PAPER

Training Multiple Support Vector Machines for Personalized Web Content Filters

Dung Duc NGUYEN[†a)], **Maike ERDMANN**[††], **Tomoya TAKEYOSHI**[††], **Gen HATTORI**[††], *Nonmembers*, **Kazunori MATSUMOTO**[††], *Member*, *and* **Chihiro ONO**[††], *Nonmember*

SUMMARY The abundance of information published on the Internet makes filtering of hazardous Web pages a difficult yet important task. Supervised learning methods such as Support Vector Machines (SVMs) can be used to identify hazardous Web content. However, scalability is a big challenge, especially if we have to train multiple classifiers, since different policies exist on what kind of information is hazardous. We therefore propose two different strategies to train multiple SVMs for personalized Web content filters. The first strategy identifies common data clusters and then performs optimization on these clusters in order to obtain good initial solutions for individual problems. This initialization shortens the path to the optimal solutions and reduces the training time on individual training sets. The second approach is to train all SVMs simultaneously. We introduce an SMO-based kernel-biased heuristic that balances the reduction rate of individual objective functions and the computational cost of kernel matrix. The heuristic primarily relies on the optimality conditions of all optimization problems and secondly on the pre-calculated part of the whole kernel matrix. This strategy increases the amount of information sharing among learning tasks, thus reduces the number of kernel calculation and training time. In our experiments on inconsistently labeled training examples, both strategies were able to predict hazardous Web pages accurately (> 91%) with a training time of only 26% and 18% compared to that of the normal sequential training.

key words: *support vector machines, sequential minimal optimization, text categorization, Web content filtering*

1. Introduction

With the help of an Internet monitoring company, we have collected a set of 3.1 million documents with 12,000 features for automatically detecting hazardous Japanese Web pages. Our task is to build multiple filters using Support Vector Machines (SVMs) [1] to accurately identify hazardous content from normal Web pages [2]. However, the scalability is a big challenge due to the fact that SVM needs to solve a computationally expensive quadratic programming (QP) problem. Training the whole data set is expected to take at least one month on a normal computer. If we assume that the training process has to be conducted only once, such a long training time might be acceptable. However, we will certainly encounter different policies on what kind of information is hazardous, depending on which organization is requesting the Web content filter and for what purpose. For instance, parents might want a Web filter that protects their children from adult content such as gambling and drugs. A company might want to set up a filter that prevents employees from visiting Web sites that are not work related. Moreover, laws and ethical standards vary widely among countries.

Under the circumstances explained above, a training example might have to be labeled hazardous in one Web content filter and harmless in another, but the majority of instances is still labeled equally. We therefore introduce several methods to train multiple SVMs in an efficient way. The first one is called Hierarchical Training for Multiple SVMs (HTMSVM), which can identify common data among similar training sets and then train the common data sets in order to obtain good initial solutions. These initial solutions help in reducing the time for training the individual training sets, without influencing classification accuracy. The second approach is to train all SVMs simultaneously. We propose a kernel-biased selection heuristic that balances the kernel calculation cost and reduction rate of multiple objective functions. The heuristic primarily relies on the optimality conditions of all optimization problems and secondly on the pre-calculated part of the whole kernel matrix. Experimental results on our real data show that both the hierarchical (HTMSVM) and simultaneous (mSVM) training methods effectively reduce the cost for calculating the kernel matrix and the training time compared to that of the normal sequential training.

The rest of the paper is structured as follows. In Sect. 2, we introduce the Web content filtering problem and the use of SVMs as personalized filters. In Sect. 3, we describe different approaches to train multiple SMVs, including the normal sequential training and our two mentioned methods. We report and analyze our experiment results in Sect. 4. In Sect. 5, we discuss related works on training time reduction for SVM as well as on the learning of multiple similar tasks. Finally, we draw our conclusions and outline future work in Sect. 6.

2. Web Content Filtering and Support Vector Machines

2.1 Personalized Web Content Filtering

For detecting hazardous Japanese Web pages, we have collected a set of about 3.1 million training examples with

Manuscript received January 16, 2013.
Manuscript revised June 7, 2013.

[†]The author is with the Institute of Information Technology, Vietnam Academy of Science and Technology, Vietnam.
[††]The authors are with the KDDI R&D Laboratories, Fujimino-shi, 356–8502 Japan.
a) E-mail: nddung@ioit.ac.vn

DOI: 10.1587/transinf.E96.D.2376

Table 1 Web content categorization examples.

Class	Category	Subcategory
Harmless	Shopping	Auctions
		Shopping, general
		Real estate
		IT related shopping
	Hobby	Music
		Celebrities
		Food
		Recreation, general
Hazardous	Illegal	Terror, extremism
		Weapons
		Defamation
		Suicide, runaway
	Adult	Sex
		Nudity
		Prostitution
		Adult search

12,000 features. The data is categorized into 27 categories with 103 subcategories. Some examples of hazardous and harmless categories are shown in Table 1. Due to different policies in the filtering of hazardous Web pages, each filter is usually trained independently. Formally speaking, we have a collection of M data sets $T^u, u = 1, \dots, M$:

$$T^u = \left\{(x_i, y_i^u) \in R^d \times \{-1, +1\} \,|i = 1, \dots, l\right\}, \tag{1}$$

and our task is to construct M decision functions:

$$y^u = sgn\,(f^u(x))\,, u = 1, \dots, M, \tag{2}$$

for classifying each document x as hazardous or harmless. Our experience with different learning approaches indicates that nonlinear SVMs with RBF kernels $K(x_1, x_2) = \exp(-\gamma \|x_1 - x_2\|^2)$ satisfy our predictive performance requirement ($\geq$ 90% accuracy). Unfortunately, training nonlinear SVMs takes much time, especially when the number of filters is very large.

