Building Classifiers With Unrepresentative Training Instances: Experiences From The KDD Cup 2001 Competition

Anuradha Bhamidipaty ANU@IT.IITB.AC.IN
Anand Janakiraman JANAKI@IT.IITB.AC.IN
Sunita Sarawagi SUNITA@IT.IITB.AC.IN
IIT Bombay, India.

Jayant Haritsa HARITSA@DSL.SERC.IISC.ERNET.IN
IISc Bangalore, India.

Abstract

In this paper we discuss our experiences in participating in the KDD Cup 2001 competition. The task involved classifying organic molecules as either active or inactive in their binding to a receptor. The classification task presented three challenges: highly skewed class distribution, large number of features exceeding training set size by two orders of magnitude, and non-representative training instances. Of these, we found the third challenge the most interesting and novel.

We present our process of experimenting with a number of classification methods before finally converging on an ensemble of decision trees constructed using a novel attribute partitioning method. Decision trees provided partial shield from the differences in data distribution and the ensemble provided stability by exploiting the redundancy in the large set of features. Finally, we employed semi-supervised learning to incorporate characteristics of the test set into the classification model.

We were second-runner's up in the competition. We followed up the competition with further research in semi-supervised learning and obtained an accuracy higher than that of the winning entry.

1. Introduction

Recently there has been a rapid growth of interest in mining biological databases. In keeping with this trend, the Annual ACM SIGKDD KDD Cup competition of the year 2001[1] was focused on the application of data mining techniques to mining biological data. The first of the three problems in the competition was "Prediction of Molecular Bioactivity for Drug Design — Binding to Thrombin". Drugs are small organic molecules that achieve their desired activity by binding to a target site on a receptor. A crucial step in drug design is to separate the active (binding) compounds from the inactive (non-binding) ones. The task in the competition was to identify the properties of compounds that enable them to bind to Thrombin, a key receptor in blood clotting.

The training dataset[2] consisted of 1909 compounds identified as either active or inactive in their ability to bind to Thrombin. The test dataset consisted of 634 compounds whose binding to Thrombin was to be predicted. The datasets presented three main challenges.

1. The training data was highly skewed. Out of 1909 compounds, 1867 were inactives and only 42 (2.2% of the total) were actives.
2. The number of features was two orders of magnitude more than the number of instances. Each compound was represented by a feature vector consisting of 139,351 binary features, which describe the three-dimensional properties of the compound. The organizers did not provide the definition of individual features. So we could not exploit domain knowledge even if we had any.
3. The third, and the most important challenge, was that the training set was not representative of the data distribution in the test set.

A lot of research has gone into addressing the first two challenges in the context of other applications.

[1] `http://www.cs.wisc.edu/~dpage/kddcup2001`

[2] Contributed by DuPont Pharmaceuticals Research Laboratories to KDD Cup 2001.

Skewed class distributions are handled in a number of ways (Nathalie Japkowicz, 2000; Japkowicz, 2000; Fawcett, 1996), including, building cost-sensitive classifiers (Domingos, 1999; Zadrozny & Elkan, 2001; Chan & Stolfo, 1998; Provost & Fawcett, 1998; Weiss & Provost, 2001) that assign higher cost to misclassifications of the minority class, stratified sampling on the training instances to balance the class distribution (Kubat & Matwin, 1997) and rule-based methods that attempt to learn high confidence rules for the minority class (Ali et al., 1997).

Large feature sets are typically handled through a feature selection step in the preprocessing stage (Dash & Liu, 1997). A number of techniques exist ranging from simplistic individual feature selection methods using measures like Entropy, Mutual Information and Fisher's index to more exhaustive approaches like Wrappers (Kohavi & John, 1997) and Markov Blankets (Koller & Sahami, 1996). In recent times these techniques have been applied to classifying very high dimensional datasets like text with much success.

We found the problem of building a classifier with a training dataset that is not representative of the test instances the most compelling and novel. Not being able to rely on the crutches of cross-validation for various stages of model selection and classifier evaluation led us to concentrate on metrics like robustness and stability more than accuracy on the validation set. The competition like several others of the kind released the data in two stages. The initial method that we designed based on high accuracy on a set-aside portion of the training set, had to be discarded when we found that the test attributes had a drastically different distribution of "1"s. Our final method was an ensemble of decision trees built in a novel way to exploit multiple redundant sets of features and that provided stability to our method. Another interesting outcome was the way we dealt with unlabeled test instances. To capture the difference in data distribution we augmented the training data of a supervised learner with unlabeled test instances.

