Binary Linear Support Vector Machines for High Dimensional Spaces with Disjoint Subspace Combining

Valentina Sulimova [1[0000-0001-9673-3863]], Mikhail Kurbakov[1[0000-0001-9767-923X]], Nikita Mityugov [1[0000-0001-7319-7068]] and Oleg Seredin [1[0000-0003-0410-7705]]

[1] Tula State University, Tula, Russia
vsulimova@yandex.ru

Abstract. This paper proposes a new relatively simple SVM-based method to solve binary classification problem in high dimensional feature space that in contrast to the traditional SVM has high level of data parallelism and in contrast to Bagging for features obtains a single decision instead of an ensemble of multiple classifiers and so is more memory efficient. Experiments show effectiveness of sequential realization of the proposed method in contrast to SVM and Bagging for features for large problems for both of l1 and l2 regularizations.

Keywords: binary linear classification, SVM, high-dimensional feature space, random subspaces.

1 Introduction

Support Vector Machines (SVM) [1] is powerful and convenient method to solve a supervised binary classification problem that has been widely used in many applications [2, 3, 4] and has an important advantage over neural network methods and deep learning [5], which lies in strict mathematical formalization and good interpretability (explainability) of the results.

Classical SVM deals with predictive binary classification. Using a training set of observations, each of which is a point in a multidimensional space of feature measurements marked by a class-label, SVM finds a maximum margin decision function that separates the observations into positive and negative classes.

It should be noted that extracting appropriate features that can reflect the intrinsic content of dataset as complete as possible to make well-quality decision function is very important stage of data analysis but it is bad formalized and still a challenge [6]. In this regard, in a number of cases there is a tendency to form a high-dimensional feature space, when the number of features can exceed the number of observations. Such situation is typical for various domains, such as public health [5], gene expression analysis [7], natural language processing [8, 9], electroencephalogram analysis [10], image processing etc. Also this situation can occur when using sampling methods, such as bagging [11], mean decision rules method with smart sampling [12] etc., where subsample size can be comparable or even less then the number of features.

At the same time, it is known that an increase in the number of features often makes classification algorithms prone to overfitting [13], which is expressed in a deterioration

in their generalization ability and, consequently, in a decrease in the quality of recognition of test objects. Besides increasing the number of features leads to increased computational complexity and complicates the interpretation of results.

2 Related approaches

To overcome the overfitting problem dimension reduction methods, such as Principal Component analysis [14] or Linear Discriminant Analysis [15], can be applied before model construction. But these methods construct new artificial features as linear combinations of the original ones, which obscure the direct contribution of the original features to classification.

Another way to overcome the overfitting consists in applying various feature selection techniques [16] that can be divided into filter, wrapper and embedded methods.

Filter methods, such as Variance Thresholding [17], Correlation [18], Chi-Squared Test [19], Mutual Information [20], F-test (ANOVA) [21] etc. evaluate the relevance of features by examining their intrinsic properties usually independently of each other and of predictive model. Filter methods are computationally efficient and well-scalable, but they ignore potential interactions between variables, may select multiple redundant and correlated features and selected feature set may not be the optimal for a specific algorithm [22]. As a result, they often used at the preprocessing step before applying more sophisticated methods [13].

Wrapper methods test multiple combinations of features by training a machine learning model for each of them and choose the best-performing one. Among methods of this category are Forward Selection [23], Backward Elimination [24], Recursive Feature Elimination (RFE) [25] and Metaheuristic Algorithms [26] that differs in way of selecting subsets of features. Though wrapper methods can result in better performance in contrast to filter ones, they may lead to overfitting on the training data if not used combination with cross-validation. Also since the model is retrained multiple times, they can take a long time, especially with large datasets and actually can be effectively applied when the number of features and objects is small to moderate [13].

Embedded methods integrate feature selection into the process of model training, making them both efficient and effective.

One of the most widely used embedded methods is LASSO (Least Absolute Shrinkage and Selection Operator) [27], that is SVM with $l1$ penalty (Lasso SVM) instead of classical SVM with $l2$ -regularization [0]. Lasso SVM (that is suitable only for linear models) allows effectively eliminate irrelevant features and leads to a sparse solution. But in a number of cases $l1$-regularization leads to relatively reduced classification accuracy [28]. To overcome this disadvantage, the elastic-net regularization technique has been proposed which uses both the $l2$ - and the $l1$-regularization in the objective function [29] and methods with controlled sparsity level [30]. Also there are generalizations for kernel-based classification (kernel selection) [30, 31]. Embedded methods are most powerful among feature selection techniques, but they are time-consuming and have significant limitations for use in real-world applications [10].

A separate series of studies is aimed at improving the efficiency of solving the SVM problem. Some approaches to improve SVM performance consist in introducing heuristics, for example, using kernel matrix approximation [32], its caching and compression [33, 34], taking into account the sparsity of features [35] etc. as well as the use of parallel data processing technologies [36-39], including expression in terms of the MapReduce paradigm [40] and the use of GPUs [41].

However, even the use of parallel data processing technologies only alleviates, but does not solve, the problem of high computational complexity of training, since most methods have an iterative nature with data dependencies, which hinders effective parallel implementation.

