Journal of Information Processing Vol.23 No.4 497–504 (July 2015)

[DOI: 10.2197/ipsjjip.23.497]

Recommended Paper

Improving the Performance of the Decision Boundary Making Algorithm via Outlier Detection

Yuya Kaneda[1,a)] Yan Pei[1,b)] Qiangfu Zhao[1,c)] Yong Liu[1,d)]

Received: December 4, 2014, Accepted: March 4, 2015

Abstract: Outlier detection is one of the methods for improving the performance of machine learning models. Since outliers often affect the performance of the learning models negatively, it is desired to detect and remove outliers before model construction. In this paper, we try to improve the performance of the decision boundary making (DBM) algorithm via outlier detection. DBM has been proposed by us for inducing compact and high performance learning models that are suitable for implementation in portable computing devices. The basic idea of DBM is to generate data that can fit the decision boundary (DB) of a high performance model, and then induce a compact model based on the generated data. In our study, a support vector machine (SVM) is used as the high performance model, and a single hidden layer multilayer perceptron (MLP) is used as the compact model. Experimental results obtained so far show that DBM performs well in many cases, but its performance still is not good enough for some applications. In this paper, we use SVM not only for obtaining the DB, but also for detecting the outliers, so that better MLP can be induced using cleaner data. We use a threshold $\delta_{outlier}$ to control the number of outliers to remove. Experimental results show that, if we select $\delta_{outlier}$ properly, the DBM incorporated with outlier detection outperforms the original DBM, and it is better than or comparable to SVM for all databases used in the experiments.

Keywords: decision boundary making, support vector machine, neural network, outlier detection

1. Introduction

For a given pattern recognition problem, we often assume that data in different classes are separable provided that a proper set of attributes or features is given. In practice, however, because of measurement errors, some data may appear in the territories of different classes. These data are called outliers, and should be excluded from the training set. Machine learning models obtained based on training sets containing a lot of outliers may take many resources (e.g., hidden neurons in a multilayer neural network) to fit the outliers, but may fail to fit unknown patterns well. In other words, outliers can degrade the generalization ability of the resulted model.

Since outliers are often close to the decision boundaries (DBs), we may detect and delete the outliers based on the DBs before constructing the learning model. However, if we do not have a model, we do not know where the DBs are. Thus, outlier detection and learning have a chicken-and-egg relation.

To detect the outliers efficiently, several methods have been proposed in past publications. In Ref. [1], Amidan et al. proposed a Chebyshev theorem based method. In this method, the authors used the Chebyshev theorem to define and detect the outliers, and they confirmed that the performance could be improved by removing the outliers. Ferdowsi, H. et al. proposed a machine learning model based method for outlier detection. Specifically, they used a neural network (NN) [12] to identify the outliers [4]. Geebelen, D. et al. used a support vector machine (SVM) [15] as the learning model, and showed that the performance of the SVM can be improved by removing the outliers [5].

In our research, we have proposed the decision boundary making (DBM) algorithm for inducing compact and high performance learning models that are suitable for implementation in portable computing devices (PCDs) [8]. The DBM algorithm is an improved version of the decision boundary learning (DBL) algorithm [16], which was proposed by us earlier. The main purpose of our research is to provide a tool that can help the users to build various aware agents in the PCDs. The aware agents, which are machine learning models, are expected to improve the usability of the PCDs by providing useful information proactively to the users. Compared with desk-top computing machines, the computing resources of a PCD are usually limited. Thus, to embed various high performance aware agents in the same device, it is necessary to reduce the implementation cost for each agent. The DBM and DBL are algorithms for this purpose.

Several methods have been proposed in the publications to reduce the cost of learning models. Since the number of support vectors (SV) of an SVM is usually proportional to the number of training data [14], some methods have been proposed based on data reduction [9], [10]. In these methods, however, some im-

[1] School of Computer Science and Engineering, University of Aizu, Aizu-Wakamatsu, Fukushima 965–8580, Japan
[a)] d8161108@u-aizu.ac.jp
[b)] peiyan@u-aizu.ac.jp
[c)] qf-zhao@u-aizu.ac.jp
[d)] yliu@u-aizu.ac.jp

A preliminary version of this paper was presented at the IPSJ Tohoku-Branch workshop held in January 2014. This paper was recommended to be submitted to Journal of Information Processing (JIP) by the Chairman of IPSJ Tohoku-Branch.

portant SVs may be lost in the data reduction step. Therefore, theoretically, the performance of the obtained model is upper-bounded by the SVM induced from the original training data. Wang, H.W. et al. improved the performance of the algorithm by using relative distance only for un-balanced data sets [17]. However, the performance of this method may not be good for balanced training data. Another way to reduce the implementation cost is to use a new kernel function of the SVM [3]. The advantage of this method is that the number of SVs can be reduced without degrading the performance significantly. But for the classification time, the computational cost with the proposed kernel is equivalent to other kernels. Therefore, it is difficult to be used in PCDs.

In DBM, we focused on the DB of a high performance learning model. If a low cost model (i.e., a model that requires less computing resources) can reconstruct the DB of the high performance model, the low cost model can have a high performance. In a probability sense, SVM is known as the optimal machine learning model. Thus, we choose SVM as the high performance model, and use a single hidden layer multilayer perceptron (MLP) as the low cost model.

In using DBM, we first generate data to fit the DB of an SVM, and then induce the MLP based on the new training data. Experimental results obtained so far show that DBM can often yield low cost MLPs that are comparable to or even better than SVMs. For some applications, however, the performance is not good enough.

