Accelerating the Training Process of Support Vector Machines by Random Partition

Hongzhi Xu, Chunping Li, Li Li, and Hongyu Shi

***Abstract*—In this paper we present a novel method, *Random Partition based SVM* (RPSVM), for speeding up SVM training. Instead of clustering the training data prior to training, RPSVM randomly partitions the training data into several clusters and then uses the centers of the clusters to train an initial SVM. This trained SVM is used to find critical clusters which are located on the decision boundary. The same procedure is applied repeatedly to each of the critical clusters, resulting in a refined SVM which consists of the supporting vectors in the initial round of training and those in the repeated round. This procedure is repeated recursively until no critical cluster exists, resulting in the final SVM. Our experiments on synthetic and real data sets have shown that RPSVM is indeed scalable to large data sets while the high performance is retained.**

***Index Terms*—Support vector machine, SVM training, classification, supervised learning.**

I. Introduction

Support Vector Machine [1], [2] is a concept, deeply rooted in statistics, that represents a set of methods for classification and regression analysis. In its original form, a support vector machine aims to find a set of support vectors of a hyper band separating two classes of data that has a maximal margin. The hyper band is then used to classify new data instances. Since its inception, support vector machine has quickly become popular and has been widely used in research and practice, and has spawned many variants.

There are two kinds of main SVM training schemes to reduce training time on large data sets. One is the *optimization oriented* training scheme which focuses on reducing time spent on the optimization process of a SVM training algorithm; the other is and the *data oriented* training scheme which focuses on reducing time spent on exploring the data by way of sampling or clustering. These two schemes can even be combined since they focus on different phases of the training process. The sampling based approach utilizes the strategy of random sampling to train SVM. The main problem is that if less support vectors existed in the training data, the probability that they are randomly sampled becomes quite lower. This would result directly in the loss of important information for data classification. The clustering-based SVM training approach would partition the original data set into smaller clusters using clustering algorithms such as k-means and BIRCH, etc. which represents the original data set by the centers or their centroids along with statistical information, and then apply the SVM algorithm on the reduced (much smaller) data set.

However, clustering itself is a costly operation and sampling may suffer from important information loss [3], [4], therefore there is a good reason for exploring new and better SVM training schemes in order to reduce the total training time on large data sets. In this paper we present a novel SVM training scheme, *Random Partition based SVM* (RPSVM), which partitions the data set randomly rather than through clustering. It first of all partitions the positive and negative data separately into *d* clusters each, and then uses the centers of the clusters as a training data set and trains an initial SVM. After that, it partitions the *critical* clusters (those that contain both positive and negative data instances) into *d* sub-clusters to get a finer partition of the data set and train SVM with centers of these sub-clusters. This process is repeated until there is no critical cluster, resulting in the final SVM.

RPSVM is a hierarchical training scheme based on data partitioning. It can be viewed as clustering based SVM training approach, but quite different to the traditional ones which usually use a costly clustering algorithm. Therefore it can be expected to be much faster than the traditional approaches and at the same time retain the outstanding performance of the original SVM and the clustering based SVM. Therefore, we compare RPSVM with two well known SVM training schemes (to deal with larger dataset problem) -- RSSVM (Random Sampling SVM) and CBSVM (Clustering Based SVM). We here use the standard SVM as the baseline.

The rest of the paper is organized as follows. In Section II, we review related works on SVM training for dealing with large data sets. In Section III, we present our training scheme RPSVM in details. In Section IV, we present experimental results on synthetic data sets and real data sets along with discussion and analysis. Section V concludes the paper and discusses future work.

II. Related Works

To speed up the optimization process of SVM training, Joachims [5] proposed the Chunking algorithm, which is based the decomposition strategy [6] and addresses the problem of selecting the variables for the training data set in an effective and efficient way. SMO, proposed by Platt [7], is another algorithm that trains SVM efficiently which decomposes the overall QP problem into sub-problems and chooses to solve the smallest possible optimization problem at every step. Joachims [8] presented a Cutting Plane

Manuscript received January 10, 2014; revised April 18, 2014.

Hongzhi Xu and Chunping Li are with School of Software, Tsinghua University, Beijing, China (e-mail: cli@ tsinghua.edu.cn).

Li Li and Hongyu Shi are with the Shannon Lab, HUAWEI Technologies CO.LTD, Beijing, China (e-mail: jollylili.li@huawei.com).

algorithm which is used to train Linear SVM in a linear time for classification task. In [9] a fast algorithm for solving linear SVMs with loss function toward data mining tasks for large data sets was developed. In [10] an iterative algorithm to solve the SVM optimization problem by alternating between stochastic gradient descent and projection was proposed.

