Subverting Knowledge Discovery in Adversarial Settings

J. G. Dutrisac D.B. Skillicorn

January 2008
External Technical Report
ISSN-0836-0227-
2008-543

School of Computing
Queen's University
Kingston, Ontario, Canada K7L 3N6

Document prepared January 23, 2008

Abstract

Knowledge-discovery technologies are assessed by how well they model real-world situations, but little attention has been paid to how robust they are when the data they use has been deliberately manipulated. We show that it is straightforward to subvert some mainstream prediction technologies (decision trees, support vector machines), and clustering technologies (expectation-maximization and matrix decompositions) by the addition of a small number of records. The resulting models have predictable regions where records describing the existence, properties or activities of 'bad guys' can be concealed from analysis.

Subverting Knowledge Discovery in Adversarial Settings

J.G. Dutrisac and D.B. Skillicorn

Abstract: Knowledge-discovery technologies are assessed by how well they model real-world situations, but little attention has been paid to how robust they are when the data they use has been deliberately manipulated. We show that it is straightforward to subvert some mainstream prediction technologies (decision trees, support vector machines), and clustering technologies (expectation-maximization and matrix decompositions) by the addition of a small number of records. The resulting models have predictable regions where records describing the existence, properties or activities of 'bad guys' can be concealed from analysis.

1 Introduction

In adversarial settings, there are those who wish to manipulate and subvert knowledge discovery to conceal their existence, properties, or activities. It is helpful to think of the knowledge-discovery process as consisting of three phases:

1. **Capture**: this encompasses all of the ways in which records about people, properties, and activities are collected. Many channels are possible: video; audio; text from email, web postings as well as more traditional writing; commercial transactions such as credit card purchases; communication transactions such as web usage or phone calls; and travel data.

2. **Analysis**: this encompasses all of the computation that is applied to data records to produce models of the systems from which they came.

3. **Decision and Action**: This encompasses the results of the knowledge discovery, both direct and indirect.

Manipulating or subverting knowledge discovery always has the ultimate goal of influencing the decision and actions taken because of it. This might be as simple as trying to evade attention because of privacy concerns; trying to manipulate a relationship with an organization to get preferential service; or trying to conceal criminal or terrorist activity.

One possible way to subvert the knowledge-discovery process is to use an insider who can alter the collected data, alter the way in which analysis is done, or change the resulting decisions and/or the actions they lead to. We will not consider this possibility further. Rather, we address subversion from the 'outside' – where it can only be done by manipulating (altering) the data that is captured. Even though all data is collected during the first phase, and so this is the only opportunity for manipulation, it is useful to separate strategies for subversion based on the phase they target. In particular, the ability to subvert a phase depends, to some extent, on knowledge of how that phase works, so different attackers will have different capabilities.

For example, manipulation aimed at the data-capture stage tries either to evade capture completely or to cause the attributes of the captured data records to contain wrong values. Putting a hood over a CCTV camera is an example of the former; wearing a false beard is an example of the latter. It is particularly attractive to cause the wrong value to be collected for the *identity* attribute in a record, for example by using someone else's credit card.

Manipulation aimed at the decision and action stage uses *social engineering*, causing wrong decisions to be made or wrong actions to be taken, despite the correct operation of the rest of the knowledge-discovery process. For example, Mezrich, in his book, *Bringing Down the House* [8], illustrates how a group of students, counting cards in casinos, worked hard to create the impression that they were typical big-spenders who, although they were winning big today, would be sure to lose big tomorrow. As a result, their large winnings did not, for a long time, raise suspicions in the casinos.

However, manipulation can also be targeted at the analysis phase of surveillance. The goal here is to manipulate the data records captured so that the analysis process will fail – in particular, to create regions in the data where records will not cause the 'right' decisions and actions to occur, and so where bad guys can hide.

Mainstream knowledge-discovery technologies use quality metrics that are concerned with the faithfulness of the models built to the real-world system they describe. Little attention has been paid to how robust such technologies are when the data are being deliberately manipulated to create a desired effect. We show that several mainstream knowledge-discovery technologies, for both prediction and clustering, are remarkably fragile with respect to manipulation; and that only a small amount of information about how each technology is being used is needed to subvert it.

2 Subverting prediction

We show how to subvert two mainstream prediction algorithms, *Support Vector Machines* (SVMs) and *Decision Trees*. In what follows, we use a common convention, and describe the records associated with good or normal behavior as *blue* records, and records associated with bad or undesirable behavior as *red* records. The aim of those evading knowledge discovery is to make it possible for some records that 'should' be predicted to be red, to actually be predicted as blue. Furthermore, it is important that the 'bad guys' should be able to tell which records will be misclassified in this way.

Manipulation of the predictors is achieved by inserting carefully chosen records into the training data used to build the prediction models. For each predictor, we illustrate two kinds of attacks:

- A *red-team* attack inserts a record labelled as red into the training data;
- A *blue-team* attack inserts a record labelled as blue into the training data.

It may seem difficult, or perhaps impossible, for an adversary to insert a particular record into the training data as desired. However, this is not as difficult as it looks. First, in many adversarial situations there is a need for ongoing retraining or updating of a prediction model to reflect the changing situation in which it is used – particularly since adversarial situations are always a kind of arms race. Second, an adversary can ensure that the inserted record is of a particularly interesting kind, making it more likely to be chosen for use in training. For example, it may be particularly close to the apparent boundary between the classes, or in a region where there are few known records. Hence, although an adversary cannot force a *particular* record into the training data, it is plausible that *some* records of the right kind can be inserted.

The costs of inserting red-labelled and blue-labelled records are not equivalent. To insert a red-labelled record it may be necessary to sacrifice some person or activity that the record represents, since its red label is the result of detection that the record represents some bad activity. For example, in airline passenger screening, a red record can be generated by arranging for a blonde, female passenger who belongs to a frequent flyer plan and has booked her return ticket well in advance to attempt to travel with the pieces of a weapon in her hand luggage. However, this will almost certainly lead to her arrest, and it may even be necessary to denounce her to ensure that the record is in fact classified as red. In contrast, a blue-labelled record is easy to generate – it requires arranging for a young male of appropriate appearance to buy a one-way ticket at the last minute with cash, in the business-class part of the plane – but carry nothing that could remotely be interpreted as suspicious. a record with apparently bad

It is also not obvious that an attacker can know enough to be able to work out what records will subvert any given prediction algorithm. Although the attacker will not have access to the *actual* data used to train the predictor, it is usually possible for the attacker to gather or infer *similar* data. We will call this the *attacker's training data.*

The attacker's training data is usually sufficient to design the desired attack. In other words, predictors can typically be subverted with only an approximate idea of the structure of the models they embody. In particular, an attacker can use this data to build a model using the same technology that is being attacked, and so get some idea of what the actual model will be like.

2.1 Support Vector Machines

A support vector machine [3, 4] is a two-class classifier that uses an record's attribute values to evaluate its position with respect to a hyperplane. The hyperplane acts as a divider between the two classes and is chosen to give the maximum possible separation between the classes. This may be thought of as inserting the largest possible plank between each class when they are represented in a Cartesian space with coordinates determined by their attribute values.

To achieve the maximum possible separation between classes the algorithm needs to only consider the attribute values of those records close to the boundary. Thus, if new records are inserted into the training data that are closer to the hyperplane than other points of the same class, the shape and direction of the hyperplane will be modified. This provides the opportunity to subvert SVMs.

When support vector machines are applied to new data records, they predict which side of the hyperplane each record lies by the sign of the predicted value. The magnitude of the value, which is the distance of the point from the hyperplane, also gives some indication of how strongly each record is predicted to belong to the class.

2.1.1 The Red-Team Attack

The strategy behind the red-team attack is to insert a record whose attribute values are typical of blue-labelled records, but that is labelled as red. If properly chosen, the result is to move the hyperplane separating the classes in such a way as to open up a region in the record space where new records would previously have been classified as red but will now be classified as blue. An example is shown in Figure 1.

Figure 1 shows two classes as red squares, and blue dots. The initial training data and the hyperplane generated from it are shown in graph (*i*). Graph (*ii*) shows how a test set is classified by this hyperplane – no records are misclassified. Graph (*iii*) shows the addition of a *single* red-labelled record in the training data, denoted by the red asterisk in the lower right-hand corner. This new record causes the hyperplane to rotate. Finally, graph (*iv*) shows that the uppermost red square from the test data is now misclassified as on the blue side of the hyperplane.

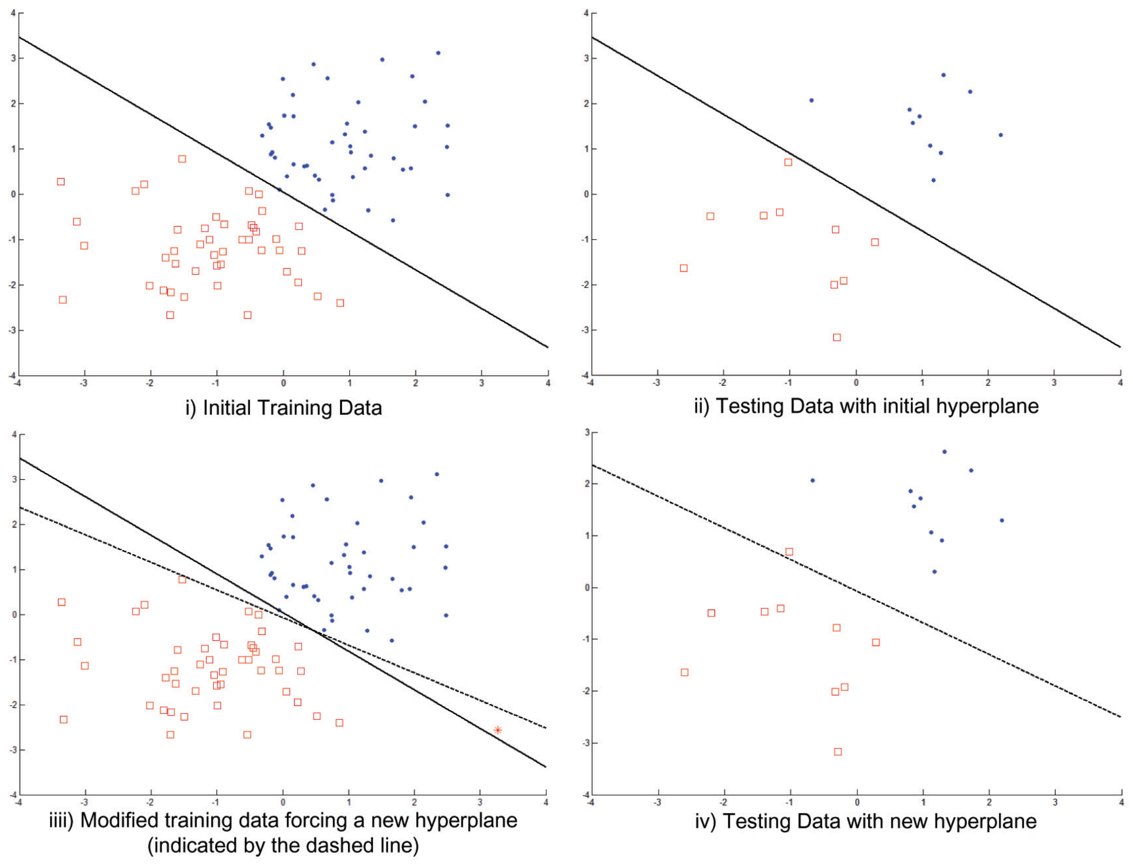

Figure 1: Illustration of a red-team attack against an SVM

The added record's attributes are derived from those of the record that is to be misclassified in the following way. The record to be misclassified is reflected about the centroid of the attacker's training data (that is each individual attribute value is reflected around the mean for that attribute). It may then be necessary to scale the new record (multiply its attributes by a scalar) so that it is on the opposite side of the hyperplane derived from the attacker's training data. In practice, scaling seems only to be needed for datasets with a small number of attributes, fewer than ten.

