Make it cheap: learning with $O(nd)$ complexity

Włodzisław Duch and Norbert Jankowski and Tomasz Maszczyk

***Abstract*—Learning methods with linear computational complexity $O(nd)$ in number of samples and their dimension often give results that are better or at least not worse that more sophisticated and slower algorithms. This is demonstrated for many benchmark datasets downloaded from the UCI Machine Learning Repository. Results provided in this paper should be used as a reference for estimating usefulness of new learning algorithms. Methods with higher than linear complexity should provide significantly better results than those presented in this paper to justify their use.**

I. Introduction

Many sophisticated machine learning methods are introduced every year and tested on relatively trivial benchmark problems from the UCI Machine Learning Repository [1]. Most of benchmark problems found there are relatively easy: simple and fast algorithms with $O(nd)$ complexity (n being the number of training samples and d dimensionality of the input vectors) give results that are not statistically significant worse than those obtained by the best known algorithms. Some benchmark problems are not trivial, have complicated decision borders and require sophisticated techniques, including specialized kernels, multiresolution, deep learning, transfer learning, committees or meta-learning [2]. Of course one may test new methods on a simple benchmark data, but to show that they are really an improvement over existing low-complexity machine learning methods they also should be tested on non-trivial data.

The purpose of this paper is to provide reference results for $O(nd)$ low-complexity algorithms on benchmark classification problems. In the next section a few popular algorithms of this sort are shortly described. In section 3 classifiers are tested on a number of benchmark calculations. Brief discussion of the usefulness of such results concludes this paper.

II. Classification algorithms

Most of the $O(nd)$ classifiers are well known since the early dates of pattern recognition and may be found in classic textbooks [3].

A. Majority Classifier

The Majority Classifier (MC) simply assigns all vectors to the most frequent class in the learning set. If there is no majority one of the classes is chosen arbitrarily. Result reported by the MC is an estimation of *a priori* probability and is most frequently used to establish a baseline for comparing results of classifiers. In the k-fold stratified crossvalidation MC results show some variance if the number of samples in each class is not exactly divisible by k.

B. Nearest prototype classifier (1NP)

The nearest prototype classifier (NP) is based here on a single prototype vector R^k for each class $k = 1 \dots K$, calculated as the mean for the class. If Mahalanobis distance is used such 1NP classifier may be derived as an approximation to Bayesian classifier if identical covariance matrices and identical a priori probabilities are assumed [3]. The cost of covariance matrix calculation is $O(nd^2)$ and the cost of its inversion is $O(d^3)$, therefore to stay within $O(nd)$ category Euclidean distance is used after data standardization.

C. Learning Vector Quantization (LVQ)

LVQ improves upon 1NP by optimizing the position of the prototypes [4]. Each class is represented by a single codebook vectors here. For K classes the $O(nd)$ complexity is increased KN_{it} times, but this is relatively small number. Following the nearest neighbor rule, the feature space is thus divided into regions corresponding to the classes. In LVQ the codebook vectors do not try to approximate the true class density functions, but are placed in order to describe the class boundaries directly. They form a piecewise linear tessellation of the feature space.

In the basic LVQ algorithm an initial set of codebook vectors m_i is first chosen from the training set. This set is iteratively adapted in the following manner : the learning vectors x are taken one by one. If the codebook vector m_c closest to x belongs to the same class as x, m_c is moved a little bit towards x, if not, m_c is moved away somewhat from x. This process is iterated until convergence.

D. Maximum Likelihood Classifier (MLC)

Another drastic simplification is based on on the assumption of Gaussian distributions and calculation of maximum likelihood. First dispersion σ_i^k for each dimension $i = 1 \dots d$ and class $k = 1 \dots K$ is calculated, and then the class probability is estimated as:

$$P(\vec{x}|C_k) = \frac{g(\vec{x}|C_k)}{\sum_{i=k}^{K} g(\vec{x}|C_k)} \tag{1}$$

where:

$$g(\vec{x}|C_k) = \prod_{i=1}^{d} G\left(\frac{(r_i^k - x_i)^2}{2(\sigma_i^k)^2}\right) \tag{2}$$

$G(\cdot)$ is a Gaussian function, C_k is the class for which classification probability of vector $\vec{x}$ is estimated, and r_i^k is $i-$th coordinate of the R^k center of $k-$th class.

