Training data reduction to speed up SVM training

Article *in* Applied Intelligence · September 2014

DOI: 10.1007/s10489-014-0524-2

Senzhang Wang
Nanjing University of Aeronautics & Astronautics
107 PUBLICATIONS **773** CITATIONS

SEE PROFILE

Xiaoming Zhang
Beihang University (BUAA)
63 PUBLICATIONS **479** CITATIONS

SEE PROFILE

Appl Intell (2014) 41:405–420
DOI 10.1007/s10489-014-0524-2

Training data reduction to speed up SVM training

Senzhang Wang · Zhoujun Li · Chunyang Liu · Xiaoming Zhang · Haijun Zhang

Published online: 15 March 2014

Abstract Traditional Support Vector Machine (SVM) solution suffers from $O(n^2)$ time complexity, which makes it impractical to very large datasets. To reduce its high computational complexity, several data reduction methods are proposed in previous studies. However, such methods are not effective to extract informative patterns. In this paper, a two-stage informative pattern extraction approach is proposed. The first stage of our approach is data cleaning based on bootstrap sampling. A bundle of weak SVM classifiers are constructed on the sampled datasets. Training data correctly classified by all the weak classifiers are cleaned due to lacking useful information for training. To further extract more informative training data, two informative pattern extraction algorithms are proposed in the second stage. As most training data are eliminated and only the more informative samples remain, the final SVM training time is reduced significantly. Contributions of this paper are three-fold. (1) First, a parallelized bootstrap sampling based method is proposed to clean the initial training data. By doing that, a large number of training data with little information are eliminated. (2) Then, we present two algorithms to effectively extract more informative training data. Both algorithms are based on maximum information entropy according to the empirical misclassification probability of each sample estimated in the first stage. Therefore, training time can be further reduced for training data further reduction. (3) Finally, empirical studies on four large datasets show the effectiveness of our approach in reducing the training data size and the computational cost, compared with the state-of-the-art algorithms, including PEGASOS, LIBLINEAR SVM and RSVM. Meanwhile, the generalization performance of our approach is comparable with baseline methods.

Keywords Data cleaning · Pattern extraction · Bootstrap · Entropy maximization

S. Wang · Z. Li (✉) · X. Zhang · H. Zhang
State Key Laboratory of Software Development Environment, Beihang University, Beijing 100191, P.R. China
e-mail: lizj@buaa.edu.cn

S. Wang
e-mail: szwang@cse.buaa.edu.cn

X. Zhang
e-mail: yolixs@buaa.edu.cn

H. Zhang
e-mail: haijun_cumtb@yahoo.com.cn

C. Liu (✉)
National Computer Network Emergency Response Technical Team, Coordination Center of China, Beijing 100029, P.R. China
e-mail: lcy@isc.org.cn

1 Introduction

As an effective machine learning algorithm based on Statistical Learning Theory (SLT), Support Vector Machine (SVM) has become increasingly popular recently. It is widely used in many fields, such as data mining [1], information retrieval [2], social network [3], and disease recognition [28]. However, traditional SVM solution needs $O(n^2)$ time complexity, which makes it impractical to large datasets. Though many effective solutions are proposed, SVM suffers a lot from its high time complexity when dealing with millions of training data.

An important research direction to speed up SVM training is training data reduction. Based on it, several effective solutions have been proposed. For example, Burges reported that the final SVM classifier only depended on a few samples which are called support vectors, and removing non-support vectors would not significantly affect the performance of classifier [4]. Panda et al. only selected the instances close to the boundary between classes, and the instances likely to be non-support vectors were eliminated [5]. Motivated by previous studies, this paper aims to address such an interesting and challenging problem: *Can we find a minimal subset of the initial training dataset on which the SVM trained can achieve desirable generalization performance?* If such a subset is found, the training time can be significantly reduced due to the elimination of a large number of training data with little information for training. To address this challenge, a novel data cleaning and informative pattern extraction approach is proposed. This approach consists of two stages: *data cleaning based on bootstrap sampling* and *informative pattern extraction based on maximum information entropy*.

In the first stage, a bulk of small subsets of the initial training dataset are randomly sampled with the bootstrap sampling method. Then with each sampled small dataset, we train a weak SVM classifier and utilized it to pre-classify the whole training dataset. Data correctly classified by all the weak classifiers are eliminated, and only those misclassified by at least one weak classifier are retained to the second stage. To further extract the most controversial data, two informative pattern extraction algorithms based on information entropy maximization are provided in the second stage. The information entropy of each data is measured by its empirical misclassification probability estimated in the first stage.

Contributions of this paper are summarized as follows:

– A bootstrap sampling based data cleaning method is proposed. Compared with existing data-processing methods, such as concept boundary detection [5] and Cascade SVM [6], our approach is more effective because it can be executed in parallel. A bundle of independent weak SVM classifiers are constructed and run simultaneously, which greatly reduce the time of data cleaning.
– To further discover more useful training data and eliminate outliers, two informative pattern extraction algorithms are presented. Information entropy of each sample is approximately measured based on its empirical misclassification probability estimated in the first stage. Advantages of the proposed approaches are as follows. First, misclassified training data from the first stage are weighted based on their information entropy. Second, outliers with little information entropy are eliminated. Moreover, training time is further reduced for training data further reduction.
– We evaluate our approach on four large real datasets. The empirical experiments show that our approach can significantly reduce the size of training data and dramatically speed up the SVM training. Comparable experiments against three state-of-the-art SVM algorithms also demonstrate that the generalization performance of our approach outperforms all the three methods with similar training time.

The rest of the paper is organized as follows: Sect. 2 reviews related work. Section 3 describes the *bootstrap sampling based data cleaning*. A heuristic method to determine the sampling proportion is presented in this section. Section 4 details the *informative pattern extraction* process. Two informative data extraction algorithms based on information entropy maximization are proposed. Experiment and evaluation are given in Sect. 5. Finally, we conclude this paper in Sect. 6.

2 Related work

Previous works related to speed up SVM training can be categorized into data-reduction level and quadratic programming (QP) algorithm level. The data-reduction level approaches mainly focus on how to reduce the training data quantity without losing much useful information. The QP-algorithms focus on how to make the QP solver faster [9–13]. As our approach falls in the data-reduction level methods, here we mainly discuss related work in this level.

To reduce the training dataset size, one straightforward idea is randomly down-sampling the training dataset. Lee and Mangasarian proposed to randomly select a subset $\overline{n}$ of the entire dataset n to be training dataset [14]. Experiments show that the reduced SVM (RSVM) they proposed is much faster while its performance is close to SVM. Compared to random down-sampling method, more informed sampling methods are proposed. For example, Garf et al. proposed a novel SVM algorithm which can be executed in parallel: Cascade SVM [6]. In this method, the dataset is split into several disjoined subsets and optimized separately with multiple SVMs. The partial results are combined and filtered again in a "Cascade SVMs", until the global optimum is reached. Panda et al. proposed a concept boundary detection method to speed up SVM [5]. In this paper, training data which are likely to be non-support vectors are eliminated in the concept-independence preprocessing stage. In the concept-specific sampling stage, they effectively select useful training data for each target concept. Yu et al. used a hierarchical micro-clustering technique to capture the training data that were close to the decision boundary [8]. Lawrence et al. proposed an Information Vector Machines (IVMs) [7]. A framework for sparse Gaussian process methods which used forward selection with criteria based on information theoretic principles was proposed in this paper.

Recently, with the growing importance of handling very large datasets, some optimization methods with a more moderate scaling on the dataset size are presented. SVM-Perf is an optimization method which uses a cutting plane approach for training linear SVM [17]. Smola et al. reported that SVM-Perf can find a solution with accuracy ε in time $O(nd/(\lambda\varepsilon))$[1] [18]. Shai et al. proposed PEGASOS, a simple and effective iterative algorithm for solving the optimization problem cast of SVM [20]. PEGASOS alternates between stochastic gradient descent steps and projection steps. The number of iterations required to obtain a solution of accuracy ε is $O(1/\varepsilon)$, compared with other methods which requires $O(1/\varepsilon^2)$. For a linear kernel, PEGASOS is guaranteed to find, with high probability, an ε-accurate solution in time $\widetilde{O}(d/(\lambda\varepsilon))$.[2] LIBLINEAR is a novel SVM algorithm proposed to process large-scale sparse data [19]. This method is especially effective for text classification. Fan et al. reported that LIBLINEAR can reach an ε-accurate solution in $O(\log(1/\varepsilon))$ iterations [19]. The run time of PEGASOS and LIBLINEAR do not increase with the sample size, so the training process runs much faster than tradition methods. The flip side is that the two methods have much worse dependence on the optimization accuracy.

3 Bootstrap sampling based data cleaning

In this section, we first give a brief introduction to SVM. Then we describe the framework of data cleaning based on bootstrap sampling. In addition, to effectively determine the appropriate sampling percentage, a heuristic method is proposed.

3.1 SVM introduction

Given the training dataset $X = \{(x_1, y_1)\dots(x_n, y_n)\}$ in space $X \subseteq R^d$, where $x_i \subseteq R^d$ represents the d-dimensional pattern and $y_i = \pm 1$ represents the class labels, the goal of SVM training is to find a linear predictor $g(x) = w^T x + w_0$ with a small empirical loss relative to a large classification "margin". With this predictor, the testing samples are separated according to the inequalities:

$$\hat{y}_i = \begin{cases} 1 & \text{if } (w^T \cdot x_i + w_0) \geq 1; \\ -1 & \text{if } (w^T \cdot x_i + w_0) \leq -1 \end{cases} \tag{1}$$

when X is linearly separable, a unique hyperplane maximizing the margin between the projections of the training data of two classes can be found. If the training data X is not linearly separable, (1) must be modified for allowing the classification violations in the SVM formulation. In the case of linear inseparable, the SVM can be generalized by introducing a non negative variables $\xi_i \geq 0$ such that (1) can be modified to

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, n \tag{2}$$

when training data x_i can not be linearly separated, its corresponding ξ_i is nonzero. Thus the term $\sum_{i=1}^n \xi_i$ can be thought of a measure of the amount of misclassification.

