Fast SVM-based Multiclass Classification in Large Training Sets

M. Yu. Kurbakov[a] , V. V. Sulimova[a]
[a] Laboratory of cognitive technologies and simulation systems,
Tula State University, Lenine Ave. 92, Tula, Russia, 300012 - muwsik@mail.ru, vsulimova@yandex.ru

Keywords: Multiclass classification, nonlinear SVM, large-scale problems, smart sampling, handwritten digit images.

Abstract

This paper addresses the actual problem of multiclass classification in large training sets. Classical Support Vector Machines (SVM) is a popular, convenient and well-interpreted classification method, but it has a high computational complexity of a training stage in a nonlinear case and a low data parallelism. The aim of this paper is to improve the scalability of nonlinear multiclass SVM.
In the basis of this paper is Kernel-based Mean Decision Rule method with smart sampling (SS-KMDR) we previously proposed for fast solving large-scale binary SVM problems. In this paper we, at first, extend SS-KMDR for the multiclass classification problem. At second, we propose the modified algorithm of smart sample construction that allows to improve its characteristics and also extend it to possess the possibility to solve large-scale multiclass SVM problems. Experimental investigation of proposed methods was made on three large handwritten digit images data sets of different size and one large intrusion detection data set. Experiments show that both proposed multiclass methods allow to reach the quality near state-of-the-art SVC quality, but they essentially outperform it in the training time. The proposed Dual-Layer Smart Sampling SVM (DLSS-SVM method) allows additionally reduce training and test times in contrast to the basic smart sampling technique.

1. Introduction

Multiclass classification problem is one of the actual problems of machine learning. There are known a big number of applied problems of multiclass classification, among which are image search and text recognition, various tasks coming from medical systems, pharmacology, molecular biology, mining and oil industries, information security and many others socially significant areas. At that many modern real-world problems require processing large data sets to build a classification model, and these requirements may even exceed the capabilities of a single computer.

Neural networks that have become widespread make it possible to solve problems of any scale, but do not provide the ability to explain the results obtained, which gives rise specialists mistrust to them (especially critical in expert and, in particular, medical systems) and in some cases serves as a basis for refusing their use in the benefit of explainable methods (Burkart et al., 2021; Charmet et al., 2022].

In this regard, this work is based on the convenient and well-proven Support Vector Machine method (SVM) (Vapnik, 1995). In contrast to neural networks, it has strict mathematical geometrically interpretable problem formulation with a single decision, the ability to explain the obtained results, a small number of parameters (Emmert-Streib et al., 2020) and remains one of the most popular approaches in the machine learning arena (Benfenati et al., 2023).

SVM is initially designed for binary classification problem. The dominant approach to extend it onto a multiclass problem is divide-and-combine one that combine decisions of independently trained binary classifiers. Two most popular strategies within this approach are “One-versus-Rest” (OvR) also known as “One-versus-All” (OvA) and “One-versus-One” (OvO) that is also called “All-versus-All” (AvA) (Hsu et al., 2002; Duan et al., 2005; Malenichev et al., 2016). Its detailed description is provided in Section 3.2.

Both OvR and OvA strategies require to train a number of binary SVM classifiers and so, efficiency of multiclass training is largely determined by efficiency of binary training.

2. Related Work

As to improving the performance of binary SVM, despite the massive amount of research in this area, a universal tool has not yet been found.

In particular, for learning in large training sets that fit in the memory or there is data streaming, incremental methods have been proposed (Bottou, 2004; Hoi et al., 2018) and methods based on decomposition, such as chunking (Boser et al., 1992), Sequential Minimal Optimization (Platt, 1998) and others (Rivas-Perea et al., 2013). Decomposition methods form the basis of popular libraries such as SVMLight (Joachims, 1999), LibSVM (Chang and Lin, 2001) and SVMTorch (Collobert et al., 2001). The main disadvantage of all these methods is either the impossibility of using kernel functions (Aizerman et al., 1970) to introduce nonlinearity, or the low efficiency of working with them.

Papers aimed at improving performance in the nonlinear case propose the introduction of heuristics, for example, kernel matrix approximation (Drineas et al., 2005) (including generation of nonlinear features, whose inner product is approximately equal to the kernel values (Rahimi and Recht, 2007; Rahimi and Recht, 2008), kernel caching and compression (Joachims, 1999; Zhu et al., 2009) and also taking into account feature sparsity (Joachims, 2006). However, an increase in the performance of these heuristic approaches is usually accompanied by a noticeable decrease in the decision quality (Le et al., 2017).

Many authors try to accelerate computations using parallel and distributed data processing technologies (Dekel et al., 2012; Niu et al., 2011; Agarwal et al., 2011; Zhao et al., 2011), including expression in MapReduce terms (Chu et al., 2006; Rizzi, 2016; Sleeman et al., 2021) and GPU applications (Wen et al., 2018). However, initial sequential algorithms have an iterative nature and between-iterations data dependencies, preventing effective parallelization. As a result, parallel and distributed processing only mitigates, but does not solve, the problem of high computational complexity. At that attempts to avoid data dependencies lead to a noticeable decrease in the decision quality (You et al., 2015).

