The 25th International Conference on Pattern Recognition

January, 10 – 15th, 2021, Italy, Milan

Mean Decision Rules Method with Smart Sampling for Fast Large-Scale Binary SVM Classification

***Alexandra Makarova*,**
Mikhail Kurbakov, Valentina Sulimova
vsulimova@yandex.ru

Tula State University
Tula, Russia
Laboratory of Data Analysis

Binary SVM Classification Problem

Training set: $[\Omega, Y]$, $\Omega = [\omega_j, j = 1,\ldots,|\Omega|]$,

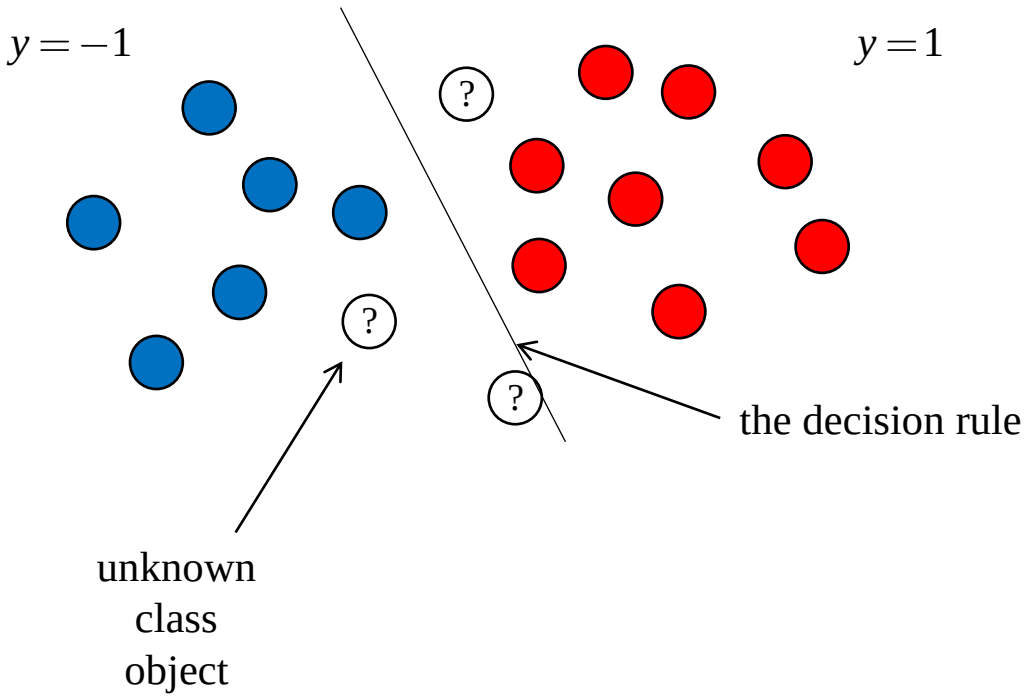
$$Y = [y_j = y(\omega_j), j = 1,\ldots,|\Omega|]$$

The decision rule in the form of a linear separating hyperplane:

$$d(\omega|\mathbf{a},b) = \mathbf{a}^T\omega + b \quad \begin{array}{l} \geq 0 \Rightarrow \hat{y}(\omega) = +1, \\ < 0 \Rightarrow \hat{y}(\omega) = -1, \end{array}$$

