteration index $k=0$.
2. $\mathbf{\Gamma}^{(k),[0]}=\mathbf{0}$. Set the iteration index $n=0$.
3. Calculate $\mathbf{e}^{(k),[n]}$.
4. Calculate $\mathbf{g}^{(k),[n]}$ using (40).
5. if $n=0$ then $\beta^{(k),[0]}=0$ else calculate $\beta^{(k),[n]}$ using (39).
6. Calculate the search direction $\mathbf{d}^{(k),[n]}$ using (22).
7. Calculate $\nu^{(k),[n]}$ using (38).
8. Update $\mathbf{\Gamma}_e^{(k),[n+1]}$ using (37).
9. Calculate $\mathbf{e}^{(k),[n]}$.
10. if $\|\mathbf{\Gamma}_e^{(k),[n+1]}-\mathbf{\Gamma}_e^{(k),[n]}\|<\zeta$ then go to (11), else $n\leftarrow n+1$, go to (3).
11. $\mathbf{H}^{(k+1)} = \operatorname{diag}\left(h_1^{(k+1)},h_2^{(k+1)},\cdots,h_N^{(k+1)}\right)$, where $h_i^{(k+1)} = f\left(e_i^{(k)}\right)$, for $i=1,2,\cdots,N$, and $f(\cdot)$ stands for selected loss function (10), (12)–(17).
12. if $k>1$ and $\left\|\mathbf{\Gamma}_e^{(k),[n_{max}^k+1]}-\mathbf{\Gamma}_e^{(k-1),[n_{max}^{k-1}+1]}\right\|<\xi$, then stop
    else $k\leftarrow k+1$, go to (2).

Remarks. The quantity n_{max}^k+1 denotes a number of iteration performed in the kth step. The iterations were stopped as soon as the Euclidean norm in a successive pair of $\mathbf{\Gamma}_e$ vectors is less than ζ, where ζ is a pre-set small positive value. In all experiments $\zeta=10^{-6}$ and $\xi=10^{-3}$ are used. The conjugate gradient algorithm converges theoretically in $p+1$ steps, where p is rank of matrix $\mathbf{G}$. If matrix $\mathbf{G}$ is full rank then number of steps is equal to $N+1$, where N denotes the cardinality of the training set. The quantity $\left(\mathbf{d}^{[n]}\right)^\top\mathbf{G}\mathbf{d}^{[n]}$ calculated in step (7) may be saved and used in step (5) of the next iteration.

4. Classifier design method – ℓ_1-regularized nonlinear case

The method for classifier design presented in the previous section is based on ℓ_2 regularization. However, it is well known that using of the ℓ_1 regularization encourages small components of $\mathbf{\Gamma}$ to become zero and inducing sparse solution [29, 30]. In contrast, solution of ℓ_2 regularization problem has all nonzero elements of $\mathbf{\Gamma}$. Now, criterion function (32) takes the form

$$\min_{\substack{\Gamma^{(k)}\in\mathbb{R}^N \\ \gamma_0^{(k)}\in\mathbb{R}}} J^{(k)}\left(\mathbf{\Gamma}^{(k)},\gamma_0^{(k)}\right)=\frac{\tau}{2}\left\|\mathbf{\Gamma}^{(k)}\right\|_1 + \frac{1}{2}\left(\mathcal{K}\mathbf{\Gamma}^{(k)}+\gamma_0^{(k)}\mathbf{\Theta}-\mathbf{1}\right)^{\top}\mathbf{H}^{(k)}\left(\mathcal{K}\mathbf{\Gamma}^{(k)}+\gamma_0^{(k)}\mathbf{\Theta}-\mathbf{1}\right), \tag{41}$$

where $\left\|\mathbf{\Gamma}^{(k)}\right\|_1=\sum_{i=1}^{N}\left|\gamma_i^{(k)}\right|$. The above criterion function is convex but not differentiable. Several optimization algorithms have been proposed to solve the above minimization, including: LARS [31], LASSO [32] and interior-point [33] methods. Among these methods gradient projection method introduced in [34, 35], and used for sparse signal reconstruction in [30], is especially interesting due to its low computational effort comparing to other methods. In this section this idea is used for minimization of criterion function (39). Introducing $\mathbf{\Gamma}^+,\mathbf{\Gamma}^-\succeq\mathbf{0}$ such that $\mathbf{\Gamma}=\mathbf{\Gamma}^+-\mathbf{\Gamma}^-$ the minimization problem (39) (omitting the iteration index k) may be written as

$$\min_{\substack{\Gamma^+\succeq\mathbf{0}\\ \mathbf{\Gamma}^-\succeq\mathbf{0}\\ \gamma_0\in\mathbb{R}}} J\left(\mathbf{\Gamma}^+,\mathbf{\Gamma}^-,\gamma_0\right)=\frac{\tau}{2}\mathbf{1}^{\top}\left(\mathbf{\Gamma}^++\mathbf{\Gamma}^-\right) + \frac{1}{2}\left(\mathcal{K}\mathbf{\Gamma}^+-\mathcal{K}\mathbf{\Gamma}^-+\gamma_0\mathbf{\Theta}-\mathbf{1}\right)^{\top}\mathbf{H}\left(\mathcal{K}\mathbf{\Gamma}^+-\mathcal{K}\mathbf{\Gamma}^-+\gamma_0\mathbf{\Theta}-\mathbf{1}\right). \tag{42}$$

So, we obtain bound-constrained quadratic programming formulation of (39). Assuming that γ_0 is fixed, after some simple matrix algebra the above criterion function takes the following form

$$J\left(\mathbf{\Gamma}^{\pm}\right)=\frac{1}{2}\left(\mathbf{\Gamma}^{\pm}\right)^{\top}\mathbf{A}\mathbf{\Gamma}^{\pm}+\mathbf{b}_1^{\top}\mathbf{\Gamma}^{\pm}+c, \tag{43}$$

where

$$\mathbf{\Gamma}^{\pm}=\begin{bmatrix}\mathbf{\Gamma}^+\\ \mathbf{\Gamma}^-\end{bmatrix},$$

$$\mathbf{A}=\begin{bmatrix}\mathcal{K}\mathbf{H}\mathcal{K} & -\mathcal{K}\mathbf{H}\mathcal{K}\\ -\mathcal{K}\mathbf{H}\mathcal{K} & \mathcal{K}\mathbf{H}\mathcal{K}\end{bmatrix},$$

$$\mathbf{b}_1=\begin{bmatrix}\tau\mathbf{1}-\mathcal{K}\mathbf{H}\left(\mathbf{1}-\gamma_0\mathbf{\Theta}\right)\\ \tau\mathbf{1}+\mathcal{K}\mathbf{H}\left(\mathbf{1}-\gamma_0\mathbf{\Theta}\right)\end{bmatrix}$$

and

$$c=\frac{1}{2}\left(\mathbf{1}-\gamma_0\mathbf{\Theta}\right)^{\top}\left(\mathbf{1}-\gamma_0\mathbf{\Theta}\right).$$

