

SVM paradoxes

*PSI09, Novosibirsk, Akademgorodok,
15 - 19 June, 2009*

Jean Beney¹ and Cornelis H.A. Koster²

¹ LCI, Département Informatique, INSA de Lyon F69621 Villeurbanne
Université de Lyon, jean.beney@insa-lyon.fr

² Radboud University, ICIS, Nijmegen, kees@cs.ru.nl

Abstract. Support Vector Machines (SVM) is widely considered to be the best algorithm for text classification because it is based on a well-founded theory (SRM): in the separable case it provides the best result possible for a given set of separation functions, and therefore it does not require tuning. In this paper we scrutinize these suppositions, and encounter some paradoxes.

In a large-scale experiment it is shown that even in the separable case SVM's extension to non-separable data may give a better result by minimizing the confidence interval of the risk. However, the use of this extension necessitates the tuning of the complexity constant.

Furthermore, the use of SVM for optimizing precision and recall through the F function necessitates the tuning of the threshold found by SVM. But the tuned classifier does not generalize well. Furthermore, a more precise definition is given to the notion of training errors.

1 Introduction and related work

Support Vector Machines (SVM) is the most popular and successful algorithm for classification. Apart from its use in text classification, SVM has been used successfully in many other classification tasks: in the classification of speech patterns [7], plasma discharge [14], cancers [16,1] as well as various kinds of images.

In text classification its success is relative: compared to other methods, SVM gives the best accuracy but in some cases this accuracy is not very good ([5,2,11]), too low for practical use.

The problem lies in the number of features. There may be as few as 8 parameters in plasma discharge classification to a few hundred pixels, but in document classification we may encounter over 500,000 word forms to be used as features. This problem is linked to the bound of the generalization error ([19], pages 77-81) which increases with the number of features.

But this large number of features has an advantage: it is always possible to find enough words to build a separator between two classes [8]. Thus, the data is linearly separable, and any kernel (polynomial, RBF) can be used as well. As no properties of the vector space can be derived from the construction of this

space, there is no way to choose the *best* kernel. We may as well work with a linear separator and use a method for linearly separable data.

The elegant theoretical foundation of SVM proves that this method gives the best error expectation and therefore it is often said that it is not subject to overfitting. As a corollary, it is also claimed that it has the advantage to have no parameter that should be tuned [8].

Even in the separable case, many authors prefer the SVM variant for non-separable data [12], or even the skew variant that attaches different weights to false negatives and false positives [20]. In order to explain this paradox, we report on an experiment with SVM on a large set of patent applications and we analyze the results of the tuning of the complexity constant (section 4).

Another paradox is that SVM was defined to minimize the number of errors whereas it is often used with another accuracy measure such as precision, recall and the *F1* function. Its result may not necessarily be optimal in terms of these other accuracy measures.

In particular, precision and recall are often very different, which makes it likely that the *F1*-value is not optimal because it is far from the break-even point. In section 5 we discuss how to optimize *F1* by modifying the threshold in order to reach this break-even point.

Other authors have written on the optimization of SVM parameters in relation to other measures, such as Mean Average Precision [20], *F1*: Li-CoNLL [13], or more sophisticated methods (derivative-free method APPSPACK: [6], Genetic Programming: [4]). These methods cannot be used with our huge document set because of the large amount of training they require. But previous experiments have shown that the optimal parameters may have rather different values when working with a small subset [9].

In the next sections, we first recapitulate the theory of SVM and then describe the experiments we have performed.

2 SVM and Structural Risk Minimization

SVM is based on the method of Structural Risk Minimization (SRM) as introduced by Vapnik [19]. It consists in the search for that function, belonging to a given set of functions, that minimizes the functional risk. If the function set is characterized by parameters α , this risk can be defined by:

$$R(\alpha) = \int Q(z, \alpha) dF(z)$$

where $Q(z, \alpha)$ is a measure of the *loss* between the true function and the estimated function and $F(z)$ is the distribution of the data in the given space. When $F(z)$ is not known, but only l points (train examples) are given, the empirical risk is used:

$$R_{emp}(\alpha) = \sum_{i=1}^l Q(z_i, \alpha)$$

These two risks are linked by the inequality:

$$R(\alpha) \leq R_{emp}(\alpha) + \Phi(\alpha)$$

where $\Phi(\alpha)$ is the confidence interval.

