

The Needles-In-Haystack Problem

Katherine Moreland¹ and Klaus Truemper²

¹The MITRE Corporation, McLean, VA 22102, U.S.A.

²Department of Computer Science, University of Texas at Dallas,
Richardson, TX 75083, U.S.A.

Abstract. We consider a new data mining problem of detecting the members of a rare class of data, the needles, that have been hidden in a set of records, the haystack. Besides the haystack, a single instance of a needle is given. It is assumed that members of the needle class are similar according to an unknown needle characterization. The goal is to find the needle records hidden in the haystack. This paper describes an algorithm for that task and applies it to several test cases.

Key words: Logic, Classification, Feature Selection

1 Introduction

In helicopters, the shaking induced by the main and tail rotors causes fatigue and ultimately failure of critical components. To aid the analysis of the fatigue process, military helicopters continuously carry out various measurements and record them in an onboard system. In part, this information is used to evaluate and schedule replacement of parts and components before they begin to fail. In a recent case, routine helicopter maintenance discovered a critical part that according to the maintenance schedule should have been still okay, but actually had failed and at any time could have caused catastrophic failure of the helicopter. Possibly other helicopters in the fleet were similarly close to catastrophic failure. How could those helicopters be identified using the measurements of the helicopter with the failed part and of all other helicopters? This paper describes a method for that task.

We begin by introducing a new data mining problem called the *needles-in-haystack problem*. A collection of vectors of length n called the *haystack* is given. In addition, a single vector of length n called a *needle* is provided. A few of the vectors in the haystack are similar to the needle vector according to an unknown relationship. An oracle is available which accepts any vector from the haystack records and tells whether or not it is a needle. The objective is to identify all hidden needle vectors within the haystack while minimizing the number of calls made to the oracle. In this paper, we describe an algorithm that solves the problem reasonably well under certain assumptions.

In the helicopter case, the data recorded for a given helicopter are summarized in a vector of length n . The vector of the helicopter that is close to catastrophic

failure is the known needle, and the vectors of the remaining fleet constitute the haystack. The vectors of the helicopters that are similarly close to failure are the hidden needles. The maintenance procedure that removes the critical part and determines whether that part has failed is the oracle.

On the surface, the needles-in-haystack problem is similar to the outlier detection problem [2, 12, 17, 18]. However, outliers do not necessarily exhibit common features, while the needles are assumed to share a common, unknown characteristic. The needles-in-haystack problem also may seem similar to the task of separating a very small set of records from a much larger one [8, 12, 15, 16, 20, 21]. But here all members of the small set save one are unknown.

2 Worst-case Performance of Solution Algorithms

Every solution algorithm for a restricted version of the needles-in-haystack problem in the worst case evaluates every haystack record with the oracle. Specifically, we have the following result.

Theorem 1. *Assume that (1) the haystack contains just one hidden needle; (2) the known needle and the haystack records are binary; (3) the unknown needle characterization can be described by a logic conjunction of the literals of the record attributes, where the 1s (resp. 0s) of records are interpreted as True (resp. False). Then for any solution algorithm, there is an infinite subclass of needles-in-haystack instances where the number of calls to the oracle is exponential in the size of the records and where the oracle evaluates every record of the instances.*

Proof. For each instance of the subclass, the needle is a record of all 1s, say of length n , while the haystack is the collection of $(0, 1)$ vectors of length n having $\lfloor n/2 \rfloor$ 1s and $n - \lfloor n/2 \rfloor$ 0s. One haystack record is marked as the hidden needle. It is easy to see that the unknown needle characterization necessarily is a conjunction of $\lfloor n/2 \rfloor$ nonnegated literals of the attributes. If a given solution algorithm finds the hidden needle without evaluating all haystack records with the oracle, then for an arbitrarily selected haystack record that has not been evaluated, there is an automorphism of the haystack that maps the hidden needle to that record. The haystack produced by the automorphism requires more oracle queries than the original instance. By induction, there is a case where the algorithm evaluates all haystack records. Since the haystack size is $\binom{n}{\lfloor n/2 \rfloor}$, it is exponential in n .

