Obtaining Calibrated Probability Estimates from Support Vector Machines

Joseph Drish
Department of Computer Science and Engineering 0114
University of California, San Diego
La Jolla, California 92037-0114
jdrish@cs.ucsd.edu

Abstract

We use a technique known as binning to convert the outputs of support vector machine (SVM) classifiers into well-calibrated probabilities. Using the KDD'98 data set as a testbed, we evaluate predicted probabilities using four metrics, and compare our results to those obtained by Zadrozny and Elkan for naive Bayes classifiers. We stress test the latest release of the LIBSVM software, and use its new functionality for training on unbalanced data sets to find the best parameters for the SVM classifiers. We demonstrate that using the F1 value as a metric for tuning SVMs is successful for making them work on the KDD'98 data set, which is highly unbalanced. On the other hand, we show that binning works better for naive Bayes classifiers, and that it is difficult to avoid overfitting directly using SVMs.

1 Introduction

In many supervised learning tasks a learned classifier automatically induces a ranking of test examples, making it possible to determine which test examples are more likely to belong to a certain class when compared to other test examples. However, for many applications this ranking is not sufficient, particularly when the classification decision is cost-sensitive. In this case, it is necessary to convert the outputs of the classifier into well-calibrated posterior probabilities. A recent paper that addresses this problem is [7], which introduces new methods for estimating the probabilities from naive Bayes and decision tree classifiers. This paper presents a replication of that work using Support Vector Machines (SVMs).

Based on the theory of Structural Risk Minimization [5], SVMs learn a decision boundary between two classes by mapping the training examples onto a higher dimensional space and then determining the optimal separating hyperplane between that space. Given a test example x, the SVM outputs a score that provides the distance of x from the separating hyperplane. The sign of the score indicates to which class j example x belongs, where $j \in \{1, -1\}$. The problem of interest is how to calibrate that score into an accurate class conditional posterior probability, or $P(j|x)$.

Our solution is to use a histogram technique known as binning, which is recommended in

[7] for naive Bayes classifiers. We selected this method because of the similarities between naive Bayes and SVM classifiers, and also because of its simplicity. The binning method proceeds by first ranking the training examples according to their scores, then dividing them into b subsets of equal size, called bins. The value of b is typically chosen experimentally such that the variance is reduced in the binned probability estimates. Given a test example x, it is placed in the bin according to the score produced by the SVM. The corresponding estimated probability $P(j|x)$ is the fraction of training examples that actually belong to the class that has been predicted for the test example.

The data set that we use to train and test the support vector classifiers is from the KDD'98 data mining competition. This data set contains information about persons in the past who either did or did not make donations to a certain charity. It consists of 95,412 training examples, each corresponding to an individual, and 481 features. The test set consists of 96,367 examples and 479 features. The training set has two additional fields: one indicating whether or not the individual has donated, and another for the amount of the donation. The goal is to choose individuals to solicit a donation so that overall profit is maximized, assuming that the cost to mail a solicitation is \$0.68.

In addition to being very large, the KDD'98 data set is highly unbalanced. There are only 4,843 positive examples in the training set and 4,873 positive examples in the test set. We elected to use the LIBSVM [1] software package for learning the SVM classifiers since its latest release contains an option to alter the SVM training specifically for unbalanced data sets. The KDD'98 data set provides the perfect challenge to test this new functionality. Also, LIBSVM uses John Platt's Sequential Minimal Optimization (SMO) [3] for training, which is the most efficient algorithm for SVM training currently known.

There are two contributions of this paper. The first is an evaluation of the quality of the binned SVM probability estimates using the 4 metrics suggested in [7]. These are squared error, log-loss or cross entropy, lift charts, and the profit obtained when we use the estimates to choose individuals to solicit a donation. Our results are compared to those obtained in [7] for naive Bayes classifiers. The second contribution is an analysis of the applicability of SVMs on the KDD'98 data set. Specifically, we investigate how to tune the parameters of the SVM to maximize profit, examine the extent to which SVMs overfit the training set, and discuss how efficient SVMs are for this task.

