ISSN 0006-3509, Biophysics, 2015, Vol. 60, No. 1, pp. 13–24. © Pleiades Publishing, Inc., 2015.
Original Russian Text © N.O. Kadyrova, L.V. Pavlova, 2015, published in Biofizika, 2015, Vol. 60, No. 1, pp. 18–31.

MOLECULAR BIOPHYSICS

Comparative Efficiency of Algorithms Based on Support Vector Machines for Binary Classification

N. O. Kadyrova and L. V. Pavlova

Institute of Applied Mathematics and Mechanics, St. Petersburg State Polytechnical University, ul. Politekhnicheskaya 29, St. Petersburg, 195251 Russia

e-mail: natalia.kadyrova@gmail.com

Received June 9, 2014

Abstract—Methods of construction of support vector machines (SVMs) require no additional a priori information and allow large volumes of multidimensional data to be processed, which is especially important for solving various problems in computational biology. The main algorithms of SVM construction for binary classification are reviewed. The issue of the quality of the SVM learning algorithms is considered, and a description of proposed algorithms is given that is sufficient for their practical implementation. Comparative analysis of the efficiency of support vector classifiers is presented.

Keywords: binary classification, support vector machine, kernel functions, SVM algorithms, comparative efficiency of support vector classifiers

DOI: 10.1134/S0006350915010145

INTRODUCTION

The main principles and concepts of the modern approach to solving tasks of binary classification and regression restoration, which is based on using support vector machines (SVMs), have been considered in our previous work [1]. SVM methods, which are part of the technology called data mining, are remarkable in that they require almost no additional a priori information and are capable of processing large databases of high dimensionality.

A new approach to inductive learning on finite sample sets is based on the principle of structural risk minimization and leads to the problem of regularized risk minimization [2]. Some SVM-based algorithms—e.g., the Naive Online Risk Minimization Algorithm (NORMA)—directly solve this problem by using kernel machines.

The standard formulation of the problem of binary classification allows the initial task of minimization of the upper bound of expected risk to be reduced to a problem of quadratic programming with equality type constraints for dual variables, which fully determine the decision rule [2]. A direct solution of the dual problem of quadratic programming is usually impossible because of its vast dimensionality. The main approaches to constructing an indirect solution are based on the property of sparseness that is inherent in SVMs. Some SVM algorithms—such as SVM-Light and Sequential Minimal Optimization (SMO)—are based on decomposition the initial problem of quadratic programming into a series of subproblems of significantly smaller dimensionality.

This work is devoted to consideration of the main SVM-algorithms construction for binary classification tasks and comparative analysis of their efficiency.

QUALITY OF LEARNING ALGORITMS

Evaluation of the generalization performance of various learning algorithms employing some selected training sets is an important stage in the SVM approach. Once a training set is given, it is important to determine how well a particular learning algorithm will operate with new data—i.e., to evaluate its generalization performance. When this information is available, it is possible to proceed to choosing the algorithm, model, and parameters of learning.

It is common practice to use methods of the generalization error estimation based on the k-fold cross-validation (CV) procedure. However, the theoretical aspects of this approach are still not sufficiently studied. The most popular CV method is based on the so-called leave-one-out (loo) procedure, which provides the corresponding error (loo-error) upon the l-fold CV, where l is the size of the training set. The loo-error is an important statistical characteristic of the quality of learning algorithms. Data that can be used to justify the use of loo-errors in machine learning are presented in [3].

Let X be the space of input objects $\mathbf{x}$ with a fixed (unknown) distribution of probabilities $P(\mathbf{x})$; Y is the space of output objects y with a fixed (unknown) distribution of probabilities $P(y/\mathbf{x})$; D is the training set of size l, i.e., a set of independent observations (samples)

$(\mathbf{x}_1, y_1)$, ..., $(\mathbf{x}_l, y_l)$ distributed according to the law $P(\mathbf{x}, y) = P(\mathbf{x}) \times P(y/\mathbf{x})$; D^i is the training set of size $l-1$ obtained by leaving one (ith) sample $(\mathbf{x}_i, y_i)$ out of D; and f_D: $X \to R$ is a solution obtained using a given learning algorithm on training set D. The loss function L: $R \times Y \to R$, $L = L(f(\mathbf{x}), y) \equiv L(f, (\mathbf{x}, y))$, rates the deviations of estimates $f(\mathbf{x})$ from observables y.

The generalization error of the given learning algorithm with respect to loss function L is characterized by the expected risk defined as

$$R[f_D] = \int_{X \times Y} L(f_D, (\mathbf{x}, y))dP(\mathbf{x}, y).$$

An empirical error of the given learning algorithm with respect to loss function L can be represented either by the empirical risk

$$R_{\text{emp}}[f_D] = \frac{1}{l}\sum_{i=1}^{l} L(f_D, (\mathbf{x}_i, y_i)),$$

or by the loo-error

$$R_{\text{loo}}[f_D] = \frac{1}{l}\sum_{i=1}^{l} L(f_{D^i}, (\mathbf{x}_i, y_i)).$$

Note that, in the latter case, sample $(\mathbf{x}_i, y_i)$ is not used in the learning process and only employed for testing a solution obtained using training set D^i of size $l-1$.

In contrast to the empirical risk, the loo-error obtained using training sets of size l gives an almost unbiased estimation of the generalization error upon learning on a set of $l-1$ samples. Since the computation of loo-error is time-consuming, many researchers have attempted to determine the simple (in respect of calculations) bounds of this value.

EVALUATION OF THE QUALITY OF SVM-BASED CLASSIFICATION

One of the characteristics of quality of a classification algorithm is its ability to correctly classify new samples. It is necessary that this ability be adequately characterized without large volume of computations.

The generalization performance is usually characterized using CV methods—in particular, loo procedures. According to this, one sample is removed from the training set, the remaining $l-1$ samples are used for algorithm learning, and the result is tested on the removed sample. The number of errors divided by the training set size represents the loo-error of generalization. The upper bounds of the loo-error of classification, which assume the construction of only one SVM based on the initial training set of size l, have been given in [3, 4].

Joachims [4] proposed the following upper bound for loo-error of generalization in the case of a standard support-vector (SV) classifier:

$$J\_err^l = \frac{d}{l},$$
$$d = \left|\{i: (\rho\alpha_i R_\Delta^2 + \xi_i) \geq 1\}\right|, \quad (1)$$

where $\rho = 2$; α_i is Lagrange's multiplier, ξ_i is the slack variable corresponding to the ith sample, R_Δ^2 is the upper bound of difference $K(\mathbf{x}, \mathbf{x})-K(\mathbf{x}, \mathbf{x}')$ for all permissible input vectors $\mathbf{x}$ and $\mathbf{x}'$, and K is the kernel function (for linear and Gaussian kernels, $R_\Delta^2 = 1$). It has been proven that estimation (1) exceeds the true fraction of classification errors [4].

Elisseeff and Pontil [3] showed that, for binary classification using the Heaviside loss function $\delta(-y \cdot f(\mathbf{x}))$, the loo-error of a kernel machine minimizing the regularized risk based on training set D is bounded from above as

$$R_{\text{loo}}[f_D] \equiv \frac{1}{l}\sum_{i=1}^{l} \delta(-y_i f_{D^i}(\mathbf{x}_i))$$
$$\leq \frac{1}{l}\sum_{i=1}^{l} \delta(|\alpha_i| K(\mathbf{x}_i, \mathbf{x}_i) - y_i f_D(\mathbf{x}_i)).$$

Receiver Operator Characteristic (ROC) analysis offers a graphical method for representation of the results of binary classification during machine learning, which reflects the quality of a given classifier of a comparative efficiency of several classifiers. The ROC curves are very useful in the case of an asymmetric distribution of input vectors and unequal ratings of classification errors. These characteristics are especially important in the case of unbalanced classes.

