Preselection of Support Vector Candidates by Relative Neighborhood Graph for Large-Scale Character Recognition

Masanori Goto*†, Ryosuke Ishida* and Seiichi Uchida†
*Research & Development Center, GLORY LTD., Hyogo, Japan
{gotou.masanori, r.ishida}@mail.glory.co.jp
†Kyushu University, Fukuoka, Japan
uchida@ait.kyushu-u.ac.jp

***Abstract*—We propose a pre-selection method for training support vector machines (SVM) with a large-scale dataset. Specifically, the proposed method selects patterns around the class boundary and the selected data is fed to train an SVM. For the selection, that is, searching for boundary patterns, we utilize a relative neighborhood graph (RNG). An RNG has an edge for each pair of neighboring patterns and thus, we can find boundary patterns by looking for edges connecting patterns from different classes. Through large-scale handwritten digit pattern recognition experiments, we show that the proposed preselection method accelerates SVM training process 5–15 times faster without degrading recognition accuracy.**

I. Introduction

Large-scale datasets are effective for training classifiers. For example, the classification accuracy of a support vector machine (SVM) [1] improves as the size of the training dataset increases [2]. However, large-scale datasets generally increase the computational complexity of the training process. Furthermore, for an SVM, we need to repeat the training process to optimize various parameters, such as the soft margin and kernel parameters. This means that the computational complexity of SVM training becomes intractable and must be reduced.

Among the various strategies for reducing the computational complexity of SVM training, preselection of relevant patterns from the entire training dataset is promising. In other words, we use only relevant patterns to train an SVM and discard irrelevant ones. It is important to note that relevant patterns for SVM training include patterns to be chosen as support vectors (SVs). Thus, if we can preselect SV candidates appropriately, we can reduce computation without any degradation of recognition accuracy.

We propose a method for preselecting SV candidates by analyzing the data distribution via network representation for an efficient SVM classifier training process. Specifically, we represent the entire training dataset as a relative neighborhood graph (RNG) [3]. An RNG is an undirected graph where neighboring patterns tend to be connected by an edge and thus, we can select boundary patterns as SV candidates by looking for the edges connecting patterns of different classes.

The contributions of this paper are as follows: First, we show through experiments on the MNIST handwritten digit dataset [4] (60,000 training images) and our original handwritten digit dataset (518,850 training images) that the proposed preselection method can reduce the training data to 10% without degradation of recognition accuracy. This accelerates the SVM training process by 5–15 times. Second, we show that the preselected SV candidates coincide well with the original SVs. In other words, we show that the classification boundary after preselection is almost the same as the original one. Third, we confirm that the above two points hold for any digit class, which implies that the proposed method works well for any data distribution. Fourth, we develop an efficient RNG constructor to minimize the overhead associated with preparing the RNG for preselection.

II. Related Work

It is often difficult to use all n data in the dataset. This is because the computation for training an SVM with given parameter values increases quadratically with n even using well-known efficient methods such as chunking [5] and sequential minimal optimization (SMO) [6]. In the case of the SMO algorithm [6], computation time for a d-dimensional vector scales with $\Omega(n^2 d)$ [7]. Moreover, since we have multiple parameters (soft margin parameter and kernel parameters) for optimization, we need to repeat the above training method many times while gradually changing the parameter values. Thus, we need to reduce the training data without, of course, degrading classification performance of the SVM. One promising observation is that the final SVM classifier relies only on SVs. Thus, if we select SV candidates from the entire training dataset, we can accelerate the SVM training process without any degradation of recognition accuracy.

Many preselection methods [2], [8]–[15] have been proposed to reduce the computational complexity. These methods can be categorized into two types; safe and non-safe methods. The safe method was initially proposed by Ghaoui et al. [8] as a way of selecting features whose coefficients would be zero in the optimal solution without actually calculating the solution. As an example of the safe preselection of training data, Ogawa et al. [9] proposed safe screening of non-SVs in pathwise SVM computation. However, as the criteria for non-SVs are different for each set of SVM parameter values, safe preselection needs to be carried out iteratively.

