Journal of Experimental & Theoretical Artificial Intelligence
Vol. 24, No. 2, June 2012, 193–210

Training SVM email classifiers using very large imbalanced dataset

Lili Diao[a]*, Chengzhong Yang[b] and Hao Wang[c]

[a]*Core Technology – Research, Trend Micro Inc., Nanjing 210012, Jiangsu Province, China;* [b]*School of Management and Engineering, Nanjing University, Nanjing 210093, Jiangsu Province, China;* [c]*Department of Computer Science, The University of Hong Kong, Hong Kong, China*

(*Received 3 June 2010; final version received 5 June 2011*)

The Internet has been flooded with spam emails, and during the last decade there has been an increasing demand for reliable anti-spam email filters. The problem of filtering emails can be considered as a classification problem in the field of supervised learning. Theoretically, many mature technologies, for example, support vector machines (SVM), can be used to solve this problem. However, in real enterprise applications, the training data are typically collected via honeypots and thus are always of huge amounts and highly biased towards spam emails. This challenges both efficiency and effectiveness of conventional technologies. In this article, we propose an undersampling method to compress and balance the training set used for the conventional SVM classifier with minimal information loss. The key observation is that we can make a trade-off between training set size and information loss by carefully defining a similarity measure between data samples. Our experiments show that the SVM classifier provides a better performance by applying our compressing and balancing approach.

Keywords: email classification; support vector machine; imbalance learning; training set compression; undersampling

1. Introduction

During the past one or two decades, with the rapid spread of the Internet, email has become the most popular method of communication. It is really hard to tell exactly how many people out there are using email today, yet it is reported that over 90% of US netizens use email (Jones and Fox 2009), consolidating our opinion that email is playing an important role in modern society. Unfortunately, the efficient and economic nature of email has been abused. The Internet is now filled with spam, which annoys people, undermine the integrity of email and directly degrade life online (Fallows 2003). What is worse is that spam is not easily annihilated via conventional methods such as black/white list or hand-crafted rules since they always keep changing and evolving. Therefore, there has been an increasing demand for smart and reliable spam filters.

The most recent advances on this tough issue are to use machine learning methods (Sahami, Dumais, Heckerman, and Horvitz 1998; Drucker, Wu, and Vapnik 1999; Carreras and Marquez 2001). There should be no doubt that machine-learning methods are quite promising on this issue since the very purpose of their development is to tackle

*Corresponding author. Email: LiLi_Diao@trendmicro.com.cn

ISSN 0952–813X print/ISSN 1362–3079 online
© 2012 Taylor & Francis
http://dx.doi.org/10.1080/0952813X.2011.610033
http://www.tandfonline.com

changing and evolving problems. Typically, machine learning methods take the spam filtering problem as a binary classification task. They first learn a classifier from the labelled training data (i.e. emails with marks telling which are spams and which are not), and then use the classifier to classify incoming data (i.e. new emails). Among all candidate machine-learning methods, support vector machines (SVMs; Vapnik 1995, 1998) are often preferred by enterprise applications due to their unique characteristics, which include but are not necessarily limited to

- Better performance on small-sample learning, non-linear and high-dimensional pattern recognition;
- Theoretical guarantee of global optimal solution; and
- Reasonable trade-off between time and performance

Whereas SVMs as well as other machine-learning methods do have excellent performance in experiments and some preliminary applications, the results are still far from satisfactory when they are used in real enterprise applications. The main reason is that the experimental performance may rely on assumptions that are actually invalid. For example, conventional machine-learning methods would generally assume a moderately-sized, balanced training set (i.e. a training set that is not too big and not too small and has a nearly equal number of positive and negative samples[1]). However, the actual process of sample collection makes it impossible for this assumption to be valid. On the one hand, millions of spams are collected every day from honeypots (also known as spamtraps) (Wikipedia 2009) distributed all over the Internet. On the other hand, it is obviously impossible to collect millions of normal emails every day due to privacy issues. In Mainland China, according to an investigation reported by the Internet Society of China (ISC), the ratio of spams to normal emails is roughly 3:2 (Anti-Spam Center of ISC 2008). However, the ratio in actual collected dataset is much more imbalanced, with tens or even hundreds times more spams than normal spams. What is more, the enterprise spam classifier needs to be upgraded frequently, using newly collected spams to capture their latest features. This dynamic process is well illustrated in Figure 1. During each training period i in this process, old training data T_{i-1} should never be simply discarded since otherwise the new classifier would probably fail to detect old-fashioned spams. As a consequence, if without any processing, the large and imbalanced training dataset T_i will expand rapidly and soon get intractable.

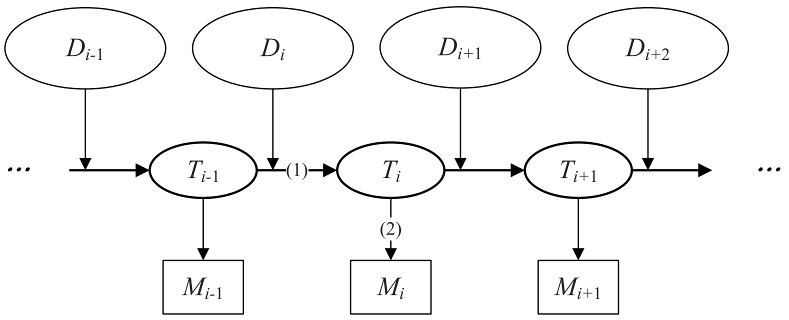

Figure 1. The SVM training process in our application. T_i is the training dataset in period i; M_i the trained model; and D_{i+1} the incoming data for training model M_{i+1}. (1) T_i is obtained by merging T_{i-1} and D_i. (2) Learn an SVM model M_i from T_i.

To sum up, the challenges to conventional SVMs and even the entire anti-spam industry mainly lie in three aspects:

(1) *Efficiency*. A very large training set always means very long training time. In practice, we find that sometimes our SVM needs weeks to train a classifier. This low efficiency not only delays release of upgraded anti-spam patterns, but also limits the opportunity of inner testing before release, causing potential flaws in security products.
(2) *Effectiveness*. An imbalanced training set leads to an irresponsible classifier that tends to classify every incoming email as a spam, which is apparently unacceptable. In fact, people would like to tolerate spams rather than miss important personal or business emails.
(3) *Economy*. Enterprises typically do not implement machine-learning algorithms by themselves. Instead, they purchase, using tens of thousands or even millions of dollars. Therefore, to throw away the current SVM and purchase a state-of-the-art one would probably be the last thing that an enterprise ever wants to do.