The closest approaches to the proposed one are random subspace method (attribute bagging, feature bagging) [42, 43] and the average decision rule method with smart samples [12]. Each of them is based on solving many small independent SVM subproblems, followed by combining the resulting solutions into a single solution of the original problem.

These approaches offer a high degree of data parallelism. However, bagging on subspaces [42, 43] has a low convergence rate and requires storing the entire classifier ensemble.

Mean decision rules method with smart samples [12] has a high convergence rate due to the special formation of subsamples. It works very effectively with a large number of objects and quickly obtains a solution close to the reference one, but it does not provide the ability to work in a feature space of high dimensionality.

Moreover, even when the number of features in the initial large-scale problem is significantly smaller than the number of objects, training on small subsets (to improve the performance) creates a situation where the number of objects is comparable to or even smaller than the number of informative features, leading to overfitting on the subsets and a reduction in the quality of the final solution. Actually, Mean Decision Rules method with Smart Samples arises the situation, which is the subject of this article.

3 Contribution of the paper

This paper proposes a new relatively simple SVM-based method to solve binary classification problem in in high dimensional feature space that in contrast to the traditional SVM has high level of data parallelism and in contrast to Bagging for features obtains a single decision instead of an ensemble of multiple classifiers and so is more memory efficient.

The paper contains experiments description that show effectiveness of sequential realization of the proposed method in contrast to SVM and Bagging for features for both of l1 and l2 regularizations.

4 Binary SVM Classification problem

Let $X = R^m$ be a set of vector representations of all possible objects of some kind – points in the m-dimensional feature space. And let $X^{tr} = \{\mathbf{x}_j, j = 1,...,|\, X^{tr}\,|\} \subset X$ be a subset of observations labeled with class identifiers $y_j = y(\mathbf{x}_j) \in \{+1; -1\}$, that forms a training set $[X^{tr}, Y^{tr}] = [\mathbf{x}_j \in X^{tr}, y_j = y(\mathbf{x}_j), j = 1,...,|\, X^{tr}\,|]$.

A binary SVM decision rule $d(\mathbf{x}\,|\,\mathbf{a}, b) = \mathbf{a}^T\mathbf{x} + b$ is an optimal linear hyperplane in the space X, that is completely defined by a direction element $\mathbf{a} \in X$ and a bias $b \in R$ and allows to estimate a hidden class membership $\hat{y}(\mathbf{x}) : X \to \{-1,1\}$ of any observation $\mathbf{x} \in X$:

$$d(\mathbf{x}\,|\,\mathbf{a}, b) = \mathbf{a}^T\mathbf{x} + b \quad \begin{matrix} < 0, \hat{y}(\mathbf{x}) = -1, \\ \geq 0, \hat{y}(\mathbf{x}) = +1. \end{matrix} \tag{1}$$

5 Binary SVM with Disjoint Subspace Combining (DSC-SVM)

5.1 Main idea of DSC-SVM

The main idea of the proposed DSC-SVM approach consists in:

1. to split the full space of features $X = R^m$ into k disjoint random subspaces $X^{(1)} \subset X, ..., X^{(k)} \subset X$, $X^{(1)} \cup ... \cup X^{(k)} = X$, $X^{(i)} \cap X^{(j)} = \varnothing, \quad \forall\, i, j = 1,...,k,$ $i \neq j$,
2. to train SVM in each subspace $X^{(i)}$, $i = 1,...,k$ with obtaining the respective particular decision rule $d^{(i)}(\mathbf{x}^{(i)}\,|\,\mathbf{a}^{(i)}, b^{(i)})$, $i = 1,...,k$,
3. and to combine obtained decisions into a unified SVM-like decision of the initial problem for the full feature space $d(\mathbf{x}\,|\,\mathbf{a}, b)$.

5.2 A principle of hyperplanes combining

Combination of two hyperplanes. Let $\dot{X} = R^{\dot{m}}$ and $\ddot{X} = R^{\ddot{m}}$ are two disjoint linear spaces, such that $\dot{X} \cap \ddot{X} = \varnothing$. And let $\dot{\mathbf{a}}_{\dot{X}}^T\dot{\mathbf{x}} + \dot{b}$ and $\ddot{\mathbf{a}}_{\ddot{X}}^T\ddot{\mathbf{x}} + \ddot{b}$ are linear hyperplanes at the respective linear spaces, defined by its direction vectors $\dot{\mathbf{a}}_{\dot{X}} = [\dot{a}_1, ..., \dot{a}_{\dot{m}}]^T$, $\ddot{\mathbf{a}}_{\ddot{X}} = [\ddot{a}_1, ..., \ddot{a}_{\ddot{m}}]^T$ and biases $\dot{b} \in R$ and $\ddot{b} \in R$.

Let X be the combined linear space $X = \dot{X} \oplus \ddot{X}$. To find the combined hyperplane $\mathbf{a}_X^T\mathbf{x} + b$ at the space X means to find its parameters: the direction vector $\mathbf{a}_X = [a_1, ..., a_{\dot{m}+\ddot{m}}]^T \in X$ and the bias $b \in R$.

In this paper we propose one way to define the direction vector and to ways to define the bias of the combined hyperplane at the combined feature space.

