

OPTIMIZATION OF SVM IN A SPACE OF TWO PARAMETERS: WEAK MARGIN AND INTERCEPT. APPLICATIONS IN TEXT CLASSIFIER DESIGN

ANDREI V. ANGHELESCU AND ILYA B. MUCHNIK

ABSTRACT. A method for optimizing parameters used by support vector machines in text classification applications is described. It is well known that in the case of overlapping classes, two parameters of SVM, effective length of weak margin, C , and intercept, b , can be changed to achieve better results than those yielded by standard values determined under the assumption that both conditional class distributions in the margin region are uniform [1, 2]. Proposed methods to optimize the parameters are usually based on the assumption that these distributions are normal, with unknown means and variances. In this paper we propose a novel approach to optimize these parameters without making use of any assumption about the distributions and estimate them efficiently from the training data. We evaluated our performance on 11 TREC topics for which best results submitted to the TREC 11 batch filtering track obtained recalls of less than 50%, with an average of 29%. We show that our method attains recalls of as much as 88% with an average recall of 50%, while maintaining good overall accuracy.

1. INTRODUCTION

A common context for designing text classifiers consists of problems (further referred to as topics) for which the training data contains a small number of positive examples and a much larger number of negative ones. For instance, for information retrieval algorithm evaluation purposes in the environment of TREC [3], a standard testbed was created using the Reuters RCV1 corpus [4] and defining 100 different topics. For each such topic, there are available some 10-400 positive examples and some 100-1500 negative ones. This example is not singular; other text databases show similar characteristics.

Such common features of the training data, namely very unbalanced proportions of positive/negative examples, pose difficult questions to many machine learning algorithms and they often lead to building poor classifiers that label all documents as negative.

We used a 1-nearest neighbor classification algorithm on the aforementioned 100 TREC topics, in the original term space (that is, the union of terms in all labelled training documents for a given topic), and in almost all cases this resulted in classifying all documents as negative. Moreover, support vector machines, which, at least theoretically, are the most powerful classification methods, also gave bad results. Based on an analysis of local observations around the error points obtained from these experiments, we devised the hypothesis that the main cause of the bad results was the aforementioned bias in data. Additionally, the SVM results pointed out that the regions most affected by this were located in the so called weak margin

region (“weak” because the classes are usually overlapping). In the SVM methods, the width of the margin is identified by the parameter C and the discriminant hyperplane is found in this region. The exact position of this hyperplane is determined by the intercept b in the equation $f(x) = (w, x) + b$, where (w, x) is the inner product of the vectors w and x (w is the vector normal to the discriminant hyperplane, x is an observed object).

Subsequently, we made the assumption that one can obtain a good result by trading a large number of erroneously classified positive examples for a small number of allowed errors on the negative class, in a context where classification mistakes are not equally evaluated. Such performance criteria are application specific and are commonly found in practice. Our goal is based on the idea that not retrieving a positive example carries a heavier penalty than erroneously retrieving a negative one. To make practical use of this assumption, we observed that it is feasible to efficiently construct a modified support vector machine. Then, in training an SVM classifier, we focused on changing the parameters C and b that affect the position of discriminant rules in the margin region. Such changes, however, do not necessarily improve the overall accuracy of the classifier, but bias the discriminant function towards the class considered more important.

Several authors [5, 6] describe how the parameters can be changed to improve the results of the SVM method, though their motivation was more general than ours. Classical theoretical results on SVM propose C as a free parameter and assume that any overlapping classes used for training can be separated by a sufficiently large C . Subsequently, the results allow the possibility of devising procedures to determine a good interval for the estimation of b . However, none of these methods take into account the bias specific to the class distribution mentioned above. For this reason, the available literature proposes some heuristics based on assumptions that the class distributions of the margin are Gaussian [5]. The method we describe in this paper does not rely on any assumption about these distributions and is only based on direct estimation of the aforementioned bias.

