Semi-supervised data organization for interactive anomaly analysis.

Javed Aslam
College of Computer Science
Northeastern University
Boston, MA 02115

Sergey Bratus
Computer Science Dept.
Dartmouth College
Hanover, NH 03755

Virgil Pavlu
College of Computer Science
Northeastern University
Boston, MA 02115

Abstract

We consider the problem of interactive iterative analysis of datasets that consist of a large number of records represented as feature vectors. The record set is known to contain a number of anomalous records that the analyst desires to locate and describe in a short and comprehensive manner. The nature of the anomaly is not known in advance (in particular, it is not known, which features or feature values identify the anomalous records, and which are irrelevant to the search), and becomes clear only in the process of analysis, as the description of the target subset is gradually refined. This situation is common in computer intrusion analysis, when a forensic analyst browses the logs to locate traces of an intrusion of unknown nature and origin, and extends to other tasks and data sets.

To facilitate such "browsing for anomalies", we propose an unsupervised data organization technique for initial summarization and representation of data sets, and a semi-supervised learning technique for iterative modifications of the latter representation. Our approach is based on information content and Jensen-Shannon divergence and is related to information bottleneck methods. We have implemented it as a part of the Kerf log analysis toolkit.

1 Introduction

In the following subsection we describe the class of real-world situations that motivate our approach. It comes from a generalization of our log analysis experience with diverse data sets, and serves to justify the scenario that is targeted by our data organization method.

1.1 The data and the scenario

We address the task commonly faced by analysts presented with large amounts of data that can be more or less naturally represented in the form of a database table. Each row in the table, represents a record of an event. The columns of the table represent features of these events, and the collection of values in a row forms a description of the corresponding event. Not all columns need to be defined for each row, because the events may be of a number of different types (for features not applicable for an event, the corresponding rows contain a special *undefined* or *not applicable* value). Most of the features are discrete-valued, with no meaningful order on their values. In particular, computer activity logs are often represented in this fashion.

A certain subset of these events is of particular interest to the analyst. These events are "anomalous" (in an activity log, they can be related to a malfunction or a malicious intrusion), in the sense that the analyst will recognize some of the values they are composed of as "anomalous", *once he or she has an understanding of the overall structure of the dataset*. In practice, it is often the case that neither these values nor the columns (features) in which to look for them are known to the analyst at the outset. In computer forensics this situation arises when the nature and origin of an intrusion are unknown, but some external evidence suggests that it has occurred, and the investigator sets out to find whatever unusual traces it may have left in the regularly collected logs.

This task of looking for unspecified anomalies is hard to automate. Automatic detection systems based on statistical profiling of normal behavior and flagging deviations from the profile cannot fully solve it, for two reasons. Firstly, to interpret the alerts resulting from such detection, the analyst needs to understand the system's model of normal behavior, whereas statistically trained models are not designed for human understanding. Secondly, such systems expect to learn their models from training data that represents normal behavior, whereas it is hard to guarantee that all of the actually available historical data does not include the targeted anomaly. Indeed, Intrusion Detection systems (IDS) have been criticized for both their high number of false positives, and their propensity for being maliciously desensitized to true positives.

Thus, although automatic anomaly detection can provide

useful starting points for analysis, it does not eliminate the need for the interactive, iterative process that we call *browsing for anomalies*.

As a rule, this process involves trying different summarized representations of the dataset until one of those shows some outlying records that can be investigated further, most often to find that they are innocuous despite being statistically unusual (and need to be treated specially so that they do not trigger further false alarms). Summarized representations are also useful for judging the significance of differences between records, especially for discrete-valued features. The choice of the appropriate representation can be a challenging task, a kind of iterative trial-and-error "blind feature selection" – blind, because the nature of the sought-after anomaly is not clear at the outset, and the analyst's idea about it is modified along the way.

We propose a simple approach to data organization and summarization to make this process more efficient. It is described in the next subsection.

1.2 The proposed approach

Our approach is based on constructing a tree representation of the dataset, initially unsupervised, and adjusting it based on markup supplied by the user. The trees are similar to those resulting from the ID3 and C4.5 decision tree algorithms (e.g., [10]), but, unlike these automatic classification algorithms, we start with entirely unlabeled data, and at each step use not only the labeled examples, but the entire dataset, to exploit its internal structure.

Our method is designed to produce data representations useful for the kind of browsing described above, starting with no markup and continuing "smoothly" with very small percentage of labeled data.

