Data Mining with Algorithmic Transparency

Yan Zhou(✉), Yasmeen Alufaisan, and Murat Kantarcioglu

Erik Jonnson School of Engineering and Computer Science,
University of Texas at Dallas, Richardson, TX 75080, USA
{yan.zhou2,yxa130630,muratk}@utdallas.edu

Abstract. In this paper, we investigate whether decision trees can be used to interpret a black-box classifier without knowing the learning algorithm and the training data. Decision trees are known for their transparency and high expressivity. However, they are also notorious for their instability and tendency to grow excessively large. We present a classifier reverse engineering model that outputs a decision tree to interpret the black-box classifier. There are two major challenges. One is to build such a decision tree with controlled stability and size, and the other is that probing the black-box classifier is limited for security and economic reasons. Our model addresses the two issues by simultaneously minimizing sampling cost and classifier complexity. We present our empirical results on four real datasets, and demonstrate that our reverse engineering learning model can effectively approximate and simplify the black box classifier.

1 Introduction

The past decade has witnessed a rapid growth in the use of data mining techniques for better decision making. Statistical implications derived from data can help us understand and critically assess risks and uncertainties. However, the ubiquity of data and data mining techniques has also sparked new concerns on transparency, as has been emphasized in the recent report by PCAST (President's Council of Advisors on Science and Technology) [16]. Many proprietary intelligent software applications provide users with interfaces to the "smart algorithms" in their data analytics systems. The inner workings of these smart algorithms are often incomprehensible and opaque to ordinary users. Therefore, the information released to the end user is usually overly simplified, abstract, and untestable, which in return raises the problem of transparency and trustability. There are practical benefits of withholding the inner structure of knowledge the algorithms have learned, for example, protecting companies' information assets. However, data mining models can be discriminatory, making biased decisions on the basis of race, class, gender, etc. Recent work [22] has shown that some online ads are selected by intelligent advertising systems based on the racial background of the names used in search queries. This type of bias may be deeply and unconsciously hidden within data mining models. Increasing the transparency of these

D. Phung et al. (Eds.): PAKDD 2018, LNAI 10937, pp. 130–142, 2018.
https://doi.org/10.1007/978-3-319-93034-3_11

models can help users spot these caveats that would otherwise be hidden, and is important for ensuring trust and reducing potential abuses and biases.

Clearly, many issues need to be addressed to acquire truly transparent data mining models, for example, understanding the impact of the structure of data on a black-box classifier [11], and identifying subspaces where a black-box classifier does (not) work [6]. In this paper, we focus on learning a simple decision-tree equivalent of a given black box classifier with a small number of query samples. An immediate challenge is that decision trees are well known for their poor stabilities especially when the number of training samples is small [7]. We can build a decision tree or extract a set of decision rules from the kernel-based classifier with existing rule extraction algorithms [20]. However, rule extraction algorithms add additional complexity to existing kernel-based methods [5,9], and the output of rule extraction algorithms may still be incomprehensible [5].

In this paper, we present a black-box classifier reverse engineering approach as illustrated in Fig. 1. Our technique builds a kernel-based classifier and a decision tree classifier simultaneously. The kernel-based classifier is responsible for sampling under the maximum uncertainty constraint, and the decision tree classifier assists to curtail unnecessary growth in its own complexity. Our method provides: (a) a reverse engineering procedure with good stability and close similarity; (b) a cost and complexity-aware sampling technique; and (c) a human-comprehensible output. We choose to use a kernel-based classifier for good stability and develop a new sampling technique to efficiently build a human-comprehensible decision tree counterpart of the black box classifier. Unlike existing kernel-based rule extraction algorithms, we do not operate on the kernel methods per se but instead focus on searching for data samples that naturally result in a simple decision tree equivalent of the black box classifier.

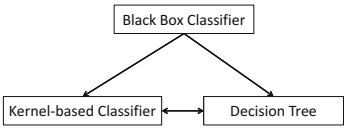


Fig. 1. Our reverse engineering method.

2 Related Work

Klivans et al. [13] study the learnability of convex bodies under the Gaussian distribution. They present a sub-exponential time algorithm for learning general convex bodies in the noise-free PAC-learning setting. Similarly, Rademacher and Goyal [17] consider learning a convex body in $\mathbb{R}^d$ given uniformly random samples from the convex body. The objective is to approximately learn the body with the fewest number of samples. They also show that it requires an exponential number of queries to learn the convex body. Also, Dyer et al. [8] present a random polynomial time algorithm for approximating the volume of a convex body in Euclidean space. Their algorithm requires a membership oracle and samples are selected nearly uniformly from within the convex body using a random walk.