A similar result can be proved when the known needle and the haystack records are real and are converted to binary vectors using cutpoints. A corresponding worst-case theorem says that the number of calls to the oracle is exponential in the size n of the records and linear in the number of cutpoints. The proof is analogous to that for Theorem 1.

The proof of exponential number of oracle calls of Theorem 1 is no longer valid when the number of literals of the unknown needle characterization is known or assumed to be either small or close to n . Indeed, in the latter case,

nearest-neighbor measures such as Hamming distance seem reasonable tools for solution algorithms. More challenging, and it would seem more interesting for applications, is the case where the number of literals is small instead of close to n .

In the general case of real vectors, the above results motivate us to consider a tight upper bound on the number of cutpoints in addition to demanding that the number of literals in the needle characterization be small. Indeed, we consider the needles-in-haystack problem under the following assumptions.

Assumption 1 *The unknown needle characterization can be described using a logic conjunction which contains only a small number of literals. The attributes giving rise to these literals are not known a priori.*

Assumption 2 *The logic conjunction of Assumption 1 is based on a discretization involving only one cutpoint for each attribute. The cutpoints are not known a priori.*

Note that the two assumptions do not impose any restriction on the number of hidden needles. But later we require that we have a rough estimate of that number.

3 Summary of Algorithm

The solution algorithm is iterative. At the outset of each iteration there are k known needles, h haystack records, and l attribute subsets which in prior iterations led to needle candidates that were identified by the oracle as non-needles. When the algorithm begins, $k = 1$, h is the total number of haystack records, and $l = 0$. Let H denote the current haystack.

For each of the k needles, several *artificial* needles, which look similar to the k needles on hand, are constructed as follows. For each attribute of the data set, the variance is estimated using the haystack records. Using the estimated standard deviation, σ , and a parameter α , we define a *width* w by

$$w = \alpha \cdot \sigma \tag{1}$$

For each of the known needles, we carry out the following step. We define an interval for each attribute centered at the attribute value and having width w . Using the continuous (resp. discrete) uniform distribution if an attribute is continuous (resp. discrete), we randomly create several artificial needles. The artificial needles are added to the set of k needles to produce a set S .

In the solution process, we invoke a separation algorithm that separates S from H . The algorithm creates an ensemble of classifiers which in turn produce a vote total ranging from -40 to 40 for each record of H . Details are included in Section 5. Generally, the records of S produce a vote total near 40 , while almost all records of H result in a vote total near -40 . Indeed, records of H with a vote total well above -40 may be needles. By enforcing a threshold, we could declare

all records of H with a vote total above the threshold to be hidden needles. This simple approach works well when the data sets are randomly generated. However, when real-life data sets are used, the method performs quite poorly. We improve upon the method as follows. After sets S and H have been constructed, we discretize them using a rather complicated process described in Section 4 that also determines candidate attribute sets. For each of these candidate attribute sets, we call the separation algorithm to separate set S from H as described previously. The record from H with the highest vote is selected as a candidate for testing with the oracle. If the record is confirmed to be a needle, it is added to the set of k needles and the process continues iteratively, now with $k + 1$ known needles, $h - 1$ haystack records, and l attribute subsets. If the record is a non-needle, the attribute configuration which led to the selection of this non-needle is stored, l is incremented, and the algorithm continues with the next candidate attribute set. The algorithm terminates if all candidate attribute sets have been exhausted without identifying any additional hidden needles.

4 Discretization

Recall that the discretization step not only discretizes the data, but also produces candidate attribute subsets that potentially provide the correct attributes needed for the characterization of the needles. Three facts are exploited by the algorithm to accomplish this task. First, needles are known to be rare. Second, Assumption 1 guarantees that the unknown logic conjunction characterizing the needles contains few literals. Third, Assumption 2 assures that we only need to consider one cutpoint for each attribute. Details of the discretization method are provided next.