In order to determine the quality of classification separately for each class (one class with positive samples and one with negative samples), it is possible to use the characteristics of sensitivity and specificity or recall and precision (where recall is identical to sensitivity). The sensitivity characterizes the efficiency of binary classifier operation and determines the fraction of true positive observations relative to the total number of actually positive observations. A classifier possessing high sensitivity ensures greater probability of the correct recognition of positive samples. The specificity characterizes the accuracy of binary classifier operation and is defined as the ratio of true negative observation to the total number of actually negative observations. A classifier possessing high specificity ensures greater probability of correct recognition of negative samples.

The value of precision determines the fraction of true positive observations relative to the total number of observations classified by the model as positive. For example, for the classification of patients into ill (positive samples) and healthy (negative samples), a high-sensitivity classifier will most reliably prevent ill

patients from being missed, while a high-specificity classifier will most reliably detect actually ill patients.

A practical guide on the correct use of ROC curves and the main characteristics of classification quality is given in [5]. The interrelation between ROC curves and precision–recall curves has been considered in [6].

DESCRIPTION OF DECISION ALGORITHMS FOR BINARY CLASSIFICATION

Let us consider the formulation of the dual task of SVM learning for binary classification:

$$\text{Find } \min_{\alpha} W(\boldsymbol{\alpha}) = -\sum_{i=1}^{l}\alpha_i + \frac{1}{2}\sum_{i,j=1}^{l}\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$
$$\text{subject to: } 0 \le \alpha_i \le C, \quad \sum_{i=1}^{l}\alpha_i y_i = 0. \tag{2}$$

Here, $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_l, y_l) \in R^n \times \{-1; +1\}$ is the training set of size l; $\alpha_i \ge 0$ are Lagrange's multipliers; $C > 0$ is the regularization parameter that controls the margin width; and $K(\mathbf{x}, \mathbf{x}')$ is the kernel function obeying Mercer's conditions [2].

The optimum decision function takes the form of a linear combination of kernel values on training set vectors:

$$f(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b.$$

Let $\mathbf{Q}$ be a symmetric positive determined an $l \times l$ matrix with elements $Q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$. The procedure of expansion [7] splits quadratic programming problem (2) into several problems of much smaller dimensionality q ($q \ll l$), thus eliminating problems related to the computation and full-volume storage of matrix $\mathbf{Q}$.

SVM-LIGHT ALGORITHM

The decomposition procedure has been effectively implemented in the SVM-Light algorithm proposed by Joachims [8]. According to this, only some of Lagrange's multipliers (the working set) are optimized at each step of the iterative learning process, while the other multipliers are fixed. Elements of the working set are the "worst" violators of the Karush–Kuhn–Tucker conditions, which are selected using a specific heuristic procedure. The size of the working set is not changed in all iterations. When dual variables for the working set are no longer available, the quadratic programming problem turns out to be solved.

Optimum conditions. Let λ^{eq} denote Lagrange's multiplier for equality type constraints in quadratic programming problem (2), while λ^{lo} and λ^{up} are Lagrange's multipliers for the lower and upper bounds, respectively, of the components of vector α. Then, the Lagrangian for dual problem (2) can be written as

$$L_W = -\sum_i \alpha_i + \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j Q_{ij} + \lambda^{eq}\sum_i \alpha_i y_i - \sum_i \lambda_i^{lo}\alpha_i + \sum_i \lambda_i^{up}(\alpha_i - C).$$

The conditions of optimum for this problem are as follows:

$$\forall i \in \{1, \ldots l\}: \; g_i(\boldsymbol{\alpha}) + (\lambda^{eq} y_i - \lambda_i^{lo} + \lambda_i^{up}) = 0, \quad \lambda_i^{lo}(-\alpha_i) = 0, \quad \lambda_i^{lo}(\alpha_i - C) = 0 \tag{3}$$

and

$$\boldsymbol{\lambda}^{lo} \ge 0, \quad \boldsymbol{\lambda}^{up} \ge 0, \quad \boldsymbol{\alpha}^T y = 0, \quad 0 \le \boldsymbol{\alpha} \le C \cdot 1, \tag{4}$$

where $g_i(\boldsymbol{\alpha})$ is the ith element of the gradient of target function $W(\alpha)$ with respect to α:

$$\mathbf{g}(\boldsymbol{\alpha}) = -1 + \mathbf{Q}\boldsymbol{\alpha}. \tag{5}$$

Decomposition of problem (2). Convergence of the expansion procedure is guaranteed, provided that the working set obeys certain minimum requirements [8]. Let us divide variables α_i of the initial problem (2) into two groups: free variables with indices constituting set B and fixed variables with indices constituting set N, $N = \{1, 2, \ldots, l\} \backslash B$. Let us decompose problem (2) by separating α, $\mathbf{y}$ and $\mathbf{Q}$ sets into parts corresponding to sets B and N:

$$\boldsymbol{\alpha} = \begin{vmatrix} \boldsymbol{\alpha}_B \\ \boldsymbol{\alpha}_N \end{vmatrix}, \quad \mathbf{y} = \begin{vmatrix} \mathbf{y}_B \\ \mathbf{y}_N \end{vmatrix}, \quad \mathbf{Q} = \begin{vmatrix} \mathbf{Q}_{BB} & \mathbf{Q}_{BN} \\ \mathbf{Q}_{NB} & \mathbf{Q}_{NN} \end{vmatrix}. \tag{6}$$

Taking into account the symmetry of matrix $\mathbf{Q}$ and excluding constant terms from the target function, we arrive at the following optimization problem:

$$\text{Find } \min_{\alpha_B} W(\boldsymbol{\alpha}_B) = -\boldsymbol{\alpha}_B^T(1 - \mathbf{Q}_{BN}\boldsymbol{\alpha}_N) + \frac{1}{2}\boldsymbol{\alpha}_B^T\mathbf{Q}_{BB}\boldsymbol{\alpha}_B \tag{7}$$

$$\text{subject to: } \boldsymbol{\alpha}_B^T y_B + \boldsymbol{\alpha}_N^T y_N = 0, \tag{8}$$

$$0 \le \boldsymbol{\alpha}_B \le C \cdot 1. \tag{9}$$

Selecting a working set. The efficiency of decomposition algorithm is directly related to the procedure of selecting a working set. This construction is based on Zoutendijk's method, according to which a steepest descent direction vector $\mathbf{d}$ is determined that has only q nonzero elements. Variables corresponding to these elements constitute the working set. For select-

ing this set on the tth iteration step, it is necessary to solve the following problem:

$$\text{Find } \min_{d} V(\mathbf{d}) = \mathbf{g}(\boldsymbol{\alpha}^{(t)})^T\mathbf{d}$$

$$\text{subject to: } \mathbf{y}^T\mathbf{d} = 0,\ -1 \le \mathbf{d} \le 1,$$

$$|\{d_i: d_i \ne 0\}| = q,$$

$$d_i \ge 0, \quad i: \alpha_i = 0, \tag{10a}$$

$$d_i \le 0, \quad i: \alpha_i = C. \tag{10b}$$

Let us define vector $\boldsymbol{\omega}$ as $\omega_i = y_i g_i(\alpha^{(t)})$ and arrange elements α_i in the order of decreasing ω_i. For an even q number, let us select $q/2$ elements from the top of the list, for which either $0 < \alpha_i^{(t)} < C$ or $d_i = -y_i$ satisfies conditions (10a) or (10b). In the same way, we select $q/2$ elements from the bottom of the list, for which either $0 < \alpha_i^{(t)} < C$ or $d_i = y_i$ satisfies conditions (10a) or (10b). The selected q elements constitute the working set.

Structure of SVM-Light algorithm. The working set size q is put such that $q \ll l$. If the optimum conditions are not satisfied,

—q variables for the working set are selected as described above and the remaining l–q elements are fixed; and

—problem (2) is decomposed and the quadratic programming subproblem (7) of dimensionality q is solved under constraints (8) and (9).

Otherwise, the process is terminated because the optimum solution is found.