Outline We describe our methodology in detail in the order in which we worked on the problem. Section 2 describes the training phase. Section 3 describes the steps we followed after the test set was released. Section 3.5 presents a postmortem of our method done after the results were declared. Section 4 describes our further post-competition research on semi-supervised learning that yielded an accuracy higher than the best achieved at the competition. Finally, we conclude in Section 5 with a list of the lessons learnt.

2. Training Phase

In the training phase we were engrossed with the challenges offered by the limited active instances and the enormous set of features. As the data was highly skewed, the organizers had announced the evaluation criteria as weighted accuracy defined as the average of the fraction of actives and the fraction of inactives correctly classified. Thus, predicting all instances as belonging to the majority class would yield an accuracy of 98% on the training instances, but a weighted accuracy of only 50%.

2.1. Familiarization with the Data

The training data was not devoid of noise — for example, there were four identical records with all 139,351 features having a value of "0" but two of them were active and the other two inactive. Overall there were lot more zeros than ones (1000 to 1). The number of features with a value of "1" was eleven times higher for the actives than for the inactives. A naive separation between the two classes based on the count of "1"s in the records. In terms of the number of "1"s the top 8 records were all actives and the bottom 1000 records were all inactives (except for the two actives mentioned earlier). Thus, purely based on the number of "1"s it was possible to gain a weighted accuracy of 60% *on the training set*.

2.2. Feature Selection

The large number of features necessitated some form of feature selection. We employed entropy based feature selection. However, we observed that selecting just the top n features with the highest entropy left us with several actives with all zeros as the feature value even for fairly large values of n (greater than 1000). This made them indistinguishable from other inactives several of which also had all zeros. Therefore, we used an alternative feature selection method. For each active a, we selected the k highest entropy features only amongst those features where the value v of that feature in a appeared in greater fraction of actives than the opposite value $1 - v$. The union of the top-k features from each active formed the final feature set. This ensured that in addition to the top n entropy features, we also picked some others that although had slightly lower entropy, would nevertheless be helpful in distinguishing actives. At the end of this process we had roughly 800 features.

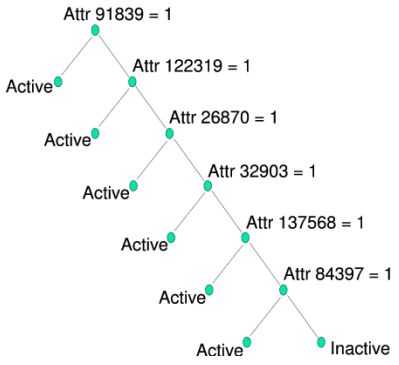


Figure 1. A single decision tree.

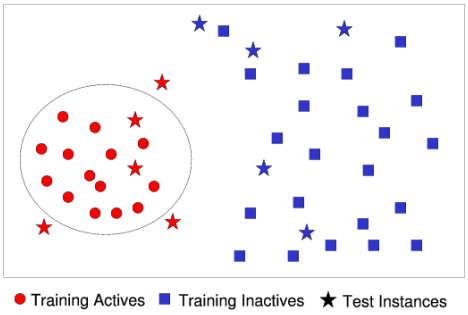


Figure 2. Actives clustered and separated from the inactives.

2.3. Classification Methods

We next sought to train different classifiers on the training data. Three-fourths of the available data was used for training and the rest one-fourth for validation. Since the training set was limited, ideally we should have employed N-fold cross-validation. However, for our first-cut experiments we used a single validation set.

We experimented with four classification methods: C4.5 Decision Tree classifier (Quinlan, 1993), $\mathcal{MLC}$++'s naive Bayes classifier (Kohavi et al., 1996) , SVMTorch Support Vector Machine classifier (SVM) (Collobert & Bengio, 2001) and a clustering method tailor made for the current problem at hand.