Finding the direction element. Direction vectors $\dot{\mathbf{a}}_{\dot{X}}$ и $\ddot{\mathbf{a}}_{\ddot{X}}$ at the combined space X have coordinates $\dot{\mathbf{a}}_X = [\dot{a}_1,...,\dot{a}_{\dot{m}},0,...,0]^T$ and $\ddot{\mathbf{a}}_X = [0,...,0,\ddot{a}_1,...,\ddot{a}_{\ddot{m}}]^T$ respectively. We propose to find the direction vector of the combined hyperplane as the sum of them: $\mathbf{a}_X = \dot{\mathbf{a}}_X + \ddot{\mathbf{a}}_X = [\dot{a}_1,...,\dot{a}_{\dot{m}},\ddot{a}_1,...,\ddot{a}_{\ddot{m}}]^T$, what is equal to the concatenation of direction vectors at the initial spaces: $\mathbf{a}_X = \dot{\mathbf{a}}_{\dot{X}} \oplus \ddot{\mathbf{a}}_{\ddot{X}} = [\dot{a}_1,...,\dot{a}_{\dot{m}},\ddot{a}_1,...,\ddot{a}_{\ddot{m}}]^T$.

Finding the bias (variant 1). First proposed way is to define the bias b so that the combined hyperplane $\mathbf{a}_X^T\mathbf{x} + b$ passes through one of the intersection points of the initial hyperplanes $\dot{\mathbf{a}}_X^T\mathbf{x} + \dot{b}$ and $\ddot{\mathbf{a}}_X^T\mathbf{x} + \ddot{b}$ at the combined space, for example, through the point $\mathbf{x}^* = [-\dot{b}/\dot{a}_1,0,...,0,-\ddot{b}/\ddot{a}_1,0,...,0]$.

The condition $\mathbf{a}_X(\mathbf{x} - \mathbf{x}^*) = 0$ for the hyperplane $\mathbf{a}_X^T\mathbf{x} + b$ to pass through the point $\mathbf{x}^*$ holds true for $b = \dot{b} + \ddot{b}$, so, in this case we propose to find the bias of the combined hyperplane as the sum of biases of the particular hyperplanes.

Finding the bias (variant 2). Second proposed way is to find the optimal bias b, that possesses the maximum accuracy for classification the training set objects.

Combination of k hyperplanes. Let $X^{(i)} = R^{m^{(i)}}$, $i = 1,...,k$ are disjoint linear spaces and $(\mathbf{a}_{X^{(i)}}^{(i)})^T\mathbf{x} + b^{(i)}$ $i = 1,...,k$ are linear hyperplanes at them. The proposed approach for combining k hyperplanes is strict generalization of the described above approach for the case of two hyperplanes. In accordance with it the resulting hyperplane is constructed at the combined linear space $X = X^{(1)} \oplus ... \oplus X^{(k)}$. Its direction vector is defined as the concatenation of particular direction vectors of hyperplanes to be combine: $\mathbf{a} = \mathbf{a}_{X^{(1)}}^{(1)} \oplus ... \oplus \mathbf{a}_{X^{(k)}}^{(k)}$.

Finding the bias is also absolutely equal to the case of two hyperplanes. We consider two ways of its computing:

1. As sum of particular biases: $b = b^{(1)} + ... + b^{(k)}$ and
2. As the optimal value that possesses the maximum accuracy for classification the training set objects.

6 Experiments

6.1 Characteristics of the computational system

Experiments described in the following sections were carried out on a computer system with the following characteristics: Intel Core i5-13450HX (2.40 GHz), RAM 32 Gb.

6.2 Synthetic data generation

All synthetic data sets were created on the basis of the procedure initially proposed in [44] for design of experiments for NIPS-2003 variable selection benchmark. Each of two classes is composed of two Gaussian clusters each located around the vertices of a hypercube in a subspace of informative features. For each cluster, informative features are drawn independently from the standard normal distribution N(0, 1) and then randomly linearly combined within each cluster in order to add covariance. The length of the hypercube side was set to 2.

Additionally, to model possible real situations, the informative feature space was expanded by redundant features (linear combinations of informative ones), repeated features (that are duplicates of some random informative and redundant ones) and non-informative (noise) features.

To produce training and test data sets in according with this description python scikit-learn realization was used (function make_classification from sklearn.datasets).

6.3 Real data sets description

In this investigation Darwin and Gisette datasets are considered.

The Darwin dataset includes handwriting data from 174 participants. It can be downloaded from UCI Machine Learning Repository [45]. The classification task consists in distinguishing Alzheimer's disease patients from healthy people. This dataset contains 174 object each of which is characterized by 450 features. More detailed description is provided in [46].

Gisette is a handwritten digit recognition problem. The problem is to separate the highly confusable digits '4' and '9'. This dataset is one of five datasets of the NIPS 2003 feature selection challenge and can be available at the UCI Machine Learning Repository [47]. Because the original competition test set does not contain labels in this paper the validation set was used instead of it. At that we use two variants of Gisette dataset:

- Gisette1 contains 6000 objects for training (the original training set) at the space of 5000 features and 1000 objects for test (the original validation set)
- Gisette2 contains 1000 objects for training (the original validation set) at the space of 5000 features and 6000 objects for test (the original training set).
  For each data set the training set was feature-wisely scaled to [-1,1]. Then the testing data were scaled based on the same scaling factors for the training data.