Parallel to the training schemes focusing on optimization as discussed above, it is also possible to speed up SVM training by applying the training algorithm on a transformed version of the original data set rather than on the original one. This scheme exploits the fact that a trained SVM centers around a set of support vectors defining the decision boundary and these vectors can be found without having to trawl through the complete data set. The *optimization oriented* training scheme and the *data oriented* training scheme focus on different phases of the training process so they can be combined one way or another, resulting in different training schemes. A data oriented training scheme can be based on *sampling* or *clustering*. Random Sampling [11] is perhaps the best known one of the sampling based approach. It works by sampling a small proportion of data to approximately reflect the distribution of the entire data set. Experiments have shown that this scheme efficiently works well but sometimes has poor performance because it may lose important information while sampling [3], [4]. *Active learning* [9] is developed to reduce the costly labeling work by selecting "important" data instances in the data set, and only requiring user to consider these important data instances for further labeling. Interestingly, active learning can also be used to accelerate the training process of SVM [12], [13]. A clustering-based training scheme needs a clustering algorithm to preprocess the training set, and a well known algorithm is k-means [14]. Boley *et al.* [15] proposed a scheme to accelerate SVM training using a clustering algorithm. Those clusters that contain only support vectors are retained and those that do not contain any support vector are ignored. Those clusters that contain both support vectors and non support vectors are partitioned into smaller clusters, which are considered recursively. The best known clustering based SVM training scheme is CBSVM [4], which is based on the clustering algorithm BIRCH [16]. It works by first building two hierarchical cluster feature (CF) trees for positive and negative classes respectively and then working through the two trees to find the support vectors.

III. Random Partition Based SVM

A. Motivation

Support Vector Machine (SVM) is a powerful classifier and is widely used for various classification tasks. However, its training algorithm has relatively high time complexity thus making it difficult to use the classifier for large data sets. To overcome this limitation, various SVM training schemes have been proposed, some being optimization oriented and some data oriented as discussed in Section II. In data oriented schemes, a data set is either sampled or clustered to create representatives which are then used as data to train the SVM. This process is repeated until it cannot go anywhere or some condition is satisfied. The sampling based scheme may suffer from information loss at times while the clustering based scheme spends much time on clustering making the scheme less efficient, therefore there is a need to search for alternative training scheme that does not have the above undesirable features.

A SVM model is represented by a set of support vectors, which are determined by the data near the boundary of the classes. We thus may hypothesize that clustering the training set, in order to find the support vectors, is not necessary. For instance, CBSVM uses BIRCH to cluster the training set to build CF trees, but in practice many nodes of the trees are not used at all during the training process. If we can identify/discard those data instances that are too far away from the support vectors, we may end up with a more efficient training scheme while retaining SVM's outstanding performance. This section will present such a scheme called RPSVM or Random Partition based SVM.

RPSVM uses a SVM training algorithm (such as SMO or Chunking) to find critical clusters which is crossed by margin and contain data instances that have a high probability to be the support vectors. The critical clusters are further split into smaller ones, and their centers are added to the training set to train SVM again. Since it would split the critical clusters, the support vectors near the boundary are likely to be found after several iterations. Gradually, it will converge to the optimal solution equivalent to that found on the entire data set.

B. Algorithm Description

Let D be a (large) data set with two classes, with CP being the class of positive instances and CN being the class of negative instances. RPSVM first selects *d* samples randomly from CP, and uses these *d* samples as seeds to partition CP into *d* subsets (or clusters) by the nearest-neighbor method. For each data instance *p*, RPSVM calculates the similarity between *p* and the center of each of the *d* clusters, and then assigns *p* to the cluster that has the largest similarity with *p*. The same process is repeated on CN. We denote a cluster C as a triple (*cc, iL, r*), where cc is the center of the cluster; *iL* is a list of indexes for the data instances that belong to the cluster; and *r* is the radius of the cluster. The cluster center and radius, *cc* and *r*, are defined as follows:

$$cc = \frac{\sum_{i=1}^{|C|} x_i}{|C|}$$

$$r = \left(\frac{\sum_{i=1}^{|C|} \|x_i - cc\|^2}{|C|}\right)^{\frac{1}{2}}$$

where |*C*| is the number of instances in cluster *C*. Critical clusters are *support clusters*, *i.e.* those that are crossed by the margin. Generally, a cluster is critical if it satisfies the following condition.

$$d_i - r_i < d_s \quad (1)$$

where *di* is the distance between the center of cluster C_i and the decision boundary, *ds* is the distance between the center

of support cluster C_s and the boundary.