2.2 Support Vector Machines

For the last decades, support vector machines [1], [3] have become a popular method in various classification applications. Given a set of training examples $x_i \in R^d$ with labels $y_i \in \{-1, +1\}\,, i = 1, \dots, l$, the training phase of SVMs solves the following QP optimization problem:

$$\min_{\alpha} L(\alpha) = \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j K_{ij} - \sum_{i=1}^{l} \alpha_i, \tag{3}$$

$$s.t. \sum_{i=1}^{l} y_i \alpha_i = 0, \tag{4}$$

$$0 \leq \alpha_i \leq C, i = 1, \dots, l. \tag{5}$$

where $K_{ij} = K(x_i, x_j)$ is a kernel function calculating dot product between two vector x_i and x_j in some feature space; C is a parameter penalizing each "noisy" training example in the given training data. The optimal coefficients $\{\alpha_i\}\,,\ i = 1, \dots, l$ will form a decision function:

$$y = sgn\left(f(x) = \sum_{\alpha_i \neq 0} y_i \alpha_i K(x_i, x) + b\right). \tag{6}$$

Despite the fact that training SVMs is very expensive [4]–[6], their performance is superior in many domain like text categorization [7], character recognition [8], [9], object detection in image [10]. The main difficulty in solving the above QP is the huge memory demand and expensive computational power for calculating and storing the kernel matrix $\{K_{ij}\},\ i, j = 1, \dots, l$. In the next section, we introduce how to train multiple SVMs in an efficient way.

3. Training Multiple Personalized SVMs

3.1 Sequential Training

A normal way to train multiple SVMs is to apply a state-of-the-art algorithm, e.g. the sequential minimal optimization (SMO) [11], and train the machines one by one. The SMO is a special case of the commonly used decomposition method for SVM training. The specialty of SMO lies in the fact that it iteratively selects only two vectors for optimization. Supposed that two vectors x_i and x_j are chosen, the best new values of the corresponding α_i and α_j in terms of reducing the objective function L in (3) are (ignoring the box constraint (5)):

$$\begin{cases} \alpha_i^{new} = \alpha_i^{old} + \frac{y_i(-E_i^{old} + E_j^{old})}{\kappa_{ij}}, \\ \alpha_j^{new} = y_j(const - y_i \alpha_i^{new}), \end{cases} \tag{7}$$

where $const = y_i \alpha_i^{old} + y_j \alpha_j^{old}$, $\kappa_{ij} = K_{ii} + K_{jj} - 2K_{ij}$, and

$$E_i = \sum_{k=1}^{l} y_k \alpha_k K(x_i, x_k) - y_i. \tag{8}$$

This updating scheme leads to a reduction of objective function L an amount of

$$\Delta L_{ij} = -\frac{(-E_i^{old} + E_j^{old})^2}{2\kappa_{ij}}. \tag{9}$$

Based on this reduction rate, different selection strategies have been proposed, e.g. [12], [13]. The second order working set selection, one of the most efficient heuristics, is as follows:

$$\begin{cases} i = \arg\max_{k \in I_{up}(\alpha)} \{-E_k\}\,, \\ j = \arg\min_{k \in I_{low}(\alpha)} \{\Delta L_{ik} | - E_i > -E_k\}\,, \end{cases} \tag{10}$$

where

$$I_{up}(\alpha) = \{k | \alpha_k < C, y_k = +1 \ or\ \alpha_k > 0, y_k = -1\}\,, \tag{11}$$

$$I_{low}(\alpha) = \{k | \alpha_k < C, y_k = -1 \ or\ \alpha_k > 0, y_k = +1\}\,. \tag{12}$$

The SMO algorithm, outlined in Table 2, has been widely implemented in various libraries and softwares for SVM training like LibSVM [14], LASVM [15], Core Vector Machines [16], [17], and Condensed SVMs [18]. However, these existing method and implementation are designed for solving single SVM. In the next section, we describe how to take advantages of the SMO algorithm for training multiple Web content filters.

3.2 Training a Hierarchy of SVMs

Based on the fact that the given M training data share a large number of commonly labeled documents, our first attempt is to train SVMs on the common data and then to use those solutions as initial solutions for training individual data sets. The two-phase algorithm is specified in Table 3.

In the first phase, we detect clusters of overlapping training data using a bottom-up hierarchical algorithm [19] (Step 1–10). Initially, one cluster is created for each training set (Step 1–3). The index set of each cluster ($c \rightarrow Idx$) has one element, which is the index of the data, and the two children ($c \rightarrow Child$) are set to zero. The α parameter ($c \rightarrow \alpha$) is initialized to be zero. In Step 4, $ClusterSet$ is defined as the initial set of M clusters. In Steps 5–10, the bottom-up hierarchical clustering is applied to $ClusterSet$. In each iteration, the two closest clusters c_i and c_j are selected (Step 6). They are then combined into a new cluster c_{ij} (Step 7). The two child clusters c_i and c_j are removed from $ClusterSet$ (Step 8) and the newly created cluster c_{ij} is added instead (Step 9). The distance between the two clusters c_i and c_j is calculated from the total number of training examples without the number of common training examples in the two clusters:

$$dist(c_i, c_j) = l - \left| \bigcap_{u \in c_{ij} \rightarrow Index} T^u \right|, \qquad (13)$$

where $c_{ij} \rightarrow Index = c_i \rightarrow Index \cup c_i \rightarrow Index$.

In Steps 11–12, the remaining cluster in $ClusterSet$ becomes the root of the hierarchy. Its coefficient vector α is initialized to be zero, and it is pushed into a FIFO structure (First-in-First-out).