E. Naive Bayes (NB)

Naive Bayesian classifier assumes that all features are independent and estimate posterior probability according to Bayes formula $P(C_k|\vec{x}) = P(\vec{x}|C_k)P(C_k)/P(\vec{x})$. Despite this unrealistic assumption the resulting classifier is remarkably successful in practice. The likelihood is calculated in the same way as for the MLC case.

Some data contains nominal features. To use them in training of NB classifier they have been converted to binary vectors using temperature coding, i.e. using as many bits as there are unique symbolic values, and setting a single 1 bit for each unique value.

F. MLP with fixed resources (K2MLP)

Training Multi-Layer Perceptron networks with fixed resources also falls into the $O(nd)$ complexity class. The number of the hidden nodes using sigmoidal functions is equal here to $K(K-1)/2$, the number of pairs of classes. The final decision is done using the winner-takes all (WTA) mechanism. The number of epochs is fixed at 30 and the learning rate is equal to 0.1. This classifier is added to show that even very simple MLP networks, far from optimal, may achieve high quality results on some data.

III. Experimental results

A summary of all datasets used is presented in Table I (all downloaded from UCI Machine Learning Repository). Vectors with missing feature values have been removed and nominal features have been replaced with n binary features. These datasets have been used very frequently for evaluation of new algorithms.

To provide unbiased comparison of low complexity $O(nd)$ methods with state-of-the art results, for each dataset and each method described above, calculations have been repeated 10 times averaging the 10-fold crossvalidation results (this requires 100 calculations). Optimized Support Vector Machine results have been provided for linear kernel (with optimization of the C hyperparameter carried within additional crossvalidation cycle at each partition) and with Gaussian kernel (with optimization of C and σ hyperparameters) [5], [6]. These SVML and SVMG results are close to the best state-of-the-art for sophisticated methods, and because we could not find much better results for these datasets in the literature they are given as the reference.

All calculations have been performed using the Intemi package developed in our group [7]. The goal of Intemi is to automatize selection of learning algorithms and compose various transformations to create new algorithms optimally biased for a given data [2]. The search for such optimal algorithms is guided by complexity (including performance time and memory use), and thus $O(nd)$ methods are an important reference point before more complex approaches are attempted.

Results of experiments are collected in Table II, with accuracies and standard deviations for each dataset given. Wilcoxon statistical tests [8] have been used to check significance of the results. In the last row the number of total wins/ties/losses is presented for each method. This means that if a method A is found significantly better the methods B for a given dataset the "wins" counter is updated by 1, and if the difference in statistical accuracy is not significant than the "ties" counter is updated. For example, for the ZOO dataset all 5 low-complexity algorithms (and 2 SVM methods) are statistically better than the Majority Class, therefore they score 1 win, and the MC scores 5 losses. In bold best result for each dataset have been marked.

To distinguish dataset that should be regarded as trivial from more difficult cases $O(nd)$ methods are compared with the SVM results. If for some dataset there are result obtained with low complexity methods that are not significantly worse than those obtained by SVML and SVMG the dataset is considered easy to analyze, and the column "Trivial" in the Table II has $+$, otherwise it has $-$.

TABLE I
SUMMARY OF DATASETS.

Data	#Vectors	#Features	#Classes
arrhythmia	63	279	11
autos	159	25	6
balance-scale	625	4	3
blood-transfusion-service-center	748	4	2
breast-cancer-wisconsin-diagnostic	569	30	2
breast-cancer-wisconsin-original	683	9	2
breast-cancer-wisconsin-prognostic	194	33	2
breast-tissue	106	9	6
car-evaluation	1728	6	4
cardiotocography-1	2126	21	10
cardiotocography-2	2126	21	3
chess-king-rook-vs-king-pawn	3196	36	2
cmc	1473	9	3
congressional-voting-records	232	16	2
connectionist-bench-sonar	208	60	2
connectionist-bench-vowel	528	10	11
cylinder-bands	277	39	2
dermatology	358	34	6
ecoli	336	7	8
glass	214	9	6
habermans-survival	306	3	2
hepatitis	80	19	2
ionosphere	351	34	2
iris	150	4	3
libras-movement	360	90	15
liver-disorders	345	6	2
lymph	148	18	4
monks-problems-1	556	6	2
monks-problems-2	601	6	2
monks-problems-3	554	6	2
parkinsons	195	22	2
pima-indians-diabetes	768	8	2
sonar	208	60	2
spambase	4601	57	2
spect-heart	267	22	2
spectf-heart	267	44	2
statlog-australian-credit	690	14	2
statlog-german-credit-numeric	1000	24	2
statlog-heart	270	13	2
statlog-vehicle-silhouettes	846	18	4
teaching-assistant-evaluation	151	5	3
thyroid-disease	7200	21	3
vote	232	16	2
wine	178	13	3
zoo	101	17	7