In this paper, we incorporate outlier detection with the DBM, expecting to improve the DBM performance further without increasing the implementation cost of the resulted learning model. Since an SVM is designed to generate a reference DB for DBM, we can use the same SVM to detect outliers. A direct method is to define all data that cannot be classified correctly by the SVM as outliers and remove them before inducing the low cost MLP. However, this method may remove some important data by mistake, and thus degrade the performance significantly. To solve this problem, we use a threshold $\delta_{outlier}$ to control the number of outliers to remove. This threshold actually defines a new margin along the DB of the SVM. If $\delta_{outlier}$ is properly selected, we can remove the outliers while keeping important data for fitting the DB. Experimental results show that, if we select $\delta_{outlier}$ properly, the DBM incorporated with outlier detection outperforms the original DBM, and it is better than or comparable to SVM for all databases used in the experiments.

The structure of this paper is as follows. Section 2 introduces the DBM algorithm in detail. Section 3 explains the proposed DBM with SVM-based outlier detection. Section 4 and Section 5 provide experimental results on several public databases with statistical tests. Finally Section 6 shows the conclusions and some topics for future work.

2. Review of the Decision Boundary Making Algorithm

The DBM algorithm was proposed to induce compact and high performance machine learning models. The main idea of DBM is to reconstruct the DB of a high performance model using a small model. We use SVM for the high performance model, and MLP as the small model. To reconstruct the DB, DBM generates new training data to approximate the DB of the SVM. Then, it obtains an MLP using the new training data. The DBM algorithm focuses on support vectors (SV) of the SVM when it generates new data. There are SVs near the DB of the SVM. Therefore, if we generate new data around the SVs, we can add new data near the DB. In the following sub-sections, we discuss and show details of the DBM algorithm.

2.1 Why Can DBM Produce Models that are Compact and High Performance?

Although SVM is known as a high performance model in the machine learning field, the model size of an SVM is often large because the model size is decided by the number of data in a training set which is often large in real applications. In general, the computational cost is related to the model sizes. For SVM, it is difficult to control the model size and the computational cost.

In comparison, the model size of an MLP is controllable. The model size of an MLP can be pre-defined. If we set the parameters to obtain a small model, the obtained MLP becomes small. Therefore, it is possible to obtain a compact and high performance model by reconstructing a high performance SVM with an MLP.

2.2 Generating New Training Data

To generate new training data, the DBM algorithm focuses on the SVs of the SVM. There are many SVs near the DB of the SVM, which are more important data than others. Therefore, if we generate new data and put them around these SVs, we may get new training data that can approximate the DB of the SVM.

We use ϵ-neighborhood to generate new data around each SV. The detail of ϵ-neighborhood based data generation is shown in **Fig. 1**. All generated data are in the ϵ-neighborhood. The ϵ is a given parameter. For each SV, the DBM algorithm generates N new data. Therefore, totally $N \times N_{SV}$ data are generated, where N_{SV} is the number of SVs.

We set conditions for newly generated data to decide whether the data should be added into the new training set or not. If the

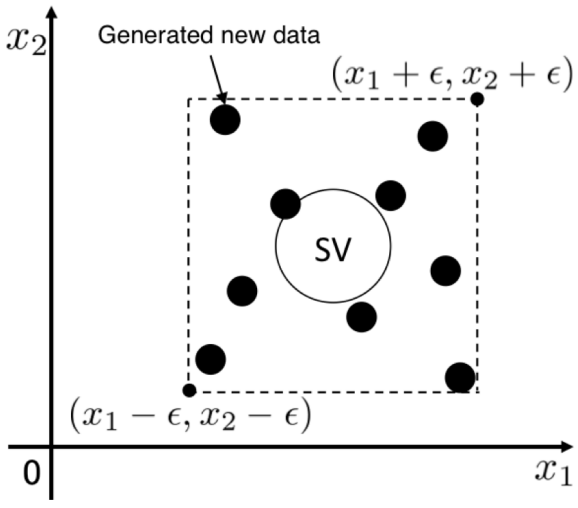

Fig. 1 New data is generated in ϵ-neighborhood area of the SV.

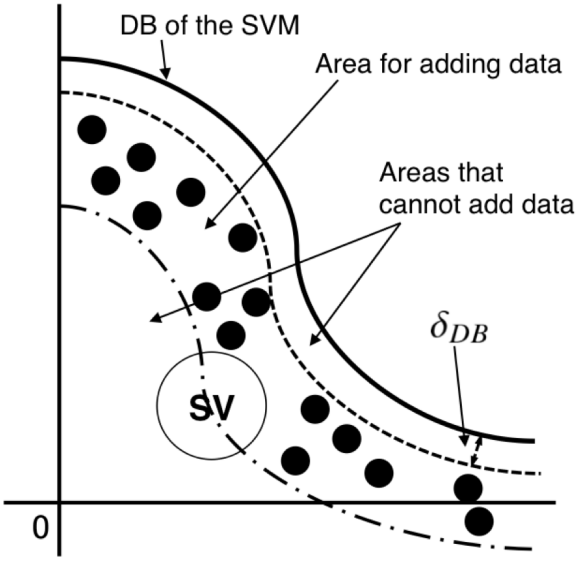

Fig. 2 Two conditions of the DBM algorithm are shown in this figure. DBM does not add new data in $[-\delta_{DB}, \delta_{DB}]$ and farther area than that of SVM.

generated data are put around SVs randomly, some data may be far away from the DB, and they are not important for approximating the DB. To reduce these data, we set a condition given in Eq. (1), where $g(X)$ is the output value of the SVM for the datum X, which is generated in the neighborhood of the SV P. If Eq. (1) is true, the generated datum X is not put into the new training set. Another condition is to limit data not to be too close to the DB of the SVM. If there are too many data around the DB in the new training set, the DB formed by the training data might be too complex for a small MLP to learn. The DB should be smooth to obtain small MLPs. To create a smooth DB, we set a margin between the DB and the generated data in Eq. (2), where δ_{DB} is the given parameter which is in [0, 1). If Eq. (2) is true, the generated data X is not added. An example of two conditions is shown in **Fig. 2**.

$$|g(X)| > |g(P)| \tag{1}$$

$$|g(X)| < \delta_{DB} \tag{2}$$

Finally, a new training set is obtained by putting the given training data, the SVs of the SVM, and the newly generated data together. It is a fact that, the SVs of SVM are a subset of the given training data. These data will appear twice in the new training set. In previous experiments, it has been found that the new training set led to the better performance when the SVs were included. It is reasonable to assume that the SVs made contributions to such enforced performance. Therefore, SVs are repeated in the new training set.