2 Feature Selection and Preprocessing

We used the following 8 features to train the support vector classifiers:

INCOME: household income code (range 1–8)
LASTDATE: date of the most recent gift
FIRSTDATE: date of the first gift
RFA_2F: frequency code (range 1–4)
RFA_2A: amount of the last gift code (range A–G)
PEPSTRFL: RFA (recency, frequency, amount) (X or blank)
PGIFT: number of gifts/number of promotions received
TARGET_B: binary indicator for response to 1997 mailing

In addition to these being the most predictive features, using only these provides an experimentally valid way to compare our results to [7], since they used the same features to obtain their results.

The feature PGIFT is the only one not directly included in the KDD'98 data set. It is obtained by dividing the feature NGIFTALL by the feature NUMPROM in the orginal data set. The LIBSVM software is only able to train SVM classifiers using numerical values,

so the RFA_2A values were mapped to integers in the range 1 through 7 corresponding to letters A through G, and the PEPSTRFL values were mapped to 1 and 0 corresponding to values X and blank.

We then scaled the data so that no one feature dominated the training process. To do this, from each feature value we subtracted the feature mean, then divided by the feature standard deviation. This is known as z-scoring the data, and is a method of scaling that prevents outliers within a feature vector from having too much influence on the entire feature vector.

3 Tuning the SVM Classifiers

This section describes the quadratic programming (QP) problem solved by the LIBSVM software to accomodate unbalanced data sets. We give an intuitive explanation for how the alteration of the problem changes SVM training. The reader can refer to [5] for a description of the original SVM QP problem formulation. We then describe our method for determining the best penalty parameter, C_+.

3.1 Training on Unbalanced Data Sets

The LIBSVM software provides an alternative method for learning classifiers on unbalanced data sets. The main idea is to use two parameters, C_+ and C_-, to tradeoff generalization ability and misclassification error for positive and negative training examples, respectively. Formally, given a set of training vectors $x_i \in R^n$, $i = 1,...l$, and a vector $y \in R^l$ such that $y_i \in \{1,-1\}$, the primal SVM problem is

$$\begin{aligned} \min_{w,b,\xi} \quad & \frac{1}{2}w^T w + C_+ \sum_{y_i=1} \xi_i + C_- \sum_{y_i=-1} \xi_i \\ & y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0, i = 1,...,l. \end{aligned} \tag{1}$$

Its dual problem is

$$\begin{aligned} \min_{\alpha} \quad & \frac{1}{2}\alpha^T Q\alpha - e^T\alpha \\ & 0 \leq \alpha_i \leq C_+, \text{if } y_i = 1 \\ & 0 \leq \alpha_i \leq C_-, \text{if } y_i = -1 \\ & y^T\alpha = 0. \end{aligned} \tag{2}$$

The variable e is the vector of all ones, Q is an l by l positive semidefinite matrix, and α_i are the support vectors. The function ϕ maps the training vectors into a higher dimensional space where the optimal separating hyperplane is formed. The resulting decision function is

$$f(x) = sign(\sum_{i=1}^{l} y_i \alpha_i K(x_i, x) + b),$$

where $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ is the kernel. To train our SVM classifiers we use a radial basis kernel of the form $K(x_i, x_j) = \exp(-||x_i - x_j||^2/2\sigma^2)$. The kernel function measures

the similarity between the vectors x_i and x_j. A well chosen value for σ accomplishes the goal of separating data clusters of one class from data clusters containing non-class members. For this project we did not spend much time optimizing the kernel. We simply set $\sigma = 50$ because informal experiments revealed that this value achieved the best classification accuracy.

The important parameters from equation (1) are the penalty parameters C_+ and C_-. The ratio of these two values determines how the training examples from each class are penalized. As the ratio of C_+ to C_- increases, so will the rate at which the SVM classifier predicts $j = 1$ for a given example x. As the ratio of C_+ to C_- decreases, the rate at which the SVM classifier predicts $j = -1$ increases.