Two common measures of classifier performance are **sensitivity** and **specificity**. Sensitivity (also known as **recall**) is the fraction of positive examples that are correctly retrieved, while specificity is the fraction of negative examples that are correctly retrieved. The former measure describes the effectiveness of the classifier at finding the examples of interest, while the latter characterises the performance of the classifier at discarding the other examples. Our goal is to develop a new SVM method which yields better recall values on problems where a traditional SVM approach fails, while still maintaining good specificity of the system.

The paper is structured into four sections, of which this introduction is first. The second one formalizes the new adaptive SVM method. The third section includes a description of the test which we chose to estimate the results of this new learning procedure and an illustration of the results that were obtained. The last section addresses three questions we considered relevant, namely: (1) applied value of the proposed approach, (2) possibilities of improvement of the method, and (3) opportunities for generalization.

2. AN ADAPTIVE METHOD FOR FINDING A MARGIN FOR A SEPARATION
HYPERPLANE

2.1. Estimating an effective weak margin. Our basic goal was to define an optimal position for the discriminant function (the above hyperplane in the SVM method) within the chosen margin (the goal is related to our assumption about the data bias). The critical requirement that we have used and that helped to achieve strong success was that all criteria controlling the parameter adaptation process were estimated on the same training data on which the optimal hyperplane was constructed.

We have proposed and used in all experiments the following procedure for adapting the value of C .

Starting from a default value of $C_{def} = \left(\overline{\|x\|}\right)^{-1}$, as suggested in [7], we consider as target set the examples in the class with fewer representatives. If the positive and negative sets have the same number of examples, we choose the positives as target set.

Then, for every element k of a pre-defined sequence of increasing positive integers, we train a linear SVM classifier with a weak margin of $C_k = k \cdot C_{def}$. We test this classifier on the training set and calculate the recall on the target class. The procedure stops increasing k when the calculated recall falls within a pre-defined range $[\rho_{min}, \rho_{max}]$ (in our experiments, we used the range $[0.3, 0.75]$, which we considered reasonable).

For the following steps, we fix the value of C^* to be the last explored C_k . If the recall yielded by the classifier trained with the last C_k is equal to 1, we fix C^* to be C_{k-1} , instead. We call the selected C^* effective. The pseudo-code for this procedure is presented in algorithm 1.

Algorithm 1 find an effective C

Require: list of examples, X , ρ_{min} , ρ_{max} , k_{max}

- 1: $C_1 \leftarrow \left(\frac{1}{|X|} \sum_{x \in X} \|x\|\right)^{-1}$
 - 2: train classifier Γ_1 on X , using C_1 , test on X
 - 3: $\rho_{min} \leftarrow \rho_1 \leftarrow \text{recall}(\Gamma_1)$
 - 4: $k \leftarrow 1$
 - 5: **while** $k \leq k_{max}$ and $\rho_k < \rho_{max}$ **do**
 - 6: $k \leftarrow k + 1$
 - 7: $C_k \leftarrow k \cdot C_1$
 - 8: train classifier Γ_k on X , using C_k , test on X
 - 9: $\rho_k \leftarrow \text{recall}(\Gamma_k)$
 - 10: **end while**
- Ensure:** $C^* = C_k$, ρ_k , $\rho_{min} \leq \rho_k < \rho_{max}$
-

Taking into account that C is a critical factor in selecting an optimal hyperplane (optimality of the SVM's hyperplane is tied to the chosen C), and that the training data provides information about class overlap, we found that it is necessary to limit the accuracy obtained from the procedure of adapting C . This is the reason we used the two parameters ρ_{min} and ρ_{max} that determine the range available for C .

2.2. Determining an optimal intercept using only training data. The theory of SVM shows that for the chosen space, the hyperplane can be defined through the maximization of a quadratic objective function, which is a function of the coefficients of the following hyperplane:

$$(1) \quad f(x) = (x, w) + b,$$

where (u, v) is the inner product of the vectors u and v [1].

The decision rule based on this function has a standard form:

$$y = \text{sign}(f(x)),$$

with $y = +1$ meaning that x is assigned to the positive class, and, conversely, $y = -1$ meaning that x is assigned to the negative class.