Gradient of criterion function (41) can be written as

$$\mathbf{g}=\begin{bmatrix}\tau\mathbf{1}+\mathcal{K}\mathbf{H}\mathbf{e}\\ \tau\mathbf{1}-\mathcal{K}\mathbf{H}\mathbf{e}\end{bmatrix}, \tag{44}$$

where error vector is $\mathbf{e}=\mathcal{K}\left(\mathbf{\Gamma}^+-\mathbf{\Gamma}^-\right)+\gamma_0\mathbf{\Theta}-\mathbf{1}$. The feasible region of solution to (41) is nonnegative orthant $\Omega=\left\{\mathbf{\Gamma}^{\pm}\middle|\,\mathbf{\Gamma}^{\pm}\in\mathbb{R}^{2N},\mathbf{\Gamma}^{\pm}\succeq\mathbf{0}\right\}$. The gradient projection method is based on successive projections of the negative gradient on the feasible region Ω and performing line search minimization. The descent direction is computed for nth iteration as [35]

$$\mathbf{d}^{[n]}=P_{\Omega}\left(\mathbf{\Gamma}^{\pm,[n]}-\alpha^{[n]}\mathbf{g}^{[n]}\right)-\mathbf{\Gamma}^{\pm,[n]}, \tag{45}$$

where P_{Ω} stands for the orthogonal projection on Ω and $\mathbf{\Gamma}^{\pm,[n]}$ denotes $\mathbf{\Gamma}^{\pm}$ for the nth iteration. For bound constraints from (40) P_{Ω} is simply replaced by componentwise maximum operation, i.e. $P_{\Omega}(\cdot)=\max(\mathbf{0},\cdot)$. The steplength $\alpha^{[n]}$ selection is based on the Barzilai-Borwein rule

$$\widehat{\alpha}^{[n]}=\frac{\left(\mathbf{d}^{[n-1]}\right)^{\top}\mathbf{d}^{[n-1]}}{\left(\mathbf{d}^{[n-1]}\right)^{\top}\mathbf{A}\mathbf{d}^{[n-1]}}, \tag{46}$$

with the following modification [34, 35]: if $\left(\mathbf{d}^{[n-1]}\right)^{\top}\mathbf{A}\mathbf{d}^{[n-1]}\leq 0$, then $\alpha^{[n]}=\alpha_{\max}$, else $\alpha^{[n]}=\operatorname{med}\left\{\alpha_{\min},\widehat{\alpha}^{[n]},\alpha_{\max}\right\}$, where $\operatorname{med}\{\cdot\}$ stands for the median operation, $0<\alpha_{\min}<\alpha_{\max}$ denote preset values (usually $\alpha_{\min}=10^{-30}$ and $\alpha_{\max}=10^{30}$). Vector $\mathbf{\Gamma}^{\pm}$ is updated using

$$\mathbf{\Gamma}^{\pm,[n+1]}=\mathbf{\Gamma}^{\pm,[n]}+\lambda^{[n]}\mathbf{d}^{[n]}, \tag{47}$$

with $\lambda^{[n]}$ given by a limited minimization

$$\lambda^{[n]}=\arg\min_{\lambda\in[0,1]} J\left(\mathbf{\Gamma}^{\pm,[n]}+\lambda\mathbf{d}^{[n]}\right). \tag{48}$$

The above minimization can be obtained for (41) analytically

$$\frac{\partial J\left(\mathbf{\Gamma}^{\pm,[n]}+\lambda\mathbf{d}^{[n]}\right)}{\partial\lambda}=0. \tag{49}$$

Some simple algebra yields

$$\widehat{\lambda}=-\frac{\left(\mathbf{d}^{[n]}\right)^{\top}\mathbf{g}^{[n]}}{\left(\mathbf{d}^{[n]}\right)^{\top}\mathbf{A}\mathbf{d}^{[n]}}. \tag{50}$$

Taking into account that (41) is strictly convex the limited minimization (46) given

$$\lambda^{[n]}=\operatorname{med}\left\{0,-\frac{\left(\mathbf{d}^{[n]}\right)^{\top}\mathbf{g}^{[n]}}{\left(\mathbf{d}^{[n]}\right)^{\top}\mathbf{A}\mathbf{d}^{[n]}},1\right\}. \tag{51}$$

The condition for optimality of (41) with respect to γ_0 is given assuming that $\mathbf{\Gamma}^{\pm}$ is fixed, differentiating (42) with respect γ_0 and setting the result equals to zero

$$\gamma_0^{[n]}=-\frac{\mathbf{\Theta}^{\top}\mathbf{H}\left(\mathcal{K}\mathbf{\Gamma}^{+,[n]}-\mathcal{K}\mathbf{\Gamma}^{-,[n]}-\mathbf{1}\right)}{\mathbf{\Theta}^{\top}\mathbf{H}\mathbf{\Theta}}. \tag{52}$$

The algorithm for design ℓ_1-regularized nonlinear IRLS classifier can be summarized in the following steps:

1. Fix $\tau>0$, $\mathbf{H}^{(0)}=\mathbf{I}$, $\alpha_{\min}=10^{-30}$ and $\alpha_{\max}=10^{30}$. Calculate the kernel matrix $\mathcal{K}$ using (31). Set the iteration index $k=0$.