Decision Tree Classifier The first method we tried was decision tree classifiers. Surprisingly, a fairly simple tree (shown in Figure 1) consisting of just 6 attributes was able to yield close to perfect accuracy on the training set. However, on the validation data it mis-classified 7 out of the 10 actives, with a confusion matrix of $\begin{bmatrix} 3 & 7 \\ 1 & 459 \end{bmatrix}$. Clearly, we needed to explore other methods.

Naive Bayes Classifier Naive Bayes classifiers have been found useful for text classification ((Chakrabarti et al., 1998)). This dataset with the sparsity of "1"s resembled a text dataset. Also, naive Bayes has been found to be useful in those high dimensional settings where no single feature holds a significant information. Instead, the information is scattered over several features. Naive Bayes by combining them into a single value gives due importance to all of them. This is in contrast to decision tree classifiers, that typically select only a few (less than a dozen) features. Unfortunately, our first cut experiments with naive Bayes yielded negative results. The confusion matrix for a single naive Bayes when applied on the validation set was found to be $\begin{bmatrix} 0 & 10 \\ 1 & 459 \end{bmatrix}$. All the active instances were mis-classified by naive Bayes. Our guess for this bad performance was the extreme skewness in class distribution. Perhaps, the prior probability of the majority class overshadows the differences in the attribute conditional probability terms.

Support Vector Machine Classifier Support vector machines (SVMs) (Vapnik, 1995; Joachims, 1998) have been shown to excel at two-class discriminative learning problems. In high dimensional settings, like text, they often outperform generative classifiers, especially those that use inaccurate generative models, such as Naive Bayes. The next natural step for us was to try out SVMs. The accuracy we achieved using SVMs was no better than with decision trees. We had experimented with linear and Gaussian kernels and a limited range of parameter values. More exhaustive tuning of the SVM parameters might have yielded better results.

We were not satisfied with the results of any of the above standard classification methods. Therefore, we proceeded to explore better methods of feature selection using a variant of a nearest neighbor classifier as the basis.

2.4. Clustering

The idea was to choose a feature set where the active instances cluster together and are separated from the inactives as shown in Figure 2. If such a cluster exists, then the proximity of a test instance to the cluster could be used to predict it as either active or inactive. Our measure of similarity was the number of overlapping "1"s between two instances. This is a slight deviation from standard similarity measures for binary attributes, since we felt that the presence of a "1" was more important in determining the activity level of the compound.

With the previously selected feature set, we found that there was a poor separation of the actives from the inactives. We therefore proceeded to deploy a more exhaustive method for feature selection that would achieve this separation. In this method, we pick features incrementally, one-by-one such that we achieve the maximum clustering of the actives and the maximum separation from the inactives at each stage. Let $\mathcal{F}$ be the given feature space and $\mathcal{G}$ be the subset of features obtained after feature selection. $\mathcal{G}$ is initialized to ϕ, the empty set. The feature selection method we used is described in Figure 3. $x_\pi(\mathcal{G})$ denotes the projection of an instance x on the set of features $\mathcal{G}$. The distance $d(c_a, c_i)$ between the active centroid c_a and the inactive centroid c_i is the Euclidean distance.

Incremental feature selection method:

Let, A = set of actives
Let, I = set of inactives
$\mathcal{F}$ =feature space
$\mathcal{G} = \phi$

$separation = 0$
do
  /* Choose a feature for inclusion */
  for each feature, $f_i \in \mathcal{F}$
    $c_a = \sum_{x \in A} \frac{x_\pi(f_i \cup \mathcal{G})}{|A|}$
    $c_i = \sum_{x \in I} \frac{x_\pi(f_i \cup \mathcal{G})}{|I|}$
    $new\_separation = d(c_a, c_i)$
    if $new\_separation > separation$
      $f = f_i$
      $separation = new\_separation$
    end if
  end for
  $\mathcal{G} = \mathcal{G} \cup f$
  $\mathcal{F} = \mathcal{F} - f$
while no significant change in separation.

Figure 3. Incremental feature selection

With this approach we selected a set of 493 features. The feature set clustered the actives together with a minimum intra-cluster similarity (in terms of number of matching "1"s) of 10 features amongst the actives while the inactives had a maximum similarity of only 8 features with any active. We also obtained a separation of the actives and the inactives in terms of the number of "1"s in the selected feature space. The actives had at least 25 features with a value of "1" and the inactives had a maximum of only 14 features with a value of "1". Thus, we felt confident about classifying instances based on distance from the centroid of the actives and inactives. This method yielded a weighted accuracy of close to 100% on the validation dataset.