This scenario is not served well by ID3 and similar algorithms. Firstly, by their very definition, they require at least some labeled examples to start with, while in practice, one needs a data representation to come up with meaningful examples, as described above, since looking for these in plain lists or tables is extremely inefficient. Secondly, classification algorithms do not work well with only a small percentage of the data labeled.

A great variety of clustering algorithms that exploit the structure of an unlabeled dataset can be applied to initially group the records, unsupervised. However, since we expect to spend some time browsing the data to label the initial subset of examples, we want the groupings to produce a limited number of groups, with further hierarchical subdivision for the larger groups. We also want the groups to have simple descriptions that would extend naturally to labeled data. In other words, we want a smooth transition from the results of clustering to decision trees. We have not found an existing clustering algorithm to fit this scenario. We have thus designed our algorithm to fill this gap.

For the purposes of estimating the efficiency of our method, we compare it with ID3 and a simplified version of C4.5 with respect to two parameters of the resulting trees: their *description length* and the *classification error* of using these trees for multi-label classification. Due to the lack of a publicly available and representative annotated dataset of computer logs, and in order to demonstrate the general usefulness of our approach, the comparison is made over a subset of the UC-Irvine datasets. We also compare the classification error of our algorithm running completely unsupervised with that of a popular clustering algorithm.

1.3 Related work

For semi-supervised classification, [8] uses a Bayesian framework where all possible data models (trees) are considered. The classification rule is given as a sum over all models, approximated with Monte Carlo methods; learning with each tree is achieved via a mutation process designed to favor labellings that are smooth with respect to the tree. For comparison, our method builds a deterministic, greedy, tree by splitting each node for the data feature that minimizes a local objective (Jensen-Shannon divergence).

The Scatter/Gather cluster-browse approach [6] is a well known information retrieval system. In an iterative manner, the algorithm presented clusters by a given criteria, then combines back relevant (to user query) clusters, and then changes the clustering criteria for the new (smaller) dataset. Significant differences in our work are apriori determined splitting possibilities (data features), non-"gathering", and data-mining orientation of tree-splitting criteria.

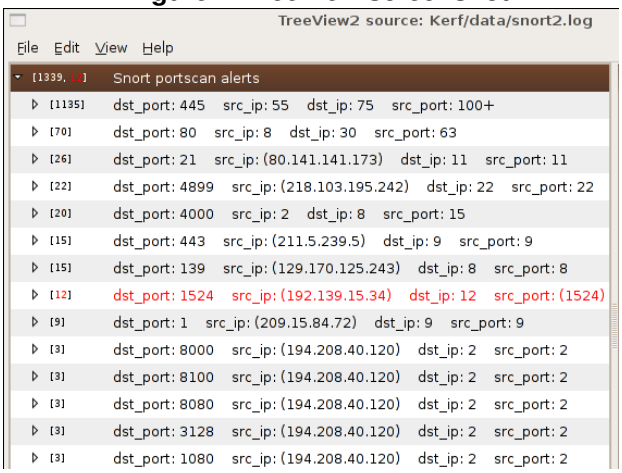
2 Making trees out of tables

In this section we detail our approach to presenting datasets to the user. While graphical visualization techniques can be very efficient in highlighting specific kinds of anomalies, they are usually designed to fit the data source and need to be carefully tuned. Thus in practice users have to fall back on familiar column-sortable table views of their data. Our approach is simple, generic and generalizes the latter, while making "browsing for anomalies" a lot more efficient.

The tool in which our approach is implemented presents the records in a tree-like view, with one or more layers of grouping. The records are leaves of the tree, and each grouping corresponds to an intermediary node of an ID3-like decision tree. The labels of intermediary nodes serve as simple summarizations of the set of leaves beneath them. Most often the grouping happens by distinct values of some feature, so that the label shows that feature value (the same across the group) and a summary of other feature values for

selected features, such as the observed range for numerical values, the actual values if they are only a few, or simply the number of distinct values across the group. The entropies of induced value distributions are also an option. Figure 1 shows a screenshot of a loaded dataset.

Figure 1. TreeView screenshot



The choice and the order of groupings are chosen by the algorithms described below so as to reduce the amount of screen real estate taken by each view and to minimize the amount of information that the user needs to peruse to make the next decision – whether to mark some nodes as relevant ("interesting") or irrelevant (these marks can be later used for semi-supervised learning stage to re-arrange the tree), expand some node and consider the subset of leaves (also possibly grouped) under it, or to discard the grouping entirely and try a different one. In what follows we explain how this representation applied to server logs improves on the traditional heuristic browsing techniques as described, e.g., in [5, 4, 2, 1, 3].