2. $\mathbf{\Gamma}^{\pm,(k),[0]} = \mathbf{0}$. Set the iteration index $n = 0$.
3. Calculate $\gamma_0^{(k),[n]}$ using (52).
4. Calculate $\mathbf{e}^{(k),[n]}$.
5. Calculate $\mathbf{g}^{(k),[n]}$ using (42).
6. Calculate $\mathbf{d}^{[n]}$ using (43).
7. Calculate $\lambda^{[n]}$ using (50).
8. Calculate $\mathbf{\Gamma}^{\pm,[n+1]}$ using (45).
9. if $\left(\mathbf{d}^{[n-1]}\right)^{\top} \mathbf{A}\mathbf{d}^{[n-1]} \leq 0$, then $\alpha^{[n]} = \alpha_{\max}$, else calculate $\widehat{\alpha}^{[n]}$ using (46) and set $\alpha^{[n]} = \text{med}\left\{\alpha_{\min}, \widehat{\alpha}^{[n]}, \alpha_{\max}\right\}$.
10. Calculate $\gamma_0^{(k),[n]}$ using (52).
11. if $\|\mathbf{\Gamma}^{\pm,(k),[n+1]} - \mathbf{\Gamma}^{\pm,(k),[n]}\| < \zeta$ then go to (12), else $n \leftarrow n+1$, go to (4).
12. $\mathbf{H}^{(k+1)} = \text{diag}\left(h_1^{(k+1)}, h_2^{(k+1)}, \cdots, h_N^{(k+1)}\right)$, where $h_i^{(k+1)} = f\left(e_i^{(k)}\right)$, for $i = 1, 2, \cdots, N$, and $f(\cdot)$ stands for selected loss function (10), (12)–(17).
13. if $k > 1$ and $\left\|\mathbf{\Gamma}^{\pm,(k),[n^k_{max}+1]} - \mathbf{\Gamma}^{\pm,(k-1),[n^{k-1}_{max}+1]}\right\| < \xi$, then stop
    else $k \leftarrow k+1$, go to (2).

Remarks. The iterations were stopped as soon as the Euclidean norm in a successive pair of $\mathbf{\Gamma}^{\pm}$ vectors is less than ζ, where ζ is a pre-set small positive value. In all experiments $\zeta = 10^{-3}$ is used. The quantity $\left(\mathbf{d}^{[n]}\right)^{\top} \mathbf{A}\mathbf{d}^{[n]}$ calculated in step (9) may be saved and used in step (7) of the next iteration.

5. Numerical experiments and discussion

All experiments were done on Hewlett-Packard HP Compaq dx7300 Intel Core 2 CPU 6300 @ 1.86 GHz with 1 GB RAM, running Windows XP (Service Pack 2) and MATLAB 6.5 environment. The lagrangian support vector machine (LSVM) was obtained by Internet – http://www.cs.wisc.edu/dmi/lsvm. For experiments 13 well-known benchmark datasets from IDA repository was used http://ida.first.fraunhofer.de/projects/bench. For 'Banana' (Ban), 'Breast Cancer' (BreC), 'Diabetis' (Diab), 'Flare Solar' (FlaSol), 'German' (Ger), 'Heart' (Hea), 'Image' Ima, 'Ringnorm' (RingN), 'Splice' (Spl), 'Thyroid' (Thy), 'Titanic' (Tita), 'Twonorm' (TwN) and 'Waveform' (WaF) this repository includes 100 predefined splits into training and testing sets. Another dataset used in experiments was Ripley synthetic two-class problem (Syn) – http://www.stats.ox.ac.uk/pub/PRNN. The union of original training and testing parts were randomly partitioned into 100 pairs of training (250 points) and testing (1000 points) sets. The structure of experiments is as follows: for each database 10% of splits were used to estimate the parameters (regularization parameter and kernel parameter). These parameters was used to 100-fold cross validation. The averege and standard deviation of generalization error as well as computing time were used to comparison with the LSVM algorithm.

5.1. Linear classification. The purpose of experiments in this subsection was to compare the generalization ability of the linear version LSVM to the linear IRLS classifier with various approximations of misclassification error. In all experiments the parameter ν for LSVM was changed in the range from 0.1 to 25 (step 0.1). The regularization parameter τ was in the range from 0.1 to 2.5 (step 0.01). Table 1 shows for each database the average generalization performance (top line) and standard deviation (middle line).

Table 1
Comparison between Lagrangian Support Vector Machine (LSVM) and linear IRLS classifier on benchmark datasets. Each cell contains: average (top), standard deviation (middle) of generalization error and running time (bottom)

Name	LSVM	ASQR	ALIN	SIG	ASIGL	AHUB	ALOG	ALOGL
	46.985	46.969	45.534	**45.196**	46.966	45.484	46.457	46.990
	4.472	4.463	3.203	**3.201**	4.466	3.898	4.216	4.479
Ban	1.00	**0.36**	0.90	0.59	**0.38**	0.42	0.50	0.48
	27.065	27.091	28.260	27.390	**26.935**	27.172	27.494	27.260
	4.580	4.799	**4.552**	4.832	4.755	4.766	4.732	4.899
BreC	1.00	**0.19**	0.73	0.23	0.27	0.31	0.32	0.59
	23.387	23.417	23.447	23.523	23.363	23.413	23.417	**23.353**
	1.778	1.796	**1.706**	1.708	1.786	1.797	1.797	1.775
Diab	1.00	**0.15**	0.68	0.16	0.27	0.21	0.24	0.38
	33.210	33.415	**32.532**	33.660	33.485	33.015	32.940	33.493
	1.686	1.525	2.253	1.973	1.555	1.690	1.693	**1.515**
FlaSol	1.00	**0.13**	0.24	0.15	0.25	0.20	0.24	0.39
	24.077	24.073	24.037	24.823	24.383	24.043	24.053	**24.017**
	2.150	2.154	2.269	**2.096**	2.183	2.236	2.186	2.154
Ger	1.00	**0.13**	0.57	0.15	0.21	0.21	0.20	0.36
	16.110	16.480	16.780	16.350	**16.030**	16.670	16.620	16.560
	2.814	2.827	2.911	3.448	2.983	3.015	3.034	**2.790**
Hea	1.00	0.33	0.89	**0.29**	0.56	0.41	0.45	0.57
	16.832	16.891	15.500	16.351	16.020	**15.411**	15.609	16.985
	0.878	0.949	0.952	1.037	0.962	0.954	**0.745**	0.975
Ima	1.00	0.13	0.53	**0.09**	0.25	0.19	0.21	0.25
	25.031	25.109	24.729	**24.580**	24.860	24.751	24.750	25.179
	0.810	0.828	0.690	**0.612**	0.768	0.726	0.724	0.847
RingN	1.00	**0.40**	0.82	0.41	0.53	0.46	0.48	0.50
	16.202	**16.172**	16.260	16.292	16.211	16.189	16.184	16.253
	0.784	0.723	0.658	0.693	**0.650**	0.783	0.721	0.778
Spl	1.00	0.26	0.46	**0.15**	0.32	0.34	0.34	0.47
	11.872	11.840	11.824	11.862	11.816	**11.811**	11.825	11.824
	0.648	0.654	**0.623**	0.688	0.661	0.665	0.683	0.643
Syn	1.00	0.20	0.53	**0.19**	0.33	0.23	0.25	0.33
	10.800	10.760	10.720	15.720	11.067	**10.067**	10.240	10.627
	2.629	2.641	2.693	3.161	2.677	2.495	2.528	**2.393**
Thy	1.00	0.31	0.70	**0.26**	0.65	0.38	0.41	0.55
	22.686	22.685	22.649	**22.593**	22.654	22.630	22.652	22.685
	1.082	1.132	1.110	1.114	1.152	1.160	1.139	1.139
Tita	1.00	**0.42**	0.66	0.56	0.57	0.56	0.61	0.85
	2.836	2.803	2.852	**2.604**	2.640	2.783	2.735	2.763
	0.262	0.258	0.264	**0.168**	0.185	0.245	0.225	0.242
TwN	1.00	0.50	0.88	**0.37**	0.68	0.53	0.54	0.57
	13.275	13.297	13.228	15.650	13.705	13.195	**13.138**	13.317
	0.673	0.673	0.623	0.999	0.818	0.606	**0.596**	0.680
WaF	1.00	0.45	0.90	**0.32**	1.08	0.53	0.58	0.68