4.1 Attribute Pairs

Define an attribute that is used in the unknown needle characterization to be a *needle attribute*. Otherwise, the attribute is a *non-needle attribute*. Suppose the needle attributes were given. For any pair of attributes, the following possible scenarios exist: (1) both attributes are needle attributes, (2) exactly one attribute is a needle attribute, or (3) both attributes are non-needle attributes. Consider the values of one such attribute pair plotted in the plane with one attribute on the x -axis and the other on the y -axis.

Consider a pair of needle attributes. Assume we have a cutpoint for each attribute. The two cutpoints define four disjoint quadrants in the plane. Each record of the data set falls into one of the quadrants. If the two cutpoints are correct for computation of the unknown needle characterization, all known and hidden needles fall within the same quadrant. Since the total number of needle records is known to be small, we expect the quadrant containing the known and hidden needles to be sparsely populated compared to other quadrants.

For example, consider two needle attributes x and y , with values ranging from 0 to 10. The cutpoint for attribute x is at 4.0 while the cutpoint for attribute y

is at 5.0. Let there be $k = 2$ known needles. Suppose the lower right quadrant defined by these cutpoints contains 4 points, two of which are the known needles. This case is depicted in Scenario C of Figure 1. The lower right quadrant is very sparsely populated compared to the other three quadrants. Since it contains all given needles and few additional points, any one of the additional points may be a hidden needle record.

Now consider the case of a needle attribute paired with a non-needle attribute. The cutpoint of the non-needle attribute is not required to characterize the needles. Assume that the needle attribute corresponds to the y -axis. Using only the needle attribute cutpoint produces two horizontal regions instead of quadrants. For example, in Scenario A of Figure 1 the needle attribute y has the cutpoint 4.0. Suppose the known needles fall within the lower region. This region is sparsely populated compared to the upper region and therefore any one of the additional points may be a needle record. Scenario B of Figure 1 shows the analogous case where x is the needle attribute with cutpoint 3.0. This produces two vertical regions with the rightmost region containing the given needles.

For the final case of two non-needle attributes, for any pair of cutpoints, either the given needles do not fall within the same quadrant or they fall within a densely populated quadrant. In either case, the two attributes likely are not useful for characterizing the needles. Scenario D of Figure 1 depicts such a case, assuming that the k known needles are near the center of the displayed region.

Since the needle attributes are actually unknown, we estimate for each attribute pair which of the aforementioned scenarios applies. Details are provided next.

4.2 Cutpoint Selection

Consider the values for the two attributes of an attribute pair plotted in the (x, y) -plane. Define R_0 to be the smallest axis-parallel rectangle of the (x, y) -plane that contains the known needles and the points of the haystack. Define another axis-parallel rectangle R_1 to be the smallest possible rectangle that encloses all known needles. We define a *box* to be the smallest rectangle that contains exactly one of the corner points of R_0 and the rectangle R_1 . There are four such boxes. Define a *band* to be the smallest rectangle that contains exactly one side of R_0 and the rectangle R_1 . There are four such bands. Each box corresponds to one cutpoint on the x -axis and to one cutpoint on the y -axis. Each band corresponds to just one cutpoint on one of the two axes.

We introduce an additional assumption.

