
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. Many questions the intelligence community has to address are related to text classification problems, more precisely to the problem of designing classifiers that assign a new document to a few relevant categories (topics) from many hundreds of predefined topics.

The speed and accuracy of the designed classifiers are also important. Moreover, it is necessary to present the assignments made by the classifier in a form convenient for expert interpretation.

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 classifiers trained using the default values provided by SVM^{Light} (1). In this paper we propose a novel approach to optimize these parameters by identifying the set of support vectors in the training data and training a new classifier using only these support vectors. Additionally, we take into consideration that the numbers of examples from each class are unbalanced, a common characteristic of text classification problems. For adjusting these parameters, we employed a local cross-validation scheme inside the training process (2). We evaluated our performance on the 50 TREC 2002 assessor topics and we compared our results with those of a linear SVM classifier trained using the default parameters provided by SVM^{Light} . On average, we obtained an improvement in sensitivity of approximately 60%, at a price of approximately 7% drop in specificity.

1. INTRODUCTION

Many questions the intelligence community has to address are related to text classification problems, more precisely to the problem of designing classifiers that assign a new document to a few relevant categories (topics) from many hundreds of predefined topics. The design has to be based on an analysis of a small number of examples of documents with known assignments (possibly in incomplete form). For any given topic, this commonly leads to having many more examples of non-relevant documents than relevant ones.

The speed and accuracy of the designed classifiers are also important. Moreover, it is necessary to present the assignments made by the classifier in a form convenient for expert interpretation.

To meet these requirements we adapted the well known method of Support Vector Machines (3) such that we address the problem of the mentioned imbalance of examples in the training data and the requirement of manual interpretation convenience. The latter restricted our work to using only linear SVM classifiers. To compensate for the imbalance of examples in the training data we devised an adaptive algorithm to change the intercept, b .

Unbalanced training sets can arise for two reasons. First, it is common that interesting classes of documents correspond to a small proportion of all available documents. (Indeed, the class often wouldn't be interesting otherwise.) Second, regardless of the frequency of the class in the population as a whole, the procedure by which a subset of the data is labeled may lead to a distribution of class members different than in the population. Often both these effects apply, with one partially, but not completely counterbalancing the other.

For instance, the Reuters RCV1 news stories corpus (4) is a standard testbed used for evaluating information retrieval algorithms. For this corpus, at TREC 2002 (5) were defined 50 assessor topics. For each such topic, there are available some 10-400 positive examples and some 100-1500 negative ones. The class frequencies here result from very low frequencies of positive examples in the population (1 in 1000 or less), partially compensated by strategies that oversampled positive examples. 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.

For example, we ran a widely used learning algorithm (1-nearest neighbor) on training data for the aforementioned 50 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 some of 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 (6; 7) 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 . It is noteworthy that C is a factor which provides a limited control over the VC dimension of a problem. Accordingly, an increase of the parameter C leads to an increase of the VC dimension, thus decreasing the capacity of generalization of the designed classifier.

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 (6). 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.

In this paper we use three common measures of classifier performance: **sensitivity**, **specificity** and **F1**. 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. Another popular measure is the **precision**, defined as the fraction of true positive examples among all examples the classifier labelled as positives. The F1 measure is a weighted harmonic mean of the precision and recall (8). 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.

$$\begin{aligned}
 \zeta &= \frac{TP}{TP + FN} && \text{(sensitivity)} \\
 \eta &= \frac{TN}{TN + FP} && \text{(specificity)}. \\
 F1 &= \frac{2 \cdot TN}{2 \cdot TN + FP + FN}
 \end{aligned}
 \tag{1}$$

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. According to Vapnik's work (3), when classes are not separable one has to add to the basic criterion ($\min \frac{1}{2} \|w\|^2$) a regularization penalty, expressed as a sum of slacks: $C \cdot \sum \varepsilon_i$. Computationally, this problem is expressed as:

$$\begin{aligned}
 \min & \frac{1}{2} \|w\|^2 + C \cdot \sum_{i=1}^l \xi_i, \text{ subject to} \\
 \forall i = 1, \dots, l & \begin{cases} \xi_i \geq 0 \\ y_i[(x_i, w) + b] \geq 1 - \xi_i, \end{cases}
 \end{aligned}
 \tag{2}$$

where l is the size of the training data set.

In this criterion the first term estimates an upper-bound of the VC-dimension. The second measures the degree of contradiction between the calculated assignments and the existing class overlaps. Geometrically, this means that the second term determines a region around the hyperplane of separation, which encloses all incorrectly classified examples. This region is defined by two hyperplanes parallel to the hyperplane of separation. It follows that a narrow region would lead to a large number of classification errors with the training data. Similarly, for a very wide region, the number of support vectors becomes very large, and the number of errors on training data is greatly reduced, but the obtained model overfits the training data. If one considered only the width of the margin C as optimization criterion, the problem would be reduced to finding the best trade-off between generality and accuracy of the

calculated classifier. In other words, one could attempt to find an optimal hyperplane of separation by changing only the width of the margin region, C .