A typical strategy for non-safe preselection is to estimate

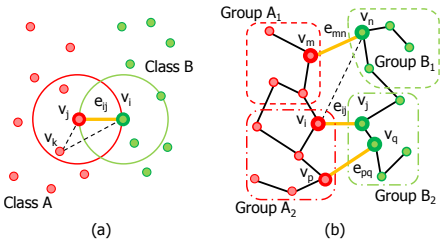

Fig. 1. Construction of RNG and preselection of SV candidates using the RNG. (a) Rule for having an edge for a node pair (v_i, v_j). There is no node in the intersection between two hyper-spheres centered at nodes v_i and v_j. (b) Rule for selecting an SV candidate. The six nodes depicted as thicker circles have an edge connecting nodes from different classes. These are called bridge vectors and are used as SV candidates.

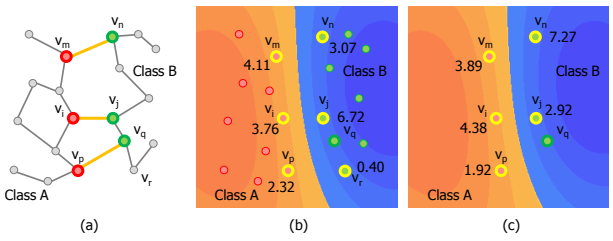

Fig. 2. Visualization of the discrimination function by a two-class SVM. Data selected as SVs are shown as yellow circles with weight α. (a) The six bridge vectors are selected as SV candidates from the training data. (b) The SVM classifier is trained with all vectors. The six data are selected as SVs with the non-bridge vector v_r included in these. (c) The SVM classifier is trained with the bridge vectors. Although the selected SVs differ from (b), the classification boundaries are almost the same.

a class distribution and exclude interior data. For example, Koggalage et al. [12] used cluster analysis to identify clusters. They excluded the interior data of clusters because the data were unlikely to be SVs. Several methods based on a proximity graph have been proposed for preselection (β-skeleton [13], Gabriel graph [14], RNG [15]). In [15], which is the study most closely related to our proposal, only a very small-scale experiment (with about 1,600 training data) and a very brief inspection (showing only the final accuracy without any internal analysis) was carried out. In contrast, we establish the usefulness of RNG-based preselection through far larger experiments with deeper inspections based on the four novel contributions in Sec. I[1].

III. Preselection of Support Vector Candidates by Relative Neighborhood Graph

A. Relative Neighborhood Graph

An RNG [3] is an undirected graph where neighboring patterns tend to be connected by an edge. It is more intuitive than the nearest neighbor graph, which is a directed graph [16]. Owing to this good property, RNGs have been used widely in research on computer vision, geographic analysis, pattern classification, and so on [17].

Let $\boldsymbol{P} = \{v_1, v_2, ..., v_n\}$ and $\boldsymbol{E} = \{e_{ij}\}$ denote the sets of nodes and edges, respectively, of an RNG. Each node corresponds to a handwritten digit pattern represented by a d-dimensional feature vector. Each edge e_{ij} is prepared iff $d(v_i, v_j) \leq \max\{d(v_i, v_k), d(v_j, v_k)\}$ for $k = 1, ..., n, k \neq i, k \neq j$, where $d(v_i, v_j)$ is the Euclidean distance between nodes v_i and v_j. Fig. 1(a) illustrates this condition; intuitively, an edge e_{ij} exists iff there is no node in the intersection between two hyper-spheres centered at nodes v_i and v_j.

B. Preselection of Support Vector Candidates

SV candidates are selected according to a distribution represented by the RNG. Specifically, the nodes around the class boundary are selected as SV candidates. In the case of Fig. 1(b), the six nodes have an edge connecting nodes from different classes. These nodes are selected as SV candidates, and are hereafter, referred to as *bridge vectors*.

[1]Unfortunately, this paper [15] showed no positive effect of RNG-based preselection. These pessimistic results may have been due to the small dataset, which is not, actually, a target of preselection.