The optimal hyperplane problem is then regarded as the solution to the problem

$$\begin{aligned} &\text{Minimize} && \frac{1}{2} w \cdot w + C \sum_{i=1}^{n} \xi_i \\ &\text{Subject to} && y_i(w \cdot x_i + b) \geq 1 - \xi_i, \quad i = 1, \dots, n \\ &&& \xi_i \geq 0, \quad i = 1, \dots, n \end{aligned} \tag{3}$$

Equation (3) is a quadratic programming problem, and its dual formulation can be rewritten as follows:

$$\begin{aligned} &\text{Maximize} && W(\alpha) = -\frac{1}{2} \times \sum_i^n \sum_j^n \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ &&& \qquad + \sum_i^n \alpha_i \qquad (4) \\ &\text{Subject to} && \sum_i^n y_i \alpha_i = 0, \quad 0 \leq \alpha_i \leq C, i = 1, \dots, n \end{aligned}$$

where $\alpha = (\alpha_1, \dots, \alpha_n)$ are nonnegative Lagrange multipliers associated with the constrain (1), and K is the kernel matrix.

If such a optimal hyperplane is found, we can use a empirical loss function to measure how different the prediction output $\hat{y}_i$ of the classifier from the true label y_i. The empirical loss function can be formally defined as

$$f(W; X) = \frac{1}{n} \sum_i \max\{0, 1 - y_i \langle w, x_i \rangle\} + \frac{\lambda}{2} \|w\|^2$$

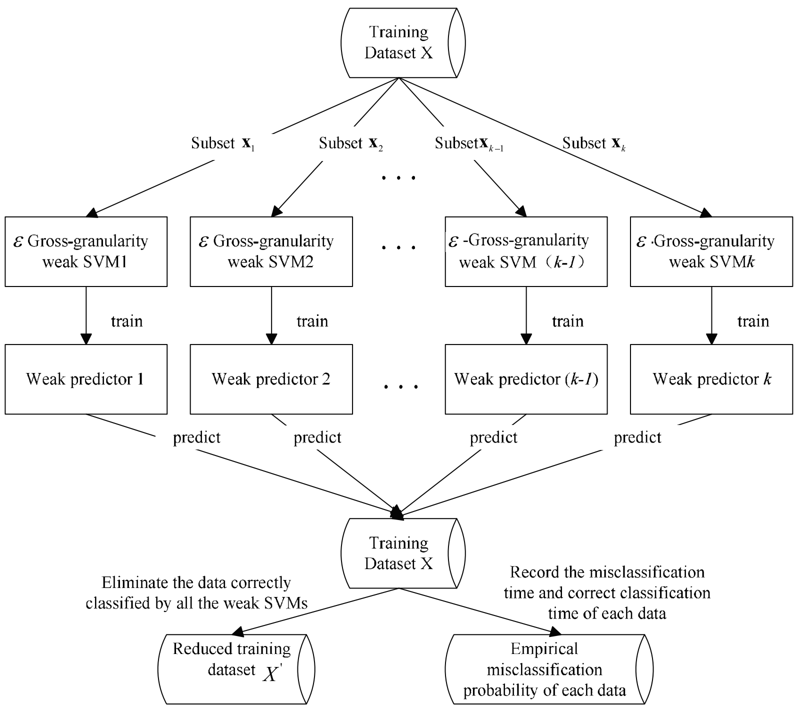
3.2 Bootstrap sampling based data cleaning with weak SVMs

Many previous works have focused on using sampling methods [7, 8, 12, 15] or ensemble methods [31, 32] to speed up SVM training and improve the classification performance. However, instead of training data cleaning, these approaches mainly focused on how to train an approximate SVM classier. Though some works tried to apply boosting method to speed up SVM training, they mainly focused on reducing the training data dimensions [30]. Different from previous works, we propose to apply the bootstrap sampling

[1] d is a bound on the number of non-zero features for dataset and λ is the regularization parameter of SVM.

[2] The $\widetilde{O}(\cdot)$ notation hides logarithmic factors.

Fig. 1 Framework of bootstrap sampling based data cleaning with weak SVMs


Training Dataset X
Subset $\mathbf{x}_1$
Subset $\mathbf{x}_2$
Subset$\mathbf{x}_{k-1}$
Subset $\mathbf{x}_k$
ε Gross-granularity weak SVM1
ε Gross-granularity weak SVM2
ε -Gross-granularity weak SVM (k-1)
ε ·Gross-granularity weak SVMk
train
Weak predictor 1
Weak predictor 2
Weak predictor (k-1)
Weak predictor k
predict
Training Dataset X
Eliminate the data correctly classified by all the weak SVMs
Record the misclassification time and correct classification time of each data
Reduced training dataset X'
Empirical misclassification probability of each data

method to clean useless training data. Meanwhile, our empirical results show that, in many cases, the performance of the weak SVMs trained on the sampled small subset can be fairly good. Therefore, we first sample some small datasets by the bootstrap sampling method. Then a bundle of weak SVMs are trained on the sampled datasets. Next, the weak SVMs are used to pre-classify the whole training data. If the data is correctly classified or misclassified by all the weak SVMs, we consider that it has little information for the training process, and hence we simply clean it. The remaining data are sent to the next stage, and their misclassification times are recorded. Moreover, as these weak SVMs are constructed independently, they can be executed in parallel. The framework of our method is shown in Fig. 1. Before describing our method in detail, we first give a definition as follow.

Definition 1 ε-Gross-granularity weak SVM: Assume $X = \{x_1, \ldots, x_n\}$ is the training dataset, $\widetilde{X}$ is a subset of X and the cardinality of $\widetilde{X}$ is much smaller than X. That is $\widetilde{X} \subset X$, and $Card(\widetilde{X}) \ll Card(X)$. Assume $g(\widetilde{x}) = \widetilde{w}^T \cdot \widetilde{x} + \widetilde{w_0} = 0$ is the SVM predictor of $\widetilde{X}$ and $g(x) = w^T \cdot x + w_0$ is the SVM predictor of X. $f(W; X)$is the empirical loss function of $g(x)$. The SVM of $\widetilde{X}$ is called ε-Gross-granularity weak SVM of X if the empirical loss function meet the following equation:

$$f(\widetilde{W}; \widetilde{X}) \leq f(W; X) + \varepsilon \tag{5}$$

where ε is a predefined constant.

According to Definition 1, if we have a bundle of ε-Gross-granularity weak SVMs with a proper ε, we can pre-classify the training dataset X with them and eliminate the useless training data. However, a difficult, yet important problem is how to find the proper ε-Gross-granularity weak SVMs. Prior study has shown that the empirical error of SVM is directly bounded above with the size of training dataset [21]. Peter et al. proved that the maximum discrepancy G_n of kernel classifications between hypothesis F and real probability distribution on training dataset X meets the following conditions [21].

$$G_n(F) \leq \frac{2B}{n}\sqrt{\sum_{i=1}^{n} K(x_i, x_j)} = 2B\sqrt{\frac{E(K(X, X))}{n}} \tag{6}$$

Here B is a nonnegative constant and $K(X, X)$ is the kernel matrix. Formula (6) shows that the performance of hypothesis predictor depends on the size of the training dataset, but they do not have a linear correlation. Shai and Srebro proved that in low-norm and linear condition, to get a desired generalization error of $f(W; X) + \varepsilon$, at least $m = O(\frac{\|w\|^2}{\varepsilon^2}) = O(\frac{E(K(X,X))}{\varepsilon^2})$ samples are needed [16]. Though the relationship between the size of training dataset and the error bound is clear, how weak the classifiers should be (the value of ε), as well as how many training data are needed, is still an open problem. To address this problem, a heuristic method is proposed. Before describing our approach, we first give the following definition.

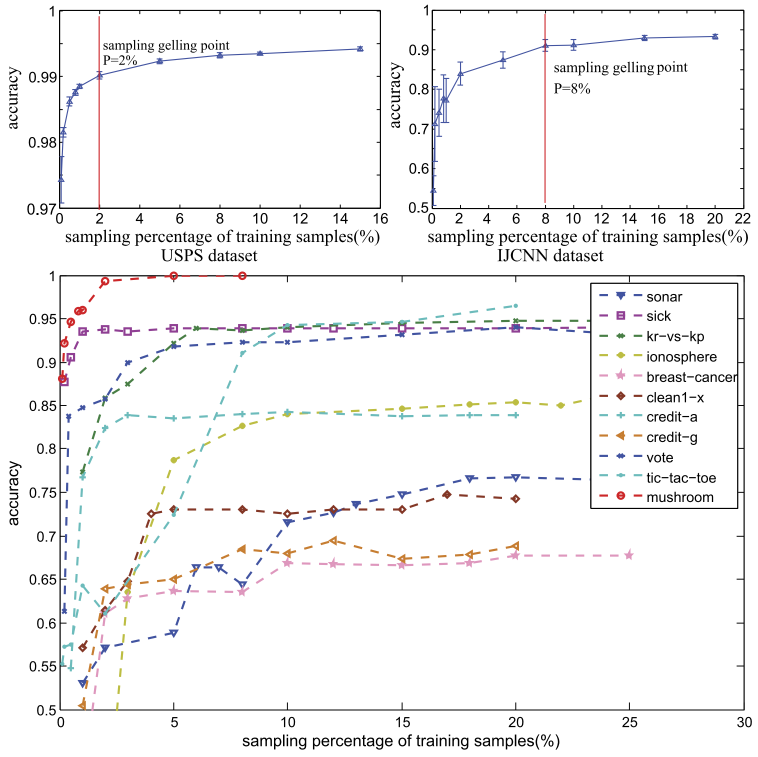


Fig. 2 Examples of sampling Gelling Point phenomenon. *The upper two figures* show experimental results of *IJCNN1* and *USPS* datasets. Gelling Point of the two datasets are 2 % and 8 % respectively. The *lower figure* shows the experimental results of 11 toy datasets from UCI repository. For these datasets, two thirds randomly selected data are used as training data and the rest are used as testing data. Experimental results show that for most of the toy datasets, the proportion at the Gelling Point is less than 5 %