Let's denote the decision function as $f(x) = \langle w^* \cdot x \rangle + b$. Then the above condition (1) is the same as the following one.

$$y_i \times f(x_i) < 1 + r_i \times \left\| w^* \right\| \quad (2)$$

where $y_i \in \{1, -1\}$ is the label of C_i. The proof of (2) is a simple geometrical problem, so we don't state it here. The centers of $2 \times d$ clusters of CP and CN are used as the initial training set to train an initial SVM. The *critical clusters* {*Ci*} are identified.

After RPSVM finds the critical clusters, it will partition them into *d* sub-clusters respectively. In other words, RPSVM only uses the centers of the critical sub-clusters as the new training set rather than all the sub-clusters. All the new clusters will be maintained in a the 'CheckList'. For example, cluster *Ci* is partitioned into *d* sub clusters{*Ci*,1,*Ci*,2, ..., *Ci* ,*d*}, and only *k* of them{*Ci*,1,*Ci*,2, ..., C*i*, *k*} are critical, so the centers of the *k* sub-clusters are added to the new training set, and the (*d*−*k*) sub-clusters{*Ci*,*k*+1,*Ci*,*k*+2, ..., *d*}are not added to the new training set. It is just maintained in the CheckList and will be checked in the next iteration. Finally, RAPSVM trains SVM with the new training set and get a new solution. Algorithm 1 describes the main algorithm of RAPSVM and its corresponding definition of functions is shown in Algorithm 2 and Algorithm 3.

Algorithm 1: Main Algorithm of RAPSVM

Input: *Training Set P and N; Degree d.*
Output: *Decision Function f.*
Notation:
partition(C): partition the cluster C randomly into d sub-clusters.
getNewSet(model): get new training set by splitting the critical clusters.
CheckList: a list to maintain all the clusters.
Algorithm:
CP = partition(P); CN = partition(N);
$C = C_P \bigcup C_N$
Add all clusters in C to CheckList;
Do Loop
model = train a new SVM with C;
C = getNewSet(model);
Until C is null

Algorithm 2: Algorithm of Getting New Training Set in Each Iteration

Reference: CheckList
Function getNewSet(h)
CNew = null;
For each ci in CheckList
If ci is a critical cluster then
C' = partition(ci);
Delete ci from CheckList;
Add all clusters in C' to CheckList;
For each c'i in C'
If c'i is a critical cluster then
$C_{New} = C_{New} \bigcup c'_i$
End If
End For
End If
End For
Return CNew;
End Function

Algorithm 3: Algorithm of Random Partition

Function partition(C)
ci = null; i = 1,…, d.
Random select d data from C as center i, i = 1,…,d.
For each di in C
Find j, such that di is nearest to center j;
$c_j = c_j \bigcup d_i$
End For
Return {ci};
End Function

IV. Experiments

A. Environment

In this section, we provide experimental evidence for our analysis for RPSVM using synthetic and real data sets. Then we will discuss the experimental results and give an explanation for them. All of our experiments are done on a computer with Xeon 2.66GHZ CPU and 2.0GB memory. Since we mentioned that the algorithms that deal with the optimization problem are of different level to the clustering based algorithms, and the two methodologies can be combined, here, we mainly compare four different algorithms: standard SVM, RSSVM, CBSVM and RPSVM.

The standard SVM is referred to LibSVM [17] here. The algorithms RSSVM, RPSVM and CBSVM are implemented based on it. We mainly use the default parameters of LibSVM. Since determined by the character of BIRCH that a fixed order of data will generate the same CF trees, we give CBSVM a random input order of the training data to evaluate the stability of performance of CBSVM. Also, in our experiments, we neglect the time cost by CBSVM for tuning the parameter t(rebuilding the trees) automatically, and just give a fixed value as suggested in [4]. All the experiments in this section give the average value of ten running results and also the overall accuracy.