3 Summarization and data organization models

Experience shows that analysts begin with simple mental models of a dataset, to understand its overall composition. Some features are assumed to be irrelevant and the remaining differ in their relevance to the search (the choice and ranking of these features is likely to change, but recall that we want to deal with the situations where the choice of the features revealing the anomaly is not initially clear).

We formalize this approach by considering a series of simple generative models of the dataset, in the order of their increasing "complexity" (or "information content", to be defined later), and for each model concentrating our attention on the records to which this model assigns its lowest probabilities.

More formally, we start with arbitrary assumptions of what constitutes a "relevant part" of a record, a lossy mapping from our dataset to some set of tuples $\mathcal{R} = \{r\}$ and a class of models $\mathcal{M} = \{M\}$ so that we can associate a probability of $P_M(r)$ with each $r \in \mathcal{R}$. Then we associate with each model M a (hopefully intuitive) measure of its "complexity" $\mu(M)$, choose $M^{(1)} = argmin_{M \in \mathcal{M}} \mu(M)$ and concentrate our attention on the records r with lower $P_{M^{(1)}}(r)$ as "anomalous". If these prove non-interesting, we choose the next $M^{(1)}$ according to μ and so on, or even switch to a different class of models. In what follows, we will define different classes $\mathcal{M}$ and their measures μ.

3.1 Simplest one–feature models

In the simplest case, the values of one feature are enough to describe the dataset and to locate the anomalies. Thus the simplest summarization is the specification of that feature's distribution (e.g., its histogram). Indeed, in our tool the dataset's records are organized in a tree, whose first level nodes correspond to the distinct values of the feature chosen at root, sorted by the number of records in each group and expandable to reveal further levels of grouping (see Figure 1).

More formally, consider the set of features F_i. Denote the set of distinct values of the feature F_i on $\mathcal{R}$ by $\mathcal{X}_i = \{x_1^i, x_2^i, \ldots, x_{|F_i|}^i\}$, and write $|F_i| = |\mathcal{X}_i|$. Let $p_k^i = P(F_i = x_k^i), i = 1, \ldots, |F_i|$ be the probabilities with which F_i takes these values on $\mathcal{R}$. The records $r \in \mathcal{R}$ can then be written as tuples $r = (F_1(r), \ldots, F_m(r))$.

3.1.1 Entropy as a criterion for feature ranking and tree construction

In this subsection we explain our use of entropy in constructing our tree representation of the dataset.

Under our previously described assumptions, the user in search of anomalies will consider these models in some order to see if they reveal any anomalies. It would be natural to start from the simplest non-trivial models (features that have one value across all of $\mathcal{R}$ are no help in locating anomalies). *We propose to use the entropy* $H(F_i) = -\sum_{k=1}^{|F_i|} p_k^i \log_2 p_k^i$ *as a measure of the information content of this model* M_{F_i}.

Thus, we would start with the feature

$$F^{(1)} = \underset{\{F_i | H(F_i) \neq 0\}}{argmin} \; H(F_i).$$

It should be noted that the perplexity $2^{H(F_i)}$ of the distribution F_i can be interpreted as the number of the *effective* values of the feature F_i, i.e. the number of distinct values that carry the majority of weight in its distribution. Under

our proposed measure, this weight is concentrated in fewer values, and vice versa.

The chosen feature $F^{(1)}$ will determine the branching factor at the root of the tree in which the records are displayed (there will be $|F^{(1)}|$ branches). We call the perplexity of $F^{(1)}$ the *effective branching factor*, because it gives the number of branches that hold most of the weight of that feature's distribution. We choose the tree with the smallest effective branching factor at the root.

Intuitively, statistically infrequent values are more likely to be "anomalous". We postulate that the simpler the distribution, the more likely it is that its infrequent values will be interesting. Consequently, we would present the user with the simplest distribution first, raising the chance of spotting such an anomaly.

There is also another argument for picking the lowest entropy feature among others for splitting the records. Namely, the child node distributions resulting from the split are more likely to retain, on average, the mutual information with the anomalous set. Arguably, this split gives us a better tree for browsing down to the anomalous nodes and leaves than other non-trivial splits.

