A Multiclass Classification Framework for Document Categorization

Qi Qiang[1] and Qinming He[1,2]

[1] College of Computer Science, Zhejiang University, Hangzhou 310027, China
qiangqi@yahoo.com
[2] Ningbo Institute of Technology, Zhejiang University, Ningbo 315100, China

Abstract. With a great amount of textual information are available on the Internet and corporate intranets, it has become a necessary to categorize large documents. As we known, text classification problem is representative multiclass problem. This paper describes a framework, which we call Strong-to-Weak-to-Strong (SWS). It transforms a "strong" learning algorithm to a "weak" algorithm by decreasing its iterative numbers of optimization while preserving its other characteristics like geometric properties and then makes use of the kernel trick for "weak" algorithms to work in high dimensional spaces, finally improves the performances of text classification. We analyzed the particular properties of learning with text and identified why this approach is appropriate for this task. Empirical results show that our approach is competitive with the other methods.

1 Introduction

Automated text classification, the supervised learning assignment of labeling free text document to predefined categories based on their content, is a significant component in many data mining and knowledge discovery tasks. Many algorithms for the multiclass problems have been developed recently. One directly considers all data in one optimization formulation. However a more general method is to reduce a multiclass problem to multiple binary problems [1].

In [2] Dietterich and Bakiri described a unifying method for reducing multiclass problems to multiple binary problems based on error correcting output codes. However there might be strong statistical correlations between the resulting classifiers. The design problem of output coding has been introduced [3]. One can regard the design problem as a constrained optimization problem. However this method is extremely time-consuming and has too much variables to solve effectively.

Recently a robust Minimax classifier where the probability of correct classification of future data should be maximized has been provided [4]. No further assumptions are made with respect to the each two class-conditional distributions. The minimax problem can be interpreted geometrically as minimizing the maximum of the Mahalanobis distances to the two classes. "Kernelization" version is also available.

An important feature of the probability machine is that a worst-case bound on the probability of misclassification of future data is always obtained explicitly. We use this probability to build a heuristic algorithm and then solve the design problem of output

H. Bunke and A.L. Spitz (Eds.): DAS 2006, LNCS 3872, pp. 474 – 483, 2006.

coding more effectively, while overcoming the limitation that the previous methods can be implemented only when given a set of binary classifiers.

Section 2 reviews the standard feature vector representation of text. Section 3 briefly states the output codes framework for multiclass categorization problems. Section 4 reviews the binary minimax machine and introduces resample for robust estimation for mean and covariance matrix. Section 5, the main part of the article, presents new algorithm in detail. In section 6, we report the experimental results. Finally, section 7 presents conclusions and discussion of future directions.

2 Text Categorization

The goal of text categorization is the classification of text into a fixed number of predefined categories. Using machine learning, the objective is to learn classifiers from examples which perform the category assignments automatically. This is a supervised learning problem.

The first step in text categorization is to transform text, which typically are strings of characters, into a representation suitable for the learning algorithm and the classification task. Text Analysis researches suggest that word stems work well as representation units and that their ordering in a text is of minor importance for many tasks. This leads to an attribute-value representation of text. Each distinct word corresponds to a feature, with the number of times word occurs in the text as its value. This representation scheme leads to very high-dimensional feature spaces containing 10000 dimensions and more. Many have noted the need for feature selection to make use of conventional learning methods possible, to improve generalization accuracy, and to avoid overfitting. Following the recommendation of [5], the information gain criterion will be used in this paper to select a subset of features.

3 Design of Output Codes

In this section we briefly review the method for designing of output codes [3]. Let $S = \{(x_1, y_1), ..., (x_m, y_m)\}$ be a set of m training examples where each instance x_i belongs to a domain χ. We assume without loss of generality that each label y_i is an integer from the set $\Upsilon = \{1, ..., k\}$. A multiclass classifier is a function $H : \chi \to \Upsilon$ that maps an instance x into an element $y \in \Upsilon$. An output codes M is a matrix of size $k \times l$ over $\mathbb{R}$ where each row of M corresponds to a class $y \in \Upsilon$. Then different binary classifiers $h_1, ..., h_l$ can be yielded. We denote the vector of predictions of these classifiers on an instance x as $\bar{h}(x) = (h_1(x), ..., h_l(x))$.