3.2 Determining the Best Value for C_+

For our experiments we fix C_- at 1, and determine the best ratio by varying C_+. For most machine learning tasks the goal is to maximize accuracy on the test set. Since our ultimate goal is to maximize profit on the test set, we would like the SVM to predict donate to as many donors as possible, while not soliciting too many non-donors. Therefore, we would like to balance precision and recall, which are defined below as

$$precision = \frac{tp}{(tp + fp)}, \quad recall = \frac{tp}{(tp + fn)}.$$

The values tp, fp, and fn stand for true positives, false positives, and false negatives, respectively. A metric that measures how well precision and recall agree is called the F1 value. This is the metric we use to identify the best C_+ parameter, and is defined precisely as

$$\text{F1} = 2/((1/precision) + (1/recall)).$$

To determine the best C_+ parameter, we learned several SVM classifiers using different values for C_+, each on 10,000 training records consisting of 509 positive examples. We then tested these SVMs on 10,000 different training records consisting of 490 positive examples. Table 1 shows the confusion matrix and the resulting F1 value for the different SVM classifiers. Notice that as the value of C_+ increases, the number of solicitations increases. As a result, the number of actual donors that we predict as donors increases, at the expense of soliciting more non-donors. To solicit a test example is equivalent to classifying a test example as positive.

As we increase C_+, the value of F1 increases until it reaches its peak at 8, and then monotonically decreases. Even though we conjecture that the value $C_+ = 8$ may work best on the KDD'98 data set for profit maximization, there is no clear intuitive link between optimizing in terms of F1 and obtaining accurate probability estimates from the SVM classifiers. The focus of this optimization effort was an attempt to maximize profit on the test set only.

4 Converting SVM Scores into Probabilities

To transform the scores of the SVM classifiers into accurate well-calibrated probabilities, we use a technique known as binning, which is recommended in [7] for naive Bayes classifiers. As mentioned in the introduction, the binning method proceeds by first sorting the training examples according to their scores, and then dividing them into b equal sized sets, or bins, each having an upper and lower bound. Given a test example x, it is placed in a bin according to its score. The corresponding probability $P(j = 1|x)$ is the fraction of positive training examples that fall within the bin. We set $b = 10$ as in [7].

Table 1: Effect of varying C_+ on SVM classification.

C_+	classification	solicitations	tp	fp	fn	F1
–	0.9479	35	2	33	488	0.0076
5	0.8891	709	45	664	445	0.0750
7.5	0.8704	924	59	865	431	0.0835
8	0.8670	968	64	904	426	0.0878
9	0.8551	1095	68	1027	422	0.0858
10	0.8440	1214	72	1142	418	0.0845
12.5	0.8231	1437	79	1358	411	0.0820
15	0.8086	1594	85	1509	405	0.0816
20	0.7758	1946	97	1849	393	0.0796

Using all the training examples from the training set sometimes results in overfitting the probability estimates. To solve this problem we use a method described in [6], where 70% of the training examples are used to learn the classifier and 30% are used for the binning process. These subsets are stratified, meaning the proportion of positive examples in both of them are fixed. There is no imposed lower or upper bound on SVM scores. Therefore, when using this method it is possible for some scores from the 30% subset to fall below or above the low and high scores, respectively, of the 70% training subset. If this happens the corresponding probability $P(j = 1|x)$ for example x is that of the nearest bin to the score of x.

Table 2 shows the resulting binned SVM class-conditional probability estimates. The ALL classifiers refer to those that use all of the training set for both learning the classifier and for the binning process. The SPLIT classifiers refer to those that use 70% of the training set to learn the classifier, and the remaining 30% for the binning process. The number in parentheses corresponds to the value of C_+. We use this convention throughout the paper to identify the SVM classifiers when explaining our results. The LIBSVM software assigns negative scores to positively classified examples, so we expect the probability estimates to be monotonically decreasing in the table for increasing bin number.