Table 1. Results obtained in the Training Phase

Method Used	Wt. Accuracy on	
	training set(%)	validation set(%)
Decision Tree	96.5	64.9
SVM	93.1	60.0
Naive Bayes	64.7	49.9
Clustering	100	95.0

We were happy with this result and awaited the release of the test dataset.

3. Testing Phase

With the release of the test dataset we faced the third and the most important of the challenges — the differences in the distributions of the training and the test datasets. The organizers had also warned that "*the compounds in the test set were made after chemists saw the activity results for the training set, so the test set might contain a higher fraction of actives than did the training set. But there would still be more inactives than the actives*".

3.1. Differences in Training and Testset Distributions

Some quick summary statistics of the distribution of "0"s and "1"s convinced us beyond doubt of the difference. We found that on an average the number of "1"s in each attribute in the test set was four times more than in the training set. In Figure 4 we show a scatter plot of the fraction of "1"s for each attribute in the test set against the fraction in the train set . From the figure we find that there were several attributes where the fraction of "1"s in the test set was significantly higher.

The increase in the number of "1"s in the test set could be due to two reasons:

- The first could be directly due to an increase in the fraction of actives. We had observed in the training dataset that the actives tended to have more "1"s than the inactives, an increase in the fraction of actives could cause the fraction of "1"s per attribute to increase. The organizers had already revealed that the fraction of actives was indeed higher in the test set.

- The second, more subtle reason could be because the test set was created in a biased manner to favor actives. It was possible that the inactives in

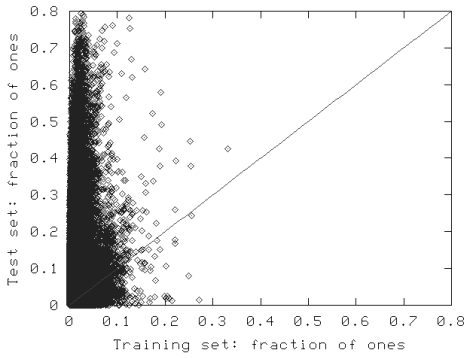
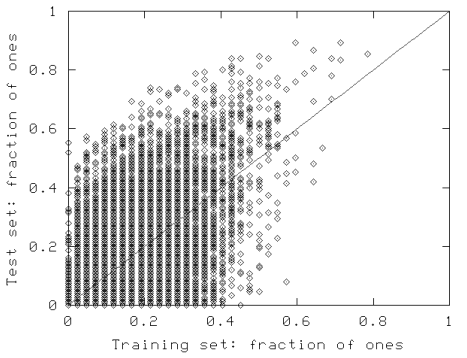
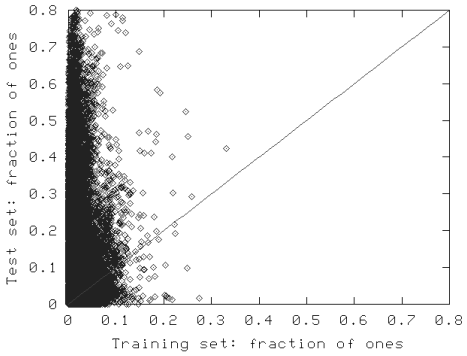


Figure 4. Scatter plot over different attributes of the fraction of ones in test and the training sets. Each of the 139,351 attributes contributes a point in this plot.



(a) Actives



(b) Inactives

Figure 5. Scatter plot of the fraction of ones in test and training sets for different attributes plotted separately for the Active and Inactive classes.

the experiment were very different from the inactives in the training dataset. In particular, they are likely to be closer to the actives and have a higher concentration of "1"s. We had no means of verifying this during the competition. However, a postmortem analysis of the actives and inactives separately shows that the inactives too had a significantly higher count of "1"s in the testset. In Figures 5 we redraw the scatter plots this time separately for the actives and inactives. We find that the inactives have significantly larger number of ones in the test set compared to the training set. The test actives had also changed but not as drastically as the inactives.