We denote the r th row of M by $\overline{M}_r$. Given an example x we predict the label y for which the row $\overline{M}_y$ is the "closest" to $\bar{h}(x)$. Naturally we can perform the calculations in some high dimensional inner-product space Z using a transformation $\bar{\phi} : \mathbb{R}^l \to Z$ and use a general notion for closeness, then define it through an inner-product function $K : \mathbb{R}^l \times \mathbb{R}^l \to \mathbb{R}$, which satisfies Mercer conditions [6]. The higher the value

of $K(\bar{h}(x), \overline{M}_r)$ is, the more confident we are that r is the correct labels of x according to the classifiers $\bar{h}(x)$. Thus $H(x) = \arg\max_{r \in \Upsilon}\{K(\bar{h}(x), \overline{M}_r)\}$.

Let $[\![\alpha]\!]$ be 1 if the predicate α holds and 0 otherwise, $\delta_{i,j}$ equals 1 if $i = j$, 0 otherwise. Denote by $b_{i,r} = 1 - \delta_{y_i,r}$. We define the 2 norm of a matrix M and introduce slack variables ζ_i. Then the problem of finding a good matrix M can be stated as the following optimization problem:

$$L(M, \zeta, \eta) = \frac{1}{2}\beta \sum_r \left\| \overline{M}_r \right\|_2^2 + \sum_{i=1}^{m} \zeta_i + \sum_{i,r} \eta_{i,r} \left[K(\bar{h}(x_i), \overline{M}_r) - K(\bar{h}(x_i), \overline{M}_{y_i}) - \zeta_i + b_{i,r} \right] \quad (1)$$
$$subject\ to:\ \forall i, r\ \ \eta_{i,r} \geq 0$$

for some constant $\beta \geq 0$ and $\eta_{i,r} \geq 0$. Let $\bar{1}_i$ be the vector with all components zero, except for the i th component which is equal to one, and let $\bar{1}$ be the vector whose components are all one. We can denote by $\bar{\gamma}_{i,r} = \bar{1}_{y_i} - \eta_{i,r}$.

Finally, the classifier $H(x)$ can be written in terms of the variable γ as:

$$H(x) = \arg\max_r \left\{ \sum_i \gamma_{i,r} K(\bar{h}(x), \bar{h}(x_i)) \right\} \quad (2)$$

However solving optimization problem (**1**) is time-consuming. In this paper our algorithm solves this optimization problem heuristically.

4 A Probability Machine

In this section we introduce a minimax probability machine (MPM) [4] proposed by Lanckriet et al., which tries to minimize the probability of misclassification of data.

Let x and y model data from each of two classes in a binary classification problem, with means and covariance matrices given by (x, Σ_x) and (y, Σ_y), respectively. We wish to determine a hyperplane $F(a,b) = \{z \mid a^T z = b\}$, where $a \in \mathbb{R}^n \setminus \{0\}$ and $b \in \mathbb{R}$ which separates the two classes of points with maximal probability with respect to all distributions having these mean and covariance matrices. This is expressed as:

$$\begin{aligned} \max_{\theta, a \neq 0, b}\ \theta\ \ s.t\quad & \inf_{x \sim (x, \Sigma_x)} \Pr\{a^T x \geq b\} \geq \theta \\ & \inf_{y \sim (y, \Sigma_y)} \Pr\{a^T y \leq b\} \geq \theta \end{aligned} \quad (3)$$

Where the notion (x, Σ_x) refers to the class of distributions that have prescribed mean $\bar{x}$ and Σ_x, but are otherwise arbitrary; likewise for y. In formulation (**3**) the term θ is the minimal probability of correct classification of future data. Let us denote

by $\kappa(\theta) = \sqrt{\frac{\theta}{1-\theta}}$. As the result of Marshall and Olkin [7], an optimal hyperplane $F(a_*, b_*)$ exists, and can be determined by solving the convex second order cone optimization problem, with complexity similar to SVM [6]:

$$\kappa_*^{-1} := \min_a \sqrt{a^T \Sigma_x a} + \sqrt{a^T \Sigma_y a} \quad s.t. \quad a^T (\bar{x} - \bar{y}) = 1, \tag{4}$$

and setting b to the value $b_* = a_*^T \bar{x} - \kappa_* \sqrt{a_*^T \Sigma_x a_*}$, where a_* is an optimal solution of (**3**). The optimal worst-case misclassification probability is obtained via

$$1 - \theta_* = \frac{1}{1+\kappa_*^2} = \frac{\left(\sqrt{a_*^T \Sigma_x a_*} + \sqrt{a_*^T \Sigma_y a_*}\right)^2}{1 + \left(\sqrt{a_*^T \Sigma_x a_*} + \sqrt{a_*^T \Sigma_y a_*}\right)^2} \tag{5}$$