Definition 2 Sampling Gelling Point: Assuming $X = \{x_1, \ldots, x_n\}$ is a training dataset. P_i and P_{i+1} $(P_i < P_{i+1})$ are two successive sampling proportions. P_i is defined as the Sampling Gelling Point of the SVM classifier $f(W;X)$ if the classification accuracy of $f(W;X)$ does not increase remarkably with the increasing of the sampling proportion ΔP $(\Delta P = P_{i+1} - P_i)$. Specifically, given (P_i, P_{i+1}) and the corresponding classification accuracies $(A_{P_i}, A_{P_{i+1}})$, P_i is the Sampling Gelling Point if the following inequality is satisfied: $\frac{A_{P_{i+1}} - A_{P_i}}{P_{i+1} - P_i} < \alpha$, where α is a const predefined.

To illustrate the Sampling Gelling Point, two large datasets, *IJCNN1* and *USPS*, and eleven toy UCI datasets are used in our experiment. Figure 2 shows the Gelling Point phenomenon over these datasets. Figure 2 demonstrates that all the datasets used in our experiment have the Gelling Point phenomenon. Based on above observation, it is possible to train an approximate SVM classifier with a small subset of the initial training dataset.

Due to the following advantages, we propose to sample the training data at the Sampling Gelling Point. First, generalization performance of a weak SVM at the sampling Gelling Point is close to that of the regular SVM. Second, if the sampling percentage is too high, most data are correctly classified and cleaned, which may leads to useful information loss. Contrarily, a lower sampling percentage may lead to inadequately data cleaning and some useless data remain. Sampling Gelling Point is a tradeoff between sampling size and useful information.

However, it is an nontrivial task to exactly find the Sampling Gelling Point, because the Sampling Gelling Point depends largely on the specific dataset and the predefined parameter α. Here we propose a heuristic method to approximately find the Sampling Gelling Point. At each step of our method, we sample the training data at a sampling proportion P_{i+1} and use the sampled data to train a weak classifier. If the ratio between the classification accuracy increment $A_{P_{i+1}} - A_{P_i}$ of two successive weak SVMs and the corresponding sampling proportion increment $P_{i+1} - P_i$ is less than a predefined threshold α, stop and P_i is considered as the approximate Sampling Gelling Point. Otherwise, increase the sampling proportion to $P_{i+2} = P_{i+1} + \beta$ (β is a predefined step length) in the next step. Meanwhile, we set the maximum sampling proportion at 10 % no matter the Sampling Gelling Point is found or not. Figure 3 illustrates the key idea of our method.

4 Informative pattern extraction based on maximum information entropy

Learning theory defines informative patterns as: *given a model trained on a sequence of patterns, a new pattern is informative if it is difficult to predict by a model trained on previously seen data* [22]. Guyon et al. and MacKay have proposed that informative pattern can be indicated by the level of surprise, and the level of surprise varies in the opposite direction as the probability of guessing the correct label

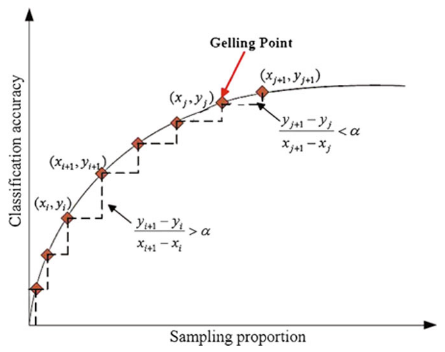


Fig. 3 A illustration showing how to approximately find the Sampling Gelling Point

[22, 23]. In this paper, we define the informative pattern as: *a pattern is informative if it is difficult to be predicted by all the ε-Gross-granularity weak SVMs*. In other words, if the number of ε-Gross-granularity weak SVMs predicting a pattern label as -1 is close to that predicting its label as 1, we believe that this pattern is informative. On the contrary, a pattern is considered to be not informative if its label is predicted as -1 or 1 by all the ε-Gross-granularity weak SVMs.

To extract more informative patterns, we propose two maximum information entropy algorithms in this section. Before introducing the two algorithms, we first give some definitions.

Assume there are n training samples $X = \{x_1, \ldots, x_n\}$ which can be categorized into m classes $C = \{c_1, \ldots, c_m\}$. We consider the probability of the sample x_i being classified to class c_j by the ε-Gross-granularity weak SVMs as a random variable. Then the information entropy of sample x_i can be defined as

$$\begin{aligned} H(x_i) &= E\big(I(x_i)\big) \\ &= -\sum_{j=1}^{m} p(c_{x_i} = c_j) \cdot \log_2 p(c_{x_i} = c_j) \end{aligned} \tag{7}$$

$I(x_i)$ is the amount of information of x_i, and $p(\cdot)$ is the probability mass function. Assume there are N ε-Gross-granularity weak SVMs predicting each training data. After prediction, we use k_i $(i = 1, \ldots, n)$ to denote the number of predictors which misclassify the training sample x_i. k_i can be formally defined as

$$k_i = \sum_{t=1}^{N} y_i^t \bigoplus Label_i \tag{8}$$

where y_i^t is the class label of x_i predicted by the t_{th} ε-Gross-granularity weak SVM, and $Label_i$ is x_i's true class label. Then the number of weak SVMs correctly predicting sample x_i's label is $N - k_i$, $i = 1, \ldots, n$. Therefore, the empirical probabilities of correctly and incorrectly predicting the label of sample x_i can be represented as

$$\begin{aligned} p(x_i\_correct) &\approx p(y_i = Label_i) = \frac{N-k_i}{N} \\ p(x_i\_wrong) &\approx p(y_i \neq Label_i) = \frac{k_i}{N} \end{aligned} \tag{9}$$

Similarly, the amount of information of sample x_i can be approximately represented as

$$\begin{aligned} I(x_i\_correct) &\approx -\log_2(y_i = Label_i) \\ &= -\log_2\left(\frac{N-k_i}{N}\right) \\ I(x_i\_wrong) &\approx -\log_2(y_i \neq Label_i) \\ &= -\log_2\left(\frac{k_i}{N}\right) \end{aligned} \tag{10}$$

Finally, the information entropy of sample x_i can be represented by Eq. (11) according to Eqs. (9) and (10).

$$\begin{aligned} H(x_i) &= \sum p(x_i) \cdot I(x_i) \\ &= -\sum p(x_i) \cdot \log_2 p(x_i) \\ &\approx -\left(\left(\frac{N-k_i}{N}\right) \cdot \log_2\left(\frac{N-k_i}{N}\right) \right. \\ &\quad \left. + \left(\frac{k_i}{N}\right) \cdot \log_2\left(\frac{k_i}{N}\right)\right) \end{aligned} \tag{11}$$

Now we obtain the information entropy of each sample, then the information entropy of the whole dataset can be obtained by adding up the information entropy of all the data.

$$H(X) = \sum_{i=1}^{n} H(x_i) \tag{12}$$

4.1 Informative pattern extraction algorithm MEFS

The intuitive idea of informative pattern extraction algorithm MEFS (Maximize the Entropy by Fixing the data Size) is to select a subset $\widetilde{X}$ with a fixed size from the initial dataset X', and the information entropy of $\widetilde{X}$ is maximum. More specifically, given the training dataset $X' = \{x_1, \ldots, x_{n'}\}$ and the sampling proportion q %, the problem is to find the subset $\widetilde{X}$ $(|\widetilde{X}| = n' \cdot q\ \%)$ with the maximum information entropy. We formally define the problem as follows:

$$\begin{aligned} &\widetilde{X} = \{x_{t1}, \ldots, x_{t\widetilde{n}}\} = \arg\max\big(H(\widetilde{X})\big) \\ &\quad \text{Subject to:} \quad n' \cdot q\ \% \leq \widetilde{n} \end{aligned} \tag{13}$$

According to above deductions, this optimization problem can be further represented as

$$\widetilde{X} \approx \arg\max\left(\sum_{i=1}^{\widetilde{n}}\left[-\left(\frac{N-k_{x_{ti}}}{N}\right)\cdot\log_2\left(\frac{N-k_{x_{ti}}}{N}\right) + \frac{k_{x_{ti}}}{N}\cdot\log_2\left(\frac{k_{x_{ti}}}{N}\right)\right]\right) \tag{14}$$
$$\text{Subject to:}\quad n'\cdot q\ \% \le \widetilde{n}$$

Pseudo code of algorithm MEFS is as follows:

```
Procedure Informative_Pattern_MEFS($X'$,$q$ %,$\widetilde{X}$,$N$,$K$)
Input: the dataset from the first stage $X'$, sampling percentage $q$ %, number of $\varepsilon$-Gross-granularity weak SVMs $N$ (Here we set $N = 10$), misclassification times $K$ (vector) of training data $X'$.
Output: subset $\widetilde{X}$ of $X'$ ($|\widetilde{X}| = |X'| \cdot q$ %) with the maximum information entropy
1. Calculate the information entropy of each training data.
   $H(x_i) \approx -\left(\left(\frac{N-k_i}{N}\right)\cdot\log_2\left(\frac{N-k_i}{N}\right) + \left(\frac{k_i}{N}\right)\cdot\log_2\left(\frac{k_i}{N}\right)\right)$
2. Sort the training data in descending order in terms of their information entropy.
   $DesSort(X') = \{x_{s1}, x_{s2}, x_{s3}, \dots, x_{sn}\}$
3. Calculate the information entropy of the whole dataset.
   $H(X') = \sum_{i=1}^{n} H(x_i)$
4. Select the subset $\widetilde{X}$ from $DesSort(X')$ with the maximum information entropy.
   $\widetilde{X} = \{x_{s1}, \dots, x_{s\widetilde{n}}\} = \arg\max(H(\widetilde{X}))$
      Subject to: $n' \cdot q\ \% \le \widetilde{n}$
return $\widetilde{X}$
end Procedure
```

4.2 Informative pattern extraction algorithm MSFE

The basic idea of algorithm MSFE (Minimize data Size by Fixing the Entropy) is quite similar to algorithm MEFS. MSFE aims to maximize the information entropy under the condition of fixing the size of extracted samples, while MSFE tries to select a minimal subset of the given samples under the condition of fixing the information entropy.