$\mathbf{a} \in \mathcal{X}$ - direction element

$b \in R$ - bias



Many approaches to solving the SVM problem

1. Zhang, Tong. Solving large scale linear prediction problems using stochastic gradient descent algorithms. In ICML, 2004.
2. Konecˇny´, Jakub and Richta´rik, Peter. Semi Stochastic Gradient Descent Methods. 2013. URL http://arxiv.org/abs/1312.1666.
3. Chauhan, Vinod Kumar, Dahiya, Kalpana, and Sharma Anuj. Mini-batch block-coordinate based stochastic av-erage adjusted gradient methods to solve big data prob-lems. In 9th ACML, 2017.
4. L. Bottou "Stochastic Learning", Advanced Lectures on Machine Learning, 146-168, Edited by Olivier Bousquet and Ulrike von Luxburg, LNAI 3176, Springer Verlag, Berlin, 2004
5. B. E. Boser, I. M. Guyon, V. N. Vapnik, "Training Algorithm for Optimal Margin Classifiers", Fifth Annual Workshop on Computational Learning Theory, ACM, 1992.
6. J. Platt, "Sequential minimal optimization: A fast algorithm for training support vector machines", (Technical Report MSR-TR-98-14). Microsoft Research, 1998.
7. Li, Mu, Zhang, Tong, Chen, Yuqiang, and Smola, Alexander J. Efficient mini-batch training for stochastic optimization. In KDD, 2014.
8. R.H. Byrd, S.L. Hansen, J. Nocedal, Y. Singer. A stochastic quasi-newton method for large-scale optimization. SIAM Journal on Optimization, 26(2):1008–1031, 2016.
9. D. Csiba, P. Richt. Importance Sampling for Minibatches. 2016. https://arxiv.org/abs/1602.02283.
10. P. Zhao, T. Zhang. Accelerating Minibatch Stochastic Gradient Descent using Stratified Sampling. 2014. http://arxiv.org/abs/1405.3080.
11. S. Gopal. Adaptive Sampling for SGD by Exploiting Side Information. Icml, 2016.
12. Yu, H.-F., Hsieh, C.-J., Chang, K.-W., Lin, C.-J., "Large Linear Classification When Data Cannot Fit In Memory", ACM Trans. Knowl. Discov. Data. 23 pages, 2011, DOI 10.1145/0000000.0000000
13. C. Li, "Training and Predicting Criteo's Terascale Data Set on a Machine with 128GB RAM", 2017.
14. E. Sadrfaridpour, T., Razzaghi, I Safro. Engineering fast multilevel support vector machines. Mach Learn 108, 1879–1917 (2019). https://doi.org/10.1007/s10994-019-05800-7
15. H.P. Graf, E. Cosatto, L. Bottou, I. Durdanovic, and V. Vapnik, "Parallel Support Vector Machines: The Cascade SVM," Advances in Neural Information Processing Systems, 17, 521–528, 2005.
16. O. Meyer, B. Bischl, and C. Weihs, "Support Vector Machines on Large Data Sets: Simpsle Parallel Approaches," In M. Spiliopoulou, L. Schmidt-Thieme, and R. Janning, ed., Data Analysis, Machine Learning and Knowledge Discovery etc., pp. 87–95, 2014.
17. etc.

SVM implementations

	LibSVM	**liblinear**	**MPI liblinear**	**lightSVM**	**SvmSgd**	**πSVM**	**CuSVM**	**GPU – accelerated LibSVM**	**Thunder SVM**	**KMDR**
programming language	C	C	C+MPI	C	C	C+MPI	C+CUDA	C+CUDA	C	C+MPI MATLAB
Interface	MATLAB python, R SciLAB, Java, и др.	MATLAB python, R SciLAB, Java, и др.	–	–	python	–	MATLAB	–	MATLAB, python, R, scikitsLearn	–
OS	cross-platform	cross-platform	Unix	cross-platform	cross-platform	Unix	Windows	cross-platform	cross-platform	cross-platform
Feature-based learning	+	+	+	+	+	+	+	+	+	+
Kernel-based learning	+	–	–	+	–	+	+	+	+	+
Work with sparse matrix	–	+	+	+	–	–	–	–	–	+
Work on 1 CPU	+	+	+	+	+	+	–	–	+	+
Work on several CPU	–	–	+	–	–	+	–	–	+	+
Work on GPU	–	–	–	–	–	–	+	+	+	–
Software hardware independency	+	+	–	+	+	–	–	–	–	+
no iteration or iteration without data dependencies	–	–	–	–	–	–	–	–	–	+

Kernel-based Mean Decision Rule (KMDR) method[1]

The main idea:

Random subsamples of the initial training set: $[\Omega, Y]^{(i)} \subset [\Omega, Y], i = 1, \ldots, k$

The mean decision rule: $d(\omega \mid \mathbf{a}, b) = \frac{1}{k} \sum_{i=1}^{k} d(\omega \mid \mathbf{a}^{(i)}, b^{(i)})$

Feature-based learning : $\mathbf{a} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{a}^{(i)}, \; b = \frac{1}{k} \sum_{i=1}^{k} b^{(i)}.$

Kernel-based learning:

An individual decision rule for *i-th* random subsample:

$$d(\omega \mid \boldsymbol{\lambda}^{(i)}, b^{(i)}) = \sum_{j=1}^{|\Omega^{(i)}|} \tilde{\lambda}_j^{(i)} y_j K(\omega_j, \omega) + b^{(i)}, \; i = 1, \ldots, k$$