Now let us consider the objective function whose maximum point defines the directed vector w and the intercept b of the hyperplane of separation, according to the description by C.J.C. Burges [8].

In the following, we denote the training data by a sequence of labelled points $\{x_i, y_i\}, i = 1, 2, \dots, l$, with $y_i \in \{-1, +1\}$. The objective function is given by the formula:

$$(2) \quad L_D = \sum_i \alpha_i - 1/2 \sum_{i,j} \alpha_i \alpha_j y_i y_j (x_i, x_j),$$

subject to $0 \leq \alpha_i \leq C$, and $\sum_i \alpha_i y_i = 0$.

The optimal vector α (of Lagrange multipliers) defines the desired vector w by the formula

$$w = \sum_{i=1}^{N_s} \alpha_i \cdot y_i \cdot x_i,$$

where N_s is the number of so-called support vectors, whose indices are the non-zero components of the optimal vector α . The intercept b can be calculated from any of the equations

$$(3) \quad \alpha_i \{y_i [(x_i, w) + b] - 1\} = 0$$

whose index i is also found in the inequalities $0 \leq \alpha_i \leq C$.

It is important to recall some basic facts that help specify our goal.

The optimal hyperplane for the case of overlapping classes was defined based on simple geometrical reasoning. Because the separable classes case dictates the optimal hyperplane by the system of inequalities

$$(4) \quad y_i ((x_i, w) + b) - 1 \geq 0 \quad \forall i = 1, 2, \dots, l$$

for the training data, V. Vapnik [2] proposed a modified version of the inequalities:

$$(5) \quad \begin{cases} (x_i, w) + b \geq 1 - \epsilon_i, & y_i = +1 \\ (x_i, w) + b \leq -1 + \epsilon_i, & y_i = -1 \end{cases},$$

where the additional slack variables are induced to compensate (regularize) the overlapping effect.

Using these new conditions he proposed to add to the direct criterion, L_D , a penalty function $C = \sum_i \epsilon_i$ to obtain a convex functional (above, we described the dual formulation of the problem). However, the above analysis, we think, gives a

good picture of why in the case of overlapping one can perturb the intercept b to obtain more appropriate results. Two problems appear to be taken into consideration: where to find a feedback signal to estimate the perturbation (particularly if one doesn't have extra data for testing), and what should be the additional criterion to optimize.

The particular quadratic programming problem to which the SVM method reduces the process of finding the optimal directed vector w and the intercept b is based on an efficient method of solution [9] (and, moreover, on the efficient industrial solver [7]). Thus, the SVM method was found efficient not only for considerations of accuracy, but also from another practical point of view: the existing software works with most types of data available [10, 11, 12]. If one assumes that classes are separable by a hyperplane, the optimal hyperplane has a very clear interpretation: it divides the margin between two classes of training points in order to get the location most distant from both classes. The location is determined by the intercept, calculated on the known optimal directed vector w . In such a situation, there is no need for our procedure.

However, if the classes overlap, the location of the corresponding discriminant hyperplane depends on the frequencies of each class in the training data. This dependence becomes critical when combined with a significant difference in probabilities of comparable classes, because objects found inside the margin region are more likely to be classified erroneously. Moreover, the disproportion of class probabilities in the margin region can incorrectly compensate for the cost of classification errors. It is often found in practice that positive examples have much smaller probability than negative ones, while the cost of mis-classifying a positive example is higher than mis-classifying a negative one.

The classifier built by most SVM software packages [7] using their default values of parameters is completely determined from the known vector w , noting that the intercept b is calculated with the assumption that the "price" of mistakes made on the overlapping points is equal for both classes. Fortunately, one can change the intercept b without affecting the optimal solution determined by the vector w . This represents a translation of the optimal hyperplane of separation, keeping the margin between classes, but modifying the cost of mistakes for points from different classes. It is important to mention that the perturbation can be done on the same training data, because the structure of errors in the training process carries enough information to make the changes.