In the classification problem, given a set of functions (e.g. the linear functions, hyperplanes), every exact separator of the train set is a function that minimizes the empirical risk (to zero). In general, it is advisable to look for the function that minimizes the confidence interval, in order to minimize the upper bound of the functional risk.

The optimal function is the separator for which the margin between positive and negative examples is maximal. Support Vector Machines compute this separator, even when it is not linear, or when the train data is not separable.

2.1 The optimal hyperplane

In this brief introduction to SVM, we follow [3]. Let there be given a set of labeled training examples $x_i \in \mathbb{R}^n$:

$$E = \{ \langle x_i, y_i \rangle \mid 1 \leq i \leq M \}, y_i = \pm 1$$

is said to be linearly separable if:

$$y_i(\langle w, x_i \rangle + b) \geq 1 \tag{1}$$

where $\langle \cdot \rangle$ is the inner product. The optimal hyperplane:

$$\langle w_0, x \rangle + b_0 = 0$$

is the unique one that separates the examples with a maximal margin. This margin is:

$$\rho(w, b) = \frac{2}{|w|} = \frac{2}{\sqrt{\langle w, w \rangle}}$$

The problem is then to find w_0 that minimizes $w.w$ under the constraints (1).

With the Lagrange multipliers method, it is easy to show that the vector w_0 can be written as a combination of the training vectors :

$$w_0 = \sum_{i=1}^M \alpha_i y_i x_i \tag{2}$$

Since the multipliers α_i are zero for many training examples, w_0 depends only on the *Support Vectors* (those vectors that are near to the margin). The maximal margin also depends only on the SV because:

$$\langle w_0, w_0 \rangle = \sum_{i=1}^M \alpha_i$$

The quadratic programming problem (called *dual*) to be solved is to maximize:

$$L(\Lambda) = \sum_{i=1}^M \alpha_i - \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^M \alpha_i \alpha_j y_i y_j \langle x_i \cdot x_j \rangle \quad (3)$$

subject to the constraints:

$$\sum_{i=1}^M \alpha_i y_i = 0 \quad (4)$$

$$\alpha_i \geq 0, \quad i = 1, M \quad (5)$$

2.2 Non-separable data: the soft margin hyperplane

When the train set is not linearly separable, the preceding problem and its dual have no solution. In that case, *training errors* should be allowed for, but their number should be minimized. This is done by introducing slack variables $\xi_i \geq 0$ to relax the constraints:

$$y_i(\langle w \cdot x_i \rangle + b) \geq 1 - \xi_i \quad i = 1, \dots, M$$

and using:

$$\sum_{i=1}^M \xi_i$$

as a measure of the number of errors³ that should be minimized. In fact, this sum is an upper bound of the number of errors.

As we now must optimize two functions, they are combined in one with the use of a constant C that expresses the relative importance of the 2 optima.

$$R = \frac{1}{2} \langle w \cdot w \rangle + C \sum_{i=1}^M \xi_i$$

The constant C is called the *complexity constant*.

The Lagrange multipliers method leads to the following dual problem: minimize $L(\Lambda)$ (equation 3) subject to the constraints (4) and :

$$C \geq \alpha_i \geq 0, \quad i = 1, M \quad (6)$$

³ The real number of errors cannot be used because it is not derivable. Other functions may be used instead.

2.3 Kernel functions

As the above objective functions only depend on the inner products between pairs of data vectors, these inner products can be replaced by a *Kernel function* $K(u, v)$, that is a symmetric positive definite function, which leads to the minimization of:

$$L(\Lambda) = \sum_{i=1}^M \alpha_i - \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^M \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

under the constraints (4) and (4) (or (6) for non-separable data). This method allows to look for non-linear separators, for example a polynomial or exponential (radial or potential) function. Since our data is linearly separable, we will not use such kernel functions.