6.4 Related approaches

The most related approach that were compared to the proposed ones in the framework of this paper are the classical SVM with $l2$ -regularization [1], Lasso-SVM (SVM with regularization $l1$) [27] and the Bagging for features [42, 43]. In all experiments of this paper python scikit-learn realizations of these methods were used: sklearn.svm.LinearSVC with penalty=l1 and penalty=l2 for $l1-SVM$ and $l2-SVM$ respectively and sklearn.ensemble.BaggingClassifier for the Bagging for features.

6.5 Experiments with synthetic data

Study of the influence of the splitting the feature space into subspaces. The proposed method supposes random splitting the full training set into disjoint subsets. The objective of the experiment in this section is to study how the choice of combined hyperplane bias and feature subsets size affects the decision making quality and training time of the proposed DSC-SVM method.

Experimental setup. In this experiment one synthetic data set is used that was generated in accordance with the section 6.2 and consists of 1000 train and 10000 test objects in the space of 500 features, 250 of which are informative ones and 250 are noise. At the experiment the full feature subspace is split into $ns \in \{10, 50, 100\}$ parts. To train the proposed DCS-SVM in each subspace, SVM with the regularization $l1$ and $l2$ were used ($DSC^{\cdot}_{l1}$ and $DSC^{\cdot}_{l2}$) for $C = 10$ and for two ways of computing the bias of the combined hyperplane: 1) sum of particular biases ($DSC^{sum}_{\cdot}$) and the optimal bias ($DSC^{opt}_{\cdot}$) (in accordance with the Section 5.2).

Experimental results. Table 1 contains mean accuracy values and mean train times in milliseconds accompanied with standard deviations for 20 runs for each variant of the proposed DCS-SVM and for two variants of classical SVM with $l1$ and $l2$ different regularization (SVM_{l1} and SVM_{l2}).

Table 1. Mean accuracies and train times (ms) for 1000 objects and 500 features.

Method	Accuracy			Training time (ms)		
SVM_{l1}	69.13 ± 0.06			899 ± 38		
SVM_{l2}	69.03 ± 0.00			1013 ± 24		
	ns =2	ns =10	ns =50	ns =2	ns =10	ns =50
DSC^{sum}_{l1}	72.83±0.25	73.45±0.27	57.94±0.01	30.3±4.3	14.5±1.7	33.9±4.6
DSC^{sum}_{l2}	72.76±0.30	73.53±0.19	57.92±0.16	25.8±2.7	17.2±1.8	35.1±3.2
DSC^{opt}_{l1}	72.79±0.47	74.11±0.35	74.30±0.17	30.6±3.2	15.7±2.4	32.6±3.4
DSC^{opt}_{l2}	72.57±0.45	74.12±0.28	74.36±0.14	26.7±2.2	16.3±2.7	33.5±3.1

Table 1 shows that defining the combined hyperplane bias as the sum of individual biases tends to decrease the solution quality with increasing number of subspaces for both types of regularization, in contrast to the optimal bias. In this connection in all the next experiments we will use only the optimal bias.

Additionally, we can observe a simultaneous increase in accuracy and a decrease in running time for DCS-SVM with optimal bias selection compared to classical SVM with both types of the regularization.

Study of the influence of the feature space properties. It is known that decision quality and training time can vary significantly depending on how the feature space is

formed. In this experiment, we compare the training results on datasets with the same number of objects and features, but with different feature space properties.

Experimental setup. Each synthetic data set of this experiment consists of 10000 train and 10000 test objects in the space of 5000 features. We consider 6 different training sets with various combinations of informative (*Inf*), redundant (*Rd*), repeated (*Rp*) and noise (*Ns*) features. Methods to be compare are SVM, Bagging for features as the ensemble of $est \in \{50,100\}$ SVM-estimators in random nfs -sized subspaces and the proposed DSC-SVM with the optimal choice of the bias (Section 5) and with splitting the full feature space into ns disjoint subspaces. Each method was considered for both of regularization types $l1$ and $l2$, and for the regularization SVM parameter $C = 10$.

Experimental results. Table 2 contains mean accuracy values and mean train times in seconds for 3 runs of each method.

Table 2. Mean accuracies and train times (s) for different feature compositions.