Support vector machines created to classify records in a densely populated feature space are the most resilient to this attack. There are two reasons for this resilience: first, it is difficult to find an record that is different enough from the training records to make it a likely candidate for misclassification. Next, the dense groupings of the classes allow little room to influence the hyperplane. It is still possible to shift the hyperplane a little, but it is important to focus on records that are as close to the hyperplane as possible. For sparse datasets, those datasets with great distance between each record, and especially those with many attributes (50 and over), this technique works extremely well.

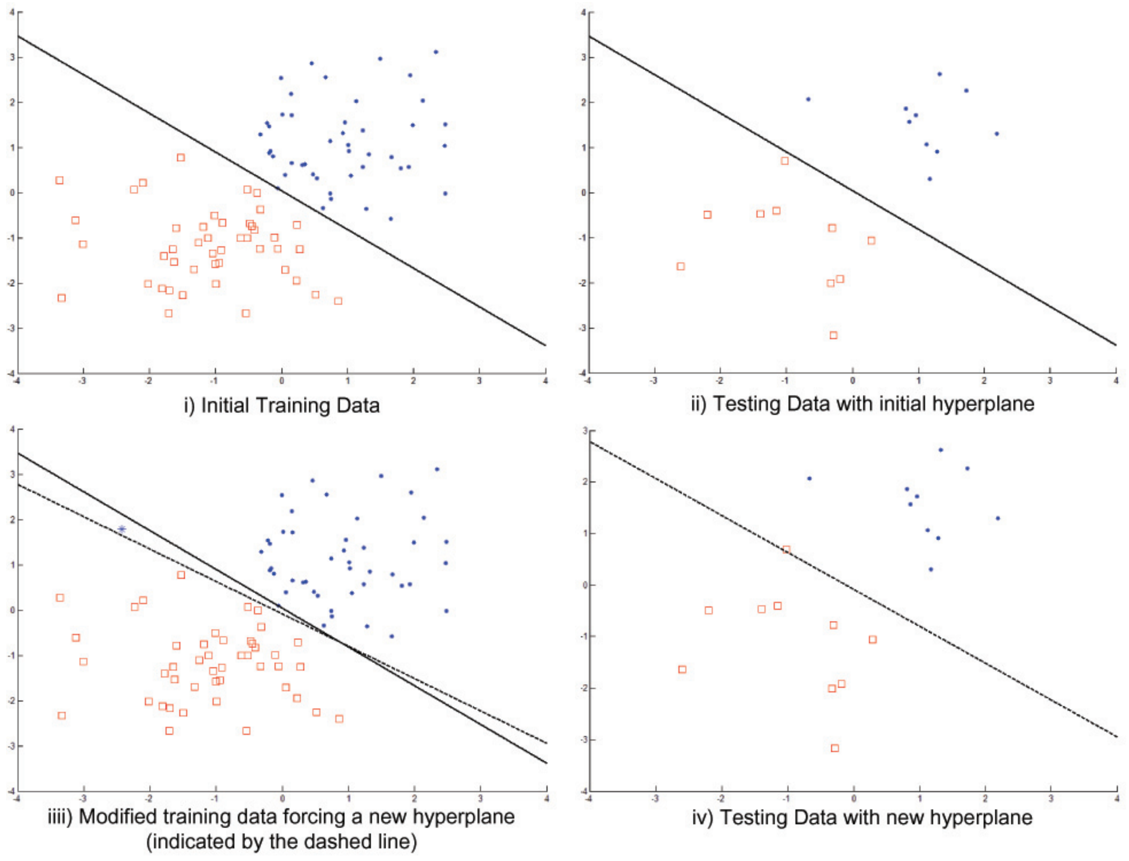

Figure 2: Illustration of a blue-team attack against an SVM

2.1.2 The Blue-Team Attack

The strategy behind the blue-team attack is to add a record with typical red-record attributes, but labelled as blue. As before, this tends to cause the hyperplane separating the classes to move slightly, opening up a region, close to the added record, where records previously labelled red will now be labelled blue.

The blue-team record to be inserted will typically have attribute values similar to the record that we want to be misclassified. As with the red-team attack, scaling is occasionally required to ensure success, but for a different reason. It ensures that the new record will not fall within a dense cluster of red-labelled records, for this will tend to cause the inserted record to be considered an outlier, and ignored.

Figure 2 demonstrates a blue-team attack on the same two-attribute dataset. Like Figure 1 the initial training data and the hyperplane generated from it are shown in graph (i), while graph (ii) shows how the test set is evaluated with respect to this hyperplane. In graph (iii) a single new blue record, denoted by the blue asterisk in the upper left-hand corner, is added to the training data. The new blue record is a copy of the record to be misclassified, scaled by a factor of two. Scaling is required in this case because of the density of the dataset (as expected with only two attributes). The new training record forces the hyperplane to rotate, resulting in the classifier shown in graph

(iv) where the uppermost red square from the test data is now misclassified.

2.1.3 Test Cases

The success of both the red-team and blue-team attacks is demonstrated using three synthetic datasets. Each dataset consists of three pieces: the training set, the test set, and the attacker's training set. The attacker's training set is generated in the same way as the training data and allows the attacker to make inferences about the approximate position of the hyperplane without knowing the real training data.

The red-team records were constructed so that each attribute has mean -1 and standard deviation 1; and the blue-team records were constructed so that each attribute has mean $+1$ and standard deviation 1.

The first dataset demonstrates the subversion of a support vector machine on a large, low-dimensional dataset. The training data consists of 2000 records, 1000 from each class. The test data consists of 200 records, 100 from each class. The attacker's training data consists of 400 records, 200 from each class.

The second dataset is sparsely populated with many attributes. The training data contains 800 records, 400 from each class, with 50 attributes each. The test data again contains 200 records, with 100 from each class. The attacker's training data consists of 400 records, 200 from each class.

The third dataset demonstrates the more probable case that there are fewer members of the red class than the blue class. It has a red class with 100 records, and a blue class with 500 records. Again, the test data consists of 200 records, 100 from each class. The attacker's training set consists of 200 records from each class – the attacker, of course, has no difficulty generating red records.

Because of the high dimensionality of the datasets, it is not possible to represent the results of the experiments graphically in a meaningful way.

The first dataset is densely populated; as a result, only those records closest to the hyperplane are good candidates for misclassification. To identify the easiest record to misclassify using a red-team attack, the distance of each red record in the test set is evaluated to find the one that is closest to the hyperplane derived from the attacker's training data. The closest red record is found to be record 200 with a distance from the hyperplane of $+0.83$ – on the positive side of the hyperplane. This is the record that we will cause the SVM to misclassify. The attribute values of record 200 are:

$$[0.067, 2.700, -0.948, 0.590, -1.600, 0.964, -0.809, -1.269, -0.200, -0.292]$$

The mean value of each attribute in the records of the attacker's training set is calculated so that the attribute values of record 200 can be reflected. This reflection suggests that inserting a new red-labelled training record with attribute values

$$[0.003, -2.665, 0.903, -0.552, 1.646, -1.000, 0.773, 1.136, 0.306, 0.333]$$

is likely to cause a misclassification of record 200. To ensure that scaling is not required, this new record's distance from the hyperplane derived from the attacker's training data is evaluated – it turns out to be -0.76 so the record is clearly on the other, negative side of the hyperplane.

Adding this record to the actual training set shows the success of the subversion: the hyperplane moves, and record 200 is now classified as -0.03 units, just on the negative (blue) side of the hyperplane.

The blue-team attack also attempts to force record 200 to be classified as a blue record. The attack is straightforward in this case – adding a blue record with the same attribute values as record number 200 causes the hyperplane to move, and record 200 is now classified as distance -1.00 from the hyperplane (rather than $+0.83$). Scaling is not required for the misclassification of this particular record because it is so close to the hyperplane.

The second dataset is a sparsely populated data set with 50 attributes and 800 records. Due to the sparse nature of this dataset and the high dimensionality of the hyperplane, all records will tend to be located close to the hyperplane, yet far from one another. This presents ideal conditions for the subversion of a support vector machine.

To illustrate the red-team attack, we choose to hide the red record from the testing set that is *furthest* from the original hyperplane. This is record number 187 at a distance of $+1.80$. This record's attribute values are reflected in the mean attribute values provided by the attacker's training set, yielding a new red record. The new record does not require scaling as it is at a distance of -1.81 from the hyperplane derived from the attacker's training data, that is already well on the 'wrong' side of the hyperplane.

Inserting this new record into the training set for the real data forces the hyperplane to change: record 187 is now at a distance of -0.31, on the blue side of the hyperplane.

The blue-team attack of this dataset, as with the previous example, simply requires the insertion of a blue record with the same attributes as record 187. Again the hyperplane changes, and record 187 is now evaluated as blue, at a distance of -1.00 from the hyperplane (instead of the original $+1.80$).

The third dataset has one-fifth as many red records as blue ones. This does not affect the construction of the hyperplane because only a few records will be used to calculate the hyperplane in any case. Thus, we proceed in a fashion similar to the process for the first dataset.

The record chosen to be misclassified is instance number 111 from the test set. Instance number 111 is at distance $+0.47$ from the hyperplane. The inserted record obtained by reflecting the attributes of this record using the attacker's training data is at distance -1.79 from the attacker's hyperplane Adding this red-labelled record to the training set moves the hyperplane so that record number 111 is classified as a blue record at distance -0.55 from the hyperplane.

Instance number 111 is similarly misclassified if a blue record with the same attribute values is inserted into the training set. Record number 111 is then at a distance of -1.00 from the hyperplane.

These attacks have been based on a particular record that we wish to have misclassified, but the resulting record to be inserted into the training data has values that depend only on information learned from the attacker's training set, that is from a rough approximation to the actual training data to which the subverted record is to be added. This suggests that the approach does not depend heavily on the precise attribute values chosen. Geometrically, we can see that the red-team attack opens up a 'wedge' of space, previously classified as blue but now classified as red, on the *opposite* side to the inserted record. The blue-team attack has a similar effect, but on the *same* side as the inserted record. From this geometric perspective, it seems plausible that many different records will have a similar effect. The advantage of the approach described here is that the 'protected' region is known in advance by construction.

In general, an inserted record should be chosen as close to the existing hyperplane as possible. This has a dual advantage: such a record has the maximum impact on the positioning of the hyperplane, but it also makes the record as 'interesting' as possible and so likely to remain in the training data if any kind of automatic or human preprocessing of the training data takes place.

For a support vector machine, there is no technical advantage to adding extra records to the training data since only a few records become support vectors, and so affect the placement of the hyperplane. The decision about how many records to use depends on human factors such as the likelihood that they will in fact be selected to build a new or updated model.

2.2 Decision Trees

A decision tree is a classifier which models classification by a tree structure. Each non-leaf node of the tree tests an attribute value, while leaf nodes indicate the predicted class. Classification begins at the root and continues until a leaf node is encountered.

The most common decision-tree algorithm is the C4.5 algorithm [2, 10, 11]. The C4.5 algorithm is a *greedy algorithm*, meaning that it builds the tree based on the next best possible step at each phase. The best attribute to test at each internal node is determined using information gain, which measures the difference in the information before a split and after it. At each internal node, the C4.5 algorithm first calculates the information gain for all attributes and then chooses the attribute (and split point) with the greatest information gain. It follows from this, that if the information gain for a given attribute changes, so too will the structure of the tree.