Our previously proposed Kernel-based mean decision rule method with smart sampling (SS-KMDR) (Makarova et al., 2020) stands out from this group due to its intellectual way of forming subsamples, which allows to increase the method's convergence speed compared to traditional random sampling (Chauhan et al., 2018; Byrd et al., 2016; Makarova et al., 2019), requires less intensive preparatory calculations in contrast to (Sadrfaridpour et al., 2019; Csiba et al., 2016; Zhao et al., 2014), has a high degree of data parallelism and is suitable for nonlinear case.

However, experimental study of the smart sampling technique that is proposed in (Makarova et al., 2020) has not a multi-class version and for binary problems has shown some limitations of its using for large training sets especially in a large-dimensional feature space.

3. Contribution of the Paper

In this paper we, at first, extend SS-KMDR for the multiclass classification problem. At second, we propose the modified algorithm of smart sample construction that allows to improve its characteristics and also extend it to possess the possibility to solve large-scale multiclass SVM problems.

We made experiments in four large data sets to compare proposed approaches with different existing accurate and heuristic methods of solving multiclass SVM problems.

4. Smart Sampling SVM for Multiclass Classification

4.1 Multiclass and Binary Classification Problems

Let Ω^* be a set of all possible objects ω of some kind, each of which can be presented by n-length real-valued feature vector $\mathbf{x}(\omega)=[x_1,...,x_n]$. We suppose that some finite subset of objects $\Omega=\{\omega_j, j=1,...,N\}\subset\Omega^*$ is oobservable through its representations $\mathbf{x}_j=\mathbf{x}(\omega_j), j=1,...,N$ jointly with class labels $y_j=y(\omega_j)\in\{1,...,m\}, j=1,...,N$, $m\geq 2$ and constitutes the training set $[\Omega,Y]=\{[\omega_j,y_j], j=1,...,N\}$. The task is to make a decision function that for any new object $\omega\notin\Omega$ will estimate the unknown class-label $\hat{y}(X_\omega)$.

The dominant approach to solve multiclass SVM problem is to divide the initial multiclass problem onto a number of binary tasks with consequent their solving and combining the obtained decisions into decision of the initial multiclass problem.

The binary problem is a special case of multiclass problem where the number of classes $m=2$.

4.2 One-versus-the-Rest (OvR) and One-versus-All (OvA) Strategies for Multiclass Classification

One-versus-the-Rest strategy consists in fitting one classifier per class, that is for m classes m classifiers should be constructed. For each classifier, the class is fitted against all the other classes and, so, the full training set is used at each times. The resulting class label we select by the maximum value of class probability.

One-vs-One strategy requires to train $m(m-1)/2$ classifiers for m classes, i.e. $O(m^2)$ in contrast to $O(m)$ for OvR. But each individual OvO learning problem only involves a relatively small subset of the data whereas, with OvR, the complete dataset is used m times. The resulted class label is determined by voting.

4.3 The Basic Idea of the Smart Sampling

The basic idea of the Smart Sampling was originally proposed by us in (Makarova et al., 2020) for Kernel-based Mean Decision Rules method and was aimed to accelerate its convergence in contrast to traditional random samples. This idea consists in forming samples in a special way. It is based on the fact that the binary SVM decision function that is found in the form of an optimal separating hyperplane depends only on so called support objects of two classes situated near the hyperplane. As a result, excluding any of non-support objects from the training set does not change the decision. This fact allows to intellectually reduce the training set and so to decrease the training time.

In (Makarova et al., 2020) we proposed to form a smart sample only from support objects that are obtained as a result of training for small simple random samples. Objects selected by such a way are good enough candidates to be support objects in the initial problem for the full training set. The formal definition of forming smart sample is given by the Algorithm 1.

```
Algorithm 1. Basic algorithm for forming Smart Sample
Parameters:
  $SSize$ – desired approximate size of smart sample
  $RSize$ – size of random samples
1:  set the smart sample as empty $\Omega_{smart}=\varnothing$.
2:  take small random sample $\Omega_{rnd}\subset\Omega$
3:  train SVM with $[\Omega,Y]_{rnd}$ to obtain support objects $\Omega_{\sup}\subseteq\Omega_{rnd}$
4:  update the smart sample $\Omega_{smart}=\Omega_{smart}\cup\Omega_{\sup}$
5:  if the smart sample size is not enough $|\Omega_{smart}|<SSize$ then repeat steps 2-5.
```

It should be noted that the resulted smart samples can be of different size, each of which is near the desired value $SSize$ in contrast to the size of random samples $RSize$. The actual number of random samples can also be different and depends on the desired smart sample size $SSize$ and on the number of support objects for random samples, which in turn is determined by parameters of SVM-training (constant C and coefficient γ of RBF kernel) and data properties.