Features Inf /Rd Rp / Ns		*SVM* (l1/l2)	*DSC*^{opt} (l1/l2)			Bagging (l1/l2)	
			$ns = 10$	$ns = 50$	$ns = 100$	$nfs = 100$ $est = 50$	$nfs = 500$ $est = 100$
5000 / 0	acc.	70.2/70.1	73.3/73.5	73.5/73.1	73.4/73.3	65.8/65.8	73.0/73.0
0 / 0	time	570/13.0	4.51/4.73	4.56/5.61	5.40/5.00	4.92/2.55	28.5/28.3
2500 / 0	acc.	69.8/70.0	74.1/74.1	74.3/74.3	74.3/74.4	66.8/66.8	73.5/72.6
0 / 2500	time	324/11.9	2.41/2.63	2.07/2.06	1.93/1.77	2.12/1.86	24.7/26.0
2500/2500	acc.	75.8/75.8	75.3/75.4	74.9/74.9	74.9/74.8	70.5/70.5	75.5/75.5
0 / 0	time	352/8.9	3.08/3.24	2.09/2.18	1.96/1.77	2.35/2.53	34.0/35.7
2500 / 0	acc.	75.6/75.6	74.9/75.0	74.2/74.3	74.2/74.3	69.5/69.5	75.0/75.0
2500 / 0	time	310/9.7	19.28/3.36	3.66/3.36	2.25/2.33	2.52/2.95	340/114
2000/1000	acc.	72.0/72.1	74.8/74.9	73.9/73.9	73.8/73.8	69.6/69.7	74.7/74.6
1000/1000	time	312/209	2.94/3.18	2.04/2.19	1.91/1.80	2.01/2.18	29.4/31.9
1000 / 0	acc.	69.8/69.0	73.7/73.9	73.7/73.9	73.7/73.8	66.4/66.4	72.5/72.4
0 / 4000	time	319/8.4	2.24/2.36	1.91/1.84	1.86/1.65	1.91/1.84	22.4/24.0

As we can see from the table 2, the proposed DSC-SVM method shows the best results for different ratios of the number of informative and noise features, including the case when the feature space consists entirely of informative features and the case when 4/5 of the features are noise. In all of these cases, just as in the case where all kinds of features participate approximately equally in the formation of the feature space, DSC-SVM essentially outperforms SVM in both of accuracy and training time. Bagging results depend on its parameter values. For small size of random feature subspace and small number of estimators it shows small training times that are comparable to DSC-SVM ones. But the respective bagging accuracy values are smaller in contrast to DSC-SVM accuracy. Larger parameter values allow bagging to reach the DSC-SVM accuracy, but require significantly more time for training.

For large number of repeated or redundant features all methods rich similar accuracy, but the proposed DSC-SVM requires much less time in contrast to others.

More detailed study of DSC-SVM for multiple repeated and noise feature cases. Experiments of this section are aimed to more detailed study of the proposed method in conditions of multiple repeated and noise features.

Experimental setup. For this experiment two basic synthetic data sets were generated in accordance with the section 6.2, each of which contains 200 train objects and 100 features. Among 100 features of the first data set are 20 informative features and 80 features that repeat informative ones. Second data set contains 20 informative and 80 noise features. Additional data sets were obtained from each of basic ones by removing some features. As a result, data sets with 5, 10, 20, 50 and 100 features were obtained for both of multiple repeated feature and multiple noise feature data sets. At that, data sets with 5, 10 and 20 features contain only informative features. Each of these data sets was then randomly split into the training and test sets, each of 100 objects.

For each of these training sets the proposed DSC-SVM method with the optimal choice of the bias (Section 5) and with the splitting the feature size into ns disjoint subspaces was compared to the SVM. For both of SVM and DSC-SVM methods l2-regularization and SVM parameter $C = 10$ were used in all experiments of this section.

Experimental results. Table 3 and 4 contain mean accuracy and standard deviation values for 100 runs for multiple repeated feature datasets and for multiple noise feature data sets respectively. Dashes in these tables correspond to impossible situations when the number of subspaces ns is grater then the number of features.

Table 3. Mean accuracy and standard deviation values for multiple repeated feature data sets

Method		Features 5	10	20	50	100
SVM		78.0 ± 0.0	82.0 ± 0.0	85.0 ± 0.0	85.0 ± 0.0	85.0 ± 0.0
DSC-SVM	$ns = 2$	90.0 ± 1.4	91.2 ± 1.7	88.9 ± 1.8	88.3 ± 1.3	87.5 ± 1.1
	$ns = 3$	89.2 ± 0.6	90.6 ± 1.8	87.6 ± 1.5	86.9 ± 1.5	86.7 ± 1.7
	$ns = 4$	89.0 ± 0.7	90.7 ± 1.6	86.8 ± 1.1	86.2 ± 1.8	85.2 ± 1.9
	$ns = 5$	89.0 ± 0.0	90.9 ± 1.6	86.6 ± 1.0	85.8 ± 1.8	84.4 ± 1.7
	$ns = 8$	-	90.8 ± 1.1	86.2 ± 0.9	85.4 ± 1.3	83.8 ± 1.3
	$ns = 10$	-	90.1 ± 0.0	86.3 ± 0.7	85.2 ± 1.4	83.1 ± 1.3
	$ns = 20$	-	-	86.0 ± 0.0	85.0 ± 1.4	82.2 ± 0.9

Table 4. Mean accuracy and standard deviation values for multiple noise feature data sets