In the second phase, the SVM training is performed in the order determined by the cluster hierarchy. In each iteration, one data cluster c is picked from $ClusterSet$ in the FIFO structure (Step 14). The SVM training algorithm finds the optimal solution α^c on that cluster. If c has two child

Table 2 The SMO algorithm.

```
Input: training data $T = \{(x_k, y_k)\}, k = 1, \ldots, l$

0. Initialize a feasible solution $\boldsymbol{\alpha}$
1. While $StoppingCondition$ is not satisfied
2.     Select a pair of vectors $(i, j)$ using (10)
3.     Update $(\alpha_i, \alpha_j)$ using (7)
4.     Update optimality violation state $E_k,\ k = 1, \ldots, l$, in (8)
5. Endwhile

Output: optimal coefficients $\boldsymbol{\alpha} = \{\alpha_k\}, k = 1, \ldots, l$
```

Table 3 Algorithm for training a hierarchy of SVMs.

```
Input: $M$-label training data $T^u = \left\{(x_i, y_i^u) \in R^d \times \{-1, +1\} | i = 1, \ldots, l\right\}, u = 1, \ldots, M$

Phase 1: label-based data clustering
1. For $u = 1$ to $M$ do
2.     Define clusters $c_u \rightarrow Idx = \{u\}, c_u \rightarrow Child_1 = c_u \rightarrow Child_2 = NULL, c_u \rightarrow \boldsymbol{\alpha} = \mathbf{0}$
3. Endfor
4. Define $ClusterSet = \{c_u\}, u = 1, \ldots, M$
5. While $|ClusterSet| > 1$
6.     $(c_i, c_j) = \arg\min_{c_u, c_v \in ClusterSet}\{dist(c_u, c_v)\}$
7.     $c_{ij} \rightarrow Idx = c_i \rightarrow Idx \cup c_j \rightarrow Idx, c_{ij} \rightarrow Child_1 = c_i, c_{ij} \rightarrow Child_2 = c_j$
8.     $ClusterSet = ClusterSet - \{c_i, c_j\}$
9.     $ClusterSet = ClusterSet \cup \{c_{ij}\}$
10. Endwhile

Phase 2: training a hierarchy of SVMs
11. Call $root$ is the remaining cluster in $ClusterSet$, set $root \rightarrow \boldsymbol{\alpha} = \mathbf{0}$
12. $Push(root, FIFO)$
13. While $FIFO$ is not empty
14.     $c = Pop(FIFO)$
15.     Solve the problem (3) on $T^c = \bigcap_{u \in c \rightarrow Index} T^u$, call $\alpha^c$ be the optimal solution
16.     If $c \rightarrow Child_1 \neq NULL$ AND $c \rightarrow Child_2 \neq NULL$ Then
17.         $c \rightarrow Child_1 \rightarrow \alpha = \alpha^c$
18.         $c \rightarrow Child_2 \rightarrow \alpha = \alpha^c$
19.         $Push(c \rightarrow Child_1, FIFO)$
20.         $Push(c \rightarrow Child_2, FIFO)$
21.     Endif
22. Endwhile

Output: optimized solutions $\alpha^u$ on $T^u, u = 1, \ldots, M$
```

clusters, α^c will be used as an initial solution for training the child clusters. When all child clusters have been trained, the training process stops. The output of the training are the SVM for each child cluster.

Our experiments show that the training hierarchy of SVMs significantly improves training performance. However, for some individual data sets, an initialization from parent's solution does not help much in reducing the total training time. We will discuss this problem in detail in Sect. 4.2.

3.3 Simultaneous Training of Multiple SVMs

Before introducing our algorithm for simultaneously training all M SVMs, lets review the SMO algorithm described in Sect. 3.1. We can see in Table 2 that the most expensive step is to update the optimal condition on each training example (Step 4). This step consists of calculating two columns of the kernel matrix K_{ik} and K_{jk} and updating the margin

$$E_k^{new} \leftarrow E_k^{old} + (y_i y_k K_{ik}(\alpha_i^{new} - \alpha_i^{old}) + y_j y_k K_{jk}(\alpha_j^{new} - \alpha_j^{old})) \tag{14}$$

This updated information plays a crucial role in the SMO. Firstly, it is used to select the first index i directly and then the second index j via estimating the reduction of the objective function (Eq. (10)). Secondly, it is used to check the optimal condition (step 1 in Table 2). Due to its high computational cost, different strategies has been proposed to improve its efficiency. The first technique is the kernel caching strategy that tries to avoid the recalculation of the kernel matrix [20], [21]. As the computer's memory is limited, only frequently used columns are calculated and store in the main memory. When the stored entries reach the memory limitation, some columns must be freed and the others are (newly) calculated. The second strategy is shrinking. This strategy uses an heuristics to select a subset of the whole training data as an "active" working set and optimization algorithms iteratively work on this subset. Both the two strategies shows their effectiveness in training SVMs on large data [14], [21].