Even in the case of a multi-class dataset, an RNG construction is required only once and we can select bridge vectors using the RNG as SV candidates for two-class SVM training for each class. For example, for the SVM for class "0" vs. the other classes, bridge vectors with an edge connected to the node labeled class "0" are selected. In the case of handwritten digit datasets, based on our empirical results, about 10% of the training data are selected as bridge vectors for training a two-class SVM.

Each SV has a weight parameter, α, where SVs with a larger weight are more important. In the case of an SVM with a radial basis function (RBF) kernel, the classification function is $f(\boldsymbol{x}) = \text{sign}(\sum_{i \in \boldsymbol{S}} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b)$, where $\boldsymbol{S}$ is the set of SVs. SVs with larger weights contribute more to the classification boundary.

An SVM classifier trained with bridge vectors only is expected to form a classification boundary almost the same as that trained with the full dataset. Fig. 2(b), which shows six SVs, depicts the actual classification boundary trained with all the (nineteen) data of (a). Note that we used an RBF kernel here. Conversely, (c) shows the boundary trained with only six bridge vectors and then formed by five SVs. Even though the SVs differ in (b) and (c), the boundaries formed by them are almost the same. In Sec. VI, we show experimentally that the results of this small example are also realizable with a larger dataset. It is worth noting that in (b), the non-bridge vector v_r is an SV with a small weight (0.40) and thus, contributes very weakly to the classification boundary.

IV. Efficient RNG Construction Algorithm

To minimize the overhead of RNG-based preselection, we need an efficient method for constructing an RNG. In other words, if the RNG construction requires a huge amount of computation, the benefits of preselection are nullified. The native algorithm for constructing an RNG [3] for N data requires $O(N^3)$ computational complexity and $O(N^2)$ memory (about 125 GB for our larger dataset). These constraints are intractable for large-scale character recognition and therefore, need to be mitigated.

We developed a memorization algorithm [18] that only stores the necessary intermediate results adaptively. This algorithm also contributes to time efficiency because we do not need to access unnecessary results. An outline of our RNG

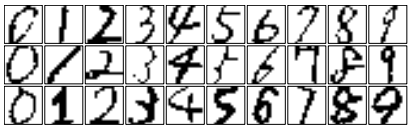

Fig. 3. Example of digit patterns from our original large-scale handwritten dataset (LS-Hand).

construction algorithm is presented below. Computation in an RNG construction is mainly associated with searching for the nearest node that prevents an edge from forming, however, it is not only the nearest node to prevent an edge from forming.

Thus, if we can detect such a node in fewer steps, construction of the RNG can be accelerated. We search for approximate nearest node set which is a set of the nearest nodes from one node to other nodes for each group. Specifically, in Step 1 of the RNG-1 algorithm [3], we randomly create m groups[2] of nodes for each class and memorize the minimum distances for each group, $d(v_i, v_k)$, and node indexes of v_k, $d(v_i, v_k) = \min\{d(v_i, v_h)\}$ for $k = 1, ..., m, h = k_1, ..., k_l, h \neq i$, where l is the number of nodes in each group. These node v_k are the approximate nearest nodes from v_i.

Node pairs which may form an edge are effectively narrowed down using those minimum distances. For example, as shown in Fig. 1(b), during the evaluation of node pair v_i and v_n, node v_j is detected as one of approximate nearest nodes from v_i using the memorized minimum distances and $d(v_n, v_j)$ is evaluated because $d(v_i, v_j)$ is smaller than $d(v_i, v_n)$. As the result, v_j is determined to prevent the edge from forming. Only those node pairs that do not have such an approximate nearest node under this condition are evaluated in Steps 2 and 3 of the RNG-1 algorithm [3].

This algorithm requires only the minimum distance to other groups from each node and node index of the approximate nearest nodes. They consumes only about 250 MB storage for our larger dataset. Moreover, as shown in Sec. VI, computation time for RNG construction using this algorithm scales with $O(n^2)$. Thus, this algorithm minimizes the overhead in preparing the RNG for preselection and is suitable for RNG construction of a large-scale dataset.