$\tilde{\lambda}_t^{(i)}, t = 1, \ldots, |\Omega^{(i)}|$ - Lagrange multipliers

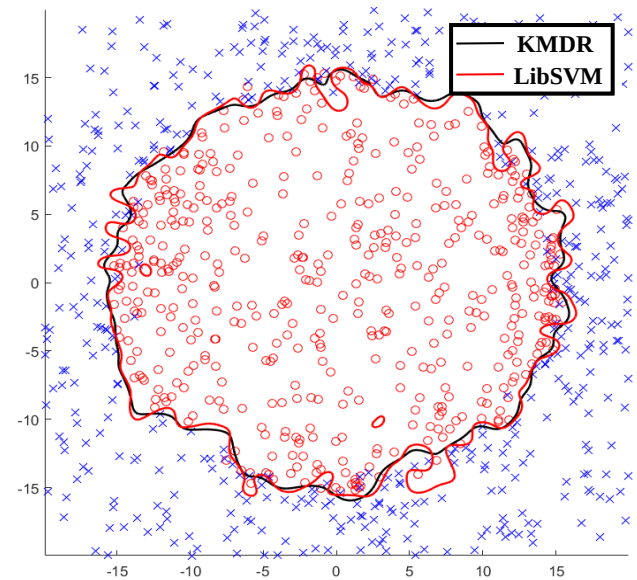
Multipliers for all training set:

$$\lambda_j^{(i)} = \begin{cases} \tilde{\lambda}_{t(j)}^{(i)}, \; t(j) \in \{1 \ldots |\Omega^{(i)}|\} \; \textit{if} \; \omega_j \in \Omega^{(i)}, \\ 0, \; \textit{otherwise}, \end{cases} \quad j = 1, \ldots, |\Omega|.$$

Mean decision rule:

$$d(\omega; \boldsymbol{\lambda}, b) = \sum_{j=1}^{|\Omega|} \lambda_j y_j K(\omega_j, \omega) + b \qquad \lambda_j = \frac{1}{k} \sum_{i=1}^{k} \lambda_j^{(i)}, \; b = \frac{1}{k} \sum_{i=1}^{k} b^{(i)}$$

KMDR's boundary is very similar to LibSVM's one



[1] A. Makarova, M. Kurbakov, V. Sulimova, "Mean Decision Rule Method for Constructing Nonlinear Boundaries in Large Binary SVM Problems", IEEE Proc., 2020, URL https://yadi.sk/i/VznTrSMuSs1sig (In print).