Thus the difference in distribution was two-fold (1) the fraction of actives was significantly higher in the test set than in the training (2) for each attribute the fraction of "1"s was higher in the test than in the training set. Some recent papers discuss the first problem of dealing with differences in the class distribution in the training and test set (Provost & Fawcett, 1998; Weiss & Provost, 2001; Zadrozny & Elkan, 2001; Domingos, 1999). However, there is relatively little work on handling the second case.

We next discuss the two techniques that we used to address this change in data distribution.

3.2. Distribution Independent Classifiers

Our clustering approach relied strongly on thresholding on the number of matching ones. With the drastic difference in the frequency of "1"s in the training and test instances, we no longer felt confident about the efficacy of this method.

We wanted a classifier that would be less critically dependent on the specific input data distribution. Discriminative classifiers like decision trees are known to be less sensitive to the input data distribution than generative classifiers like Naive Bayes. However, a single decision tree like the one shown in Figure 1 can pick only a small set of important features in this large dimensional space. We therefore, decided to build an ensemble of decision trees to cover the large feature space through other groups of important features.

The trees in an ensemble are built serially by successively removing attributes already used in the previous trees of the ensemble as shown in Figure 6. Trees are added to the ensemble until the accuracy of the tree on the validation dataset does not drop. This method of constructing tree ensembles presents a departure from the normal method of creation by partitioning or resampling the training dataset. In our case, training data is limited but the number of attributes is abundant. This helped us identify all important groups of useful yet redundant attributes. We were surprised that we could construct nine such serial trees each of which yielded accuracy close to the first tree in spite of the reduced attribute set. This exposed the redundancy present in the input feature space and enabled

us to exploit it to build a stable classifier.

3.3. Semi-supervised Learning

Semi-supervised learning (Nigam et al., 2000; Blum & Mitchell, 1998; Goldman & Zhou, 2000; Bennett & Demiriz, 1998; Joachims, 1999) attempts to incorporate a fraction of the test instances to the training data when building a model. This provides a mechanism of incorporating the characteristics of the test data distribution in the trained model. We used the following method for semi-supervised learning.

1. Repeat for a few rounds
   (a) For each test instance, get predictions using the current ensemble of trees.
   (b) Select a subset of test instances on which the ensemble's prediction has very high confidence.
   (c) Add the subset to the training set with the current prediction and create a new ensemble of decision trees.

Our first attempt at semi-supervised learning selected for inclusion all test instances that got the same prediction from each tree in the ensemble. However, we had no guarantee that unanimity amongst the trees implied correctness. It was important to ensure that we were not drifting away from the "correct" model by amplifying our mistakes in each round of semi-supervised learning. In the second attempt, we set aside a validation dataset and evaluated the errors on the unanimous cases. We did find cases where the prediction was wrong in spite of unanimity. We therefore, wanted a finer-grained distinction between instances.

We made each tree output a weight that denotes the confidence that the predicted instance is active. The weight was equal to the fraction of the total actives correctly classified by the leaf node at which the classification was done. The score of each prediction was the sum of the scores from each tree. The validation dataset was used to determine the score ranges where predictions were error-free on the validation instances. Thus, test instances with score only within this range were incorporated via semi-supervised learning.

At the end of three rounds of semi-supervised learning we had exhausted instances on which the prediction score was high. At the end of the process, we had three rounds of 6 trees each.

Out of the 634 unlabeled instances we incorporated 437 instances (126 actives and 311 inactives) into the training set via the above process. The rest 200 odd instances were treated as confusing instances. To predict

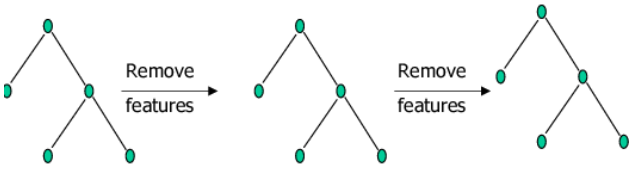


Figure 6. Forming an ensemble of decision trees.

labels for the confusing ones, we built decision trees with the correctly predicted test instances as training data. We verified that the accuracy of these trees on the original training data was high.

With this we submitted our predictions to the organizers of the competition.