2.4 Solving the quadratic programming problem

Due to the large number of training data x_i , and consequently the large number of Lagrange multipliers to compute, it may be too time consuming to solve the dual problem. Several investigations have led to efficient methods to solve quadratic programming problems by working on subsets of the variables to be computed. *Sequential Minimal Optimization* [15] considers two variables in each step. In *SVM^{light}* [18], each step is not limited to two variables.

3 Experimentation setup

We briefly describe the software and data sets used in the experiments.

3.1 The programs

In the experiment we made use of *SVM^{light}* implemented by Thorsten Joachims [18]. This program proceeds by solving sub-problems (several vectors are selected and their multipliers are moved towards the solution) until all Lagrange multipliers have been computed.

The data were preprocessed by term selection (selecting 105 000 out of 558 000 terms according to the Simplified ChiSquare criterion). The strength of the terms (raw words) in the documents were computed using the LTC formula, followed by a cosine normalization.

Due to the large size of the corpus EPO2F and the high number of runs to perform (44 classes, many parameter values) we did not use cross-validation but a single shuffle of the given example into 80% for training and 20% for testing.

3.2 The data

EPO2F is a corpus of patent applications selected by the European Patent Office for the evaluation of classification programs [10]. The documents were chosen by EPO, from one year of input, in such a way that each of the training sets corresponding to the 44 classes (called *directorates*) contains at least 2000 documents. Each patent was labeled by EPO with one or more classes (up to 7 in practice), so that we chose to classify each directorate separately.

Some statistics of EPO2F are given in the table 1.

| | |
|-----------------------|---------|
| number of documents | 68 418 |
| number of classes | 44 |
| classes/doc : mean | 1,43 |
| classes/doc : maximum | 7 |
| docs/class : mean | 2 227 |
| docs/class : minimum | 2 000 |
| docs/class : maximum | 2 947 |
| words/doc : mean | 59 528 |
| words/doc : minimum | 357 |
| words/doc : maximum | 163 261 |
| unique words | 557 790 |

Table 1. statistics of the data set EPO2F.

3.3 The quality measures

The patent applications that arrive at EPO must be sent only to the relevant directorate(s) (EPO calls this *preclassification*) and within each directorate to the relevant *examiner*(s) that will check that the invention is really new. When a document is sent to a wrong directorate, it costs time; therefore a high precision is required.

The examiners need to search in a database, to find all the related patents already registered. At this stage, a very high recall is needed.

As well as the number of errors (E), we will compute the precision (P), the recall (R) and the F_1 measure that combines them giving equal weight to both:

$$F_1 = \frac{2PR}{P + R}$$

It is known that at the point where the precision equals the recall (*break-even point*), F_1 is approximately maximal.

In other applications, where precision is more important than recall or vice versa, a more general measure F_γ can be used, where γ controls the relative importance of precision and recall. In that case, a variant of SVM for unbalanced data can be used which includes two parameters C^+ and C^- , expressing the cost