To account for the limitations described above, which come from the usage of the default parameters of the SVM software packages, we propose the following procedure, which optimizes the weak margin C and the intercept b :

- Use the standard SVM method to build the optimal hyperplane.
- Set C to the C^* resulting from algorithm 1, maintaining a small number of mistakes in the training data, which will be later reduced by optimizing b .
- Calculate the confusion matrix of results (obtained on the training data) and estimate the performance of the classifier. Letting TP be the number of correctly classified positive examples, FN the number of positive examples classified as negatives and, similarly, TN the number of negatives classified as negatives and FP the number of negatives classified as positives, we

define the following quantities:

$$(6) \quad \begin{aligned} \alpha_P &= \frac{TP}{TP + FN} \\ \alpha_N &= \frac{TN}{TN + FP} \end{aligned}$$

These are called the **sensitivity** (also known as **recall**) and **specificity**, respectively.

- We propose to use $\alpha_N^* = 0.85$ as a lower bound for α_N because, in our opinion, it is a strong signal that the classifier performs poorly on negative documents. It is worth mentioning that the value of α_N^* is entirely application specific.
- Having fixed the weak margin to C^* , search for the intercept b in order to obtain the largest sensitivity, α_P , while meeting the specificity constraint $\alpha_N \geq \alpha_N^*$. Our search consists of successive changes of the intercept b , re-calculating the classifier, using the newly determined hyperplane of separation, $f(x) = (w, x) + b$, and the estimates of performance specificity (α_P) and sensitivity (α_N). The search stops when either all positive examples have been retrieved (thus achieving a sensitivity of 100%) or when the specificity (α_N) drops below the set bound, α_N^* .

We denote by b^* the intercept found by this procedure. Then, the final classifier will be determined by the hyperplane $f^*(x) = (w, x) + b^*$. The performance of this classifier can be evaluated using some standard cross-validation procedure (or directly on some particular test set).

3. EXPERIMENTAL RESULTS

As illustration of our procedure, we performed an experimental study on text data set up for TREC 2002. The data set consisted of the Reuters RCV1 collection [4], i.e., Reuters newswire stories from 20 August 1996 through 19 August 1997, accounting for a total of 806,791 documents. For TREC purposes, 100 topics were identified and for each topic there were a number of labelled documents, On average, there were some 800 labelled documents per topic, of which 90 were positive.

From these 100 topics we selected those for which the participants at the TREC 2002 batch filtering track submitted poor results. Namely, we selected those topics for which the best recall attained by the submitted classifiers was less than or equal to 0.5, accounting for a total of 11 topics.

The sizes of the data sets for the selected 11 topics are presented in table 1, along with some basic statistics on these sets. There is a striking discrepancy between these topics and the rest of the data, because the average number of examples is 418.5, approximately half of the size of training data for the other topics. The scarcity of the training data could well be the reason for the poor performance of the classifiers submitted at TREC 2002. We considered the 11 two-class problems associated with these topics. For each problem, t , we split the sets of labelled documents pertaining to t into two halves, using stratified random sampling, that is, randomly sampling the given data such that the proportions of the two classes are maintained.

Topic	n	n^+	n^-
R151	437	22	415
R154	469	39	430
R155	489	63	426
R161	463	47	416
R171	394	68	326
R176	411	37	374
R179	510	32	478
R180	426	72	354
R184	361	13	348
R192	367	29	338
R200	277	86	191
avg	418.5	46.2	372.4

TABLE 1. Labelled document set sizes for the selected topics. n is the size of the set of examples, n^+ and n^- are the numbers of positive and negative examples, respectively. $n = n^+ + n^-$.