3.4. Final Results

With our ensemble of 18 trees built using the limited semi-supervised learning we achieved a weighted accuracy of 64.3% and an unweighted accuracy of 72%. We were ranked second-runner's up in the competition. The winners had a weighted accuracy of 68.5% and unweighted accuracy of 70%. The winners used Bayesian Belief Networks for the task. They perform an initial feature selection step using mutual information to pick about 200 relevant features. Then they learn a few Bayesian networks on the training data and further prune the feature set using a Markov Blanket approach. The final model they used for prediction surpisingly had as low as 4 features in it! Details of the method used by the winners can be found in (Cheng et al., 2002).

3.5. Postmortem

With the release of the true predictions of the test dataset, we carried out some postmortem analysis of our methods. We found that the accuracy of the test set with a single decision tree was 49%. The first round ensemble of decision trees with unweighted majority prediction yielded 57% accuracy and weighted majority predictions yielded 63% accuracy. Semi-supervised learning boosted this accuracy to 64.3%.

4. Further Work on Semi-supervised Learning

The competition got us hooked into doing further research in semi-supervised learning. Over the next few months our research on this topic led to the design of a new algorithm, and on applying this on the KDD Cup data we were surprised to get a weighted accuracy of

71% — higher than what was obtained by the winning entry!

A primary goal we had in this research on semi-supervised learning was to design a method that was not tied to a particular supervised learning method. Most existing approaches to semi-supervised learning (Nigam & Ghani, 2000; Blum & Mitchell, 1998; Goldman & Zhou, 2000; Bennett & Demiriz, 1998; Joachims, 1999) are tied to a particular underlying supervised learners. None of these were directly applicable to our supervised learner — an ensemble of decision tree classifiers. Hence our key motivation was to design a semi-supervised technique independent of any supervised algorithm.

Let Λ_s and Λ_u denote arbitrary supervised and unsupervised learning algorithms, and D_l and D_u denote the set of labeled and unlabeled instances respectively. Our algorithm begins with a hypothesis h that Λ_s learns using only D_l. This h is used to label D_u. Simultaneously, we use Λ_u to cluster $D_u \cup D_l$ into clusters such that each cluster has diameter less than d. The d is a function of the minimum distance between two instances of the opposite class in the initial training set D_l. Modulo noise, we want d to be less than this minimum distance. Unlabeled instances from clusters satisfying a purity criterion are included in the training set as follows. A cluster is called *pure* if it contains instances belonging to the same class (as determined using the current hypothesis). The pure clusters are then ranked based on the fraction of instances in that cluster that come from the labeled set D_l. Larger the number of labeled instance in a cluster, higher is its rank. Each of these pure clusters are then evaluated in that order for possible inclusion in the training set. We evaluate the worth of including unlabeled instances in a cluster using N fold cross-validation on the supervised classifier Λ_s. A cluster is included if its inclusion results in a decrease in the cross-validation error. Once a set of unlabeled instances are included, the classifier is retrained and the process of including new unlabeled data repeated until no more pure clusters are found.

As already mentioned, the algorithm achieved a weighted accuracy of 71% on the KDD Cup dataset. We also tested its performance on some of the standard UCI datasets and found good improvements over supervised learning. Further details of the algorithm can be found at (Janakiraman & Sarawagi, 2002).

5. Lessons Learnt

The challenges presented by the competition, namely that of skewed class distribution, large feature space and non-representativeness of the training data are relevant when dealing with real world applications.

Our experience of working on this dataset taught us the following lessons:

1. An ensemble of decision tree classifiers appears to be a promising method of dealing with differences in training and test data distribution.

2. High-dimensional datasets often contain redundant sets of attributes. This redundancy can be exploited to build robust and stable classifiers like an ensemble of trees. This is particularly useful when the size of the training set is small.

3. Non-representativeness of the test instance could be because of differences in the class distribution or due to a more fundamental difference in the distribution of the input attributes. While there has been some work in recent times in addressing the first type of difference, there is relatively little known about how to handle the second kind of difference.

4. Semi-supervised learning can be useful in handling differences in training and test data distributions by allowing some of the sure test instances to influence the predictions of the other test instances.

References

Ali, K., Manganaris, S., & Srikant, R. (1997). Partial Classification using Association Rules. *Proc. of the 3rd Int'l Conference on Knowledge Discovery in Databases and Data Mining.* Newport Beach, California.

Bennett, K. P., & Demiriz, A. (1998). Semi-supervised support vector machines. *Advances in Neural Information Processing Systems*, *12*, 368–374.