The intuitive idea of algorithm MSFE is to minimize the size of extracted samples by fixing the information entropy. For example, how many samples do we need at least to obtain 95 % information entropy of all the training samples?

More specifically, given the training dataset X' and the information entropy percentage m %, the task is to find the minimal subset $\widetilde{X}$ whose information entropy is no less than $m\ \% \times H(X')$. This problem can be formally represented as follows:

$$\widetilde{X} = \{x_{t1}, \dots, x_{t\widetilde{m}}\} = \arg\min(Card(\widetilde{X})) \qquad \text{Subject to:}\quad H(\widetilde{X}) \ge m\ \% \cdot H(X') \tag{15}$$

$Card(\widetilde{X})$ is the cardinality of subset $\widetilde{X}$.

Pseudo code of algorithm MSFE is as follows:

```
Procedure Informative_pattern_MSFE($X'$, $m$ %, $\widetilde{X}$, $N$, $K$)
Input: the dataset $X'$ from the first stage, entropy percentage $m$ %, number of $\varepsilon$-Gross-granularity weak SVMs N, misclassification times $K$ of each training data.
Output: the minimal subset $\widetilde{X}$ of $X'$ whose information entropy is no less than $m$ % of the total entropy
1. Calculate the information entropy of each training data.
   $H(x_i) \approx -\left(\left(\frac{N-k_i}{N}\right)\cdot\log_2\left(\frac{N-k_i}{N}\right) + \left(\frac{k_i}{N}\right)\cdot\log_2\left(\frac{k_i}{N}\right)\right)$
2. Sort training data in descending order in terms of their information entropy.
   $DesSort(X') = \{x_{s1}, x_{s2}, x_{s3}, \dots, x_{sn}\}$
3. Calculate the information entropy of the whole dataset.
   $H(X') = \sum_{i=1}^{n} H(x_i)$
4. Select the minimum subset $\widetilde{X}$ from $DesSort(X')$ with the entropy no less than $m\ \% \cdot H(X')$.
   $\widetilde{X} = \{x_{s1}, \dots, x_{s\widetilde{m}}\} = \arg\min(Card(\widetilde{X}))$
      Subject to: $H(\widetilde{X}) \ge m\ \% \cdot H(X')$
return $\widetilde{X}$
end Procedure
```

5 Experiment and evaluation

In this section, we empirically evaluate our approaches by conducting extensive experiments on four large datasets.

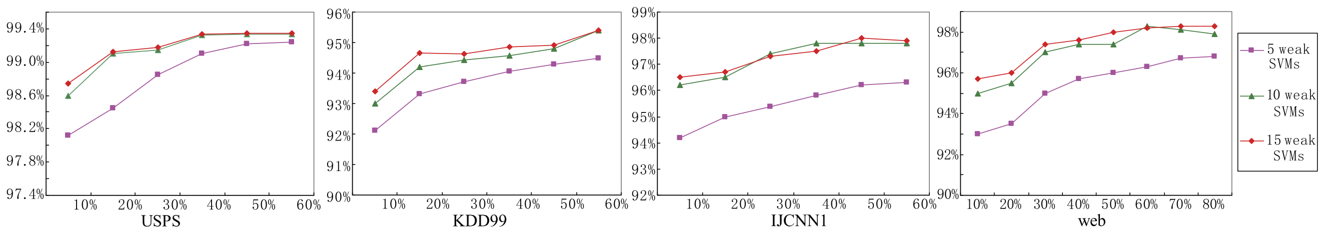


Fig. 4 Testing accuracy with different # of weak SVMs

The objectives of these experiments are three-fold. (1) To evaluate the generalization performance of our approaches. Two metrics are used here: accuracy value and AUC (area under curve) value. (2) To evaluate the speedup obtained by our approaches. (3) To compare our approaches with state-of-the-art fast SVM algorithms, including PEGASOS, RSVM and LIBLINEAR.

LIBSVM is used to train weak classifiers and the final classifier in all our experiments for its perfect generalization performance [24]. We report the average result on 10 runs. The experiments are executed in parallel on 5 WindowsXP machines with four 2.2 GHz processor and 4 GB DRAM.

5.1 Datasets

Four large datasets are used in our experiment, and we briefly describe them as follows.

KDD99 The KDDCUP-99 intrusion detection dataset has been used for the Third International Knowledge Discovery and Data Mining Tools Competition. The training set contains 4,898,431 connection records, which are processed from seven weeks of binary TCP dump data of network traffic. The test set contains 311,029 patterns which are from another two weeks of network connection.

Extended USPS USPS dataset is the US Postal Service handwritten digits recognition corpus. It contains the normalized gray scale images of size 16 × 16, divided into a training set of 7291 images and a test set of 2007 images. In this experiment, we use the extended version of the USPS dataset. The extended dataset has a total of 266,079 patterns while the extended test set has 753,83 patterns.

IJCNN1 The IJCNN (task 1) dataset is from the IJCNN 2001 challenge [25]. It contains 49,990 training patterns and 91,701 testing patterns. Each pattern has 22 features.

Web Web page dataset consists of 49,749 web pages as training dataset and 14,951 web pages as testing dataset, with 300 sparse binary keyword attributes extracted from each web page.

Table 1 Experiment datasets

Dataset	KDD99	USPS	IJCNN1	Web
Attribute	127	676	22	300
Training Data	4,898,431	266,079	49,990	49,749
Testing Data	311,029	75,383	91,701	14,951
Gelling Point	1 %	2 %	8 %	1 %

Table 1 shows some key information of the four datasets. Currently, our approach is only applied to the binary classification problem. For the KDD99 intrusion detection dataset, the normal connections are considered as the positive samples and all the abnormal connections are considered to be the negative samples.

5.2 Preliminary Experiment 1: effect of the size of weak SVMs

To apply the proposed approach, firstly we need to determine how many weak SVMs should be used in the data cleaning stage. Theoretically, more weak SVMs can lead to a more precise estimation of the misclassification probability distribution of the training data. As a result, the performance of final SVM classifier can be better. However, more weak SVMs need more training time and computing resources. Therefore, tradeoff between the computing resources and the generalization performance should be carefully considered. To study the effect of the number of weak SVMs to the testing accuracy, we conduct a preliminary experiment. In the experiment, we use 5, 10 and 15 weak SVMs on the four datasets, respectively. For each dataset, we record the average testing accuracy with a different number of weak SVMs and different information entropy.

Figure 4 shows the experiment result. (For the convenience of visualization, only the result of algorithm MEFS is presented. The result of MSFE is very similar to the result of MEFS.) The result demonstrates that the testing accuracy of the final classifiers with 10 and 15 weak SVMs is obviously superior to that only with 5 weak SVMs. Nevertheless, the results of using 10 and 15 weak SVMs are very similar, which implies the performance of final classifier keeps stable when the number of weak SVMs is large enough. Based

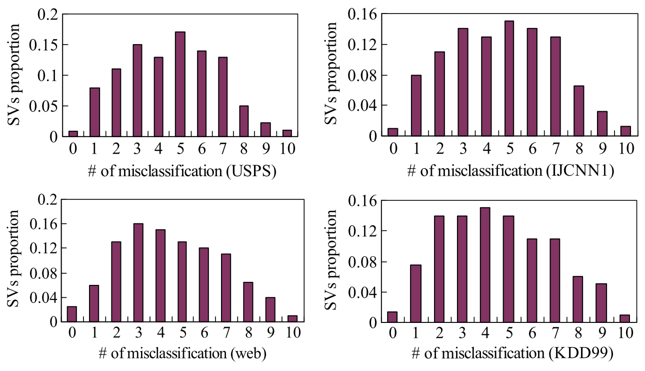


Fig. 5 SVs distribution

on above observation, to save computational resources, we only use 10 weak SVMs in the following experiments.

5.3 Preliminary Experiment 2: whether, and how many support vectors are also cleaned?

The SVM classifier only depends on a few samples called support vectors (SVs). Therefore, the SVs and their corresponding coefficients largely determine the classification performance. As most of the training samples are simply removed in the data cleaning stage, an interesting problem is: are there any SVs that are also cleaned? To investigate the relationship between the cleaned data and SVs, we conduct another preliminary experiment in this subsection.