Assumption 3 *A rough estimate e of the total number of known and hidden needles is available.*

All points contained within a box or band are considered to be potential needles. We want to ensure that the boxes and bands do not contain too many points since needles are known to be rare. Let $p(B)$ be the number of points in a box/band B . We use B for the discretization decision only if it satisfies the

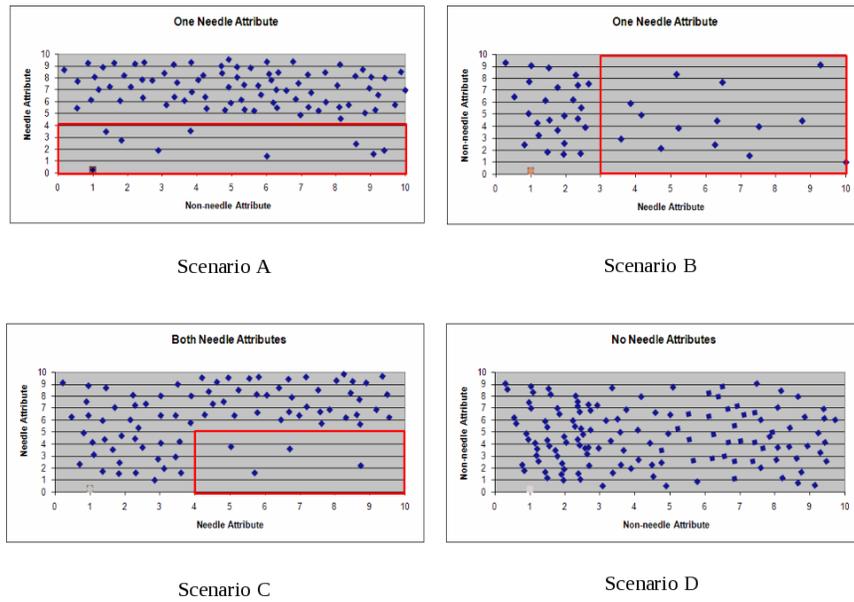


Fig. 1. The graphs illustrate the four possible scenarios for cutpoint selection. Scenarios A and B show examples of a horizontal and vertical band, respectively, when only one attribute in the pair is a needle attribute. Scenario C illustrates an example of both attributes being needle attributes, which yields a sparsely populated quadrant. Scenario D shows an example of neither of the attributes being needle attributes, assuming that the points of the given needles are near the center of the displayed region.

following condition using a parameter $\beta > 1.0$.

$$p(B) \leq \beta e \tag{2}$$

By definition, boxes/bands contain all given needles. Let k be the number of given needles. Since we want to use boxes/bands B to identify additional hidden needles, we only consider boxes/bands B that contain at least one additional point. Thus, we enforce the constraint

$$p(B) \geq k + 1 \tag{3}$$

Boxes/bands meeting these criteria are stored as *candidate boxes/bands*.

4.3 Box/Band Comparisons

We need a way to compare two boxes/bands of any two attribute pairs so that we can determine the attribute pairs that most likely are part of the characterization of the needles. For a box/band B , let $a(B)$ be its area, and define $v(B)$ to be the density of points in B . That is,

$$v(B) = \frac{p(B)}{a(B)} \tag{4}$$

The smaller $v(B)$, the more likely the box/band B is useful for construction of the needle characterization.

Let B be a box. As described above, two cutpoints correspond to B . Using just one of those cutpoints at a time, we derive two bands B_1 and B_2 . We evaluate the usefulness of the box B relative to the usefulness of B_1 and B_2 via

$$d(B) = \min\{v(B_1), v(B_2)\} - v(B) \tag{5}$$

That is, the larger $d(B)$, the more we prefer B over B_1 and B_2 .

Suppose there is at least one candidate box for a given attribute pair. We select from the possible choices one candidate B that maximizes $d(B)$, breaking ties randomly. If $d(B) > 0$, we declare B to be the *representative box* of the attribute pair. Note that a given attribute pair may not produce a representative box.