Method		Features 5	10	20	50	100
SVM		78.0 ± 0.0	80.0 ± 0.0	84.0 ± 0.0	81.0 ± 0.0	82.0 ± 0.0
DSC-SVM	$ns = 2$	90.0 ± 2.2	91.0 ± 1.8	90.5 ± 1.9	86.9 ± 2.5	79.9 ± 2.9
	$ns = 3$	89.3 ± 1.0	90.9 ± 1.6	88.9 ± 1.6	86.3 ± 2.7	83.9 ± 2.7
	$ns = 4$	89.5 ± 0.9	91.5 ± 1.8	88.6 ± 1.5	85.7 ± 1.9	84.5 ± 2.4
	$ns = 5$	89.0 ± 0.0	91.6 ± 1.7	87.8 ± 1.1	85.8 ± 2.1	84.7 ± 2.1
	$ns = 8$	-	91.4 ± 1.0	87.5 ± 1.1	85.2 ± 1.7	84.7 ± 2.0
	$ns = 10$	-	91.0 ± 0.0	87.4 ± 0.7	85.3 ± 1.5	85.2 ± 1.8
	$ns = 20$	-	-	88.0 ± 0.0	84.7 ± 1.5	84.8 ± 1.3

As we can see from tables 3 and 4, mean DSC-SVM accuracy has a tendency to decrease with increasing the number of features and feature subsets. But nevertheless, in most of cases DSC-SVM reaches the higher accuracy in contrast to SVM.

Study of the influence of data set sizes. The objective of the experiment in this section is to study methods performance with an increase in the number of objects and features.

Experimental setup. Each synthetic data set was generated in accordance with the section 6.2 and contains different number of objects from 1000 to 30000 and different number of features from 500 to 15000. In all cases, the feature space consists only of informative and noise features (in equal quantities). Test set size consists of 10000 objects in all cases of this experiment.

As at the previous experiment, methods to be compare are SVM, Bagging for features as the ensemble of $est \in \{50,100\}$ SVM-estimators in random nfs-sized subspaces and the proposed DSC-SVM with the optimal choice of the bias (Section 5) and with splitting the full feature space into ns disjoint subspaces. Each method was considered for both of regularization types $l1$ and $l2$, and for the regularization SVM parameter $C = 10$.

Experimental results. Table 5 contains mean accuracy values and mean train times in seconds for 3 runs of each method.

Table 5. Mean accuracy values and train times (s) for different training set sizes.

Objects/ features		*SVM* (l1/l2)	DSC^{opt} (l1/l2)		Bagging (l1/l2)	
			$ns = 50$	$ns = 100$	$nfs = 100$ $est = 50$	$nfs = 250$ $est = 100$
1000/ 500	acc.	70.60/71.20	75.30/75.90	75.20/74.46	74.40/74.40	76.40/76.50
	time	0.85/0.06	0.04/0.03	0.05/0.06	1.59/0.27	11.4/1.62
2000/ 1000	acc.	74.25/73.25	75.85/75.15	75.80/75.45	73.75/73.75	76.75/76.75
	time	4.86/0.36	0.07/0.08	0.16/0.12	0.55/0.58	2.45/2.14
5000/ 2000	acc.	72.30/72.10	76.40/76.55	76.40/76.75	71.75/71.17	76.40/76.45
	time	45.7/6.34	1.22/1.47	1.60/1.06	1.22/1.49	6.77/7.07
10000/ 5000	acc.	69.15/69.10	72.85/72.90	73.20/72.95	65.55/65.55	71.20/71.25
	time	551/24.9	2.12/2.10	1.96/1.98	6.95/6.96	41.4/44.61
30000/ 15000	acc.	69.75/69.14	72.68/72.86	72.95/73.04	59.88/59.90	67.05/67.05
	time	3410.7/99.6	20.15/21.71	20.12/22.36	7.22/8.10	37.99/39.88

Table 5 shows that increasing the number of objects leads to an increase in the training time of DSC-SVM, but the rate of increase is significantly lower than that of SVM. As a result, for 30000 objects, DSC-SVM even for sequential realization is approximately 5 times faster than SVM for l2 regularization and almost 170 times faster for l1 regularization, while achieving higher quality scores in both cases.

Bagging appears more suitable for small datasets. As the number of objects increases, a time comparable to that of DCS-SVM proves insufficient to achieve the required accuracy. Moreover, the difference in achievable accuracy between DSC-SVM and Bagging, with roughly the same running time, increases with the number of objects.

6.6 Experiments with real data

Experimental setup. Eat this section 3 data sets are used (Darwin, Gisette1 and Gisette2), that are described at the section 6.3.

In this experiment we compare SVM, Bagging for features as the ensemble of $est = 50$ SVM-estimators in random nfs -sized subspaces and the proposed DSC-SVM with the optimal choice of the bias (Section 5) and with splitting the full feature space into ns disjoint subspaces. Each method was considered for both of regularization types $l1$ and $l2$, and for the regularization SVM parameter $C = 0.01$.

For both of Gisette datasets accuracy was computed for the test set after training on the respective training set. Darwin data set is very small-sized and has no official test set. So, in this case the 5-folds cross-validation to estimate accuracy was used.

Experimental results. Table 6 contains mean accuracy values and mean train times in seconds for 3 runs of each method.

Table 6. Mean accuracies and train times (s) for real data sets.