For the Web content filtering task, our solution is to train all M SVMs simultaneously. As all training data sets share the same set of l object descriptions $\{x_i\}$, $i = 1, \dots, l$, the M optimization problems share the same kernel matrix. Normally we could apply the same selection heuristic to select M pairs of vectors, one for each optimization problem, (i^u, j^u), $u = 1, \dots, M$, where:

$$\begin{cases} i^u = \arg\max_{k \in I_{up}(\alpha^u)} \left\{-E_k^u\right\}, \\ j^u = \arg\min_{k \in I^u_{low}(\alpha^u)} \left\{\Delta L_{ik}^u | - E_i^u > -E_k^u\right\}. \end{cases} \tag{15}$$

The updating marginal information becomes

$$E_k^{u\_new} \leftarrow E_k^{u\_old} + (y_i^u y_k^u K_{i^u k}(\alpha_i^{u\_new} - \alpha_i^{u\_old}) + y_j^u y_k^u K_{j^u k}(\alpha_j^{u\_new} - \alpha_j^{u\_old})). \tag{16}$$

If the whole kernel matrix is pre-calculated and stored in the main memory, then kernel entities are calculated only once and the training process for M SVMs becomes very fast. However, the size of the kernel matrix grows rapidly with the number of selected pairs $\{i^u, j^u\}$ that corresponds to the number of kernel columns $\{K_{i^u}, K_{j^u}\}$, $u = 1, \dots, M$. When M problems are solved together, these columns might be different and the memory vanishes quickly even for a small number of iterations. As the result, some pre-calculated columns must be freed and the newly selected columns are (re-)calculated. This problem wastes computational power and increase the training time, especially when the training size l and number of problems M are large.

In order to reduce the wasted kernel calculations, we used an heuristic called *kernel biased* selection as follows. Firstly, we select a problem U that violates the optimal conditions the most:

$$U = \arg\max_u \{-E_{max}^u + E_{min}^u|, u = 1, \dots, M\}, \tag{17}$$

where

$$E_{max}^u = \max\left\{-E_k^u | k \in I_{up}(\alpha^u)\right\},$$
$$E_{min}^u = \min\left\{-E_k^u | k \in I_{low}(\alpha^u)\right\}.$$

We then select pairs of vectors

$$\begin{cases} i^U = \arg\max_k \left\{-E_k^U | k \in I_{up}(\alpha^U)\right\}, \\ j^U = \arg\min_k \left\{\Delta L_{ik}^U | k \in I_{low}(\alpha^U), -E_i^U > -E_k^U\right\}, \end{cases} \tag{18}$$

and for $v \neq U$:

$$\begin{cases} i^v = \begin{cases} \arg\max_k \left\{-E_k^v | k \in I_{up}^U(\alpha^v)\right\} \text{ if } I_{up}^U \neq \emptyset, \\ -1 \text{ otherwise,} \end{cases} \\ j^v = \begin{cases} \arg\min_k \left\{\Delta L_{ik}^v | k \in I_{low}^U, -E_i^v > -E_k^v\right\} \\ \qquad\qquad \text{if } I_{low}^U \neq \emptyset, \\ -1 \text{ otherwise,} \end{cases} \end{cases} \tag{19}$$

where

$$I_{up}^U(\alpha^v) = I_{up}(\alpha^v) \cap CacheIdx \cup \{i^U, j^U\},$$
$$I_{low}^U(\alpha^v) = I_{low}(\alpha^v) \cap CacheIdx \cup \{i^U, j^U\},$$

and $CacheIdx$ is the set of indexes of pre-calculated kernel matrix columns. As $I_{up}^U(\alpha^v)$ and $I_{low}^U(\alpha^v)$ are subsets of $I_{up}^u(\alpha^v)$ and $I_{low}^u(\alpha^v)$, the selection heuristics (18)–(19) leads to a smaller total reduction rate of M objective functions $L(\boldsymbol{\alpha}^u)$. However, it requires calculation of at most two columns of the kernel matrix. We will show in the experiment section that this trade-off is effective and suitable for the Web content filtering problem. The outline of simultaneously training multiple SVMs is described in Table 4.

4. Experiment

4.1 Data Preparation and Experiment Setup

For the experiment, we used part of our corpus of about 3,1

Table 4 Algorithm for simultaneously training of multiple SVMs.

```
Input: M training data $T^u = \{(x_k, y^u_k)\}, k = 1, \ldots, l; u = 1, \ldots, M$.

0. Initialize M feasible solutions $\alpha^u, u = 1, \ldots, M$
1. While all M StoppingConditions are not satisfied
2.     Select M pairs of vectors $\{(i^u, j^u)\}$ using (18)–(19)
3.     Forall $\{(i^u, j^u)\}$
4.         If $i^u \geq 0$ and $j^u \geq 0$ Then
5.             Update $(\alpha^u_i, \alpha^u_j)$ using (7)
6.             Update optimality violation state $E^u_k$ using (16)
7.         Endif
8.     Endfor
9. Endwhile

Output: M sets of optimal coefficients $\alpha^u = \{\alpha^u_k\}, u = 1, \ldots, M$.
```

million examples manually labeled as hazardous or harmless Web pages. The feature extraction is based on the dependency relations of hazardous keywords and their neighboring segments, thus the features are numerical. The extraction process consists of several steps, including generation of keywords using morphological analysis on the training data, generation of segment pairs and expansion the segment pair using a thesaurus [2]. We also conducted our experiments on an extended set of the previously reported data in [2], which contains nearly equal distribution of hazardous and harmless Web pages. From the corpus, we constructed five training sets representing five different data labeling policies ranging between conservative (e.g. parental control) Web filters to very liberal Web filters. For the first set, we used the original labels. In order to construct the other four sets, we changed the labels of selected categories and subcategories from harmless to hazardous or vice versa. As a result, 80% of the training examples were identical in all five sets, 10% were different in only one of the training sets and another 10% were different in two training sets. Although the training sets were created solely for the experiment, we made sure that they represented realistic Web filtering policies.