More formally, consider the record set $\mathcal{R}$ with an unknown subset of "anomalous" records, identified by an indicator variable A. We assume that $I(\mathcal{R}; A) > 0$, otherwise we have no real hope of recognizing the anomalies in this representation $\mathcal{R}$.

Splitting on the $|F_i|$ distinct values $\{x_k | k = 1, \ldots, |F_i|\}$ of a feature F_i will produce $|F_i|$ distributions $\mathcal{R}_k = \mathcal{R}_{x_k}$.

We want to choose F_i so that this splitting is most likely to preserve, on average, as much mutual information with A. Averaging the mutual information over these $\mathcal{R}_{x_k}$, we get:

$$\begin{aligned}\sum_{k=1}^{|F_i|} p_k^i I(\mathcal{R}_k; A) &= I(\mathcal{R}; A) - I(A; F_i) = \\ &= I(\mathcal{R}; A) - H(A) + H(A|F_i)\end{aligned}$$

From this we can see that the smaller the entropy of F_i, the more likely it is to increase $H(A|F_i)$ and together with it the entire average mutual information (from the Venn diagram visualization of entropy, the smaller the disk representing $H(F_i)$, the smaller part of the fixed area representing $H(A)$ it is likely to overlap; the remaining area of $H(A)$ corresponds to $H(A|F_i)$). This justifies the choice of lower entropy features for splitting the record set.

3.1.2 Jensen–Shannon divergence as a measure of dissimilarity between subtrees

In this subsection we introduce Jensen–Shannon divergence into the tree construction algorithm, and motivate its use.

Mere small information content of a single feature may not qualify that feature as a good separator of records. Should this simple model fail to reveal any interesting anomaly, we would need to either discard it, or look further inside one of the groups of records that the splitting of $\mathcal{R}$ produces (in our tool this corresponds to expanding a node of the tree, collapsing its siblings, and further actions on the subtree). Consequently, we would like to modify our criteria to increase the dissimilarity between the subtrees resulting from our choice of the splitting feature.

The Jensen–Shannon divergence is a measure of dissimilarity between distributions. Following the information bottleneck approach [7, 12], we modify our criterion for choosing the one-feature model to minimize its information content, while maximizing the divergence of the resulting distributions. More precisely,

$$F^{(1)} = \underset{\{F_i | H(F_i) \neq 0\}}{argmin} \; H(F_i) - \beta \cdot JS_{\pi_i}(\mathcal{R}_1, \ldots, \mathcal{R}_{|F_i|}),$$

where JS_{π_i} is the weighted Jensen–Shannon divergence between the joint distributions of the remaining features, induced by partitioning the set $\mathcal{R}$ according to the unique values of the feature F_i and weighted by the probabilities of F_i's distribution. More precisely,

$$JS_{\pi_i}(\mathcal{R}_1, \ldots, \mathcal{R}_{|F_i|}) = H\big(\sum_{k=1}^{|F_i|} p_k^i \mathcal{R}_k\big) - \sum_{k=1}^{|F_i|} p_k^i H(\mathcal{R}_k),$$

and the coefficient β (a Lagrange multiplier) expresses the trade-off between the two goals, and $\mathcal{R}_k$ is the joint distribution of the tuples all features remaining after the choice of F_i.

At this point we must make two important compromises. Recall that our intention is to help an *interactive* process, therefore lengthy computations are undesirable. Thus we allow two shortcuts. Firstly, when it is computationally unrealistic to compute the full $\mathcal{R}_k$, we exclude one or more features with the highest entropy from the $argmin$ operation. Secondly, β is left to be a user–controlled parameter, the initial value of which was chosen to make the two parts of the above difference commensurate, and was mostly left at 1.0.

Our choice of the Jensen–Shannon divergence, and our weighting of the joint distributions $\mathcal{R}_i$ in particular, is based on the following coding-type consideration. Its first term represents the average number of bits needed to encode the averaged distribution of all $\mathcal{R}_k$, and the summands of the second term represents the average numbers of bits needed to encode the sets $\mathcal{R}_k$, which are drawn from with the probabilities p_k^i. The difference thus represents the information reduction from encoding the sets $\mathcal{R}_k$ separately rather than all together. The Jensen–Shannon divergence grows with the difference between the $\mathcal{R}_k$ and would be 0 if these were identical.

Notice that the first term of our measure represents the average number of bits necessary to encode the distribution of the first feature, thus our trade-off is between this number and the average coding length win from the partitioning, commensurable quantities.