Data set		*SVM* (l1/l2)	DSC^{opt} (l1/l2)			Bagging (l1/l2)	
			$ns = 5$	$ns = 10$	$ns = 30$	$nfs = 20$ $est = 50$	$nfs = 50$ $est = 50$
Darwin	acc.	82.8/77.0	86.8/85.0	88.0/86.2	87.4/87.9	86.2/86.2	90.8/90.2
	time	0.22/0.14	0.38/1.18	0.28/0.35	0.14/0.13	0.48/0.47	1.79/2.66
Gisette1	acc.	97.3/97.8	97.1/97.9	96.2/97.9	94.8/96.0	82.6/84.2	90.0/91.3
	time	8.76/10.3	13.0/9.56	7.35/10.6	8.12/8.23	5.84/2.43	12.62/5.1
Gisette2	acc.	92.0/95.8	92.2/96.0	92.2/95.6	91.1/94.6	82.45/85.9	86.2/90.0
	time	0.92/1.00	0.23/0.42	0.22/0.36	0.25/0.44	0.26/0.32	0.57/0.71

As we can see from the table 6, the proposed DSC-SVM reaches approximately the same accuracy as SVM for both of Gisette datasets and significantly outperforms it for Darwin dataset. A that the training time for a number of parameter values is comparable to the training time of SVM. But it should be noticed, that the proposed method in contrast to SVM has high level of data parallelism and so its parallel realization can additionally reduce the training time. As to the Bagging, as with synthetic data, in most cases it cannot achieve the same accuracy as DSC-SVM with comparable training time.

7 Conclusions

The paper proposes the simple method to solve the binary classification problem named DSC-SVM that consists in independent SVM training in random disjoint feature subspaces with further combining the obtained results into SVM-like decision of the initial problem at the full feature space in the form of a linear separating hyperplane. At that DCS-SVM in contrast to the traditional SVM has high level of data parallelism and can be easily parallelized. In contrast to Bagging for features DSC-SVM obtains single decision instead of ensemble of multiple classifiers and so is more memory efficient.

Experiments show that in a large number of cases the proposed method significantly outperforms SVM in accuracy and training time at once. Especially it is appropriate for large-scale problems. So, for 30000 objects and 15000 features DSC-SVM even for sequential realization is approximately 5 times faster than SVM for l2 regularization and almost 170 times faster for l1 regularization, while achieving higher quality scores in both cases. As to Bagging for features, for large data sets it cannot achieves the same accuracy as DSC-SVM with comparable training time.

Acknowledgements

The study was supported by a grant from the Russian Science Foundation, No. 26-21-20045.

References

1. Vapnik, V.: Statistical Learning Theory. John Wiley & Sons, New York (1998)
2. Thomas, A.: Kaggle 2017 Survey Results. http://www.kaggle.com/amberthomas/kaggle-2017-survey-results, last accessed 2026/05/29
3. Partheeban, I., Siluvai, S., Govindaraj, K.: Support vector machines: A literature review on their application in analyzing mass data for public health. Cureus 17 (2025)
4. Alhumaidi, N.H., Dermawan, D., Kamaruzaman, H.F., Alotaiq, N.: The use of machine learning for analyzing real-world data in disease prediction and management: Systematic review. JMIR Medical Informatics 13, e68898 (2025)
5. Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S., Dehmer, M.: An introductory review of deep learning for prediction models with big data. Frontiers in Artificial Intelligence 3, 4 (2020)
6. Salau, A.O., Jain, S.: Feature extraction: A survey of the types, techniques, applications. In: 2019 International Conference on Signal Processing and Communication (ICSC), pp. 158–164. IEEE (2019)
7. Vanitha, D.A., Devaraj, D., Venkatesulu, M.: Gene expression data classification using support vector machine and mutual information-based gene selection. Procedia Computer Science 47, 13–21 (2015)
8. Elvas, L.B., Almeida, A., Ferreira, J.C.: Natural language processing in medical text processing: A scoping literature review. International Journal of Medical Informatics 204, 106049 (2025)
9. Li, X.H., Liu, S.X.: The applications of support vector machine in natural language processing. Applied Mechanics and Materials 427–429, 2572–2575 (2013)
10. Sulimova, V., Bukhonov, S., Krasotkina, O., Mottl, V., Sterr, A., Wells, K., Windridge, D.: Sparse multimodal classification of EEG signals from rapid serial visual presentation of diagnostic images. In: Perner, P. (ed.) Machine Learning and Data Mining in Pattern Recognition (MLDM 2019), vol. 1, pp. 355–366. Ibai Publishing (2019)
11. Breiman, L.: Bagging predictors. Machine Learning 24(2), 123–140 (1996)
12. Makarova, A., Kurbakov, M., Sulimova, V.: Mean decision rules method with smart sampling for fast large-scale binary SVM classification. In: 25th International Conference on Pattern Recognition (ICPR 2020), pp. 8212–8219. IEEE (2021)