2.2.1 The Red-Team Attack

The red-team attack modifies the tree structure in a desired way by adding specific red records to the training data. By carefully choosing training records, an attacker can lower the information gain of some attributes while increasing it for others, disguising one attribute's ability to act as a good discriminator, while making another attribute seem like a much better discriminator than it is.

Intuitively, it might seem that it is best to alter the information gain of attributes that occur near the root of the tree (those attributes with greater discrimination power or with high levels of information gain). This, however, is not the best approach. Few records can be correctly predicted near the root of the tree; instead, most records require many attribute tests before a reliable prediction can be made. Targeting records with medium to low levels of information gain provides two benefits: The first is that modifying the information gain of attributes near the leaves will have a greater effect on the classification of records. The second is that each subtree is formed from a subset of training data that reaches it. Thus, modifying information gains in one of these subsets is easier, since fewer records influence the calculation.

Consider the example decision tree presented in Figure 3. This figure represents a tree formed from a ten-dimensional dataset, and shows each record's traversal through the tree. The dataset was created in a similar fashion to the support vector machine test cases with red records having mean attribute values of -1 and blue records having mean attribute values of $+1$, both with a standard deviation of 1. A red line indicates the path taken by each red training record, while a blue line indicates the path of each blue record. The value before the colon indicates the attribute being tested at that node; the value after the colon indicates the split value for the test. Records whose attribute value is less than or equal to the split value are displayed below the line; while records whose attribute value is greater than the split value are displayed above the line. The classifications are denoted by the text "red" or "blue" for red and blue, respectively. The test data used to evaluate this decision tree is shown in Figure 4. Each record is correctly classified.

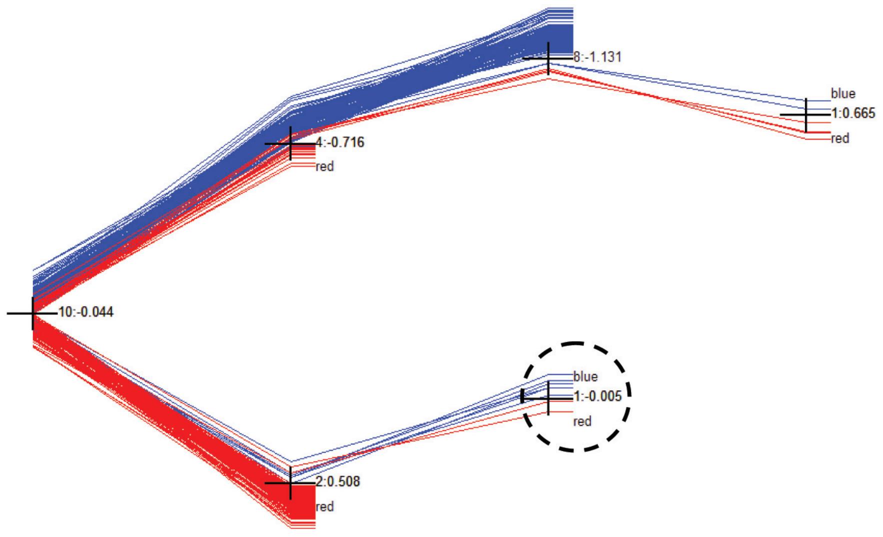

Figure 3: Example decision tree built from training data

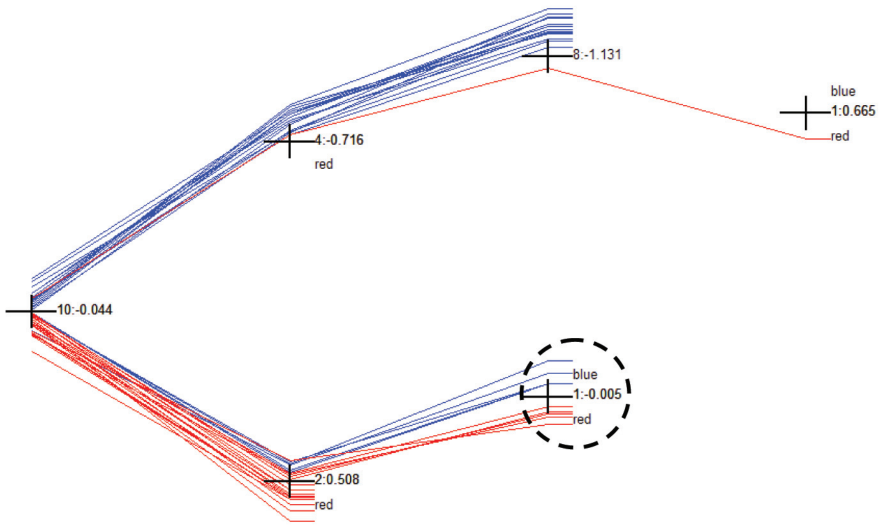

Figure 4: Example test data evaluated by the decision tree

In each of the two figures there is a dashed circle indicating the area that has been identified as an opportunity to force misclassification, for two reasons. First, the test is deep in the tree, so that the information gain will be easy to modify. Second, as can be seen in Table 4.1, the information gain for Attribute 1 is the same as the information gain for Attributes 3, 4, 5, and 6. Each of the records classified based on the value that they have for Attribute 1 in this circle have ambiguous values for Attribute 3.

A red-labelled record with attribute values

$$[0.817, 1.608, -1.639, 0.464, -0.509, 0.579, -2.675, -1.251, -1.459, -2.079]$$

Attribute	Before	After
1	0.8113	0.5577
2	0.4669	0.5577
3	0.8113	0.9183
4	0.8113	0.5577
5	0.8113	0.5577
6	0.8113	0.5577
7	0.4669	0.5577
8	0.4669	0.5577
9	0.4669	0.5577
10	0.2936	0.1972

Table 1: Example Information Gain

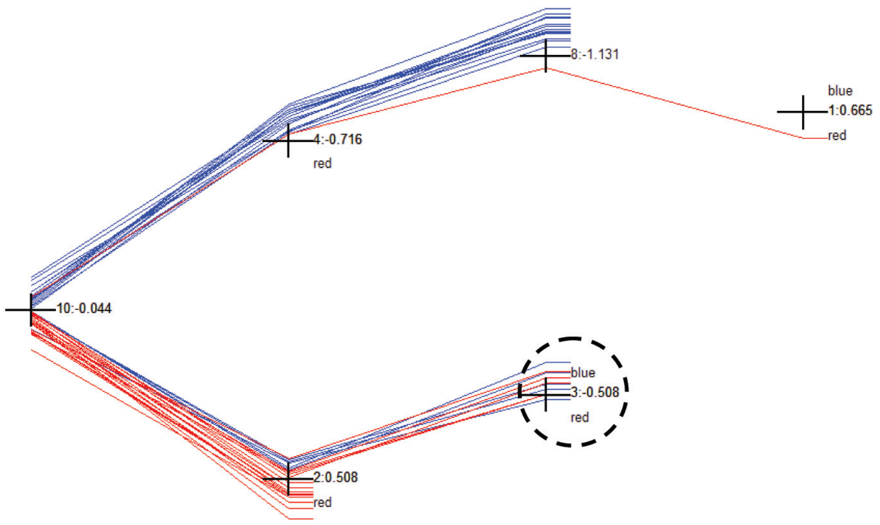

Figure 5: Illustration of red-team attack against a decision tree (note that the misclassified records are located within the dashed circle)

is inserted into the training data. This record has been selected for its ability to lower the information gain of Attribute 1 and all other competing attributes, while raising the information gain for Attribute 3, as can be seen in the "After" column of Table 4.1. This shift in information gain successfully causes the misclassification of five red records as shown in Figure 5.

It is clear that it helps the attacker if the tree structure is known in advance. However, the critical property is not so much what the tree looks like, but which attributes typically appear close to the root of the tree, and which tend not to appear in the tree at all. This can plausibly be estimated from attacker's training data, especially by building multiple trees using slightly different versions of such data.

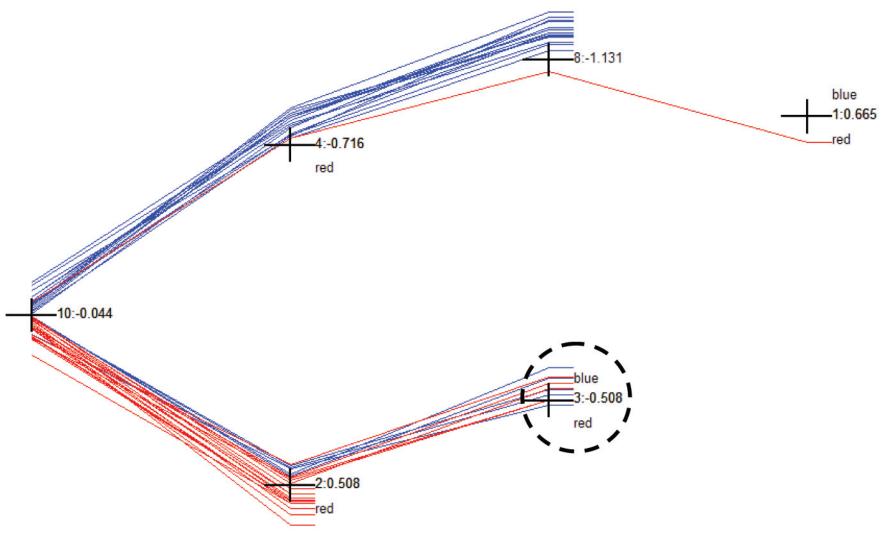

Figure 6: Illustration of a blue-team attack on a decision tree

2.2.2 The Blue-Team Attack Model

The blue-team attack follows very closely from the red-team attack and is very much like the blue-team attack against support vector machines. Again, this modification of the tree structure will come about because of changes in the information gain.

As with the red-team attack, it is best to attempt to misclassify records that are discriminated close to the leaves of the tree. To demonstrate this, consider the data from the red-team attack in the previous section. The same outcome can be achieved using a blue-team attack by inserting a blue-labelled record with attribute values:

$$[-2.418, 2.344, 1.545, -1.976, -1.774, -0.544, -1.674, -0.811, -0.935, -1.781].$$

This modifies the information gain to : 0.458, 0.320, 0.764, 0.458, 0.458, 0.458, 0.320, 0.320, 0.320, and 0.281 for the ten attributes. Attribute 3 now has the highest information gain and thus produces the tree structure seen in Figure 6.

Even if the structure of the tree is unknown, specific records can be targeted for misclassification by inserting similar blue-labeled records. If multiple blue-labelled records are inserted, then the chances are good that the desired red-labelled record will be successfully misclassified.