| C | train set | | | | test set | | | |
|-----|-----------|--------|--------|-----|----------|-------|--------------|------|
| | P | R | F1 | E | P | R | F1 | E |
| 0.3 | 92.75 | 70.58 | 80.16 | 830 | 88.89 | 63.05 | 73.77 | 1065 |
| 0.5 | 94.01 | 79.29 | 86.02 | 612 | 87.70 | 67.43 | 76.24 | 998 |
| 0.7 | 94.94 | 84.51 | 89.42 | 475 | 86.52 | 69.70 | 77.20 | 978 |
| 0.9 | 96.01 | 88.13 | 91.90 | 369 | 85.50 | 71.28 | 77.74 | 970 |
| 1 | 96.38 | 89.65 | 92.89 | 326 | 85.24 | 71.80 | 77.94 | 965 |
| 1.2 | 97.33 | 92.05 | 94.61 | 249 | 85.15 | 72.33 | 78.21 | 957 |
| 1.3 | 97.66 | 92.93 | 95.23 | 221 | 84.80 | 72.33 | 78.07 | 965 |
| 1.4 | 97.81 | 93.86 | 95.79 | 196 | 84.41 | 73.03 | 78.30 | 961 |
| 1.5 | 98.16 | 94.40 | 96.24 | 175 | 84.18 | 72.68 | 78.00 | 974 |
| 1.7 | 98.45 | 96.13 | 97.27 | 128 | 83.84 | 72.68 | 77.86 | 982 |
| 2 | 98.85 | 97.35 | 98.09 | 90 | 83.20 | 72.85 | 77.68 | 995 |
| 3 | 99.54 | 99.49 | 99.51 | 23 | 81.60 | 73.03 | 77.07 | 1032 |
| 5 | 99.79 | 99.83 | 99.80 | 9 | 80.19 | 73.03 | 76.44 | 1059 |
| 10 | 99.96 | 99.87 | 99.91 | 4 | 80.27 | 73.38 | 76.67 | 1061 |
| 20 | 100.00 | 99.96 | 99.97 | 1 | 79.96 | 73.38 | 76.52 | 1069 |
| 50 | 100.00 | 100.00 | 100.00 | 0 | 80.19 | 73.03 | 76.44 | 1069 |
| 100 | 100.00 | 100.00 | 100.00 | 0 | 80.15 | 72.85 | 76.32 | 1074 |
| 150 | 100.00 | 100.00 | 100.00 | 0 | 80.15 | 72.85 | 76.32 | 1074 |
| 200 | 100.00 | 100.00 | 100.00 | 0 | 80.15 | 72.85 | 76.32 | 1074 |

Table 2. Varying C (dir01).

of false positives and false negatives, in place of the C parameter of SVM for the non-separable data.

4 Tuning the complexity constant

Although the data are linearly separable with a large margin for each classification (directorate), we have experimented with different values for the C parameter (the *complexity constant*). For smaller C values, the errors have less importance in the goal function, errors are allowed provided that at the same time the margin is enlarged.

We first consider the case of one class, then of 44 (the number of directorates).

4.1 Results on 1 class

Table 2 gives the results obtained on the directorate number 01. Figure 1 shows F(1) on dir01 and its average on the 44 directorates.

These results call for the following remarks:

- When C is greater than or equal to 50, we get a perfect classifier of the train data (no errors) but the results on test data are not optimal. The limit on the Lagrange multipliers (condition 6) allows to improve the results.

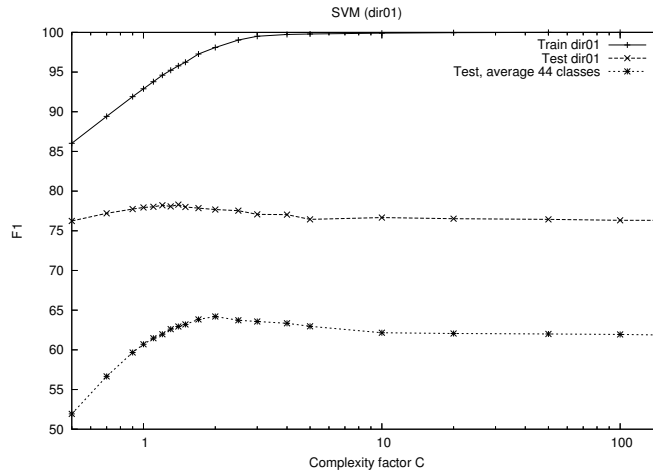


Fig. 1. Accuracy as a function of C

- When C is lower, we have a few errors on train data () and a better result on test data (+2% on F(1), -10% on the number of errors).
- The difference between the results on train and test data is very large.

These phenomena can be explained by the following arguments:

- The so-called training errors (2.2) are not necessarily errors: they are vectors that are either on the wrong side of the separation hyperplane or on the right side but inside the margin. They should rather be called *marginal errors*.
- Allowing marginal errors can give a larger margin and a better generalization error, because the confidence interval of the risk is decreased.
- The train data is not perfectly representative of the test set. This is always the case with documents when the terms are the words (raw or lemmatized) because it is known that every document brings a lot of new words.