Only 13 out of 45 analyzed datasets should be considered non-trivial. For these datasets SVM with Gaussian kernel gives significantly better result than all other methods used in this comparison. Moreover, MLP results are better than other low-complexity approaches indicating that further learning and more complex models may have advantage here. In general K2MLP and Naive Bayes show the lowest number of losses and highest number of wins, as should be expected. Naive Bayes is in almost all cases significantly better or equivalent to the maximum likelihood classifier, as theory predicts, showing the importance of calculation of a posteriori probabilities. However, the nearest neighbor prototype (1NP) has frequently been winning and for some datasets (dermatology, lymph, statlog-australian-credit, zoo) gave better results than all other methods. This shows the usefulness of simplest prototype-based rules that frequently have natural interpretation as the similarity estimation; there are of course many ways to optimize such rules [9], [10]. Majority Classifier gives the worst results suitable only as the baserate.

IV. Conclusions

The main purpose of this paper was to show that the simplest low complexity $O(nd)$ methods provide results on many benchmark datasets that are not significantly worse than the best results that may be achieved with more sophisticated methods. 32 out of 45 benchmark datasets from UCI are in this sense trivial and **should not be used as the only basis for evaluation** of the new algorithms. Identification of trivial datasets is important to improve methodology of comparison of new methods in computational intelligence and machine learning. We suggest that the results reported here are quite relevant as a baseline for testing new methods, a bit more difficult than the majority classifier baserate.

The second aspect of this work is that the performance of low-complexity methods may be worth studying in more detail, as there are other computationally inexpensive approaches that can be applied to discover trivial data. We should provide in the near future more extensive comparison of such methods on even larger collection of data.

There are many other classic, simple algorithms such as 1R, although their computational complexity is higher than O(nd) and therefore were not considered in this article. It is worth mentioning also the influence of representation on the problem complexity. For example, feature selection or choice of a small number $m < n$ of prototypes could simplify and speed up the learning process, but the complexity of such methodology is increased by the complexity of such techniques. In this paper we try to not combine different learning mechanisms, and only focus on the simple and fast methods of classification. In this form it should be easy to compare the presented results with other algorithms.

References

[1] A. Asuncion and D. Newman, "UCI machine learning repository," http://www.ics.uci.edu/∼mlearn/MLRepository.html, 2007. [Online]. Available: http://www.ics.uci.edu/\hbox{$\sim$}mlearn/MLRepository.html

[2] N. Jankowski, W. Duch, and K. Grąbczewski, Eds., *Meta-learning in Computational Intelligence.*, ser. Studies in Computational Intelligence. Springer, 2011, vol. 358.

[3] R. O. Duda, P. E. Hart, and D. Stork, *Patter Classification*. New York: J. Wiley & Sons, 2001.

[4] T. Kohonen, *Self-Organization and Associative Memory*. Heidelberg Berlin: Springer-Verlag, 1984.

[5] N. Cristianini and J. Shawe-Taylor, *An Introduction to Support Vector Machines and other Kernel-Based Learning Methods*. Cambridge University Press, 2000.

[6] B. Schölkopf and A. Smola, *Learning with Kernels. Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA: MIT Press, 2001.

[7] N. Jankowski and K. Grąbczewski, *Meta-learning in Computational Intelligence.*, ser. Studies in Computational Intelligence. Springer, 2011, vol. 358, ch. Universal Meta-learning Architecture and Algorithms, pp. 1–76.

[8] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics*, vol. 1, pp. 80–83, 1945.

[9] W. Duch and M. Blachnik, "LVQ algorithm with instance weighting for generation of prototype-based rules." *Neural Networks*, vol. 24, p. 824–830, 2011.

[10] W. Duch, "Towards comprehensive foundations of computational intelligence," in *Challenges for Computational Intelligence*, W. Duch and J. Mandziuk, Eds. Springer, 2007, vol. 63, pp. 261–316.

TABLE II
COMPARISON OF 5 LOW-COMPLEXITY CLASSIFIERS WITH LINEAR AND GAUSSIAN KERNEL SVM, 10X10CV RESULTS.