Figure 1 illustrates the idea of forming a smart sample. Figures 1a-1e present random samples and the results of SVM-training for them. The Figure 1f shows the smart sample that consists of support object (circled at figures 1a-1e) obtained as a result of training for the respective random samples.

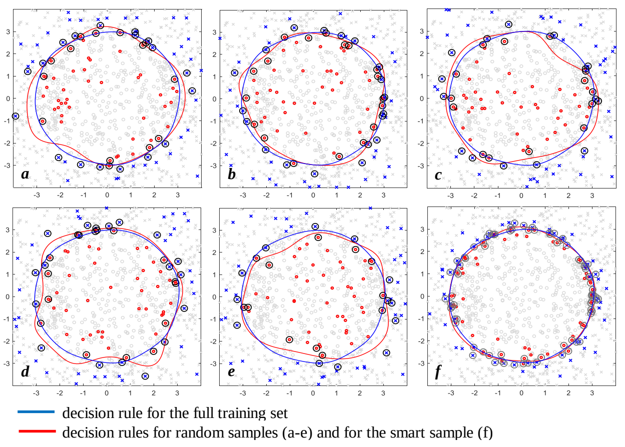

Figure 1. Results of training for random (a-e) and smart (f) sample (red) in contrast to the result of training for the full training set (blue)

4.4 Double-Layer Smart Sampling with Early Stopping

Experimental study of basic smart sampling technique that is described in Section 2.3 has shown that when increasing the training set size especially in large-dimensional feature space, to ensure the required decision quality, it is necessary to increase the size of the random and smart samples. However, this in turn leads to an increase in the operating time due to the nonlinear dependence of the training time on the number of objects (in the nonlinear case). In work (Makarova et al., 2020), to increase the quality with a limitation on the smart sample size, averaging of decision rules constructed over several smart samples is carried out. But in this case the quality increases slightly, and the operating time increases in proportion to the number of smart samples.

In this connection we propose the modified algorithm to form a smart sample. At first, to cover more support objects of the full training set with limited smart sample size we propose double-layer procedure to form a smart sample. It consists in constructing a number of first-layer (L1) smart samples, to train in them and to form the second-layer (L2) smart sample from support objects that obtained at the 1L-training.

At second, if a smart sample from some moment fills slowly (a small number of new support objects are added), then to reduce the execution time it may be advantageous to stop the process of its formation early. With this purpose we introduce additional threshold Δ.

The proposed procedure of forming the double-layer smart sample with early stopping is given by Algorithms 2 and Algorithm 3.

Algorithm 2. First-layer Smart Sample with early stopping

```
Parameters:
SSize1 - desired size of first-layer smart sample
RSize – size of random samples
$\Delta_1$ - the threshold for early stopping
1: set the 1-layer smart sample as empty $\Omega^{L1}_{smart} = \varnothing$ and
   its actual size as zero $SSize1_{act} = 0$
2: take small random sample $\Omega_{rnd} \subset \Omega$ of size RSize
3: train SVM with $[\Omega, Y]_{rnd}$ to obtain support objects
   $\Omega_{\sup} \subseteq \Omega_{rnd}$
4: update the smart sample $\Omega^{L1}_{smart} = \Omega^{L1}_{smart} \cup \Omega_{\sup}$
5: if the smart sample size is not enough
   $|\Omega_{smart}| < SSize1$ and early stopping criterion
   $|\Omega^{L1}_{smart}| - SSize1_{act} \geq \Delta_1$ does not hold true then
   upgrade $SSize1_{act} = |\Omega^{L1}_{smart}|$ and repeat steps 2-5.
```

Algorithm 3. Second-layer Smart Sample with early stopping

```
Parameters:
SSize1, SSize2 - desired sizes of first-layer and second-layer
   smart samples, respectively
RSize – size of random samples
$\Delta_1, \Delta_2$ - thresholds for early stopping at the 1-st and 2-nd
   levels, respectively

1: set the 2-layer smart sample as empty $\Omega^{L2}_{smart} = \varnothing$ and
   its actual size as zero $SSize2_{act} = 0$.
2: form the 1-layer smart sample $\Omega^{L1}_{smart}(SSize1, Rsize, \Delta_1)$
   $\subset \Omega$ (Algorithm 2)
3: train SVM with $[\Omega, Y]^{L1}_{smart}$ to obtain support objects
   $\Omega^{L1}_{\sup} \subseteq \Omega^{L1}_{smart}$
4: update the smart sample $\Omega^{L2}_{smart} = \Omega^{L2}_{smart} \cup \Omega^{L1}_{\sup}$
5: if the smart sample size is not enough $|\Omega^{L2}_{smart}| < SSize2$
   and early stopping criterion $|\Omega^{L2}_{smart}| - SSize2_{act} \geq \Delta_2$
   does not hold true then upgrade $SSize2_{act} = |\Omega^{L2}_{smart}|$
   and repeat steps 2-5.
```

5. Experiments

5.1 Data description

In experiments of this paper 3 large MNIST data sets and the KDDcup data set are used.