Despite the financial considerations, various approaches have been proposed to deal with the first two challenges. For example, Osuna, Freund, and Girosi (1997), Platt (1998) and Joachims (2006) aim to speed up SVM training process in case of large training dataset. Cauwenberghs and Poggio (2001), Diehl and Cauwenberghs (2003) and Fung and Mangasarian (2002) follow the idea of incremental learning. Akbani, Kwek, and Japkowicz (2004), Wu and Chang (2005) and Tang, Zhang, Chawla, and Krasser (2009) focus on improving the performance of SVM on imbalanced datasets. We will give a brief introduction and discussion of these works in Section 2. But generally speaking, none of these solutions meet our practical requirement of efficiency, effectiveness and economy.

Therefore in this article, we propose an undersampling method to compress and balance the training dataset with minimal information loss. The key observation here is that we can make a trade-off between information loss and the size of the training set by carefully defining a similarity measure between data samples. Then by using different compression rates in positive and negative data sample sets, we can get a smaller and more balanced training set. We consider this compressed and balanced dataset a core subset of the original very large imbalanced dataset. To achieve better performance, we keep all support vectors (SVs) in this core subset as well as other auxiliary data vectors. Empirical studies show that training using this core subset does yield a better performance than training using the original dataset.

The rest of this article is organised as follows. In the following section, we briefly introduce SVM classification as well as some recent advances in dealing with very large imbalanced training sets. Then, in Section 3, we describe our approach in detail. In Section 4, we do some experiments using a real-world training set and analyse the results. In Section 5, we further discuss some issues in our approach as well as some possible improvements. Finally, Section 6 concludes the article.

2. Background

2.1. *SVM classification*

SVM is a statistical learning method proposed by Vapnik (1995, 1998) in the late 1990s. As illustrated in Figure 2, the training set T of SVM may be represented as $T = \{(x_1, y_1),$

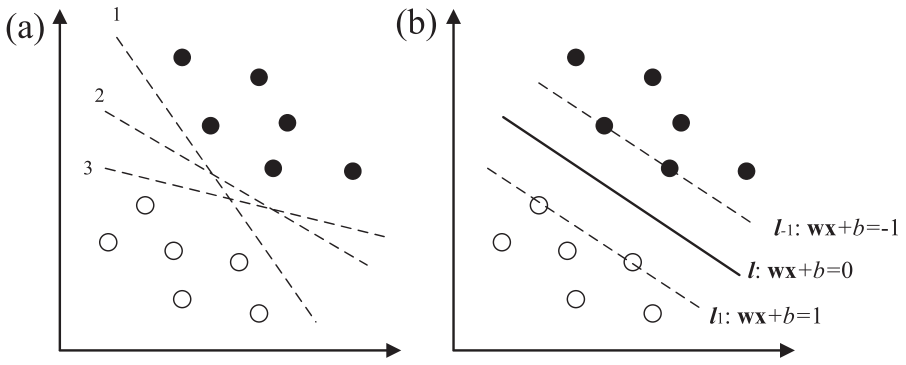

Figure 2. (a) Three possible hyperplanes that may correctly classify the dataset and (b) the optimal hyperplane that maximises the margin of two classes.

$(x_2, y_2)\ldots(x_n, y_n)\}$ where each $x_i \in R^d$ is an d-dimensional vector and each $y_i \in \{(-1, +1)\}$ is the class label indicating whether it is a negative sample or a positive sample. To separate the two classes, an intuitive idea is to place a hyperplane between them. Figure 2(a) shows three possible hyperplanes that may correctly classify the data. In consideration of the generalisation ability, SVM is biased towards the optimal hyperplane that maximises the margin between two classes, as shown in Figure 2(b).

Mathematically speaking, in order to find the optimal hyperplane l in Figure 2(b) one should solve the following optimisation problem:

$$\begin{aligned} &\min_{\mathrm{w,b}} && w \cdot w \\ &\text{s.t.} && y_i(w \cdot x_i + b) \geq 1. \quad (i = 1, 2, \ldots, n) \end{aligned} \tag{1}$$

This can be done via a Lagrange multiplier method which transforms Problem (1) into a technically easier problem:

$$\begin{aligned} &\max_{\alpha} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{n} y_i y_j \alpha_i \alpha_j x_i \cdot x_j \\ &\text{s.t.} \sum_{i=1}^{n} y_i \alpha_i = 0, \\ &\alpha_i \geq 0. \quad (i = 1, 2, \ldots, n) \end{aligned} \tag{2}$$

Here α is the Lagrange multiplier. Given solution of Problem (2), α^*, we can get the solution of Problem (1), w^* and b^*, via formula:

$$w^* = \sum_{i=1}^{n} y_i \alpha_i^* x_i, \tag{3}$$

$$b^* = -\frac{1}{2}\left(\max_{y_i=-1} w^* \cdot x_i + \min_{y_i=1} w^* \cdot x_i\right). \tag{4}$$

Theoretical studies also prove that only those α_i corresponding to SVs are non-zero (Cristianini and Shawe-Talyor 2005). Here SVs refer to sample data closest to the optimal hyperplane just like the ones on l_{-1} and l_1 in Figure 2(b). Thus for an incoming sample, x, we have the classifier $f(x)$:

$$f(x) = \operatorname{sgn}\left(\sum_{x_i \in SV} y_i \alpha_i^* x_i \cdot x + b^* \right). \tag{5}$$

Although Equations (1)–(5) are valid only when the input data are linearly separated, in many cases, we can map the non-linear input into *a feature space* in which the image data are indeed linearly separated. The mapping, denoted by ϕ, is implicitly done via *a kernel function*, K:

$$K : \mathbf{R}^n \times \mathbf{R}^n \to \mathbb{R}, \quad (x_1, x_2) \to \phi(x_1) \cdot \phi(x_2). \tag{6}$$

Hence we get a more powerful classifier as long as we replace inner products in Equations (1)–(5) with kernel function K.