Thus we would pass over a feature with a simpler distribution in favor of one with a somewhat more complex one, which would give a partition into subtrees more likely to be useful for browsing. In particular, first term favors the choice of the most compact one-feature summarization, and the Jensen–Shannon term is intended to serve as a tie-breaker in case there are two features such that their entropies are low and close.

Another way to interpret this measure is suggested by the information bottleneck method ([7, 12]). Information bottleneck can be interpreted as finding a trade-off between a compression (in our case, summarization) level of channel information while preserving as much relevant information as possible. In our case the role of a constrained channel is played by the user attention and the limited amount of screen space (data displays that necessitate scrolling through dozens of pages are virtually useless for our scenario). In the unsupervised scenario, the role of relevance is played by the divergence of the resulting partitioning of data due to the summarization.

3.2 Multi-feature models

In the previous section we discussed the simplest way of summarizing a dataset, based on the values of a single feature. Essentially, our method is to look at the feature with the simplest-looking histogram first and then, possibly, consider other such single feature summaries in the order of increasing information content. In this section, we extend these representations.

Instead of using a single feature F_i for some i, we can consider combined features $F_{ij} = (F_i, F_j)$ and choose the pair with the simplest histogram in the same fashion,

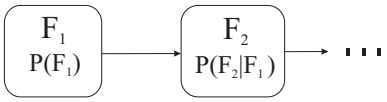
$$F^{(12)} = \underset{\{F_{ij}|H(F_i)\neq 0, H(F_j)\neq 0\}}{argmin} H(F_{ij}) - \beta \cdot JS_{\pi_{ij}}(\mathcal{R}'_1, \ldots, \mathcal{R}'_{|F_{ij}|}),$$

where the $|F_{ij}|$ distinct values of the pair of features F_{ij} induce a different partition $\{\mathcal{R}'_k\}, k = 1 \ldots |F_{ij}|$.

Combined with the above single-feature method, this method would give us the pair of features whose joint distribution's histogram is one of the simplest, and whose distinct values partition the rest of the distribution into joint distributions that are dissimilar. Such a representation will highlight anomalies in this joint distribution, i.e. any anomalies that are described in terms of one feature's correlation with another.

This search is quadratic in the number of features, which may be computationally prohibitive for an interactive iterative process. Thus we propose a greedy way of modeling the set $\mathcal{R}$ based on two features (and considering others as irrelevant for the purposes of the summary).

Consider $\mathcal{R}$ as generated by drawing values from the distribution of F_1 and then, based on the drawn value of F_1, by drawing values from some F_2 based on the distribution $P(F_2|F_1)$. We can extend this generative model further in the obvious way, using the Markov assumption (drawing from some $P(F_3|F_2)$ and so on).



This class of generative models gives more expressive summaries. Again, we postulate that it is natural to start looking at the simplest summaries, as these are more likely to reveal anomalies. These now would include values of F_1 for which the character of mutual dependence between F_1 and F_2 is atypical (e.g., most values of F_1 are good predictors for the value of F_2 in the pair, but some are not).

We can consider such a sequence of features $F_1, F_2, \ldots, F_l$ in some order, up to some l beyond which the remaining features are deemed completely irrelevant to our search for anomalies or our idea about the data. We can expect the features whose distributions are closer to the uniform to be further away in this sequence, and vice versa, according to its likelihood to distinguish between records of interest and others. In other words, we can assume that the closer a feature to being uniform, the less likely it is to be relevant to our search.

As before, we propose an information measure for complexity of these models, namely

$$H(F_1, F_2) = H(F_1) + H(F_2|F_1)$$

where $H(F_1|F_2)$ is the conditional entropy of the distribution of F_2 over $\mathcal{R}$ given F_1, $H(F_2|F_1) = -\sum_{(f_1,f_2)} p(f_1, f_2) \log p(f_2|f_1)$. From the coding perspective, this is the necessary average bit length for encoding this model.

A greedy approximation to avoid the quadratic nature of this measure,

$$F_{(1)} = \underset{\{F_i|H(F_i)\neq 0\}}{argmin} H(F_i) - \beta \cdot JS_{\pi_i}(\mathcal{R}_1, \ldots, \mathcal{R}_{|F_i|}),$$
$$F_{(2)} = \underset{\{F_i|H(F_i)\neq 0\}}{argmin} H(F_i|F_{(1)})$$

appeared to produce acceptable results, although we did not study the performance of this method systematically. We leave the study of this matter for future work.