The entire dataset is divided into three parts after the first stage of our approach: the samples that are correctly classified by all the weak SVMs, the samples that are misclassified by all the weak SVMs, and the others. In Fig. 5, we show the distribution of misclassification times of SVs over the four datasets. Some interesting conclusions can be drawn from Fig. 5. First, only a small proportion (around 5 %) of SVs are cleaned in the first stage, because only the samples which are misclassified or correctly classified by all the weak SVMs are removed. Samples misclassified by all the weak SVMs are mostly SVs. This is mainly because samples which are misclassified by all the weak SVMs are very unlikely to be correctly classified by a regular SVM. Interestingly, the subsequent experiment will show that simply removing these data will not significantly affect the classification performance, since some outliers are also cleaned and the classifier boundary becomes more smooth. Second, few SVs are correctly classified by all the weak SVMs. Therefore, only a small proportion (2–3 %) of SVs are in the samples which are correctly classified by all the weak SVMs. Quite naturally, if a sample is correctly classified by a weak SVM, it is not a tough task for a regular SVM to classify it correctly. As only a small proportion of SVs are cleaned, most SVs remained to the second stage as the training data.

5.4 Preliminary Experiment 3: effect of mixing kernels

In this subsection, we conduct an experiment to study the effect of mixing kernels to the classification performance. SVM is a typical kernel method which maps the training data (nonlinearly) into a higher dimensional feature space (kernel space) when the original space is linear inseparable. For mapping the original space to a higher dimensional space, many kernel functions are constructed. Generally speaking, the kernel functions used in SVM can be mainly divided into two categories: *local kernels* and *global kernels* [26].

Local kernels Only the training data that are close or in the proximity of each other have an influence on the kernel values. Examples of local kernels include Gaussian kernel, Radial Basis Function kernel (RBF) and KMOD kernel.

- *Gaussian kernel*:

$$K(x, x_i) = \exp\left(-\frac{1}{2}(x - x_i)A^{-1}(x - x_i)^T\right) \tag{16}$$

where A have three following norms:

$A = I$	Euclidean Norm
$A = D_j^{-1}$	Diagonal Norm
$A = C_j^{-1}$	Mahalonobis Norm

- *Radial Basis Function Kernel (RBF)*:

$$K(x, x_i) = \exp - \frac{\|x - x_i\|^2}{2\sigma^2} \tag{17}$$

where σ is the kernel parameter.

- *KMOD kernel*:

$$K(x, x_i) = \exp\left(\frac{1}{1 + \|x - x_i\|^2}\right)^{-1} \tag{18}$$

Global kernels Data that are far away from each other still have an influence on the kernel values. Typical global kernels include Linear kernel, Polynomial kernel and Sigmoid kernel.

- *Linear kernel*:

$$K(x, x_i) = x \cdot x_i \tag{19}$$

- *Polynomial kernel*:

$$K(x \cdot x_i + 1)^p \tag{20}$$

where the kernel parameter p is the degree of polynomial to be used.

- *Sigmoid kernel*:

$$K(x, x_i) = \tanh(x \cdot x_i + 1) \tag{21}$$

Table 2 Classification accuracy while using different mixture kernels

# of weak SVMs	Kernel	KDD99	USPS	IJCNN1	Web
10	Gaussian	0.934 ± 0.003	0.982 ± 0.004	0.954 ± 0.002	0.956 ± 0.005
10	RBF	0.942 ± 0.004	0.985 ± 0.003	0.967 ± 0.003	0.960 ± 0.005
10	Linear	0.937 ± 0.003	0.975 ± 0.004	0.967 ± 0.003	0.960 ± 0.005
10	Polynomial	0.947 ± 0.004	0.988 ± 0.005	0.965 ± 0.004	0.96 ± 0.004
10	Sigmoid	0.945 ± 0.003	0.99 ± 0.004	0.97 ± 0.003	0.974 ± 0.004
10	Gaussian (5) + RBF (5)	0.957 ± 0.002	0.996 ± 0.003	0.978 ± 0.003	0.978 ± 0.004
10	Gaussian (5) + Linear (5)	0.954 ± 0.003	0.99 ± 0.003	0.98 ± 0.005	0.98 ± 0.003
10	Gaussian (5) + Polynomial (5)	0.948 ± 0.004	0.992 ± 0.004	0.983 ± 0.005	0.984 ± 0.004
10	Gaussian (5) + Sigmoid (5)	0.955 ± 0.005	0.994 ± 0.004	0.976 ± 0.002	0.979 ± 0.003
10	RBF (5) + Linear (5)	0.948 ± 0.003	0.989 ± 0.002	0.982 ± 0.003	0.98 ± 0.003
10	RBF (5) + Polynomial (5)	0.954 ± 0.004	0.995 ± 0.003	0.98 ± 0.003	0.983 ± 0.004
10	RBF (5) + Sigmoid (5)	0.952 ± 0.003	0.993 ± 0.004	0.978 ± 0.003	0.983 ± 0.003
10	Linear (5) + Polynomial (5)	0.95 ± 0.002	0.995 ± 0.004	0.983 ± 0.003	0.98 ± 0.004
10	Linear (5) + Sigmoid (5)	0.945 ± 0.003	0.985 ± 0.003	0.982 ± 0.003	0.978 ± 0.005
10	Polynomial (5) + Sigmoid (5)	0.952 ± 0.004	0.99 ± 0.003	0.982 ± 0.004	0.982 ± 0.003
	Average of single kernel	0.941 ± 0.003	0.984 ± 0.004	0.967 ± 0.003	0.962 ± 0.004
	Average of mixture kernels	$\mathbf{0.952 \pm 0.003}$	$\mathbf{0.992 \pm 0.003}$	$\mathbf{0.980 \pm 0.003}$	$\mathbf{0.981 \pm 0.003}$

Depending on particular applications and specific datasets, the performance of SVM classifiers trained with different kernels may differ significantly. A desirable characteristic for learning may not be a desirable characteristic for generalization. Smits and Jordan have reported that using mixtures of kernels can result in having both good interpolation and extrapolation abilities [26]. The regression performance can be improved by combing two or more kernels. Similarly, Kumar et al. have studied the effect of different mixed kernels on overall sub-pixel classification accuracy of remote sensing data using Fuzzy Error Matrix (FERM) [27]. Their experimental results showed that overall sub-pixel classification accuracy of multi-spectral remote sassing data varied while using a different mixture of kernel functions in SVM. In addition, Collobert et al. presented a new mixture of SVM kernels that can be easily implemented in parallel, and each SVM is trained on a small subset of the whole dataset [29].

There are several methods of mixing kernels, of which linear mixture is the most common form. It can be defined as follows:

$$K(x, x_i) = \theta K_a(x, x_i) + (1-\theta) K_b(x, x_i) \tag{22}$$

A bulk of weak SVMs are used in the first stage of our approach. Therefore, we propose to use different kernels for different weak SVMs. In our method, a single weak SVM corresponds to a single kernel and different weak SVMs correspond to different kernels. The mixture of kernels here is defined as follows:

$$\begin{aligned} &K(x, x_i) = \sum_{t=1}^{N} \alpha_t \cdot K_j(x, x_i) \\ &\text{Subject to:} \quad \alpha_t = 0 \quad \text{when } j \neq t \\ &\qquad\qquad\qquad \alpha_t = 1 \quad \text{when } j = t \end{aligned} \tag{23}$$

The final mixing kernels classifier can be defined according to the kernels used in weak SVMs as follows:

$$K(x, x_i) = \sum_{j=1}^{J} \sum_{t=1}^{N} (\alpha_t / N) \cdot K_j(x, x_i) \tag{24}$$

where j denotes the jth weak classifier.

To test whether mixing kernels will affect the classification performance, some preliminary experiments are performed. For simplicity, only 2 different kernels are mixed in our experiment. We then record the classification accuracy by various single kernels and mixed kernels. The experimental results are shown in Table 2. Table 2 demonstrates that mixing kernels sightly improves the classification performance compared with single kernels on the four datasets. Almost all the mixing kernels methods outperform single kernel methods regardless of what kernels we use. The last two lines of the table show the average results of the two methods. It also shows that the average result obtained by using mixing kernels is better than that obtained by only using single kernels. Therefore, in the following experiments, we use mixing kernels, and for simplicity, only two kernels are mixed.

5.5 Preliminary Experiment 4: effect of sampling proportion P

As another important parameter, the sampling proportion P is also needed to be pre-determined. Therefore, in this subsection, we conduct an experiment to study the effect of sampling proportion P to the classification performance and training time. In Sect. 3.2, we propose to sample the training dataset at Sampling Gelling Point. Here we empirically show that Sampling Gelling Point is a rather reasonable sampling point due to its desirable classification performance and remarkable reduction of training time complexity.

Table 3 shows our experimental results on the four datasets. For each dataset, we sample training data at several different sampling points around the Sampling Gelling Point. Then for each sampling point, we utilize the bootstrap sampling based approach to clean useless data and extract informative data on various entropy proportions. Finally, we report the number of misclassified data N_{mis}, the number of extracted informative data n on each entropy proportion, the training time t (in seconds) and the testing accuracy (acc) for each experiment. Here, the training time t is based on the entire process, including the bootstrap sampling and data cleaning process in the first stage and the informative pattern extraction and final classifier training process in the second stage. But the entire process is based on all the parameters being confirmed, such as sampling percentage P and number of weak SVMs N in the first stage and information percentage m % in the second stage. Therefore, the time of tuning parameters is excluded from t. Additionally, as the first stage is processed in parallel on several computers, we only consider the time consumed on one computer.