Furthermore, we always use a *soft margin* technique to handle practical noisy data. That is, we solve the dual of the following problem instead of Problem (2):

$$\begin{aligned} &\min_{\xi, w, b} \; K(w, w) + C \sum_{i=1}^{N} \xi_i^2 \\ &\text{s.t.} \quad y_i(K(w, x_i) + b) \geq 1 - \xi_i, \\ &(i = 1, 2, \ldots, n). \end{aligned} \tag{7}$$

Here ξ is the *slack vector* and C the *penalty factor* controlling the degree of fault-tolerance.

2.2. *Previous works on handling large-scale training sets*

The efforts on handling large training sets have long been made and appeared in the form of various SVM implementation techniques, among which the chunking method (Boser, Guyon, and Vapnik 1992), the Osuna algorithm (Osuna et al. 1997), SMO (Platt 1998) and SVM^{light} (Joachims 1999) are the most famous ones. These algorithms all iteratively optimise and update a *working set* (i.e. a small portion of the training data) to approximate the global solution. They mainly differ in the heuristics for updating the working set in each iteration. The detailed analysis and comparison of these methods are beyond the scope of this article, yet Joachims (1999) performs a series of experiments showing that SVM^{light} has a much better performance than SMO and chunking algorithms. More recently, Joachims (2006) proposes a cutting-plane algorithm that can train a classification SVM in linear time.

Another basic idea to handle large training sets is incremental learning. Syed, Liu, and Sung (1999) divide the entire training set into several parts. In the kth training process, it uses the kth part training data and the SVs obtained in the previous training processes. Although this incremental learning method is simple and can be implemented within the standard SVM learning framework, in practice, the classification performance is always compromised. Besides, many new algorithms are developed. For example, Cauwenberghs and Poggio (2001) and Diehl and Cauwenberghs (2003) propose a pure on-line incremental learning algorithm. Fung and Mangasarian (2002) propose a very efficient learning algorithm that can easily handle mega-level training sets. Although these new algorithms are promising, they are not mature yet. Their actual performances still need further assessment. Moreover, new algorithms are not economic from the standpoint of enterprises.

2.3. *Previous works on handling imbalanced training sets*

The problem of handling imbalanced training sets (i.e. imbalance learning) is a hot topic of current machine-learning community. Wu and Chang (2003) give a good example to show the impact of imbalanced training data on the result classifier. They also give two possible reasons for this impact: (1) in statistics, negative data (i.e. data of minority class) are always far from ideal boundary and (2) low negative SV ratio biases Equation (5) towards the positive class. Akbani et al. (2004) add the third possible reason – the weakness of soft-margin (i.e. C in Equation (7)).

An intuitive approach to handle imbalance is to sample the original training set. There are two basic sampling methods, *undersampling* and *oversampling*. Undersampling is to somehow reduce the negative data (Kubat and Matwin 1997), whereas oversampling is to somehow construct new negative data (Chawla, Bowyer, Hall, and Kegelmeyer 2002). An extreme case of undersampling is to use negative data only (Raskutti and Kowalczyk 2004). Stefanowski and Wilk (2008) propose three techniques based on the nearest neighbour rule (NNR) to amplify the impact of minority class.

Despite the sampling methods, many new algorithms are developed to handle imbalance. Briefly speaking, most of these algorithms try to push the learned hyperplane towards the positive side so that the impact of imbalance might be neutralised. Wu and Chang (2003) modify the kernel function. Akbani et al. (2004) use different penalty factors, C^+ and C^-, for different classes, respectively. Yan, Liu, Jin, and Hauptmann (2003) propose an ensemble learning method where the training set of each SVM is a balanced set consisting of the entire negative data and part of the positive data. Tang et al. (2009) minimise information loss in their undersampling process and speeds up SVM learning by extracting less SVs. Most recently, ensemble methods are also used to build more powerful classifiers. For instance, Wang and Japkowicz (2010) investigate the usage of the boosting method, and Błaszczynski, Deckert, Stefanowski, and Wilk (2010) propose an integration of selective pre-processing (Stefanowski and Wilk 2008) and Ivotes ensemble. As pointed out by Yan et al. (2003) and Tang et al. (2009), some imbalance learning methods are not efficient enough. They need longer training time, which may become unbearable in real-world applications. In addition, they are not economic.

3. Our approach of compressing and balancing

As mentioned above, we always have a very large imbalanced training dataset and we want a training approach that is effective, efficient and economic. With such consideration, we would prefer to perform data-level processing rather than develop new algorithms because new algorithms are neither economic nor mature enough to support enterprise applications. Moreover, among data pre-processing methods, oversampling is obviously infeasible because our training dataset is already very large. In contrast, undersampling methods are promising since they can compress and balance the training dataset at the same time.

However, there are two potential problems with undersampling (Akbani et al. 2004):

(1) *Distribution disturbance*. Once we perform undersampling on the negative class, we change the sample distribution, which is essential to statistical learning methods such as SVM.
(2) *Information loss*. Once we perform undersampling, we probably throw away valid data which contain valuable information. This information loss may cause a descent of overall performance.

Fortunately, in our application, neither of the two problems applies. First, as mentioned in Section 1, the ratio of spams to normal emails is highly imbalanced. The most important reason is that our sample data are not drawn randomly. Spams are always easily collected, but to collect normal emails is not that easy due to privacy issues. In other words, the sampling mechanism is biased towards the negative class and undersampling is a way to neutralise this bias. Second, it is worthy to note that many spams in our training dataset are duplicated or highly similar. For example, a spam may be sent many times to a honeypot within several hours or days. Eliminating these redundant or highly similar spams might cause limited information loss but can significantly reduce the dataset and bring improvement in performance.

Unlike other one-sided sampling methods or hybrid sampling methods, we perform undersampling on negative data too. This is because the negative dataset, although relatively small, is still large in absolute size and it is unnecessary to keep them all.

Figure 3 is a sketch of our basic idea. Figure 3(a) illustrates the original very large imbalanced dataset, in which there are many clutters of data samples either on the negative side or on the positive side. As shown in Figure 3(b), to compress and balance the original dataset, we use different thresholds, indicated by the radii of the circles, to deal with the two data classes. Data samples within a clutter (i.e. a circle) are considered to be redundant or highly similar and thus we merely keep one of them and drop all the others. Then the compressed and balanced dataset is obtained, as shown in Figure 3(c).