2.2.3 Test Cases

Due to the level of detail required to show a complete test, we perform only one demonstration of the techniques described above. We will demonstrate these techniques using a particularly difficult dataset. The dataset consists of 2000 records, 1000 from each class. As with the test cases from the support vector machines, a 200 record test set and a 800 record attacker training set have been created with the same parameters. Each record is defined by a series of five attributes, with the attributes of the red class having a mean of -1 and the attributes of the blue class having a mean of $+1$. However, unlike the support vector machine test, each attribute has a different

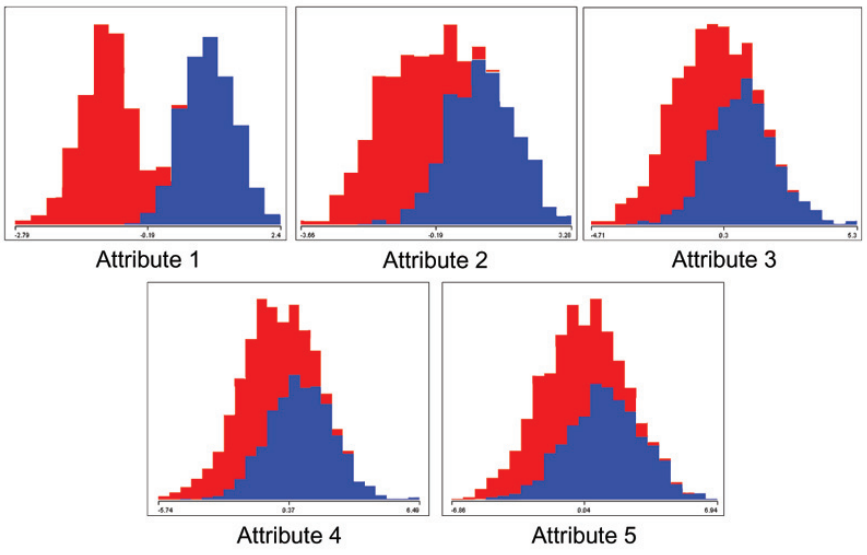

Figure 7: Distribution of records by attribute

	Training Data	Attack Training Data
Attribute	Info Gain	Info Gain
1	0.8563	0.8128
2	0.4753	0.4865
3	0.2616	0.2962
4	0.1735	0.1848
5	0.1309	0.1157

Table 2: Information Gain

standard deviation as follows: Attribute 1: 0.5; Attribute 2: 0.85; Attribute 3: 1.2; Attribute 4: 1.55; Attribute 5: 1.9. Thus, attribute 1 has the greatest discriminating power, while attribute 5 has almost no discriminating power. This can be seen in Figure 7. The information gain from this dataset for all attributes, before any splits, is presented in Table 4.2.

The training data is used by the C4.5 algorithm to generate the tree shown in Figure 8. This classifier is not entirely successful as it incorrectly classifies record 24 from the test data as red and record 162 as blue.

The attacker's training set is used by the C4.5 algorithm to form the decision tree shown in Figure 9. Again, this classifier is not entirely successful, as it causes the misclassification of records 24, 114, and 144.

The red-team attack relies heavily upon understanding the tree structure, but Figures 8 and 9 show that the structures of the two trees are quite different. Still, this attack is possible. The approach, however, is a more generic one that attempts to lower the quality of the tree in some specific areas that seem to be important. From the attacker's training data represented by Figure 9 and Table 2, Attribute 1 is the best discriminator, and will probably always appear at the root

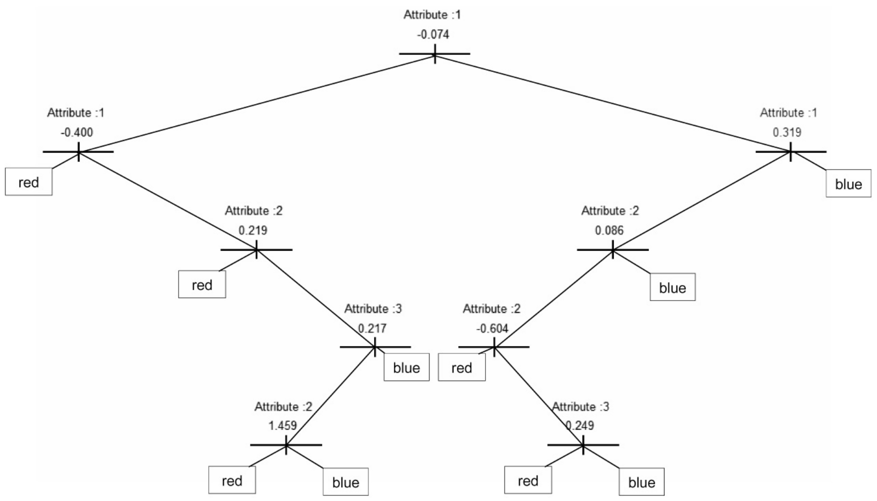

Figure 8: Decision tree built from the training data

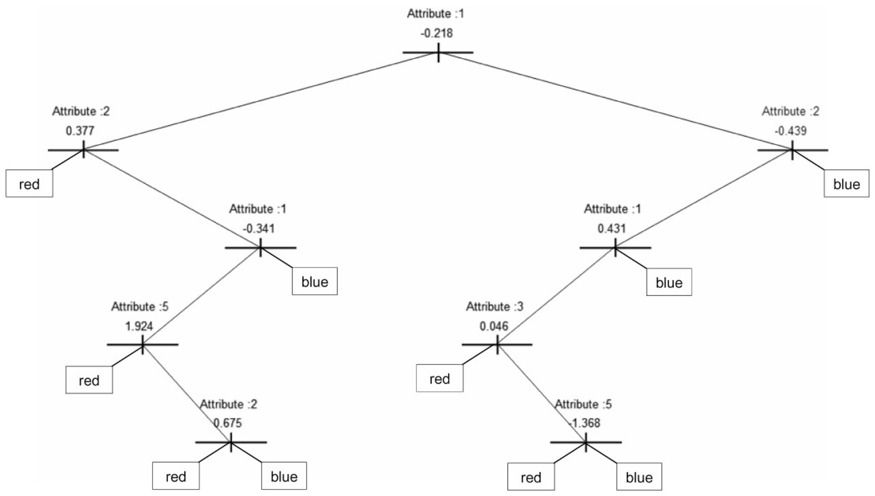

Figure 9: Decision tree built from theattacker's training data

of the tree. Attribute 2 is still a valuable discriminator, but it is not quite as useful as Attribute 1 and its discriminating power is much closer to that of Attributes 3 and 4. Finally, Attribute 5 has very little discriminating power.

The first task is the selection of a general class of record which one wishes to misclassify. With the information have considered above, a red-team attack may be quite successful at causing the misclassification of records that traverse the left side of the tree and are predicted based on the values of Attributes 2 and 3. Yet, if Attributes 4 and 5 suggest that the record is of the blue class, then there is the potential for misclassification. As such, we attempt to create a region for red

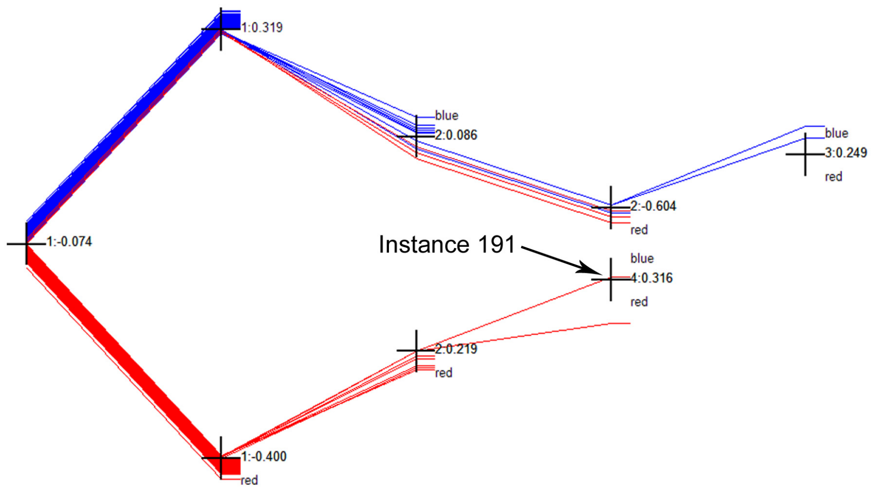

Figure 10: Decision tree after the red-team attack is applied

records to hide in by inserting a red record with attribute values

$$[-0.285, 1.512, 1.324, -0.513, -1.325]$$

This approach results in the successful alteration of the decision-tree structure. However, this new tree is similar to the original one. In fact, only one more record is misclassified: record 191, with attribute values: $[-0.200, 0.253, -1.790, 0.551, -0.301]$, as seen in Figure 10. This example shows that a red-team attack that has limited knowledge of the training data will also have limited success.

The blue-team attack without access to the training data is much more reliable than the equivalent red-team attack. This attack works best by attempting to misclassify multiple red records with similar attribute values. This is because it is often necessary to insert multiple blue records to change the information gain enough to cause the tree structure to change. If two red-labelled records are alike, then the blue-labelled records inserted may work to misclassify both. In addition to this, it is advisable to select red-labelled records to be misclassified that are somewhat similar to blue-labelled records, so that the blue-labelled records added will be seen as an extension of the blue class, and not as outliers.

The attack begins by searching for suitable red-labelled records to misclassify. The goal is to find red-labelled records with ambiguous (that is, close to the split point of an internal decision) values for Attributes 1 and 2, the two most important attributes for classification. A search of the test records for red-labelled records with ambiguous values for Attributes 1 and 2 yields two results that are quite similar, records 128: $[-0.256, 0.252, -1.618, -3.80, -2.619]$, and 197 $[-0.449, 0.411, -1.706, -2.687, 1.040]$. While these two records do not traverse exactly the same path to leaves in the attacker's training tree, a comparison of their first four attribute values shows that they are quite similar. As such, a blue-team attack is attempted on both of these instances. The blue-team attack consists of inserting records from the blue class with similar attribute values to those red-labelled records that we wish to misclassify. In fact, we use exactly their attribute values. The two inserted records cause the decision tree shown in Figure 11 to be built. This figure shows that instances 128 and 197 are in fact misclassified in the new tree.

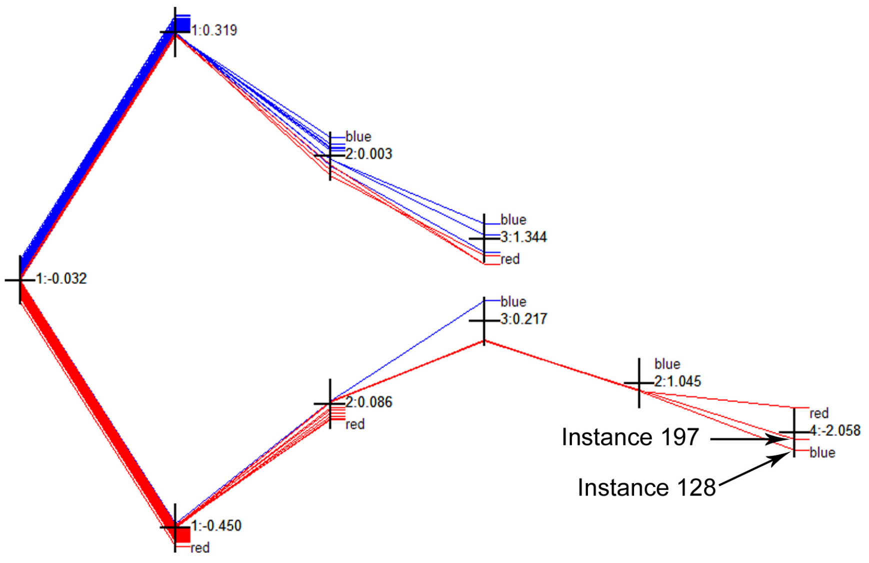

Figure 11: Decision tree after the blue-team attack is applied

2.2.4 Summary

While decision trees are simple classifiers, they are quite resilient to red-team attacks, because they are weak learners. Even slightly different datasets may result in totally different trees due to variations in information gain. These variations are compounded by the fact that, with each change in the tree structure, new sets of attributes are considered in the various subtrees. The red-team attack without exact knowledge of the tree structure may be successful, as in our example, if one carefully considers the likely structure. Still, this attack cannot be considered entirely reliable. Thus, a blue-team attack should be preferred if access to the training data is not possible. The blue-team attack works well because the C4.5 algorithm builds models that tightly fit the data. Thus, so long as one chooses a set of red records that require a deep traversal of the tree to be classified (probably the red records that are like blue records to begin with) the blue-team attack will work easily.