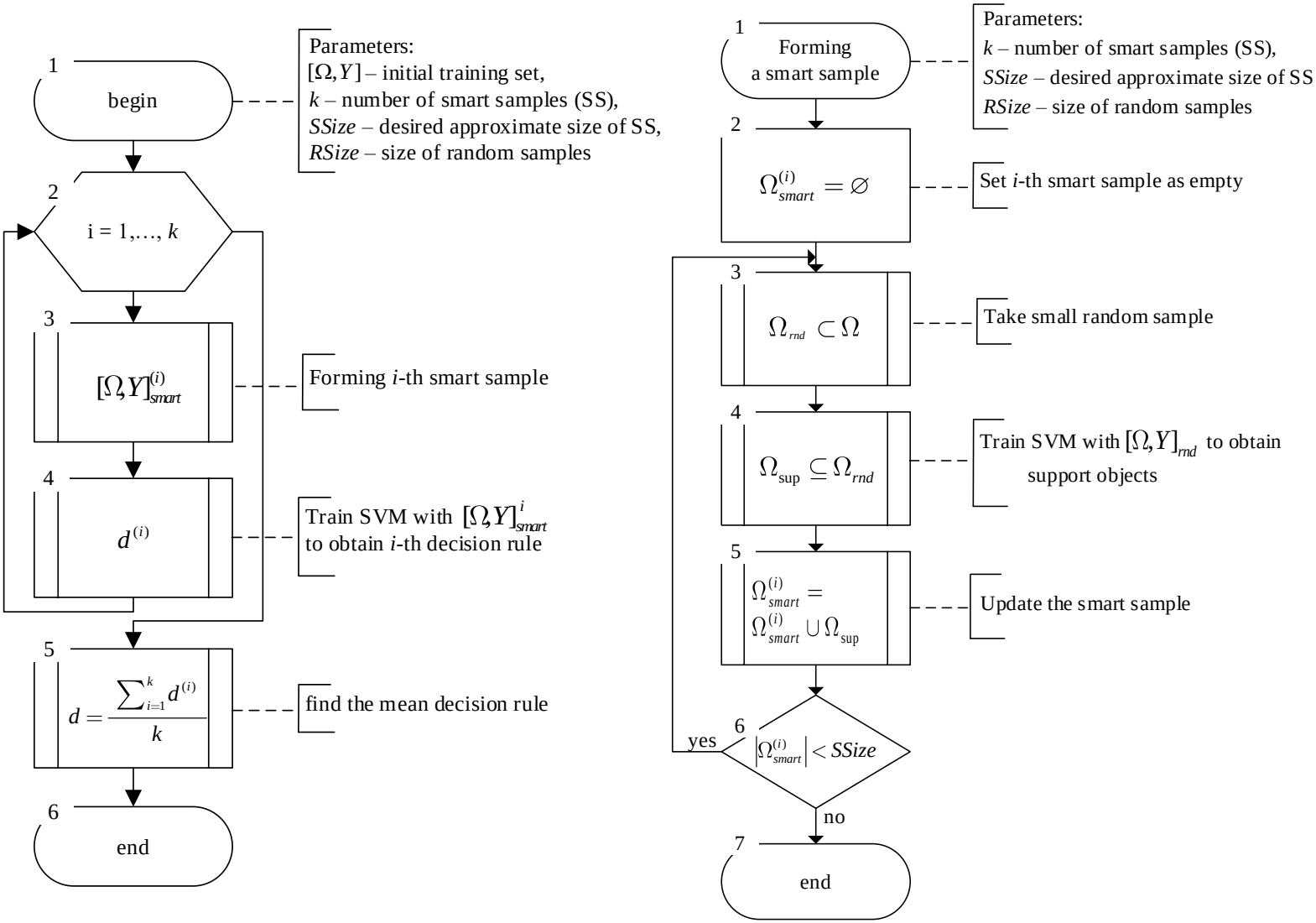
Contribution

Two new approaches to improve the KMDR:

1) **a new sampling technique (Smart Sampling)**,
that exploits SVM and MDR properties to fast select only those objects that are candidates to be the support ones.

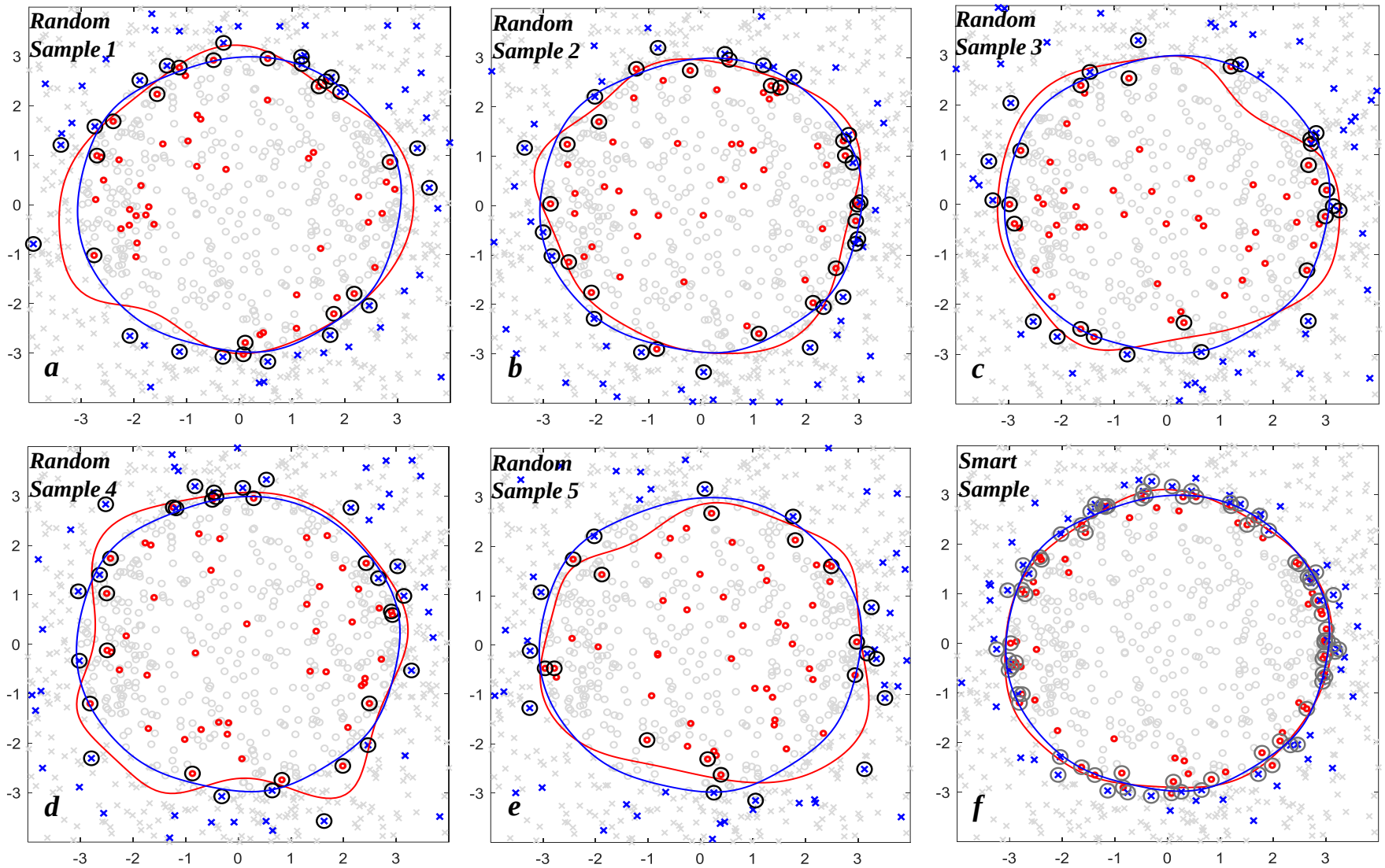
2) **a new data strategy**
to accelerate random access to large datasets stored in the traditional libsvm format.