V. Experimental Setup

A. Dataset and Distance Metric

We applied the RNG-based preselection method to two-digit image datasets: the MNIST dataset [4] and our original large-scale handwritten (*LS-Hand*) dataset [16]. Fig. 3 shows several digit patterns from our dataset. The ground-truth, i.e., correct class label ("0",..., "9"), is attached to each data. Each pattern is binarized and represented as a 784-dimensional (MNIST) or 256-dimensional (LS-Hand) binary vector. MNIST contains 60,000 training images and 10,000 test images, while LS-Hand contains 518,850 training images and 100,000 test images.

As the distance metric, we employed Hamming distance for RNG construction and Euclidean distance for SVM classifier training. Hamming distance can accelerate RNG construction because it has lower computational complexity than Euclidean distance. The distribution of the RNG by Hamming distance is the same as that by Euclidean distance, because the magnitude relation of the Hamming distance of binary vectors is the same as that of Euclidean distance.

B. SVM Training and Evaluation

In this paper, we used a soft-margin SVM implemented in LIBSVM [19] with an RBF kernel. LIBSVM implements the SMO-based algorithm. The soft-margin SVM with the RBF kernel has two parameters: C controls the margin from a class boundary, and σ controls the variance in the RBF kernel. We optimized the parameters using cross-validation accuracy [20].

We analyzed the total computation time for the SVM classifier training with a full dataset and bridge vectors of the dataset for 10 classes[3]. Bridge vectors were selected only once for each training dataset and the computation time for RNG construction was included in the preparation time for the RNG and preselection of bridge vectors by the RNG for each class. RNG construction was carried out six times, five times with the cross-validation subsets and once using all the training data.

In addition, we evaluated the error rate of the SVM classifiers and properties of the SVs, that is, the number of SVs and the weight α of the SVs. As shown in Fig. 2, even if different SVs are selected, the important SVs with larger weights form a classification boundary that is almost the same as the original one, and there is no degradation of recognition accuracy.

VI. Results and Analysis

A. Results on MNIST

First, we prepared three different datasets from the MNIST dataset: (i) full dataset including all the images in the dataset, (ii) bridge vectors containing selected images from RNG-based preselection for each class, and (iii) random subset with randomly selected images. The number of images selected for the random subset was the same as the number of bridge vectors for each class. We evaluated the SVM classifiers trained with each dataset.

The computation times are shown in Table I. On average, 5,313 images, which are about 10% of the full dataset, were selected as bridge vectors for each class. The bridge vectors accelerated the computation time by about 15 times from that of the full dataset. The computation time of the full dataset and the random subset scales with $O(n^2)$. However, although the size of the bridge vectors is the same as the random subset, the bridge vectors require more computation time. We surmise that the computation complexity of optimizing the SVs is greater, because the bridge vectors consist of patterns around the class boundary only.

In addition, the overhead of preparing the RNG is sufficiently small to realize speedup. When excluding the parameter

[2]Having experimented with parameter m, we found that it was not sensitive to the computation time of RNG construction. Therefore, we used $m = 10$ in this experiment.

[3]In this experiment, parameters C and σ were optimized by 5-fold cross-validation and grid search with the grid interval set to 4 and 9, respectively. Computation time was measured in a computing environment consisting of an Intel Core i7 @ 3.40 GHz 4-core (8-core), with 8 GB RAM running Windows 7 64-bit operating system.

TABLE I. COMPUTATION TIMES FOR MNIST

Training Dataset	Full	Bridge (Proposed)	Random
#Training Data	60,000	5,313	5,313
RNG Construction (s)	–	7.61×10^2	–
Parameter Optimization (s)	1.20×10^6	7.57×10^4	2.42×10^4
SVM Training (s)	4.48×10^3	2.06×10^2	6.71×10^1
Total (s)	1.20×10^6	$\mathbf{7.67 \times 10^4}$	2.43×10^4

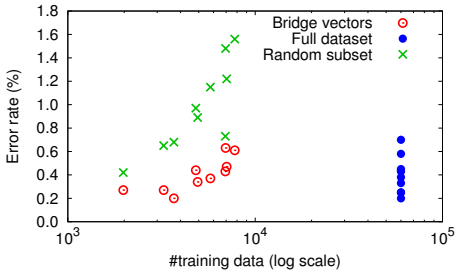

Fig. 4. Error rate for different training data from MNIST.