Decision trees are known to be rather weak predictors, so it is not surprising that they are fragile in the face of manipulation. However, support vector machines are one of the best known predictors so their fragility is of more concern. Ensemble techniques are inherently more resistant to the attacks described here, so such predictors should be preferred in adversarial settings.

3 Subverting Clustering and Association Rules

We now consider the problem of subverting clustering algorithms: in particular, *expectation-maximization* and *decomposition-based clustering*. While classification systems use inductive reasoning to determine an record's type, clustering systems identify groups of similar records in populations. We will focus on the situation where there is one large cluster, representing normal records, and a smaller cluster representing bad or undesirable records. The goal of subversion is to conceal the existence of the smaller cluster.

As before, we assume that it is possible to manipulate data collection so as to insert a few carefully chosen records into the data to be clustered in such a way as to conceal some structure that would otherwise be detectable. We also assume that attackers do not have access to all of the available data, but can collect or infer data of a generally similar kind – we call this the *attacker's dataset*.

In adversarial settings, the records describing 'bad guys' are likely to be clustered, rather than appear as single-point outliers, because of the connections among them, and the requirements of their activity. However, it is unlikely that such clusters will be well-separated from normal clusters since, unless they are naive, they will be aware of the possibility of analysis, and will try to appear as normal as possible.

3.1 Expectation-maximization

Expectation-Maximization (EM) [5] is a clustering algorithm that assumes that the attribute values of records belonging to a cluster are distributed around mean values that are the underlying reason for the cluster's existence. The distribution is usually assumed to be Gaussian.

Given a set of records, and a set of k distributions to be fitted to them, there are two different unknowns: the parameters of the distributions (means and variances), and the allocations of records to clusters. The EM algorithm solves for these two unknowns alternately so as to maximize the likelihood of the clustering.

Initially, parameters for the distributions are guessed. In the Expectation step, the likelihood that the observed records were generated from these distributions is estimated, producing tentative assignments of records to clusters. In the Maximization step, estimates for the distribution parameters are calculated by maximizing the likelihood function, based on the tentative assignment of records to clusters. This two-step process is then repeated until the parameters cease to change. The algorithm is guaranteed to converge to a local optimum and, in practice, works quite well.

We now turn to ways of subverting EM clustering. Two clusters can be made to look like one if a group of records (referred to as a mask) is added to the dataset in such a way that the records from both clusters and the mask seem to form a single cluster; or the two original clusters seems to be one cluster, and the mask seems to be another. An example of this is shown in Figure 12, for records with only a single attribute. Before inserting the mask, in graph (i), there are two distinct distributions within the data, each labelled with a dashed line showing its approximate shape. Graph (ii) shows how adding the mask changes the scale of the dataset and causes the distribution to appear closer to Gaussian, illustrated by the dashed line in graph (ii). While the smaller peak is still easily visible to human inspection, it is considered to be part of a single, larger distribution by the EM algorithm – thus effectively hiding the smaller cluster.

Hiding one cluster within another from the EM algorithm requires adding new records to make the overall distribution of attribute values appear more Gaussian. The surprising fact is that the number of records required is not very large. The number of records in the mask should be at least half the number of records in the smaller of the two clusters being merged (but recall that, in adversarial settings, one of these clusters will typically be small to begin with). The mean of the distribution for each attribute value for the records in the mask should be the midpoint of the means for the corresponding attribute in the two clusters. The standard deviation for each attribute value for the records in the mask should be approximately the difference between the mean attribute value of one cluster and the most extreme value of that attribute from the other cluster. The mask therefore introduces a new multivariate gaussian overlaying those of the existing

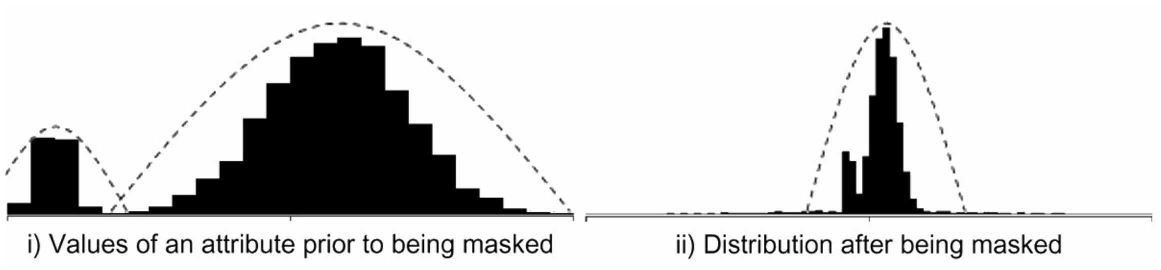

Figure 12: Before and after distribution of attribute values (the scale of graph (*ii*) is approximately 3 times that of graph (*i*) to accommodate the mask)

clusters, centered between them, and with a much wider spread. The records in the mask run some risk, individually, of seeming anomalous, but they play exactly the right role in subverting the EM clustering. Such records are likely to seem like those of eccentrics rather than 'bad guys'.

As before, the required information about the likely clustering structure present in the data, required to generate an appropriate mask, is determined from the attacker's training data.

3.1.1 Test Data

The effectiveness of this attack against expectation-maximization will be demonstrated using three datasets. Each of the datasets has 4500 records and 30 attributes. For each dataset, 4000 records belong to one cluster, and 500 belong to the other. In each case, the two clusters differ from one another in the values of the first three attributes. The attacker's dataset contains 450 records, one tenth the size of the actual dataset. The specific properties of each dataset are described below.

The first dataset contains two clusters, the larger of which has attribute values Gaussian generated with mean 1 and standard deviation 1. The smaller cluster has Gaussian generated values for the first three attributes with means of -1, -0.5, and -1.5, respectively, and standard deviations of $1/5$. All of the other attributes are Gaussian generated with mean 1 and standard deviation 1.

The second dataset demonstrates the application of the attack when the smaller cluster is denser. This dataset is the same as the first except that the first three attributes of the smaller cluster have standard deviations of $1/10$.

The third dataset illustrates the power of this attack when the clusters are further apart. This dataset is the same as the first, except the smaller cluster is further away from the larger cluster. The mean values for the first three attributes of the smaller cluster are -3, -2, and -4 respectively, while the remaining attributes still have means of 1.

The first dataset is easily clustered by the EM algorithm. Using the attacker's training set, we are able to identify the attributes that best separate the two clusters. This is shown in Figure 13.

After analyzing the attacker's dataset and the distributions presented in Figure 13, the parameters for the mask can be determined. The mean value for each attribute in the mask is set to the midpoint between the two cluster means for each attribute. As only the first three attributes have means that differ between the clusters, all other attributes of the mask have a mean of 1. The first attribute of the mask has mean $(1 - 1)/2 = 0$, the second attribute has mean $(1 - 0.5)/2 = 0.25$, and the third attribute has mean $(1 - 1.5)/2 = -0.25$. Next the standard deviation of the mask is determined to be approximately 5.5. This is the greatest distance in any one attribute between

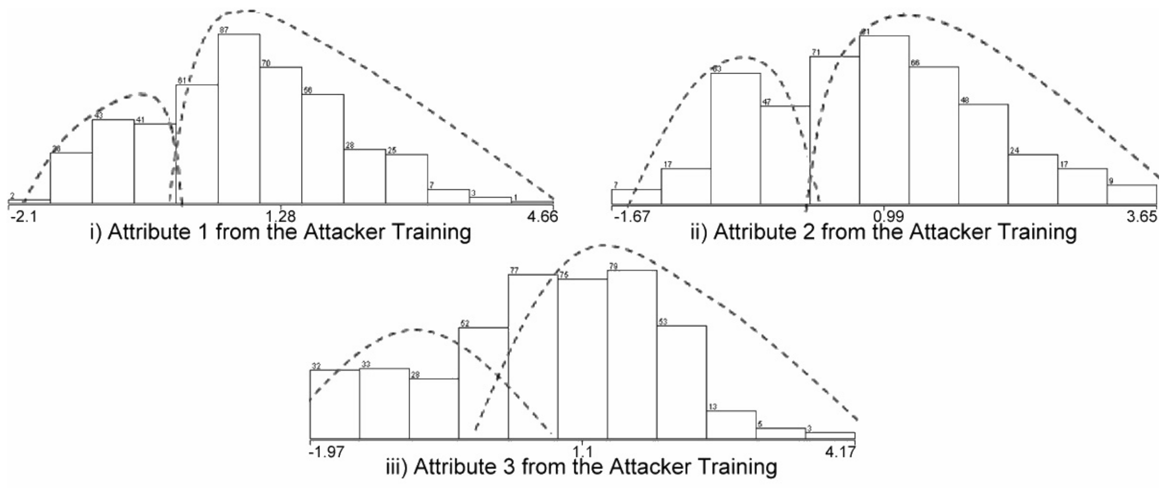

Figure 13: EM Dataset 1: Attacker's dataset attribute distributions

the mean of the small cluster and the furthest record of the larger cluster. This value has been visually determined from Attributes 1 and 3 in Figure 13. This is done by subtracting the mean of the small cluster (-1) from the furthest record of the large cluster (4.66) in Attribute 1 and (4.17) in Attribute 3.

Using the parameters determined, a mask is created for use with the attacker's dataset. The mask that is created uses the same number of records as in the smaller cluster (50). After applying the mask to the attacker's dataset we find that the standard deviation for each attribute may be lowered to 5. With these parameters, the mask is applied to the actual dataset. The resulting distributions are shown in Figure 14, where each attribute appears to have a Gaussian distribution. Performing the EM clustering algorithm on this dataset produces the desired clustering, with the smaller cluster being hidden in the larger cluster and the mask being detected as a new cluster.

The second dataset is the same as the first except that the smaller cluster is twice as densely distributed in the three attributes where it deviates from the larger cluster. This dense distribution makes the smaller cluster much more pronounced when analyzing the distribution of the first three attributes in Figure 15. As the cluster means have not changed, we know that the attribute means for the mask will again be 0 for Attribute 1, 0.25 for Attribute 2, -0.25 for Attribute 3, and 1 for the other attributes. Again, the standard deviation for each attribute is set to 5.5 as this is the greatest distance in any one attribute between the mean of the small cluster and the furthest record of the larger cluster.

Applying this mask to the attacker's training dataset, the standard deviation may again be lowered to 5. With this in mind, a mask of 500 records is constructed using the means and standard deviation outlined above. This mask is then applied to the actual dataset yielding the distributions shown in Figure 16. Inspection certainly shows the large spike in each attribute due to the smaller cluster, but the distributions are actually close to Gaussian. Applying the EM algorithm to the dataset with the mask find the desired clustering, with the small cluster hidden in the larger cluster and the mask appearing as its own cluster. Increasing the density of one of the clusters seems like it might make it more difficult to make that cluster look like part of another Gaussian distribution, but in fact the EM algorithm is still readily fooled.