For the ALL classifiers, the probabilities are very large in the first couple bins, and afterwards severely decrease. All the SPLIT classifiers have probability estimates that are smoothed towards the base rate for the test set, which is close to 5%. Note also for the SPLIT classifiers, the minimum probability estimate occurs in bin 4 for $C_+ = 1$, bin 5 for $C_+ = 8$, bin 6 for $C_+ = 15$, and bin 7 for $C_+ = 20$. For the ALL classifiers the probability estimates tend to quickly decrease monotonically, whereas the estimates for the SPLIT classifiers slowly decrease nonmonotonically. This is an indication of severe overfitting. When using different training examples for the binning process, positive examples are not consistently being classified positively when they should be.

5 Evaluating the Probability Estimates

There are four methods we use to evaluate the binned SVM probability estimates. These are squared error, log-loss or cross-entropy, lift charts, and the profit achieved when we solicit donations according to the decision $P(j = 1|x)y(x) > \$0.68$, where $y(x)$ is the expected donation amount for person x. Since this paper is only concerned with calibrating accurate probabilities, the expected donation amounts are fixed and based on a regression technique discussed in [6].

We report the mean squared error (MSE), mean log-loss (MLL), and profit in Table 3 for the various support vector classifiers. Also included in the table are results for naive Bayes, binned naive Bayes, using the base rate for all examples, and using $P(j = 1|x) = 1$ for all

Table 2: Binned SVM probability estimates.

bin	ALL(1)	ALL(8)	SPLIT(1)	SPLIT(8)	SPLIT(15)	SPLIT(20)
1	0.1523	0.2080	0.0625	0.0604	0.0583	0.0534
2	0.0456	0.1157	0.0486	0.0688	0.0625	0.0548
3	0.0428	0.0770	0.0496	0.0520	0.0562	0.0615
4	0.0487	0.0238	0.0451	0.0454	0.0538	0.0569
5	0.0476	0.0169	0.0475	0.0363	0.0507	0.0489
6	0.0593	0.0308	0.0482	0.0437	0.0384	0.0493
7	0.0662	0.0258	0.0503	0.0503	0.0454	0.0402
8	0.0362	0.0053	0.0489	0.0552	0.0517	0.0531
9	0.0021	0.0022	0.0528	0.0461	0.0465	0.0479
10	0.0068	0.0020	0.0542	0.0493	0.0440	0.0416

examples (corresponding to mailing to everyone). These were copied directly from [7].

Squared error is defined as $\sum_j (t(j|x) - p(j|x))^2$, where $p(j|x)$ is the probability estimated by the method for example x and class j, and $t(j|x)$ is the true probability of class j for x. For data sets where true labels are known and probabilities are not known, $t(j|x)$ is defined to be 1 if the label of x is j and 0 otherwise. We calculated the mean squared error for each classifier on both the training and test sets.

Both the ALL classifiers severely overfit the training data. The MSE for these are smaller than the MSE for any other classifier on the training set, but larger than for any other classifier on the test set. The SPLIT classifiers overfit the training data, but not as badly. A C_+ value of 8 achieves the lowest MSE for both the ALL and SPLIT classifiers. Overall, binned naive Bayes does the best, followed by SPLIT(8).

The second metric is log-loss or cross-entropy, defined as $-\sum_j t(j|x) \log_2 \frac{p(j|x)}{t(j|x)}$. The MLL results for the SVM classifiers exhibit the same pattern as the results for MSE. Both the ALL and SPLIT classifiers overfit the training data; the SPLIT classifiers do not overfit as much. Like MSE, setting $C_+ = 8$ for the SVM classifiers achieves the best performance, but none of these are better than the binned naive Bayes classifier. A disadvantage of this metric is that if any method estimates $P(j = 1|x) = 0$ for any example whose actual class is $j = 1$, the mean log-loss of the method is infinite. This explains the infinite entry for the mail-to-everyone case.