Craven and Shavlik present an algorithm TREPAN to extract decision trees from artificial neural networks [3]. They modify the way a decision tree is built to limit the number of internal nodes. Henelius et al. [11] present a randomization approach to measuring the impact of groups of variables on a classification

model. Duivesteijn and Thaele [6] present the SCaPE model class that highlights subspaces where the classifier performs particularly well or poorly. Both [6,11] require a sufficient number of queries to the black-box classifier for their models to work properly. Most recently, Ribeiro et al. [18] present a sparse linear model (LIME) for local exploration—providing interpretable representation locally faithful to the classifier. However, the global effectiveness of their model is questionable when the black-box classifier is highly non-linear. Datta et al. [4] present Quantitative Input Influence (QII) to measure the most influential inputs on the output of a classification model. QII also provides local transparency for a single instance or groups of instances. Unlike existing research discussed above, our technique reveals the global knowledge of the entire data domain by leveraging the inherent transparency and interpretability of decision trees. Three important aspects set our technique apart from the existing ones: (1) we assume querying the black-box classifier is limited; (2) our model can use any fast implementation of existing and future kernel methods and decision trees; and (3) our decision tree interpreter is global.

3 Problem Definition

We define the classifier reverse engineering learning problem as follows: given a black box classifier C and one random sample $x^c \in \mathbb{R}^d$ from each class $c = \{c_1, c_2, \ldots c_K\}$, we would like to reverse engineer C with a finite set of samples S from a distribution D and transform C to a user understandable classification model C':

$$\begin{aligned} \underset{C',S}{\arg\min} \quad & \ell(C', S) \\ s.t. \quad & \mid S \mid < \delta_S \\ & \Pr_{x \sim D}[C(x) \neq C'(x, S)] < \delta \end{aligned} \tag{1}$$

where ℓ is a function that measures classifier complexity, and δ and δ_S are predefined constants. The problem has to be solved heuristically because of its exponential complexity [8,13,17]. At first glance, we can solve the problem using traditional active learning techniques [2,14,19,23]. However, existing active learning techniques cannot accomplish the task when it has to overcome both *cost* and *complexity* in the desired solution. This merits further research investigations on transparency inspired issues in terms of *cost* and *complexity*. We refer to *cost* as the total number of queries sent to the black box classifier and *complexity* as the size, that is, the number of leaf nodes in the decision tree.

4 The Reverse Engineering Approximate Learning (REAL) Model

For increasing human-understandability, we choose to use a decision tree to represent the approximation of the black box classifier. As mentioned earlier,

decision trees are well known for their poor stabilities. Since the reverse engineering learning process begins with a small training set, the poor stability of the decision tree classifier may significantly impact the end result. Existing results show that when the underlying classifier is a decision tree, query-by-bagging [1] is more stable and more accurate compared to its competitors [7]. In our study, we implement a benchmark strategy referred to as *direct hypothesis formation* in which we adopt the `query-by-bagging` method in the query-by-committee strategy for data sampling.

To circumvent the instability problem of decision trees, we introduce an intermediate kernel-based learner that is more stable than a decision tree learner. The kernel-based learner is used in data sampling where query points are selected. After selecting data samples, we build a decision tree approximation of the black box classifier. We refer to this strategy as *indirect hypothesis formation*.

4.1 Minimum Cost and Complexity Sampling

To reverse engineer a black box classifier and transform it into a tree-structured classifier, we seek a set of training samples that is sufficient to construct a decision-tree counterpart of the black box classifier under the cost and complexity constraints. To minimize the sampling cost, we follow the principle of maximum uncertainty and select samples that maximally prune the version space. In the mean time, to limit the growth of the complexity of the decision tree classifier, we select samples that have a higher probability to be assigned to a leaf node with a large population of samples given the topology of the current decision tree. We provide a list of notations used throughout this section in Table 1.

Table 1. List of notations

$\mathcal{H}$	classifier built on existing training data
$\phi_\alpha(x)$	probability x is assigned to a leaf node α
$\phi^*_\alpha(x)$	maximum $\phi_\alpha(x)$ over all leaf nodes
μ_α	prototype of labeled samples in leaf node α
μ	set of prototypes of samples in all leaf nodes α
β	Lagrange multiplier in Gibbs distribution
ψ_u	function that measures uncertainty
c_i	the i^{th} class label where $i = 1, \ldots, K$