In practice, useful values of C are difficult to estimate, and several attempts have been documented (3; 9). Such estimates of C are far from optimal. One method for determining the performance of a classifier built using a given weak margin, is to evaluate the accuracy of the classifier using additional labelled test data. Our approach consists of using a cross-validation method, in order to determine which parameters have the most chances of providing a good discrimination of the test data. Prior to this local validation step, we reduce the training data to the set of its support vectors, and we estimate the value of C using formula (3) recommended in (3). A related approach for estimating good values of C was proposed in (10).

We consider the modification of the intercept a process that substitutes one type of error for another. In our case, we trade a small number of erroneously classified positive examples for a large number of negative examples.

The solution we propose consists of using a sequential optimization strategy: partially adjust C to obtain a sufficiently accurate classifier, followed by an optimization of b in order to find the best trade-off between errors in assignments of labels. In other words, our heuristic works such that a large number of negative class errors on the training data can be substituted by a very small number of positive class errors if on the first stage (by manipulating C) we obtained a classifier that would minimize the number of positive class errors. By changing the intercept, b , we try to improve the accuracy of the final classifier, trading a large number of negative examples for a very small number of positives.

As suggested in (3), a reasonable initial value for the weak margin, C , is the inverse of the average norm of the examples available for training:

$$(3) \quad C_{def} = \frac{1}{\|x\|^2} = \frac{l}{\sum_{i=1}^l \|x_i\|^2},$$

where $\{x_1, \dots, x_l\}$ are the training examples, and $\|x\|$ represents the Euclidian norm.

We believe that the margin region would be better estimated if we would train a classifier using only information from the support vectors. In particular, we consider that calculating C using formula (3) applied to the set of support vectors, would be better than the one calculated with the same formula applied to the entire training data. Since the support vectors are known only *a posteriori*, we propose the following procedure for determining C : given a training data set $X = \{x_1, \dots, x_l\}$, we begin with building a linear SVM classifier using the default parameters of *SVM^{Light}*, a publicly available software package for training and applying SVM classifiers (1). From the resulting classifier, we select the support vectors¹ and use them as training

¹These support vectors are only a good approximation of the ideal set of support vectors. In theory, this set of support vectors could be calculated by training an SVM using a specific set of

data, repeating the process. This yields two values for the parameter C . Using additional labelled data (i.e. data which was not used for training), $Z = \{z_1, \dots, z_k\}$, we estimate the efficiencies of the classifiers built using these two values of C . Using some performance measure, π , (such as $F1$ (8), whose formula was presented in (1)), we select the value which yields better performance and denote it by C^* .

Algorithm 1 Find an effective C

Input: $X = \{x_1, \dots, x_l\}$, k_{in} , π

- 1: $(X^1, X^2, \dots, X^{k_{in}}) \leftarrow \text{stratifiedSampling}(X, k_{in})$
- 2: **for** $f = 1$ to k_{in} **do**
- 3: $Z \leftarrow X^f$
- 4: $T \leftarrow X \setminus Z$
- 5: $C_1^f \leftarrow \frac{|T|}{\sum_{x \in T} \|x\|^2}$
- 6: train classifier Γ_1^f on T , using C_1^f
- 7: $\pi(C_1^f) \leftarrow \Gamma_1^f(Z)$
- 8: $SV \leftarrow SV(\Gamma_1^f)$
- 9: $C_2^f \leftarrow \frac{|SV|}{\sum_{x \in SV} \|x\|^2}$
- 10: train classifier Γ_2^f on SV , using C_2^f
- 11: $\pi(C_2^f) \leftarrow \Gamma_2^f(Z)$
- 12: $C_f^* \leftarrow \arg \max\{\pi(C_1^f), \pi(C_2^f)\}$
- 13: **end for**
- 14: $C^* \leftarrow \arg \max\{\pi(C_1^*), \dots, \pi(C_{k_{in}}^*)\}$

Output: C^*

In order to obtain labelled evaluation data, we split the available training data into k_{in} folds. Using a stratified sampling procedure, we divided the training data set X into k_{in} subsets, $(X^1, \dots, X^{k_{in}})$. The stratified sampling insures that the proportions of different classes in training data are reflected in the k_{in} subsets. For each fold, f , we obtained two candidate values of C : C_1^f, C_2^f . For each of the values C_1^f, C_2^f , we calculated a linear SVM on the same training data that was used for calculating C_1^f, C_2^f . Each classifier was then evaluated on the complementary part of data (denoted by Z in Algorithm 1). We used $F1$ (8) as a measure of the performance of each classifier. Considering that sizes and distributions of the testing data sets are quasi-equal, the two values of $F1$ are directly comparable. From C_1^f, C_2^f we selected as C_f^* the one

parameters. The existing theoretical background and practical evidence do not provide sufficient information for determining these parameters.

which yielded the better classifier (i.e. the larger $F1$ measure). Each fold yielded one such value of C . From the set of winners of the first round, $\{C_1^*, \dots, C_{k_{in}}^*\}$, we selected the one that obtained the best $F1$ measure on its corresponding validation fold and defined it as C^* . The pseudo-code of this procedure is presented in Algorithm 1.