In each cell the bottom line shows computing time normalized to the computing time of the LSVM. The best results for each database are in boldface. The results demonstrate that IRLS performs better than LSVM. The average and standard deviation (confidence interval) of generalization error for each database excluding Tita are lower for the IRLS. Only for Tita

the standard deviation of the generalization error is lower for the LSVM. Running time is also lower for the IRLS classifier, except one case – WaF database and the ASIGL approximation of misclassification error. For 11 databases the AHUB and ALOG approximations lead to lower average generalization error comparing to the LSVM. However, the AHUB and ALOG approximations lead to lower standard deviation of generalization error, respectively for 6 and 7 databases. Both average and standard deviation of generalization error are better with respect to the LSVM for 7 databases in case of the ALOG approximation and for 6 databases in case of the AHUB approximation. The worse results are obtained for the ASQR and SIG approximations – only for 3 databases both average and standard deviation of generalization error are better comparing to the LSVM. The running time for the IRLS is up to 10 times lower comparing to the LSVM.

5.2. ℓ_2-regularized nonlinear classification. The purpose of experiments in this subsection was to compare the generalization ability of the kernel version LSVM to the kernel ℓ_2-regularized IRLS classifier with various approximations of misclassification error. Comparison was performed for the IRLS classifier with bias γ_0 and for bias equals zero. In all experiments the parameter ν for LSVM was changed in the range from 0.1 to 25 (step 0.1). The regularization parameter τ was in the range from 0.1 to 2.5 (step 0.01). The Gaussian kernel (34) was used. The kernel parameter χ was selected from the set $\{0.01, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0, 20.0, 50.0\}$. Tables 2 and 3 show the results for IRLS classifier with bias and without bias, respectively. As previously, for each database the average generalization performance (top line) and standard deviation (middle line) are presented. In each cell the bottom line shows computing time normalized to the computing time of the kernel LSVM. The best results for each database are in boldface.

Taking Table 2 into account, several observations can be made. The average of generalization error for 4 databases (Ban, Syn, Thy and TwN) are lower for the LSVM. However, only for TwN both the average and standard deviation of the generalization error are lower for the LSVM. For the ALIN approximation the IRLS classifier leads to lower average generalization error for 10 databases. For the ASQR, ASIGL, AHUB, ALOG and ALOGL approximations the IRLS performs better with respect to the average generalization error for 8 databases. Both average and standard deviation of generalization error are better for TwN database in case of the LSVM, whereas in case of the AHUB approximation for 7 databases. Running time is lower for the IRLS classifier for Ban, RingN, Spl, Tita, TwN and WaF databases up to 10 times. For the rest of databases the running time is lower for the LSVM up to 10 times. However, the running time is usually lower for the IRLS classifier for large databases.

Taking Table 3 which contains comparison LSVM to IRLS without bias, the following observations can be made. The average of generalization error for 5 databases (Ban, BreC Syn, Thy and TwN) are lower for the LSVM. However, for BreC, Syn and TwN both the average and standard deviation of the generalization error is lower for the LSVM. For the ALIN, ASIGL and ALOG approximations the IRLS classifier leads to lower average generalization error for 9 databases. Both average and standard deviation of generalization error are better in case of the ALIN approximation for 7 databases. Running time is usually lower for the IRLS classifier up to 10 times. However, for small databases (BreC, Thy) the running time is lower for the LSVM classifier up to 4 times.

Table 2
Comparison between kernel version of Lagrangian Support Vector Machine (LSVM) and kernel IRLS classifier (with bias) on benchmark datasets. Each cell contains: average (top), standard deviation (middle) of generalization error and running time (bottom)