Data	Trivial	MajorityClass	1NP	MLC	LVQ	NaiveBayes	K2MLP	SVML	SVMG
arrhythmia	+	29.43±7.82	44.00±27.42	29.79±14.76	**63.69±17.46**	10.19±9.80	58.00±16.63	50.92±17.31	43.36±21.47
autos	+	28.18±3.14	56.83±10.39	62.23±11.97	30.23±10.18	63.32±10.28	**70.44±10.31**	54.48±13.75	74.29±12.58
balance-scale	+	45.37±0.55	70.07±6.19	53.80±5.56	89.88±9.10	90.80±1.40	**91.15±4.03**	84.47±3.17	89.83±2.09
blood-transfusion-service-center	+	**76.20±0.57**	69.13±3.74	60.28±3.08	76.20±0.57	75.01±3.02	70.59±8.35	76.20±0.48	79.14±4.57
breast-cancer-wisconsin-diagnostic	+	62.74±0.73	93.08±3.47	91.78±3.44	93.04±3.47	93.18±3.47	**97.00±2.33**	97.36±2.51	97.54±2.50
breast-cancer-wisconsin-original	+	65.01±0.83	96.44±2.22	94.46±2.62	95.83±2.22	96.24±2.30	**96.72±2.17**	96.48±2.49	96.77±2.48
breast-cancer-wisconsin-prognostic	+	**76.34±3.02**	62.64±10.02	74.17±8.28	76.34±3.02	66.39±8.83	73.00±9.92	77.84±8.03	76.86±8.20
breast-tissue	+	17.68±3.64	61.73±11.70	**68.34±11.45**	15.85±6.55	64.03±12.78	63.04±11.17	53.00±10.87	63.27±7.59
car-evaluation	−	70.02±0.16	73.22±2.90	84.08±2.50	73.59±3.65	3.76±0.33	**91.13±2.54**	69.57±2.03	98.84±0.77
cardiotocography-1	+	27.23±0.16	54.80±3.02	70.98±2.46	27.23±0.16	72.57±2.28	**77.88±2.83**	57.90±4.12	80.43±2.79
cardiotocography-2	−	77.85±0.31	76.57±1.78	73.72±2.21	77.85±0.31	82.54±1.86	**87.32±2.69**	87.53±1.48	92.09±2.01
chess-king-rook-vs-king-pawn	−	52.22±0.12	86.25±1.33	83.86±1.68	61.40±15.17	67.27±1.81	**90.90±3.29**	96.21±1.38	99.28±0.36
cmc	+	42.70±0.23	46.03±3.52	47.89±3.62	22.61±0.34	**49.64±3.96**	48.82±3.40	19.14±2.14	34.09±3.67
congressional-voting-records	+	53.46±2.26	89.86±5.40	94.73±4.37	89.69±5.27	94.25±5.11	**94.95±3.94**	94.48±3.62	92.65±4.11
connectionist-bench-sonar	−	53.35±2.26	69.65±7.49	70.62±5.99	71.67±7.44	69.03±8.68	**76.73±8.05**	75.47±8.26	85.52±5.28
connectionist-bench-vowel	−	7.58±0.06	51.98±6.61	52.00±5.99	9.09±1.36	67.53±6.28	**80.95±5.02**	25.76±5.01	96.77±2.20
cylinder-bands	+	64.25±1.17	68.97±8.43	38.68±7.02	64.47±4.26	**74.07±7.51**	71.35±8.32	74.58±5.23	76.89±7.57
dermatology	+	31.01±0.97	**96.87±3.15**	88.40±4.55	91.30±3.79	90.13±4.52	94.88±3.84	94.01±3.54	94.49±3.88
ecoli	+	42.57±1.58	81.38±5.76	77.13±12.09	78.50±9.39	70.76±20.46	**83.55±6.04**	78.48±5.90	84.17±5.82
glass	+	35.54±2.56	48.82±9.88	48.67±6.25	34.79±4.52	43.34±8.44	**59.78±8.84**	42.61±10.05	62.43±8.70
habermans-survival	+	73.54±1.86	74.45±7.19	71.21±7.65	73.19±4.07	**74.83±5.58**	64.34±14.53	73.52±1.86	71.55±8.42
hepatitis	+	83.75±5.76	82.50±13.06	90.38±10.18	83.75±5.76	**91.25±9.15**	84.00±12.32	83.25±11.54	84.87±11.98