In this situation, every method (including SVM) is subject to *overfitting*: obtaining much better results on train data than on test data. When the method includes certain parameters (C for SVM, promotion factor and number of iterations for Winnow or Perceptron [9]), they sometimes can be used to reduce the effect of overfitting. When the method has no such parameters (Rocchio, SBC), there is no way to reduce overfitting.

4.2 Results on the 44 classes

We have trained each of the 44 directorates using the strategy *one against all*. The values given are the micro-averaged F1. As in EPO2F the number of examples is stable across the classes, the macro-averaged values are very similar.

With a good choice of C, the possible gain for each class varies from 0.37% to 6.16% with an average value of 2.43%. Unfortunately, the maximum is obtained

for different value of C for the different classes (between 1 and 150), which means that tuning is necessary for each class separately.

However, by choosing $C = 2$ for all the classes, the gain is 2.20% ($F(1)=64.20$) compared to the case $C = 200$ (perfect classifier of the train data).

5 Tuning the threshold

When the result hyperplane:

$$\langle w_0 . x_i \rangle + b_0 = 0$$

is used for the classification of new data, a document (vector) is said to belong to the class when:

$$\langle w_0 . x_i \rangle \geq -b_0$$

Then b_0 is called the threshold. It is computed using one (unbounded⁴) support vector x_i, y_i because such a vector lies on the margin:

$$b_0 = y_i - \langle w_0 . x_i \rangle$$

The above results (table 2) are not at the break-even point (3.3): the precision is always much larger than the recall. Therefore, we may expect that it is possible to get a larger F1 value. One possible method is to use the SVM variant for unbalanced data (3.3) with $C^+ > C^-$, which is also paradoxal because we are looking for a balanced result.

We have experimented with another method: to chose a smaller threshold whose result will be to select more documents, then enlarging the recall and probably decreasing the precision.

5.1 Results on 1 class

Table 3 and figure 2 show that setting the threshold at 0.67 improves F1 on the train data by .95% above the result given in table 2. The threshold at the break-even point is different (about 0.8) but F1 is rather stable between these two points.

The corresponding F1 increase on test data is even larger (+1.36%) but this is still not the maximum possible (+1.83% for a threshold 0.7).

This improvement causes also a decrease of the number of errors on train documents (196 to 156), which may be surprising as SVM is already supposed to minimize the number of errors. But, as before, we must consider that SVM minimizes the number of marginal errors while we are interested in the number of true errors.

⁴ a support vector x_i is bounded if $\alpha_i = C$; it is unbounded when $\alpha_i < C$

| b | train | | | | test | | | |
|------------|-------|-------|--------------|--------|-------|-------|--------------|--------|
| | P | R | F1 | Nb Err | P | R | F1 | Nb Err |
| 0.5 | 94.45 | 98.78 | 96.56 | 167 | 70.10 | 87.04 | 77.65 | 286 |
| 0.6 | 95.22 | 98.15 | 96.66 | 161 | 74.05 | 84.94 | 79.12 | 256 |
| 0.65 | 95.68 | 97.77 | 96.71 | 158 | 75.12 | 84.06 | 79.33 | 250 |
| 0.66 | 95.75 | 97.73 | 96.72 | 157 | 75.55 | 83.89 | 79.50 | 247 |
| 0.67 | 95.82 | 97.69 | 96.74 | 156 | 75.99 | 83.71 | 79.66 | 244 |
| 0.68 | 95.82 | 97.56 | 96.68 | 159 | 76.57 | 83.54 | 79.90 | 240 |
| 0.69 | 95.82 | 97.52 | 96.66 | 160 | 77.02 | 83.36 | 80.06 | 237 |
| 0.7 | 95.90 | 97.43 | 96.65 | 160 | 77.45 | 83.01 | 80.13 | 235 |
| 0.75 | 96.32 | 96.93 | 96.62 | 161 | 78.89 | 79.86 | 79.37 | 237 |
| 0.8 | 96.71 | 96.55 | 96.62 | 160 | 80.54 | 78.28 | 79.39 | 232 |
| 0.85 | 97.14 | 95.79 | 96.46 | 167 | 81.60 | 76.88 | 79.16 | 231 |
| 0.9 | 97.50 | 95.08 | 96.27 | 175 | 82.82 | 75.13 | 78.78 | 231 |
| 0.99830698 | 97.81 | 93.86 | 95.79 | 196 | 84.41 | 73.03 | 78.30 | 231 |
| 1 | 97.81 | 93.77 | 95.74 | 198 | 84.38 | 72.85 | 78.19 | 232 |
| 1.1 | 98.26 | 92.42 | 95.25 | 219 | 86.39 | 70.05 | 77.36 | 234 |