Topic	aiSVM				SVM				1-NN			
	TP	FN	FP	TN	TP	FN	FP	TN	TP	FN	FP	TN
R151	3	19	5	410	1	21	0	415	6	16	13	402
R154	19	20	7	423	10	29	3	427	19	20	19	411
R155	33	30	31	395	11	52	6	420	34	29	23	403
R161	22	25	10	406	13	34	4	412	21	26	8	408
R171	9	59	6	320	2	66	0	326	20	48	51	275
R176	28	9	10	364	16	21	0	374	23	14	3	371
R179	6	26	1	477	3	29	1	477	16	16	7	471
R180	63	9	13	341	46	26	3	351	48	24	6	348
R184	2	11	1	347	1	12	0	348	3	10	5	343
R192	11	18	1	337	4	25	1	337	12	17	3	335
R200	70	16	9	182	59	27	8	183	70	16	21	170

TABLE 2. Confusion tables for the selected topics, based on results from three tested classifiers (aiSVM is the adaptive intercept SVM, SVM is the classifier calculated by svmLight using the default parameters, and 1-NN is the 1-nearest neighbor calculated in the original space). The results from 1-NN were calculated using leave-one-out cross-validation, thus having an advantage by using more training data.

The performance of the classifier we built was evaluated using 2-fold cross-validation, namely, training on one half of the data at a time and testing on the remaining half, followed by repeating the process with the two halves interchanged.

As emphasized in the introduction, our goal was to estimate the improvement in the results yielded by the new procedure, in comparison with the results of other standard procedures. In this respect, we compared our results with those obtained from 1-nearest-neighbour (1-NN) and the standard SVM with linear kernel [7]. Our modified SVM is also based on a linear kernel.

In table 2 we present the confusion tables obtained from using the three algorithms on all selected 11 topics. In the first column we present the results of our algorithm, in the second column, the results of SVM using default parameters and in the third column the results of 1-NN, via leave-one-out cross-validation. We consider that the results obtained from 1-NN characterize the difficulty of the problem. For instance, topics R184 and R151 constitute the most difficult problems of the set, as measured by the number of correctly identified positive examples (column TP), whose retrieval we strived to achieve. The easiest topic is R200. In the same table, again as measured by TP, it can be observed that our modified SVM method comes in a close second after the 1-NN in the sense that in most examples, 1-NN has the highest TP with our modified SVM method coming in second, while in some examples the modified SVM method has the highest TP. We think that this result is due to the fact that 1-NN used almost the entire training data for classification purposes, while both SVM methods used a randomly selected half of the available labelled data.

Additionally, we provide in table 3 the sensitivity (α_P) and specificity (α_N) coefficients, which present a normalized view of the results presented in table 2.

From Table 3, we conclude that our procedure improves the recall of the standard SVM classifier, while maintaining good specificity. Indeed, the improvement of recall over the standard SVM is on average by a factor of 2.25, with a minimum of 1.17. At the same time, the specificity of the classifier was degraded by at most a factor of 0.03 of the value for SVM, being always above 0.95. Also, the recall of our classifier is better than those submitted to TREC, achieving a maximal recall of 0.88 (compared with 0.5 in TREC) and an average recall of 0.5 (compared with 0.29 in TREC).

Topic	aiSVM		SVM		1-NN	
	α_P	α_N	α_P	α_N	α_P	α_N
R151	0.14	0.99	0.05	1.00	0.27	0.97
R154	0.49	0.98	0.26	0.99	0.49	0.96
R155	0.52	0.93	0.17	0.99	0.54	0.95
R161	0.47	0.98	0.28	0.99	0.45	0.98
R171	0.13	0.98	0.03	1.00	0.29	0.84
R176	0.76	0.97	0.43	1.00	0.62	0.99
R179	0.19	1.00	0.09	1.00	0.50	0.99
R180	0.88	0.96	0.64	0.99	0.67	0.98
R184	0.15	1.00	0.08	1.00	0.23	0.99
R192	0.38	1.00	0.14	1.00	0.41	0.99
R200	0.81	0.95	0.69	0.96	0.81	0.89

TABLE 3. Tables of values of sensitivity (α_P) and specificity (α_N) coefficients, based on results from three tested classifiers (aiSVM is the adaptive intercept SVM, SVM is the classifier calculated by svmLight using the default parameters, and 1-NN is the 1-nearest neighbor calculated in the original space).