13. Zou, H., Hastie, T.: Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B 67(2), 301–320 (2005)
14. Greenacre, M., Groenen, P.J., Hastie, T., d'Enza, A.I., Markos, A., Tuzhilina, E.: Principal component analysis. Nature Reviews Methods Primers 2(1), 100 (2022)
15. Tharwat, A., Gaber, T., Ibrahim, A., Hassanien, A.E.: Linear discriminant analysis: A detailed tutorial. AI Communications 30(2), 169–190 (2017)
16. Bondarenko, K.: Feature selection methods in machine learning: From simple filters to interpretability with SHAP. Professional Bulletin: Information Technology and Security 3 (2025)
17. Harris, D.P., Harris, S.L.: Digital Design and Computer Architecture. Morgan Kaufmann, Burlington (2012)
18. Hall, M.A.: Correlation-based feature selection for machine learning. PhD thesis, University of Waikato (1999)
19. Liu, H., Setiono, R.: Chi2: Feature selection and discretization of numeric attributes. In: Proceedings of the Seventh IEEE International Conference on Tools with Artificial Intelligence, pp. 388–391. IEEE (1995)
20. Battiti, R.: Using mutual information for selecting features in supervised neural net learning. IEEE Transactions on Neural Networks 5(4), 537–550 (1994)
21. Fisher, R.A.: The use of multiple measurements in taxonomic problems. Annals of Eugenics 7(2), 179–188 (1936)
22. Cheng, X.: A comprehensive study of feature selection techniques in machine learning models. Insights in Computer, Signals and Systems 1, 65–78 (2024)
23. Whitney, A.W.: A direct method of nonparametric measurement selection. IEEE Transactions on Computers C-20(9), 1100–1103 (1971)
24. Miller, A.: Subset Selection in Regression. Chapman and Hall/CRC, Boca Raton (2002)
25. Guyon, I., Weston, J., Barnhill, S., Vapnik, V.: Gene selection for cancer classification using support vector machines. Machine Learning 46(1–3), 389–422 (2002)
26. Faizan, M., Mohammad, M.: Metaheuristic algorithms for feature selection (2014–2024). International Journal of Applied Metaheuristic Computing 16, 1–23 (2026)
27. Bradley, P., Mangasarian, O.: Feature selection via concave minimization and support vector machines. In: International Conference on Machine Learning (1998)
28. Bomze, I., D'Onofrio, F., Palagi, L., Peng, B.: Feature selection in linear support vector machines via a hard cardinality constraint: A scalable conic decomposition approach. European Journal of Operational Research (2025)
29. Wang, L., Zhu, J., Zou, H.: The doubly regularized support vector machine. Statistica Sinica 16, 589–615 (2006)
30. Tatarchuk, A., Mottl, V., Eliseyev, A., Windridge, D.: Selectivity supervision in combining pattern-recognition modalities by feature- and kernel-selective support vector machines. In: Proc. ICPR (2008)
31. Kloft, M., Brefeld, U., Sonnenburg, S. et al.: Efficient and accurate lp-norm multiple kernel learning. In: Bengio, Y. et al. (eds.) Advances in Neural Information Processing Systems 22, pp. 997–1005. MIT Press (2009)
32. Drineas, P., Mahoney, M.W.: Approximating a gram matrix for improved kernel-based learning. In: International Conference on Computational Learning Theory, pp. 323–337. Springer, Berlin, Heidelberg (2005)
33. Joachims, T.: Making large-scale support vector machine learning practical. In: Advances in Kernel Methods: Support Vector Learning, pp. 169–184. MIT Press, Cambridge (1999)
34. Zeyuan, A.Z. et al.: P-packSVM: Parallel primal gradient descent kernel SVM. In: Ninth IEEE International Conference on Data Mining, pp. 677–686. IEEE (2009)

35. Joachims, T.: Training linear SVMs in linear time. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 217–226 (2006)
36. Agarwal, A., Duchi, J.C.: Distributed delayed stochastic optimization. In: Advances in Neural Information Processing Systems, pp. 873–881 (2011)
37. Zhao, H.X., Magoulès, F.: Parallel support vector machines on multi-core and multiprocessor systems. In: 11th International Conference on Artificial Intelligence and Applications (AIA 2011). IASTED, Innsbruck (2011)
38. Dekel, O. et al.: Optimal distributed online prediction using mini-batches. Journal of Machine Learning Research 13, 165–202 (2012)
39. Jaggi, M. et al.: Communication-efficient distributed dual coordinate ascent. In: Advances in Neural Information Processing Systems 27, pp. 3068–3076 (2014)
40. Rizzi, A.M.: Support vector regression model for BigData systems. arXiv:1612.01458 [cs.DC] (2016)
41. Wen, Z. et al.: ThunderSVM: A fast SVM library on GPUs and CPUs. Journal of Machine Learning Research 19(1), 797–801 (2018)
42. Ho, T.K.: The random subspace method for constructing decision forests. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 832–844 (1998)
43. Bryll, R.: Attribute bagging: Improving accuracy of classifier ensembles by using random feature subsets. Pattern Recognition 36(6), 1291–1302 (2003)
44. Guyon, I.: Design of experiments NIPS 2003 variable selection benchmark, (2003)
45. Fontanella, F. DARWIN [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C55D0K (2022)
46. Cilia, Nicole D. et al.: Diagnosing Alzheimer's disease from on-line handwriting: A novel dataset and performance benchmarking, (2022)
47. Guyon, I., Gunn, S., Ben-Hur, A., & Dror, G. (2004). Gisette [Dataset]. UCI Machine Learning Repository. https://doi.org/10.24432/C5HP5B.