B. Datasets

Synthetic Data Sets: We use a data generator to generate 2-dimension data(*x, y*) for some visual comparison and analysis. Without losing generalization, we first fix the decision function to be $y = x$. The generator has 5 parameters: *cn, N, δmax, δmin, margin*. *cn* is the total number of cluster that will be generated, and all the data in one cluster are generated by a non-standard Gaussian distribution, with the cluster center as the expectation and δ as the standard variance. In our experiments all the data values are in the interval (0.0, 1.0), i.e. $0 < x < 1$ and $0 < y < 1$. *N* is sample size. *δmax* is maximal value of δ, and *δmin* is minimal value of δ. The value of δ is randomly chosen from (δmin, δmax). *margin* is the nearest distance from the data point in positive and negative classes to the decision hyper-plane, e.g. $y = x$.

The generation is involved two main steps as follows. The first step is to generate cluster centers, and the second step is to generate data points around these cluster centers.

The generator first randomly selects *cn* points as the cluster centers, if the distance between the center and $y = x$ is less than margin, the generator will regenerate a new point as the new cluster center. Then it assigns a random δ to each cluster respectively. If the cluster center point (xi,yi) is beyond the $y = x$, that is $yi > xi$, the cluster will be labeled as positive, and if $yi < xi$, the cluster will be labeled as negative.

Then, the generator begins to generate *N* data points. It first randomly selects one center *ci* from the *cn* clusters, and then generates a new data point through a non-standard Gaussian distribution with expectation *ci* and standard variance δi, finally the data is labeled the same as *ci*. The process repeats until all of *N* data are generated.

We use this data generator to generate a data set of size 400000 and a test set of size 10000, denoted by data set 'SYN-1' and generate a data set of size 5000000 and a test set of size 10000, denoted by data set 'SYN-3'. Similarly, we change the decision function to be $y = sin(x)$ to generate nonlinear separable data sets. We use this generator to produce a data set of size 400000, and a test set of size 10000, denoted by data set 'SYN-2' and a larger nonlinear data set of size 5000000 and a test set of size 10000, denoted by 'SYN-4'..

Real Data Sets Besides synthetic data sets, we also use a large real data set, the UCI Archive used by KDD Cup 1999 (http://kdd.ics.uci.edu/databases/kddcup99.html). This data set consists of about five millions of training data and three hundred thousands of testing data. Each data object consists of 41 features. We normalized the continuous feature values into between zero and one as follows.

$$Val_{norm} = \frac{val - \min Val}{\max Val - \min Val}$$

where *Val*norm is the normalized value; *val* is the original value; *maxVal* is the maximum value of the attribute in the data set and *minVal* is the minimum value of the attribute in the data set. In order to fit the all training set in memory we only randomly select 50% of the its original training set, which consists of about 2449215 data, test set of 311029 data. We denote this data set by 'NWI'.

C. Experiment and Results

We conduct an experiment on SYN-1 and SYN-2, two relatively small data sets, to compare the performance of standard SVM, CBSVM and RPSVM in terms of running time and accuracy. We set d =4 for RPSVM, and d = 100 and t =0.01 for CBSVM on SYN-1 and set 4 referential data to generate artificial data for all nonlinear experiments and use RBF kernel on SYN-2. The experimental result is shown in Table I, where *Sampl* is the sampling time and here the clustering time for CBSVM and the partition time for RPSVM; *Train* is the training time and *Total* is the sum of *Sampl* and *Train*. *Accu* is the accuracy. We can see that, RPSVM and CBSVM are both faster than standard SVM, while RPSVM is much faster. All the algorithms get a high accuracy, though CBSVM shows a slight difference. This is mainly because of that the data set is almost evenly distributed in the input space and it is easy to separate. Our principal purpose is to compare the running time of different algorithms, however, the result also show that RPSVM can guarantee the accuracy while significantly reducing the running time.

TABLE I: EXPERIMENTAL RESULTS ON SYN-1 AND SYN-2

DATASET	ALGORITHM	SAMPL.	TRAIN.	TOTAL	ACCU.
SYN-1	SVM		48.156	48.156	0.9953
	CBSVM	16.554	0.056	16.610	0.9932
	RAPSVM	1.378	0.320	1.698	0.9953
SYN-2	SVM		38.578	38.578	0.9999
	CBSVM	17.987	0.056	18.043	0.9996
	RAPSVM	2.367	0.251	2.618	0.9999

Fig. 1 and Fig. 2 display the data used in the whole training process of RSSVM, CBSVM and RPSVM. Fig. 1 is on the data set SYN-1 and Fig. 2 is on the data set SYN-2. We could see that RSSVM loses important data near the bound of each class, while CBSVM and RPSVM both have a bias to select more data near the decision boundary.