According to Equations (3)–(5), SVs x_i play an important role in an SVM classifier. Therefore, much like what Syed et al. (1999) does, we preserve all these SVs during the above compressing and balancing process. The existence of other non-SVs overcomes the deficiency in performance to some extent.

In more detail, our proposed approach involves the following problems:

- *Data representation*. How to represent an email as a data sample vector?
- *Similarity measure*. How to define the similarity between two emails?
- *Undersampling algorithm*. How to perform undersampling using the similarity measure?

3.1. *Data representation*

To represent emails in the form of vectors, first, we need to determine a feature set from all possible characteristics of emails. In our application, the selected features of emails include tokens, formats, hit-rules, etc., and are primitive units for email analysis. For example, the regular expression,

$$\text{https}\backslash : \backslash/\backslash/[0-9]+\backslash.[0-9]+\backslash.[0-9]+\backslash.[0-9]$$

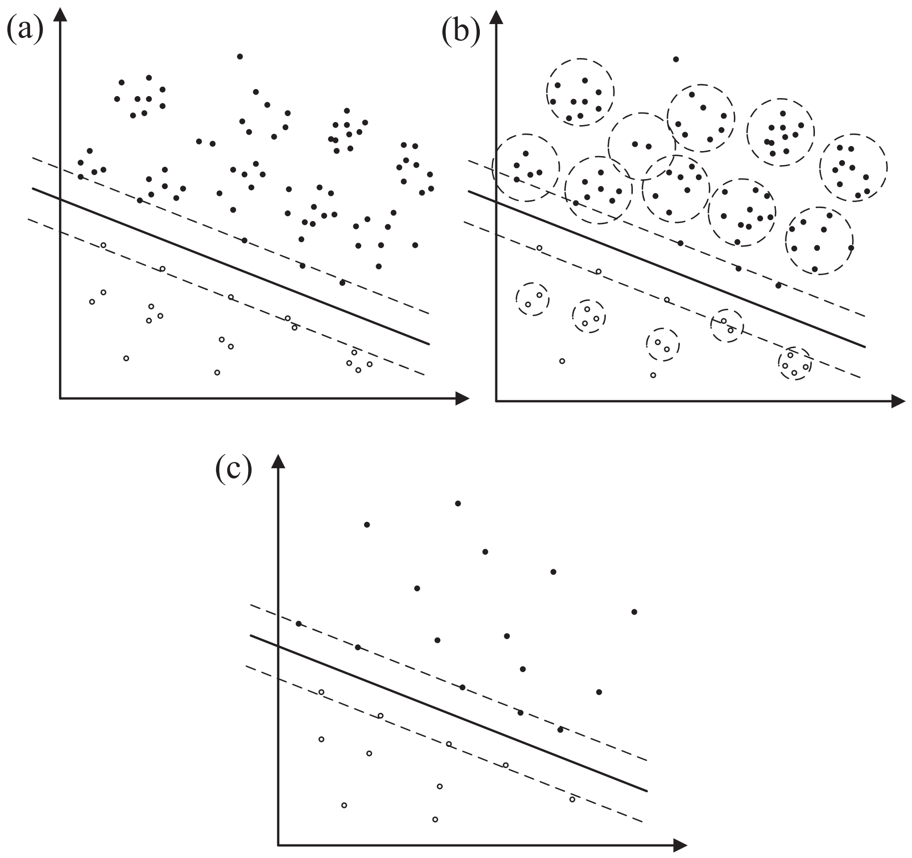

Figure 3. Our basic idea of compressing and balancing: (a) the original dataset and its SVM classification; (b) undersampling with different thresholds and (c) the compressed and balanced dataset.

judges whether there is any secure web address in the email, and another feature,

[a\@] + d + u + [li1\|\!] + t+

judges whether the word ‘adult’ or its variations appears. Secure web may be evidence of normal emails, whereas the word ‘adult’ frequently appears in spams. Thus, the features are not only designed for spam tendencies but also for normal emails.

Unsurprisingly, there are tens of thousands of features like the two listed above, all of which are tokens, regular expressions, or domain characteristics determined by human experts. These features are all binary, thus all data vectors are high-dimensional binary vectors. High-dimensional data vectors may cause serious decline of SVM classification performance and negatively impact the optimisation of solutions. Therefore, we need to reduce the number of features to construct feature space. To do this, the information gain of each feature is computed and features with the highest information gains are selected.

Let S^+ be the set of all positive sample vectors and S^- the set of all negative sample vectors. Thus $S = S^+ \cup S^-$ is the entire sample set. For any feature f, $S^i_{f,v}$

denotes the set $\{x \in S^i | x_f = v\} \cdot S_{f,v} = S^+_{f,v} \cup S^-_{f,v}$. The information gain of feature A, $G(f)$, is then calculated as:

$$G(f) = I(S^+, S^-) - \sum_v \frac{(|S_{f,v}|)}{|S|} I(S^+_{f,v}, S^-_{f,v}),$$

Where the expected information, I, is defined as

$$I(S_1, S_2) = -\sum_i \frac{|S_i|}{|S_1| + |S_2|} \log_2 \frac{|S_1|}{|S_1| + |S_2|}.$$

Using information gain, we finally get a much more concise representation of emails, the length of which seldom exceeds a few hundreds.

3.2. *Similarity measure*

Given two email vectors, u and v, to measure the similarity between them can be considered as an information retrieval (IR) task of query matching. Incoming emails are queries. We check the current email pool to see if there is any match. If there is, then this incoming email is redundant and thus discarded. Otherwise, the incoming email is added to current pool. We use the cosine similarity:

$$\text{sim}(u, v) = \cos(u, v) = \frac{u \cdot v}{\|u\| \, \|v\|}. \tag{8}$$

Obviously, $0 \le \text{sim}\ (u,v) \le 1$, and the larger sim (u, v) is, the more similar u and v are, vice versa. Unlike the standard IR methods, we do not calculate *term frequency* or *inverse document frequency* (Raghavan 1997) because our features are in instances of general regular expressions.