Our sampling objective function is:

$$\begin{aligned} &\arg\max_{x \sim D} \quad \phi^*_\alpha(x) \\ &\quad s.t. \quad \max_{x, i \in [1,K]} \Pr[\mathcal{H}(x) = c_i] < Pr(c_i) + \delta \end{aligned} \tag{2}$$

where $\mathcal{H}$ is the intermediate classifier built on existing labeled examples L, ϕ^*_α is the maximum probability that x falls in a leaf node of the decision tree built on L, and $\delta > 0$ is a small constant. The objective function selects a sample x for which its classification by $\mathcal{H}$ into any class $c_{i \in [1,K]}$ is no better than random guessing according to the prior when δ is very small, while in the mean time, the probability $\phi^*_\alpha(x)$ that x is assigned to a leaf node in the decision-tree counterpart

of $\mathcal{H}$ is the greatest compared to other unlabeled samples. The selected sample achieves maximum homogeneity at leaf nodes, and therefore is unlikely to cause an internal node to split into new leaves. Our handling of classifier complexity during the sampling process draws a distinct line between our problem and the traditional active learning problem.

To estimate the probability that a sample x is assigned to a leaf node given the topology of a decision tree, we resort to the principle of maximum entropy. Let $X = \{x_i \in \mathbb{R}^d \mid i = 1, \ldots, N\}$ be a set of N unlabeled examples, and $\mu = \{\mu_\alpha \in \mathbb{R}^d \mid \alpha = 1, \ldots, J\}$ be a set of $J \ll N$ prototypes of labeled examples assigned to J leaf nodes in the current decision tree. Given no prior knowledge, the best way to relate an unlabeled data point in X and the representatives μ_α of the labeled data points is the maximum entropy distribution. Let $\phi_\alpha(x) = Pr(x \to \alpha)$ be the probability that data point x is assigned to leaf α, and we seek to optimize:

$$\max \sum_{\alpha=1}^{J} -\Pr[x \to \alpha] \log \Pr[x \to \alpha]$$

$$s.t. \quad \mathbb{E}(d_x) = \sum_{\alpha=1}^{J} \Pr(x \to \alpha) d(x, \mu_\alpha)$$

where $\mathbb{E}(d_x)$ is the expected distance between x and the prototypes of all the leaf nodes, and $d(\cdot, \cdot)$ is a distance measure, for example, Euclidean distance. The solution is the Gibbs distribution:

$$\Pr(x \to \alpha) = \frac{\exp(-\beta d(x, \mu_\alpha))}{\sum_j \exp(-\beta d(x, \mu_j))} \tag{3}$$

where β is the Lagrange multiplier that controls the degree of fuzziness of the probability distribution [10]. When $\beta = 0$, x is equally probable to be assigned to any leaf node. When β is large, the assignment of x conforms to the nearest neighbor philosophy. In a sequential sampling process, β can be incremented gradually in each iteration as more samples are used to estimate the leaf prototypes. $\phi^*_\alpha(x)$, the maximum probability of x over $\alpha = \{1, \ldots, J\}$, is:

$$\phi^*_\alpha(x) = \max_\alpha \frac{\exp(-\beta d(x, \mu_\alpha))}{\sum_j \exp(-\beta d(x, \mu_j))} \tag{4}$$

The optimization objectives with respect to decision tree complexity, specified in Eqs. (1) and (2), can be rewritten as: $\arg\max_{x \in X} \phi^*_\alpha(x)$, given the set of unlabeled examples X.

Putting everything together, we have a sampling technique uniquely designed for reverse engineering a black box classifier with minimum query cost and classification complexity. Without loss of generality, let $\mathcal{S}^t$ be the training set after the t^{th} sample s_t has been added to the training set, where $t \geq 0$. When $t = 0$, $\mathcal{S}^t$ represents the initial training data we have at our disposal. Let the next query point be s^{t+1}. We estimate $\mu^t = \{\mu^t_\alpha \in \mathbb{R}^d \mid \alpha = 1, \ldots, J\}$, the set of J prototypes of $\mathcal{S}^t_\alpha \subset \mathcal{S}^t$ assigned to the J leaf nodes in the decision tree built on $\mathcal{S}^t$. We choose a query point s^{t+1} as follows:

$$s^{t+1} = \arg\max_{s \in \mathcal{S}_u^t} \phi_\alpha^*(s)$$
$$s.t. \quad \psi_u(s) > \psi_u(s' \in \mathcal{S}_u^t | s' \neq s)$$

where ψ_u is a function that measures uncertainty, ϕ_α^* is defined in Eq. (4), and $\mathcal{S}_u^t$ is the set of unlabeled points.