KMDR with Smart Sampling (SS-KMDR)[1]



[1] A. Makarova, M. Kurbakov, V. Sulimova, “Mean Decision Rules Method with Smart Sampling for Fast Large-Scale Binary SVM Classification”, IEEE, International Conference on Pattern Recognition 2020 (In print).

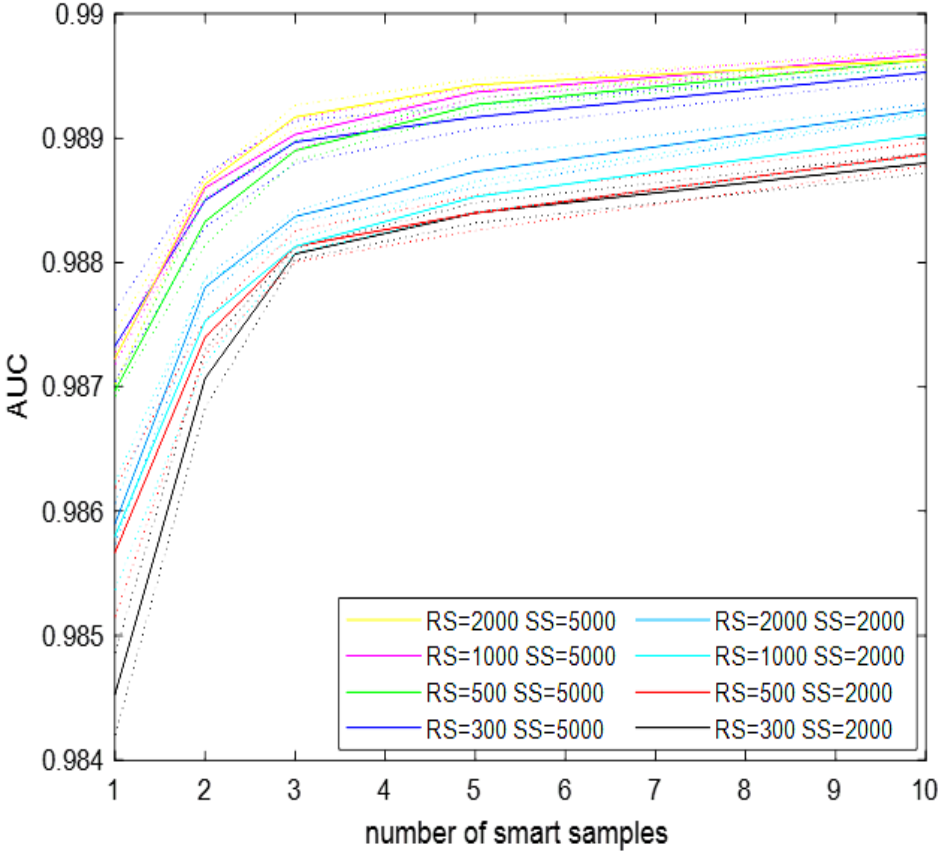
Training results for random and smart samples vs full set