Learning large margin classifiers [8] has become an active research topic. SVM [6] aims to find a hyperplane, which can separate two classes of data with the maximal margin. However, this margin is defined in a "local" way, i.e., the margin is exclusively determined by some support vectors, whereas all other points are totally irrelevant to the decision hyperplane. MPM considers data in a global fashion, while SVM actually discards the global information of data including geometric information and the statistical trend of data occurrence. Nonlinear decision boundaries can be obtained by "Kernel" trick that has been used in support vector machine.

5 SWS (Strong-to-Weak-to-Strong) Algorithm

Assume that the binary classifiers are chosen from some hypothesis class H .The following natural learning problems arise,

1. Given a matrix M, find a set binary classifiers $\bar{h}$ which have small empirical loss.
2. Given a set of $\bar{h}$, find a matrix M which has small empirical loss.
3. Find both a matrix M and a set $\bar{h}$ which have small empirical loss.

The previous methods have focused mostly on the first problem. Most of these works have used predefined output codes, independently of the specific application and the learning algorithm. Furthermore, the "decoding" assigns the same weight to each learned binary classifier, regardless of its performances.

The research [3] has mainly concentrated on the code design problem (problem 2). However , solving optimization problem (1) is time-consuming.

We mainly handle the 3rd problem. However it is so hard to solve the designing problem not to mention finding a "good" classifier and a wonderful output codes simultaneously by using general optimization methods. Therefore a heuristic algorithm has been proposed instead of solving directly the optimization problem (1).

In our framework SWS (Strong-to-Weak-to-Strong), we generalize the notion of "weak" algorithm [9]. We can view an algorithm with less iterative step of optimization

as a "weak" algorithm and make use of the kernel trick for "weak" algorithm to work in high dimensional spaces, finally improve the performances.

The model of SWS and the heuristic algorithm make it realizable to solve both problems mentioned above with acceptable complexion.

Recently a number of powerful kernel-based learning machines, e.g. SVM [6], Kernel Fisher Discriminant [10] and Kernel Principal Component [11], have been proposed. In KPCA, kernel serves as preprocessing while in SVM kernel has effect on classification in the middle of process.

There could be two stages for kernel to affect the result of our algorithm. The first is in the middle of process as it behaves in SVM. The second is where algorithm transforms several weak classifiers to a strong classifier. For simplicity, from now on, we do not make any distinction between two kernels and transformation $\bar{\phi}$.

5.1 Strong-to-Weak Stage

We use l MPMs as binary classifiers and initially predefine different partitions of the set of the labels, on which l binary weak classifiers are based.

In the Strong-to-Weak stage, we transform "Strong" classifier to "Weak" classifier by equipping less iterative number of optimization while preserving its characteristics like large margin and geometric properties.

On one hand, it significantly decreases total time-consuming in the case of large numbers of classes because each classifier needs less iterative step of optimization.

On the other hand, our algorithm takes the geometric difference of classes into account while other methods ignore the difference because MPM uses Mahalanobis distance that involves geometric information. Therefore SWS preserves the characteristics.

While the algorithm faces the problem with small data sets, estimation errors in the means and covariances of the classes would affect the results. Based on our experiments, we found that adding a regularization term [4] of a classical form has less effect and is sensitive to choices of the parameters. Thus we used some filter methods and found that resample method was well done.

Therefore we are able to use a simple iterative least-squares approach [4] with less iterative step to solve the kernelization version of problem (4) and train the binary classifiers because the algorithm only requires "Weak" learning algorithms.

5.2 Weak-to-Strong Stage

In the Weak-to-Strong stage, we make use of the kernel trick for "Weak" algorithm to work in high dimensional spaces and finally improve the performances. Multiclass classifier obtained in our algorithm becomes a much more "Strong" learning algorithm according to the classification performance.