Effective implementation of SVM-Light algorithm. The validity of optimum conditions (3) and (4) is conveniently checked using the following relations:

$$\lambda^{\text{eq}} - \varepsilon \le y_i - \sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) \le \lambda^{\text{eq}} + \varepsilon, \quad i: 0 < \alpha_i < C,$$

$$y_i\left(\sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) + \lambda^{\text{eq}}\right) \ge 1 - \varepsilon, \quad i: \alpha_i = 0, \tag{11}$$

$$y_i\left(\sum_{j=1}^{l} \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) + \lambda^{\text{eq}}\right) \le 1 + \varepsilon, \quad i: \alpha_i = C,$$

$$\sum_{j=1}^{l} \alpha_j y_j = 0,$$

which are equivalent to relations (3) and (4) for $\varepsilon = 0$. In solving most problems, an acceptable value is $\varepsilon = 1\text{e} - 3$.

Let us define vector $\mathbf{s}^{(t)}$ on the tth iteration step as

$$s_i^{(t)} = \sum_{j=1}^{l} \alpha_j^{(t)} y_j K(\mathbf{x}_i, \mathbf{x}_j), \quad i \in \{1, 2, \ldots l\}.$$

Then, when vector $\boldsymbol{\alpha}$ changes from $\boldsymbol{\alpha}^{(t-1)}$ to $\boldsymbol{\alpha}^{(t)}$ on the tth iteration step, the sums $s_i^{(t)}$ in relations (11) can be rapidly recalculated as

$$s_i^{(t)} = s_i^{(t-1)} + \sum_{j \in B} (\alpha_j^{(t)} - \alpha_j^{(t-1)}) y_j K(\mathbf{x}_i, \mathbf{x}_j),$$

and, hence, the value of the gradient $\mathbf{g}(\boldsymbol{\alpha}^{(t)})$ (see formula (5)).

In the course of iterations, it is possible to reduce the dimensionality of initial problem (2) (i.e., produce its "shrinking") via removal of α_i variables that will probably be equal 0 or C. Since these values are fixed and, hence, neither gradient $\mathbf{g}(\boldsymbol{\alpha}^{(t)})$ (5) nor the optimum conditions for these quantities are recalculated, which significantly decreases the volume of kernel computations. Detailed description of the "shrinking" procedure is presented in [8].

SEQUENTIAL MINIMAL OPTIMIZATION (SMO) ALGORITHM

The SMO algorithm that has been proposed by Platt [9, 10] implements an extremal form of expansion for problem (2) with the minimum possible working set size of 2, since the vector of dual variables $\boldsymbol{\alpha}$ must obey a linear equality type constraints. A very important advantage of SMO over other SVM algorithms consists in that a solution of the problem with two variables is found analytically.

Computational problems that can arise during the implementation of the SMO algorithm are only related to the kernel matrix $\{K(\mathbf{x}_i, \mathbf{x}_j)\}$. The SMO procedure is especially effective in the case of sparce data (more than 80% zero values of components in input vectors $\mathbf{x}$) and rapidly converges for linear SVMs.

Here we consider the second modification of the SMO algorithm [11], which proved to be more effective than the original Platt variant [9]. An investigation of the influence of the working set selection on the efficiency of SMO classification was reported in [12]. Convergence of the SMO algorithm in classification problems was proved in [13].

At each step, the SMO algorithm selects by certain means two violators $\{\alpha_i, \alpha_j\}$ of the optimum conditions (i.e., working set), determines the optimum values of these variables, and accordingly improves the SVM. When no new variables for the working set are available, the initial problem (2) turns out to be solved.

Optimum conditions and SMO stopping criterion. In order to establish a criterion of stopping for the SMO algorithm in solving dual problem (2), let us consider the conditions of optimum for this problem. The corresponding Lagrangian can be written as

$$L_W = \frac{1}{2}w(\boldsymbol{\alpha})w(\boldsymbol{\alpha}) - \sum_{i=1}^{l}\alpha_i - \sum_{i=1}^{l}\delta_i\alpha_i + \sum_{i=1}^{l}\mu_i(\alpha_i - C) - \beta\sum_{i=1}^{l}\alpha_i y_i,$$

where $w(\boldsymbol{\alpha}) = \sum_{i=1}^{l}\alpha_i y_i \mathbf{x}_i$. Let us

$$F_i = w(\boldsymbol{\alpha})\mathbf{x}_i - y_i = \sum_j \alpha_j y_j K(\mathbf{x}_i, \mathbf{x}_j) - y_i. \quad (12)$$

The Karush–Kuhn–Tucker conditions for any i can be expressed as follows:

$$\frac{\partial L_W}{\partial \alpha_i} = (F_i - \beta)y_i - \delta_i + \mu_i = 0,$$

$$\delta_i \geq 0, \quad \delta_i\alpha_i = 0, \quad \mu_i \geq 0, \quad \mu_i(\alpha_i - C) = 0.$$

Consider three possible variants of variables α_i. In each case, the Karush–Kuhn–Tucker conditions acquire a simple form:

$$\text{(i) } \alpha_i = 0, \text{ then } \delta_i \geq 0,\ \mu_i = 0 \Leftrightarrow (F_i - \beta)y_i \geq 0,$$

$$\text{(ii) } 0 < \alpha_i < C, \text{ then } \delta_i = 0,\ \mu_i = 0 \Leftrightarrow (F_i - \beta)y_i = 0, \quad (13)$$

$$\text{(iii) } \alpha_i = C, \text{ then } \delta_i = 0,\ \mu_i \geq 0 \Leftrightarrow (F_i - \beta)y_i \leq 0.$$

Let us define the following sets of indices for a fixed vector $\boldsymbol{\alpha}$:

$$I_0 = \{i: 0 < \alpha_i < C\};$$
$$I_1 = \{i: y_i = 1,\ \alpha_i = 0\};$$
$$I_2 = \{i: y_i = -1,\ \alpha_i = C\};$$
$$I_3 = \{i: y_i = 1,\ \alpha_i = C\};$$
$$I_4 = \{i: y_i = -1,\ \alpha_i = 0\};$$

and introduce the quantities

$$b_{\text{up}} = \min\{F_i: i \in I_0 \cup I_1 \cup I_2\}, \quad (14)$$

$$b_{\text{low}} = \max\{F_i: i \in I_0 \cup I_3 \cup I_4\}. \quad (15)$$

Then, the conditions of optimum are satisfied for vector $\boldsymbol{\alpha}$ such that

$$b_{\text{low}} \leq b_{\text{up}}. \quad (16)$$

Let us introduce the positive parameter of tolerance τ and use it to write the approximate optimum conditions (13) in the following form that will be used as the criterion of SMO algorithm stopping:

$$(F_i - \beta)y_i \geq -\tau, \text{ for } \alpha_i = 0,$$
$$|F_i - \beta| \leq \tau, \text{ for } 0 < \alpha_i < C, \quad (17)$$
$$(F_i - \beta)y_i \leq \tau, \text{ for } \alpha_i = C.$$

For a given vector $\boldsymbol{\alpha}$, a pair of indices $\{i, j\}$ is classified as violating the Karush–Kuhn–Tucker conditions in one of the following cases:

$$i \in I_0 \cup I_3 \cup I_4 \text{ and } j \in I_0 \cup I_1 \cup I_2 \text{ and } F_i > F_j, \quad (18a)$$

$$i \in I_0 \cup I_1 \cup I_2 \text{ and } j \in I_0 \cup I_3 \cup I_4 \text{ and } F_i < F_j. \quad (18b)$$

Note that the optimum conditions for $\boldsymbol{\alpha}$ are satisfied when and only when no one pair of indices $\{i, j\}$ exists that corresponds to violation for this $\boldsymbol{\alpha}$.

In practical calculations, it is convenient to use tolerance parameter τ. Then, condition (16) takes the following form:

$$b_{\text{low}} \leq b_{\text{up}} + 2\tau. \quad (19)$$

For detecting violations, conditions (18a) and (18b) are replaced by

$$i \in I_0 \cup I_3 \cup I_4 \text{ and } j \in I_0 \cup I_1 \cup I_2 \text{ and } F_i > F_j + 2\tau, \quad (20a)$$

$$i \in I_0 \cup I_1 \cup I_2 \text{ and } j \in I_0 \cup I_3 \cup I_4 \text{ and } F_i < F_j - 2\tau. \quad (20b)$$

The SMO algorithm employs condition (19) or (20) to check the optimum for current vector $\boldsymbol{\alpha}$. Note that it is not necessary to know the current value of b. In addition, with this method of checking for validity of the Karush–Kuhn–Tucker conditions, the second term is automatically determined provided that the first term of the working set is given.