Originally MNIST is the data set of 60000 handwritten digit images for the training and 10000 images for the testing. In these experiments the basic data set was extended by addition synthetic images obtained as pseudo-random deformations and translations of original MNIST images. Synthetic images were generated using infiMNIST program by Leon Bottou that is available at https://leon.bottou.org/projects/infimnist.

Main characteristics of obtained data sets are presented at the Table 1.

Training sets	Objects (original / synthetic)	features
MNIST60k	60 000 (60 000 / 0)	784
MNIST200k	200 000 (60 000 / 140 000)	784
MNIST500k	500 000 (60 000 / 440 000)	784
Test set		
MNIST10k	10 000 (10 000 / 0)	784

Table 1. MNIST data sets characteristics

Feature values of all MNIST data sets have the meaning of image pixel brightness and are in the range of 0…255. Before training and testing all MNIST data are normalized by dividing by 255.

KDDcup data set contains a standard set of data to be audited, which includes a wide variety of intrusions simulated in a military network environment. It contains 4 898 431 objects for the training and 311 029 objects for test. The original KDDcup data set with data description can be downloaded at http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

Each object of KDDcup data set is initially represented by 38 numerical and 3 categorical features. Each categorical feature was encoded through the One-Hot-Encoding procedure. As the result of encoding 122 numerical features were obtained.

Originally KDDcup data set contains 5 class labels: normal (no attack) and 4 types of attacks: dos, u2r, r2l and probe. But since 3 last of them contain a very small number of samples, in this experiment we combined them into one class, thus obtaining three classes: "no attack", "dos attack" and "other types of attacks".

5.2 Computational System Characteristics

Experiments described in the paper were carried out on a computational system with the following characteristics: Intel Core i7- 9700k, RAM 16 Gb. Due to the space consumed by the operating system, the available memory that we can use is about 14 Gb. The reading speed of the disk is about 100 Mb/sec.

5.3 Related Approaches

This section describes popular approaches and open-access tools for solving the binary SVM classification problem. Each of them is considered below as the basis for construction the multiclass SVM classifier.

5.3.1 SVC and LinearSVC. First of all, we consider implementations that allow to obtain an accurate decision of the SVM problem. In this study we use implementations that are based on the traditional state-of-the-art LibSVM 0 library and included in python scikit-learn package: sklearn.svm.LinearSVC (optimized for linear problems) and universal sklearn.svm.SVC that is used for construct nonlinear decisions.

5.3.2 RBFSampler + LinearSVC. This is one of popular heuristic approaches aimed to accelerate SVM training in the nonlinear case. It combines generation of nonlinear features whose inner product is approximately equal to the kernel values (Rahimi and Recht, 2008).

We use python scikit-learn implementation here sklearn. kernel_approximation.RBFSampler with consequent well-scalable linear training by sklearn.svm.LinearSVC.

5.3.3 Smart Sampling Kernel-based Mean Decision Rule method (SS-KMDR). This approach for binary SVM problem was proposed in (Makarova et al., 2020) for fast solving both of linear and nonlinear binary SVM problems. It is based on averaging individual SVM decisions obtained for smart samples of the initial training set. In this paper we firstly extend it to the multiclass SVM problem.

5.4 Experimental Setup

Each of methods for solving binary SVM problem described in Section 4.3, was coupled with both OvR and OvO strategies for extending binary SVM to multiclass one and applied to each of training sets from the Table 1.

In all experiments we set the SVM parameter C=10 and for all nonlinear methods (including RBFSampler) the RBF kernel with $\gamma = 0.01$ was used.

For LinearSVC (as a separate classifier and after RBFSampler transformation) default parameters were set in (except of C=10).

The main parameter of RBFSampler, named *nc* (number of components that corresponds to the dimensionality of the computed feature space) was taken equal to 1000, 2000 and 3000.

For SS-KMDR method three parameters were varied: the random sample size (*rs*), the smart sample size (*sz*) and the number of smart samples (*ns*).
For the proposed Dual-Layer Smart Sample SVM (DLSS-SVM) method next parameters were varied: the random sample size (*rs*), the 1-st and 2-nd layer smart sample size (*ss1* and *ss2*, respectively). The early stopping thresholds for the 1-st and 2-nd layers were set in as 100 in all experiments.

5.5 Experimental Results

Tables 2-5 contain training and testing times and accuracy averaged through 3 runs. Standard deviations of accuracy for MNIST60k and MNIST200k does not exceed 0.001, for MNIST500k does not exceed 0.002, and for KDDcup does not exceed 0.005.