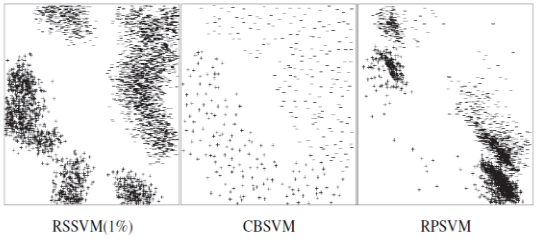

Fig. 1. The display of the data used in the training process on SYN-1.

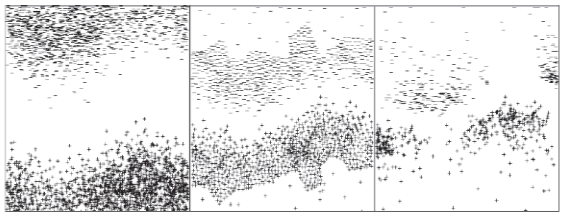

Fig. 2. The display of the data used in the training process on SYN-2.

TABLE II: EXPERIMENTAL RESULTS ON SYNTHETIC DATA SET SYN-3 AND REAL DATA SET NWI WITH LINEAR KERNEL

DateSet	Algorithm	Sampl.	Train.	Total.	Accu.
SYN-3	RSSVM(0.5%)	0.032	1.990	2.023	0.9991
	RSSVM(1.0%)	0.129	18.181	18.310	0.9992
	RSSVM(5.0%)	0.237	45.923	46.160	0.9992
	CBSVM	142.771	0.062	142.834	0.9991
	RPSVM	13.375	0.380	13.755	0.9990
NWI	RSSVM(0.5%)	0.018	1.125	1.143	0.9121
	RSSVM(1.0%)	0.028	3.562	3.590	0.9190
	RSSVM(5.0%)	0.059	58.200	58.259	0.9198
	CBSVM	529.715	17.006	546.721	0.8413
	RPSVM	24.253	77.653	101.906	0.9192

We do our second experiments to mainly evaluate the linear algorithm that we proposed and compare it to RSSVM, CBSVM on SYN-3 and NWI. We didn't consider the standard SVM here, because the time and space complexity of standard SVM has made it impossible to run on a very large data set. We tried to run standard SVM on 40% of SYN-3 data set and it had run out of our available memory. In this experiment, we have sample 0.5%, 1% and 5% for RSSVM on both data sets, and here we have selected best results in several random sampling trials respectively. The

experimental result is shown in Table II. We can see that RPSVM is always faster than CBSVM and sometimes faster than RSSVM. All the algorithms get high precision, except that CBSVM perform poor on NWI data set. This is because some importance data may have been compressed by clustering. RSSVM shows a high accuracy on these two data sets, because the sampling data just meets with the distribution of the training data set, namely, most support vectors are still selected during the sampling.

Our third experiment is done on SYN-4 and NWI and we use RBF kernel to testify our algorithm to deal with nonlinear cases. We present the result for RBF kernel shown in Table III, and for RSSVM on both data sets, sample is still proportional to 0.5%, 1% and 5% respectively. It shows a similar result as our second experiment, except that RSSVM got a lower precision on NWI data set, which means that RSSVM is not a reliable algorithm like shown in [3] and the performance of RSSVM depends on the distribution of the data set. Similar problem also exists for CBSVM, since it doesn't have a tendency to keep important data near the boundary uncompressed during building the CF trees. RPSVM partitions the critical clusters until it is a single data point, so the importance data near the boundary will not be lost.

TABLE III: EXPERIMENTAL RESULTS ON SYNTHETIC DATA SET SYN-4 AND REAL DATA SET NWI WITH RBF KERNEL

DateSet	Algorithm	Sampl.	Train.	Total.	Accu.
SYN-3	RSSVM(0.5%)	0.023	0.700	0.723	0.9995
	RSSVM(1.0%)	0.039	1.832	1.871	0.9996
	RSSVM(5.0%)	0.125	15.834	15.959	0.9997
	CBSVM	145.237	0.034	145.271	0.9995
	RPSVM	15.767	0.185	15.953	0.9999
NWI	RSSVM(0.5%)	0.012	0.970	0.982	0.8873
	RSSVM(1.0%)	0.020	2.965	2.985	0.9124
	RSSVM(5.0%)	0.059	48.734	45.793	0.9185
	CBSVM	550.470	23.455	573.925	0.9202
	RPSVM	148.462	29.895	178.357	0.9240

V. CONCLUSION

In this paper we present a novel SVM training scheme, RPSVM. It adopts the idea of gradually refining critical clusters found in every iteration, and doesn't use any clustering algorithms to preprocess the training set to get meaningful clusters for reflecting the original data distribution, which degrades the time complexity. We give the theoretical analysis which concludes that RPSVM has a linear time complexity on a large data set and further validate it with experiments. The other advancement is that RPSVM avoids important information loss occurred probably in the approach of random sampling SVM during data compression or reduction and guarantee the classification accuracy.