Working set selection. At each iteration step, the SMO algorithm selects a pair of variables of dual problem (2) for their joint optimization:

$$\{\alpha_{i_1}, \alpha_{i_2}\}.$$

The initial approximation of unknown vector $\boldsymbol{\alpha}$ usually represents a zero vector. The initial bound quantities are $b_{\text{up}} = -1$ and $b_{\text{low}} = 1$, i_up is set equal to any index from the first class, and i_low is selected to be any index from the second class. First, all members of the training set are tried to reveal violators of the optimum conditions. During the trial cycle involving only indices of set I_0, the SMO algorithm always works with the worst violating pair—i.e., that selected from set $B = \{i_1, i_2\} \subset \{1, 2, ..., l\}$ as follows: $i_2 = i\_\text{low}$ and $i_1 = i\_\text{up}$, where i_low and i_up are found from the conditions $F_{i\ \text{low}} = b_{\text{low}}$ and $F_{i\ \text{up}} = b_{\text{up}}$. Thus, the first and second members of the working set are selected simultaneously, in contrast to the original SMO variant [10], where the second member was chosen using certain heuristic procedure under assumption that the first member is already known.

After a successful step of the algorithm with indices $\{i_1, i_2\}$, a new set I^* is formed such that $I^* = I_0 \cup \{i_1, i_2\}$. Note that both sets $I^* \cap \{I_0 \cup I_1 \cup I_2\}$ and $I^* \cap \{I_0 \cup I_3 \cup I_4\}$ are not empty. Therefore, it is possible to partly calculate b_{low} and b_{up} using set $I^* = I_0 \cup \{i_1, i_2\}$. How-

ever, we abstained from this attractive possibility because it leads to an "underlearned" machine. Indeed, the optimum hyperplane is constructed correctly, but the margin is too wide. In the proposed implementation of the SMO algorithm, the approach involving $I^* = I_0 \cup \{i_1, i_2\}$ set is not employed and the values of b_{low} and b_{up} at each iteration step are determined according to rules (14) and (15), respectively.

Upon establishing the optimum conditions according to relation (19) on set I_0, the SMO algorithm returns to trial over the entire (initial) training set for checking the optimum with respect to all indices. Since (b_{low}, i_low) and (b_{up}, i_up) have been calculated only on set I_0, these quantities are updated for each current index by calculating F_i and checking for the optimum using current (b_{low}, i_low) according to condition (20). If this trial cycle with running i reveals no violations, it is concluded that the optimum conditions are satisfied for all elements of vector $\boldsymbol{\alpha}$, that is, the optimum solution is found.

Solution of the quadratic programming problem for a selected working set. The working set consists of a pair of variables $\{\alpha_i, \alpha_j\}$. A constraint of the equality type can be used for excluding one of them. For the convenience of presentation, all quantities related to the first and second variable will be distinguished by indices 1 and 2, respectively.

Then, if the labels of classes are different ($y_1 \neq y_2$), then $\alpha_1 - \alpha_2 = \text{const}$, while identical labels ($y_1 = y_2$) imply that $\alpha_1 + \alpha_2 = \text{const}$. Thus, at each step of the algorithm, we obtain a quadratic programming problem for only one variable (for certainty, α_2). This problem has the following analytical solution (for a detailed derivation, see [10, Appendix]):

$$\alpha_2^* = \alpha_2 + \frac{y_2(F_2 - F_1)}{\eta},$$

where $\eta = K(\mathbf{x}_1, \mathbf{x}_1) + K(\mathbf{x}_2, \mathbf{x}_2) - 2K(\mathbf{x}_1, \mathbf{x}_2)$ is the second derivative of target function (7). Taking into account constraints of the inequality type and assuming that $\eta > 0$, the new value of variable α_2 can be written as

$$\alpha_2^{\text{new}} = \begin{cases} H, & \text{for } \alpha_2^* \geq H \\ \alpha_2^*, & \text{for } L < \alpha_2^* < H, \\ L, & \text{for } \alpha_2^* \leq L \end{cases}$$

where L and H are the bounds for α_2, which are defined as follows.

$$(y_1 \neq y_2)\colon L = \max(0, \alpha_1 - \alpha_2),$$
$$H = \min(C, C + \alpha_1 - \alpha_2);$$
$$(y_1 = y_2)\colon L = \max(0, \alpha_1 + \alpha_2 - C),$$
$$H = \min(C, \alpha_1 + \alpha_2).$$

If the training set contains at least two identical vectors $\mathbf{x}$, then η can be zero. The SMO algorithm will also operate in this case by calculating the values W_L and W_H of target function (7) on the ends of a straight segment determined by the linear relation between α_1 and α_2:

$$f_1 = y_1F_1 - \alpha_1K(\mathbf{x}_1, \mathbf{x}_1) - s\alpha_2K(\mathbf{x}_1, \mathbf{x}_2),$$

$$f_2 = y_2F_2 - s\alpha_1K(\mathbf{x}_1, \mathbf{x}_2) - \alpha_2K(\mathbf{x}_2, \mathbf{x}_2),$$

$$L_1 = \alpha_1 + S(\alpha_2 - L),$$

$$H_1 = \alpha_1 + S(\alpha_2 - H),$$

$$W_L = L_1f_1 + Lf_2 + \frac{1}{2}L_1^2K(\mathbf{x}_1, \mathbf{x}_1)$$

$$+ \frac{1}{2}L^2K(\mathbf{x}_2, \mathbf{x}_2) + sLL_1K(\mathbf{x}_1, \mathbf{x}_2),$$

$$W_H = H_1f_1 + Hf_2 + \frac{1}{2}H_1^2K(\mathbf{x}_1, \mathbf{x}_1)$$

$$+ \frac{1}{2}H^2K(\mathbf{x}_2, \mathbf{x}_2) + sHH_1K(\mathbf{x}_1, \mathbf{x}_2),$$

where $s = y_1y_2$.

If $W_L < W_H - \tau$, then α_2^{new} takes the value of L. For $W_L > W_H + \tau$; α_2^{new} takes the value of H. If the target functions values W_L and W_H differ by less than τ, the current value of α_2 remains unchanged and a new set is selected. The corresponding value of α_1^{new} is calculated using the value of α_2^{new} obtained during optimization: $\alpha_1^{\text{new}} = \alpha_1 + s(\alpha_2 - \alpha_2^{\text{new}})$.

Brief description of the SMO algorithm. The SMO algorithm is a process in which every operation consists of two steps: (i) choice of a working set $\{\alpha_i, \alpha_j\}$ of size 2 (heuristic method) and (ii) solution of a quadratic programming problem for the selected working set (analytical method). When no new variables for the working set are available, the initial problem (2) turns out to be solved.

The main steps of the SMO algorithm are as follows.

1. Take some vector $\boldsymbol{\alpha}^1$ (usually, zero vector) as the initial approximation of unknown vector $\boldsymbol{\alpha}$; set $k = 1$.

2. If $\boldsymbol{\alpha}^k$ satisfies the optimum condition (19), the solution process is terminated. Otherwise, a working set of dual elements is found with indices constituting set $B = \{i, j\} \subset \{1, 2, \ldots, l\}$, where l is the training set size. Determine set $N = \{1, 2, \ldots, l\} \backslash B$ and vectors $\boldsymbol{\alpha}_B^k$ and $\boldsymbol{\alpha}_N^k$ as parts of vector $\boldsymbol{\alpha}$ corresponding to sets B and N. Sets B and N vary from one iteration to another (i.e., depend on k).

3. Solve a quadratic programming problem for the selected $\{\alpha_i, \alpha_j\}$ according to Eqs. (7)–(9). Put $\boldsymbol{\alpha}_B^{k+1}$ equal to optimum solution of the problem.