Table 3. Threshold tuning (dir01, C=1.4).

| | dir01 | | average 44 classes | |
|-----------------------|-----------|------------|--------------------|------------|
| | b_0 SVM | best b_0 | b_0 SVM | best b_0 |
| C>100 | 72.85 | | 61.80 | |
| C=2 | 76.32 | 78.82 | 64.20 | 64.00 |
| best C for each class | 78.30 | 79.33 | 64.23 | 63.50 |

Table 4. F1 on unseen data after tuning of the threshold

5.2 Results on the 44 classes

Unfortunately, the positive result obtained on the class dir01 cannot be repeated on all other classes: the tuning of the threshold on the train data does not necessarily improve F1 on the test data. The micro-averaged effect on F1 may even be negative as shown in table 4, where we compare the best C value for each class with a *standard* value C=2.

Note that for a perfect classifier of the train data (C>100), no tuning is possible since there are no errors to remove.

It means that the result given by SVM, even if it not the best possible on train data, has a very good generalization capacity. In most cases, trying to reduce the number of *true train errors* leads to overfitting, and therefore a worse result on test data.

6 Conclusions and further work

In this paper, we have addressed a number of paradoxes and common misunderstandings associated with the SVM classification method.

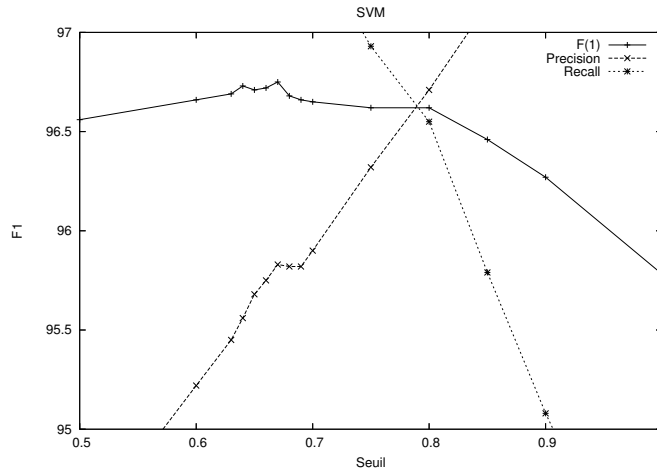


Fig. 2. accuracy on unseen data as a function of the threshold (dir01)

To begin with, we have shown experimentally that SVM is subject to overfitting just like many other classification methods. It is not perfect in this respect.

The basic method for separable data generalizes rather well. But the use of the variant for non-separable data (which would seem useless) allows the reduction of overfitting by reducing the confidence interval, at the price of a less than perfect separation of the train data (the empirical risk is not null).

Then, the complexity constant must be chosen carefully for each class by trying several values, with the possible help of the gradient.

The large difference between the accuracy on train and test documents is due to the many new terms that appear with new documents. When we build a test set independent from the training set, they are not identically distributed. Therefore, the train data cannot be said to belong to the same distribution as the whole document set, and the same holds for the test set.

SVM is not defined to optimize F1 so that it is possible to increase the value of this measure on the train data by moving the threshold in order to be nearer from the break-even point (precision equals recall). But the classifier obtained by this process does not generalize as well as the one given by SVM.