Name	LSVM	ASQR	ALIN	SIG	ASIGL	AHUB	ALOG	ALOGL
	10.344	10.390	10.450	10.398	10.378	10.403	10.386	10.400
	0.429	0.420	0.397	**0.394**	0.415	0.421	0.423	0.432
Ban	1.00	0.54	0.62	**0.20**	0.36	0.35	0.34	0.47
	25.195	26.325	**24.805**	25.727	25.623	25.753	25.506	26.805
	3.963	4.738	4.167	4.308	4.662	4.522	4.501	4.794
BreC	**1.00**	2.30	4.22	2.46	2.75	3.30	4.11	10.75
	23.143	23.086	23.003	**22.993**	23.113	23.117	23.050	23.067
	1.687	1.775	1.654	**1.638**	1.758	1.674	1.710	1.668
Diab	1.00	2.32	4.29	**0.62**	1.27	11.57	1.17	2.40
	33.645	33.318	**32.657**	33.727	33.410	33.193	32.990	33.407
	1.759	**1.540**	1.598	1.758	1.844	1.593	1.627	1.579
FlaSol	1.00	**0.66**	2.13	1.03	1.11	0.89	1.06	1.77
	23.587	23.303	23.397	23.537	23.400	23.317	**23.300**	23.417
	2.148	2.272	2.163	2.167	2.211	2.155	2.195	2.164
Ger	**1.00**	1.54	3.70	2.07	1.65	1.99	2.08	4.33
	15.680	15.850	**15.440**	15.910	15.700	15.720	15.700	15.760
	3.345	**3.066**	3.163	3.282	3.261	3.101	3.112	3.159
Hea	1.00	**0.96**	2.90	1.53	1.48	1.84	1.95	3.50
	4.619	3.054	3.059	**2.906**	2.916	3.074	3.074	2.871
	0.709	0.640	0.675	0.616	0.625	**0.600**	0.601	0.609
Ima	1.00	1.09	1.44	0.24	**0.06**	0.41	1.31	1.77
	9.214	1.529	1.607	1.823	1.755	1.529	**1.521**	1.524
	1.388	0.092	0.113	0.157	0.138	0.091	0.092	**0.084**
RingN	1.00	0.11	0.18	**0.08**	0.10	0.10	0.11	0.30
	12.028	10.956	**10.816**	10.885	10.885	10.949	11.005	11.023
	0.731	0.685	0.646	0.606	**0.592**	0.687	0.734	0.780
Spl	1.00	0.89	0.95	1.19	**0.69**	0.87	0.90	2.66
	9.460	9.570	9.639	9.491	9.531	9.521	9.580	9.496
	0.545	0.577	0.622	0.534	0.564	0.564	**0.531**	0.564
Syn	1.00	1.13	1.54	**0.40**	1.10	1.07	1.93	0.79
	4.147	4.293	4.187	4.160	4.507	4.333	4.360	4.333
	2.305	2.002	2.186	2.104	2.065	2.079	2.066	**1.973**
Thy	**1.00**	2.37	3.27	1.51	2.10	2.37	2.57	7.14
	22.455	22.372	**22.083**	22.456	22.406	22.412	22.370	22.380
	1.250	1.018	1.680	1.012	1.050	1.061	**0.980**	1.041
Tita	1.00	**0.08**	0.11	0.09	0.12	0.10	0.11	0.18
	2.387	2.518	2.557	2.498	2.472	2.505	2.474	2.488
	0.121	0.157	0.179	0.149	0.178	0.156	0.144	0.159
TwN	1.00	0.07	0.09	0.08	**0.05**	0.07	0.07	0.09
	10.274	10.004	10.072	**9.732**	9.750	10.037	10.055	10.023
	0.407	0.385	0.508	0.385	0.401	**0.374**	0.386	0.376
WaF	1.00	0.19	0.21	**0.09**	0.10	0.19	0.20	0.25

Table 3
Comparison between kernel version of Lagrangian Support Vector Machine (LSVM) and kernel IRLS classifier (without bias) on benchmark datasets. Each cell contains: average (top), standard deviation (middle) of generalization error and running time (bottom)

Name	LSVM	ASQR	ALIN	SIG	ASIGL	AHUB	ALOG	ALOGL
	10.344	10.358	10.416	10.367	10.370	10.369	10.366	10.364
	0.429	0.411	0.415	0.417	0.418	**0.406**	0.409	0.412
Ban	1.00	0.37	0.47	**0.17**	0.25	0.38	0.38	0.47
	25.195	26.351	25.312	25.286	26.026	25.766	25.442	26.766
	3.963	4.804	4.447	4.497	4.622	4.636	4.548	4.824
BreC	1.00	**0.63**	1.87	0.60	1.06	1.03	1.14	2.20
	23.143	23.097	23.077	**22.933**	23.087	23.047	23.020	23.140
	1.687	1.710	**1.589**	1.692	1.673	1.690	1.709	1.681
Diab	1.00	0.64	2.19	**0.50**	1.06	0.92	1.00	1.99
	33.645	33.343	**32.540**	33.688	33.475	33.208	32.938	33.315
	1.759	**1.592**	1.704	1.676	1.795	1.742	1.645	1.744
FlaSol	1.00	**0.56**	1.84	0.79	0.97	0.77	0.92	1.57
	23.587	23.273	23.463	23.487	23.507	23.637	23.270	**23.197**
	2.148	2.216	2.236	2.296	2.252	2.240	2.194	2.220
Ger	1.00	1.18	3.05	**0.73**	1.33	1.59	1.66	3.52
	15.680	15.770	**15.470**	15.780	15.620	15.690	15.580	15.780
	3.345	**3.051**	3.189	3.246	3.234	3.123	3.260	3.060
Hea	1.00	0.86	1.64	**0.64**	0.84	1.01	1.08	2.01
	4.619	2.960	2.960	2.876	2.881	2.936	2.975	**2.792**
	0.709	0.561	0.585	0.600	0.549	**0.544**	0.547	0.596
Ima	1.00	0.90	0.92	**0.27**	0.46	0.89	0.88	1.24
	9.214	6.011	6.039	8.126	8.176	6.011	**6.010**	6.190
	1.388	**0.808**	0.908	1.525	1.497	0.808	0.815	0.869
RingN	1.00	0.08	0.08	**0.05**	0.06	0.08	0.08	0.17
	12.028	11.278	11.168	11.099	**11.067**	11.278	11.274	11.306
	0.731	0.694	0.612	0.704	0.664	0.714	**0.662**	0.726
Spl	1.00	0.68	0.72	**0.40**	0.56	0.68	0.69	2.13
	9.460	9.558	9.652	9.538	9.482	9.505	9.484	9.509
	0.545	0.578	0.628	0.566	0.558	0.556	0.563	0.582
Syn	1.00	0.64	0.99	**0.33**	0.50	0.66	0.68	1.09
	4.147	4.293	4.787	4.333	4.400	4.560	4.307	4.187
	2.305	**2.090**	2.259	2.197	2.302	2.252	2.201	2.110
Thy	1.00	2.46	2.98	**0.78**	1.56	2.55	2.55	4.30
	22.455	22.435	22.423	22.473	**22.398**	22.447	22.417	22.412
	1.250	1.086	1.148	**1.010**	1.039	1.084	1.100	1.067
Tita	1.00	**0.08**	0.11	0.09	0.10	0.10	0.10	0.14
	2.387	2.522	2.556	2.506	2.461	2.503	2.472	2.478
	0.121	0.157	0.151	0.148	0.145	0.155	0.144	0.155
TwN	1.00	0.06	0.07	**0.03**	0.05	0.06	0.06	0.07
	10.274	10.053	10.041	**9.715**	9.745	10.064	10.075	10.117
	0.407	0.435	0.453	**0.407**	0.436	0.414	0.414	0.454
WaF	1.00	0.14	0.16	**0.07**	0.08	0.14	0.15	0.19

Comparing the IRLS classifier with and without bias, it can be noted that excluding bias leads to a lower running time and a bit worse performance of the classifier. Moreover, for classifier with bias the ALOG approximation is the best, whereas for classifier without bias the ALIN approximation is the best. Taking into account the average of generalization error the IRLS with ALIN approximation usually gives the best performance in case of including a bias. For the IRLS classifier without a bias the LSVM usually outperforms it. In conclusion, for the kernel version of classifiers, the IRLS is not so concurrent to the LSVM as in linear case. It also can be noted that the superiority of a classification method depends on the dataset. The above presented conclusions relate to the 'average' behavior of the tested methods to the classifiers design. For each new database selection method of designing a classifier should be performed experimentally. The above is consistent with the no-free-lunch theorem, which says that if a classifier generalizes better to certain databases, then it is a result of a better match for a specific problem than its superiority over other classifiers.