4.2 Direct Hypothesis Formation

As mentioned earlier, we can build a decision tree directly from the training set with the query points. Query-by-bagging [1] is more stable and more accurate in decision tree active learning. In query-by-bagging, a committee of decision tree classifiers is built on subsets of training data, and query points are selected if the committee has the largest variance on the predictions. We modify the sampling technique by incorporating the minimum classifier complexity objective. Let M be the number of component classifiers in the committee, and $h_{i|i=1\cdots M}$ be the i^{th} component classifier. Let c be the total number of classes, at the t^{th} step the total number of component classifiers that predict $s \in \mathcal{S}_u^t$ as $c_{k|k\in\{1,\cdots,K\}}$, denoted as T_k, is: $T_k(s) = |\{m \leq M | h_m(s) = c_k)\}|$ and $T(s) = [T_1(s), \cdots, T_K(s)]$ records the total number of component classifiers that classify s as $c_{k,\forall k=1,\cdots,K}$. We select $s^{t+1} \in \mathcal{S}_u^t$ by solving the following optimization problem:

$$s^{t+1} = \arg\max_{s \in \mathcal{S}_u^t} \phi_\alpha^*(s)$$
$$s.t. |\max(T(s)) - \min(T(s))| < |\max(T(s')) - \min(T(s'))| \quad \forall s' \in \mathcal{S}_u^t | s' \neq s.$$

where the largest variance constraint is equivalent to the maximum uncertainty constraint specified in Eq. (2).

4.3 Indirect Hypothesis Formation

In indirect hypothesis formation, we introduce an intermediate kernel-based classifier for selectively sampling query points. Let $\mathcal{S}^t$ be the training set after the t^{th} round of sampling, we update the intermediate classifier and the decision tree classifier on $\mathcal{S}^t$. Let $\mathcal{S}_u^t$ be the set of unlabeled data from which query points are selected. In this study, we choose SVM as the intermediate learning algorithm. We estimate the uncertainty of a sample point using margin distance.

We now include the minimum complexity constraint in the sampling process. We relax the minimax margin constraint by including a small group of candidate points that are δ-*close* to the one that satisfies the minimax margin constraint:

$$\max \phi_D(s) < \min_{\forall s' \in \mathcal{S}_u^t} \max \phi_D(s') + \delta \tag{5}$$

where ϕ_D measures the margin distances of s to the positive and negative borders, and $\delta > 0$ is a small constant. We select a query point s^{t+1} from $\mathcal{S}_u^t$ such that

$$s^{t+1} = \arg\max_{s \in \mathcal{S}_u^t} \phi_\alpha^*(s)$$
$$s.t. \quad \max \phi_D(s) < \min_{\forall s' \in \mathcal{S}_u^t} \max \phi_D(s') + \delta.$$

where $\phi^*_\alpha(s)$ is the maximum probability s is consistent with a leaf node and the minimax margin constraint is equivalent to the maximum uncertainty constraint in Eq. (2). In general, given a dataset of m features and n instances, the time complexity of our algorithm is $O(mn^2)$ with a nonlinear kernel and the standard implementation of decision trees. It can be reduced to $O(mn)$ with a linear kernel [12] and a fast decision tree learning algorithm [21].

5 Experimental Results

We design a set of experiments to verify the applicability of our reverse engineering techniques for increasing transparency, with `DT REAL` referring to the direct hypothesis formation and `SVM-DT REAL` referring to the indirect hypothesis formation. The success is measured by examining the tree size and the **fidelity**—percentage of matching predictions by the reverse engineered classifier and the black-box classifier on independent *unseen* data sets. We run our experiments on four real data sets from the UCI repository [15]. δ in Eq. (5) is set to twice of the difference between the smallest and the second smallest margin. We clarify a few issues regarding our experimental setup in Table 2.

Table 2. Empirical study related issues

Issue	Clarification
(1) Should we use random sampling or active learning as the baseline?	In all our experiments, we choose to compare against the active learning baselines since they are significantly better than random sampling
(2) What learning algorithms should be used to train the black-box classifier?	Our reverse engineering learning model is classifier agnostic. It is not designed to gear towards any particular learning models
(3) Which design of decision tree should we use to train our classifier in the reverse engineering process?	We do not favor one type of decision tree over another in either our algorithmic design or our empirical study, because our algorithm is applicable to any decision tree design
(4) When should we terminate the reverse engineering process?	In many real applications, querying black-box for labels is not free (for example, getting credit score report). In addition, frequently querying actions may be considered as a suspicious abnormal behavior and would not be granted by companies' security standard. In practice, one can stop when either the budget or a desired fidelity measure has been reached. In our experiment, we allow the number of query points to be at most 10% of the size of the training data used to train the black-box classifier