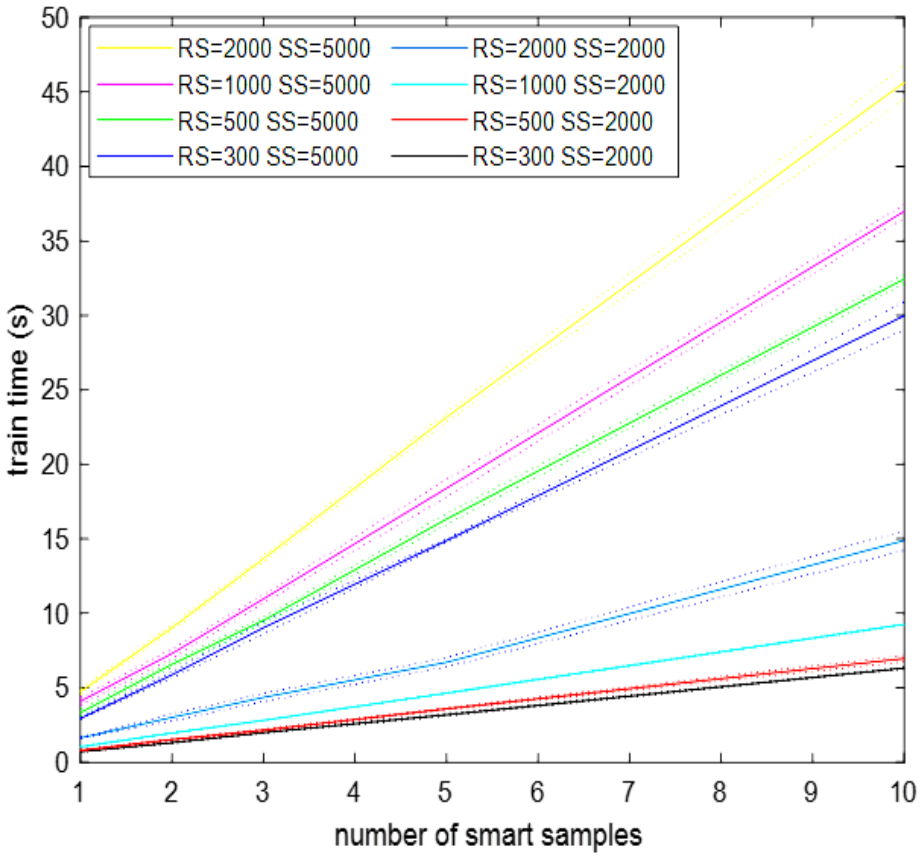


decision rule for the full training set

decision rules for random samples (a-e) and for the smart sample (f)

Investigation of the effect of MDR parameters onto its performance





RS – the number of objects in each random sample

SS – the number of objects in each smart sample

Loading data optimization

The LibSVM data format

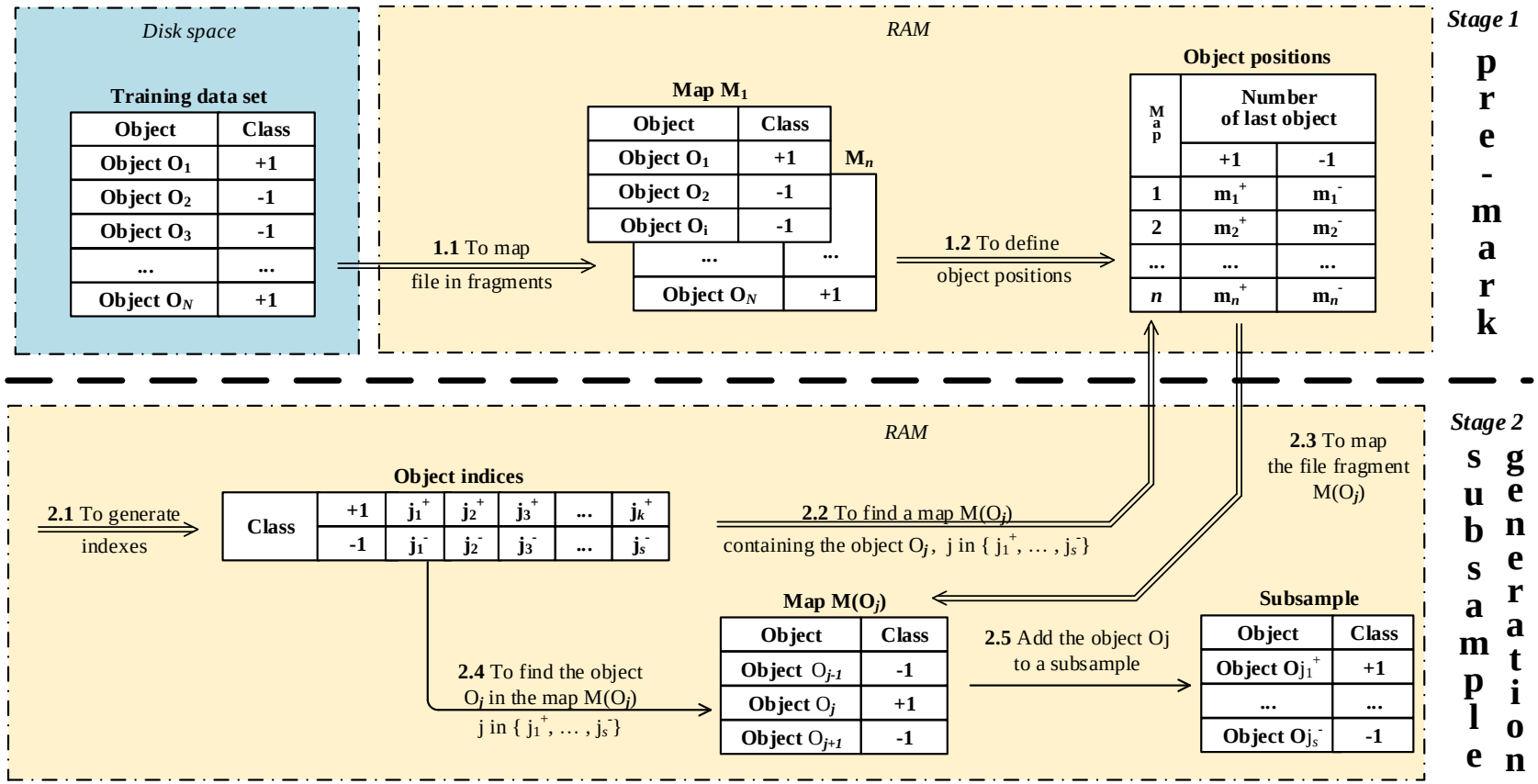
Each row corresponds to the information about 1 object:

```
<class label> <number of feature>:<value> ... <number of feature>:<value>
```