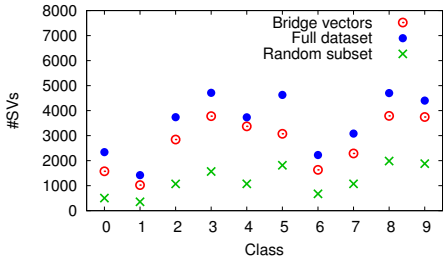

Fig. 5. Number of SVs using different training data from MNIST.

optimization process, the total SVM training time with bridge vectors takes 3.89×10^2 s (RNG construction 1.83×10^2 s for the full dataset and SVM training 2.06×10^2 s for ten classes). This is faster than the SVM training time for the full dataset (4.48×10^3 s).

Fig. 4 plots the error rate of the SVM classifiers for each class. While the proposed preselection method suffers no degradation of recognition accuracy using the full dataset, the training dataset size is reduced to 10%. Comparison with the random selection shows that bridge vectors are far better candidates as SVs. This means that the proposed method works well for any data distribution.

Fig. 5 plots the number of SVs for each class. The number of SVs for the bridge vectors is slightly smaller than that for the full dataset, which means that SVM classifiers trained with bridge vectors have better performance in the classification step. By contrast, the number of SVs for the random subset is obviously smaller than that of the others. This is the reason for degradation of recognition accuracy of the random subset; there are not enough training data to learn class boundaries.

We observed that about 60% of the SVs for the full dataset were common to the SVs for the bridge vectors and the average weight α of these SVs was larger than that of the SVs not included in the SVs for the bridge vectors as shown in Fig. 6. This means that important SVs are included in bridge vectors and are selected as SVs for the SVM classifier trained with bridge vectors. It is interesting that the same vectors are selected as important SVs, although some different vectors are selected as SVs. This is the reason for the preservation of recognition accuracy shown in Fig. 4.

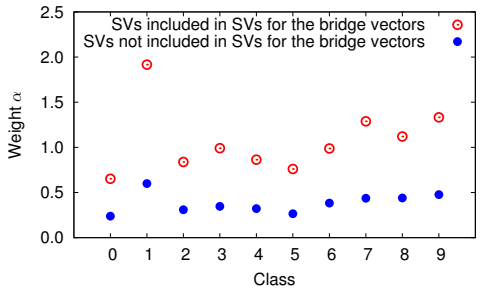

Fig. 6. Average weight α of SVs for the full dataset of MNIST.

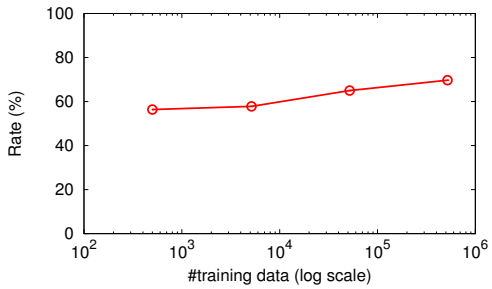

Fig. 7. Average population of SVs for the full dataset common to the SVs for the bridge vectors of LS-Hand.

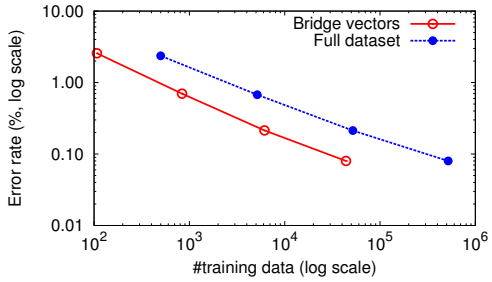

Fig. 8. Average error rate for different training data sizes of LS-Hand.