In fact, we do find advantages of this binary data representation. It is worthy to note that, most of the features are designed according to spams. That means spams tend to have more 1s in their vector representation in general. For example, see the two features listed in Section 3.1. The emails containing a numerical hyperlink or the word ‘adult’ are more likely to be spams. Therefore in the sample space, spam vectors tend to be distributed far away from the coordinate origin, while the cosine similarity, Equation (8), tends to cluster these spams. As shown in Figure 4, the distance between u_1 and v_1 equals the distance between u_2 and v_2, but, with a proper threshold, the vectors in the inner spam area are more likely to be considered similar since they have a sharper included angle.

This bias is essentially good for balancing our SVM training dataset. Let $u, v \in \{0, 1\}^d$ be two vectors with m and n 1s respectively. Without loss of generality, we can assume that $m = n - k$ $(k \ge 0)$ and $m + n \ge d$, then we have

$$2n - k - d \le u \cdot v \le n - k = m.$$

Thus, the lower bound and upper bound of sim (u, v) denoted by $\underline{\text{sim}}(u, v)$ and $\overline{\text{sim}}\ (u, v)$ should be

$$\underline{\text{sim}}(u, v) = \frac{2n - k - d}{\sqrt{n(n - k)}} \ge 0, \tag{9}$$

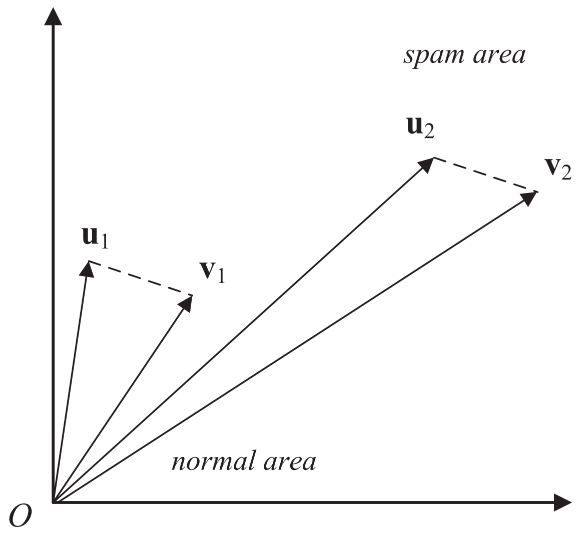

Figure 4. The bias of cosine similarity. $|u_1 - v_1| = |u_2 - v_2|$ but $((\widehat{u_1, v_1}) > (\widehat{u_2, v_2})$ thus with a proper threshold θ, u_2 and v_2 are similar while u_1 and v_1 are not.

and

$$\overline{\mathrm{sim}}(u, v) = \sqrt{\frac{n-k}{n}} \leq 1. \tag{10}$$

It is easy to find that both $\underline{\mathrm{sim}}(u, v)$ and $\overline{\mathrm{sim}}(u, v)$ are increasing functions with respect to $n \in \left[\frac{d+k}{2}, d\right]$. Then the following two formulas hold:

$$\lim_{n \to d} \underline{\mathrm{sim}}(u, v) = \frac{d-k}{d},$$

and

$$\lim_{n \to \frac{d+k}{2}} \overline{\mathrm{sim}}(u, v) = \frac{d-k}{d+k}.$$

Note that

$$\lim_{n \to d} \underline{\mathrm{sim}}(u, v) \geq \lim_{n \to \frac{d+k}{2}} \overline{\mathrm{sim}}(u, v). \tag{11}$$

In a special case, we may assume that $u \cdot v = n - k$ where k is fixed, thus $\|u - v\|_2 \equiv \sqrt{k}$. If we set a threshold between $\lim_{n \to d} \underline{\mathrm{sim}}(u, v)$ and $\lim_{n \to \frac{d+k}{2}} \overline{\mathrm{sim}}(u, v)$, then, although with the same Euclidean distance, emails with more spam characteristics (i.e. more 1s) are judged similar while the ones with less spam characteristics are not.

3.3. *Double-undersampling*

The SVM training in our application is a dynamic process as shown in Figure 1. In each period i, the training dataset T_i is obtained from two sources, the previous training dataset

Table 1. Double-undersampling.

Input	Dataset K containing both positive data and negative data K^*
Output	the compressed and balanced subset of K
1	Train SVM using K and get the SV set, Vs
2	Divide K into positive set K^+ and negative set K^-
3	Divide Vs into positive set Vs^+ and negative set Vs^-
4	For $l \in \{+, -\}$ do Steps 5–6
5	$K^l \leftarrow compress(K^l, k^l)$ *Here k^l is the threshold for cosine similarity*
6	$K^l \leftarrow merge(K^l, V_s^l, k^l)$ *Merge but remove redundant or highly similar data*
7	$K^* \leftarrow merge(K^+, K^-, k)$ *The threshold k is used to handle ambiguous data*

T_{i-1} and the incoming raw dataset D_i. To get a compressed and balanced dataset, we have developed a double-undersampling procedure which is shown in Table 1. Briefly speaking, our double-undersampling procedure resembles the incremental learning method used in Syed et al. (1999) and overcomes its performance reduction via supplying more data.

There are three thresholds in the double-undersampling procedure, k^+, k^- in Step 5 and k in Step 7. Among them, k is relatively easy to determine since its function is to identify and remove ambiguous data. Here, by the term *ambiguous data* we mean those data pairs like $(x, +1)$ and $(x, -1)$, i.e. vectors that have the same or highly similar contents but different labels. Therefore, we just pick a small positive value for k.

In comparison, k^+ and k^- are not as trivial as k and, at the same time, are essential to the overall performance of double undersampling. In practice, we employ a stepwise searching process to determine the optimal values of the two thresholds based on some heuristics.

Let us write $|K^l| = n^l$ and $|V_s^l| = v^l$ and where $l \in \{+, -\}$. If the desired size after compressing and balancing is q, then we can estimate the reduction rates r^+ and r^- by

$$(1 - r^+)n^+ + v^+ \approx (1 - r^-)n^- + v^- \approx \frac{q}{2},$$

because in the final K^* it is reasonable to keep approximately the same number of positive data and negative data. That is

$$r^l = 1 - \frac{q}{2n^l} - \frac{v^l}{n^l}(l \in \{+, -\}). \tag{12}$$

We cannot directly use r^l to determine k^l because the values of k^l should be relevant to the nature of the datasets. For example, in order to achieve the same reduction rate r^l, if the dataset is sparse, we should set k^l a bit smaller; while if the dataset is compact, we should then set it a bit larger. Nevertheless, r^l is a valuable heuristics for determining k^l. A basic qualitative relation between r^l and k^l is, the larger the former is, the smaller the latter should be. We thus determine an empirical range $[k_S^l, k_e^l]$ according to r^l, and then search in the range for the optimal k^l.