2.3 The DBM Algorithm

There are 2 phases in the DBM algorithm. The first phase is the learning phase, while the second is the classification phase. The two phases are shown in **Fig. 3**. In the classification phase after the learning phase, the DBM algorithm needs only the MLP obtained in the learning phase. The MLP can be obtained on servers by the DBM algorithm in the learning phase, and it can be used on mobile devices in the classification phase.

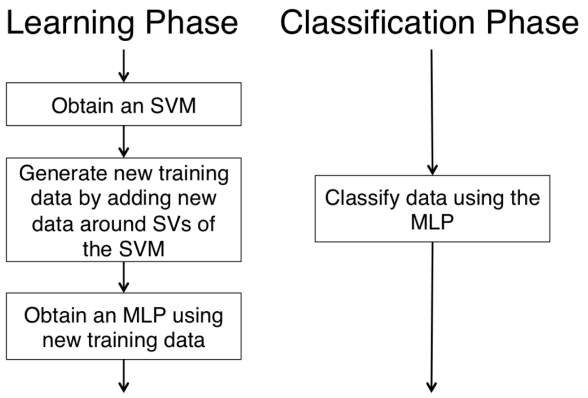

Fig. 3 Brief flows of the DBM algorithm. There are 3 steps in the learning phase, and 1 step for classification step.

The steps of the learning phase are described in Algorithm 1.

Algorithm 1 Learning phase of the DBM algorithm

Ensure: Obtain an MLP using given training data Ω.

```
Obtain an SVM based on Ω
U_SV = {SV_1, SV_2, ..., SV_{N_SV}} (SV_i is an SV of the SVM)
New training set Ω_new = U_SV + Ω
for all P in U_SV do
    for i = 1 to N do
        Create a vector V that each element is random value in [ϵ, −ϵ]
        X_new = P + V
        if |g(X_new)| < δ_DB or |g(X_new)| > |g(P)| then
            continue
        end if
        Set label of X_new by sgn(g(X_new))
        Ω_new = Ω_new + X_new
    end for
end for
Obtain an MLP using Ω_new
```

3. Decision Boundary Making Using SVM-Based Outlier Detection

In this paper, we try to improve the performance of the DBM algorithm by using an SVM-based outlier detection method. Some previous experimental results have shown that in some cases the performance of the DBM is lower than SVM. One reason is that there might be outliers in the new training set. To generate better models, the outliers should be deleted.

The outlier is a noisy datum having a negative impact on the performance of the model. It means that the obtained models based on training data with outliers could have poor performance. By removing the outliers from the training data, the models could have higher performance. The main problem is how to define the outlier properly. If the definition is not correct, some important data might be detected as outliers, and the performance of the model will be degraded.

The SVM-based outlier detection method is more efficient than other methods for the DBM algorithm. In the learning phase of the DBM algorithm, we obtain an SVM first, and then generate new training data to approximate the DB of an SVM. We can use the SVM directly for detecting the outliers. Therefore, using SVM-based outlier detection in DBM will not increase the

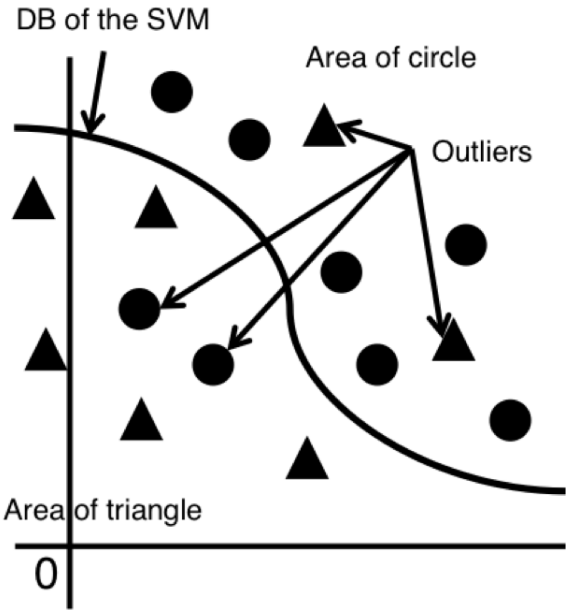

Fig. 4 An example of SVM-based outlier detection. An outlier lies on the wrong side of its own area.

computational cost at all.

The SVM-based outlier detection defines a datum as an outlier if the given label is different from the labels classified by the SVM (See **Fig. 4**). In this definition we assume that the DB of the SVM is a perfect boundary for classification. After detecting outliers, we usually delete these outliers from the training set, and then obtain a new model using the training set without outliers.