B. *Results on Large-scale Handwritten Dataset*

We evaluated the scalability of RNG-based preselection using subsets of the LS-Hand dataset. We prepared subsets by random selection of each degree from the dataset, and bridge vectors were selected from each subset. The computation times are shown in Table II. The bridge vectors optimized the parameters and trained SVM classifiers in 9.54×10^5 s. On the other hand, the SVM classifier training with 518,850 training data took 5.50×10^6 s (about 64 days). This is one of the constraints of large-scale character recognition, even if multiple PCs are used to parallelize and accelerate the training.

In the case of bridge vectors, about 10% of the images for each subset were selected as bridge vectors and the computation time accelerated by about 5 times, despite an increase in the size of the training data. For example, on average 43,967 images were selected as bridge vectors for each class and the bridge vectors accelerated the SVM classifier training to about 17.3% for the full dataset (518,850 images).

The overhead of preparing the RNG is sufficiently reduced thereby achieving acceleration for the results on MNIST. Even

TABLE II. COMPUTATION TIMES FOR LS-HAND

Training Dataset Size (%)	100	10	1	0.1	100	10	1	0.1
Preselection Method	None				Bridge Vectors of RNG (Proposed)			
#Training Data	518,850	51,880	5,180	510	43,967	6,142	837	107
RNG Construction (s)	–	–	–	–	4.75×10^4	3.55×10^2	5.60×10^0	≤ 1
Parameter Optimization (s)	5.46×10^6	1.62×10^5	3.32×10^3	2.91×10^2	9.02×10^5	2.14×10^4	9.71×10^2	1.43×10^2
SVM Training (s)	3.73×10^4	1.15×10^3	1.52×10^1	≤ 1	4.34×10^3	7.66×10^1	4.46×10^0	≤ 1
Total (s)	5.50×10^6	1.64×10^5	3.34×10^3	2.92×10^2	$\mathbf{9.54 \times 10^5}$	$\mathbf{2.18 \times 10^4}$	$\mathbf{9.81 \times 10^2}$	$\mathbf{1.44 \times 10^2}$

when excluding the parameter optimization process, the total SVM training time with the bridge vectors took 1.58×10^4 s (RNG construction 1.15×10^4 s for the full dataset and SVM training 4.34×10^3 s for ten classes). This is faster than the SVM training time for the full dataset (3.73×10^4 s).

Fig. 7 plots the average population of SVs for the full dataset common to SVs for the bridge vectors of LS-Hand. The average population of those SVs increases as the dataset increases. The RNG of a large-scale dataset represents the class distribution well, while preselected bridge vectors are more suitable as SV candidates.

Fig. 8 plots the average error rate of the SVM classifiers for each class. The error rates for both classifiers decrease exponentially as the dataset increases. The reduction in error rate for the bridge vectors is slightly larger than that for the full dataset. In addition, for each class, we observed that recognition accuracy is preserved and the number of SVs of the bridge vectors is slightly smaller than that of the full dataset, similar to the case for the results on MNIST. This result confirms that the proposed method works well for any data distribution. Thus, the proposed RNG-based preselection is scalable and effectively reduces the training data even if the dataset increases.

VII. CONCLUSION

In this paper, we proposed a method for preselecting SV candidates by RNG for large-scale character recognition. This method preselects bridge vectors, which have an edge connecting the different labeled nodes on the RNG as SV candidates. The RNG has an edge for each pair of neighboring patterns and thus, we can find boundary patterns by looking for the edges connecting patterns of different classes. The contributions of this paper are summarized below.

- The proposed RNG-based preselection method reduces the training data to 10% without degradation of recognition accuracy, thereby accelerating the SVM training process by 5–15 times.
- The preselected SV candidates coincide well with the original SVs. In other words, the important SVs with a larger weight α are selected from the bridge vectors and recognition accuracy is preserved, because the classification boundary after preselection is almost the same as the original classification boundary.
- The above two points hold for any digit class and the proposed method works well with any data distribution.
- We developed an efficient method for constructing RNGs. The computation time of the method scales with $O(n^2)$ and the algorithm is memory-efficient. The method minimizes the overhead for preparing the RNG for preselection.