5.3. ℓ_1-regularized nonlinear classification. The purpose of experiments in this subsection was to compare the generalization ability of the kernel version LSVM to the kernel ℓ_1-regularized IRLS classifier with various approximations of misclassification error. Comparison was performed for IRLS classifier with bias γ_0 and for bias equals zero. In all experiments the parameter ν for LSVM was changed in the range from 0.1 to 25 (step 0.1). The regularization parameter τ was in the range from 0.1 to 2.5 (step 0.01). The Gaussian kernel (34) was used. The kernel parameter χ was selected from the set $\{0.01, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0, 20.0, 50.0\}$. Tables 4 and 5 show the results for IRLS classifier with bias and without bias, respectively. As previously, for each database the average generalization performance (top line) and standard deviation (middle line) are presented. In each cell the bottom line shows computing time normalized to the computing time of the kernel LSVM. The best results for each database are in boldface.

Taking Table 4 into account, we see that the average of generalization error for 6 databases (Ban, BreC, Diab, Spl, Syn and TwN) are lower for the kernel LSVM. However, only for BreC, Spl, Syn, TwN both the average and standard deviation of the generalization error is lower for the kernel LSVM. For the ASQR and ASIGL approximations the IRLS classifier leads to lower average generalization error for 5 databases. Both average and standard deviation of generalization error are better for BreC, Spl, Syn, TwN databases in case of the LSVM, whereas in case of the ASIGL approximation also for 4 databases (FlaSol, Hea, RingN and Thy). Running time is lower for the kernel IRLS classifier for 11 databases up to 20 times. For the rest of databases (BreC, Hea and Thy databases) the running time is lower for the LSVM up to 15 times. However, the running time is usually lower for the IRLS classifier for large databases.

Table 4
Comparison between kernel version of Lagrangian Support Vector Machine (LSVM) and ℓ_1-regularized kernel IRLS classifier (with bias) on benchmark datasets. Each cell contains: average (top), standard deviation (middle) of generalization error and running time (bottom)

Name	LSVM	ASQR	ALIN	SIG	ASIGL	AHUB	ALOG	ALOGL
	10.344	10.544	10.435	10.502	10.487	10.570	10.569	10.529
	0.429	0.469	**0.397**	0.427	0.418	0.449	0.411	0.423
Ban	1.00	0.46	0.20	**0.13**	0.14	0.51	0.53	0.61
	25.195	25.623	25.325	25.351	25.519	25.740	25.779	25.740
	3.963	4.172	4.183	4.109	4.239	4.207	4.029	4.291
BreC	**1.00**	3.18	2.36	1.73	1.36	1.65	1.80	4.30
	23.143	23.223	23.257	23.227	22.227	23.197	23.287	23.753
	1.687	**1.627**	1.667	1.814	1.659	1.591	1.593	1.841
Diab	1.00	0.65	0.65	**0.55**	0.65	0.65	0.65	0.65
	33.645	33.053	33.985	**32.425**	33.175	33.237	33.263	33.005
	1.759	1.829	2.542	1.779	1.683	1.775	1.797	**1.745**
FlaSol	1.00	0.88	0.50	**0.47**	0.49	0.50	0.50	0.50
	23.587	24.107	24.990	24.127	24.150	24.743	25.190	**23.407**
	2.148	2.062	2.189	2.166	2.034	**2.013**	2.230	2.239
Ger	1.00	**0.45**	**0.45**	**0.45**	**0.45**	**0.45**	**0.45**	**0.45**
	15.680	15.670	15.710	**15.410**	15.480	15.740	15.670	15.810
	3.345	3.337	3.288	3.216	3.252	3.199	**3.111**	3.228
Hea	**1.00**	2.16	3.68	1.90	2.21	3.44	3.44	7.27
	4.619	4.911	7.431	7.292	**4.594**	5.144	5.297	5.010
	0.709	0.648	1.550	1.461	0.900	0.672	**0.638**	0.769
Ima	1.00	0.36	**0.23**	0.48	0.42	0.62	0.63	0.86
	9.214	1.579	**1.528**	1.804	1.713	1.567	1.547	1.555
	1.388	0.117	0.108	0.131	0.126	0.114	**0.105**	0.110
RingN	1.00	0.14	0.13	**0.05**	0.08	0.14	0.14	0.19
	12.028	14.209	14.260	14.260	14.234	13.963	13.883	14.253
	0.731	1.325	1.316	1.332	1.337	1.352	1.330	1.417
Spl	1.00	**0.26**	0.30	**0.26**	**0.26**	0.35	0.37	**0.26**
	9.460	9.622	9.603	9.540	9.549	9.649	9.654	9.643
	0.545	0.586	0.581	0.556	0.557	0.640	0.631	0.601
Syn	1.00	1.14	0.73	**0.34**	0.40	1.47	1.37	1.60
	4.147	4.587	4.027	3.747	**3.680**	4.493	4.400	4.507
	2.305	2.205	1.960	1.775	**1.748**	2.341	2.393	2.280
Thy	**1.00**	8.10	3.21	2.06	2.35	8.27	8.42	15.64
	22.455	22.384	22.635	22.633	22.578	22.466	22.585	**22.216**
	1.250	1.123	0.941	**0.880**	0.931	1.102	0.962	1.021
Tita	1.00	0.10	0.11	0.10	**0.09**	0.11	0.11	0.12
	2.387	2.561	2.467	2.455	2.467	2.544	2.512	2.552
	0.121	0.181	0.159	0.142	0.144	0.187	0.165	0.175
TwN	1.00	0.07	0.05	**0.04**	**0.04**	0.07	0.06	0.05
	10.274	10.254	10.269	10.293	10.292	**10.201**	10.297	10.300
	0.407	0.861	0.920	0.962	0.946	0.638	0.677	0.841
WaF	1.00	0.13	0.12	**0.11**	**0.11**	0.14	0.15	0.13

Taking Table 5 which contains comparison of the kernel LSVM to the kernel IRLS without bias, the following observations can be made. The average of generalization error for 4 databases (Ban, Diab, Hea and Spl) are lower for the kernel LSVM. However, only for Diab and Spl both the average and standard deviation of the generalization error is lower for the kernel LSVM. For the ALOGL approximation the IRLS classifier leads to lower average generalization error for 7 databases. Both average and standard deviation of generalization error are better in case of the ASQR approximation for 4 databases. Running time is usually lower for the kernel IRLS classifier up to 33 times. However, for small databases (Hea, Hea and Thy) the running time is lower for the kernel LSVM classifier up to 12 times.