3.1 The Improved SVM-Based Outlier Detection

We set a parameter $\delta_{outlier}$ to improve the SVM-based outlier detection method. If outliers are identified by using the DB of SVMs only, sometimes important data may be falsely classified as outliers and be removed. Since the DB of the SVM is not perfect, data close to the DB can be wrongly classified with a certain probability. To prevent from removing important data, we define the outlier as Eq. (3), where $d(X) \in \{-1, +1\}$ is the label of the data X, and $\delta_{outlier}$ is a real number specified by the user. If Eq. (3) is true, datum X is identified as the outlier. **Figure 5** shows an example of the improved SVM-based outlier detection.

$$g(X) * d(X) < -\delta_{outlier} \tag{3}$$

The SVM-based outlier detection is implemented in the DBM algorithm to identify outliers from given training data. In the new training set of the DBM algorithm, there are the given training data that are not outliers, the SVs of the SVM, and the newly generated data.

3.2 The Algorithm of DBM Using SVM-Based Outlier Detection

The new DBM algorithm using SVM-based outlier detection is described in Algorithm 2.

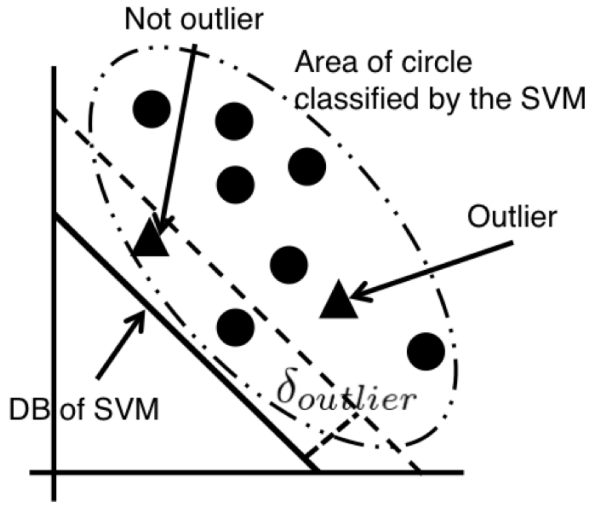

Fig. 5 $\delta_{outlier}$ defines the range of noisy data (outliers) that influence performance of machine learning models. Vectors within the range of $\delta_{outlier}$ area are detected as outlier data.

Algorithm 2 Learning phase of the DBM algorithm

Ensure: Obtain an MLP using given training data Ω.
1: Obtain an SVM based on Ω
2: Detect outliers from Ω by using Eq. (3) as
$U_{outlier} = \{X_{outlier1}, \ldots, X_{outlierN_{outlier}}\}$ ($N_{outlier}$ is the number of outliers).
3: $U_{SV} = \{SV_1, SV_2, \ldots, SV_{N_{SV}}\}$ (SV_i is an SV of the SVM)
4: New training set $\Omega_{new} = U_{SV} + \Omega - U_{outlier}$
5: **for all** P in U_{SV} **do**
6: **for** $i = 1$ to N **do**
7: Create a vector V that each element is random value in $[\epsilon, -\epsilon]$
8: $X_{new} = P + V$
9: **if** $|g(X_{new})| < \delta_{DB}$ or $|g(X_{new})| > |g(P)|$ **then**
10: continue
11: **end if**
12: Set label of X_{new} by $sgn(g(X_{new}))$
13: $\Omega_{new} = \Omega_{new} + X_{new}$
14: **end for**
15: **end for**
16: Obtain an MLP using Ω_{new}

Table 1 Features of public databases from Ref. [2].

	Number of Classes (N_c)	Number of Features (N_f)	Number of Data (N_d)
Australian	2	14	690
Breast cancer (Breast)	2	10	683
Diabetes	2	8	768
German	2	24	1,000
Indian L Patient (ILP)	2	10	582
Ionosphere	2	34	351

4. Experiments on Comparing Performance of SVM, MLP, and DBMs with and without Outliers

In this section, we compare the performance of SVM, MLP, and DBMs with and without Outliers by experiments with statistical tests using some public databases taken from the machine learning repository of the University of California at Irvine [2]. **Table 1** shows the parameters of each database.

For each database, 360 times of 5-fold cross validation was

Table 2 Machine specs and environments.

Machine	Apple iMac 21.5-inch, Late 2013
OS	Mac OS X 10.9.
CPU	Intel Core i5 2.7 GHz
Memory	8 GB
Program Language	C++
Compiler	Apple LLVM version 5.0

conducted. The number of training data is $N_t = [N_d \times \frac{4}{5}]$, and the number of testing data is $N_d - N_t$ for each database. For the result, we calculate the recognition rate (RR) for testing data, and it is averaged over 360×5 runs. The RR is calculated as Eq. (4). We normalize the given training data by using rescaling method. The computer system configuration and environment used in the experiment are shown in **Table 2**.

$$RR = \frac{\text{Number of correct data in test data}}{\text{Number of test data}} \tag{4}$$

4.1 Experimental Design

In this paper, we proposed the DBM algorithm incorporated with the SVM-based outlier detection method. We confirm whether the performance of the proposed algorithm can be improved by removing outliers from the new training set. To compare the performance, we define four methods' abbreviations as follows:

- SVM: SVM with RBF kernel ($\kappa(x, z) = \exp(-\|x - z\|^2)$).
- MLP: MLP obtained using the original training data.
- DBM0: MLP obtained using the DBM with outliers (original DBM algorithm).
- DBM1: MLP obtained using the DBM without outliers (detected by the SVM-based method).

For the given training data, we normalized the training data by using rescaling normalization. The rescaling normalization converts the range $[\min(col_i), \max(col_i)]$ to $[F_{\min}, F_{\max}]$ for each feature, where col_i is a set of i-th features in given training set, $\min(X)$ and $\max(X)$ return the minimum and maximum values from set X, and $F_{\min}$ and $F_{\max}$ are given parameters ($F_{\min} < F_{\max}$). In our experiment, the rescaled range was fixed to $[-1, +1]$.