5.1 Experiments on UCI Datasets

We test our techniques on four UCI Datasets: *Banknote Authentication*, *Cardiotocography*, *Phishing Websites* and *Human Activity Recognition with Smartphones* (referred to as *Smartphone* hereafter). The black-box classifiers are trained with support vector machine (`SVM`), logistic regression (`LOGIT`), decision tree (`DT`), naïve Bayes (`NB`), and neural network (`NN`). For `SVM`, we use Gaussian kernels with C = 10000; for `NN`, we set the number of nodes on the hidden layer to be 10. All the algorithms in our experiments are implemented in Matlab. All experiments are repeated 10 times and the average results are reported. *The accuracy of the black box classifier is shown as a dashed line in all figures as auxiliary information.* Detailed results are shown in Appendix A.

Banknote Authentication. The dataset has 1372 instances and two classes *genuine* or *forged*. We divide the data set equally into two parts: one for training the black box classifiers and the other is for active learning. The latter is further dived into two parts: one fifth is used for selecting query points to reverse engineer the black box classifier, and the rest is used for independent testing. The number of examples used at the beginning of reverse engineering is 1% of the size of the training data used to train the black box classifiers.

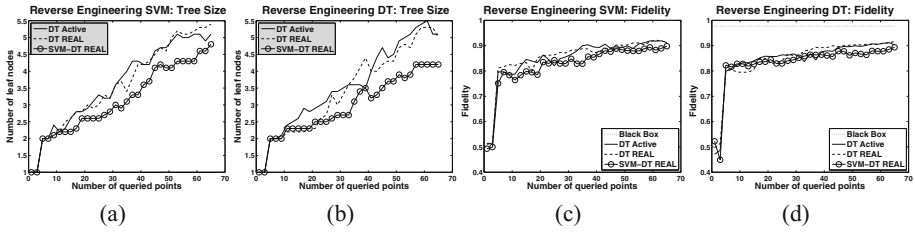
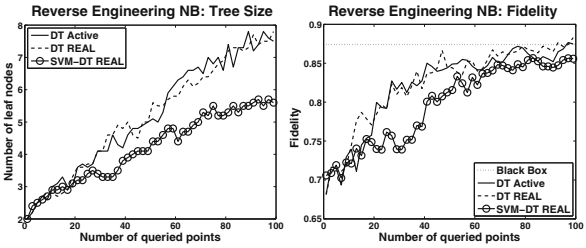


Fig. 2. Reverse engineering `SVM` and `DT` on the *Banknote Authentication* data set, using `DT Active`, `DT REAL`, and `SVM-DT REAL` decision tree learners.

Figure 2 shows the results of the two reverse engineering (RE) learners—`DT REAL` and `SVM-DT REAL`, and the baseline decision tree active learner—`DT Active` [1] with support vector machine (`SVM`) and decision tree (`DT`) as the black-box classifiers. Figures 2(a) and (b) show the growth of the complexity of the decision trees in terms of the number of leaf nodes as the number of queried samples increases. The solid line (—) is the baseline decision tree active learner (`DT Active`), the dashed line (- - -) is the decision tree active learner using our minimum cost and complexity sampling technique (`DT REAL`). The solid line with circular markers (–○–) is the SVM-DT RE classifier (`SVM-DT REAL`) also using the minimum cost and complexity sampling technique. It is clear that the complexity of the SVM-DT RE learner is consistently lower than that of the DT active learner. Although applying the same minimum cost and

complexity sampling technique, the DT RE learner cannot effectively produce decision trees with lower complexity. Figures 2(c) and (d) show the fidelity of the three classifiers. Note that fidelity is the percentage of agreement between the predictions made by each classifier and the black box classifier. All three classifiers have comparable performance in terms of fidelity. The flat dashed lines show the accuracy of the black box classifiers. All three classifiers manage to predict similarly as the black box classifier more than 90% of the time, with a sample size less than 10% of the size of the training data used to train the black box classifiers. The results for the rest of the black-box classifiers are similar. Due to page limitations, we do not show the plots.

Cardiotocogram. The dataset consists of 2126 instances of fetal cardiotocograms. We select 21 features and classify a cardiotocogram to one of the three fetal states: {N, S, P} where N is `normal`, S is `suspect`, and P is `pathologic`. We again divide the data set equally into two subsets, one for training the black-box classifiers, and the other for query point sampling and testing (among which 20% is used as query data, and the rest is used as the independent test set).



(a) (b)

Fig. 3. Reverse engineering the *naïve Bayes* classifier on the *Cardiotocogram* data set, using `DT Active`, `DT REAL`, and `SVM-DT REAL` decision tree learners.