Using double-sampling described above, we can then get the training dataset of the ith period, $T_{\rm i}$, in a two-phase manner. First, we compress and balance incoming dataset D_i, getting D_i^*. Then, we compress and balance the union set of T_{i-1} and D_i^* to get T_i.

And since we carefully choose the thresholds, the model trained from T_i is always satisfactory.

4. Empirical studies

We design and perform some experiments to justify the effectiveness and efficiency of our approach. All experiments are conducted on a machine with an Intel® Xeon 3.00 GHz CPU and 16.0 G memory. The SVM used in the experiments is SVM$^{\text{light}}$ (Joachims 1999) commercial version (v 6.0), implemented by its designer Joachims.

4.1. *Dataset and methodology*

We do the experiments within the company product release cycle as illustrated in Figure 1. All data, either for training or for testing, are real-time data sharing the same sources with those used in real anti-spam products. The key concern of our experiments is to demonstrate the effectiveness and efficiency of the double-undersampling process. Ideally, the model trained from double-sampled (i.e. compressed and balanced) dataset should be as good as, if not better than, the one trained from original very large imbalanced dataset.

We use a training dataset of size 64,880 and testing dataset of size 280,904. The training dataset contains 5416 normal emails and 59,464 spams. The training and testing sets are sampled uniformly at random from a large email pool. There is no overlap between the training and testing datasets. The testing set is organised by timestamp (1 month by 1 month), but we mask the timestamps here due to business reasons.

The testing dataset is naturally divided into 21 subsets since these testing data are collected in different topic domains, from different sources and at different times in the past. Among the 21 testing datasets (subsets), there are 14 negative datasets with a total number of 60,289 normal emails and 7 positive datasets with a total number of 220,615 spams.

Furthermore, we also test our approach using a very large training dataset of size 2,818,343. The size of the corresponding testing dataset is 320,176. This testing dataset shares the same sources with the above 21 testing datasets.

4.2. *Performance measure*

In the machine learning community, there are four essential values to reflect the performance of a classifier: false positive (FP), true positive (TP), false negative (FN) and true negative (TN). Here FP is the number of negative data that are classified as positive, and the other three values can be interpreted analogously. In conventional machine learning tasks, accuracy is often used as the final measure (Joachims 2006):

$$Acc = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}^*}$$

However, accuracy might be bogus when the training dataset is highly imbalanced because, for example, if a ‘classifier’ classifies every data as negative it can still reach a high accuracy but this ‘classifier’ is obviously useless.

In fact, accuracy values in classifying positive and negative data should be considered separately. That is, consider *true positive rate*

$$Acc_{+} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

and *true negative rate*

$$Acc_{-} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

instead of overall accuracy Acc.

Furthermore, we use the weighted geometric mean of Acc_{+} and Acc_{-} to evaluate the overall performance of our approach:

$$G = \sqrt[\alpha+\beta]{Acc_{-}^{\alpha} \cdot Acc_{+}^{\beta}},$$

where the non-negative integers α and β reflect the different importance of Acc_{+} and Acc_{-}. In fact, misclassifying a normal email is always much worse than misclassifying a spam. α and β are set to 5 and 1, respectively, in our experiments.

4.3. *The experimental results on the 64,880-dataset*

We conducted nine experiments using different reduction rates to compress and balance the original dataset. Detailed information about the training dataset used in the experiments is shown in Table 2. There NR is the original large imbalanced training dataset.

The results are shown in Table 3. As previously introduced, the testing data are naturally divided into positive and negative datasets. For negative datasets, our concern is their false positive rates, whereas for positive datasets, we care more about their true positive rates. The top part of Table 3 is false positive rates for normal testing samples. We feed the classifier all normal emails and see how many are classified as spam. The lower part is the true positive rates, which are obtained by feeding the classifier all spam emails and seeing how many are really classified as spam. We hope to detect more spams while make less mistakes in classifying normal emails.

The values shown in Table 3 can be used directly to calculate Acc_{+} and Acc_{-}. To calculate the G-measure in Equation (13), the summary results are listed in Table 4. We can see that the reduced training datasets have a distinct advantage over the original one, and then the increasing rates of G_{5}^{1} range from +0.66% to +0.78%. But the differences between different reduced training datasets are relatively small (within $\pm$0.12%). This indicates that we can better performance by compressing and balancing the original dataset and we need not worry about performance descent when high reduction rates are used.

Table 2. Training dataset in our experiments.

	NR	R1	R2	R3	R4	R5	R6	R7	R8	R9
Size	64,880	10,570	45,939	40,366	13,976	15,506	17,051	17,064	18,532	20,892
Reduction	N/A	83.71%	29.19%	37.78%	78.46%	76.10%	73.72%	73.70%	71.44%	67.80%

Table 3. False positive rates and true positive rates.