2.2. Determining an optimal intercept using only training data. If the classes overlap, the location of the discriminant hyperplane generated by SVM^{Light} with default parameters 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 erroneously classified. Moreover, the difference 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 lower frequency than negative ones, while the cost of mis-classifying a positive example is higher than mis-classifying a negative one.

In the dual formulation of the SVM training process, the classifier is calculated by solving an optimization problem. The solution consists of the hyperplane of separation, w , and an intercept, b . When using SVM^{Light} with default parameters, the intercept, b , is calculated with the assumption that the “price” of mistakes made on the overlapping points is equal for both classes. Let us explain why one can change the intercept b without affecting the optimal solution determined by the vector w .

A linear SVM classifier based on the minimization of Vapnik’s VC-dimension bound criterion consists of two components: a discriminant hyperplane, w and a scalar (intercept), b . The optimization procedure yields an optimal value only for w , while the constant b is determined not uniquely from a set of feasible values, determined from a predefined set of equalities. One could search the set of these feasible values and determine the one that insures the highest sensitivity while keeping the specificity above some acceptable threshold. Computationally, this approach would be very expensive, due to the structure of this set of feasible values. At the price of having more training data available, one could circumvent this expensive step. In these circumstances, we think that altering the intercept, b , would maintain the optimality of the discriminant hyperplane, w . Such calculations are not very practical, due to the paucity of training data, and one should apply some resampling procedure, which, on its turn would be expensive. Considering all these aspects, we investigated experimentally how such a search procedure could work using only the available training data, in other words, ignoring the requirement of independence of training and test data. We found that such risk-seeking heuristic works well and we present its results in this paper.

Changing the intercept, b , 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 ².

Algorithm 2 Find an effective b

Input: $X = \{x_1, \dots, x_l\}, \{y_1, \dots, y_l\}, w, \eta_{min}, \varepsilon$

1: $X^- \leftarrow \{x_i | x_i \in X \wedge y_i = -1\}, |X^-| = l^-$

2: $\Delta^- \leftarrow (\delta_1, \dots, \delta_{l^-}), \forall i \delta_i = (x_i, w)$

3: order the elements in X^- by their corresponding values in Δ^-

$$\Rightarrow \begin{cases} X_{sorted}^- = \langle x_{i_1}, \dots, x_{i_{l^-}} \rangle \\ \delta_{i_1} \geq \delta_{i_2} \geq \dots \geq \delta_{i_{l^-}} \end{cases}$$

4: $k \leftarrow 1$

5: $stop \leftarrow false$

6: **while** $stop \neq true$ **do**

7: $b_k \leftarrow \delta_{i_k}$

8: $\eta_K \leftarrow \frac{l^-(k-1)}{l^-}$

9: calculate ζ_k using w and b_k

10: $stop \leftarrow (\eta_k < \eta_{min}) \vee (\zeta_k = 1)$

11: **end while**

Output: $b^* = b_k - \varepsilon$

We propose the following algorithm, which optimizes the weak margin C and the intercept b :

- Use SVM^{Light} with default parameter settings to build the hyperplane of separation.
- Set C to the C^* resulting from Algorithm 1.
- Run the classifier trained using C^* with default value of b , back over the training data. Calculate the confusion matrix of results (obtained on the training data) and estimate the performance of this classifier. Let 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.
- Based on the properties of the data analyzed in our experiments (described in section 3), namely the much larger number of negative examples, we found that it is unacceptable to allow the specificity of a classifier to drop below a

²It is clear that the change of b will also change the value of the criterion (2), because it would change the conditions under which (2) is minimized. In other words, with a different value of b , one could find a smaller value of (2). This, however, is not the goal of our work, where we prefer to maintain the hyperplane of separation (along with it, the original margin) and only change the discriminating rule within.