5 Construction of Candidate Attribute Sets

We construct a graph G . Each attribute produces a node. If we have determined a representative box for a pair of attributes, then we connect the corresponding two nodes by an edge. At this point, the graph may have isolated nodes. We check for each isolated node whether we have at least one candidate band. In the affirmative case, we assign to the isolated node a candidate band B with minimum $v(B)$ value, breaking ties randomly, and consider B to be the *representative band* of the isolated node. Finally, we delete all isolated nodes having

no representative band. The resulting graph is G . The nodes of G corresponding to the attributes in the as-yet-unknown characterization of the needles likely define a clique (= complete subgraph) of G with, say, m nodes. Accordingly, we use the cliques of G to define candidate attribute subsets for the iterative algorithm. Generally, any appropriate method may be employed to find the cliques of G . In our implementation, we limit m to 3 and apply direct enumeration. To each clique with $m = 2$ or $m = 3$ nodes, we assign as value the average of the $d()$ values associated with the representative boxes of the clique edges. We sort these cliques in decreasing order, then append to the list the cliques with $m = 1$, that is, the isolated nodes, in order of increasing $v()$ values associated with the representative bands of the isolated nodes.

The needle detection algorithm processes the cliques of the list one by one in the given order and declares the attributes corresponding to the node sets of the cliques to be the candidate attribute sets. We select the discretization cutpoints as follows. In the case of a clique with $m = 1$ or $m = 2$ nodes, the associated representative band or box directly supplies the cutpoints. For $m = 3$, the two edges incident at a node of the clique correspond to two representative boxes that may imply two distinct cutpoints for the node. There are two cases, depending on whether the projections of the two boxes onto the line of the attribute of the node are nested. If the projections are nested, we take the cutpoint of the larger projection. Otherwise, we select the cutpoint of the projection produced by the box with larger $d()$ value, breaking ties randomly.

The evaluation of each attribute set as described in Section 3 can be carried out by any separation algorithm as long as the algorithm also identifies the haystack records which cannot be separated, as these records are candidates for being hidden needle records. For candidate separation algorithms, see for example [1, 3–7, 9–11, 13, 14, 19]. We have elected to use the Lsquare algorithm of [13, 14]. The Lsquare algorithm produces vote totals ranging from -40 to 40 for all records of the data set by creating an ensemble of classifiers. Based on [13], Lsquare also computes two probability distributions for the vote totals that may be used to estimate the probability that classification based on the vote total is correct. In the specific case here, a -40 vote total for a haystack record signifies that the record likely is not a needle. As the vote total increases from -40 , the record is less likely to be a non-needle record, and thus may well be one of the hidden needle records.

We call the entire algorithm consisting of the box/band discretization, the clique selection, and the Lsquare evaluation, *NeedleSearch*. We emphasize that *NeedleSearch* requires a rough estimate e of the total number of needles for the bound (2), but that the termination condition does not use e . Indeed, *NeedleSearch* stops when all cliques of G have been evaluated and none of them has produced an additional needle. Therefore, if the haystack does not contain any hidden needles contrary to expectations, *NeedleSearch* will terminate with that conclusion after processing of the cliques of the initial graph G .

6 Computational Results

For testing of NeedleSearch, we used sets of the UC Irvine Machine Learning Repository as well as a data set supplied by D. Thévenin of the University of Magdeburg in Germany. Of the 11 most popular data sets from the repository, we selected the Heart Disease, Iris, and Wine sets since they are of reasonable size and mainly have continuous-valued attributes. The data set from the University of Magdeburg is a fluid dynamics data set called Optim which has continuous-valued attributes. Table 1 summarizes the data sets.

Table 1. Summary of Data Sets

Data Set	No. of		Needle Records	Non-needle Records
	Rec's	Attr's		
HeartDisease	303	14	Class = 0	Class > 0
Iris-1	150	5	Class = 1	Class > 1
Iris-2	150	5	Class = 2	Class = 1 or 3
Iris-3	150	5	Class = 3	Class < 3
Wine-1	178	14	Class = 1	Class > 1
Wine-2	178	14	Class = 2	Class = 1 or 3
Wine-3	178	14	Class = 3	Class < 3
Optim-1	60	9	Low value for 1st target	High value for 1st target
Optim-2	60	9	Low value for 2nd target	High value for 2nd target
Optim-3	60	9	Low value for 3rd target	High value for 3rd target
Optim-4	60	9	Low value for 4th target	High value for 4th target