4 Semi-supervised extension

In this section, we describe the transition from unsupervised to semi-supervised data organization, based on user labeling of nodes performed while browsing the tree. We use labels as a side information channel, and modify our unsupervised algorithms to take it into account and re-adjust the tree, to bring possibly overlooked dataset structure and "anomalies" to the user's attention. This approach is motivated by conditional bottleneck clustering [7].

We still want a concise summary of the data provided, so we keep the requirement for the information content of the "splitting" feature to be minimized. However, in the presence of user labels identifying the partition of records into relevant and non-relevant, we want to take relevance with respect to this labeling into account. Let the variable L be the new feature with discrete values in the set of labels. We propose to choose the summary feature for the one-feature summary as

$$F^{(1)} = \underset{\{F_i|H(F_i)\neq 0\}}{argmin} H(F_i|L)$$

where the conditional entropy of a feature X given Y is defined as $H(X|Y) = H(X,Y) - H(Y)$. Notice that for any feature F

$$\begin{aligned} H(F|L) &= H(F,L) - H(L) = \\ &= H(F) + H(L) - I(F;L) - H(L) = \\ &= H(F) - I(F;L) \end{aligned}$$

and so by requiring to minimize the conditional entropy $H(F|L)$ we are actually favoring features with higher mutual information with our labeling variable L.

Also, we still want the joint distributions of other features to be divergent, so that the choice of the summarizing feature (or pair of features) separates the rest of the data into dissimilar groups. In order to express this trade-off, we restore to the above measure the weighted Jensen–Shannon divergence, also conditioned on the labeling L:

$$F^{(1)} = \underset{\{F_i|H(F_i)\neq 0\}}{argmin} H(F_i|L) - \beta \cdot JS_{\pi_i}(\mathcal{R}_1, \ldots, \mathcal{R}_{|F_i|}|L),$$

where the conditional form of JS_{π_i} is obtained by replacing the respective entropies $H(F)$ with $H(F|L)$. In accordance with the above observation about mutual information, the resulting win in average encoding length will be adjusted by the loss in mutual information with L:

$$\begin{aligned} JS_{\pi_i}(\mathcal{R}_1, \ldots, \mathcal{R}_{|F_i|}|L) &= JS_{\pi_i}(\mathcal{R}_1, \ldots, \mathcal{R}_{|F_i|}) \\ &- \left(I(\overline{\mathcal{R}}; L) - \sum_{k=1}^{|F_i|} p_k^i I(\mathcal{R}_k; L)\right) \end{aligned}$$

The last term adjusts the divergence measure by its relevance to the labeling L.

5 Results

In this section we explain our choice of metrics for comparing our algorithm with ID3 and describe the results of this comparison.

We compare our algorithm to ID3 and C45 on 8 sets from the UC-Irvine collection (*bal, band, car, cmc, crx, monk, tic, vote*) using the following method. For each of these sets, we fix a percentage of labeled records (starting at 1% through 30%), randomly select the respective number of records from the set and use these records and their true labels as labeled input to our algorithm (in the subsequent graphs and tables referred to as JSV, for "Jensen–Shannon Visualization") and to ID3. We then compare the classification error and the minimal description lengths of the labeling using the trees output by JSV and ID3.

We chose the minimal description length as the measure most closely related to the browsing scenario described in section 1.1. In particular, we assume that the user will keep expanding nodes in the branches of the tree until reasonable separation of categories (e.g., different kinds of "normal" and "abnormal") is achieved. If a group has a few clearly defined exceptions, the user is likely to accept them as such and accept the majority category for that group. The user will prefer more compact trees even if they have a certain number of exceptions.

Accordingly, we use the minimal description length (MDL) of a dataset's labelling derived from the tree as a quality measure of the dataset's presentation provided by that tree.

The MDL measure is computed as follows. We fix an encoding scheme for the labeling of the dataset based on the pruned tree, in the spirit of the seminal paper [11], by encoding the pruned tree and the exceptions, and adding these two lengths. The total corresponds to the length of the message necessary to transmit the full labelling column of the dataset between the two parties who both possess all the other columns of the records (the receiver reproduces the labelling using the decision tree and the exceptions list). We prune the decision trees produced by ID3 and JSV to achieve the shortest encoding under this scheme, to express the user's preference for the simplest tree.