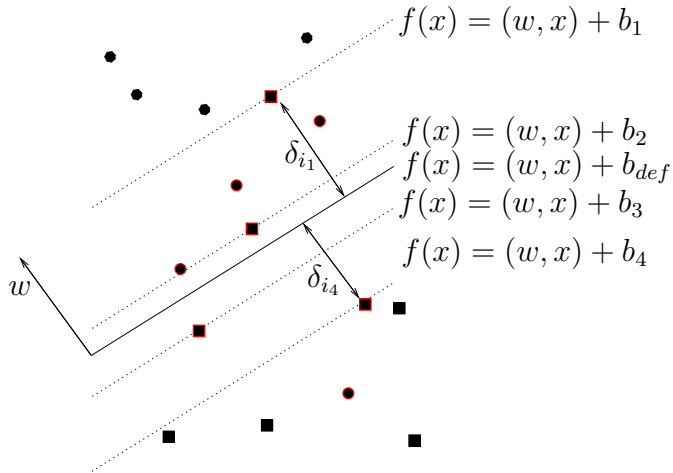


Figure 1: Adjustment of the intercept b . The two classes of objects are denoted by round and square dots, respectively. The hyperplane of separation is successively shifted such that it misclassifies exactly $0, 1, 2, 3, \dots$ examples from the square class. For each such position one can calculate the performance of the classifier and choose the best obtained result. The quantities $\delta_{i_1}, \dots, \delta_{i_4}$ were calculated as specified in Algorithm 2, lines 2 and 3. In order to reduce the complexity of the figure, we did not depict δ_{i_2} and δ_{i_3} . The solid line, labelled with the equation $f(x) = (w, x) + b_{def}$ represents the classifier learnt using the default training parameters of SVM^{Light} .

threshold. We propose to use $\eta^* = 0.99$ as a lower bound for η because, in our opinion, lower values are a strong signal that the classifier performs poorly on negative documents. It is worth mentioning that the value of η^* is entirely application specific.

- Having fixed the weak margin to C^* , search for the intercept b in order to obtain the largest sensitivity, ζ , while meeting the specificity constraint $\eta \geq \eta^*$. Let X^- be the set of negative examples and l^- its cardinality. We ordered the elements of X^- by their distances to the separation hyperplane: $\delta_i = (x_i, w) \forall x_i$ s.t. $y_i = -1$ (as previously defined, (x, w) represents the inner product of the vectors x and w). Let $\{x_{i_1}, \dots, x_{i_{l^-}}\}$ be the set of negative examples, ordered descendingly by their distances to the separation hyperplane. We, then, scan this ordered set, setting b such that at each step j the hyperplane of separation passes through point x_{i_j} . The procedure stops when either all positive examples have been correctly classified or when the specificity falls below the set threshold. In order to insure the correct classification of the last visited negative example, we subtract from the determined b a very small positive amount, denoted as ε . For practical purposes, we set this value to 10^{-300} , which is slightly larger than the smallest positive number

that can be represented in floating point double precision. The pseudo-code is presented in Algorithm 2, accompanied by figure 1, which is a graphical representation of the algorithm.

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). The pseudo-code for the algorithm of building a classifier is presented in Algorithm 3.

Algorithm 3 Calculate a classifier using algorithms 1 and 2

Input: $Train, Test, k_{in}, \pi, \eta_{min}, \varepsilon$

- 1: $C^* \leftarrow \text{Algorithm1}(Train, k_{in}, F1)$ {optimize C }
- 2: $[w, b] \leftarrow \text{svmLearn}(Train, C^*)$ {optimize b }
- 3: $b^* \leftarrow \text{Algorithm2}(Train, w, \eta_{min}, \varepsilon)$

Output: $[w, b^*]$

3. EXPERIMENTAL RESULTS

As illustration of our procedure, we performed an experimental study on the 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.

From these 100 topics we selected the 50 available assessor topics, identified by the labels R101 to R150. In table 1 we present a summary of the available training data and an excerpt from the list of topics. The complete table of topic sizes is presented in Appendix A.

We considered the 50 two-class problems associated with these topics. For each problem, t , we split the sets of labelled documents pertaining to t into 5 parts, using stratified random sampling, that is, randomly sampling the given data such that the proportions of the two classes are maintained.

The performance of the classifier we built was evaluated using 5-fold cross-validation, namely, training on four parts of the data at a time and testing on the fifth, followed by repeating the process until each part was used as testing data (11). In order to increase the confidence of the obtained results, we performed each such experiment 3 times, each time with a different random split of the data. This process is presented in Algorithm 4 and described in more detail in the following paragraph.

On each fold of the cross-validation procedure presented in the previous paragraph, we used three data sets for the purposes of training, evaluation and testing. As

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. $n = n^+ + n^-$. The average values presented in the last line are calculated on all 50 topics we analyzed. The complete table of cardinalities is presented in Appendix A.

| Topic | n | n^+ | n^- |
|----------|--------|--------|--------|
| R101 | 1634 | 318 | 1316 |
| R102 | 866 | 204 | 662 |
| R103 | 1355 | 82 | 1273 |
| R104 | 903 | 98 | 805 |
| R105 | 1267 | 157 | 1110 |
| R106 | 1227 | 45 | 1182 |
| R107 | 1509 | 48 | 1461 |
| R108 | 1097 | 16 | 1081 |
| R109 | 814 | 77 | 737 |
| R110 | 1498 | 59 | 1439 |
| ⋮ | | | |
| avg | 1233.2 | 101.2 | 1132 |
| σ | 195.19 | 122.36 | 206.82 |

obtained from the 5-fold cross-validation, the training and evaluation data represents 80% of the available labelled data and the remaining 20% are used for testing. In the pseudo-code shown in Algorithm 4, this division of the data corresponds to lines 2-5. Relative to the same algorithm, the value of the parameter k_{out} was set to 5. Subsequently, the training and evaluation data was used for choosing the value of the parameter C^* , as specified in Algorithm 1. In this algorithm, we applied a 2-fold cross-validation ($k_{in} = 2$), where one fold was used for training and the other for evaluating the performance of the classifiers built using the candidate C values, as specified in Algorithm 1. That is, at each step of the procedure we used 40% of the labelled data for training, another 40% for evaluating the quality of the determined parameters and 20% for testing. In particular, for each combination of training + evaluation and testing data sets, Algorithm 1 trained four classifiers. Let C^* denote the value calculated by Algorithm 1 (line 6).