Table 3 shows that the number of misclassified data N_{mis} drops with the rising of sampling proportion P. It means more training data are cleaned and less training time is needed for a large P. However, if the sampling proportion is too large, some informative data may also be cleaned due to overfitting, and testing accuracy will be decrease consequently. Take IJCNN1 dataset for example, the testing accuracy at sampling proportion 1 %, 5 % and 8 % are similar and stable, all between 0.97 and 0.98. When we sample the data at the proportion of 10 %, the testing accuracy drops remarkably to around 0.95. What's worse, the testing accuracy becomes more and more unstable with the increasing of P.

The table data with black body represent the results at Sampling Gelling Point. It also demonstrates that Sampling Gelling Point is a reasonable sampling point. This is because testing accuracy at Sampling Gelling Point is comparable to that at smaller sampling points, and most useless training data are cleaned.

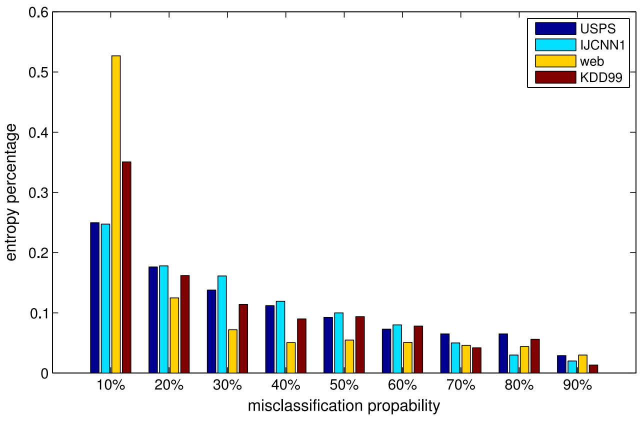


Fig. 6 Entropy distribution

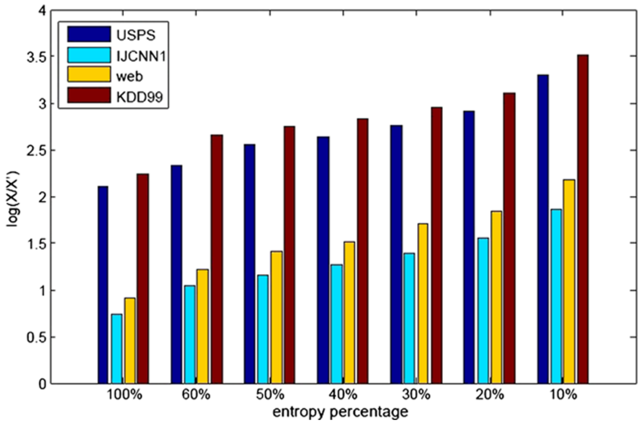


Fig. 7 Variation of training data reduction under various entropy proportions

5.6 Speedup and generalization performance experiments

To evaluate the speedup of the training process and generalization performance of our algorithms, we compare them with regular LIBSVM. Similar to Sect. 5.5, the training time of our approaches also consists of two parts, the time consumed on data cleaning stage and the time consumed on the informative patterns extraction stage. The two parts are added up as the whole training time in this experiment and the subsequent experiment in Sect. 5.7. Before comparison, we first investigate the entropy distribution of training data on the four datasets. Figure 6 demonstrates the entropy proportions under different misclassification probabilities over the four datasets. From Fig. 6 we can see that the entropy proportion decreases with the increasing of misclassification probability. It means that most data can be classified correctly and only a small part of the data are likely to be misclassified. This result is also consistent with the Gaussian distribution hypothesis of training datasets. Figure 7 shows the variation of training data reduction under various entropy proportions. For the convenience of the demonstration, we utilize the logarithmic form. The x axis is $\log(X/X')$,

Table 3 Experiment on four datasets with different sampling proportion P. N_{mis} denotes the number of misclassified training data. Three metrics are reported, the number of extracted informative data n, training time t (s) and testing accuracy *acc*

Datasets	P	N_{mis}	Metrics	Entropy proportion					
				10 %	20 %	30 %	40 %	50 %	60 %
USPS	0.5 %	4780 ± 245	n	325 ± 24	895 ± 45	1453 ± 57	2400 ± 112	3025 ± 178	3542 ± 210
			t (s)	3.8 ± 1.0	8.5 ± 1.2	12.5 ± 1.8	16.8 ± 2.4	25.4 ± 2.7	30.5 ± 3.4
			acc	.974 ± .003	.978 ± .002	.983 ± .003	.988 ± .003	.992 ± .004	.994 ± .002
	1 %	3895 ± 224	n	286 ± 14	625 ± 20	986 ± 24	1562 ± 45	2045 ± 54	2564 ± 78
			t (s)	3.2 ± 0.4	6.8 ± 0.8	10.5 ± 1.6	14.5 ± 2.1	22.3 ± 2.6	25.4 ± 3.4
			acc	.965 ± .004	.972 ± .003	0.978 ± .003	.982 ± .004	.990 ± .002	.992 ± .001
	2 %	**2675 ± 165**	n	**187 ± 14**	**402 ± 22**	**687 ± 35**	**1254 ± 65**	**1686 ± 87**	**2052 ± 104**
			t (s)	**2.8 ± 0.5**	**6.2 ± 1.1**	**9.5 ± 1.7**	**12.7 ± 2.2**	**15.6 ± 2.4**	**21.8 ± 3.1**
			acc	**.960 ± .010**	**.975 ± .006**	**.978 ± .004**	**.980 ± .003**	**.985 ± .004**	**.992 ± .001**
	5 %	1020 ± 91	n	87 ± 12	165 ± 25	257 ± 45	356 ± 46	456 ± 54	754 ± 68
			t (s)	1.5 ± 0.3	2.7 ± 0.7	3.1 ± 0.6	4.2 ± 1.0	6.7 ± 0.8	10.3 ± 1.1
			acc	.915 ± .012	.945 ± .010	.956 ± .008	.965 ± .007	.972 ± .004	.975 ± .003
IJCNN1	1 %	2121 ± 178	n	187 ± 14	368 ± 21	586 ± 43	754 ± 59	1042 ± 101	1245 ± 114
			t (s)	4.2 ± 0.6	12.5 ± 1.1	18.7 ± 1.3	23.4 ± 2.1	32.4 ± 2.6	43.8 ± 3.2
			acc	.940 ± .008	.946 ± .006	.949 ± .002	.965 ± .003	.978 ± .002	.981 ± .002
	5 %	1745 ± 102	n	164 ± 22	315 ± 34	502 ± 54	654 ± 56	875 ± 68	1025 ± 97
			t (s)	3.7 ± 0.4	10.5 ± 1.1	17.4 ± 1.4	22.4 ± 3.2	27.3 ± 2.2	35.6 ± 2.4
			acc	.926 ± .006	.946 ± .004	.948 ± .002	.958 ± .003	.972 ± .002	.976 ± .002
	8 %	**1324 ± 112**	n	**97 ± 12**	**203 ± 21**	**354 ± 28**	**453 ± 34**	**587 ± 48**	**685 ± 68**
			t (s)	**2.8 ± 0.6**	**4.6 ± 0.8**	**12.3 ± 1.4**	**16.5 ± 1.5**	**20.8 ± 2.0**	**24.8 ± 2.3**
			acc	**.920 ± .015**	**.934 ± .010**	**.945 ± .006**	**.957 ± .008**	**.968 ± .004**	**.972 ± .002**
	10 %	1256 ± 103	n	78 ± 8	168 ± 12	245 ± 22	385 ± 28	454 ± 32	585 ± 48
			t (s)	2.5 ± 0.4	3.8 ± 1.0	8.6 ± 1.3	12.6 ± 2.1	16.7 ± 2.3	21.2 ± 3.5
			acc	.897 ± .021	.916 ± .020	.927 ± .012	.935 ± .010	.947 ± .010	.948 ± .008
Web	0.5 %	2875 ± 212	n	189 ± 12	356 ± 23	568 ± 35	758 ± 57	1542 ± 87	1865 ± 112
			t (s)	2.8 ± 0.8	14.2 ± 1.4	30.5 ± 2.5	44.7 ± 3.7	64.2 ± 4.6	87.5 ± 7.2
			acc	.878 ± .043	.923 ± .014	.954 ± .012	.970 ± .008	.975 ± .005	.980 ± .003
	1 %	**2056 ± 165**	n	**154 ± 14**	**302 ± 23**	**424 ± 34**	**561 ± 45**	**785 ± 65**	**1243 ± 103**
			t (s)	**2.4 ± 0.4**	**6.7 ± 0.8**	**18.4 ± 1.4**	**28.6 ± 2.7**	**46.8 ± 5.7**	**55.6 ± 8.6**
			acc	**.856 ± .025**	**.918 ± .015**	**.932 ± .008**	**.957 ± .006**	**.972 ± .006**	**.975 ± .004**
	2 %	1856 ± 154	n	132 ± 11	254 ± 18	387 ± 24	542 ± 38	658 ± 45	876 ± 83
			t (s)	2.2 ± 0.4	4.8 ± 0.8	15.3 ± 1.4	24.6 ± 3.5	36.7 ± 4.2	50.4 ± 4.6
			acc	.823 ± 0.024	.854 ± .021	.902 ± .014	.945 ± .011	.956 ± .008	.965 ± .004
	5 %	1500 ± 112	n	112 ± 11	245 ± 18	354 ± 22	523 ± 35	587 ± 56	724 ± 68
			t (s)	2.1 ± 0.4	4.6 ± 0.8	13.7 ± 1.6	20.6 ± 2.7	28.8 ± 3.8	40.5 ± 4.3
			acc	.810 ± .023	.845 ± .020	.878 ± .017	.910 ± .008	.926 ± .006	.945 ± .003
KDD99	0.5 %	4678 ± 324	n	324 ± 26	627 ± 45	1067 ± 87	1534 ± 103	1786 ± 142	2345 ± 234
			t (s)	20.1 ± 1.8	43.2 ± 4.5	75.5 ± 8.6	104.7 ± 10.5	164.2 ± 12.7	202.4 ± 20.4
			acc	.930 ± .008	.934 ± .006	.938 ± .004	.942 ± .003	.945 ± .003	.946 ± .002
	1 %	**3876 ± 245**	n	**289 ± 24**	**568 ± 48**	**878 ± 67**	**1257 ± 114**	**1658 ± 124**	**1743 ± 153**
			t (s)	**18.7 ± 1.6**	**34.7 ± 3.5**	**65.4 ± 4.6**	**91.6 ± 8.4**	**112.8 ± 11.3**	**168.6 ± 13.4**
			acc	**.924 ± .006**	**.927 ± .005**	**.934 ± .004**	**.936 ± .005**	**.940 ± .002**	**.942 ± .003**
	2 %	1856 ± 126	n	156 ± 12	284 ± 22	357 ± 32	582 ± 45	758 ± 65	1122 ± 120
			t (s)	12.3 ± 1.1	18.8 ± 1.6	25.3 ± 2.3	40.6 ± 3.6	54.7 ± 5.4	87.4 ± 7.7
			acc	.845 ± .021	.865 ± .017	.889 ± .012	.924 ± .008	.932 ± .006	.935 ± .004
	5 %	1045 ± 97	n	87 ± 10	156 ± 12	224 ± 21	315 ± 31	458 ± 43	546 ± 47
			t (s)	8.6 ± 1.1	12.6 ± 2.1	15.7 ± 2.2	24.6 ± 3.2	28.7 ± 3.4	38.5 ± 4.2
			acc	.820 ± .023	.835 ± .021	.870 ± .018	.875 ± .013	.889 ± .012	.915 ± .010