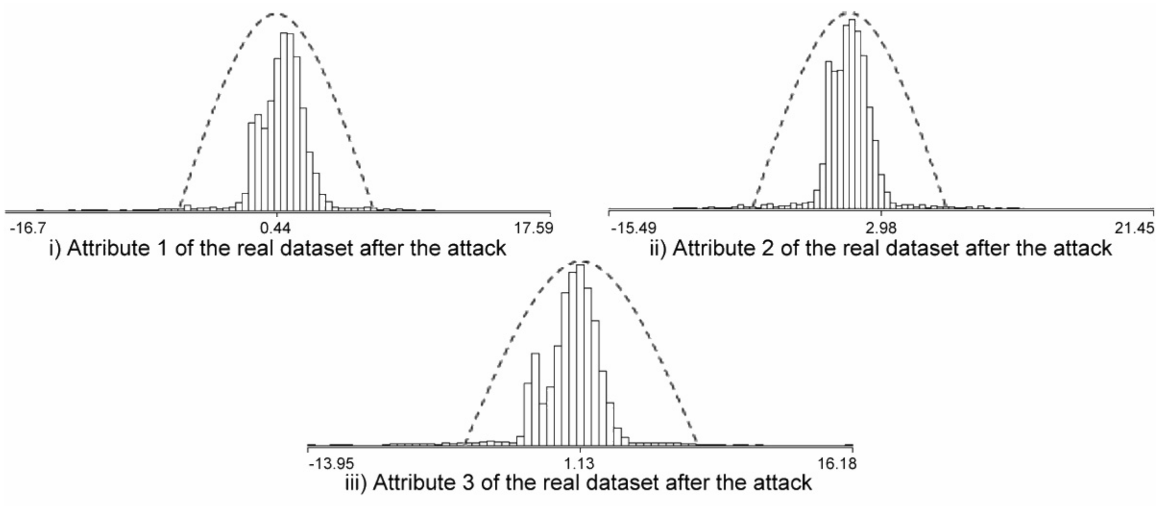

Figure 14: EM Dataset 1: Actual training data attribute distributions after attack

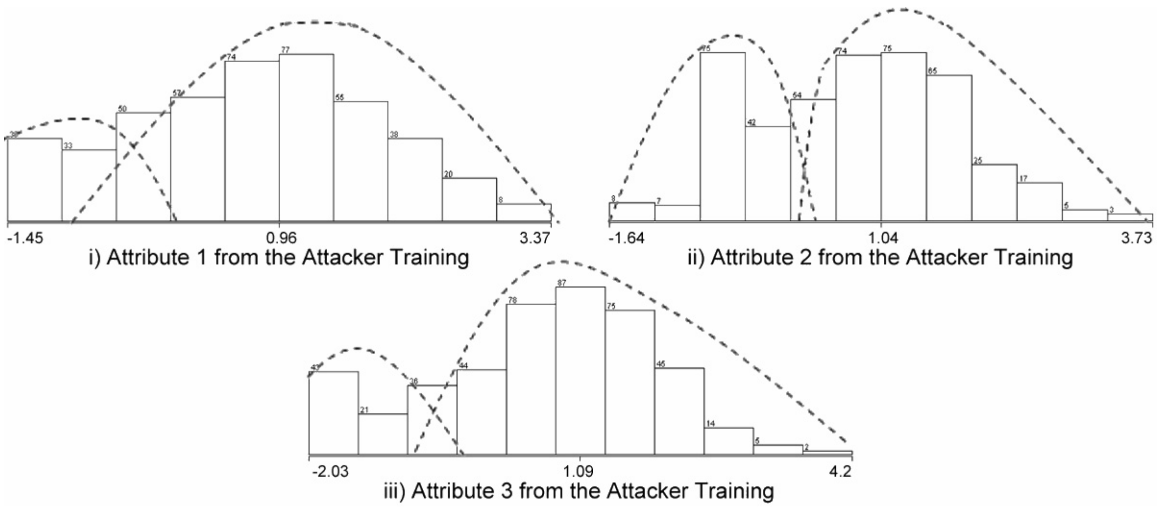

Figure 15: EM Dataset 2: Attacker training data attribute distributions

The third dataset has an increased distance between the two clusters, illustrated in Figure 17 where each attribute shows two clear peaks. The attack that hides the smaller cluster inside of the larger is more difficult with this dataset for obvious reasons. The increased distance between the two clusters requires that the mean values and standard deviation for the mask change. The midpoints between the two clusters, and thus the attribute means for the mask are now: -1, -0.5, and -1.5 for the first three attributes and 1 for all others. The standard deviation for each attribute must now be set to 7.5 as this is the greatest distance in any one attribute between the mean of the small cluster to the furthest record of the larger cluster, as seen in Figure 17.

A mask of 500 records is created and applied to the actual dataset. This mask forces the first three attributes to take on a Gaussian distribution as seen in Figure 18. Furthermore, clustering

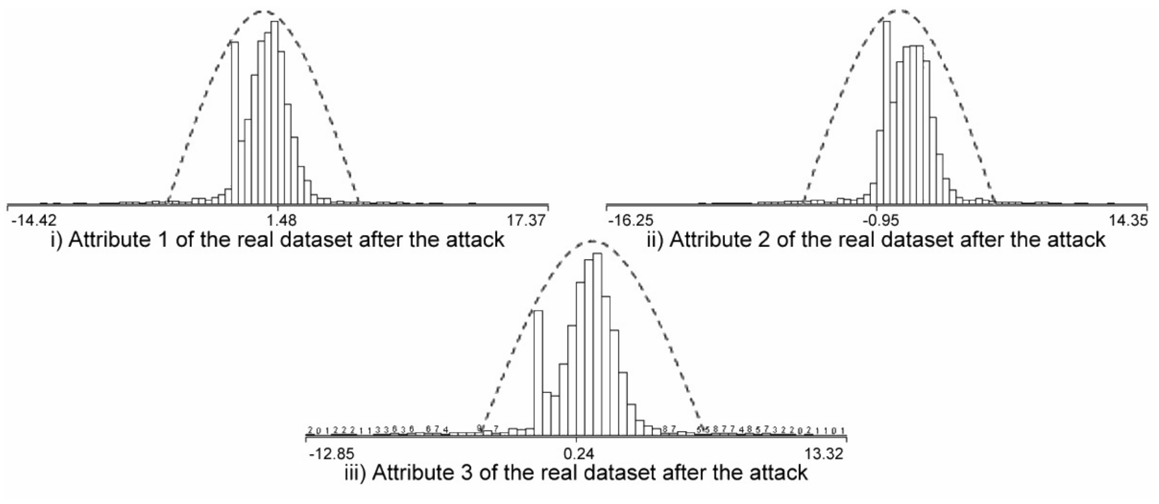

Figure 16: EM Dataset 2: Actual dataset attribute distributions after attack

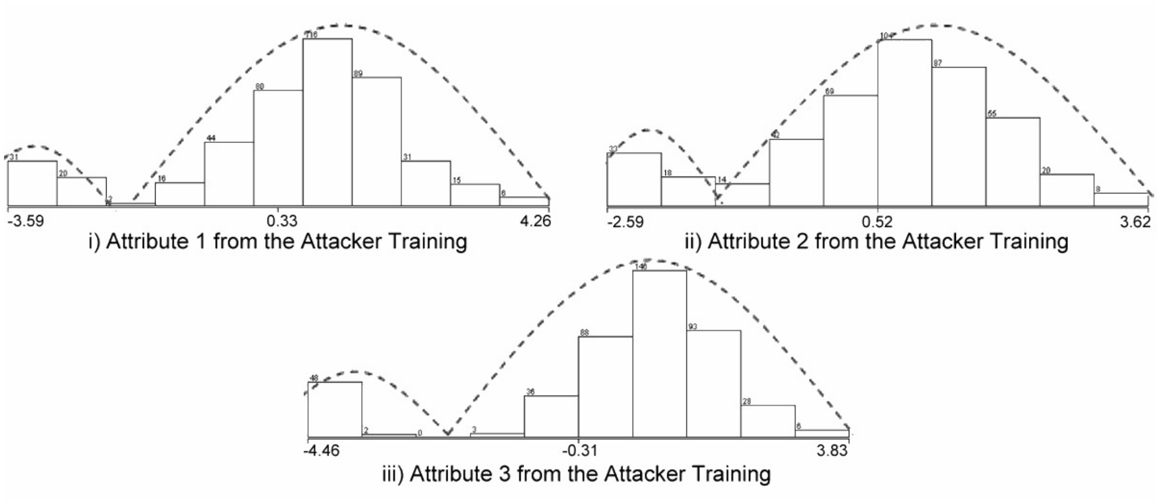

Figure 17: EM Dataset 3: Attacker's dataset attribute distributions

the dataset with the mask yields the desired results, with the small and large cluster seen as one and the mask as another. The success of this attack shows how easily the attack against the EM algorithm scales as clusters move apart.

3.2 Clustering by Decomposition

Matrix decompositions provide ways of clustering data based on particular kinds of structures that each expects to find. We will consider two decompositions that have been used together to discover clusters in large and complex datasets.

The first decomposition is the Singular Value Decomposition (SVD) [6, 12]. Given an $n \times m$

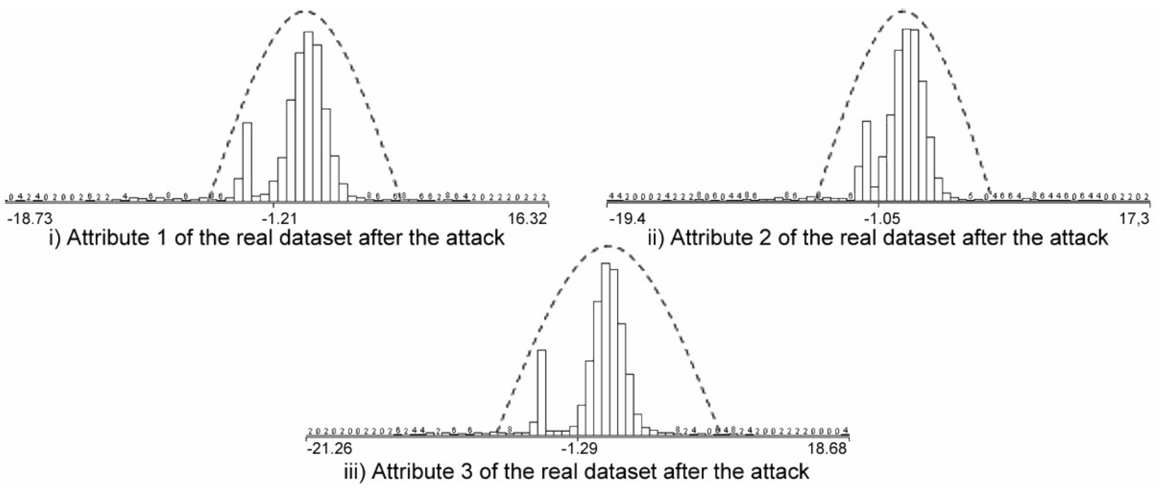

Figure 18: EM Dataset 3: Actual training data attribute distributions after attack

matrix A, representing the records of a dataset, each row can naturally be interpreted as a point in m-dimensional space. However, m is typically quite large, so there is no straightforward way to visualize the records and their relationships using this geometry. The singular value decomposition of A is

$$A = USV'$$

where U is an $n \times m$ orthogonal matrix, V is an $m \times m$ orthogonal matrix (the dash indicates transposition), and S is a diagonal matrix with non-increasing diagonal entries called the singular values.

An SVD implements a change of basis (the new basis captured by V) in which the rows of U correspond to the coordinates of the point associated with each record in the transformed space. However, its most useful property is that the first axis in the transformed space points in the direction of maximum variation in the data, the second axis in the direction of next greatest variation and so on. The SVD can be truncated at some number of dimensions, say k, thus

$$A \approx U_k S_k V_k'$$

where U is the $n \times k$ matrix of the first k columns of U, and S_k is the $k \times k$ upper triangle of S, and so on. This truncation is the most faithful representation of A in k dimensions. If k is chosen to be 2 or 3, the corresponding columns of U_k can be plotted, confident that any visible structure is genuinely present in the dataset, although some structure may not be representable. Clusters can often be detected in this visualization.