We validated these contributions through experiments on MNIST (60,000 training images) and our original handwritten digit dataset (518,850 training images).

REFERENCES

[1] C. Cortes, and V. Vapnik, "Support-vector networks," in *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995.

[2] J. Wang, P. Neskovic, L.N. Cooper, "Training Data Selection for Support Vector Machines," in *Proc. ICNC*, pp. 554–564, 2005.

[3] G.T. Toussaint, "The Relative Neighbourhood Graph of a Finite Planar Set," in *PR*, vol. 12, pp. 261–268, 1980.

[4] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, "Gradient-Based Learning Applied to Document Recognition," in *Proc. IEEE*, vol. 86, no 11, pp. 2278–2324, 1998.

[5] B.E. Boser, I.M. Guyon, Y. Bengio, V.N. Vapnik, "A Training Algorithm for Optimal Margin Classifiers," in *Proc. COLT*, pp. 144–152, 1992.

[6] J.C. Platt, "Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines," B. Schölkopf, C.J.C. Burges, A.J. Smola, Eds, in *Advances in Kernel Methods - Support Vector Learning*, MIT Press, Cambridge, USA, 1998.

[7] A.K. Menon, "Large-Scale Support Vector Machines: Algorithms and Theory," Res. Exam, University of California, San Diego, 2009.

[8] L.E. Ghaoui, V. Viallon, T. Rabbani, "Safe Feature Elimination in Sparse Supervised Learning," eprint arXiv:1009.3515, 2010.

[9] K. Ogawa, Y. Suzuki, I. Takeuchi, "Safe Screening of Non-Support Vectors in Pathwise SVM Computation," in *Proc. ICML*, 2013.

[10] N. Jankowski, M. Grochowski, "Comparison of Instances Selection Algorithms I. Algorithms Survey," in *Proc. ICAISC*, pp. 598–603, 2004.

[11] M. Grochowski, N. Jankowski, "Comparison of Instances Selection Algorithms II. Results and Comments," in *Proc. ICAISC*, pp. 580–585, 2004.

[12] R. Koggalage, S. Halgamuge, "Reducing the Number of Training Samples for Fast Support Vector Machine Classification," in *Neural Inform. Process.-Lett. and Rev.*, vol. 2, no 4, pp. 57–65, 2004.

[13] W. Zhang, I. King, "Locating Support Vectors via β-Skeleton Technique," in *Proc. ICNIP*, pp. 1423–1427, 2002.

[14] W. Zhang, I. King, "A Study of the Relationship between Support Vector Machine and Gabriel Graph," in *Proc. IJCNN*, vol. 1, pp. 239–244, 2002.

[15] X. Liu, J.F. Beltran,, N. Mohanchandra, G.T. Toussaint, "On Speeding Up Support Vector Machines: Proximity Graphs Versus Random Sampling for Pre-Selection Condensation," in *Proc. ICCSM*, vol. 73, pp. 1037–1044, 2013.

[16] M. Goto, R. Ishida, Y. Feng, S. Uchida "Analyzing the Distribution of a Large-Scale Character Pattern Set Using Relative Neighborhood Graph," in *Proc. ICDAR*, pp. 3–7, 2013.

[17] G.T. Toussaint, "Applications of the Relative Neighbourhood Graph," in *IJCSIA*, vol. 4, no 2, pp. 77–85, 2014.

[18] D. Michie, "Memo Functions and Machine Learning," in *Nature*, vol. 218, no 5136, pp. 19–22, 1968.

[19] C.-C. Chang, C.-J. Lin, "LIBSVM : A Library for Support Vector Machines," in *ACM TIST*, vol. 2, no 3, pp. 27:1–27:27, 2011.

[20] C.-W. Hsu, C.-C. Chang, C.-J. Lin, "A Practical Guide to Support Vector Classification," Tech. Rep., National Taiwan University, 2003.