In our experiment, we compared the training time of HTMSVM and mSVM with those of sequential training using LibSVM [14] for different amounts of training data. The smallest set contained 50,000 training examples per training set whereas the largest contained 250,000 training examples. Due to the strict requirement on predictive accuracy of Web content filters, we select the RBF kernel for all training methods. The experiment was conducted on a server with 8 CPUs and 60GB of memory; 10GB was used for caching the kernel matrix. The stopping condition is the same for all three algorithms: the maximum of KKT violations, $(-E_i + E_j)$ for LibSVM, HTMSVM and $(-E_{i^U} + E_{j^U})$, for mSVM is smaller than $\epsilon = 10^{-3}$. Actually, the stopping condition used by the mSVM is tighter than the other due to the fact that $(-E_{i^u} + E_{j^u}) \leq (-E_{i^U} + E_{j^U}) < \epsilon$ for all $u \neq U$, or ϵ is the threshold for the most of the most violation conditions of all M training problems.

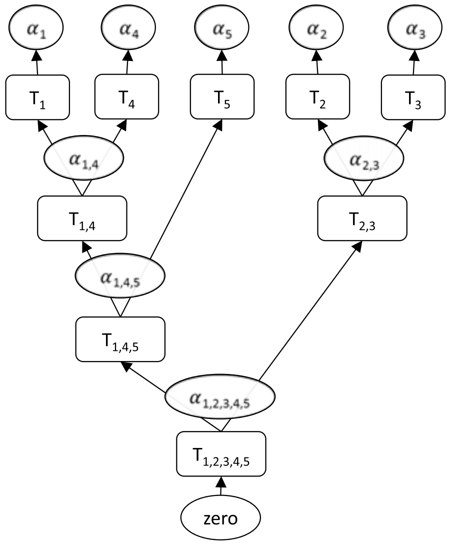

Fig. 1 Data clusters and training order for the Web content filter using the HTMSVM algorithm.

4.2 Training Time

For the first hierarchical training strategy, the result of data clustering and training order is visualized in Fig. 1. The first training is conducted on $T_{1,2,3,4,5}$, i.e. the training data which is identical in all five sets (80% of the data). After that, the set $T_{1,4,5}$ is trained, which consists of the training data that is identical in three of the training sets (90% of the data). Subsequently, the sets $T_{1,4}$ and $T_{2,3}$ are trained, both containing 95% of the training examples. Finally, each training set is trained individually. Figure 2 details the training time of the HTMSVM and LibSVM for the 250,000 training examples. As the results show, the training time for the private training data is noticeably shorter for HTMSVM than for LibSVM. As a consequence, the overall training time of HTMSVM lies between 26% and 41% of that of LibSVM, even though the training time for the common data sets has to be added. In this experiment, we include the clustering time in the total training time of the HTMSVM. In fact, it takes only several seconds to create the hierarchy, because the calculation of Eq. (13) is based on a set of indexes, not a float function like Euclidean distance, and it can be calculated very efficiently.

If we look at the Fig. 2 more closely, we can see that the training times for individual data sets are different in the normal sequential training, e.g. training time for P2 is much longer than for P3. Normally, the more complex problem is, the more time it takes the SVM to converge. However, the HTMSVM takes nearly equal time for training private data sets. It means that the HTMSVM is more effective on the P3 and is less effective on the P2 data set, or the solution on a parent node might have different effect of reducing the training time on its children nodes (P2 and P3 share the same parents).

In Table 5, we report training time and number of kernel calculations (calls to the function $K(x_i, x_j)$) of all train-

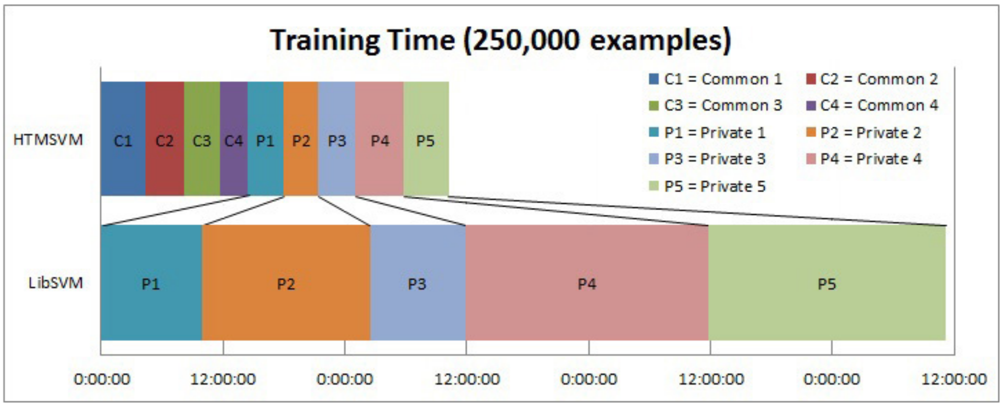

Fig. 2 Detailed training time of the HTMSVM on the largest experiment data.

Table 5 Number of kernel calculation and time of the sequential training (LibSVM), training a hierarchy of SVM (HTMSVM), and simultaneous training (mSVM) for the Web content filter.