Table 5
Comparison between kernel version of Lagrangian Support Vector Machine (LSVM) and ℓ_1-regularized kernel IRLS classifier (without bias) on benchmark datasets. Each cell contains: average (top), standard deviation (middle) of generalization error and running time (bottom).

Name	LSVM	ASQR	ALIN	SIG	ASIGL	AHUB	ALOG	ALOGL
	10.344	10.461	10.429	10.476	10.478	10.520	10.518	10.490
	0.429	0.439	**0.380**	0.425	0.427	0.443	0.441	0.460
Ban	1.00	0.39	0.17	**0.13**	**0.13**	0.45	0.47	0.54
	25.195	25.260	**25.104**	25.312	25.286	25.130	25.610	25.325
	3.963	4.616	4.469	4.358	4.259	4.377	4.446	4.467
BreC	**1.00**	5.07	3.15	2.86	3.07	4.85	4.73	7.14
	23.143	23.580	23.580	24.050	23.980	23.537	23.590	23.637
	1.687	1.974	1.915	2.050	2.010	1.831	1.928	1.927
Diab	1.00	0.97	0.48	**0.39**	**0.39**	1.36	1.52	1.07
	33.645	33.960	**33.587**	33.815	33.922	33.697	33.752	33.917
	1.759	1.684	1.717	1.856	1.749	1.646	**1.590**	1.697
FlaSol	1.00	0.66	1.04	0.60	**0.57**	0.74	0.82	0.63
	23.587	23.783	23.440	23.517	23.603	23.540	23.563	**23.327**
	2.148	**2.101**	2.202	2.258	2.184	2.312	2.230	2.162
Ger	1.00	**0.75**	5.15	3.12	4.22	8.33	8.28	8.32
	15.680	16.340	17.060	16.640	16.020	16.330	16.060	15.860
	3.345	3.334	**3.104**	3.433	3.387	3.300	3.372	3.226
Hea	**1.00**	11.40	6.33	3.32	4.81	12.43	12.00	12.23
	4.619	5.460	8.228	7.807	7.817	5.579	5.708	5.624
	0.709	0.618	1.029	1.202	1.120	0.608	0.714	**0.495**
Ima	1.00	0.26	0.09	**0.05**	**0.05**	0.30	0.31	0.24
	9.214	6.638	5.808	**4.508**	4.627	6.662	6.649	7.646
	1.388	0.923	1.176	**0.832**	0.884	1.015	0.920	0.992
RingN	1.00	0.29	0.13	**0.09**	0.10	0.30	0.29	0.35
	12.028	15.954	18.837	18.363	18.140	16.182	16.361	16.124
	0.731	0.757	1.622	1.552	1.509	1.022	1.043	1.039
Spl	1.00	0.60	0.46	**0.24**	0.36	0.76	0.79	0.74
	9.460	9.480	9.492	9.458	9.460	9.557	9.501	**9.426**
	0.545	0.562	**0.530**	0.544	0.546	0.590	0.582	0.571
Syn	1.00	0.64	0.47	**0.28**	**0.28**	0.78	0.90	1.13
	4.147	4.293	4.787	4.600	4.600	4.133	4.867	**4.107**
	2.305	2.310	2.390	2.385	2.392	**2.270**	2.531	2.311
Thy	**1.00**	6.86	1.53	1.29	1.41	6.82	6.63	5.58
	22.455	22.446	22.493	22.524	22.477	22.458	22.486	**22.385**
	1.250	0.953	**0.930**	1.035	1.044	0.993	0.955	0.955
Tita	1.00	0.12	0.11	**0.10**	0.11	0.12	0.11	0.14
	2.387	**2.347**	2.352	2.350	**2.347**	2.349	2.348	2.362
	0.121	0.119	0.116	**0.105**	0.106	0.115	0.116	0.122
TwN	1.00	0.04	**0.03**	**0.03**	**0.03**	0.04	0.05	0.04
	10.274	9.732	9.637	**9.486**	9.593	9.782	9.785	9.749
	0.407	**0.390**	0.509	0.458	0.505	0.438	0.430	0.422
WaF	1.00	0.78	0.35	**0.19**	0.29	0.82	0.80	0.75

Comparing the ℓ_1-regularized kernel IRLS classifier with and without bias, it can be noted that excluding bias leads to a lower running time and similar performance of the classifier. For classifier with bias the ASIGL approximation is the best, whereas for classifier without bias the ASQR approximation is the best. Taking into account both the average and standard deviation of the generalization error the IRLS classifier leads to better results for 4 databases in both cases, ie. with and without bias. However, the LSVM classifier leads to better

results for 4 databases when bias is used in the IRLS classifier and for 2 databases when bias is not used in the IRLS classifier. For the ℓ_1-regularized kernel IRLS classifier better results are obtained for its version without bias. Comparing the ℓ_1- and ℓ_2-regularized kernel IRLS classifier, it must be noted that ℓ_2-regularization leads to better performance and similar running time. In conclusion, the LSVM gives best performance in case of the ℓ_1-regularized kernel IRLS classifier.