A few previous heuristics attempting [12] to modify the output codes were suggested. However, they did not yield significant improvements. Unlike those methods, our algorithm implements implicit update in high dimensional spaces by using a transformation $\bar{\phi}: \mathbb{R}^l \to Z$. And the output codes update implicitly occurs in final discrimination from (2):

$$H(x) = \arg\max_r \{K(\bar{h}(x), \overline{M}_r^{update})\} = \arg\max_r \left\{\sum_i \gamma_{i,r} K(\bar{h}(x), \bar{h}(x_i))\right\} \tag{6}$$

We notice that the saddle point from optimization problem (1) we are seeking is a minimum for the primal variables with respect to ζ_i. We can get

$$\sum_r \eta_{i,r} = 1, \ i.e. \ \bar{\gamma}_i \le \bar{1}_{y_i} \ \ and \ \bar{\gamma}_i \cdot \bar{1} = 0 \tag{7}$$

And $\bar{1}_{y_i}$ may be viewed as the correct point distribution, $\eta_{i,r}$ could be viewed as the distribution obtained by the algorithm over the labels for each example. Then we can view $\bar{\gamma}_i$ as the difference between the former and the later. It is natural to say that an example x_i affects the solution (6) if and only if $\bar{\eta}_i$ is not a point distribution concentrating on the correct label y_i. Further we can say that only the questionable points contribute to the learning process and regard them as "critical points".

We have realized that it is difficult to solve the optimization problem (1) directly. What have been mentioned above motivate us to develop a heuristic algorithm to solve the problem. We notice that one "critical point" may contribute to more than one class while "support vector"[6] contributes to only one class. Further we believe that it is limitation with SVM. It is typically assumed that the set of labels has no underlying structure, however there exist lots of different relation among category in practice especially in the case of document classification. It means that it is reasonable that one example makes different contributions to some classes or classifiers simultaneously.

We denote the critical degree by element $B_{i,j}^c$ of a $m \times k$ weight matrix B^c. The i th training data contributes to the j th class with degree $B_{i,j}^c$ and row B_i^c of weight matrix is approximation of $\bar{\gamma}_i$. Note that we need a partition of l classifiers into k sets ($l \ne k$). That is to say elements of each set correspond one certain class not one classifier. There are two confidence matrices: $W^c : m \times k$, whose elements are confidences of examples to classes, and $W^m : m \times l$, whose elements are confidences of examples to classifiers. Row W_i^c of confidence matrix is approximation of $\bar{\eta}_i$. We can easily obtain W^c from W^m that can be directly constructed according to margin magnitude $(a^T x - b)$. Therefore we can construct B^c from W^c as shown in Fig. 2. For simplicity, from now on, we do not make any distinction between W^c and W^m. We call them as confidence matrix. Especially $W^c = W^m$ if l=k. We then propose a heuristic strategy using vector V^c whose elements are probability outputs of MPMs in Fig. 1 and example margin to build confidence matrix whose elements are feasible solution to the optimization problem (1), further to construct weight matrix.

The algorithm enforces that each row of weight matrix satisfies the constraint (7). It is easy to verify rationality of these approximations for probability and margin indicate contributive degree of an example to a class as $\eta_{i,r}$ behaves.

Get confidence matrix $W^c \leftarrow EvaluateFromClassfier(W^m), W^c : m \times k \quad W^m : m \times l$

Get confidence vector $V^c \leftarrow EvaluateClassfier(V^m), \qquad V^c : 1 \times k \quad V^m : 1 \times l$

Repeat $i = 1, \ldots, m \quad j = 1, \ldots k$

if $W^c_{i,:} < 0$ or $W^c_{i,j} > 0$

$W^c_{i,j} \leftarrow V^c_j \times W^c_{i,j}$

else $W^c_{i,j} \leftarrow 0$

end get fresh confidence matrix

Fig. 1. Algorithm description of constructing confidence matrix. Note: V^c_j has been normalized.

Get fresh confidence matrix $\left[W^c_{i,j}\right]$, $i = 1, \ldots, m \quad j = 1, \ldots k$ and sparseRate that controls sparseness

Repeat $j = 1, \ldots k$

$V_n \leftarrow GetSlackData(W^c_{:,j})$

$mvalue = \min(V_n)$

$Removeneg(W^c_{:,j})$ (remove all other negative elements in the j th column)

$W^c_{:,j} \leftarrow W^c_{:,j} + |mvalue|$

$mavalue = \max(W^c_{:,j})$ (get maximal value of the j th column)

$W^c_{:,j} \leftarrow mavalue - W^c_{:,j}$

$B^c_{:,j} = pickCritical(W^c_{:,j}, sparseness)$

end $B^c_{:,j} = Constrain(B^c_{:,j})$

$m \times k$ matrix B^c whose elements are critical degree of each point to each class