Figure 3 shows the results of `DT REAL`, `SVM-DT REAL`, and the baseline `DT Active` with *naïve Bayes* as the black-box classifier. Figure 3(a) shows the growth of the complexity of the decision trees in terms of the number of leaf nodes. Again, the complexity of the SVM-DT RE learner is consistently lower than that of the DT active learner and the DT RE learner. Figures 3(b) shows the fidelity of the three classifiers. The results for the rest of the black-box classifiers are similar. Due to page limitations, we do not show the plots.

Phishing Websites Dataset. The dataset has 30 attributes that characterize phishing websites. The learning task is a binary classification problem. There are 2456 instances in the data set. We again use 50% of the data for training the

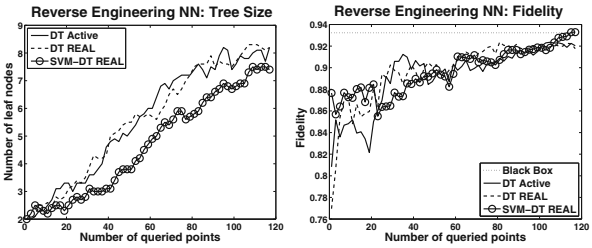


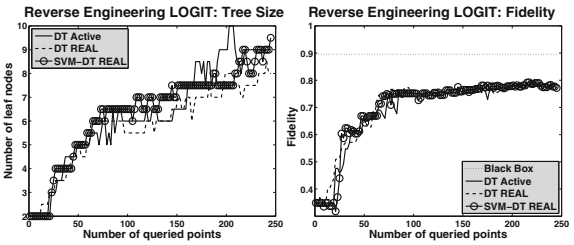
Fig. 4. Reverse engineering the *neural network* classifier on the *Phishing Website* data set, using `DT Active`, `DT REAL`, and `SVM-DT REAL` decision tree learners.

black box classifiers, and the other 50% for query point sampling and testing ($\frac{1}{5}$ as query data and the rest $\frac{4}{5}$ is used as independent test data).

Figure 4 shows the results of the three algorithms reverse engineering the *artificial neural network* black box classifier. Figure 4(a) shows the growth of the complexity of the decision trees in terms of the number of leaf nodes as the number of queried data points increases. Again, the complexity of the SVM-DT RE learner is consistently lower than that of the other two DT learners. Figure 4(b) shows the fidelity of the three classifiers. Except for the case where the black box classifier is trained with SVM, all three learners manage to exceed 90% fidelity with a sample size less than 10% of the training data used to train the black box classifiers. In the case of SVM as the black box classifier, the fidelity of the three learners is slightly less than 90% (above 88%). The results for the rest of the black-box classifiers are similar. Due to page limitations, we do not show the plots of the rest of the black box classifiers.

Smartphone. The dataset contains 10299 instances. Each instance has 562 attributes, and there are six class labels. This is the most complicated data set we used in our experiment. We randomly select 25% of the data for training the black box classifiers, 5% of the data for reverse engineering the black box classifiers, and then 25% random samples for independent testing.

Figure 5 shows the results of the three reverse engineering classifiers for the black box classifier: *logistic regression*. Figure 5(a) shows the growth of the complexity of the decision trees in terms of the number of leaf nodes. In this case, the complexity of the DT RE learner is in general lower than that of the other two learners. The SVM-DT RE learner is mostly comparable to the DT active learner in terms of complexity. Figure 5(b) shows the fidelity of the three classifiers. All three learners manage to achieve approximately 80% fidelity with a sample size less than 10% of the size of the training data for the black box classifier. The results for the rest of the black-box classifiers are similar. Due to page limitations, we do not show the plots of the other four black box classifiers.



(a) (b)

Fig. 5. Reverse engineering the *logistic regression* classifier on the *Smartphone* data set, using `DT Active`, `DT REAL`, and `SVM-DT REAL` decision tree learners.

6 Conclusions and Future Work

We investigate the feasibility of improving model transparency of data mining algorithms by reverse engineering a black-box classifier and transforming it to a decision tree. Our objective is to increase the transparency of the original black-box classifier with a small number of query points. We develop a reverse engineering learning technique that samples unlabeled data according to the principle of maximum uncertainty and minimum classifier complexity. Our experimental results demonstrate that our idea of reverse engineering classifiers is both feasible and practical. We also show that our reverse engineering model with indirect hypothesis formation is superior to traditional active learning with decision trees and SVMs. In the future, we would like to consider the problem in a game theoretic setting in which the black-box classifier employs a defense strategy against this type of reverse engineering, and the user counters the defense strategy with more sophisticated reverse engineering techniques.