We have also seen that, when we decrease the risk (less true errors and better F1 on train data), we often increase its confidence interval; this *improvement* leads to overfitting. In this situation, the direct SVM result is the best and must be taken as it is, even if the method was not designed to optimize F1..

Incidentally, this result suggest that optimizing F1 on the train data is not the best strategy because the classifier obtained does not generalize very well on a train set.

In consequence of these results, we are working on the following ideas:

- is it possible to design a better problem setting based on true errors instead of marginal errors?
- how can we establish a theory of overfitting, based on statistics of new words in each documents?

Acknowledgement: The experimentation was performed on the Large Data Collider of the Information Retrieval Facility.

References

1. N. E. Ayat and M. Cheriet C. Y. Suen. Kmod-a two parameter svm kernel for pattern recognition. In *ICPR*, pages 30331–30334, 2002.
2. A. Basu, C. Watters, and M. Shepherd. Support vector machines for text categorization. In *HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 4*, page 103.3, Washington, DC, USA, 2003. IEEE Computer Society.
3. Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
4. Ronan Cummins and Colm O’Riordan. Evolved term-weighting schemes in Information Retrieval: an analysis of the solution space. *Artificial Intelligence Review*, pp 35–47, nov. 2007.
5. Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive learning algorithms and representations for text categorization. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*, pages 148–155, New York, NY, USA, 1998. ACM Press.
6. Tatjana Eitrich and Bruno Lang. Efficient optimization of support vector machine learning parameters for unbalanced datasets *J. Comput. Appl. Math.*, 196(2):425–436, 2006.
7. Jennifer Huang. Face recognition using component-based svm classification and morphable models. In *SVM*, pages 334–341, 2002.
8. Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Claire Nédellec and Céline Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.
9. C. H. A. Koster and Jean Beney. On the importance of parameter tuning in text classification. In *Proceedings of PSI 2006*, LNCS 4378, pages 269–283. Springer-Verlag, 2006.
10. Marc Krier and Francesco Zaccà. Automatic categorisation applications at the european patent office. *World Patent Information*, 24:187–196, 2002.
11. Boris Lauser and Andreas Hotho. Automatic multi-label subject indexing in a multilingual environment. In *Proc. of the 7th European Conference in Research and Advanced Technology for Digital Libraries, ECDL 2003*, volume 2769, pages 140–151. Springer, 2003.
12. Yaoyong Li, Kalina Bontcheva, and Hamish Cunningham. Svm based learning system for information extraction. In *Proceedings of Sheffield Machine Learning Workshop, Lecture Notes in Computer Science*. Springer Verlag, 2005.

13. Yaoyong Li and Kalina Bontcheva and Hamish Cunningham. Using Uneven Margins SVM and Perceptron for Information Extraction. In *Proceedings of Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pp , 2005
14. A A Lukianitsa, F M Zhdanov, and F S Zaitsev. Analyses of iter operation mode using the support vector machine technique for plasma discharge classification. *Plasma Physics and Controlled Fusion*, 50(6):065013 (14pp), 2008.
15. J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, Cambridge, 1998.
16. Ryan Rifkin, Sayan Mukherjee, Pablo Tamayo, Sridhar Ramaswamy, Chen hsiang Yeang, Michael Angelo, Michael Reich, Tomaso Poggio, Eric S. L, Todd R. Golub, and Jill P. Mesirov. An analytical method for multiclass molecular cancer classification. *SIAM Review*, 45:706–723, 2003.
17. Fabrizio Sebastiani. Classification of text, automatic In *The Encyclopedia of Language and Linguistics*, pages 457–463. Elsevier Science Publishers, 2006.
18. Joachims Thorsten. Making large-scale svm learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*, chapter 11, pages 41–56. MIT Press, Cambridge, 1999.
19. V. Vapnik. *The nature of Statistical Learning Theory*. Springer-Verlag, New-York, 2^e edition, 1995-2000.
20. Yisong Yue and Thomas Finley. A support vector method for optimizing average precision. In *Proceedings of SIGIR'07*, 2007.