We make sure that each case satisfies Assumptions 1 and 2 by selecting needles from the specified set as follows. For a given case, let a set N contain all records matching the needle class value while a set P set contains the records with the other class values. The Lsquare method is called to obtain a separating formula for the two sets. The first clause in the separating formula is chosen to be the underlying needle relationship. Six records of set N receiving the highest possible vote of 40 are retained as they are well-separated from the set P using the selected clause. Likewise, the P records with the lowest possible vote of -40 are declared to be the non-needle records. The haystack is composed of all needles save one and all non-needles. An exception is the Optim case, where only four needle records could be derived.

The Iris-2 data set required more than one cutpoint for separation, while the separation formula for the Wine-1 data set did not supply a short conjunction. Accordingly, we could not verify Assumptions 1 and 2 and hence deleted these two test cases.

Table 2 shows the results achieved by NeedleSearch for the test cases. The number of iterations required to detect the 1st, 2nd, 3rd, 4th, and 5th hidden needles are given in the table for each of the cases. For example, all but the third hidden needle of the Wine-3 case was identified on the first try. The third

hidden needle took a total of three iterations to be identified by the algorithm. This means the algorithm identified two records which were declared by the oracle to be non-needles before correctly identifying a hidden needle.

Table 2. Needle Detection Results

Case	Number of Runs to Detect Needle Number					Total Runs
	One	Two	Three	Four	Five	
HeartDisease	1	1	1	1	1	5
Iris-1	1	1	1	1	1	5
Iris-3	4	1	3	1	9	18
Wine-2	1	1	1	1	10	14
Wine-3	1	1	3	1	1	7
Optim-1	5	1	1	n/a	n/a	7
Optim-2	1	1	1	n/a	n/a	3
Optim-3	1	1	1	n/a	n/a	3
Optim-4	1	1	1	n/a	n/a	3
Average	1.78	1.0	1.44	1.0*	4.4*	
Cum Avg	1.78	2.78	4.22	5.22*	9.62*	

* Values do not include cases Optim-1 - Optim-4

In all runs, the parameters α of (1) and β of (2) are selected as $\alpha = 0.1$ and $\beta = 1.5$. On average, the algorithm detects the first hidden needle in 1.78 tries. The second hidden needle is detected on the first attempt without identifying any non-needles. The fifth and final hidden needle is the most difficult for the algorithm to detect and on average involves 4.4 tries. Overall, the algorithm on average makes 1.75 calls to the oracle to find one needle.

A possible criticism of the test process is that the Lsquare algorithm is first employed to determine the logic conjunctions defining the needles of the test data, and later is used in NeedleSearch to separate the known and artificial needles from the haystack records to find candidate needle records. However, the discretization of attributes and the construction of candidate attribute sets of NeedleSearch as described in Sections 4 and 5 involve no part of Lsquare. In addition, NeedleSearch restricts each application of Lsquare to a specified subset of attributes and their respective cutpoints. Due to the small size of the candidate attribute sets and the enforced discretization, that separation task is simple, and it is reasonable to suppose that substitution of Lsquare in NeedleSearch by other logic-based separation methods involving an ensemble of classifiers would produce similar results.

7 Summary

This paper introduces the needles-in-haystack problem in which a small number of needle records are hidden among haystack records and are to be found. As a

guide for the search, just one needle is given. It is shown that worst-case performance of any solution algorithm requires evaluation of all records of haystacks whose size is exponential in the dimension of the records. Relying on two reasonable assumptions, a solution algorithm is proposed that discretizes the needle and haystack records by a particular method and also identifies attribute subsets likely to be useful for characterization of the needles. The algorithm attempts to separate the needle and haystack records using those attribute subsets. Records in the haystack which are not readily separated from the needle class are candidates for the hidden needles, and an oracle is called to determine whether they belong to the needle class. The algorithm is iterative in nature and uses newly discovered needles to help characterize the needle class in subsequent iterations.