The semidiscrete decomposition (SDD) [7, 9] is a similar decomposition

$$A = XDY$$

where X is an $n \times k$ matrix, Y is a $k \times m$ matrix, the entries of X and Y are from the set $\{-1, 0, +1\}$, and D is a diagonal matrix. The value of k can be arbitrary, and may be larger than m. SDD expresses the clusters in the dataset in terms of rectilinearly-aligned regions in the data matrix of similar value. These are captured in the decomposition in terms of stencils expressed by the

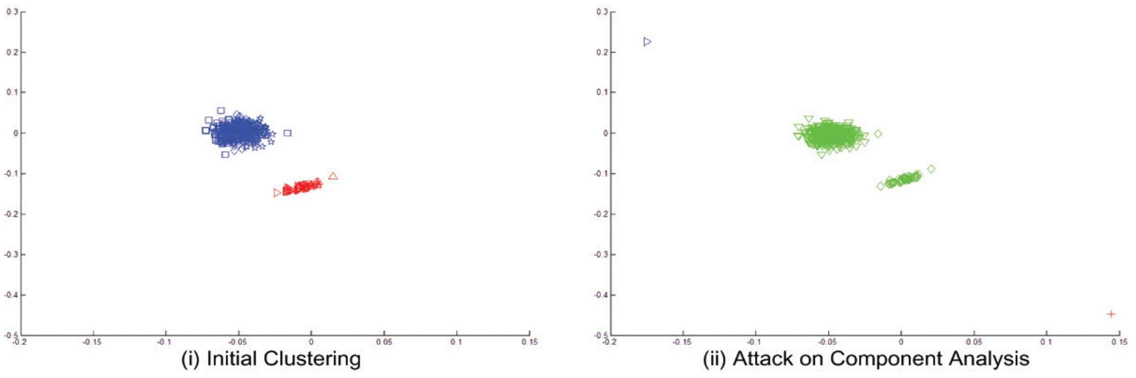

Figure 19: Subverting component analysis by adding two outlying clusters

product of a column of X and a row of Y, with an average height given by the corresponding entry in D. However, for our purposes we need only observe that an SDD provides a ternary hierarchical clustering of the records, given by the entries in the first, and subsequent, columns of X. At the top level the records are divided into three clusters, those with a $+1$, -1, and 0 respectively in column 1 of the X matrix. Each of these clusters can be further subdivided according to the entries in column 2 of the X matrix, and so on. SDD produces the most accurate results when applied to the correlation matrix (AA') of the dataset, and so it will be used in this way.

As the SVD and SDD have different but complementary roles, clustering by decomposition may be considered a two-part process. It is necessary to consider both these parts when attempting to subvert decomposition-based clustering.

Clustering by decomposition relies heavily on the distribution of records in a dataset. Our method of attack again requires adding records to a dataset so that a smaller cluster is hidden by making it seems as if it is part of a larger cluster. The attack against decomposition-based clustering is easier than the EM attack. Only a small number of records need to have extreme values.

The attack against decomposition-based clustering focuses on the cluster labelling generated from the SDD. However, since it is often easy to detect clusters by visualizing the entries of the SVD U matrix, we also consider how to blur cluster boundaries to make them harder to visualize.

The basic strategy is to add records to the dataset that SDD will interpret as extreme clusters, leaving the original two clusters seeming like a single cluster. An example of this is shown in Figure 19.

Graph (i) of Figure 19 shows the clustering of a dataset prior to the attack. In this graph, the smaller cluster is red while the larger cluster is blue. The records of one cluster are labelled $+1$ in column 1 of the X matrix, and those of the other cluster are labelled -1, so it is obvious from the decomposition that there are two clusters in the data. Two new clusters, in fact only two new records, are added; and when the decomposition is applied to this dataset, the result is as shown in graph (ii) of Figure 19. Each of the two added records is detected as a cluster of size 1 (the blue triangle and the red cross), while all of the original records are considered as belonging to a single cluster. (Of course, column 2 of the X matrix will separate the central cluster into its two original pieces, but the same strategy can be applied with two closer records to create new structure at the second level, and so on.) The attribute values of the records in the added clusters are determined

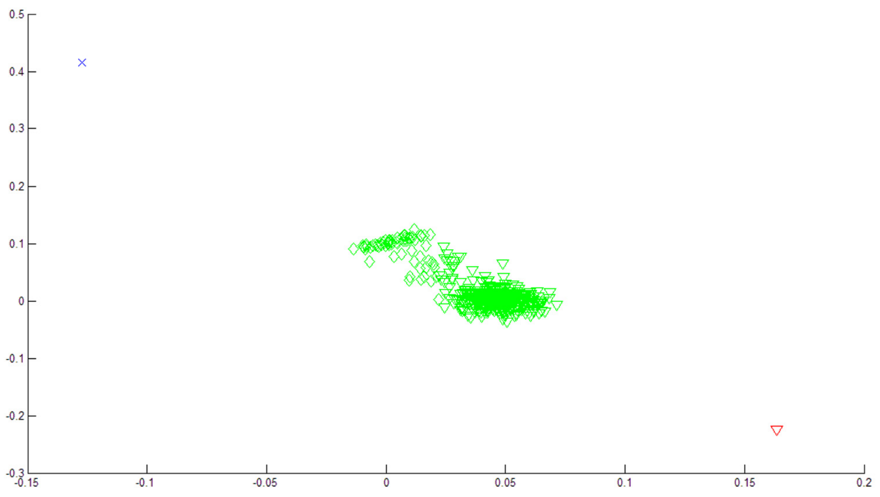

Figure 20: Bridging the two clusters

by the relationship between the two original clusters. The idea is to place the added clusters far from the original clusters along the line joining them.

As the figure shows, it is still easy to detect that the central cluster is really two clusters by inspection. To make this harder, a bridge of extra records that lie in the gap between the existing clusters can be inserted to make the separation less obvious. This is illustrated in Figure 20, where we have inserted $1/5^{th}$ of the number of records in the smaller cluster in between the two original clusters. The mean of each attribute for records in this bridge is the midpoint of the means of the two original clusters, and the standard deviation is two-thirds of the distance between the two original cluster means. This ensures that the records mix well into both clusters. For attributes that have the same mean in both clusters, the bridge is set to that mean and to the standard deviation of the larger cluster.

3.2.1 Test Cases

As before, the test datasets used consist of 4500 records, 4000 of which are in the larger cluster and 500 of which are in the smaller cluster. The attacker's dataset is ten percent of the size of the real dataset.

The first dataset consists of 4500 records with 30 attributes. The two clusters are different in the first eight of these attributes. While the larger cluster has a mean of 1 and a standard deviation of 1 for each attribute, the first eight attributes of the smaller cluster have a mean of -2 with a standard deviation of $1/5$. The remaining attributes in the smaller cluster have both mean and standard deviations of 1.

The second dataset differs from the first only in the standard deviation of the attributes between the two classes. The records of the smaller cluster are twice as densely distributed in the first eight attributes, with a standard deviation of $1/10$.

The third dataset shows how the attack is affected as the clusters move further apart. The first eight attributes of the smaller cluster have a mean of -5 while maintaining the standard deviation of $1/5$. The remaining attributes are unchanged with a mean of 1 and a standard deviation of 1.

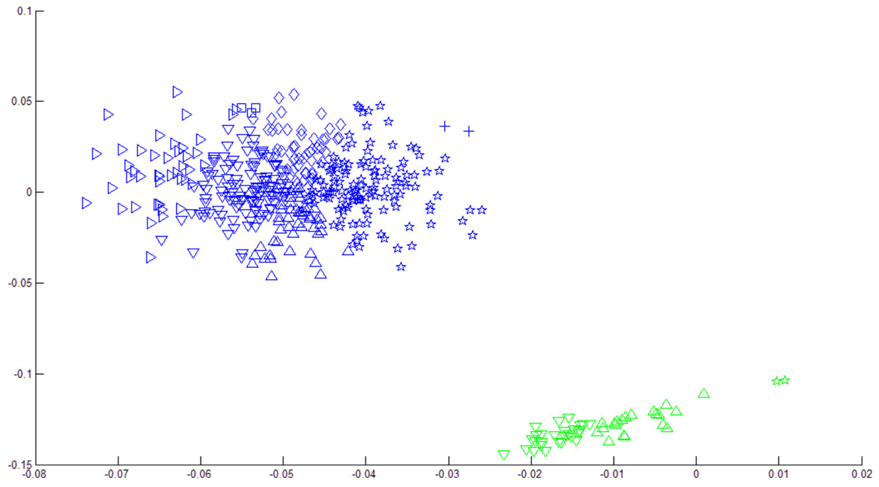

Figure 21: SVD Dataset 1: Cluster assignments in the attacker's training data

The clustering of the attacker's dataset for the first dataset is shown in Figure 21. Here the records of the large cluster have $+1$ in the first column of the X matrix (blue symbols), while the records of the smaller cluster have 0 (green symbols). Even with the clearly defined clustering visible in Figure 21, it is a simple task to hide the smaller cluster within the larger by inserting only 102 new records into the dataset. The mask contains three different kinds of records. The first is a set of 100 records, $1/5^{th}$ the size of the small cluster. As the two clusters differ only in the first eight attributes, the last 22 attributes have a mean of 1 and a standard deviation of 1 to match the rest of the records in the dataset. The first eight attributes are used to bridge the two clusters. As such, these records have a mean set to the midpoint between the two clusters (-0.5), and a standard deviation of two thirds the distance between the two classes or 2.

The next two pieces are the two single records that are placed on opposite sides of the current clusters. The record on the far side of the larger cluster has its first eight attribute values set to 9 and the record on the far side of the smaller cluster has its first eight attribute values set to -11. The remaining 22 attributes are each set to Gaussian distributed random values.

Testing this with the attacker's training set proves successful so the same attack is applied to actual dataset. This mask successfully hides the small cluster within the large cluster as seen in Figure 22. The shapes of the symbols in this (and other figures) are based on the second and subsequent columns of the X matrix. The figure shows that, although the symbols in the larger and smaller clusters are different, the symbols in the bridge overlap with both – showing that further analysis using SDD will not trivially separate the two original clusters.

The second dataset differs from the first in that the smaller cluster is twice as dense. Despite this, little changes in the attacker's dataset, shown in Figure 23.

As the clusters in this dataset are quite similar to the first dataset, the attack required to subvert the cluster analysis should be the same. Applying the same mask to the real dataset produces the clustering shown in Figure 24. The denser distribution of the smaller cluster has no significant effect on the attack.