As we can see from the Tables 2-5, the accurate state-of-the-art LinearSVC method is scalable well enough. Its accuracy is higher for OvO strategy in contrast to OvR one, but absolute values are much less then SVC because binary SVM subproblems are as a rule linear inseparable and nonlinear decision functions are required.

The accurate state-of-the-art SVC method is nonlinear, but despite the use of a number of heuristics to speed up the work with kernels, its scalability is very low and for MNIST500k the training time for OvR strategy as well as for KDDcup for both OvR and OvO strategies exceeds 10 hours. Applying OvO strategy essentially reduces the training time due to decreasing size of binary problems, but the absolute values remain to be very large.

An attempt to replace work with kernels with a specially generated feature space using RBFSampler allows to reduce training time, but leads to loss of information and consequent a noticeable loss of accuracy.

Method		Time (s)		Accuracy
		train	test	
SVC OvR		2738	101.90	0.9735
SVC OvO		203.50	232.50	0.9694
LinearSVC OvR		79.75	0.44	0.8744
LinearSVC OvO		21.99	1.28	0.8839
RBFSampler + LinearSVC	*nc* = 1000 OvO	84.65	0.9284	0.9284
	nc = 1000 *OvR*	78.79	14.67	0.9399
	nc = 2000 OvO	142.71	1.70	0.9471
	nc = 2000 *OvR*	113.09	30.85	0.9560
	nc = 3000 OvO	249.44	2.79	0.9547
	nc = 3000 *OvR*	180.89	49.50	0.9590
SS-KMDR	*rs*=*ss*=3000 ns = 1	56.46	94.95	0.9656
	rs=*ss*=6000 *ns* = 1	333.42	148.53	0.9720
	rs=*ss*=10000 *s* = 1	1050.45	167.10	0.9725
	rs=*ss*=10000 *s* = 2	1980.14	310.17	0.9728
DLSS-SVM	*rs*=3000 *ss1*=*ss2*=1000	51.20	77.02	0.9607
	rs=3000 *ss1*=*ss2*=2000	151.20	109.41	0.9679
	rs=5000 *ss1*=*ss2*=3000	476.82	146.03	0.9726
	rs=6000 *ss1*=1000 *ss2*=10000	874.07	288.25	0.9736

Table 2. Experimental results for MNIST60k

Method		Time (s)		Accuracy
		train	test	
SVC OvR		20688	302.2	0.987
SVC OvO		2686	504.9	0.9854
LinearSVC OvR		1098	0.184	0.8869
LinearSVC OvO		297.6	0.639	0.9144
RBFSampler + LinearSVC	*nc* = 1000 OvO	320.6	1.73	0.9487
	nc = 1000 *OvR*	282.7	10.81	0.9545
	nc = 2000 OvO	521.47	1.75	0.9631
	nc = 2000 *OvR*	509.34	31.73	0.9672
	nc = 3000 OvO	787.28	2.68	0.9694
	nc = 3000 *OvR*	688.08	45.87	0.9743
SS-KMDR	*rs*=*ss*=3000 ns = 1	101.45	110.37	0.9737
	rs=*ss*=6000 *ns* = 1	321.16	242.12	0.9800
	rs=*ss*=10000 *s* = 1	1035.12	286.59	0.9837
	rs=*ss*=10000 *s* = 2	2017.92	466.15	0.9838
DLSS-SVM	*rs*=3000 *ss1*=*ss2*=1000	80.00	87.59	0.9673
	rs=3000 *ss1*=*ss2*=2000	164.05	133.20	0.9744
	rs=5000 *ss1*=*ss2*=3000	304.71	194.08	0.9811
	rs=6000 *ss1*=1000 *ss2*=10000	854.75	280.09	0.9839

Table 3. Experimental results for MNIST200k

Both SS-KMDR and DLSS-SVM, firstly proposed here for multiclass classification, allow to reach accuracy values near SVC and essentially outperform LinearSVC and RBFSamler+LinearSVC in accuracy. At that they are essentially more computationally efficient in contrast to SVC. As we can see from the Table 4, MNIST500k data set, 14x speedup is achieved for SS-KMDR and almost 20x speedup for DLSS-SVM compared to state-of-the-art SVC method.

Moreover, the advantage in the training speed grows exponentially with increasing data volume.