Examples of such minimal tress can be seen in Figures 4 and 5. In these figures, we first plot the tree in thin lines as constructed by the method (JSV or ID3) and then, on top of it, the minimum MDL subtree (in bold). Leafs are represented with squares. The bar on top of each node represents the number of labels for each class, at that node. The number inside each non-terminal node is the feature-id used to split the data at the node.

Figures 2 and 3 show the dependence of the MDL measure and the classification error on the fraction of the labeled records for the dataset *crx* (on which JSV had the best

performance over ID3), and Tables 1 and 2 describe the performance for other sets.

Figure 2. MDL measure

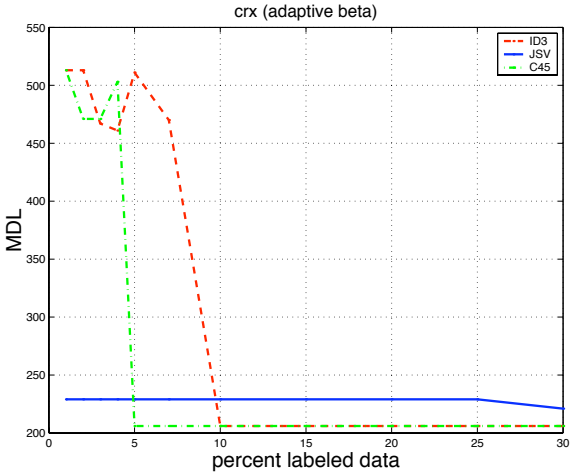
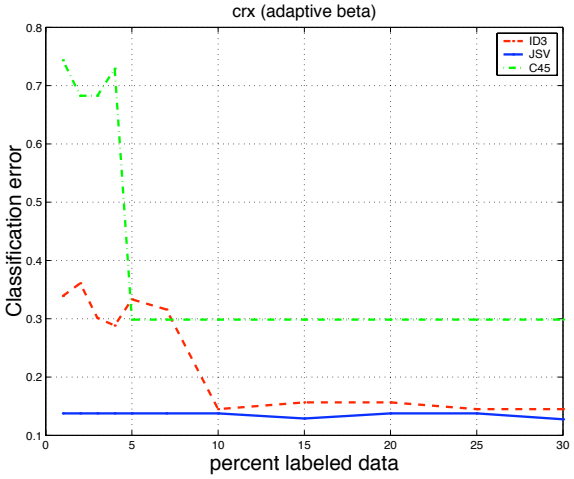


Figure 3. Classification error



The results of these comparisons can be summarized as follows. At a small fraction of labeled records, the behavior of ID3 is quite erratic, whereas JSV performs more predictably. In the range of 1%–5%, JSV generally performs much better than ID3. After 5% ID3 begins to catch up, and at 30% almost uniformly outperforms it (it must be noted that marking up a third of the data during analysis in one go is hardly ever possible, which is why we did not continue the comparison beyond the 30% mark).

5.1 Comparison with k-Means clustering

As a semi-supervised techniques, the performance of JSV is compared (section 4) with that of classical clustering algorithms, ID3 and C4.5. Next, we compare the classification error of unsupervised JSV (label class determined by majority) with that of *batch k-Means*, as described in [9]. Initial centroids are picked to be random K data points with different labels, one per label class. The results can be found in Table 3; we observe large, consistent outperformance by JSV in terms of accuracy.

Table 1. Performance with 5% of labeling

	MDL(ID3)	MDL(JSV)	Err(ID3)	Err(JVS)
BAL	436	422	0.30	0.28
BAND	359	343	0.29	0.39
CAR	707	699	0.18	0.17
CMC	1612	1648	0.68	0.52
CRX	381	221	0.22	0.14
MONK	284	284	0.32	0.32
TIC	663	595	0.33	0.25
VOTE	119	47	0.12	0.04

Table 2. Performance with 10% of labeling

	MDL(ID3)	MDL(JSV)	Err(ID3)	Err(JVS)
BAL	431	422	0.30	0.29
BAND	346	348	0.28	0.35
CAR	630	703	0.13	0.18
CMC	1596	1638	0.60	0.53
CRX	206	221	0.14	0.14
MONK	284	284	0.32	0.32
TIC	521	589	0.29	0.24
VOTE	47	47	0.04	0.04

Table 3. Unsupervised: Comparison with k-Means clustering classification.