For SVMs, we used free software SVM^{Light} given in Ref. [7]. This paper used the radial basis function (RBF) kernel as the kernel function. For other parameters, we set the cost-factor to 1; leave-one-out estimates was 0; maximum size of quadratic programming subproblems was 100; number of variables entering the working set in each iteration was 99; error for termination criterion was 0.001; number of iterations was 1,000; value of rho for XiAlpha-extimator and for pruning leave-one-out computation was 1.0; and search depth for extended XiAlpha-extimator was 0.

In MLP learning, we used the back-propagation algorithm [12] to train the model. For the BP algorithm, the learning rate was fixed to 0.5. The maximum number of learning epoch was 1,000. The number of input neurons of the MLPs was N_f. The number of hidden neurons was fixed to 10. The number of output neurons was N_c, which were 2 classes for all databases in this experiment.

For DBM parameters, we set to $\epsilon = 0.1$, $\delta_{DB} = 0.1$, and $N = 10$. For DBM1, we changed the parameter $\delta_{outlier}$ from 0.0 to 0.5 with step size 0.1. For statistical tests, we set 5% for significance level of all results.

Table 3 Best recognition rates with standard deviations and 95% confidence intervals in each method. The bold font values present the best recognition rate in all methods. DBM1 is the best method in 5 out of 6 databases.

Database	Methods	RR (%)	SD (%)	Confidence Interval (%)
Australian	SVM	**84.630**	**2.744**	**[84.503, 84.757]**
	MLP	82.332	3.005	[82.193, 82.471]
	DBM0	84.324	2.721	[84.199, 84.450]
	DBM1 ($\delta_{outlier} = 0$)	84.591	2.808	[84.461, 84.720]
Breast	SVM	96.109	1.537	[96.038, 96.180]
	MLP	95.952	1.606	[95.878, 96.026]
	DBM0	96.249	1.513	[96.179, 96.319]
	DBM1 ($\delta_{outlier} = 0$)	**96.417**	**1.446**	**[96.350, 96.484]**
Diabetes	SVM	76.120	3.182	[75.973, 76.267]
	MLP	72.970	3.362	[72.815, 73.125]
	DBM0	75.850	3.159	[75.704, 75.996]
	DBM1 ($\delta_{outlier} = 0$)	**76.127**	**3.120**	**[75.983, 76.271]**
German	SVM	69.945	2.887	[69.811, 70.078]
	MLP	71.274	2.927	[71.138, 71.409]
	DBM0	72.978	2.945	[72.841, 73.114]
	DBM1 ($\delta_{outlier} = 0.2$)	**73.205**	**2.962**	**[73.068, 73.342]**
ILPD	SVM	71.243	3.677	[71.073, 71.413]
	MLP	68.181	4.390	[67.978, 68.384]
	DBM0	70.024	4.058	[69.837, 70.212]
	DBM1 ($\delta_{outlier} = 0.2$)	**71.261**	**3.614**	**[71.095, 71.428]**
Ionosphere	SVM	89.530	3.567	[89.365, 89.695]
	MLP	90.814	3.523	[90.652, 90.977]
	DBM0	91.534	3.290	[91.382, 91.686]
	DBM1 ($\delta_{outlier} = 0.4$)	**91.885**	**3.166**	**[91.739, 92.031]**

Table 4 The number of training data with the number of outlier data (for DBM1) for each method. The results were obtained using the same $\delta_{outlier}$ values as in Table 3. The N_{ld} means the number of learning data. Numbers of training data of DBMs are usually big.

	SVM, MLP	DBM0	DBM1	
	N_{ld}	N_{ld}	N_{ld}	$N_{outlier}$
Australian	552.0	3541.0	3501.8	43.9
Breast-cancer	547.0	1865.8	1851.2	15.9
Diabetes	615.0	2616.7	2499.0	116.8
German	800.0	6959.6	6765.9	194.3
ILPD	466.0	2315.2	2185.4	133.0
Ionosphere	281.0	2375.7	2374.8	1.6

4.2 Analysis and Discussion on Performance Comparison of SVM, MLP, and DBMs with and without Outliers

Table 3 shows the best RRs, standard deviations (SDs), and confidence intervals with N_{SV} (for SVM), and $\delta_{outlier}$ (for DBM1) of experimental results for each method. The bold values mean the best RRs in all methods in each database. **Table 4** reveals the number of learning data N_{ld} and the number of outliers $N_{outlier}$ (for DBM1) of the experimental results. The results were obtained using the same $\delta_{outlier}$ values as in Table 3. The N_{ld} for SVM and MLP is the same as given in Table 4 because these two methods just use the given training data. **Table 5** indicates the results of multiple comparison of each method for all databases. We show the results using "+" and "−". The "+" means there is

Table 5 Results of statistical tests in each method for all databases. The "+" means there is a significance, and the "−" means there is no significance. DBM1 results are significantly different from DBM0.

Method	Australian	Breast	Diabetes	German	ILPD	Ionosphere
DBM1 vs. DBM0	+	+	+	+	+	+
DBM1 vs. SVM	−	+	−	+	−	+
DBM1 vs. MLP	+	+	+	+	+	+
DBM0 vs. SVM	+	+	−	+	+	+
DBM0 vs. MLP	+	+	+	+	+	+
SVM vs. MLP	+	+	+	+	+	+

significance, and no significance if the result is "−". For the statistical tests, we confirm normality by using Shapiro-Wilk test [13]. There is no normality for all results by confirming values of Shapiro-Wilk test results. Therefore, we use Mann-Whitney U test [11] with Holm-Bonferroni method [6] for paired comparison.