Method		Time (s)		Accuracy
		train	test	
SVC OvR		>36000	-	-
SVC OvO		17133	784.5	0.9876
LinearSVC OvR		2457	0.174	0.8839
LinearSVC OvO		729.5	0.641	0.916
RBFSampler + LinearSVC	*nc* = 1000 OvO	782.8	1.958	0.9464
	nc = 1000 *OvR*	750.41	1.047	0.9478
	nc = 2000 OvO	1616.53	1.999	0.9647
	nc = 2000 *OvR*	1407.64	3.769	0.9641
	nc = 3000 OvO	2548.19	2.243	0.9693
	nc = 3000 *OvR*	2266.32	3.371	0.9716
SS-KMDR	*rs*=*ss*=3000 ns = 1	132.84	114.22	0.9737
	rs=*ss*=6000 *ns* = 1	331.00	191.71	0.9797
	rs=*ss*=10000 *ns* =1	1216.11	327.24	0.9843
	rs=*ss*=10000 *s* = 2	2440.73	410.88	0.9844
DLSS-SVM	*rs*=3000 *ss1*=*ss2*=1000	135.05	99.71	0.9719
	rs=3000 *ss1*=*ss2*=2000	250.08	142.41	0.9744
	rs=5000 *ss1*=*ss2*=3000	379.87	184.67	0.9781
	rs=6000 *ss1*=1000 *ss2*=10000	861.32	308.2	0.9853

Table 4. Experimental results for MNIST500k

Method		Time (s)		Accuracy
		train	test	
SVC OvR		>36000	-	-
SVC OvO		>36000	-	-
LinearSVC OvR		3335.13	0.308	0.7967
LinearSVC OvO		860.58	0.624	0.7840
RBFSamp+ LinearSVC	*nc* = 100 OvO	822.26	3.399	0.8620
	nc = 100 *OvR*	447.54	1.686	0.8506
	nc > 100 OvO	Internal error		
	nc > 100 *OvR*			
SS-KMDR	*rs*=*ss*=3000 *ns* = 1	455.19	83.89	0.883
	rs=*ss*=6000 *ns* = 1	604.31	133.49	0.888
	rs=*ss*=10000 *s* = 1	835.17	206.34	0.888
	rs=*ss*=10000 *s* = 2	1715.22	245.11	0.889
DLSS-SVM	*rs*=3000 *ss1*=*ss2*=2000	11.98	14.69	0.830
	rs=5000 *ss1*=*ss2*=3000	475.63	91.68	0.891
	rs=6000 *ss1*=2000 *ss2*=10000	1740.70	192.40	0.893

Table 5. Experimental results for KDDcup

Though in a number of cases (for some parameters) SS-KMDR and DLSS-SVM allows to reach similar accuracy values, but DLSS-SVM is faster in contrast to SS-KMDR and so gives the possibility to train in larger data sets. Besides, it should be noted training and testing times for both of proposed algorithms can be additionally decreased via using parallel and distributed computing technologies. So, the proposed algorithms improve multiclass SVM scalability.

Figures 1 and 2 present training times (in seconds) and accuracy values respectively for main methods under compare for MNIST data sets. It should be noted, that the upper row (for SVC) does not fully displayed due to very large value that equals to 17133 second.

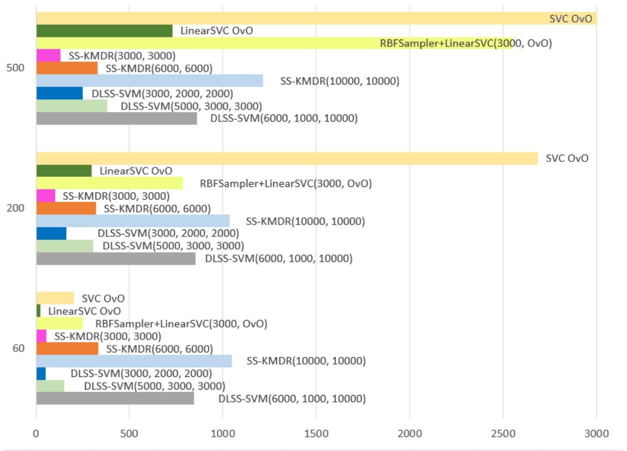

Figure 1. Training times (s) for main methods under compare.

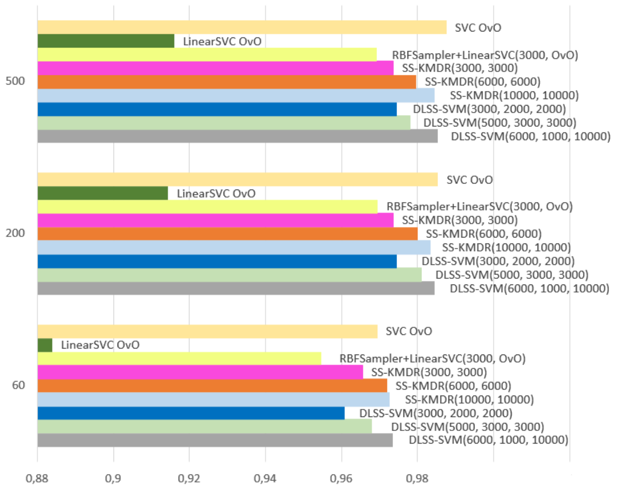

Figure 2. Accuracy for main methods under compare.