Testing dataset	Type	NR (%)	R1 (%)	R2 (%)	R3 (%)	R4 (%)	R5 (%)	R6 (%)	R7 (%)	R8 (%)	R9 (%)
20 KN	Normal (−)	1.30	1.64	1.63	1.63	1.64	1.64	1.64	1.63	1.63	1.64
LF News		3.60	3.26	3.09	3.02	3.14	3.15	3.14	3.12	3.10	3.10
LF News 07		4.18	3.95	3.87	3.76	3.96	3.90	3.87	3.88	3.87	3.86
LF News 08		4.42	3.88	3.80	3.75	3.88	3.84	3.81	3.75	3.77	3.77
LF Normal		7.24	6.57	6.73	6.73	6.57	6.57	6.40	6.57	6.57	6.57
LF Norm 07 LF Norm 08 Normal		4.00	3.94	4.00	3.72	4.00	4.00	4.00	4.00	3.94	4.05
LF Norm 09		2.88	2.77	3.20	2.99	3.09	3.09	3.09	3.09	3.09	3.09
LF Norm 11		4.93	4.73	5.41	5.02	5.41	5.41	5.41	5.41	5.41	5.41
LF Norm 12		8.77	8.45	8.98	8.29	8.66	8.66	8.77	8.77	8.77	8.88
LF Norm 200901		8.52	8.23	8.80	8.73	8.45	8.52	8.52	8.59	8.66	8.80
LF Norm 200902		11.29	10.87	11.23	11.17	11.17	11.11	11.11	11.11	11.05	11.23
LF Norm 200904											
OpusOne		9.93	10.50	10.22	9.93	10.41	10.41	10.31	10.31	10.31	10.22
LF Spam 08	Spam (+)	9.67	9.97	10.27	9.67	9.97	9.97	9.97	9.97	9.97	9.97
LF Spam 10		4.63	1.85	3.70	3.70	3.70	3.70	3.70	3.70	3.70	3.70
Spam 0810		55.38	58.48	58.13	57.81	58.41	58.36	58.34	58.24	58.24	58.15
LF Spam 200901 Spam		67.24	68.79	68.74	68.61	68.83	68.84	68.79	68.81	68.78	68.74
LF Spam 200902		45.01	48.44	48.48	48.67	48.48	48.51	48.48	48.45	48.44	48.43
Spam 200903		76.05	77.89	77.59	77.47	77.71	77.79	77.79	77.68	77.66	77.63
LF Spam 200904		73.26	75.12	74.73	74.70	74.94	74.95	74.94	74.94	74.91	74.85
		58.12	61.90	61.50	61.26	61.14	61.14	61.23	61.27	61.27	61.34
		40.22	43.75	44.23	45.23	43.99	44.08	44.02	44.05	44.08	44.20

4.4. *Towards mega-level training dataset*

We also conducted some experiments on a very large (mega level) training dataset, of which the size is 2,818,343. The original training dataset took SVM$^{\text{light}}$ about 1 week's time to get the final indices of $Acc_{-} = 93.24\%$ and $Acc_{+} = 77.80\%$, and thus $G_5^1 = 0.9047$. Then, we reduced the training dataset to size 861,195, using a reduction rate of 69.44%. The reduction process took about 24 h. Within another 3 h, SVM$^{\text{light}}$ trained a classifier of which the indices were $Acc_{-} = 94.11\%$, $Acc_{+} = 76.30\%$ and $G_5^1 = 0.9088$. Obviously, much time was saved and a better performance was achieved.

5. Discussion and future work

Although our approach shows a good performance in experiments, there are still many issues to be discussed and further studied.

First, there are some alternatives for the cosine measure (Equation (8)), among which the following Tanimoto coefficient has been widely used in dealing with binary data vectors:

$$T(u, v) = \frac{u \cdot v}{|u|^2 + |v|^2 - u \cdot v}.$$

Table 4. Acc_+, Acc_- and the G-measure of each experiment.

	NR	R1	R2	R3	R4	R5	R6	R7	R8	R9
TP (/64,880)	58,053	58,078	58,080	58,061	58,070	58,079	58,083	58,092	58,097	58,088
Acc_-	96.30%	96.34%	96.34%	96.31%	96.32%	96.34%	96.35%	96.36%	96.37%	96.35%
TN (/220,615)	138,360	144,715	144,327	143,926	144,551	144,556	144,496	144,421	144,387	144,294
Acc_+	62.71%	65.59%	65.42%	65.23%	65.52%	65.52%	65.49%	65.46%	65.44%	65.40%
G_5^1	0.8966	0.9036	0.9032	0.9025	0.9033	0.9034	0.9035	0.9035	0.9035	0.9032

Actually, even Euclidean distance can be used as the similarity measure. However, despite of its bias discussed in Section 3.2, cosine measure is worthy to consider because of the nature of spam filtering. Although data features introduced in this article are all regular expressions, most of them are rules about keywords. With a small extension, we can record the TFIDF of these words in our data representation and then follow the standard IR procedure to calculate the similarity between *queries* and *documents*. Anyway, we are planning a series of experiments using different ways to measure the similarity.

Second, inspired by Figure 3, it is a sound idea to compress and balance the dataset using clustering methods such as *k*-Means (Han and Kamber 2001). In this way, Equation (12) could be good estimation of the numbers of classes. We have tried several simple clustering methods including *k*-Means and *k*-Medoids. The results showed that clustering methods consumed more time than IR methods did, but their final indices (Acc_{-}, Acc_{+} and G_5^1) are identical in the sense of statistics. In other words, the preliminary results showed that the benefits of clustering methods did not match their cost. Nevertheless, we believe it advisable to find or design appropriate clustering algorithms for our anti-spam application.

Finally, in the machine-learning community, there are also many other ways to evaluate the effectiveness of an approach – for example, the *F*-measure. However, beyond various index definitions, we believe that accuracy values in classifying each class are essential. In Section 4.3, we can see that after reduction the accuracy values in classifying both classes are increased. This is independent of what overall measure is used. Yet the geometric mean measure, first proposed by Kubat and Matwin (1997), is indeed widely used. In this article, we augment this measure with weights to put emphasis on negative accuracy.

6. Conclusion

In this article, we propose an effective, efficient and economic approach to handle very large imbalanced training datasets in an enterprise application of anti-spam. The basic idea is to remove the redundant and highly similar data samples from the original dataset. Within this reduction process, different reduction rates are used in different classes so that the final dataset is much smaller and much more balanced and thus is more suitable for SVM learners such as $\text{SVM}^{\text{light}}$. In the real enterprise product release cycle, the key issue is to keep or even improve the classifying performance while reducing the training dataset. We guarantee the performance by keeping all SVs in the reduced dataset as well as other auxiliary data obtained in an IR-style procedure.

The experimental results show that our approach is promising. With a reduction rate of nearly 80%, the G_5^1 value gets an increase of about 0.8%. In addition, perhaps the most significant benefit lies in the fact that much time is saved. As we all know, time is really important for the development of an enterprise.

Acknowledgements

The authors thank Grant Chen for his great effort in carrying out the experiments and Jon Oliver at Trend Micro Inc. for the grant to publish the article.

Note

1. By the convention of spam filtering, in this article, a sample is called positive if it is a spam and negative otherwise. Note that the positive class is thus the majority class and negative class the minority class.

References

Akbani, R., Kwek, S., and Japkowicz, N. (2004), 'Applying Support Vector Machines to Imbalanced Datasets', in *Proceedings of the 15th European Conference on Machine Learning (ECML)*, 20–24 September, Pisa, Italy, pp. 39–50.