Fig. 2. Algorithm description of constructing weight matrix

One can obtain worst-case bound on the probability of misclassification of future data in MPM. This probability belongs to one binary classifier and each training data x classified by a certain binary classifier has a corresponding margin magnitude in Fig. 1. If this point can be correctly predicted by a classifier h_j, corresponding element of confidence matrix should be $a_j^T x - b^T$ (positive), otherwise this element should be zero that indicates the point rejects the class except for one case, where we should set this element to be $a_j^T x - b^T$ (negative) if no classifiers can correctly predict the corresponding point. Although the value is negative, it actually carries available information that could predict one class, to which this point most possibly belongs. Once all conditions above are available, for each class we pick some maximal negative (their absolute value are small), which indicate that corresponding data reject the class with least degree, from each column of this matrix according to a predefined parameter that controls tolerance level of class to errors if there are some negative in this column. Then algorithm removes all other negative in the same column and adds absolute value

of the smallest negative chosen to all the non-zero elements in the same column. We get maximal value in each column and subtract each non-zero element in the same column from this maximal value. Then we choose some elements, whose corresponding data are 'critical points' in descending order according to a parameter that controls sparseness, then construct weight matrix in Fig. 2. We make the matrix elements satisfy the constraint (7). In addition, one can find underlying patterns among classes through relation between data and classes.

After training and constructing weight matrix, one can predicate a label given an instance:

$$H(x) = \arg\max_r \left\{ \sum_i B^c{}_{i,r} K(\bar{h}(x), \bar{h}(x_i)) \right\} \tag{8}$$

6 Experiments

In our experiment we set $l = k$, i.e. one-against-rest method that is less competitive as shown in most researches [13,14]. However our experiments report that better performances can be achieved although using one-against-rest method. We use tf×idf and feature selection based on information gained to build training and test data.

The following experiment compares the performance of the new algorithm with five conventional learning methods commonly used for text categorization.

The empirical evaluation is done on the "ModApte" spite of the Reuters-21578 dataset that was compiled by Lewis at AT&T. The ModApte spite, which has 12,902 stories and the average length of 200 words, leads to a corpus of 9603 training stories (75% of the total) and 3299 test stories (25% of the total). Of the 135 potential topic categories only those 90 are used for that reason that there is at least one training and one test example, after preprocessing the training corpus, containing 9962 distinct terms.

We try to replicate the experimental setup in [15, 16], and the best results of all the methods are described in Table 1. Table 2 displays the results of using different iterative steps of optimization for "Weak" algorithm.

Table 1. Precision/recall-breakeven point on the ten largest categories

	Bayes	Rocchio	C4.5	k-NN	SVM	SWS
Earn	95.9	96.1	96.1	97.3	**98.5**	98.0
Acq	91.5	92.1	85.3	92.0	**95.4**	93.5
Money-fx	62.9	67.6	69.4	78.2	76.3	**82.5**
Grain	72.5	79.5	89.1	82.2	**93.1**	92.5
Crude	81.0	81.5	75.5	85.7	89.0	**89.2**
Trade	50.0	77.4	59.2	77.4	78.0	**82.1**
Interest	58.0	72.5	49.1	74.0	76.2	**79.3**
Ship	78.7	83.1	80.9	79.2	87.6	**89.1**
Wheat	60.6	79.4	85.5	76.6	85.9	**86.6**
Corn	47.3	62.2	87.7	77.9	85.7	**90.3**

Table 2. Precision/recall-breakeven point with various iterative numbers of weak algorithm

Iterative numbers	5	10	30	50	100	150
Earn	**98.0**	**98.0**	**98.0**	**98.0**	**98.0**	**98.0**
Acq	92.7	**93.5**	**93.5**	**93.5**	**93.5**	**93.5**
Money-fx	81.2	**82.5**	**82.5**	**82.5**	81.5	81.7
Grain	91.0	**92.5**	**92.5**	**92.5**	**92.5**	90.0
Crude	88.3	**89.2**	**89.2**	**89.2**	**89.2**	87.5
Trade	**82.1**	**82.1**	**82.1**	**82.1**	79.7	78.0
Interest	76.2	**79.3**	**79.3**	**79.3**	78.0	76.5
Ship	89.0	**89.1**	**89.1**	**89.1**	**89.1**	**89.1**
Wheat	86.0	**86.6**	**86.6**	**86.6**	86.1	85.0
Corn	88.7	**90.3**	**90.3**	**90.3**	**90.3**	**90.3**

Best results can be achieved by choosing 2-degree polynomial kernel in SWS.