Conclusion

In this paper we extend the existing SS-KMDR method that was initially proposed for binary SVM problems to efficient solve large multiclass SVM problems. In addition, we upgrade the algorithm to form smart samples that underlies SS-KMDR and a new Dual-Layer Smart Sample SVM (DLSS-SVM) method for large-scale multiclass problems was proposed on the basis of the respective new smart sampling approach.

Both SS-KMDR and DLSS-SVM, firstly proposed here for multiclass classification, allow to reach accuracy values near state-of-the-art SVC method (that is based on libsvm) and essentially outperform LinearSVC and RBFSamler+LinearSVC in accuracy. At that they are essentially more computationally efficient in contrast to SVC. So, for MNIST500k data set, 14x speedup is achieved for SS-KMDR and almost 20x speedup for DLSS-SVM compared to state-of-the-art SVC method.

The proposed approaches give the possibility to train in large data sets, have inner data parallelism, allow to reach near state-of-the-art libsvm quality in much less time in contrast to SVC. So, both of them improve multiclass SVM scalability, but DLSS-SVM is more efficient in the training and test stages.

Acknowledgements

This research is funded by the Ministry of Science and Higher Education of the Russian Federation within the framework of the state task FEWG-2024-0001.

References

Agarwal, A., Duchi, J.C., 2011. Distributed delayed stochastic optimization. *Advances in Neural Information Processing Systems*, 24, 873-881. doi.org/10.48550/arXiv.1104.5525.

Aizerman, M.A., et al., 1970. Potential functions method in machine learning theory (in Russian). Nauka, Moscow, Russia.

Benfenati, A., Chouzenoux, E., Franchini, G., Latva-Aijo, S., Narnhofer, D., Pesquetm, J.-C., Scott, S.J., Yousefi, M., 2023. Majorization-Minimization for sparse SVMs. doi.org/10.48550/arXiv.2308.16858.

Boser, B.E., Guyon, I.M., Vapnik, V.N., 1992. A Training Algorithm for Optimal Margin Classifiers, *Fifth Annual Workshop on Computational Learning Theory*, Association for Computing Machinery, New York, USA, 144–152. doi.org/10.1145/130385.130401.

Bottou, L., 2004. Stochastic Learning. Advanced Lectures on Machine Learning, *Lecture Notes in Artificial Intelligence*, 3176, 146-168, Springer Verlag, Berlin.

Burkart, N., Huber, M.F., 2021. A Survey on the Explainability of Supervised Machine Learning. *Journal of Artificial Intelligence Research (JAIR)*, 70, 245-317. doi.org/10.1613/jair.1.12228.

Byrd, R.H., Hansen, R.H., Nocedal, J., Singer, Y., 2016. A stochastic quasi-newton method for large-scale optimization. *SIAM Journal on Optimization*, 26(2), 1008-1031. doi.org/10.1137/140954362.

Chang, C.-C., Lin, C.-J., 2001. LIBSVM: a library for support vector machines. Open Source Software. http://www.csie.ntu.edu.tw/ cjlin/libsvm (06.08.2024).

Chu, C.-T., Kim, S.K., Lin, Y.-A., Yu ,Y., Bradski, G., Ng, A.Y., Olukotun, K., 2006. Map reduce for machine learning on multicore. Advances in neural information processing systems, 19, 281-288.

Charmet, F., Tanuwidjaja, H.C., Ayoubi, S. et al., 2022. Explainable artificial intelligence for cybersecurity: a literature survey. *Ann. Telecommun*, 77, 789–812. doi.org/10.1007/s12243-022-00926-7.

Chauhan, V.K., Sharma, A., Dahiya, K., 2018. Faster learning by reduction of data access time. *Applied Intelligence*, 48(12), 4715-4729. doi.org/10.1007/s10489-018-1235-x.

Collobert, R., Bengio, S., 2001. SVMTorch: Support vector machines for large-scale regression problems. *Journal of machine learning research*. 1, 143-160.

Csiba, D., Richt, P., 2016. Importance Sampling for Minibatches. doi.org/10.48550/arXiv.1602.02283.

Dekel, O., Gilad-Bachrach, R., Shamir, O., Xiao, L., 2012. Optimal distributed online prediction using mini-batches. *Journal of Machine Learning Research*, 13(1), 165-202. doi.org/10.48550/arXiv.1012.1367.

Duan, K.B., Keerthi, S.S., 2005. Which Is the Best Multiclass SVM Method? An Empirical Study. *Multiple Classifier Systems. MCS 2005. Lecture Notes in Computer Science*, 3541, 278-285. doi.org/10.1007/11494683_28.

Drineas, P., Mahoney, M.W., 2005. Approximating a Gram Matrix for Improved Kernel-Based Learning. *Learning Theory. Lecture Notes in Computer Science*, 3559, 323-337. doi.org/10.1007/11503415_22.