4. Put $\boldsymbol{\alpha}_N^{k+1} = \boldsymbol{\alpha}_N^k$; $k \leftarrow k+1$ and pass to step 2.

SUCCESSIVE OVERRELAXATION (SOR) ALGORITHM

Upon adding variable $b^2/2$ to the target function of the standard SVM problem and minimizing the obtained function with respect to w, ξ, and b, the dual problem of Lagrangian minimization (still remaining a quadratic programming problem) will have no equality type constraints:

$$\text{Find } \min_{\alpha} \frac{1}{2}\boldsymbol{\alpha}^T(\mathbf{Q}+\mathbf{y}\mathbf{y}^T)\boldsymbol{\alpha} - \mathbf{e}^T\boldsymbol{\alpha}, \quad \text{subject to: } 0 \le \boldsymbol{\alpha} \le C\mathbf{e}, \tag{21}$$

where $\mathbf{Q}$ is a symmetric positive determined $l \times l$ matrix with elements $Q_{ij} = y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{y}$ is a column vector with elements representing class labels $y_i = \pm 1$, $\mathbf{e}$ is a unit vector of l components, and $\boldsymbol{\alpha}$ is the vector of Lagrange's multipliers with l components.

Solving problem (21) is equivalent to solving the system of linear algebraic equations

$$\mathbf{M}\boldsymbol{\alpha} = \mathbf{e}, \tag{22}$$

where $\mathbf{M} = \mathbf{Q} + \mathbf{y}\mathbf{y}^T$ is a symmetric positive determined matrix.

The SOR algorithm proposed by Mangasarian and Musicant [14, 15] solves system of equations (22) by the method of successive over-relaxation with allowance for inequality type constraints from problem (21) at every step. At the $(k+1)$th iteration, the components or vector $\boldsymbol{\alpha}^{k+1}$ are calculated as follows:

$$\alpha_i^{(k+1)} = \left(\omega\left(e_i - \sum_{j=1}^{i-1} m_{ij}\alpha_j^{(k+1)} - \sum_{j=i+1}^{l} m_{ij}\alpha_j^{(k)}\right)\Big/ m_{ii} + (1-\omega)\alpha_i^{(k)}\right)_{\#}, \quad i = \overline{1, l}, \tag{23}$$

where the symbol $()_{\#}$ denotes

$$(\alpha_i)_{\#} = \begin{cases} 0, & \text{for } \alpha_i \le 0 \\ \alpha_i, & \text{for } 0 \le \alpha_i \le C, \\ C, & \text{for } \alpha_i \ge C \end{cases}$$

and $\omega \in (0, 2)$ is the relaxation parameter. For $\omega = 1$, iterations (23) correspond to the Gauss–Seidel iterative method. By varying ω, it is possible to increase the convergence rate of algorithm. The iterative process converges for any initial approximation α^0.

The main steps of the SOR algorithm are as follows: 1. Select $\omega \in (0, 2)$. It is possible to begin with $\omega = 1$ and then select a different parameter in case of slow convergence of the algorithm.

2. Select the initial approximation $\boldsymbol{\alpha}^0 \in R^l$ (usually, zero vector); set $k = 0$.

3. Once $\boldsymbol{\alpha}^k$ is known, calculate $\boldsymbol{\alpha}^{k+1}$ by formulas (23). Calculations are continued until $\left\|\boldsymbol{\alpha}^{k+1} - \boldsymbol{\alpha}^k\right\| > \tau$, where τ is a preset tolerance level (typically, $\tau = 10^{-3}$).

INCREMENTAL UPDATING (IU) ALGORITHM

The IU algorithm [16, 17] stipulates online delivery of samples. For a finite number of steps, the algorithm leads to validity of the Karush–Kuhn–Tucker conditions for the previous data, as well as for the new sample. This is achieved by changing key variables of the system through the maximum possible increment of the dual variable corresponding to the new sample.

Optimum conditions. If an equality type constraint is directly incorporated into the target function of dual problem (2) by adding the term

$$b\sum_{i=1}^{l}\alpha_i y_i,$$

to $W(\boldsymbol{\alpha})$, the optimum conditions take the following form:

$$g_i = \frac{\partial W}{\partial \alpha_i} = \sum_{j=1}^{l} Q_{ij}\alpha_j + y_i b - 1 = y_i f(\mathbf{x}_i) - 1 \begin{cases} \ge 0; & \alpha_i = 0, \\ = 0; & 0 < \alpha_i < C, \quad i = 1, \ldots, l, \\ \le 0; & \alpha_i = C \end{cases} \tag{24}$$

$$\frac{\partial W}{\partial b} = \sum_{i=1}^{l} y_j \alpha_j = 0. \tag{25}$$

Before the delivery of a new sample, Karush–Kuhn–Tucker conditions (24) and (25) are valid for all samples of the training set—i.e., an SVM is constructed for l samples. According to these optimum conditions, vectors of the training set can be subdivided into three groups:

(i) $S = \{\mathbf{x}_i: 0 < \alpha_i < C\}$, the set of support vectors occurring on the optimum support boundaries, for which $g_i = 0$;

(ii) $E = \{\mathbf{x}_i: \alpha_i = C\}$, the set of support vectors violating these boundaries, for which $g_i \le 0$; and

(iii) $O = \{\mathbf{x}_i: \alpha_i = 0\}$, the set of nonsupport vectors, for which $g_i \geq 0$.

In addition, let us also introduce the set of vectors $R = E \cup O$.

The sets of indices (denoted by the corresponding low-case symbols) corresponding to the above sets of vectors induce certain partition of matrix $\mathbf{Q}$, vector of labels $\mathbf{y}$, vector of coefficients $\boldsymbol{\alpha}$, and target function gradient $\mathbf{g}$. These partitions will be denoted by the corresponding subscripts.

Sensitivity relations. Upon delivery of a new vector $\mathbf{x}_c$, the corresponding coefficient α_c is initially put equal to zero. If the expanded vector of coefficients (with dimensionality $l + 1$) is not optimum (which implies that $\mathbf{x}_c$ must become a support vector), then all coefficients α_i and shift b must be updated so as to obtain an optimum solution corresponding to the increased training set of $l + 1$ size.

Coefficients α_i corresponding to vectors of set S change their values at every step of the IU algorithm so as to hold all elements of the training set in equilibrium—that is, to maintain validity of the Karush–Kuhn–Tucker conditions.

By writing the Karush–Kuhn–Tucker conditions before and after the updating of $\boldsymbol{\alpha}$ and assuming that the compositions of sets S and R remained unchanged, we obtain the following relations that must be valid after the updating:

$$\begin{bmatrix}\Delta g_c\\ \Delta\mathbf{g}_s\\ \Delta\mathbf{g}_r\\ 0\end{bmatrix} = \begin{bmatrix}y_c & \mathbf{Q}_{cs}\\ \mathbf{y}_s & \mathbf{Q}_{ss}\\ \mathbf{y}_r & \mathbf{Q}_{rs}\\ 0 & \mathbf{y}_s^T\end{bmatrix}\begin{bmatrix}\Delta b\\ \Delta\alpha_s\end{bmatrix} + \Delta\alpha_c\begin{bmatrix}Q_{cc}^T\\ \mathbf{Q}_{cs}^T\\ \mathbf{Q}_{cr}^T\\ y_c\end{bmatrix}. \quad (26)$$

Since conditions (24) imply that $\Delta\mathbf{g}_s = 0$, it follows that the second and fourth rows of system (26) lead to a system of linear equation with respect to $\Delta\alpha_j$ (for all indices of set s) and Δb:

$$\Delta b = \beta_0\Delta\alpha_c, \quad \Delta\alpha_j = \beta_j\Delta\alpha_c, \quad (27)$$

where β is the vector of sensitivities of coefficients α_j ($j \in s$) and shift b with respect to a change in α_c. This vector has the structure of $\begin{bmatrix}\beta_0 & \boldsymbol{\beta}_s^T\end{bmatrix}$ and is determined as

$$\beta = -\begin{bmatrix}0 & \mathbf{y}_s^T\\ \mathbf{y}_s & \mathbf{Q}_{ss}\end{bmatrix}^{-1}\begin{bmatrix}y_c\\ \mathbf{Q}_{cs}^T\end{bmatrix}, \quad (28)$$

where the first matrix term in the product will be denoted as $\Re$.