From Table 1 we observe an interesting phenomenon that our algorithm (SWS) outperforms others in the case that much poor performances could be achieved by all other methods, while in other cases our results are somewhat worse than the best performances.

This phenomenon also inspires us to try to find the reason from the characteristic of MPM that considers data in a geometric fashion, while SVM [6] ignores this kind of information. As known in chapter 4 MPM uses Mahalanobis distance that involves geometric information. It could be the reason that large difference between subspaces of classes leads to poor performance.

It is clear that Table 2 shows the algorithm of SWS can preserve whole performances although decreasing total iterative step of optimization.

7 Conclusion and Future Works

We have introduced a new method for multiclass problems. Results suggest that our algorithm outperforms other algorithms although using one-against-rest strategy.

The subspace of each category of text is quite different from others. This could be the reason why some methods like SVM fail to achieve high performance on some "difficult" datasets where subspaces of different categories are quite different.

Our algorithm takes the geometric difference of classes into account while other methods ignore the difference because MPM uses Mahalanobis distance involving geometric information. Therefore SWS can improve the performance on some "difficult" datasets for taking the different information of subspaces into account.

Generally there are large numbers of categories in text classification problem. Traditional methods are immersed in complex computation.

Our algorithm achieved acceptable complexion through transforming a "strong" learning algorithm to "weak" one. SWS also makes it possible to be insensitive to optimization approach, therefore it can use a simple iterative least-squares approach.

Finally, a nonlinear transformation is utilized to improve the performance.

These characteristics make our method appropriate for text classification.

As we mentioned in section 5, there is a parameter that controls sparseness in the algorithm. How to adaptively tune this parameter is an interesting topic for future work.

Acknowledgements

This research was supported by Ningbo Doctor Science Fund grant 2005A610002.

References

1. Allwein, E., Schapire, R., & Singer, Y.: Reducing multiclass to binary: A unifying approach for margin classifiers. In Machine Learning: Proceedings of the SeventeenthInternational Conference (2000)
2. Dietterich, T. G., & Bakiri, G.: Solving multiclass learning problems via error correcting output codes. Journal of Artificial Intelligence Research, (1995) 2, 263–286
3. Koby Crammer, Yoram Singer: On the Learnability and Design of Output Codes for Multiclass Problems. Proceedings of the Thirteenth Annual Conference on Computational Learning Theory (2000) 35–46
4. Lanckriet, R. G., Ghaoui, L.E., Bhattacharyya, C., and Jordan, M. I.: A robust minimax approach to classification. Journal of Machine Learning Research (2002) 3:555–582
5. Y. Yang and J. Pedersen: A comparative study on feature selection in text categorization. In International Conference on Machine Learning (ICML) (1997)
6. V. Vapnik: The Nature of Statistical Learning Theory. Spinger Verlag, New York (1995)
7. Marshall, A. W. and Olkin, I.: Multivariate Chebyshev inequalities. Annals of Mathematical Statistics (1960) 31(4): 1001–1014
8. Smola, A. J., Bartlett, P. L., Scholkopf, B., & Schuurmans, D.: Advances in large margin classi-fiers. MIT Press (2000)
9. Yoav Freund: Boosting a weak learning algorithm by majority. Information and Computation (1995) 121(2): 256–285
10. S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K. -R. Müller: Fisher discriminant analysis with kernels. Neural Networks for Signal Processing IX (1999) 41–48
11. B. Schölkopf, A.J. Smola, K. -R. Müller: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation (1998) 10:1299–1319
12. Aha, D. W., & Bankert, R. L.: Cloud classification using error-correcting output codes. In Artificial Intelligence Applications: Natural Science, Agriculture, and Environmental Science (1997) 11:13–28
13. Hsu, C., & Lin, C. A.: comparison of methods for multiclass support vector machines. Technical report, Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan (2001) 19
14. J. C. Platt, N. Cristianini, and J. Shawe-Taylor: Large margin DAGs for multiclass classifycation. In Advances in Neural Information Processing Systems, MIT Press (2000) 12:547–553
15. Joachims, T.: Text cateforization with support vector machines: Learning with many relevant features. In Proceedings 10th European Conference on Machine Learning (ECML), Springer Verlag (1998)
16. Taku Kudo and Yuji Matsumoto: Fast methods for kernel-based text analysis. In Proceedings of the 41est Annual Meeting of the Association for Computational Linguistics (2003) 24-31