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *COLT: Proceedings of the Workshop on Computational Learning Theory.* Morgan Kaufmann Publishers.

Chakrabarti, S., Dom, B., Agrawal, R., & Raghavan, P. (1998). Scalable feature selection, classification and signature generation for organizing large text databases into hierarchical topic taxonomies. *VLDB Journal: Very Large Data Bases*, *7*, 163–178.

Chan, P. K., & Stolfo, S. J. (1998). Toward scalable learning with non-uniform class and cost distributions: A case study in credit card fraud detection.

Knowledge Discovery and Data Mining (pp. 164–168).

Cheng, J., Hatzis, C., Hayashi, H., Krogel, M.-A., Morishita, S., Page, D., & Sese, J. (2002). Kdd cup 2001 report. *Sigkdd Explorations*, *3*. `http://www.acm.org/sigs/sigkdd/explorations`.

Collobert, R., & Bengio, S. (2001). Svmtorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, *1*, 143–160. Software available from `http://www.idiap.ch/learning/SVMTorch.html`.

Dash, M., & Liu, H. (1997). Feature selection for classification. *Intelligent Data Analysis*, *1*, 131–156.

Domingos, P. (1999). Metacost: A general method for making classifiers cost-sensitive. *Knowledge Discovery and Data Mining* (pp. 155–164).

Fawcett, T. (1996). Learning with skewed class distributions– summary of responses. *Machine Learning List*, *8*.

Goldman, S., & Zhou, Y. (2000). Enhancing supervised learning with unlabeled data. *Proc. 17th International Conf. on Machine Learning* (pp. 327–334). Morgan Kaufmann, San Francisco, CA.

Janakiraman, A., & Sarawagi, S. (2002). A hybrid approach to semi-supervised learning. Master's thesis, Kanwal Rekhi School of Information Technology. sunita@it.iitb.ac.in.

Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. *Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI'2000)*.

Joachims, T. (1998). Text categorization with support vector machines: learning with many relevant features. *Proceedings of ECML-98, 10th European Conference on Machine Learning*.

Joachims, T. (1999). Transductive inference for text classification using support vector machines. *Proc. 16th International Conf. on Machine Learning* (pp. 200–209). Morgan Kaufmann, San Francisco, CA.

Kohavi, R., & John, G. H. (1997). Wrappers for feature subset selection. *AI Journal special issue on relevance*, *97*, 273–324.

Kohavi, R., Sommerfield, D., & Dougherty, J. (1996). Data mining using MLC++: A machine learning library in C++. *Tools with Artificial Intelligence* (pp. 234–245). IEEE Computer Society Press, available from `http://www.sgi.com/tech/mlc/`.

Koller, D., & Sahami, M. (1996). Toward optimal feature selection. *International Conference on Machine Learning* (pp. 284–292).

Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: one-sided selection. *Proc. 14th International Conference on Machine Learning* (pp. 179–186). Morgan Kaufmann.

Nathalie Japkowicz, C. (Ed.). (2000). *Learning from imbalanced data sets, papers from the aaai workshop, technical report ws-00-05*. AAAI Press.

Nigam, K., & Ghani, R. (2000). Understanding the behavior of co-training.

Nigam, K., Mccallum, A. K., Thrun, S., & Mitchell, T. (2000). Text classification from labeled and unlabeled documents using EM. *Machine Learning*, *39*, 103–134.

Provost, F. J., & Fawcett, T. (1998). Robust classification systems for imprecise environments. *AAAI/IAAI* (pp. 706–713).

Quinlan, J. R. (1993). *C4.5: Programs for machine learning*. Morgan Kaufman. software available from `http://www.cse.unsw.edu.au/~quinlan/c4.5r8.tar.gz`.

Vapnik, V. N. (1995). *The nature of statistical learning theory*. Heidelberg, DE: Springer Verlag.

Weiss, G., & Provost, F. (2001). *The effect of class distribution on classifier learning* (Technical Report ML-TR 43). Department of Computer Science, Rutgers University.

Zadrozny, B., & Elkan, C. (2001). Learning and making decisions when costs and probabilities are both unknown. *In Proceedings of the Seventh International Conference on Knowledge Discovery and Data Mining (KDD)*.