REFERENCES

[1] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annual Workshop on Computational Learning Theory*, Pittsburgh, ACM, 1992, pp. 144-152.

[2] V. N. Vapnik, *Statistical Learning Theory*, John Wiley and Sons, 1998.

[3] J. Wang, P. Neskovic, and L. N. Cooper, "Training data selection for support vector machines," in *Proc. International Conference on Neural Computation*, 2006.

[4] H. Yu, J. Yang, and J. Han, "Classifying large data sets using SVM with hierarchical clusters," in *Proc. 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[5] T. Joachims, "Making large-scale support vector machine learning practical," *Advances in Kernel Methods: Support Vector Learning*, MIT Press, 1999.

[6] E. Osuna, R. Freund, and F. Girosi, "An improved training algorithm for support vector machines," in *Proc. IEEE NNSP*, 1997.

[7] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," *Advances in Kernel Methods: Support Vector Learning*, MIT Press, 1999.

[8] T. Joachims, "Training linear SVMs in linear time," in *Proc. 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2006.

[9] G. Schohn and D. Cohn, "Less is more: active learning with support vector machines," in *Proc. 17th International Conference on Machine Learning*, 2000.

[10] S. Shwartz, Y. Singer, and N. Srebro, "Pegasos: Primal Estimated sub-GrAdient SOlver for SVM," in *Proc. 24th International Conference on Machine Learning*, 2007.

[11] J. L. Balczar, Y. Dai, and O. Watanabe, "Watanabe, a random sampling technique for training support vector machines," in *Proc. 13th International Conference on Algorithmic Learning Theory*, 2001.

[12] S. Ong and D. Koller, "Support vector machine active learning with applications to text classification," in *Proc. 17th International Conference on Machine Learning*, 2000.

[13] R. Koggalage and S. Halgamuge, "Reducing the number of training samples for fast support vector machine classification," *Neural Information Processing, Letters and Reviews*, 2004.

[14] M. B. Almeida and A. P. Braga, "SVM-KM: Speeding SVMs learning with a priori Cluster Selection and k-Means," in *Proc. Neural Networks 2000, 6th Brazilian Symposium*, pp. 162-167, 2000.

[15] D. Boley and D. Caoy, "Training support vector machine using adaptive clustering," in *Proc. SIAM International Conference on Data Mining*, 2004.

[16] T. Zhang, R. Ramakrishnan, and M. Livny, "BIRCH: an efficient data clustering method for very large databases," in *Proc. ACM SIGMOD International Conference on Management of Data*, 1996.

[17] C. C. Chang and C. Lin. LIBSVM: a library for support vector machines. [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm

Hongzhi Xu received the BSc degree in computer science from Chengdu Polytechnic University, China, in 2004 and the MSc degree in software engineering from Tsinghua University, China in 2008. He worked at NEC Institute in Beijing as a software engineer from 2008 to 2011. He is currently a PhD candidate in the Hong Kong Polytechnic University. His research interests include machine learning, data mining and natural language processing.

Chunping Li received the BS and MSC degrees in computer science from Jilin University, China in 1985 and 1988, and the PhD degree from Darmstadt University of Technology, Germany in 1999. He is currently an associate professor in Tsinghua University, China. His research interests include machine learning and automated reasoning, data analysis and mining. He has published more than 80 research papers in related fields.

Hongyu Shi received his BSc and MSc degree in computer science from Harbin Institute of Technology, China in 2010 and 2013, respectively. Now he is a research engineer of Central Research Institute's Shannon Lab, Huawei Technologies Co., Ltd. His research interests lie in context-aware computing, mobile data mining and human-computer interface.

Li Li received her BSc and MSc degrees in School of Information and Communication Engineering, from Beijing University of Posts and Telecommunications, China, in 2008 and 2011 respectively. Now she is a research engineer of Central Research Institute's Shannon Lab, Huawei Technologies Co., Ltd. Her research interests lie in context-aware computing, mobile data mining and human-computer interface.