DBM1 results are the best in 5 out of 6 databases from Table 3. For the Australian database, the best result is SVM, but the SVM's RR is almost the same with DBM1. From Table 5, there is no significance between SVM and DBM1. It indicates that the DBM1 performance is equivalent to the SVM in the Australian database. The DBM1 performance is also equivalent to SVM in Diabetes and ILPD databases. For other databases, the RRs of DBM1 are higher than SVMs. From the comparison of RR results between DBM0 and DBM1 in Table 3, DBM1's RRs are about 0.2–1.2% higher than the DBM0, and these results are significant different in all databases from Table 5. Therefore, the SVM-based outlier detection method is an effective method for improving the performance of DBM algorithm. Especially for ILPD database, the DBM0's RR is less than the SVM. There is significance among them from Table 5. However, to implement the SVM-based outlier detection into DBM algorithm, the performance of DBM1 is the same as that of SVM. We can achieve the main purpose of this paper, i.e., improving DBM performance, by removing outliers.

For the numbers of learning data, we can see differences among methods from Table 4. In the DBM algorithm, it generates new data, and then the number of new training data increases. However, the numbers of training data of DBM0 and DBM1 are greatly increased to approximate the DB of SVMs. If the number of training data is too huge, learning times become extremely long. To reduce the new training data without removing effective data, we will consider a new algorithm to remove insignificance data in the next step.

5. Experiments on Investigating Effective Parameter $\delta_{outlier}$

In this section, we set a new parameter $\delta_{outlier}$ to improve an SVM-based outlier detection method. The parameter $\delta_{outlier}$ can be used to control the boundary of outliers. In this section, we conduct experiments using some different values for $\delta_{outlier}$, and discuss about results. For the experimental environments, i.e., machine specs and databases, etc., are the same as Section 4.

5.1 Experimental Design

To obtain better performance, we confirm the effectiveness of $\delta_{outlier}$ by using some values, and discuss the results. The method for comparison is below:

- DBM1: MLP obtained using the DBM without outliers.

To confirm the effective parameter $\delta_{outlier}$, we changed the values from 0.0 to 0.5 with step size 0.1. Other parameters for SVM and MLP were the same as used in the Section 4.

Table 6 Averaged recognition rates of DBM1 for each $\delta_{outlier}$ setting. The bold font values mean the best recognition rate in all settings. $\delta_{outlier} = 0$ or $\delta_{outlier} = 0.2$ is the best result in most cases.

Database	$\delta_{outlier}$	RR (%)	SD (%)	Confidence Interval (%)
Australian	0	**84.591**	**2.808**	**[84.461, 84.720]**
	0.1	84.588	2.849	[84.457, 84.720]
	0.2	84.490	2.753	[84.363, 84.618]
	0.3	84.477	2.824	[84.347, 84.608]
	0.4	84.426	2.763	[84.298, 84.554]
	0.5	84.490	2.782	[84.362, 84.619]
Breast	0	**96.417**	**1.446**	**[96.350, 96.484]**
	0.1	96.399	1.430	[96.333, 96.466]
	0.2	96.404	1.440	[96.337, 96.471]
	0.3	96.353	1.481	[96.284, 96.421]
	0.4	96.353	1.471	[96.285, 96.421]
	0.5	96.312	1.494	[96.243, 96.382]
Diabetes	0	**76.127**	**3.120**	**[75.983, 76.271]**
	0.1	76.108	3.153	[75.963, 76.254]
	0.2	76.046	3.074	[75.904, 76.188]
	0.3	75.931	3.235	[75.781, 76.080]
	0.4	75.890	3.088	[75.748, 76.033]
	0.5	75.890	3.089	[75.747, 76.032]
German	0	73.035	2.885	[72.901, 73.168]
	0.1	73.092	2.879	[72.959, 73.225]
	0.2	**73.205**	**2.962**	**[73.068, 73.342]**
	0.3	72.591	2.877	[72.819, 73.084]
	0.4	72.986	2.964	[72.849, 73.123]
	0.5	72.832	2.957	[72.695, 72.968]
ILPD	0	71.253	3.707	[71.082, 71.424]
	0.1	71.236	3.613	[71.069, 71.403]
	0.2	**71.261**	**3.614**	**[71.095, 71.428]**
	0.3	71.230	3.842	[71.052, 71.407]
	0.4	71.239	3.759	[71.065, 71.412]
	0.5	71.207	3.838	[71.030, 71.384]
Ionosphere	0	91.877	3.191	[91.730, 92.024]
	0.1	91.809	3.280	[91.657, 91.960]
	0.2	91.741	3.196	[91.594, 91.889]
	0.3	91.685	3.302	[91.532, 91.837]
	0.4	**91.885**	**3.166**	**[91.739, 92.031]**
	0.5	91.699	3.281	[91.548, 91.851]

Table 7 Percentages of number of outliers for each $\delta_{outlier}$ setting of DBM1. The percentages of Diabetes, German and ILPD databases are more than other databases. The percentage of German database greatly decreases when $\delta_{outlier}$ is 0.3.

	DBM1 $N_{outlier}$/Number of given training data (%)					
$\delta_{outlier}$	0	0.1	0.2	0.3	0.4	0.5
Australian	7.96	7.11	6.38	5.64	4.94	4.31
Breast	2.91	2.31	1.78	1.51	1.37	1.12
Diabetes	18.99	16.92	15.08	13.31	11.54	9.67
German	28.17	27.09	24.29	5.38	2.07	1.03
ILPD	28.65	28.59	28.55	28.37	28.15	27.68
Ionosphere	1.67	0.90	0.71	0.64	0.59	0.53

5.2 Analysis and Discussion on Effectiveness of the Parameter $\delta_{outlier}$

Table 6 reveals RRs with SDs and confidence intervals of DBM1 for all $\delta_{outlier}$ settings. The best RR values are bold font in each database. **Table 7** shows percentages of the number of outliers in number of given training data for each result. And **Table 8** indicates the results of paired comparisons to confirm significances between each $\delta_{outlier}$ setting. The "+" shows there is a

Table 8 Results of statistical tests in each DBM1 setting for all databases. The "+" means there is a significance, and the "−" means there is no significance.