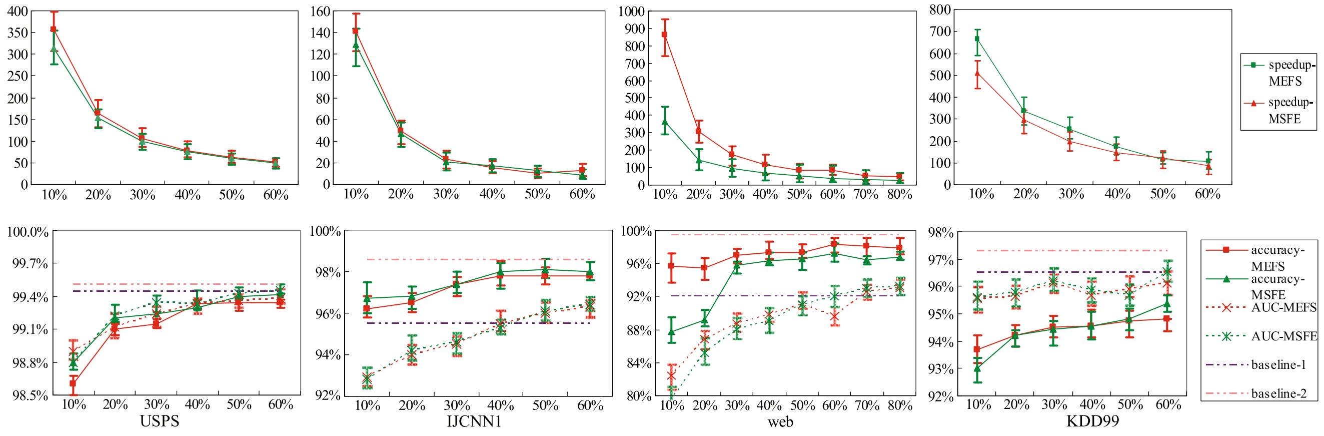


Fig. 8 Speedup, test accuracy & AUC under various entropy proportions

where X is the size of the initial dataset and X' represents the size of the reduced dataset. We then evaluate the speedup of training time, test accuracy and AUC of our approaches under various entropy proportions. As a comparison, the test accuracy and AUC of regular LIBSVM are used as the baseline. Figure 8 shows the experimental results. The upper figures show the speedup and the lower figures show the accuracy and AUC under different entropy proportions. (For the convenience of the comparison, we also calculate the entropy proportion according to the misclassified samples selected by algorithm MEFS.) Baseline-1 and baseline-2 represent the test accuracy and AUC of regular LIBSVM, respectively.

As shown in Fig. 8, the improvement on speedup of training time is significant on all four datasets. For the datasets KDD99, web and USPS, the speedup achieved by our approach can be up to 100–200, while the testing accuracy and AUC are very close to that obtained by the regular LIBSVM. Although for the IJCNN1 dataset, the speedup is not that significant, both algorithms are nearly 20 times faster than regular LIBSVM. Experimental results on IJCNN1 and web datasets also demonstrate that our methods may perform better than regular LIBSVM when the entropy proportion is high enough.

It is also shown that the performance of SVM trained on the data with only about 50–60 % information entropy is comparable to that trained on the whole dataset. This is probably because in many cases only a small quantity of dominant training data determine the performance of the SVM classifier to a great extent. As our methods effectively extract these informative patterns, the training time is significantly reduced with little performance loss.

5.7 Comparative experiments with the state-of-the-art methods

To further compare our approaches with existing fast SVM training methods, we conduct the comparative experiment against three state-of-the-art algorithms, PEGASOS, RSVM and LIBLINEAR SVM.

PEGASOS and LIBLINEAR are two effective SVM speedup algorithms. Both algorithms have shown excellent performance on very large datasets, especially datasets with very sparse, linear kernels. RSVM aims to generate a nonlinear kernel-based separating surface that requires a subset of a large dataset for its explicit evaluation. The basic idea of RSVM is using only a small random sample of training data as a representative sample with respect to the entire dataset, which is very similar to our method. Besides the three methods, LIBSVM with random sampling is also used as a baseline to show the effectiveness of our sampling method. Figure 9 shows the experimental results. The following conclusions can be drawn from the results. First, from the training time perspective, there is no significant difference between our approaches and the three state-of-the-art methods. For web and USPS datasets, our approaches need less time to converge than PEGASOS and LIBLINEAR. Second, the testing accuracy of our approaches outperform other algorithms. For web, IJCNN1 and USPS datasets, PEGASOS and LIBLINEAR are even inferior to LIBSVM with random sampling. It was reported that both PEGASOS and LIBLINEAR are linear SVM [15, 16]. Therefore, the two algorithms are effective for large, sparse datasets like text classification with linear kernels. However, our experiment indicates that PEGASOS and LIBLINEAR might not be so effective for large yet not sparse datasets. Comparison between our approaches and RSVM demonstrates that our methods outperform RSVM in test accuracy. Meanwhile, our methods need less training time. This shows that our approaches are more effective to extract informative samples than RSVM.

6 Conclusion

To make SVM practical to large datasets, an algorithm to clean useless training data and extract informative patterns