Profit is the third metric we use to evaluate the probability estimates. Like in [7], the differences in profit are much more accentuated than the differences in MSE and MLL. As Table 3 demonstrates, the SVM classifiers are overfitting the training data. The best result among the SVM classifiers is again the SPLIT classifier with $C_+ = 8$, which achieves a profit of \$12,969. For the SPLIT classifiers, the amount of overfitting is roughly the same for all values of C_+. When we use the SPLIT classifiers, although our probability estimates are shifted toward the base rate, we still achieve a profit better than using a probability equal to the base rate. Neither binned naive Bayes nor raw naive Bayes overfit the training data. Overall, the best profit of \$14,642 is achieved by binned naive Bayes.

The final metric we use to evaluate the probabitiy estimates are lift charts. The method for obtaining a lift chart is explained in [7]. We repeat that explanation here for clarity. We first sort the examples in descending order according to their scores. Given $0 \leq a \leq 1$, let $T(a)$ be the fraction a of examples with the highest scores. The lift at a is defined as $l(a) = r(a)/P(j = 1)$ where $r(a)$ is the proportion of positive examples in $T(a)$. SVM lift charts for the training and test sets are found in figures 1 and 2, respectively. The lift charts for the naive Bayes and binned naive Bayes classifiers can be seen in [7].

Table 3: Probability evaluation results.

classifier	Training set MSE	Training set MLL	Training set Profit	Test set MSE	Test set MLL	Test set Profit
ALL(1)	0.09325	0.26715	$25699	0.10314	0.33692	$7919
ALL(8)	0.08850	0.23948	$38040	0.10215	0.34670	$8026
SPLIT(1)	0.09596	0.28685	$13699	0.09600	0.28862	$12512
SPLIT(8)	0.09534	0.28220	$17550	0.09586	0.28763	$12969
SPLIT(15)	0.09551	0.28322	$16748	0.09592	0.28807	$12817
SPLIT(20)	0.09593	0.28604	$14700	0.09594	0.28823	$12483
NAIVE BAYES	0.10089	0.30198	$10083	0.10111	0.30400	$9531
BINNED NAIVE BAYES	0.09546	0.28336	$14897	0.09528	0.28365	$14642
ALL BASE RATE	0.09636	0.28961	$11923	0.09602	0.28880	$12252
ALL ONE	1.89848	∞	$10790	1.89886	∞	$10560

For the SVM training set lift chart, the ALL(8) classifier performs the best overall. Between $a = 10$ and $a = 70$, the worst classifiers are those with no bias ($C_+ = 1$). The biased SPLIT classifiers all achieve about the same performance. All the scores are unusually high when a is between 0 and 10, which is an indication of overfitting. The biased SVMs achieve better performance on the training set lift charts than for both variations of naive Bayes. For $10 \leq a \leq 100$, the lift for naive Bayes and binned naive Bayes is less than 2. For all the biased SVM classifiers, the lift remains greater than 2 when a is between 0 and 40, and coverges to 1 at a slightly lower rate.

For the test set, the worst SVM scores are also generated when there is no bias. Initially, when a is between 0 and 40, both SVM classifiers using $C_+ = 8$ do very well, then the SPLIT classifier with $C_+ = 15$ tends to dominate, but only slightly. The SPLIT classifiers with $C_+ = 15$ and $C_+ = 20$ do poor initially, which is due to both of them having a relatively low probability estimate in their first bin. As with our other metrics, setting $C_+ = 8$ achieves the best performance overall for SVMs. However, the test set performance of binned naive Bayes and raw naive Bayes are better than the SVM lift charts, as they converge to 1 at a more consistent rate. This again is evidence of SVM overfitting.