+ *It allows to store the sparse data in a compressed form*

− *it is impossible to compute where some object starts*

The proposed data strategy



Experimental setup

Statistics of benchmark datasets

Dataset		Number of objects		Features
		Train	*Test*	
Artificial	Set 1	10 000	10 000	100
	Set 2	100 000	100 000	10
	Set 3	100 000	100 000	100
Real	ijcnn1	35 000	91 701	22
	mnist-576	60 000	10 000	576
	mnist-784	60 000	10 000	784
	kddcup99	4 898 430	311 029	122

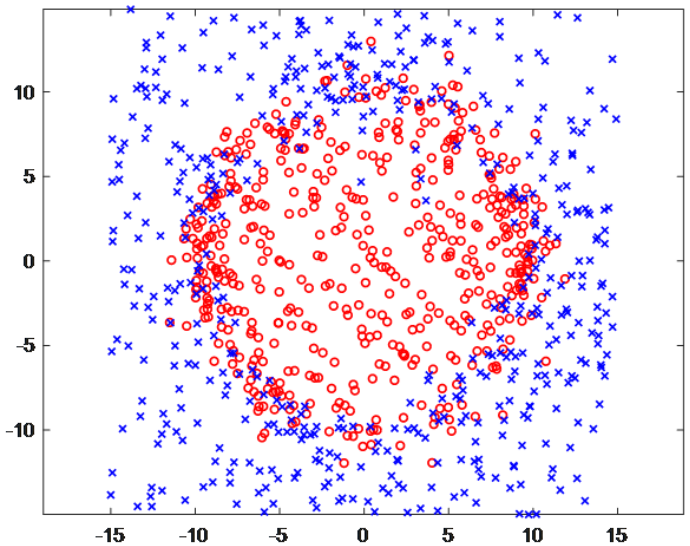
Characteristics of the computational system

Intel Core i7-9700k
RAM 16 Gb (14 Gb is available)
The reading speed: 100 Mb/sec

Artificial data generation model

- Objects of two classes form intersecting sets,
- Positive objects:
  -- uniform distribution inside a hypersphere
  -- outside their quantity decreases exponentially with increasing of the distance from the hypersphere
- Negative objects:
  -- uniform distribution outside the hypersphere
  -- inside their number decreases exponentially with increasing the distance from the hypersphere.

An example of artificial dataset



Investigation of the proposed data strategy

Statistics of benchmark datasets

Dataset	Number of objects		Features	Non-zeros	Size of memory (bytes)
	Train	***Test***			
mnist-784-poly-8vr	60 000	10 000	784	33 878 116	542 049 856
kddcup99-nvr	4 898 430	311 029	122	65 023 484	1 040 375 744