	Err(k-Means)	Err(JVS)
BAL	0.59	0.31
BAND	0.45	0.34
CAR	0.40	0.18
CMC	0.61	0.53
CRX	0.20	0.14
MONK	0.45	0.32
TIC	0.37	0.24
VOTE	0.13	0.04

6 Conclusion

Our data organization method was designed to fit the common data analysis scenario that can be described as "browsing for anomalies". In order to keep the results understandable to the human user, we limit the initial hier-

Figure 4. ID3 MDL Tree, 5% data labeled

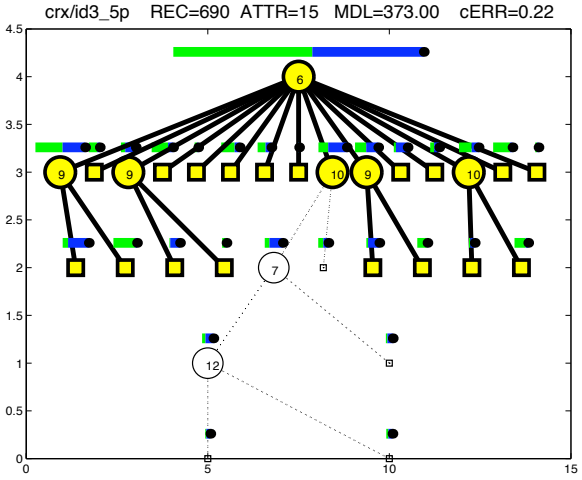
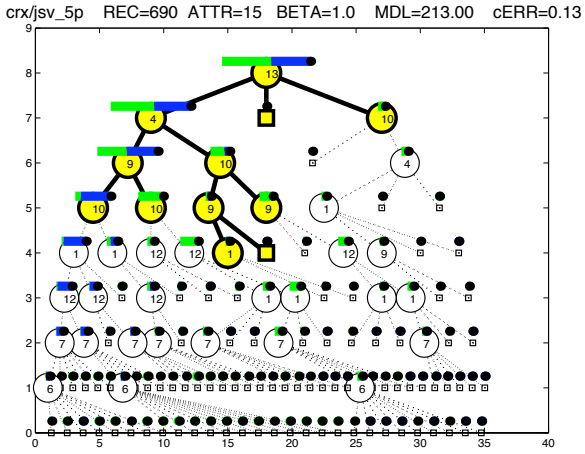


Figure 5. JSV MDL Tree, 5% data labeled



archical data groupings to those based on decision trees. It provides a smooth transition from this unsupervised process to semi-supervised data organization that takes advantage of the user markup on small amounts of data.

In the target scenario, where the percentage of labeled records is in the low single digits, our method is more stable than the classical decision tree methods based on information gain and information gain ratio (ID3 and a simplified variant of C4.5), and produces empirically useful representations. For larger fractions of labelled input, it tends to perform somewhat worse than these methods, but its performance remains generally comparable.

Informally, our experience suggests that this method allows the user to significantly speed up the computer log analysis tasks.

Acknowledgements

The Kerf project is funded under award number 2000-DT-CX-K001 from the Office for Domestic Preparedness, US Department of Homeland Security. Points of view in this document are those of the authors and don't necessarily represent the official position of the US Department of Homeland Security.

References

[1] J. Allison. Automated log processing. *;login:*, 27(6):17–20, 2002.

[2] S. Barish. Windows forensics: A case study. http://www.securityfocus.com/infocus/1672, Mar. 2003.

[3] T. Bird and M. Ranum. Log analysis resources. www..loganalysis.org, 2004.

[4] M. Burnett. Forensic log parsing with microsoft's logparser. http://www.securityfocus.com/infocus/1712, July 2003.

[5] A. Chuvakin. Advanced log processing. http://online.securityfocus.com/infocus/1613, Aug. 2002.

[6] D. R. Cutting, J. O. Pedersen, D. Karger, and J. W. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. In *Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 318–329, 1992.

[7] D. Gondek and T. Hofmann. Conditional information bottleneck clustering. In *3rd IEEE International Conference on Data Mining, Workshop on Clustering Large Data Sets*, 2003.

[8] C. Kemp, T. Griffiths, S. Stromsten, and J. Tenenbaum. Semi-supervised learning with trees, 2003.

[9] J. Kogan, C. Nicholas, and M. Teboulle. *Grouping Multidimensional Data*. Springer, 2006.

[10] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[11] J. R. Quinlan and R. L. Rivest. Inferring decision trees using the minimum description length principle. *Information and Computation*, 80:227–248, 1989.

[12] N. Tishby, F. Pereira, and W. Bialek. The information bottleneck method. In *Proceedings of the 37-th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.