	LibSVM		HTMSVM		mSVM	
Training Size	#Kernel	Time	#Kernel	Time	#Kernel	Time
50,000	4.68E+09	2:11:58	1.42E+09	0:54:44	1.38E+09	0:40:01
100,000	2.74E+10	15:14:17	5.19E+09	3:58:56	5.15E+09	2:43:29
150,000	4.78E+10	25:38:48	1.11E+10	8:52:46	1.10E+10	5:46:46
200,000	1.04E+11	59:06:13	2.1E+10	16:58:05	1.92E+10	11:17:50
250,000	1.42E+11	83:10:33	4.6E+10	34:13:55	3.43E+10	18:34:18

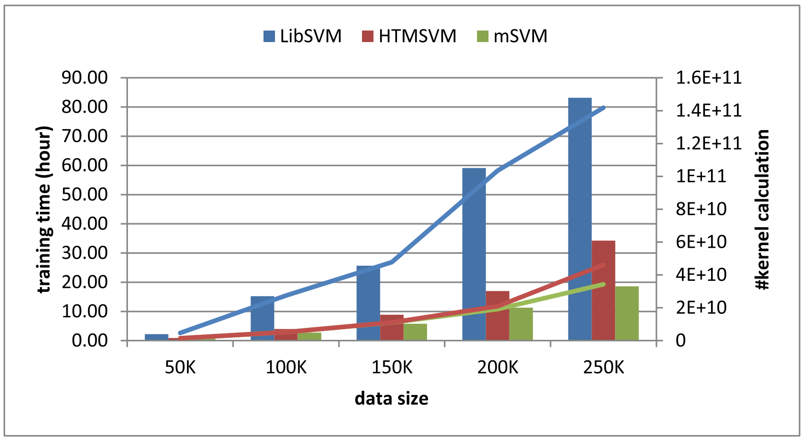

Fig. 3 Comparison of training time (column) and number of kernel calculations (line) of three training strategies on the Web content filtering data.

ing methods: sequential training using LibSVM, data clustering and then training a hierarchy of clusters (HTMSVM), and simultaneous training of all data sets using the kernel-biased selection heuristic (mSVM). We can see that mSVM effectively reduces the training time for multiple Web content filters: only 30% to 18% of that of the sequential training.

Figure 3 shows the direct relationship between training time and number of kernel calculations. The HTMSVM and mSVM requires a much smaller number calculations than those of the normal training. As the result, the two methods significantly reduce the training time for our Web content filters. The HTMSVM prefers working on the kernel submatrix of the commonly labeled data, whereas the mSVM biases the optimization on the pre-calculated columns of the whole matrix.

4.3 Predictive Accuracy

After the training phase, we tested the classification accuracy of the individual filters. For each of the five training sets, we have a corresponding test set containing 50,000 samples. We predicted the classification accuracy for each of the test sets and then calculated the average accuracy of the five results. As shown in Table 6, the classification result of three training strategies is not identical, yet the difference in accuracy ($< 0.04\%$) is neglectible. Furthermore, the classification accuracy of the linear kernel was noticably lower than that of the RBF kernel, which shows that saving training time by using the linear kernel is not an option.

Table 6 Predictive accuracy of Web content filters.

	Training Size				
Method	50,000	100,000	150,000	200,000	250,000
LibSVM (linear)	82.1988%	83.7752%	84.6660%	85.3120%	85.7216%
LibSVM (RBF)	89.0704%	90.0500%	90.7504%	91.0444%	91.2540%
HTMSVM (RBF)	89.0696%	90.0500%	90.7528%	91.0432%	91.2208%
mSVM (RBF)	89.0708%	90.0492%	90.7536%	91.0420%	91.2696%

For the classifier trained on 250,000 examples, a classification accuracy of about 91% was achieved. Since a minimum accuracy of 95% is required to ensure applicability of the classifier to real applications, we need to train the whole data set of 3.1 million training examples, which is expected to take more than one month. For that reason, we can expect that the reduction of training time using the presented methods will be substantial.

5. Discussion

Support Vector Machines have proved to be an effective tool for a wide range of classification problems including text categorization and, in particular, Web content filtering. Many attempts have been made to train SVMs efficiently on large-scale data, e.g. on-line and active learning [15], approximation [18], [22], parallelization [23]. However, most of the mentioned works focused on solving a single large optimization problem. Our attempt is to train multiple problems on the same data description but with different labeling policies.

In order to differentiate our contribution, we will introduce research on the training of similar tasks which focuses on improving classification accuracy and/or reducing training time.

Divide-and-conquer is one of the most widely used strategies for dealing with large scale problem, not only for the expensive SVM training but also for machine learning applications in general. In [24], [25] and [23], the authors proposed to divide the whole training data into clusters and then "expert" classifiers are learned from these clusters. The final decision function is formed based on the set of these experts. While the parallel mixture of SVMs [23] solves the large scale problems in terms of training examples, the CombNET framework [24] focuses on the problems with large number of classes like Chinese character recognition [24]. In our application of Web content filters, the HTMSVM and mSVM are proposed to train multiple SVMs on the same data description but different labeling policies. The HTMSVM uses the divide-and-conquer approach to find locally optimized solutions and then, as the result, individual classifiers could be trained more quickly.

In multi-task learning [26]–[29], related tasks are learned simultaneously or sequentially. The approach is often used when the number of training examples is insufficient for a single task and the learned model could be low in generalization performance. One typical example for multi-task learning is the simultaneous recognition of attributes such as age, sex and facial expression of a person in a photograph. Multi-task learning is a form of transfer learning, but transfer learning [30], [31] does not necessarily require the feature space of the related tasks to be identical. Transfer learning can be applied, for instance, if the knowledge obtained from training a classifier for customer reviews should be transfered to customer reviews on a different product. The previous knowledge can be transfered in form of e.g. reusing features or training examples that are present in both tasks. Several proposals have been made to apply transfer learning to the task of spam filtering, but based on the assumption that the personalization of the classifiers has to be undertaken using unlabeled data [32].

Our task is also not to be confused with multi-label classification, e.g. [33], in which more than one label can be assigned to each training example. Multi-label classification often occurs in e.g. text classification. For instance, a newspaper article might be assigned both the label "politics" and the label "economy". In our classification problem, however, each training example is assigned only one label per training set.