Results for simple KMDR

datasets	Stage	Size of a random sample		
		500	***1000***	***5000***
mnist784	Mark-up/ data loading	0.284/7.526	0.291/7.482	0.289/7.501
	Forming samples	0.924/0.002	1.834/0.008	9.241/0.144
	Training	0.534/0.565	1.912/2.070	37.85/42.74
	Total	1.744/8.094	4.038/9.561	47.38/50.39
kddcup99	Mark-up/ data loading	0.613/11.71	0.618/11.69	0.620/11.71
	Forming samples	0.035/0.004	0.068/0.010	0.405/0.124
	Training	0.014/0.017	0.039/0.047	0.382/0.538
	Total	0.663/11.73	0.727/11.75	1.407/12.37

The proposed data strategy is especially efficient for large sparse datasets (such as kddcup99)

Comparison results for artificial datasets

Method	dataset (N objects/N features)								
	1 (10 000/100)			*2 (100 000/10)*			*3 (100 000/100)*		
	time (s)		*AUC*	*time (s)*		*AUC*	*time (s)*		*AUC*
	*train**	*test*		*train**	*test*		*train**	*test*	
LibSVM	93,5	32,2	0,977	3373	216,5	0,99	4371	1053	0,985
πSVM	13,9	27,8	0,976	1130	284,6	0,99	3724	2269	0,984
SVMlight	42,8	14,9	0,977	1874	203,1	0,99	8569	1873	0,985
Bagging-1	5,1	3,4	0,951	78,5	23,3	0,97	308,3	264,2	0,969
Bagging-2	8,8	6,7	0,954	156	46,4	0,971	668,6	559,1	0,969
Bagging-5	20,3	6,7	0,953	387	117,6	0,972	1528	1369	0,97
KMDR50	0,06	1,9	0,954	0,03	2,8	0,969	0,2	20,3	0,958
KMDR500	0,9	8,03	0,972	0,2	12,8	0,983	1,1	107,1	0,976
KMDR1000	2,6	9,7	0,975	0,6	21,4	0,986	2,7	182,8	0,979
KMDR2000	7,5	10,5	0,977	1,8	36,4	0,988	7,8	303,6	0,981
for random sample size of 50									
SS-KMDR1000	0,2	1,24	0,98	0,09	1,54	0,988	0,26	11,9	0,982
SS-KMDR2000	0,33	1,49	0,981	0,12	2,11	0,99	0,49	15,9	0,985
SS-KMDR5000	1,2	2,2	0,984	0,44	3,51	0,992	1,97	26,5	0,986
for random sample size of 500									
SS-KMDR1000	0,23	1,36	0,981	0,08	1,88	0,989	0,33	14,2	0,983
SS-KMDR2000	0,51	1,77	0,983	0,16	2,65	0,991	0,73	19,61	0,985
SS-KMDR5000	1,59	2,54	0,984	0,63	4,81	0,992	3,29	34,3	0,987

KMDR#### : #### - RS (random sample) size, number of RS = 20
SS-KMDR#### : #### - SS (smart sample) size, number of SS = 1

* training time includes time of data loading

best ***worse***

AUC

time

Comparison results for benchmark datasets

method	dataset							
	ijcnn1		*mnist576*		*mnist784*		*kddcup99*	
	*time(s)**	*acc*	*time(s)**	*acc*	*time(s)**	*acc*	*time(s)**	*acc*
LibSVM	15,18	98,12	324,2	99,85	396,74	99,15	>3600	-
PiSVM	14,31	98,12	110,9	99,85	470,5	99,15	mem. Error	-
Liblinear	0,32	91,3	9,54	98,92	19,35	92,08	25,94	92,08
Bagging-1	4,33	98,48	79,46	99,81	157,65	99,47	2284	92,74
Bagging-2	7,54	98,35	139,6	99,83	305,48	99,52	>3600	-
Bagging-5	21,25	98,46	319,7	99,86	794,67	99,49	>3600	-
SGD	0,337	91,24	17,35	94,44	20,07	94,39	60,67	91,96
ASGD	0,342	91,4	17,46	95,13	20,21	94,47	59,67	91,95
KMDR500	0,21	92,76	5,27	99,17	1,74	94,95	0,663	92,14
KMDR1000	0,35	93,12	7,62	99,45	4,04	95,76	0,727	92,08
KMDR5000	3,48	96,3	43,83	99,67	47,28	97,88	1,407	91,98
SS-KMDR2000	0,19	97,31	3,04	99,71	2,66	98,46	0,95	92,12
SS-KMDR3000	0,31	97,63	4,45	99,76	4,25	98,72	1,06	92,14
SS-KMDR5000	0,59	97,93	7,82	99,82	7,84	98,91	1,41	92,17

KMDR#### : #### - RS (random sample) size, number of RS = 50

SS-KMDR#### : #### - SS (smart sample) size, number of SS = 1, RS size = 500

best ***worse***

accuracy

time

* training time includes time of data loading

Thank you for your attention!