With the found value, C^* , we construct a classifier using the training and evaluation data, in other words the 80% of data available for training in the validation process (line 7). Finally, the learning process adjusts the classifier by changing the intercept, b , using the training data from the current validation fold. Because the parameter b represents the threshold in the decision rule of the constructed linear SVM, we adjust it by searching for sensible values using labelled examples in the margin region. In particular, the margin region is identified by the linear SVM trained using C^* . We

Algorithm 4 experimental set-up

Input: $X = \{x_1, \dots, x_l\}, \{y_1, \dots, y_l\}, k_{out}, k_{in}, r, \eta_{min}, \varepsilon$

- 1: **for** $run = 1$ to r **do**
- 2: $(X^1, X^2, \dots, X^{k_{out}}) \leftarrow \text{stratifiedSampling}(X, k_{out})$
- 3: **for** $outFold = 1$ to k_{out} **do**
- 4: $Test \leftarrow X^{outFold}$
- 5: $Train \leftarrow X \setminus Test$
- 6: $[w, b^*] \leftarrow \text{Algorithm3}(Train, Test, k_{in}, F1, \eta_{min}, \varepsilon)$
 //evaluation of classifier
- 7: classify $Test$ using $[w, b^*]$
- 8: **end for**
- 9: **end for**

choose the intercept b which provides the best sensitivity while meeting a certain minimal specificity (η_{min} criterion). Our experimental results were obtained using a minimal specificity of 0.99 and we advise for values in the interval $[0.95, 0.995]$. Finally, the resulting classifier, described by the hyperplane of separation w and the intercept b^* , is evaluated on the testing data (line 9). Note, that the sensitivity and specificity are calculated on the training data. For this reason, we expect that results obtained on un-seen data will have lower specificity and this is why we recommend to use relatively large values for the minimum acceptable specificity ³.

Because the training and testing procedures are organized on a cross-validation scheme, the values found for C^* and b^* are relevant only to the current fold. The resulting values of the parameters were not combined in a single value which would characterize the labelled data. Instead, we considered that each cross-validation fold is associated with a distinct classifier, which is the natural result of a cross-validation scheme. To obtain a single classifier for the entire data set, one should use the procedures described in this section without the outer cross-validation.

As emphasized in the introduction, our goal was to estimate the improvement of classification results yielded by the new procedure, when compared with the results of other standard procedures. In this respect, we compared our results with those obtained from a linear SVM trained using the standard parameters suggested in (1). Our modified SVM is also based on a linear kernel.

In table 2 we present a comparative analysis of the performances of two algorithms evaluated in the environment described heretofore. For each algorithm we present as standard results the sensitivity (ζ), specificity (η) and the F1 measure. Only the

³ In general, strong results on training data (thus high specificities and sensitivities) cannot guarantee equally good results on testing data. Our heuristic is based on the assumption that the performance measures calculated on the training data should be strongly related (even if not tightly bound) to those obtained on testing data. In other words, we believe that the performance measure is a continuous function of b , in the margin region.

Table 2: A comparative analysis of the results obtained from two different procedures. **SVM-def** is the linear SVM built using the default parameters provided by *SVM^{Light}*. **aiSVM-Cb0.990** was obtained by adapting C , then b on SVM-def (using algorithms 1,2). Because of space constraints, only the first ten topics are presented in detail. The complete results are presented in Appendix B. The statistics (averages and standard deviations), are calculated on the entire set of 50 topics.

| Topic | SVM-def | | | aiSVM-optC-b0.990 | | |
|-------------|-------------------------|-----------------------|-------|-------------------------|------------------------|------|
| | sensitivity (ζ) | specificity(η) | F1 | sensitivity (ζ) | specificity (η) | F1 |
| R101 | 0.8 | 0.99 | 0.87 | 0.94 | 0.93 | 0.85 |
| R102 | 0.53 | 0.98 | 0.67 | 0.49 | 0.98 | 0.63 |
| R103 | 0.046 | 1 | 0.088 | 0.85 | 0.93 | 0.58 |
| R104 | 0.44 | 0.99 | 0.58 | 0.62 | 0.97 | 0.67 |
| R105 | 0.44 | 0.99 | 0.58 | 0.63 | 0.97 | 0.69 |
| R106 | 0 | 1 | 0 | 0.73 | 0.86 | 0.27 |
| R107 | 0 | 1 | 0 | 0.76 | 0.85 | 0.24 |
| R108 | 0 | 1 | 0 | 0.77 | 0.94 | 0.27 |
| R109 | 0.32 | 1 | 0.47 | 0.93 | 0.9 | 0.64 |
| R110 | 0.061 | 1 | 0.12 | 0.77 | 0.87 | 0.32 |
| avg | 0.15 | 0.99 | 0.19 | 0.74 | 0.92 | 0.47 |
| std | 0.24 | 0.032 | 0.27 | 0.14 | 0.039 | 0.18 |