$\delta_{outlier}$	Australian	Breast	Diabetes	German	ILPD	Ionosphere
0 vs. 0.1	-	-	-	-	-	-
0 vs. 0.2	-	-	-	-	-	-
0 vs. 0.3	-	-	-	-	-	-
0 vs. 0.4	-	-	-	-	-	-
0 vs. 0.5	-	-	-	-	-	-
0.1 vs. 0.2	-	-	-	-	-	-
0.1 vs. 0.3	-	-	-	-	-	-
0.1 vs. 0.4	-	-	-	-	-	-
0.1 vs. 0.5	-	-	-	-	-	-
0.2 vs. 0.3	-	-	-	-	-	-
0.2 vs. 0.4	-	-	-	-	-	-
0.2 vs. 0.5	-	-	-	+	-	-
0.3 vs. 0.4	-	-	-	-	-	-
0.3 vs. 0.5	-	-	-	-	-	-
0.4 vs. 0.5	-	-	-	-	-	-

significance, and the "−" means there is no significance. There is no normality for all results confirmed by using the Shapiro-Wilk test, and the results are calculated by the Mann-Whitney U test with Holm-Bonferroni method.

About the effective value for $\delta_{outlier}$ from Tables 6–8, the parameter $\delta_{outlier}$ should be set a small value, and the sensitivities of the parameter $\delta_{outlier}$ are low. In all cases except the Ionosphere database, the best results are $\delta_{outlier} = 0$ or 0.2. For the Ionosphere database, the best $\delta_{outlier}$ is 0.4, but the result is almost the same as the result with $\delta_{outlier} = 0$. To see the significances in Table 8, there is no significance for these databases except for the German database. In the German database, the percentages of the number of outliers from Table 7 rapidly decreases when $\delta_{outlier} = 0.3$. True outliers in the German database can be detected if $\delta_{outlier} < 0.3$, and the RR with $\delta_{outlier} \geq 0.3$ is degraded because the outliers are not deleted from the new training set. For other databases, RRs are not significantly different. Moreover, in the experiment in Section 4, DBM1 is significantly different from DBM0. In fact, DBM0 is equivalent with DBM1 with $\delta_{outlier} = \infty$. Thus, if the parameter $\delta_{outlier}$ increases a great deal, the performance of DBM1 becomes low. In other words, the performance of the induced models are not sensitive to the value of $\delta_{outlier}$, if it is small.

We do not need more experiments with other settings for $\delta_{outlier}$. In this experiment, we use only values in [0, 0.5] for $\delta_{outlier}$. If we use other values for the parameter $\delta_{outlier}$, we may get different results. However, to use values more than 0.5 for $\delta_{outlier}$, RRs may decrease because many outliers are not identified as the outlier. Therefore, we should not use large values for $\delta_{outlier}$.

We should reduce the size of the new training set without removing important data. In the experiment of Section 5, we discussed that the size of the new training set may become too large. Data around the DB may be more important than other data, and farther data from the DB may be less informative. Therefore, if we design a boundary to separate whether data are informative or not, we can reduce the number of new training data. Moreover, DBM generates new data to approximate the DB of SVMs. In other words, DBM creates a high density training set to form the DB. High density is necessary, but too high a density is not needed. If we avoid too high a density by reducing some data, we can decrease the number of new training data. We will consider methods to reduce the number of new training data without deleting informative data.

6. Conclusion

In this paper, we have proposed a new DBM algorithm by incorporating the SVM-based outlier detection method. This new DBM algorithm detects outliers from given training data by using the SVM-based outlier detection method, and remove them from the new training set. The DBM with the SVM-based outlier detection generates more efficient new training data than the original DBM algorithm. The main objective of this paper is to avoid performance decreasing in DBM algorithm compared with SVM, because some results of DBM are lower than SVM in previous experiments. Experimental results obtained for 6 databases with statistical tests show that the DBM with the SVM-based outlier detection is significantly improved compared with the original DBM. For other methods, the DBM with the SVM-based outlier detection is the best or equivalent to SVM in all databases. Therefore, we can improve the performance of the DBM algorithm with the SVM-based outlier detection to preserve the performance. And the SVM-based outlier detection method is an effective method for the DBM algorithm.

For the effective parameter $\delta_{outlier}$, we should set small values and the sensitivity of the model performance is low to $\delta_{outlier}$. We can get the best recognition rates when we set $\delta_{outlier}$ to 0 or 0.2 in most cases of this experiment. From the results of statistical tests, there is no significance except for the German database. For other databases, the numbers of outliers are not so different when $\delta_{outlier}$ is in [0, 0.5]. On the other hand, if we set too large values to the parameter $\delta_{outlier}$, the performance of DBM will decrease because the DBM with outlier (it means the parameter $\delta_{outlier}$ is ∞) is lower performance than the DBM without outliers. Therefore, we should use small values for $\delta_{outlier}$.

For future works, we will try to reduce the number of new training sets of the DBM algorithm by removing insignificant data. From the experimental results, the number of new training data is too large. Moreover, the new training set becomes extremely large when we use large databases because the number of newly generated data is about $N \times N_{SV}$. We need to remove less informative data for the objective of the DBM algorithm. More data are not necessarily needed to design compact and high performance MLPs. To generate more efficient new training data, we will propose proper methods to remove ineffective data for obtaining a better DBM performance.