ionosphere	−	64.10±1.43	81.14±6.41	59.23±6.24	83.72±5.34	84.24±6.15	**86.46±5.48**	87.72±4.63	94.61±3.68
iris	+	33.33±0.00	85.80±8.67	94.60±5.42	85.67±8.50	95.40±5.42	**95.60±4.76**	72.20±7.59	94.86±5.75
libras-movement	−	4.64±1.31	56.86±6.22	51.92±7.07	6.67±1.48	**65.50±6.57**	52.67±8.17	49.16±5.24	84.44±6.02
liver-disorders	+	57.96±1.62	57.60±8.18	**65.12±7.97**	57.85±3.43	56.28±7.93	62.61±8.29	68.46±7.36	70.30±7.90
lymph	+	54.71±4.52	**86.43±8.61**	78.79±9.36	82.49±9.35	81.18±8.95	82.67±9.24	81.26±9.79	83.61±9.82
monks-problems-1	−	49.46±0.60	74.64±4.18	74.64±4.18	70.58±10.05	52.28±2.05	**83.07±3.03**	65.81±6.50	99.82±0.56
monks-problems-2	−	65.72±0.84	54.90±5.85	53.85±6.19	65.54±1.28	54.50±4.21	**74.76±5.14**	65.72±0.82	84.86±4.91
monks-problems-3	+	51.99±0.86	96.39±2.17	96.39±2.17	96.39±2.17	94.78±5.79	**98.57±1.69**	80.13±4.91	96.75±2.22
parkinsons	−	75.44±3.19	73.55±8.71	78.22±8.46	77.76±6.89	69.83±9.09	**85.64±7.57**	86.26±10.17	93.26±5.61
pima-indians-diabetes	+	65.10±0.54	72.72±4.84	68.63±4.66	75.02±4.50	**75.30±4.39**	73.82±5.03	77.08±4.27	76.04±3.69
sonar	−	53.35±2.26	69.65±7.49	70.62±5.99	71.67±7.44	69.03±8.68	**76.73±8.05**	73.71±9.62	86.42±7.65
spambase	+	60.60±0.10	89.50±1.31	87.41±1.44	82.79±11.23	81.78±1.52	**91.58±1.57**	92.96±1.30	93.69±1.04
spect-heart	+	79.42±2.05	72.15±8.20	**83.64±6.80**	79.42±2.05	72.19±6.99	77.37±7.75	82.72±7.43	83.50±6.62
spectf-heart	+	**79.42±2.05**	66.48±8.43	79.31±2.06	79.42±2.05	67.68±8.35	72.89±11.01	78.61±9.73	80.18±7.34
statlog-australian-credit	+	55.51±0.67	**84.45±4.35**	79.55±4.81	83.13±7.53	79.54±4.40	82.43±4.69	85.50±3.86	84.49±3.48
statlog-german-credit-numeric	+	70.00±0.00	72.81±4.26	67.55±4.73	72.09±3.38	72.84±4.33	**72.93±4.61**	77.50±2.32	76.01±3.16
statlog-heart	+	55.56±0.00	84.63±6.08	82.41±7.41	**85.07±6.24**	84.22±7.13	82.44±7.00	82.96±7.65	81.48±4.61
statlog-vehicle-silhouettes	−	25.12±0.54	45.33±4.57	52.96±4.20	25.77±0.86	45.69±3.98	**72.85±4.65**	69.86±2.74	79.78±2.66
teaching-assistant-evaluation	+	34.45±2.74	50.85±12.71	48.58±12.82	33.05±3.60	24.32±8.88	**52.09±12.00**	13.25±9.94	42.37±9.44
thyroid-disease	−	92.58±0.09	71.07±1.84	86.56±1.44	92.58±0.09	36.55±3.36	**94.74±2.17**	93.76±0.47	97.47±0.66
vote	+	53.46±2.26	89.86±5.40	94.73±4.37	89.69±5.27	91.93±4.98	**95.29±4.04**	96.12±3.85	96.89±3.11
wine	+	39.91±1.85	97.25±3.94	96.84±3.72	97.25±3.94	**97.30±3.80**	96.18±3.91	97.71±2.95	97.15±4.08
zoo	+	40.63±3.19	**91.55±6.91**	86.34±8.54	83.45±7.11	86.82±8.58	83.25±8.42	91.61±6.67	93.27±7.53
wins ǀ ties ǀ losses		**31 ǀ 64 ǀ 130**	**75 ǀ 73 ǀ 77**	**73 ǀ 64 ǀ 88**	**80 ǀ 59 ǀ 86**	**81 ǀ 63 ǀ 81**	**137 ǀ 61 ǀ 27**		