Substituting expressions (27) into the first and third rows of system (26), we obtain the following relation:

$$\begin{bmatrix}\Delta g_c\\ \Delta\mathbf{g}_r\end{bmatrix} = \gamma\Delta\alpha_c, \quad (29)$$

where γ is the vector of margin sensitivities defined as

$$\gamma = \begin{bmatrix}y_c & \mathbf{Q}_{cs}\\ \mathbf{y}_r & \mathbf{Q}_{rs}\end{bmatrix}\boldsymbol{\beta} + \begin{bmatrix}Q_{cc}\\ \mathbf{Q}_{cr}^T\end{bmatrix}. \quad (30)$$

Thus, the updating of SVM related to the delivery of a new training set member must be controlled by sensitivity relations (27) and (29).

Upper limit of $\Delta\alpha_c$ value. Since the compositions of sets S and R change when coefficients α_j and b are updated, relations (26) cannot be used directly for obtaining a new state of the SVM. The IU algorithm determines the maximum possible increment $\Delta\alpha_c$ for which some vector can exhibit transition from its initial set to an adjacent one. In order to take these changes into account, let us consider the following situations.

(i) Some coefficient α_i, which is an element of vector α_s, reaches a boundary (0 or C). Let $\varepsilon > 0$ be a small value, define the following sets of indices:

$$I_S = \{I_+^S \cup I_-^S\}, \quad I_+^S = \{i \in s: \beta_i > \varepsilon\},$$
$$I_-^S = \{i \in s: \beta_i < -\varepsilon\}$$

and put

$$\Delta\alpha_i^{\max} = \begin{cases}C - \alpha_i, & i \in I_+^S\\ -\alpha_i, & i \in I_-^S.\end{cases}$$

The maximum possible increment of α_c preceding the transition of some vector from S to R is then as follows:

$$\Delta\alpha_c^S = \min_{i \in I^S}\left|\frac{\Delta\alpha_i^{\max}}{\beta_i}\right|\operatorname{sgn}\left(\frac{\Delta\alpha_p^{\max}}{\beta_p}\right),$$

where p is the index of the minimum $\left|\Delta\alpha_i^{\max}/\beta_i\right|$ value. If this minimum value is attained for several indices, then index p is defined as

$$p = \arg\min_{i \in I^S}\left|\frac{\Delta\alpha_i^{\max}}{\beta_i}\right|.$$

(ii) Some g_i from set R reaches zero. The maximum possible increment of α_c preceding the transition of some vector from R to S is then calculated in the following way:

$$\Delta\alpha_c^R = \min_{i \in I^R}\frac{-g_i}{\gamma_i},$$

where $I^R = \{I_+^R \cup I_-^R\}$, $I_+^R = \{i \in \mathrm{e}: \gamma_i > \varepsilon\}$, and $I_-^R = \{i \in \mathrm{o}: \gamma_i < -\varepsilon\}$.

(iii) $g_c = 0$. Then, if a change is still allowed (i.e., $\gamma_c > \varepsilon$), the maximum possible increment of α_c is

$$\Delta\alpha_c^g = \frac{-g_c}{\gamma_c}. \quad (31)$$

(iv) $\alpha_c = C$. In this case, the maximum possible increment of α_c is as follows:

$$\Delta\alpha_c^\alpha = C - \alpha_c.$$

A minimum of the four values listed above,

$$\Delta\alpha_c^{\max} = \min(\Delta\alpha_c^S, \Delta\alpha_c^R, \Delta\alpha_c^g, \Delta\alpha_c^a) \quad (32)$$

is the ultimate maximum possible increment of α_c. It is important to establish on which set of indices minimum (32) is attained (for determining the maximum possible increment of $\Delta\alpha_c$) and to define the corresponding value of this index.

Recursive updating of matrix $\Re$. After determining the maximum possible increment of α_c), the IU algorithm calculates the increments of key variables of the system ($\Delta\boldsymbol{\alpha}_s$, Δb, and Δg) using sensitivity relations (27) and (29). Matrix $\Re$ must also be recalculated so as to take into account a change in the composition of set S. Let us consider an effective method of updating this matrix in typical cases.

(i) Some vector $\mathbf{x}_k$ is added to set S. In this case, the Sherman–Morrison–Woodbury formula for inversion of a block matrix [18] leads to the following relation:

$$\Re \leftarrow \begin{bmatrix} \Re & \boldsymbol{\eta}_k \\ \boldsymbol{\eta}_k^T & Q_{kk} \end{bmatrix}^{-1} = \begin{bmatrix} \Re & 0 \\ 0 & 0 \end{bmatrix} + \frac{1}{Q_{kk} - \boldsymbol{\eta}_k^T \Re \boldsymbol{\eta}_k} \begin{bmatrix} \boldsymbol{\beta}_k \\ 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{\beta}_k^T & 1 \end{bmatrix},$$

where $\boldsymbol{\eta}_k = \begin{bmatrix} y_k & Q_{ks}^T \end{bmatrix}^T$ and $\boldsymbol{\beta}_k = -\Re\eta_k$.

(ii) Some vector $\mathbf{x}_k$ is removed from set S. In this case, matrix $\Re$ is constructed in the following manner:

$$\Re_{ij} \leftarrow \Re_{ij} - \Re_{kk}^{-1}\Re_{ik}\Re_{kj}, \quad \forall\Re_{ij} \in \Re; \quad i \in s \cup \{0\},$$

$i, j \neq k$, where index 0 corresponds to b.

Structure of the IU algorithm. The main block of the IU algorithm represents a procedure for updating the existing optimum solution upon adding a single new sample. We have an SVM constructed on a training set of size l. Upon adding new sample $\{\mathbf{x}_c, y_c\}$, we use the existing solution $\{\alpha_i^l, b^l\}$, matrix $\Re$, and the new sample to obtain the new optimum solution $\{\alpha_i^{l+1}, b^{l+1}\}$, $i = 1, ..., l+1$ with the aid of the IU algorithm.

The main steps of the IU algorithm ($l \leftarrow l + 1$) are as follows.

1. Set $\alpha_c = 0$.

2. If $g_c > 0$, the existing solution is retained and the procedure is terminated.

3. As long as $g_c < 0$ and $\alpha_c < C$, the algorithm

—calculates β and γ using formulas (28) and (30);

—calculates $\Delta\alpha_c^{\max}$ according to Eq. (32) and determines index k for which the minimum in (32) is attained;

—updates $\alpha_c \leftarrow \alpha_c + \Delta\alpha_c^{\max}$;

$$\begin{bmatrix} \Delta b \\ \Delta\boldsymbol{\alpha}_s \end{bmatrix} \leftarrow \boldsymbol{\beta}\Delta\alpha_c^{\max}, \quad \begin{bmatrix} b \\ \boldsymbol{\alpha}_s \end{bmatrix} \leftarrow \begin{bmatrix} b \\ \boldsymbol{\alpha}_s \end{bmatrix} + \begin{bmatrix} \Delta b \\ \Delta\boldsymbol{\alpha}_s \end{bmatrix};$$

$$\begin{bmatrix} \Delta g_c \\ \Delta\mathbf{g}_r \end{bmatrix} \leftarrow \boldsymbol{\gamma}\Delta\alpha_c^{\max}, \quad \begin{bmatrix} g_c \\ \mathbf{g}_r \end{bmatrix} \leftarrow \begin{bmatrix} g_c \\ \mathbf{g}_r \end{bmatrix} + \begin{bmatrix} \Delta g_c \\ \Delta\mathbf{g}_r \end{bmatrix};$$

{if $k \in s$, then {$\mathbf{x}_k$ passes from S to R}, otherwise {if $k \in r$, then {$\mathbf{x}_k$ passes from R to S}, otherwise {if $k = c$, then the procedure is terminated}. (Note that, if the delivery of new samples is continued, termination of the procedure implies passage to step 1. In this case, if $k = c$ and the minimum in (32) is provided by formula (31), then {$\mathbf{x}_c$ is added to S and matrix $\Re$ is recursively updated}, otherwise {$\mathbf{x}_c$ is added to E}. After that, $l \leftarrow l + 1$ and the algorithm passes to step 1.)