Emmert-Streib, F., Yang, Z., Feng, H., Tripathi, S., Dehmer, M., 2020. An Introductory Review of Deep Learning for Prediction Models With Big Data. *Front Artif. Intell*. 3(4). doi.org/10.3389/frai.2020.00004.

Hoi, S., Sahoo, D., Lu, J., Zhao, P., 2018. Online Learning: A Comprehensive Survey. *Neurocomputing*, 459, 249-289. doi.org/10.1016/j.neucom.2021.04.112.

Hsu, C.-W., Lin, C.-J., 2002. A simple decomposition method for support vector machines. *Machine Learning*, 46, 291–314, 2002. doi.org/10.1023/A:1012427100071.

Joachims, T., 1999. Making large-scale support vector machine learning practical. *Advances in kernel methods: support vector learning*, MIT Press, Cambridge, MA, USA, 169–184.

Joachims, T., 2006. Training linear SVMs in linear time. *12th ACM SIGKDD international conference on Knowledge discovery and data mining*, 217–226. doi.org/10.1145/1150402.1150429.

Le, T., Nguyen, T.D., Nguyen, V., Phung, D.Q., 2017. Approximation Vector Machines for Large-scale Online Learning. *Journal of Machine Learning Research*, 18(111), 1-55.

Makarova, A.I., Sulimova V.V., 2019. Fast approximate two-class SVM learning for large training sets. *Procs. of Int. Conference Information Technology and Nanotechnology* (In Russian), 21-24.

Makarova, A.I., Kurbakov, M.Y., Sulimova, V.V., 2020. Mean Decision Rules Method with Smart Sampling for Fast Large-Scale Binary SVM Classification. *2020 25th International Conference on Pattern Recognition*, 8212-8219, doi.org/10.1109/ICPR48806.2021.9412232.

Malenichev, A., Krasotkina, O., Mottl, V., Seredin, O., 2016. Multi-class Classification in Big Data. *Fifth International Conference on Analysis of Images, Social Networks and Texts (AIST 2016)*, 1710, 203-211.

Niu, F., Recht, B., Re, C., Wright, S.J., 2011. HOGWILD: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent. *Advances in Neural Information Processing Systems*, 21(1). doi.org/10.48550/arXiv.1106.5730.

Platt, J., 1998. Sequential minimal optimization: A fast algorithm for training support vector machines. *Technical Report MSR-TR-98-14*. Microsoft Research.

Rahimi, A., Recht, B., 2007. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 1177–1184.

Rahimi, A., Recht, B., 2008. Weighted Sums of Random Kitchen Sinks: Replacing minimization with randomization in learning. *Advances in neural information processing systems*, 21, 1313-1320.

Rivas-Perea, P., Cota-Ruiz, J., 2013. An Algorithm for Training a Large Scale Support Vector Machine for Regression Based on Linear Programming and Decomposition Methods, *Pattern Recognition Letters*, 34(4), 439-451. doi.org/10.1016/j.patrec.2012.10.026.

Rizzi, A.M., 2016. Support vector regression model for BigData systems. doi.org/10.48550/arXiv.1612.01458.

Sadrfaridpour, E., Razzaghi, T., Safro, I., 2019. Engineering fast multilevel support vector machines. *Mach Learn*, 108, 1879-1917. doi.org/10.1007/s10994-019-05800-7.

Sleeman, W.C., Krawczyk, B., 2021. Multi-class imbalanced big data classification on Spark. *Knowledge-Based Systems*, 212. doi.org/10.1016/j.knosys.2020.106598.

Vapnik, V.N., 1995. The nature of statistical learning theory. Springer-Verlag New York, Inc.

Wen, Z., Shi, J., Li, Q., He, B., Chen, J., 2018. ThunderSVM: a fast SVM library on GPUs and CPUs. *The Journal of Machine Learning Research*, 19(21), 1-5.

Zhao, H.X., Magoules, F., 2011. Parallel support vector machines on multi-core and multiprocessor systems. *11-th International Conference on Artificial Intelligence and Applications*, doi.org/10.2316/P.2011.717-056.

Zhao, P., Zhang, T., 2014. Accelerating Minibatch Stochastic Gradient Descent using Stratified Sampling. doi.org/10.48550/arXiv.1405.3080.

Zhu, Z.A., Chen, W., Wang, G., Zhu, C., Chen, Z., 2009. P-packSVM: Parallel primal gradient descent kernel SVM. *Ninth IEEE International Conference on Data Mining*, 677-686. doi.org/10.1109/ICDM.2009.29.

You, Y., Demmel, J., Czechowski, K., Song, L., Vuduc, R., 2015. CA-SVM: Communication-Avoiding Parallel Support Vector Machines on Distributed Systems. *2015 IEEE International Parallel and Distributed Processing Symposium*, 847-859. doi.org/10.1109/IPDPS.2015.117.