Anti-Spam Center of ISC. (2008), '4Q Anti-spam Investigation Report', In Chinese.

Błaszczynski, J., Deckert, M., Stefanowski, J., and Wilk, S. (2010), 'Integrating Selective Pre-processing of Imbalanced Data with Ivotes Ensemble', in *Proceedings of the 7th International Conference on Rough Sets and Current Trends in Computing*, 28–30 June, Warsaw, Poland, pp. 148–157.

Boser, B.E., Guyon, I.M., and Vapnik, V.N. (1992), 'A Training Algorithm for Optimal Margin Classifiers', in *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, 27–29 July, Pittsburgh, PA, pp. 144–152.

Carreras, X., and Marquez, L. (2001), 'Boosting Trees for Anti-spam Email Filtering', in *Proceedings of the European Conference Recent Advances Natural Language Processing*, 5–7 September, Tzigov Chark, Bulgaria, pp. 58–64.

Cauwenberghs, G., and Poggio, T. (2001), 'Incremental and Decremental Support Vector Machine Learning', in *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*, Denver, CO, USA, pp. 409–415.

Chawla, N.V., Bowyer, K.W., Hall, L.O., and Kegelmeyer, W.P. (2002), 'SMOTE: Synthetic Minority Over-sampling Technique', *Journal of Artificial Intelligence Research (JAIR)*, 16, 321–357.

Cristianini, N., and Shawe-Talyor, J. (2005), *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods* (1st ed.), Beijing, China: China Machine Press.

Diehl, C.P., and Cauwenberghs, G. (2003), 'SVM Incremental Learning, Adaptation and Optimization', in *Proceedings of the International Joint Conference on Neural Networks*, 20–24 July, Portland, OR, pp. 2685–2690.

Drucker, H., Wu, D., and Vapnik, V.N. (1999), 'Support Vector Machines for Spam Categorization', *IEEE Transactions on Neural Networks*, 20, 1048–1054.

Fallows, D. (2003), *'Spam: How it is Hurting Email and Degrading Life on the Internet', Pew Internet and American Life Project (PIP)*, Washington, DC: PEW Research Center.

Fung, G., and Mangasarian, O.L. (2002), 'Incremental Support Vector Machine Classification', in *Proceedings of the 2nd SIAM International Conference on Data Mining (SDM)*, 11–13 April, Arlington, VA, pp. 247–260.

Han, J., and Kamber, M. (2001), *Data Mining: Concepts and Technologies*, Beijing, China: Higher Education Press and Morgan Kaufmann Publishers.

Joachims, T. (1999), 'Making Large-scale SVM Learning Practical', in *Advances in Kernel Methods – Support Vector Learning*, eds. B. Schölkopf, C. Burges and A. Smola, Cambridge, MA: MIT Press, pp. 169–184.

Joachims, T. (2006), 'Training Linear SVMs in Linear Time', in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 20–23 August, Philadelphia, PA, pp. 217–226.

Jones, S., and Fox, S. (2009), *'Generations Online in January 2009', Pew Internet and American Life Project (PIP)*, Washington, DC: PEW Research Center.

Kubat, M., and Matwin, S. (1997), ‘Addressing the Curse of Imbalanced Training Sets: One-sided Selection’, in *Proceedings of the 14th International Conference on Machine Learning (ICML)*, 8–12 July, Nashville, TN, pp. 179–186.

Osuna, E., Freund, R., and Girosi, F. (1997), ‘An Improved Training Algorithm for Support Vector Machines’, in *Proceedings of the 1997 IEEE Workshop*, 24–26 September, Amelia Island, FL, pp. 276–285.

Platt, J.C. (1998), ‘Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines’, Technical Report MSR-TR-98-14, Microsoft Research.

Raghavan, P. (1997), ‘Information Retrieval Algorithms: A Survey’, in *Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 5–7 January, Philadelphia, PA, pp. 11–18.

Raskutti, B., and Kowalczyk, A. (2004), ‘Extreme Rebalancing for SVMs: A Case Study’, *SIGKDD Explorations*, 6, 60–69.

Sahami, M., Dumais, S., Heckerman, D., and Horvitz, E. (1998), ‘A Bayesian Approach to Filtering Junk E-mail’, in *Proceedings of the AAAI Workshop Learning for Text Categorization*, 26–30 July, Madison, WI, pp. 55–62.

Stefanowski, J., and Wilk, S. (2008), ‘Selective Pre-processing of Imbalanced Data for Improving Classification Performance’, in *Proceedings of the 10th International Conference on Data Warehousing Knowledge Discovery*, 2–5 September, Turin, Italy, pp. 283–292.

Syed, N.A., Liu, H., and Sung, K. (1999), ‘Incremental Learning with Support Vector Machines’, in *Proceedings of the IJCAI’99 Workshop on Support Vector Machines*, 31 July–6 August, Stockholm, Sweden.

Tang, Y., Zhang, Y.Q., Chawla, N.V., and Krasser, S. (2009), ‘SVMs Modeling for Highly Imbalanced Classification’, *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, 39, , 281–288.

Vapnik, V.N. (1998), *Statistical Learning Theory*, New York, NY: John Wiley and Sons Inc.

Vapnik, V.N. (1995), *The Nature of Statistical Learning Theory*, Berlin, German: Springer-Verlag.

Wang, B.X., and Japkowicz, N. (2010), ‘Boosting Support Vector Machines for Imbalanced Data Sets’, *Knowledge and Information Systems*, 25, 1–20.

Wikipedia. (2009), Spamtrap, http://en.wikipedia.org/wiki/Spamtrap

Wu, G., and Chang, E.Y. (2003), ‘Class-boundary Alignment for Imbalanced Dataset Learning’, in *Proceedings of the 20th ICML Workshop on Learning from Imbalanced Datasets*, 21–24 August, Washington, DC, pp. 49–56.

Wu, G., and Chang, E.Y. (2005), ‘KBA: Kernel Boundary Alignment Considering Imbalanced Data Distribution’, *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 17, 786–795.

Yan, R., Liu, Y., Jin, R., and Hauptmann, A. (2003), ‘On Predicting Rare Classes with SVM Ensembles in Scene Classification’, in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, 6–10 April, Hong Kong, pp. 21–24.