6. Conclusions

In the paper it is shown that various approximations of misclassification error can be written as iteratively reweighted least squares (IRLS) criterion function. The IRLS procedure can be used for a better than squared approximation of misclassification error as well as may be used for relaxation of a "force" to the separating hyperplane created by properly classified data. The conjugate gradient algorithm is computationally effective method for minimization of proposed criterion function. Furthermore, ℓ_2- and ℓ_1-regularized kernel version of the classifier is introduced. For ℓ_1-regularized criterion function the gradient projection is used to optimization. An extensive experimental analysis on 14 benchmark datasets showing that proposed classifier design methods are a good alternative, in terms of both generalization performance and computational cost, to the-state-of-the-art algorithm as Lagrangian support vector machine. For the linear classification the best performance is observed on the ALOG approximation of misclassification error. For the kernel version of classifier the best are the ALOG and ALIN approximations for classifier with and without bias, respectively. Excluding bias leads to a lower running time and a bit worse performance of the classifier in case of ℓ_2-regularization. In case of the ℓ_1-regularized kernel IRLS classifier better results are obtained for its version without bias. Comparing the ℓ_1- and ℓ_2-regularized kernel IRLS classifier, it must be noted that ℓ_2 regularization leads to a better performance and a similar running time.

REFERENCES

[1] R.O. Duda and P.E. Hart, *Pattern Classification and Scene Analysis*, John Wiley&Sons, New York, 1973.

[2] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, John Wiley&Sons, New York, 2001.

[3] B.D. Ripley, *Pattern Recognition and Neural Networks*, Cambridge University Press, Cambridge, 1996.

[4] J.T. Tou and R.C. Gonzalez, *Pattern Recognition Principles*, Addison-Wesley, London, 1974.

[5] A. Webb, *Statistical Pattern Recognition*, Arnold, London, 1999.

[6] B. Schölkopf and A.J. Smola, *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*, The MIT Press, London, 2002.

[7] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer, New York, 1995.

[8] V. Vapnik, *Statistical Learning Theory*, Wiley, New York, 1998.

[9] O.L. Mangasarian and D.R. Musicant, "Lagrangian support vector machines", *J. Mach. Learn. Res.* 1 (1), 161–177 (2001).

[10] J.A.K. Suykens and J. Vandewalle, "Least squares support vector machine classifiers", *Neur. Proc. Lett.* 9 (3), 293–300 (1999).

[11] I.W. Tsang, J.T. Kwok, and P.-M. Cheung, "Core vector machines: Fast SVM training on very large data sets", *J. Mach. Learn. Res.* 6 (1), 363–392 (2005).

[12] I.W. Tsang, J.T. Kwok, and J.M. Zurada, "Generalized core vector machines", *IEEE Trans. Neur. Net.* 17 (5), 1126–1140 (2006).

[13] I.W. Tsang, A. Kocsor, and J.T. Kwok, "Large-scale maximum margin discriminant analysis using core vector machines", *IEEE Trans. Neur. Net.* 19 (4), 610–623 (2008).

[14] S. Mika, G. Rätsch, J. Weston, S. Schölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels", in: *Neur. Net. Sig. Proc. IX*, pp. 41–48, eds. Y-H. Hu, J. Larsen, E. Wilson, S. Douglas, IEEE Press, New York, 1999.

[15] W. Zheng and L. Zou, "Foley-Sammon optimal discriminant vectors using kernel approach", *IEEE Trans. Neur. Net.* 16 (1), 1–9 (2005).

[16] Y. Freund and R.E. Schapire, "Large margin classification using the perceptron algorithm", *Mach. Learn.* 37 (1), 277–296 (1999).

[17] J.-H. Chen and C.-S. Chen, "Fuzzy kernel perceptron", *IEEE Trans. Neu. Net.* 13 (6), 1364–1373 (2002).

[18] E. Pękalska, P. Paclik, and R.P.W. Duin, "A generalized kernel approach to dissimilarity-based classification", *J. Mach. Learn. Res.* 2 (1), 175–211 (2001).

[19] Y.-C. Ho and R.L. Kashyap, "An algorithm for linear inequalities and its applications", *IEEE Trans. Elec. Comp.* 14 (5), 683–688 (1965).

[20] Y.-C. Ho and R.L. Kashyap, "A class of iterative procedures for linear inequalities", *J.SIAM Control.* 4 (2), 112–115 (1966).

[21] M.H. Hassoun and J. Song, "Adaptive Ho-Kashyap rules for perceptron", *IEEE Trans. Neu. Net.* 3 (1), 51–61 (1992).

[22] J.M. Łęski, "Ho-Kashyap classifier with generalization control", *Pattern Recognition Letters* 24 (2), 2281–2290 (2003).

[23] J.M. Łęski, "Kernel Ho-Kashyap classifier with generalization control", *Int. J. App. Math. Comp. Sci.* 14 (1), 53–62 (2004).

[24] Z. Wang, S. Chen, J. Liu, and D. Zhang, "Pattern representation in feature extraction and classifier design: matrix versus vector", *IEEE Trans. Neu. Net.* 19 (5), 758–769 (2008).

[25] Z. Wang, S. Chen, and T. Sun, "MultiK-MHKS: a novel multiple kernel learning algorithm", *IEEE Trans. Patt. Ana. Mach. Intel.* 30 (2), 348–353 (2008).

[26] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multiclass support vector machines", *IEEE Trans. Neu. Net.* 13 (2), 415–425 (2002).

[27] P.J. Huber, *Robust Statistics*, Wiley, New York, 1981.

[28] S. Haykin, *Neural Networks: a Comprehensive Foundation*, Prentice Hall, Upper Saddle River, 1999.

[29] T. Blumensath and M.E. Davies, "Gradient pursuit", *IEEE Trans. Sig. Proc.* 56 (6), 2370–2382 (2008).

[30] M.A.T. Figueiredo, R.D. Nowak, and S.J. Wright, "Gradient projection for sparse reconstruction: application to compressed sensing and other inverse problems", *IEEE J. Select. Top. Sig. Proc.* 1 (4), 586–597 (2007).

[31] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani, "Least angle regression", *The Annals of Statistics* 32 (2), 407–451 (2004).

[32] R. Tibshirani, "Regression shrinkage and selection via the lasso", *J.R. Statist. Soc.* B 58 (1), 267–288 (1996).

[33] S.-J. Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky, "An interior-point method for large-scale ℓ_1-regularized least squares", *IEEE J. Select. Top. Sig. Proc.* 1 (4), 606–617 (2007).

[34] V. Ruggiero and L. Zanni, "A modified projection algorithm for large strictly convex quadratic programs", *J. Optim. Theory Appl.* 104 (2), 281–299 (2000).

[35] T. Serafini, G. Zanghirati, and L. Zanni "Gradient projection methods for quadratic programs and applications in training support vector machines", *Optim. Meth. Soft.* 20 (2), 353–378 (2004).