Yet another related research area is incremental learning [34]–[36], which is suitable particularly for dynamic applications, where new training data is added or existing training data needs to be revised frequently. Thereby, a first classifier is built on the training data available at a given time, and a second classifier is built on the updated training set, i.e. new and modified examples, by reusing the results of the first training process. Incremental learning can also be used to train subsets of a training corpus in cases where training of the whole data at once would take too much time.

6. Conclusion

We have introduced two methods for training personalized Web content filters using SVMs: training a hierarchy of SVMs and simultaneously training multiples of SVMs using a kernel-bias selection heuristic. Both the two methods HTMSVM and mSVM aim at reducing the number of kernel calculations, thus reducing the training time. We tested our proposed method in an experiment in which we trained five personalized Web content filters from an identical training corpus. In the experiment, 20% of the training examples were labeled inconsistently, assuming that different policies exist on which Web pages should be labeled as hazardous. The filters using SVMs were able to predict hazardous Web pages with an accuracy of more than 91%, but they require only 26% (HTMSVM) and 18% (mSVM) of the training time of the sequential training using LibSVM.

The two proposed methods could be applied to other kinds of transfer learning problems, where we have several classification tasks with overlapping training sets. For that reason, we also want to apply HTMSVM and mSVM to other applications, such as recommendation systems. In order to avoid negative transfer, i.e. causing an increase in training time, it is necessary to develop a method to estimate for which applications the proposed method is useful.

In the next step, we want to not only further reduce the training time but also increase the performance of the filters. One direction is to combine the two methods into one framework that could train a large number of filters. The second direction is to improve the predictive performance of individual filters using different transfer learning techniques. With a large number of carefully labeled data, we are going to build highly accurate filters for a large number of Web users.

Acknowledgment

We would like to thank the reviewers and the Associate Editor for their valuable comments and suggestions on the submission version of this paper. The first author was supported by the Vietnam National Foundation for Science and Technology Development under NAFOSTED Grant 102.02.16.09.

References

[1] C. Cortes and V. Vapnik, "Support vector networks," Mach. Learn., vol.20, pp.273–297, 1995.
[2] K. Ikeda, T. Yanagihara, G. Hattori, K. Matsumoto, and Y. Takishima, "Hazardous document detection based on dependency relations and thesaurus," Australasian Conference on Artificial Intelligence, pp.455–565, 2010.
[3] V. Vapnik, The Nature of Statistical Learning Theory, Springer, N.Y., 1995.
[4] C.J.C. Burges, "A tutorial on support vector machines for pattern recognition," Data Mining and Knowledge Discovery, vol.2, no.2, pp.121–167, 1998.
[5] L. Bottou, O. Chapelle, D. DeCoste, and J. Weston, Large-Scale Kernel Machines (Neural Information Processing), The MIT Press, 2007.
[6] C. Cristianini and J. Shawe-Taylor, An Introduction to Support Vector Machines, Cambridge University Press, 2000.
[7] T. Joachims, "Text categorization with support vector machines: Learning with many relevant features," Proc. European Conference on Machine Learning, ed. C. Nedellec and C. Rouveirol, pp.137–142, Berlin, 1998.
[8] Y. LeCun, L. Botou, L. Jackel, H. Drucker, C. Cortes, J. Denker, I. Guyon, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, "Learning algorithms for classification: A comparison on handwritten digit recognition," Neural Netw., pp.261–276, 1995.
[9] C. Liu, K. Nakashima, H. Sako, and H. Fujisawa, "Handwritten digit recognition: bench-marking of state-of-the-art techniques," Pattern Recognit., vol.36, pp.2271–2285, 2003.
[10] E. Osuna, R. Freund, and F. Girosi, "Training support vector machines:an application to face detection," IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico, pp.130–136, Jan. 1997.
[11] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in Advances in Kernel Methods - Support Vector Learning, ed. B. Schoelkopf, C.J.C. Burges, and A.J. Smola, pp.185–208, MIT Press, Cambridge, MA, 1999.
[12] S. Keerthi, S. Shevade, C. Bhattacharyya, and K. Murthy, "Improvements to platt's smo algorithm for svm classifier design," Neural Computation, vol.13, pp.637–649, March 2001.
[13] R.E. Fan, P.H. Chen, and C.J. Lin, "Working set selection using the second order information for training svm," J. Machine Learning Research, vol.6, pp.1889–1918, 2005.
[14] C.C. Chang and C.J. Lin, "LIBSVM: A library for support vector machines," ACM Transactions on Intelligent Systems and Technology, vol.2, pp.27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.
[15] B. Antoine, E. Seyda, W. Jason, and B. Léon, "Fast kernel classifiers with online and active learning," Journal of Machine Learning Research, vol.6, pp.1579–1619, Sept. 2005.
[16] I.W. Tsang, J.T. Kwok, and P.M. Cheung, "Core vector machines: Fast svm training on very large data sets," J. Mach. Learn. Res., vol.6, pp.363–392, 2005.
[17] I.W. Tsang, A. Kocsor, and J.T. Kwok, "Simpler core vector machines with enclosing balls," ICML '07: Proc. 24th International Conference on Machine Learning, New York, NY, USA, pp.911–918, 2007.
[18] D.D. Nguyen, K. Matsumoto, Y. Takishima, and K. Hashimoto, "Condensed vector machines: Learning fast machine for large data," IEEE Trans. Neural Netw., vol.21, no.12, pp.1903–1914, 2010.
[19] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning, Springer Series in Statistics, Springer New York, York, NY, USA, 2001.
[20] B. Léon, C. Olivier, D. Dennis, and W. Jason, eds., Large Scale Kernel Machines, MIT Press, Cambridge, MA, 2007.
[21] T. Joachims, "Making large-scale support vector machine learning practical," in Advances in Kernel Methods: Support Vector Machines, ed. A.S.B. Scholkopf, C. Burges, MIT Press, Cambridge, MA, 1998.
[22] S.S. Keerthi, O. Chapelle, and D. Decoste, "Building support vector machines with reduced classifier complexity," J. Machine Learning Research, vol.7, pp.1493–1515, 2006.
[23] C. Ronan, B. Samy, and B. Yoshua, "A parallel mixture of svms for very large scale problems," Neural Comput., vol.14, no.5, pp.1105–1114, 2002.
[24] M. Kugler, S. Kuroyanagi, A.S. Nugroho, and A. Iwata, "Combnet-iii: A support vector machine based large scale classifier with probabilistic framework," IEICE Trans. Inf. & Syst., vol.E89-D, no.9, pp.2533–2541, Sept. 2006.
[25] M. Kugler, S. Kuroyanagi, A.S. Nugroho, and A. Iwata, "Combnet-iii with nonlinear gating network and its application in large-scale classification problems," IEICE Trans. Inf. & Syst., vol.E91-D, no.2, pp.286–295, Feb. 2008.
[26] Y.S. Abu-Mostafa, "Learning from hints in neural networks," J. Complexity, vol.6, no.2, pp.192–198, 1990.
[27] R. Caruana, "Multitask learning: A knowledge-based source of inductive bias," Proc. Tenth International Conference on Machine Learning, pp.41–48, 1993.
[28] S. Thrun, "Is learning the n-th thing any easier than learning the first?," Advances in Neural Information Processing Systems, pp.640–646, 1996.
[29] J. Baxter, "A model of inductive bias learning," J. Artificial Intelligence Research, vol.12, pp.149–198, 2000.
[30] A. Arnold, R. Nallapati, and W.W. Cohen, "A comparative study of methods for transductive transfer learning," Proc. Seventh IEEE International Conference on Data Mining Workshops, pp.77–82, 2007.
[31] S.J. Pan and Q. Yang, "A survey on transfer learning," IEEE Trans. Knowl. Data Eng., vol.22, no.10, pp.1345–1359, 2010.
[32] S. Bickel, "Ecml-pkdd discovery challenge 2006 overview," ECML-PKDD Discovery Challenge Workshop, pp.1–9, 2008.
[33] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," Int. J. Data Warehousing and Mining, vol.2007, pp.1–13, 2007.