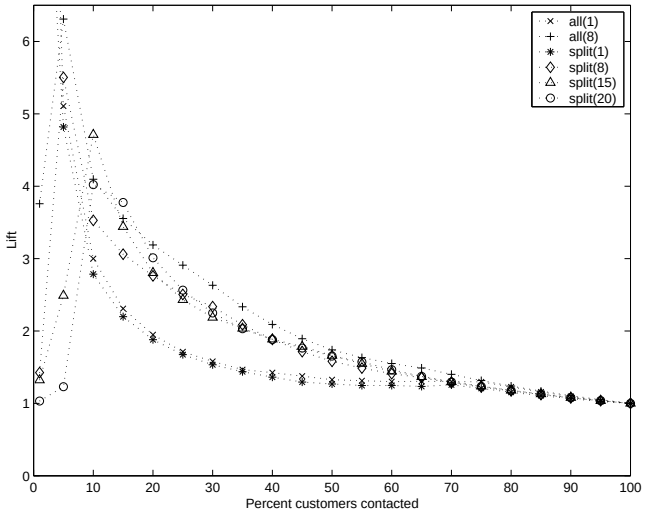

Figure 1: SVM lift charts for the training set.

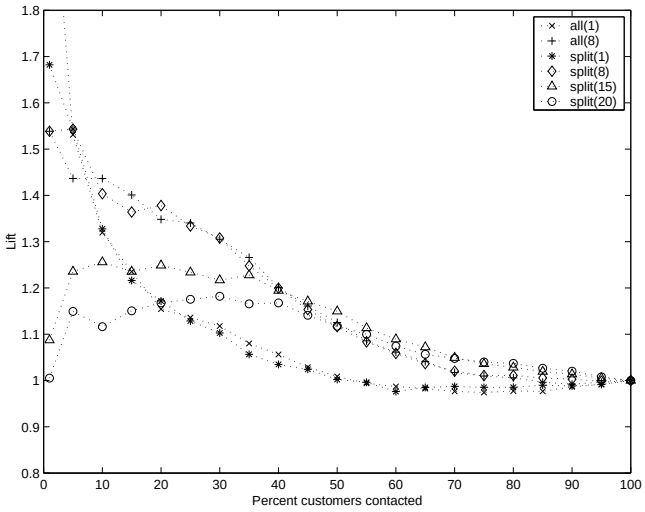

Figure 2: SVM lift charts for the test set.

6 Discussion

The experimental results reveal that overfitting is the major problem of our support vector classifiers. We were able to reduce overfitting somewhat by using the SPLIT classifiers, but not enough to achieve an overall performance similar to that of the binned naive Bayes classifier. We may be able to mitigate the problem of overfitting by selecting a different kernel. Using a sufficiently high degree polynomial kernel has been known to be able to generalize well without overfitting.

One promising aspect of this analysis is how much the performance of the best C_+ value on a subset of the training set parallels the success of SVMs using the same C_+ value on the test set. All of the probability evaluation metrics show that the best performance is achieved when setting $C_+ = 8$, regardless of how we trained the SVM. This indicates that optimizing C_+ in terms of the F1 value is a good idea.

Currently it is unclear whether the ratio of C_+ to C_- or the values of these parameters have the greatest impact on SVM classification. It would be interesting to experiment with different values of these parameters on the KDD'98 data set. For example, we could set $C_+ = 400$ and $C_- = 50$ for the same ratio of 8. What we have noticed is the effect C_+ has on the range of scores when classifying examples. Table 4 provides the lowest and highest scores for the training and test sets for our SVM classifiers when C_- is fixed at 1. As C_+ increases, so does the range of scores.

Another aspect of SVMs we are interested in is how well they scale to a data set the size of the KDD'98 data set. We trained five SVM classifiers with an RBF kernel, and set $C_+ = 8$ and $C_- = 1$ using an increasing number of training examples. The results of these experiments are shown in Table 5. We report the number of support vectors, the time in hours it takes for the SVM to learn the classifier, and the raw classification accuracy on the entire training and test sets. The number of support vectors generated is systematically about half the number of total examples used to learn the classifier.

The timing results of Table 5 show that SVMs can take up to 50 hours to train. The question of whether the LIBSVM software scales well according to these results is subjective, since these numbers may or may not be acceptable depending on the engineering or business cir-

Table 4: Range of scores for different values of C_+.