Acknowledgement. The research reported herein was supported in part by NIH award 1R01HG006844, NSF awards CNS-1111529, CICI-1547324, and IIS-1633331 and ARO award W911NF-17-1-0356.

Appendix A Tree Size and Fidelity

Table 3 lists the results of all the test cases. Each row shows the tree size and the fidelity results of the last round right before the reverse engineering or active learning process terminates. As can be observed from the table, the `SVM-DT REAL` learner almost always produces smaller trees than the `DT Active` learner while producing comparable fidelity values. The accuracy results are very similar to the fidelity and therefore are eliminated from the paper because of page limitations.

Table 3. Reverse engineering results on all the datasets.

Banknote Authentication						
BC	DT-A		DT-R		SVM-DT-R	
	Tree Size	Fidelity	Tree Size	Fidelity	Tree Size	Fidelity
SVM	5.0000 ± 0.8433	0.9009 ± 0.0162	5.5000 ± 1.0328	0.9024 ± 0.0304	4.7000 ± 1.1547	0.9005 ± 0.0308
Logit	5.3000 ± 0.9487	0.9035 ± 0.0348	5.4000 ± 0.8433	0.9089 ± 0.0274	4.0000 ± 1.2472	0.8949 ± 0.0256
DT	5.2000 ± 1.1972	0.9171 ± 0.0333	5.2000 ± 0.9944	0.9072 ± 0.0383	4.2000 ± 1.0801	0.9023 ± 0.0417
NB	4.2000 ± 1.3166	0.9322 ± 0.0143	3.9000 ± 0.8165	0.9335 ± 0.0189	3.7000 ± 0.8233	0.9379 ± 0.0139
ANN	5.8000 ± 1.1005	0.9172 ± 0.0298	5.6000 ± 0.9087	0.9029 ± 0.0364	4.1000 ± 0.9033	0.9076 ± 0.0141
Cardiotocogram						
BC	DT-A		DT-R		SVM-DT-R	
	Tree Size	Fidelity	Tree Size	Fidelity	Tree Size	Fidelity
SVM	3.0000 ± 1.4907	0.9779 ± 0.0158	3.2000 ± 1.6193	0.9740 ± 0.0189	3.1000 ± 1.1972	0.9329 ± 0.0411
Logit	6.2000 ± 1.3984	0.9116 ± 0.0232	5.6000 ± 1.7764	0.9102 ± 0.0306	4.7000 ± 1.3375	0.9141 ± 0.0165
DT	5.5000 ± 0.9718	0.9629 ± 0.0222	5.5000 ± 1.0801	0.9574 ± 0.0222	4.6000 ± 1.0750	0.9267 ± 0.0312
NB	7.5000 ± 1.5811	0.8747 ± 0.0256	7.8000 ± 1.3984	0.8829 ± 0.0297	5.6000 ± 0.9661	0.8558 ± 0.0223
ANN	5.4000 ± 2.0656	0.9249 ± 0.0300	4.9000 ± 1.6633	0.9293 ± 0.0322	4.6000 ± 1.4298	0.9108 ± 0.0112
Phishing Websites						
BC	DT-A		DT-R		SVM-DT-R	
	Tree Size	Fidelity	Tree Size	Fidelity	Tree Size	Fidelity
SVM	9.1000 ± 1.7288	0.8894 ± 0.0167	9.7000 ± 2.2632	0.8965 ± 0.0168	7.6000 ± 1.3499	0.8879 ± 0.0109
Logit	9.4000 ± 1.7764	0.9214 ± 0.0139	8.9000 ± 1.7920	0.9199 ± 0.0113	7.8000 ± 1.6193	0.9181 ± 0.0131
DT	8.9000 ± 1.3703	0.9189 ± 0.0192	9.3000 ± 1.3375	0.9198 ± 0.0260	7.6000 ± 0.5164	0.9187 ± 0.0119
NB	8.8000 ± 2.2010	0.9054 ± 0.0190	8.4000 ± 1.0750	0.9018 ± 0.0193	7.6000 ± 1.3499	0.9066 ± 0.0152
ANN	7.7000 ± 1.5670	0.9225 ± 0.0233	8.1000 ± 1.7288	0.9223 ± 0.0148	7.5000 ± 1.2693	0.9328 ± 0.0098
Smartphone						
BC	DT-A		DT-R		SVM-DT-R	
	Tree Size	Fidelity	Tree Size	Fidelity	Tree Size	Fidelity
SVM	8.5000 ± 2.0790	0.7348 ± 0.0110	10.5000 ± 1.6364	0.7377 ± 0.0175	9.8000 ± 2.3118	0.7488 ± 0.0152
Logit	9.1000 ± 1.1785	0.7783 ± 0.0175	8.0000 ± 1.7029	0.7767 ± 0.0112	9.5000 ± 1.9120	0.7927 ± 0.0159
DT	10.9000 ± 1.3038	0.7693 ± 0.0087	10.5000 ± 2.1679	0.7689 ± 0.0152	11.0000 ± 1.0954	0.7634 ± 0.0086
NB	8.0000 ± 2.1145	0.8292 ± 0.0445	8.0000 ± 1.3012	0.8252 ± 0.0422	8.5000 ± 1.8166	0.8280 ± 0.0357
ANN	10.2000 ± 1.1145	0.7878 ± 0.0218	8.9000 ± 1.5166	0.7842 ± 0.0309	10.0000 ± 2.5100	0.7655 ± 0.0125