[34] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," Advances in Neuronal Information Processing Systems, vol.13, pp.409–415, 2000.
[35] S. Ruping, "Incremental learning with support vector machines," IEEE International Conference on Data Mining, pp.641–642, 2001.
[36] A. Shilton, M. Palaniswami, D. Ralph, and A.C. Tsoi, "Incremental training of support vector machines," IEEE Trans. Neural Netw., vol.16, no.1, pp.114–131, 2005.

Dung Duc Nguyen received the Bachelors degree in mathematics in 1994. He received the Masters and Ph.D. degrees in knowledge science from the Japan Advanced Institute of Science and Technology, Japan, in 2003 and 2006, respectively. He was a Research Engineer at KDDI R&D Laboratories Inc., Japan. He is now with the Institute of Information Technology, Vietnam Academy of Science and Technology, Ha Noi, Vietnam. His current research interests include machine learning, pattern recognition, and data mining. Dr. Nguyen was awarded the Innovative Medal from the Youth Union of Vietnam in 1998 for developing the first Vietnamese optical character recognition software, and the Technical Support Achievement Award in 2008 for his contributions at KDDI Laboratories.

Maike Erdmann received her B.Sc. in Computing Science from CvO University Oldenburg, Germany in 2006 and her Master and Ph.D. of Information Science and Technology from Osaka University, Japan in 2008 and 2011 respectively. She joined KDDI R&D Laboratories, Japan as an Associate Research Engineer in the Intelligent Media Processing Laboratory. Her research interests include knowledge extraction from the WWW and natural language processing.

Tomoya Takeyoshi received the B.E. degree of Information Engineering,the M.S. degree of Computer Science from Hokkaido University, Japan in 2005 and 2007 respectively. Since joining KDDI in 2007, he has been working on Web data mining technology and social media analysis.

Gen Hattori received the B.E. and M.E. degrees of Electrical and Electronic Engineering from Kobe University in 1996 and 1998 respectively, and PhD degree in information science from Osaka University, Japan. He subsequently joined Kokusai Denshin Denwa Co., Ltd. (now KDDI) in 1998, he has been working on network management systems, intelligent transportation systems and software agent systems, Web text mining. He is currently a research engineer of Intelligent Media Processing Lab. in KDDI R&D Laboratories, Inc. He received Young Engineers Award of IEICE in 2004.

Kazunori Matsumoto received the B.E. and M.E. degrees in information science from Kyoto University, Kyoto, Japan, in 1984 and 1986, and the Ph.D. degree from Ritsumeikan University, Kyoto, in 2009, respectively. He has been with KDDI Research and Development Laboratories, Saitama, Japan, since 1986, and is currently a Senior Research Engineer of the Intelligent Media Processing Group, Osaka, Japan. His current research interests include multimedia retrieval and content analysis.

Chihiro Ono received the B.E. degree of Electrical Engineering, the M.S. degree of Computer Science, Ph.D degree of Science for Open and Environmental Systems from Keio University, Japan, in 1992, 1994 and 2009 respectively. Since joining KDD in 1994, he has been working on database systems, software agent technologies, personalization technologies and social media analysis, from 1999 to 2000, he was a visiting researcher at Stanford University. He is currently a Senior Manager in KDDI R&D Laboratories, Inc. He received Best Paper Award for Young Researchers of the National Convention of IPSJ in 1996.