—recursively updates matrix $\Re$ and returns to step 3.

NAIVE ONLINE RISK MINIMIZATION ALGORITHM (NORMA)

Investigation [19] of the online learning of kernel machines led to the development of the simple and effective Naive Online Risk Minimization algorithm (NORMA) for a wide circle of problems including classification, reconstruction of regression, and classification in the absence of labels (novelty detection) based on the stochastic gradient descent in a Hilbert space of approximating functions.

The aim of any learning algorithm consists in constructing a machine ($f: X \to R$) based on l-dimensional training set D of samples $(\mathbf{x}_1, y_1), ..., (\mathbf{x}_l, y_l) \in X \times Y$. Let the loss function $L: R \times Y \to R$, $L = L(f(\mathbf{x}), y)$ rate the deviation of estimates $f(\mathbf{x})$ from observed labels y. In the case of classification, where $Y = \{-1, +1\}$, $\text{sign}(f(\mathbf{x}))$ is interpreted as the prediction of class $\mathbf{x}$ while $|f(\mathbf{x})|$ reflects the level of confidence in this classification.

Result f of the learning algorithm operation is called a hypothesis. The set of all possible hypotheses is denoted H. Let H be a Hilbert space with reproducing kernel. This implies that the Hilbert space has a kernel function $K: X \times X \to R$ and scalar product $\langle\cdot,\cdot\rangle_H$ such that

(i) K possesses the reproducing property:

$$\langle f, K(\mathbf{x}, \cdot)\rangle_H = f(\mathbf{x}), \quad \mathbf{x} \in X, \quad \forall f \in H; \quad (33)$$

(ii) space H is a closure of the linear span of all functions $K(\mathbf{x}, \cdot)$, $\mathbf{x} \in X$.

The scalar product in space H is introduced according to (33) as

$$K(\mathbf{x}, \mathbf{x}') = \langle K(\mathbf{x}, \cdot), K(\mathbf{x}', \cdot)\rangle_H$$

and generates a norm that is a measure of the complexity of functions. For any element, we have

$$f \in H: \|f\|_H = \langle f, f\rangle_H^{1/2}.$$

Various classes of functions can be learned using different kernels. A function that minimizes regularized risk as

$$R_{\text{reg}}[f, D] = R_{\text{emp}}[f, D] + \lambda\|f\|_H^2$$

is unique and represents a kernel machine, i.e., has the form of

$$f(\mathbf{x}) = \sum_{i=1}^{l} \beta_i K(\mathbf{x}_i, \mathbf{x}),$$

where $\beta \in R^l$ is the vector of attribute weights. Positive regularization parameter λ matches a small empirical error to the degree of smoothness (complexity) of the decision function. Regularization parameter C for standard SVM learning is related to parameter λ as $C = 1/(2l\lambda)$.

The NORMA generates a sequence of hypotheses $f_1, \ldots, f_{l+1}$, where f_1 is some arbitrary hypothesis and f_i ($i > 1$) is a hypothesis constructed upon the delivery of $(i-1)$th sample. Let $L(f_k(\mathbf{x}_k), y_k)$ be the loss function of the learning algorithm for the attempt at predicting y_k based on $\mathbf{x}_k$ and preceding samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_{k-1}, y_{k-1})$. An appropriate measure of accuracy of the algorithm employing training set D of size l then would be the cumulative loss function

$$L_{\text{cum}}[f, D] = \sum_{k=1}^{l} L(f_k(\mathbf{x}_k), y_k).$$

Note that f_k is tested on sample $(\mathbf{x}_k, y_k)$ that was not involved in the learning of f_k. Thus, if a small value of the accumulated loss function will be guaranteed, the situation of overfitting can be avoided.

Since we are considering an online algorithm, it is expedient to define (in addition to the regularized risk) the instantaneous regularized risk for a single sample $(\mathbf{x}_k, y_k)$ as follows:

$$R_{\text{inst}}[f, x, y] = R_{\text{reg}}[f, (\mathbf{x}, y)].$$

The NORMA produces a gradient descent in the direction of minimum instantaneous risk. The general form of the updating rule is as follows:

$$f_{k+1} = f_k - \eta_k \frac{\partial R_{\text{inst}}[f, \mathbf{x}_k, y_k]}{\partial f}\bigg|_{f=f_k},$$

here $f_i \in H$, $\partial/\partial f$ is the gradient with respect to f, and $\eta_k > 0$ is the learning rate.

Taking into account the reproducing property of the kernel, we obtain the following relation:

$$f_{k+1} = (1 - \eta_k\lambda)f_k - \eta_k L'(f_k(\mathbf{x}_k), y_k)K(\mathbf{x}_k, \cdot),$$

where $L'(z, y) = \partial L(z, y)/\partial z$. It is evident that, for any k, we have $\eta_k < 1/\lambda$.

Let us select the initial hypothesis to be $f_1 = 0$ and represent hypothesis f_k in the form of kernel expansion as

$$f_k(\mathbf{x}) = \sum_{i=1}^{k-1} \alpha_i K(\mathbf{x}_i, \mathbf{x}), \quad \mathbf{x} \in X,$$

where coefficients at the kth step are updated as

$$\alpha_k = -\eta_k L'(f_k(\mathbf{x}_k), y_k), \quad i = k;$$
$$\alpha_i = (1 - \eta_k\lambda)\alpha_i, \quad i < k.$$

We then add shift b ($b \in R$) to the decision function and update b at each step as

$$b_{k+1} = b_k - \eta_k \frac{\partial R_{\text{inst}}[f + b, \mathbf{x}_k, y_k]}{\partial b}\bigg|_{f+b=f_k+b_k}.$$

NORMA for classification problem. The classification problem is solved using the so-called soft margin loss function $L_\rho(f(\mathbf{x}), y) = \max(0, \rho - yf(\mathbf{x}))$, where $\rho \geq 0$ is the margin parameter. The function $L_\rho(f(\mathbf{x}), y)$ is positive, provided that f for $(\mathbf{x}, y)$ does not exceed the margin (at least by ρ value). In this case, f is said to have a margin error. If f actually made this error, then $L_\rho(f(\mathbf{x}), y) \geq \rho$.

Let σ_k be an indicator of f having a margin error at $(\mathbf{x}_k, y_k)$. Then,

$$L'_\rho(f_k(\mathbf{x}_k), y_k) = -\sigma_k y_k = \begin{cases} 0, & y_k f_k(\mathbf{x}_k) > \rho \\ -y_k, & y_k f_k(\mathbf{x}_k) \leq \rho, \end{cases}$$

and the updating rule takes the following form:

$$f_{k+1} = (1 - \eta_k\lambda)f_k + \eta_k\sigma_k y_k K(\mathbf{x}_k, \cdot),$$
$$b_{k+1} = b_k + \eta_k\sigma_k y_k.$$

In terms of kernel expansion coefficients α_i ($i = 1, \ldots, k-1$), the updating appears as

$$(\alpha_i, \alpha_k) \leftarrow ((1 - \eta_k\lambda)\alpha_i, \eta_k\sigma_k y_k).$$

A key point in algorithms of the type under consideration is the choice of step (learning rate) η_k. It is a usual practice either to set a constant step $\eta_k = \eta < 1/\lambda$ or to select some rule according to which η_k decreases with increasing k. A method of automated step correction using the so-called Stochastic Meta-Descent algorithm was proposed in [20].