The third dataset differs from the first in that the distance between the large and small clusters is doubled. This increase in distance allows the decomposition-based cluster analysis not only to

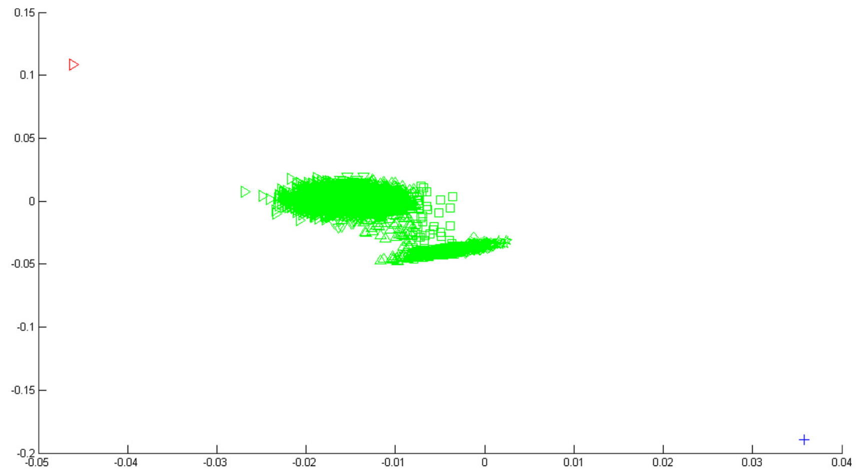

Figure 22: SVD Dataset 1: Cluster assignments of the actual dataset after the attack

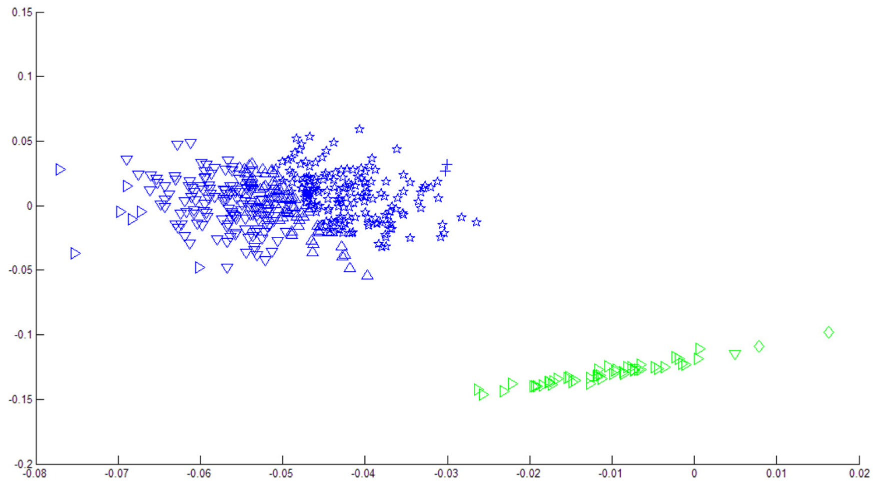

Figure 23: SVD Dataset 2: Cluster assignments in the attacker's dataset

be able to identify the two clusters as different, but indeed, opposite.

Figure 25 shows the cluster analysis for the attacker's training set. This shows the large cluster as blue, records with $+1$ in the first column of X, and the small cluster as red, records with -1 in the first column of X.

Despite the increased distance between the two clusters, it is still possible to subvert the cluster analysis performed on this dataset using the same approach. We create the bridge between the two clusters consisting of $1/5^{th}$ the records in the small cluster with mean the midpoint between the two clusters. The mean of the first eight attributes is -2.5 and of the next 22 attributes is 1. The bridge has a standard deviation for each attribute of two thirds the distance between the two clusters; 4 for the first eight attributes and 1 for the rest. The one-record clusters are again 8 units beyond the mean of each cluster in those attributes in which they differ. Thus, the first eight

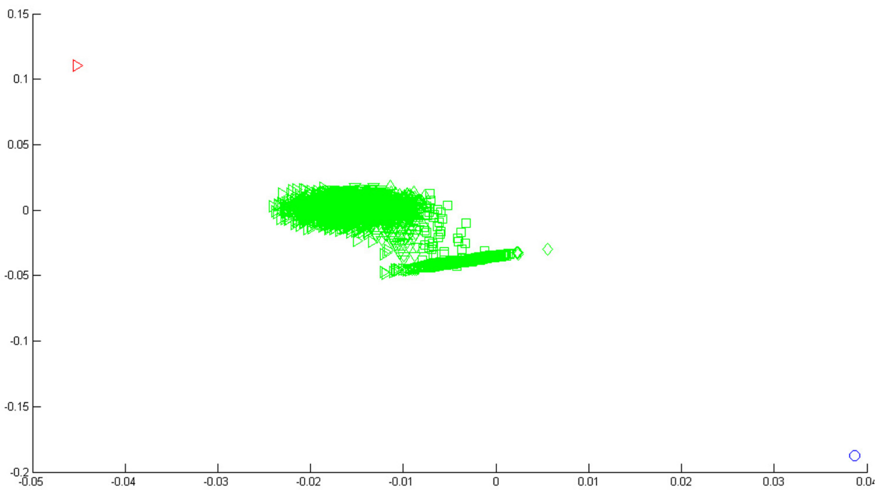

Figure 24: SVD Dataset 2: Cluster assignments of the real dataset after the attack

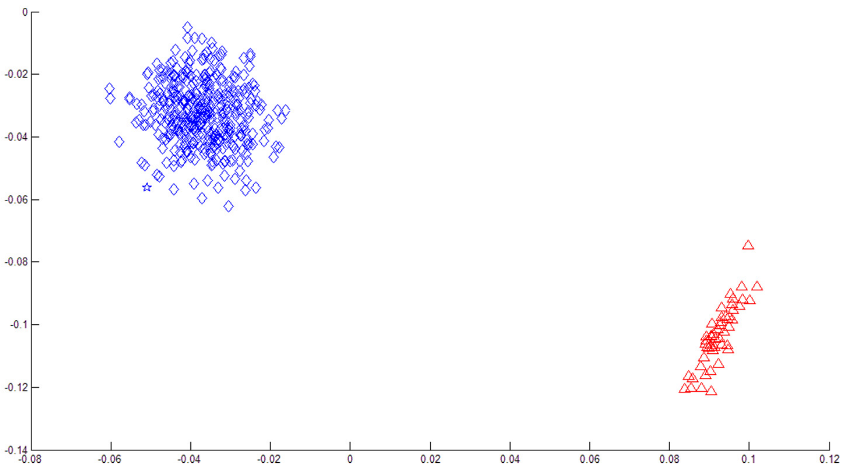

Figure 25: SVD Dataset 3: Cluster assignments of the attacker's training data

attributes of the record on the far side of the larger cluster have value 9 and those of the record on the far side of the smaller cluster have value -13. The remaining attributes are assigned Gaussian distributed random values. This mask is adequate when applied to the attacker's training data, so we apply it to the actual dataset. The result of adding these 102 records is shown in Figure 26. The position of the points may create some suspicion about the presence of more than one cluster, but it is certainly not obvious. Adding extra records to the bridge can further blur the hints of two clusters within the apparently single central cluster.

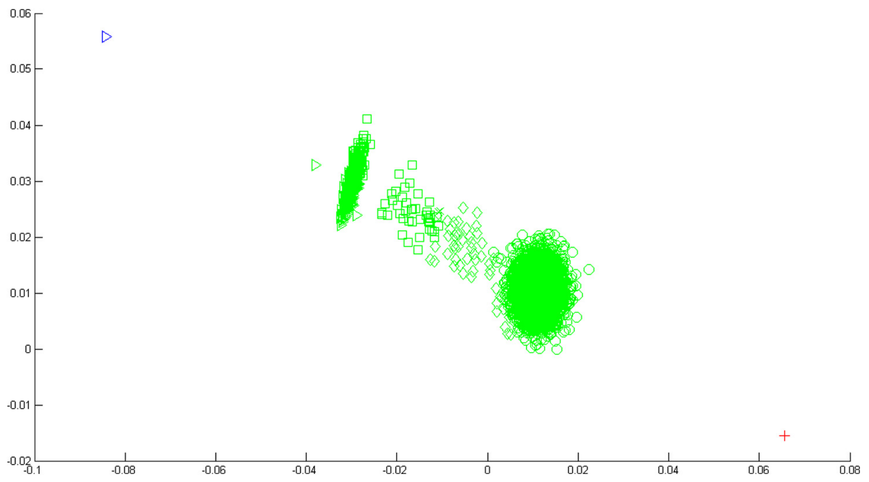

Figure 26: SVD Dataset 3: Cluster assignments of the actual dataset after the attack

3.3 Association Rules and the *a priori* Algorithm

Association rules predict the structure of itemsets within a dataset. In particular, association rules are concerned with identifying cases where the occurrence of one event suggests the occurrence of another. The rules have the form "if A and B then C", where A and B are the premise and C is the consequent. For example, in the case of a supermarket conducting market analysis, an association rule may consist of the premise "a customer purchases a banana" and the consequence "the customer is likely to purchase yogurt".

Association rules are most frequently mined from datasets using the *a priori* algorithm [1]. The *a priori* algorithm's purpose is to identify those itemsets that occur frequently in the dataset, or have high support. For an itemset to be considered an association rule it must typically have both high support and confidence. The confidence is the support for the rule over the support for the premise – essentially the probability that the premise and consequent appear together.

The datasets that are used to identify association rules typically amass new data but do not remove old data. This makes it is impossible to lower the support of a given rule. Thus, the attack against association rules focuses on the lowering the confidence of a rule. This is done by inserting rules with the same premise, but with a different consequences. For example, to eliminate the "if a customer purchases bananas then s/he will purchase yogurt" rule, one may make many transactions where s/he purchases bananas and items that are not yogurt.

The precise number of false premises that need to be inserted into a dataset to guarantee a rule is not produced may be calculated. In Equation 1, *fp* denotes the number of false premises required to be inserted while the support of the premise is the number of premises in support of a given rule.

$$fp = support(Premise) \times \frac{confidence(Rule)}{confidence(Desired)} - support(Premise) \tag{1}$$

Though it is possible to lower the confidence of a given rule so that it is no longer flagged as important, it is probably not practical to do so. Association rules are usually formed from datasets that are always changing. This attack would require the constant monitoring and insertion of a large number of false premises to hide a given rule.

4 Conclusion

The datasets used in adversarial settings are unusual because they may contain records that have been deliberately manipulated to cause particular outcomes. These manipulations may be directed at any of the three stages of knowledge discovery: data capture, analysis, and decision and action. Possible ways in which the overall process can be subverted by manipulating (and evading) data capture; and by social engineering to affect decisions and actions have previously been considered. However, little attention has been paid to the robustness of prediction and clustering technologies when data may have been manipulated.

We have shown that predictors and clustering algorithms in common use are remarkably fragile in the face of such data. Often only a small number of records need to be manipulated for the entire outcome to be subverted. While it is not trivial to insert such records into the data that will be analyzed, we have showed that deep knowledge of the algorithm parameters or the data is not required to design attacks. Moreover, the effect of attacks is largely predictable – knowledge of the blind spots created in the knowledge-discovery process can be inferred from the properties of the manipulated records.

The main implications for knowledge discovery in adversarial settings are:

- Knowledge-discovery algorithms should be designed explicitly for adversarial settings rather than adopting mainstream algorithms, as if adversarial settings were just another application domain;
- Algorithms that use ensembles are, in general, to be preferred, since it much harder to manipulate them because of the inherent randomness of the selection of records (and sometimes attributes) used to build each submodel. Hence, random forests is much more robust than support vector machines in adversarial settings.
- Attention should be paid to the way in which data is collected and used in adversarial settings, especially as models are updated to reflect changes in the real world. There is considerable motivation on the part of 'bad guys' to insert records with particular properties, and they can make such records seem particularly 'interesting' with respect to the existing model by making them seem like false positives and negatives.

References

[1] R. Agrawal and J. Shafer. Parallel mining of association rules: Design, implementation and experience. Technical Report RJ10004, IBM Research Report, February 1996.

[2] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees.* Chapman and Hall, New York, 1984.

[3] C.J.C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[4] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[5] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39:138, 1977.

[6] G.H. Golub and C.F. van Loan. *Matrix Computations.* Johns Hopkins University Press, 3rd edition, 1996.

[7] T.G. Kolda and D.P. O'Leary. Computation and uses of the semidiscrete matrix decomposition. *ACM Transactions on Information Processing*, 1999.

[8] B. Mezrich. *Bringing Down the House: How Six Students Took Vegas for Millions.* Arrow Books, 2002.

[9] D.P. O'Leary and S. Peleg. Digital image compression by outer product expansion. *IEEE Transactions on Communications*, 31:441–444, 1983.

[10] J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.

[11] J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan-Kaufmann, 1993.

[12] D.B. Skillicorn. *Understanding Complex Datasets: Data Mining with Matrix Decompositions.* CRC Press, 2007.