results of the first 10 topics are presented in this table, along with the averages and standard deviations calculated over the entire set of 50 topics. The reader is referred to Appendix B for a complete set of results.

These two classifiers are: the linear SVM obtained from using *SVM^{Light}* with the default parameters and the classifier calculated using the procedure outlined in section 2 (that is, adapt C , then adapt b on the resulting classifier). It can be observed that the gain in sensitivity is on average 60%, at the price of approximately 7% loss in specificity. In absolute numbers, considering there are approximately ten times more negative examples, this accounts to trading one positive example for 1.2 negatives.

Also, the lower bound for specificity used in Algorithm 2, is not completely reflected in the results obtained from using the classifier on un-seen data. The results presented in table 2 show that only two topics among the first 10 yield specificities above 0.95 and none above 0.99. These results are not contradicted by the fact that we used 0.99 as minimum acceptable specificity in Algorithm 2. In the process of optimizing the intercept, b , the specificity is calculated on training data, while the results presented in table 2 are calculated on previously un-seen test data. We observed that the specificities obtained on training data are consistently 5%-7% below the minimum specificity threshold, η_{min} , suggesting a trend that could help in choosing a more appropriate threshold, where desired.

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 (12). The literature related to trained classifier design contains very little published material on general procedures of achieving the desired compromise (13; 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 the 50 TREC 2002 assessor topics and we compared our results with those obtained from a similar classifier trained using the default parameter values set by the software. We showed that the parameter values calculated using our procedure lead to an average sensitivity improvement of 60% and to a drop of 7% on specificity.

In future work, we would like to investigate the effect of using a more finely grained cross-validation scheme (which implies more training data) in the process of calculating C^* . We believe this aspect alone, although computationally expensive, would help estimate a better C^* , thus improve the classifier on which b is adapted, leading to better final results. This process, however, is likely to attain a maximum classifier performance, for some number of folds, beyond which extra computational power will not bring any improvement. Documenting the interaction of our algorithm with the size of folds would be useful.

It would be desirable to obtain experimental evidence of the effect of different representations of the data. In our current work, we used only term frequency and we would like to extend the results by using geometric and statistical normalizations of the data, as well as TFIDF (15).

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 (16; 17).

ACKNOWLEDGEMENTS

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

We owe a big debt of gratitude to Dr. David D. Lewis for his valuable remarks, which helped us write a vastly improved version of this paper.

The authors thank the KD-D group for its support through National Science Foundation grant number EIA-0087022 to Rutgers University. The views expressed in this

article are those of the authors, and do not necessarily represent the views of the sponsoring agency.

APPENDIX A.

Table 3: 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 | n | n^+ | n^- | Topic | n | n^+ | n^- |
|-------|------|-------|-------|----------|--------|--------|--------|
| R101 | 1634 | 318 | 1316 | R127 | 1164 | 48 | 1116 |
| R102 | 866 | 204 | 662 | R128 | 1080 | 70 | 1010 |
| R103 | 1355 | 82 | 1273 | R129 | 1373 | 71 | 1302 |
| R104 | 903 | 98 | 805 | R130 | 974 | 19 | 955 |
| R105 | 1267 | 157 | 1110 | R131 | 1219 | 87 | 1132 |
| R106 | 1227 | 45 | 1182 | R132 | 1237 | 23 | 1214 |
| R107 | 1509 | 48 | 1461 | R133 | 1103 | 29 | 1074 |
| R108 | 1097 | 16 | 1081 | R134 | 1527 | 75 | 1452 |
| R109 | 814 | 77 | 737 | R135 | 1366 | 599 | 767 |
| R110 | 1498 | 59 | 1439 | R136 | 1306 | 94 | 1212 |
| R111 | 1182 | 17 | 1165 | R137 | 1159 | 9 | 1150 |
| R112 | 1221 | 24 | 1197 | R138 | 926 | 46 | 880 |
| R113 | 1453 | 100 | 1353 | R139 | 1003 | 17 | 986 |
| R114 | 1048 | 77 | 971 | R140 | 1538 | 190 | 1348 |
| R115 | 1131 | 67 | 1064 | R141 | 1357 | 89 | 1268 |
| R116 | 1211 | 96 | 1115 | R142 | 1194 | 27 | 1167 |
| R117 | 1114 | 56 | 1058 | R143 | 1175 | 30 | 1145 |
| R118 | 1007 | 20 | 987 | R144 | 1204 | 57 | 1147 |
| R119 | 1400 | 50 | 1350 | R145 | 1358 | 32 | 1326 |
| R120 | 1465 | 174 | 1291 | R146 | 1049 | 178 | 871 |
| R121 | 1411 | 95 | 1316 | R147 | 1373 | 39 | 1334 |
| R122 | 1036 | 54 | 982 | R148 | 1333 | 327 | 1006 |
| R123 | 1152 | 24 | 1128 | R149 | 1430 | 75 | 1355 |
| R124 | 1087 | 37 | 1050 | R150 | 1390 | 117 | 1273 |
| R125 | 1564 | 132 | 1432 | avg | 1233.2 | 101.2 | 1132 |
| R126 | 1169 | 586 | 583 | σ | 195.19 | 122.36 | 206.82 |