It was shown in [19] that, in the case of soft limitations imposed on the loss function, the average instantaneous risk

Table 1. Dependence of the quality characteristics of the SMO algorithm on regularization parameter C

C	CPU time	SV	BSV	(SV–BSV)/SV
1	2.203	161	135	0.161
10	4.547	82	38	0.293
100	10.343	53	5	0.906
250	**9.500**	**51**	**2**	**0.961**
500	13.391	49	0	1

$$\frac{1}{l}\sum_{k=1}^{l} R_{\text{inst}}[f_k, \mathbf{x}_k, y_k]$$

of hypotheses produced by the NORMA tends to the minimum regularized risk

$$\min_{f+b} R_{\text{reg}}[f+b, D]$$

at a speed on the order of $O(l^{-1/2})$. If samples of the training set are independent and equally distributed, the regularized risk of averaged hypothesis with a high probability also tends to the minimum of expected risk.

EXPERIMENTAL RESULTS

We have experimentally studied the comparative efficiency of four algorithms, including SVM-Light, SMO, SOR, and IU, in solving a classification problem. The SVMs were constructed on real data with allowance for commonly accepted recommendations:

—scaling of data;

—use of a nonlinear (in the given case, Gaussian) kernel; and

—selection of the regularization (C) and kernel (σ) parameters determined for a small-size sample using a procedure of search on a rough grid, followed by thorough refinement in a region of best values of the adopted quality characteristics.

The efficiency of algorithms was evaluated in terms of the following characteristics:

—time spent for the SVM construction (CPU time, s);

—total number of support vectors (SV);

—number of bound support vectors (BSV) for which $\alpha_i = C$;

—fraction of unbound support vectors, (SV–BSV)/SV, which characterizes stability of the SVM (by stable SVM is implied a machine for which at least support vector exists such that $\alpha_i \neq C$; these support vectors are called unbound);

—algorithm quality (error) rating obtained by 10-fold cross-validation test (CV_err);

—upper bound of estimated SV classifier (1) generalization error (J_err); and

—fraction of incorrectly classified objects in the testing set (Test_err).

The SV classifiers were tested on real data sets. Data of the first sample set (size, $l = 1050$; number of attributes, $n = 10$) were dense, while data of the second sample set ($l = 4974$; $n = 50$; test set size, $m = 4998$) were sparce.

Table 1 gives an example of preliminary tuning of the regularization parameter C at a fixed value of the kernel parameter ($\sigma = 2$) on 282 samples of the second set. Evidently, a successful choice of parameter C provides a reasonable balance between the algorithm operation time and quality/stability of a constructed SV classifier.

Table 2 presents data of numerical experiments on the comparative efficiency of SV classifiers. Let us point out the main advantages of algorithms under consideration. The IU algorithm is accurate. Data for SVM learning can be delivered both in packages and one-by-one in the online regime. This is the fastest SV classifier. The SMO method is the fastest among the iterative algorithms. As the volume of a training set

Table 2. Characteristics of SV classifiers

Algorithm	C	σ	CPU time	SV/BSV	(SV – BSV)/SV	CV_err	
First sample: $l = 1050$							J_err
SMO	8	1	46.812	429/410	0.044	0.158	0.227
SOR ($\omega = 0.7$)	4	0.25	161.531	328/247	0.247	0.149	0.068
IU	2000	2	8.297	278/247	0.111	0.134	0.264
SVM-Light	100	10	315.937	415/15	0.964	0.127	0.036
Second sample: $l = 4974$, $m = 4998$							Test_err
SMO	50	1	11721.704	375/40	0.893	0.002	0.006
SOR ($\omega = 0.7$)	128	1	14326.578	333/5	0.985	0.001	0.005
IU	250	1.5	418.188	234/8	0.957	0.001	0.006
SVM-Light	150	4	17963.963	431/3	0.993	0.001	0.004

grows, the operation time increases most slowly as compared to that for other algorithms under consideration. The SOR algorithm admits learning on vast training sets provided effective organization of the computational process. Preliminary tuning of relaxation parameter ω controlling the convergence speed allows operation time of the SOR algorithm to be significantly reduced. SVM-Light is a fast iterative algorithm that leads to construction of a very stable SV classifier.

The results of experiments showed that, in processing every sample set, each particular algorithm requires using its own parameters, which should be very thoroughly selected. This choice should be oriented toward both the average quality characteristics and the average algorithm operation time. A successful choice of parameters significantly decreases the operation time without loss of the SVM learning quality.

The SVM-based algorithms are widely used in many fields of basic and practical investigations. Some algorithms of SV classifiers are implemented in various program packages such as MATLAB and R. The most complete information concerning commonly accessible SVM algorithm software is presented on site [21]. Review [22] presents a summary and comparative analysis of various SVM algorithms included in various program packages written on programming language R. A powerful and convenient modern means of biological data processing is offered by Bioconductor project with open code [23], which is also written in R and provides highly effective software for genome data processing.

ACKNOWLEDGMENTS

This work was supported by the Ministry of Education and Science of the Russian Federation in the framework of program no. 5-100-2020.

REFERENCES

1. N. O. Kadyrova and L. V. Pavlova, Biophysics (Moscow) 59 (3), 364 (2014).
2. V. N. Vapnik, *The Nature of Statistical Learning Theory* (Springer, 2000).
3. A. Elisseeff and M. Pontil, in *Advances in Learning Theory: M ethods, Models and Applications*, Ed. by J. A. K. Suykens et al. (2003), p. 111.
4. T. Joachims, in *Proc. 17th Int. Conf. on Machine Learning* (Stanford, CA, 2000), p. 431.
5. T. Fawcett, in *ROC Graphs: Notes and Practical Considerations for Researchers* (Kluwer, 2004), p. 1.
6. J. Davis and M. Goadrich, in *Proc. 23rd Int.. Conf. on Machine Learning* (Pittsburgh, PA, 2006).
7. E. Osuna, R. Freund, and F. Girosi, in *Proc. of IEEE NNSP'97* (Amelia Island, FL, 1997), p. 279.
8. T. Joachims, in *Advances in Kernel Methods – Support Vector Learning* (MIT Press, 1998), p. 41.
9. J. C. Platt, in Advances in Kernel Methods – Support Vector Learning (MIT Press, 1999), p. 185.
10. J. Platt, in *Advances in Neural Information Processing Systems*, Vol. 11 (MIT Press, 1999), p. 557.
11. S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, Technical Report CD-99-14 (1999), p. 1 (1999).
12. P.-H. Chen, R.-E. Fan, and C.-J. Lin, *Lecture Notes in Artifical Intelligence* 3734, 45 (2005).
13. S. Keerthi and E. Gilbert, Machine Learning **46** (1–3), 351 (2002).
14. O. Mangasarian and D. Musicant, IEEE Trans. Neural Networks **10** (5), 1032 (1999).
15. O. Mangasarian and D. Musicant, in *Applications and Algorithms of Complementarity*, Ed. by M.C. Ferris, O.L. Mangasarian, and J.-S. Pang (Kluwer, Boston 2000). ftp://ftp.cs.wisc.edu/math-prog/tech-reports/99-03.ps.
16. G. Cauwenberghs and T. Tomaso Poggio, in *Advances in Neural Information Processing Systems*, Vol. 13 (MIT Press, 2001), p. 409.
17. P. Laskov, C. Gehl, S. Krüger, and K.-R. Müller, J. Mach. Learn. Res. **7**, 1909 (2005).
18. G. H. Golub and C. F. van Loan, *Matrix Computations*, 3rd ed. (Johns Hopkins Univ. Press, Baltimore, 1996).
19. J. Kivinen, S. Smola, and R. Williamson, IEEE Trans. Signal Process. **52** (8), 2165 (2004).
20. S. Vishwanathan, N. Schraudolph, and A. Smola, J. Mach. Learn. Res. **6**, 1 (2005).
21. http://www.kernel-machines.org/software.
22. A. Karatzoglou, D. Meyer, and K. Hornik, J. Stat. Soft. **15** (9), 1 (2006).
23. http://www.bioconductor.org/packages/devel/bioc/html/MLSeq.html.

Translated by P. Pozdeev