4. CONCLUSIONS

The idea of substituting one type of error for another is a classical approach in statistics when the cost of these two types of errors are very different [13]. The literature related to trained classifier design contains very little published material on general procedures of achieving the desired compromise [14] and we consider our work to be a contribution to this topic. We attained the goal of finding such a procedure based on a modification of a linear SVM method and its parameters C and b .

We evaluated our performance on 11 TREC topics for which best results submitted to the TREC 11 batch filtering track obtained recalls of less than 50%, with an average of 29%. We show that our method attains recalls of as much as 88% with an average recall of 50%. The smallest specificity obtained was 0.95, well above our lower bound of 0.85.

It would be interesting to investigate the quality of the results of our procedure when applied to different types of data.

Because the method can work without any change in any linear vector space in which one built the SVM optimal classifier, in principle the method can be generalized for design of non-linear classifiers which have equivalent representation in a particular transformed space by using an appropriate kernel function [15, 16].

ACKNOWLEDGEMENTS

The authors are grateful to Dr. Fred Roberts for thoroughly editing the paper and for his very helpful comments.

This research was supported by the National Science Foundation under Grant Number EIA-0087022. The views expressed in this article are those of the authors, and do not necessarily represent the views of the sponsoring agency.

REFERENCES

- [1] Vladimir Naumovich Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.
- [2] Bernhard E. Boser, Isabelle Guyon, and Vladimir Vapnik. A training algorithm for optimal margin classifiers. In *Computational Learning Theory*, pages 144–152, 1992.
- [3] Ellen M. Voorhees and Donna Harman. Overview of trec 2001. In *Proceedings of the 2002 Text REtrieval Conference*, 2001.
- [4] Reuters corpus 1996-08-20 to 1997-08-19.
- [5] N. Cancedda, C. Goutte, J.-M. Renders, N. Cesa-Bianchi, A. Conconi, Y. Li, J. Shawe-Taylor, A. Vinokourov, T. Graepel, and C. Gentile. Kernel methods for document filtering. In *Proceedings of the Eleventh Text REtrieval Conference*, 2002.
- [6] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*. Morgan Kaufmann, 2000.
- [7] Thorsten Joachims. Making large-scale support vector machine learning practical. In B. Scholkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, 1998.
- [8] C. J. C. Burges and B. Schölkopf. Improving the accuracy and speed of support vector learning machines. *Advances in Neural Information Processing Systems*, 9:375–381, 1997.
- [9] E. Osuna, R. Freund, and F. Girosi. Improved training algorithm for support vector machines. In *Proc. of IEEE NNSP'97*, 1997.
- [10] Chih-Wei Hsu and Chih-Jen Lin. A simple decomposition method for support vector machines. *Machine Learning*, 46:291–314, 1999.
- [11] Stefan Rüping. *mySVM-manual*, 2000.

- [12] R. Collobert and S. Bengio. On the convergence of SVM Torch, an algorithm for large-scale regression problems. IDIAP-RR 24, IDIAP, 2000.
- [13] George Casella and Roger L. Berger. *Statistical Inference*. Brooks Cole, 2001.
- [14] Pedro Domingos. Metacost: A general method for making classifiers cost-sensitive. In *Knowledge Discovery and Data Mining*, pages 155–164, 1999.
- [15] M.A. Aizerman, E.M. Braverman, and L.I. Rozonoer. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837, 1964.
- [16] M.A. Aizerman, E.M. Braverman, and L.I. Rozonoer. The Robbins-Monroe process and the method of potential functions. *Automation and Remote Control*, 28:1882–1885, 1965.

DEPARTMENT OF COMPUTER SCIENCE, RUTGERS UNIVERSITY
E-mail address: `angheles@cs.rutgers.edu`

DIMACS, RUTGERS UNIVERSITY
E-mail address: `muchnik@dimacs.rutgers.edu`