APPENDIX B.

Table 4: A comparative analysis of the results obtained from two different procedures. **SVM-def** is the linear SVM built using the default parameters provided by *SVM^{Light}*. **aiSVM-Cb0.990** was obtained by adapting C , then b on SVM-def (using algorithms 1,2).

| Topic | SVM-def | | | aiSVM-optC-b0.990 | | |
|-------------|-------------|-------------|--------|-------------------|-------------|------|
| | sensitivity | specificity | F1 | sensitivity | specificity | F1 |
| R101 | 0.8 | 0.99 | 0.87 | 0.94 | 0.93 | 0.85 |
| R102 | 0.53 | 0.98 | 0.67 | 0.49 | 0.98 | 0.63 |
| R103 | 0.046 | 1 | 0.088 | 0.85 | 0.93 | 0.58 |
| R104 | 0.44 | 0.99 | 0.58 | 0.62 | 0.97 | 0.67 |
| R105 | 0.44 | 0.99 | 0.58 | 0.63 | 0.97 | 0.69 |
| R106 | 0 | 1 | 0 | 0.73 | 0.86 | 0.27 |
| R107 | 0 | 1 | 0 | 0.76 | 0.85 | 0.24 |
| R108 | 0 | 1 | 0 | 0.77 | 0.94 | 0.27 |
| R109 | 0.32 | 1 | 0.47 | 0.93 | 0.9 | 0.64 |
| R110 | 0.061 | 1 | 0.12 | 0.77 | 0.87 | 0.32 |
| R111 | 0 | 1 | 0 | 0.84 | 0.89 | 0.18 |
| R112 | 0 | 1 | 0 | 0.74 | 0.95 | 0.34 |
| R113 | 0.06 | 1 | 0.11 | 0.7 | 0.93 | 0.54 |
| R114 | 0.0026 | 1 | 0.0052 | 0.81 | 0.91 | 0.56 |
| R115 | 0.22 | 1 | 0.36 | 0.85 | 0.92 | 0.56 |
| R116 | 0.085 | 1 | 0.16 | 0.93 | 0.92 | 0.65 |
| R117 | 0 | 1 | 0 | 0.62 | 0.89 | 0.33 |
| R118 | 0 | 1 | 0 | 0.8 | 0.94 | 0.34 |
| R119 | 0.02 | 1 | 0.039 | 0.69 | 0.9 | 0.31 |
| R120 | 0.27 | 1 | 0.42 | 0.92 | 0.91 | 0.71 |
| R121 | 0.019 | 1 | 0.037 | 0.69 | 0.96 | 0.6 |
| R122 | 0.093 | 1 | 0.16 | 0.91 | 0.93 | 0.56 |
| R123 | 0 | 1 | 0 | 0.79 | 0.94 | 0.34 |
| R124 | 0 | 1 | 0 | 0.8 | 0.85 | 0.27 |
| R125 | 0.032 | 1 | 0.061 | 0.59 | 0.96 | 0.59 |
| R126 | 0.89 | 0.8 | 0.85 | 0.31 | 0.98 | 0.47 |
| R127 | 0.17 | 1 | 0.27 | 0.71 | 0.96 | 0.54 |
| R128 | 0.011 | 1 | 0.023 | 0.49 | 0.95 | 0.46 |
| R129 | 0 | 1 | 0 | 0.8 | 0.89 | 0.41 |
| R130 | 0 | 1 | 0 | 0.75 | 0.92 | 0.26 |
| R131 | 0.39 | 1 | 0.56 | 0.89 | 0.93 | 0.64 |