References

1. Abe, N., Mamitsuka, H.: Query learning strategies using boosting and bagging. In: ICML, pp. 1–9 (1998)
2. Beygelzimer, A., Dasgupta, S., Langford, J.: Importance weighted active learning. In: ICML, pp. 49–56 (2009)
3. Craven, M.W., Shavlik, J.W.: Extracting tree-structured representations of trained networks. In: Proceedings of the 8th International Conference on Neural Information Processing Systems, pp. 24–30 (1995)
4. Datta, A., Sen, S., Zick, Y.: Algorithmic transparency via quantitative input influence: theory and experiments with learning systems. In: 2016 IEEE Symposium on Security and Privacy (SP), pp. 598–617 (2016)
5. Diederich, J. (ed.): Rule Extraction from Support Vector Machines. Studies in Computational Intelligence, vol. 80. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-75390-2
6. Duivesteijn, W., Thaele, J.: Understanding where your classifier does (not) work - the scape model class for EMM. In: ICDM, pp. 809–814 (2014)
7. Dwyer, K., Holte, R.: Decision tree instability and active learning. In: Kok, J.N., Koronacki, J., Mantaras, R.L., Matwin, S., Mladenič, D., Skowron, A. (eds.) ECML 2007. LNCS (LNAI), vol. 4701, pp. 128–139. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-74958-5_15

8. Dyer, M., Frieze, A., Kannan, R.: A random polynomial-time algorithm for approximating the volume of convex bodies. J. ACM **38**(1), 1–17 (1991)
9. Fung, G., Sandilya, S., Rao, R.B.: Rule extraction from linear support vector machines. In: ACM SIGKDD, pp. 32–40 (2005)
10. Held, M., Buhmann, J.M.: Unsupervised on-line learning of decision trees for hierarchical data analysis. In: Advances in Neural Information Processing Systems, pp. 514–520 (1998)
11. Henelius, A., Puolamäki, K., Boström, H., Asker, L., Papapetrou, P.: A peek into the black box: exploring classifiers by randomization. Data Min. Knowl. Discov. **28**(5–6), 1503–1529 (2014)
12. Joachims, T.: Training linear SVMs in linear time. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006, pp. 217–226. ACM, New York (2006)
13. Klivans, A.R., O'Donnell, R., Servedio, R.A.: Learning geometric concepts via Gaussian surface area. In: Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 541–550 (2008)
14. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: ACM SIGIR, pp. 3–12 (1994)
15. Lichman, M.: UCI machine learning repository (2013). http://archive.ics.uci.edu/ml
16. PCAST: Big data and privacy: a technological perspective (2014). https://www.whitehouse.gov/sites/default/files/microsites/ostp/PCAST/pcast_big_data_and_privacy_-_may_2014.pdf
17. Rademacher, L., Goyal, N.: Learning convex bodies is hard. In: COLT (2009)
18. Ribeiro, M.T., Singh, S., Guestrin, C.: "Why should i trust you?": explaining the predictions of any classifier. In: SIGKDD, pp. 1135–1144 (2016)
19. Roy, N., McCallum, A.: Toward optimal active learning through sampling estimation of error reduction. In: ICML, pp. 441–448 (2001)
20. Saad, E.W., Wunsch II, D.C.: Neural network explanation using inversion. Neural Netw. **20**(1), 78–93 (2007)
21. Su, J., Zhang, H.: A fast decision tree learning algorithm. In: Proceedings of the 21st National Conference on Artificial Intelligence, AAAI 2006, vol. 1, pp. 500–505. AAAI Press (2006)
22. Sweeney, L.: Discrimination in online ad delivery. Commun. ACM **56**(5), 44–54 (2013)
23. Tong, S., Koller, D.: Active learning for parameter estimation in Bayesian networks. In: NIPS, pp. 647–653 (2001)