In other works, we will consider incremental methods. In general, when new data are added into the training data, we need to re-design a model using the training data. However, an incremental method can fix the model only using the new data. Thus, learning costs for new data are very low. If the DBM algorithm can adapt to new data on PCDs, usability of the DBM can be higher. We will improve the DBM algorithm to make it suitable for incremental learning.

References

[1] Amidan, B.G., Ferryman, T.A. and Cooley, S.K.: Data outlier detection using the chebyshev theorem, *2005 IEEE Aerospace Conference*, pp.3814–3819 (2005).
[2] Bache, K. and Lichman, M.: UCI machine learning repository (2013).
[3] Dong, E.Q. and Huang, X.: A new compact support kernel of support vector machines, *14th Asia-Pacific Conference on Communications, 2008* (*APCC 2008*), pp.1–4 (2008).
[4] Ferdowsi, H., Jagannathan, S. and Zawodniok, M.: A neural network based outlier identification and removal scheme, *2013 IEEE Conference on Prognostics and Health Management* (*PHM*), pp.1–6 (2013).
[5] Geebelen, D., Suykens, J.A.K. and Vandewalle, J.: Reducing the number of support vectors of svm classifiers using the smoothed separable case approximation, *IEEE Trans. Neural Networks and Learning Systems*, Vol.23, No.4, pp.682–688 (2012).
[6] Holm, S.: A simple sequentially rejective multiple test procedure, *Scandinavian Journal of Statistics*, No.6, pp.65–70 (1979).
[7] Joachims, T.: SVM light (2008), available from ⟨http://svmlight.joachims.org⟩.
[8] Kaneda, Y. and Zhao, Q.F.: Inducing high performance and compact neural networks based on decision boundary making, *IEEJ Trans. Electronics, Information and Systems*, Vol.134, No.9, pp.1299–1309 (2014).
[9] Lee, Y.J. and Huang, S.Y.: Reduced support vector machines: A statistical theory, *IEEE Trans. Neural Networks*, Vol.18, No.1, pp.1–13 (2007).
[10] Lin, K.M. and Lin, C.J.: A study on reduced support vector machines, *IEEE Trans. Neural Networks*, Vol.14, No.6, pp.1449–1459 (2003).
[11] Mann, H.B. and Whitney, D.R.: On a test of whether one of two random variables is stochastically larger than the other, *The Annals of Mathematical Statistics*, Vol.18, No.1, pp.50–60 (1947).
[12] Rumelhart, D.E., Hinton, G.E. and Williams, R.J.: Learning representations by back-propagating errors, *Nature*, Vol.323, No.6088, pp.533–536 (1986).
[13] Shapiro, S.S. and Wilk, M.B.: An analysis of variance test for normality (complete samples), *Biometrika*, Vol.3, No.52, pp.591–611 (1965).
[14] Steinwart, I.: Sparseness of support vector machines – Some asymptotically sharp bounds, *Advances in Neural Information Processing Systems 16*, MIT Press (2004).
[15] Vapnik, V.N.: *Statistical learning theory*, Wiley, 1 edition (1998).
[16] Watarai, K., Zhao, Q. and Kaneda, Y.: Decision boundary learning based on an improved pso algorithm, *2012 IEEE International Conference on Systems, Man, and Cybernetics* (*SMC*), pp.2958–2962 (2012).
[17] Wang, H.W., Kong, B. and Zheng, X.Y.: An Improved Reduced Support Vector Machine, *2010 IEEE Youth Conference on Information Computing and Telecommunications* (*YC-ICT*), pp.170–173 (2010).

Editor's Recommendation

The authors propose a method to improve the performance of the decision boundary making (DBM) algorithm via outlier detection. The DBM algorithm with compact and high performance learning models is suitable for implementation in portable computing devices. Experimental results show that the improved DBM incorporated with outlier detection outperforms the original DBM, and is better than or comparable to SVM for all the databases used in the experiments.

Yuya Kaneda was born in 1990. He received his B.E. from University of Aizu in 2012 and his M.E. in 2014. He is pursuing his Ph.D. at University of Aizu. His research interests include computational intelligence, evolutionary computing, neural networks, and awareness computing. He is a student member of IEEE, and the IPSJ.

Yan Pei is an assistant professor at the School of Computer Science and Engineering, University of Aizu, Japan. He received a doctorate from Kyushu University, Japan, in 2014. He received his B.Eng. and M.Eng. degrees from Northeastern University, China in 2006 and 2009, respectively. His professional career includes software engineer, project manager at Neusoft Co. Ltd.; software engineer at Alpine Electronics Europe D&R GmbH. He research interests include in computational intelligence and machine learning. He obtained the Best Paper Award from ICGEC2012, and the Best Excellent Presentation Award from SOFT-Kyushu2012. He is a member of IEEE SMC, IEEE CIS and the Japanese Society for Evolutionary Computation.

Qiangfu Zhao received his Ph.D. degree from Tohoku University of Japan in 1988. He joined the Department of Electronic Engineering, Beijing Institute of Technology, China in 1988, first as a post doctoral fellow and then an associate professor. He was an associate professor from October 1993 at the Department of Electronic Engineering, Tohoku University, Japan. He joined University of Aizu, Japan from April 1995 as an associate professor, and became a tenure full professor in April 1999. Professor Zhao's research interests include image processing, pattern recognition and understanding, computational intelligence, neurocomputing, evolutionary computation and awareness computing.

Yong Liu is currently a senior associate professor at University of Aizu, Japan. He received his Ph.D. degree from Wuhan University, China, and the University of New South Wales, Australia, in 1994, and 1999, respectively. His research interests include evolutionary computation and neural networks. He is a member of IEEE.