continued on the next page

| Topic | SVM-def | | | aiSVM-optC-b0.990 | | |
|-------------|-------------|-------------|-------|-------------------|-------------|------|
| | sensitivity | specificity | F1 | sensitivity | specificity | F1 |
| R132 | 0 | 1 | 0 | 0.81 | 0.89 | 0.21 |
| R133 | 0 | 1 | 0 | 0.87 | 0.91 | 0.33 |
| R134 | 0.19 | 1 | 0.32 | 0.67 | 0.94 | 0.49 |
| R135 | 0.84 | 0.9 | 0.86 | 0.6 | 0.97 | 0.74 |
| R136 | 0 | 1 | 0 | 0.72 | 0.8 | 0.33 |
| R137 | 0 | 1 | 0 | 0.78 | 0.97 | 0.26 |
| R138 | 0.013 | 1 | 0.026 | 0.72 | 0.97 | 0.61 |
| R139 | 0 | 1 | 0 | 0.86 | 0.94 | 0.34 |
| R140 | 0.0095 | 1 | 0.018 | 0.49 | 0.93 | 0.49 |
| R141 | 0.088 | 1 | 0.16 | 0.95 | 0.91 | 0.58 |
| R142 | 0 | 1 | 0 | 0.59 | 0.89 | 0.19 |
| R143 | 0 | 1 | 0 | 0.74 | 0.93 | 0.34 |
| R144 | 0.056 | 1 | 0.11 | 0.91 | 0.97 | 0.73 |
| R145 | 0.062 | 1 | 0.12 | 0.84 | 0.9 | 0.28 |
| R146 | 0.4 | 0.99 | 0.55 | 0.57 | 0.98 | 0.68 |
| R147 | 0.046 | 1 | 0.088 | 0.81 | 0.92 | 0.36 |
| R148 | 0.67 | 0.97 | 0.77 | 0.76 | 0.97 | 0.81 |
| R149 | 0 | 1 | 0 | 0.69 | 0.89 | 0.37 |
| R150 | 0.07 | 1 | 0.13 | 0.65 | 0.94 | 0.56 |
| avg | 0.15 | 0.99 | 0.19 | 0.74 | 0.92 | 0.47 |
| std | 0.24 | 0.032 | 0.27 | 0.14 | 0.039 | 0.18 |

REFERENCES

- [1] T. Joachims, “Making large-scale support vector machine learning practical,” in *Advances in Kernel Methods: Support Vector Machines*, B. Scholkopf, C. Burges, and A. Smola, Eds. MIT Press, 1998. [Online]. Available: citeseer.nj.nec.com/joachims98making.html
- [2] J. S. Urban Hjorth, *Computer Intensive Statistical Methods*. Chapman & Hall, 1994.
- [3] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer Verlag, 1995.
- [4] T. Rose, M. Stevenson, and M. Whitehead, “The Reuters Corpus Volume 1 – from yesterday’s news to tomorrow’s language resources,” in *Proceedings of the Third International Conference on Language Resources and Evaluation*, 2002.
- [5] E. M. Voorhees and D. Harman, “Trec 2002 overview,” in *Proceedings of the 2002 Text REtrieval Conference*, 2002. [Online]. Available: http://trec.nist.gov/act-part/guidelines/filter2002_guide.html
- [6] 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.
- [7] 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.
 - [8] D. D. Lewis, “Evaluating and Optimizing Autonomous Text Classification Systems,” in *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, E. A. Fox, P. Ingwersen, and R. Fidel, Eds. Seattle, Washington: ACM Press, 1995, pp. 246–254. [Online]. Available: citeseer.nj.nec.com/lewis95evaluating.html
 - [9] O. Chapelle and V. Vapnik, “Model selection for support vector machines,” *Advances in Neural Information Processing Systems*, no. 12, 2000. [Online]. Available: citeseer.nj.nec.com/chapelle00model.html
 - [10] C.-W. Hsu and C.-J. Lin, “A simple decomposition method for support vector machines,” *Machine Learning*, vol. 46, pp. 291–314, 1999. [Online]. Available: citeseer.nj.nec.com/hsu99simple.html
 - [11] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*. Springer Verlag, 2001.
 - [12] G. Casella and R. L. Berger, *Statistical Inference*. Brooks Cole, 2001.
 - [13] P. Domingos, “Metacost: A general method for making classifiers cost-sensitive,” in *Knowledge Discovery and Data Mining*, 1999, pp. 155–164. [Online]. Available: citeseer.nj.nec.com/domingos99metacost.html
 - [14] Y. Yang, “A study on thresholding strategies for text categorization,” in *SIGIR 2001: Proceedings of The 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, W. B. Croft, D. J. Harper, D. H. Kraft, and J. Zobel, Eds. New York, NY: Association for Computing Machinery, 2001, pp. 137–145.
 - [15] G. Salton and C. Buckley, “Term-weighting approaches in automatic text retrieval,” *Information Processing and Management*, vol. 24, no. 5, pp. 513–523, 1988.
 - [16] M. Aizerman, E. Braverman, and L. Rozonoer, “Theoretical foundations of the potential function method in pattern recognition learning,” *Automation and Remote Control*, vol. 25, pp. 821–837, 1964.
 - [17] —, “The Robbins-Monro process and the method of potential functions,” *Automation and Remote Control*, vol. 26, pp. 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