	Training set		Test set	
SVM(C_+)	low	high	low	high
ALL(1)	-1.0003	1.3156	-1.0049	1.3464
ALL(8)	-2.1314	3.2473	-2.3212	2.8845
SPLIT(1)	-1.0014	1.3162	-1.0042	1.3470
SPLIT(8)	-2.0207	2.6436	-1.8441	2.6572
SPLIT(15)	-2.3052	3.2440	-2.5326	3.5163
SPLIT(20)	-2.3177	3.3035	-2.6287	3.2476

Table 5: Effect of training size on training time and raw SVM accuracy.

examples	#SVs	time (hours)	train accuracy	test accuracy
10000	6129	0.61	0.8693	0.8639
20000	10824	2.85	0.8604	0.8534
40000	20025	7.77	0.8685	0.8543
66788	32386	24.57	0.8834	0.8634
95412	45709	48.79	0.8950	0.8714

cumstances when SVMs are being used. It is important to note that increasing the number of features would not have a largely influential effect on training time, since SVM training time is quadratic in the number of examples, independent of the number of features. Comparing the training efficiency between SVMs and naive Bayes classifiers is beyond the scope of this paper.

Table 5 also shows how the classification accuracy of the raw SVM increases as we include more training examples. The classification results show that we do not lose that much information with smaller training data, at least for this data set. An interesting result is that using only 10,000 training examples has better raw classification accuracy than using 20,000 and 40,000 examples on both the training set and test set, and also better on the test set when using 70% of the training set. The best performance on both the training and test sets is achieved when we train using all of the training set, as expected.

7 Alternative Methods

Binning is a general technique that can be applied to any classifier that outputs a score given an example x. This method is assumed to work if the classifier ranks examples well, which is why binning is effective for naive Bayes classifiers. There are other techniques for probability calibration specifically designed for support vector machine classifiers. One such method is described in [4], where the training set is used to fit a sigmoid function. The calibrated probability estimate then becomes

$$P(j=1|f) = \frac{1}{1+\exp(Af+B)}.$$

Another method for converting the scores of SVMs into probabilities is based on a Bayesian framework and is discussed in [2]. It would be interesting to see if these methods of calibration would improve SVM performance on the KDD'98 data set.

8 Conclusion

We used a method known as binning to convert the outputs of support vector machine classifiers into accurate posterior probabilities. We evaluated the probabilities using four metrics, and compared our results to those obtained in [7] for naive Bayes classifiers. We used the challenging KDD'98 data set as a testbed for this task, allowing us to stress test the latest release of the LIBSVM software. Using new functionality provided by the software, we were able to tune the SVMs specifically for unbalanced data sets. We used the F1 value as a metric for tuning the SVMs, which was a good idea since the F1 value that was determined to be the best on a subset of the training set achieved the best performance on the test set.

Our results show that probability calibration using binning works better for naive Bayes classifiers than for support vector machine classifiers, mostly because the naive Bayes classifiers rank examples better than SVMs. The most significant problem that SVMs encounter is overfitting, something that is avoided by the naive Bayes classifiers. Although it has proven to be very difficult to get SVMs to work on the KDD'98 data set, it would be premature to say that they do not work or scale well for large, unbalanced data sets in general. It is fair to conclude that practical usage of SVMs requires a significant amount of user knowledge and experience.

References

[1] Chang, C.C. and Lin, C. (2001) LIBSVM: a Library for Support Vector Machines (Version 2.3)

[2] Kwok, J.T. (1995) Moderating the Outputs of Support Vector Machine Classifiers. In *IEEE - NN*.

[3] Platt, J. (1999) Fast Training of Support Vector Machines using Sequential Minimal Optimization. In *Advances in Kernel Methods - Support Vector Learning*.

[4] Platt, J. (1999) Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*.

[5] Vapnick, V. (1998) Statistical Learning Theory. John Wiley & Sons.

[6] Zadrozny, B. and Elkan, C. (2001) Learning and making decisions when costs and probabilities are both unknown. Technical Report CS2001-0664). University of California, San Diego.

[7] Zadrozny, B. and Elkan, C. (2001) Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. To appear in *Proceedings of the Eighteenth International Conference on Machine Learning*.