The algorithm has been tested using several data sets. On average, the algorithm made 1.75 calls to the oracle to find each hidden needle. In each case, all hidden needles were detected. Potential application areas include aircraft maintenance, fraud detection, and homeland security.

A key assumption in the current work is that the characterization of the needles can be achieved using a small number of literals and only one cutpoint per attribute. In future work, we will relax these constraints to handle more complex needle characterizations.

References

1. S. Abidi and K. Hoe. Symbolic exposition of medical data-sets: A data mining workbench to inductively derive data-defining symbolic rules. In *Proceedings of the 15th IEEE Symposium on Computer-based Medical Systems (CBMS'02)*, 2002.
2. C. Aggarwal and P. Yu. Outlier detection for high dimensional data. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, 2001.
3. R. Agrawal, T. Imielinski, and A. N. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, 1993.
4. A. An and N. Cercone. Discretization of continuous attributes for learning classification rules. In *Proceedings of the Third Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-99)*, 1999.
5. S. Bay and M. Pazzani. Detecting group differences: Mining contrast sets. *Data Mining and Knowledge Discovery*, 5:213–246, 2001.
6. E. Boros, P. Hammer, T. Ibaraki, and A. Kogan. A logical analysis of numerical data. *Mathematical Programming*, 79:163–190, 1997.
7. E. Boros, P. Hammer, T. Ibaraki, A. Kogan, E. Mayoraz, and I. Muchnik. An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, 12:292–306, 2000.
8. N. V. Chawla, N. Japkowicz, and A. Kotcz. Editorial: special issue on learning from imbalanced data sets. *SIGKDD Explor. Newsl.*, 6(1):1–6, 2004.
9. P. Clark and R. Boswell. Rule induction with CN2: Some recent improvements. In *Proceedings Fifth European Working Session on Learning*, 1991.
10. W. W. Cohen. Fast effective rule induction. In *Machine Learning: Proceedings of the Twelfth International Conference*, 1995.

11. W. W. Cohen and Y. Singer. A simple, fast, and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, 1999.
12. P. Dokas, L. Ertöz, V. Kumar, A. Lazarevic, J. Srivastava, and P.-N. Tan. Data mining for network intrusion detection. In *Proc. 2002 NSF Workshop on Data Mining*, 2002.
13. G. Felici, F. Sun, and K. Truemper. Learning logic formulas and related error distributions. In *Data Mining and Knowledge Discovery Approaches Based on Rule Induction Techniques*. Springer, 2006.
14. G. Felici and K. Truemper. A MINSAT approach for learning in logic domain. *INFORMS Journal of Computing*, 14:20–36, 2002.
15. M. V. Joshi, R. C. Agarwal, and V. Kumar. Mining needle in a haystack: classifying rare classes via two-phase rule induction. In *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 91–102, 2001.
16. M. V. Joshi, V. Kumar, and R. Agarwal. Evaluating boosting algorithms to classify rare classes: Comparison and improvements. *Data Mining, IEEE International Conference on*, 0:257, 2001.
17. W. Lee and S. Stolfo. Real time data mining-based intrusion detection. In *Proceedings of the 7th USENIX Security Symposium*, 1998.
18. K. Sequeira and M. Zaki. Admit: Anomaly-based data mining for intrusions. In *Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.
19. E. Triantaphyllou. *Data Mining and Knowledge Discovery via a Novel Logic-based Approach*. Springer, 2008.
20. G. M. Weiss. Mining with rarity: a unifying framework. *SIGKDD Explor. Newsl.*, 6:7–19, 2004.
21. R. Yan, Y. Liu, R. Jin, and A. Hauptmann. On predicting rare classes with svm ensembles in scene classification. *Acoustics, Speech, and Signal Processing, 2003. Proceedings. (ICASSP '03). 2003 IEEE International Conference on*, 3:III–21–4 vol.3, April 2003.