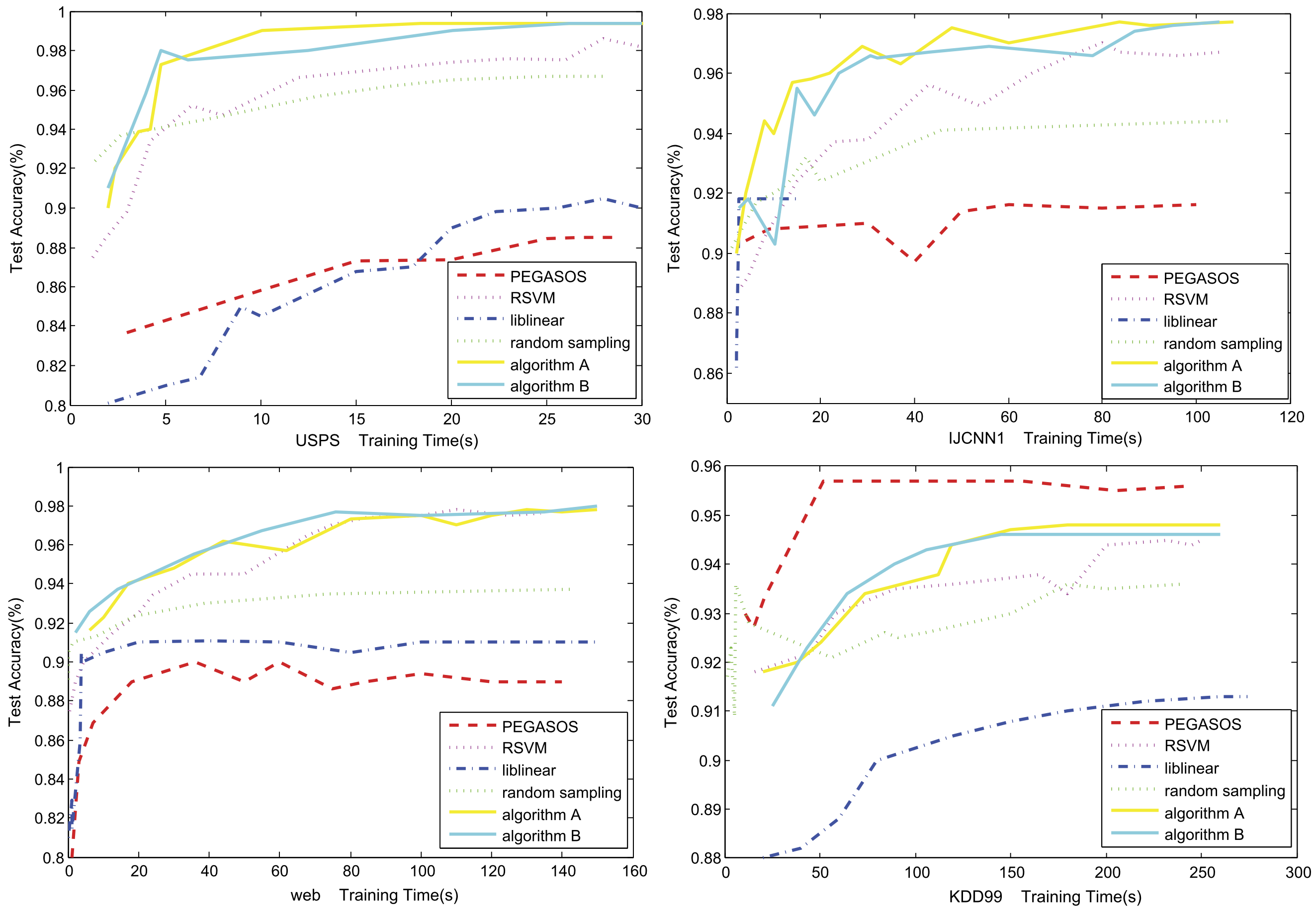


Fig. 9 Testing accuracy versus training time

for the SVM classifier is proposed. It includes two stages, useless data cleaning and informative patterns extraction. A bootstrap sampling based data cleaning method is proposed in the first stage, and two maximum entropy based informative patterns extraction methods are presented in the second stage. In most cases, the training data size is reduced significantly after the two stages. Therefore, training time complexity will be remarkably decreased. Empirical results on four large datasets show that the training process of our approach can be up to 200 times faster than regular SVM while with little performance loss. Meanwhile, comparisons between our approach against three state-of-the-art algorithms demonstrate that our approach is more efficient for large but not sparse datasets.

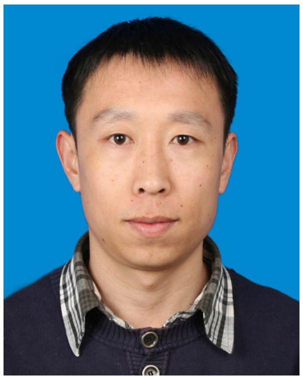
Although the proposed approach is effective, there are some problems that need to be solved in future work. First, in the data cleaning stage, the sampling percentage selection is empirical. The Gelling Point phenomenon may not be prominent for all the datasets. In such cases, to find an appropriate sampling percentage might largely depend on the specific classification task and prior knowledge. Second, how many weak classifiers are needed? Theoretically, the more the better. In our experiments, only 10 weak SVMs classifiers are used. However, more weak classifiers need more training time and more parallel computers. Tradeoff between computing resource and testing accuracy should be carefully considered.

Acknowledgements This work was supported by the National Natural Science Foundation of China (Grant Nos. 61170189, 61370126, 61202239), the Research Fund for the Doctoral Program of Higher Education (Grant No. 20111102130003), and the Fund of the State Key Laboratory of Software Development Environment (Grant No. SKLSDE-2013ZX-19).

References

1. Wang SZ, Li ZJ, Chao WH, Cao QH (2012) Applying adaptive over-sampling technique based on data density and cost-sensitive SVM to imbalanced learning. In: Proceedings of IJCNN
2. Cao YB, Xu J, Liu TY, Li H, Huang YL, Hon HW (2006) Adapting ranking SVM to document retrieval. In: Proceedings of SIGIR, pp 186–193
3. Hasan MA, Chaoji V, Salem S, Zaki M (2006) Link prediction using supervised learning. In: SIAM workshop on link analysis, counter-terrorism and security

4. Burges C (1999) Geometry and invariance in kernel based methods. In: Advances in kernel methods: support vector learning. MIT Press, Cambridge
5. Panda N, Edward YC, Wu G (2006) Concept boundary detection for speeding up SVMs. In: Proceedings of ICML, pp 681–688
6. Graf HP, Cosatto E, Bottou L, Durdanovic I, Vapnik V (2006) Parallel support vector machines: the cascade SVM. In: Advances in neural information processing system, vol 17. MIT Press, Cambridge, pp 521–528
7. Lawrence ND, Seeger M, Herbrich R (2003) Fast sparse Gaussian process methods: the informative vector machine. In: Advances in neural information processing systems. MIT Press, Cambridge
8. Yu H, Yang J, Han J (2003) Classifying large datasets using SVM with hierarchical clusters. In: Proceedings of KDD
9. Vapnik V (1998) Statistical learning theory. Wiley, New York
10. Platt JC (1999) Fast training of support vector machines using sequential minimal optimization. In: Advances in kernel methods—support vector learning. MIT Press, Cambridge, pp 185–208
11. Joachims T (1999) Making large-scale support vector machine learning practical. In: Advances in kernel methods—support vector learning. MIT Press, Cambridge, pp 169–184
12. Kao WC, Chung KM, Sun CL, Lin CJ (2004) Decomposition methods for linear support vector machines. Neural Comput. 16(8):1689–1704
13. Tsang IW, James TK, Cheung PM (2005) Core vector machines: fast SVM training on very large data sets. J Mach Learn Res 6:363–392
14. Lee YJ, Mangasarian OL (2001) RSVM: reduced support vector machines. In: Proceedings of SDM
15. Fine S, Scheinberg K (2001) Efficient SVM training using low-rank kernel representations. J Mach Learn Res 2:243–264
16. Shai SS, Srebro N (2008) SVM optimization: inverse dependence on training set size. In: Proceedings of ICML
17. Joachims T (2006) Training linear SVMs in linear time. In: Proceedings of KDD
18. Smola A, Vishwanathan S, Le Q (2008) Bundle methods for machine learning. In: Advances in neural information processing systems
19. Fan RE, Chang KW, Hsieh CJ, Wang XR, Lin CJ (2008) LIBLINEAR: a library for large linear classification. J Mach Learn Res 9:1871–1874
20. Shai SS, Singer Y, Srebro N (2007) Pegasos: primal estimated subGrAdient solver for SVM. In: Proceedings of ICML
21. Peter LB, Mendelson S (2002) Rademacher and Gaussian complexities: risk bounds and structural results. J Mach Learn Res 3:463–482
22. Guyon I, Matic N, Vapnik V (1994) Discovering informative patterns and data cleaning. In: Proceedings of AAAI workshop on knowledge discovery in databases
23. MacKay D (1992) Information-based objective functions for active data selection. Neural Comput 4(4):590–604
24. Chang CC, Lin CJ (2011) LIBSVM: a library for support vector machines. ACM Trans Intell Syst Technol 2(3):27
25. Chang CC, Lin CJ (2001) IJCNN 2001 challenge: generalization ability and text decoding. In: Proceedings of IJCNN
26. Smits GF, Jordan EM (2002) Improved SVM regression using mixtures of kernels. In: Proceedings of IJCNN
27. Kumar A, Ghosh SK, Dadhwal VK (2006) Study of mixed kernel effect on classification accuracy using density estimation. In: Midterm ISPRS symposium, ITC
28. Shi YH, Gao Y, Wang RL, Zhang Y, Wang D (2013) Transductive cost-sensitive lung cancer image classification. Appl Intell 38(1):16–28
29. Collobert R, Bengio S, Bengio Y (2002) A parallel mixtures of SVMs for very large scale problems. Neural Comput 14:1105–1114
30. Wang CW, You WH (2013) Boosting-SVM: effective learning with reduced data dimension. Appl Intell 39(3):465–474
31. Idris A, Khan A, Lee YS (2013) Intelligent churn prediction in Telecom: employing mRMR feature selection and RotBoost based ensemble classification. Appl Intell 39(3):659–672
32. Maudes J, Diez JJR, Osorio CG, Pardo C (2011) Random projections for linear SVM ensembles. Appl Intell 34(3):347–359



Senzhang Wang was born in Yantai, China. He received the M.Sc. degree in Southeast University, Nanjing, China in 2009. He currently is a third-year Ph.D. student in the School of Computer Science and Engineering at Beihang University, Beijing, China. His main research focus is on data mining and social network analysis.

Zhoujun Li received the B.S. degree in the School of Computer Science from Wuhan University in 1984, and the M.S. and Ph.D. degrees in the School of Computer Science from National University of Defense Technology. Currently, he is working as a professor of Beihang University. His research interests include data mining, information retrieval and information security. He is a member of the IEEE.

Chunyang Liu is affiliated with the China's National Computer Network Emergency Response Technical Team Coordination Center in Beijing. He holds the rank of Associate Professor. He obtained his Ph.D. in EE from Xi'An Jiaotong University in 1997. He received his B.S. in mathematics from Lanzhou University in 1983 and his M.S. in computational mathematics from Xi'An Jiaotong University in 1989. Before joining the National Computer Network Emergency Response Technical Team Coordination Center, he served on the faculty of Xi'An Jiaotong University for more than 10 years. His main research interest is information security.

Xiaoming Zhang was born in Hunan, China, on December 7, 1980. He received the B.Sc. degree, and the M.Sc. degrees in computer science and technology from the National University of Defence Technology, China, in 2003, 2007 respectively. He received his Ph.D. degrees in computer science from Beihang University, in 2012. He is currently working at the school of computer, Beihang University, and he has been the lecturer since 2012. His major interests are text mining, image tagging, and TDT.

Haijun Zhang was born in Henan, China, on December 10, 1975. He received the B.Sc. degree, and the M.Sc. degree in management science from the China University of Mining and Technology, in 1997, 2004 respectively. He worked at China Petroleum First Construction Corporation from 1997 to 2001. He is working at school of information, Beijing Wuzi University since 2001. He is currently the Ph.D. candidate of the school of computer, Beihang University. His major interests are data mining and recommendation system.