The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

A Coverage-Based Utility Model for Identifying Unknown Unknowns

Gagan Bansal, Daniel S. Weld
Paul G. Allen School of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{bansalg, weld}@cs.washington.edu

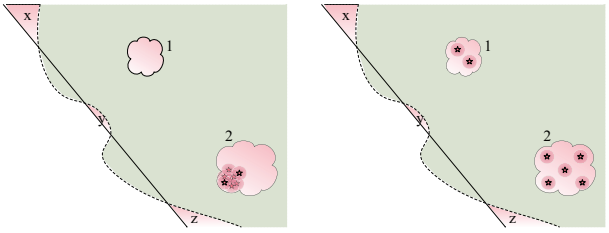
Abstract

A classifier's low confidence in prediction is often indicative of whether its prediction will be wrong; in this case, inputs are called *known unknowns*. In contrast, *unknown unknowns* (UUs) are inputs on which a classifier makes a high confidence mistake. Identifying UUs is especially important in safety-critical domains like medicine (diagnosis) and law (recidivism prediction). Previous work by Lakkaraju et al. (2017) on identifying unknown unknowns assumes that the utility of each revealed UU is independent of the others, rather than considering the set holistically. While this assumption yields an efficient discovery algorithm, we argue that it produces an incomplete understanding of the classifier's limitations. In response, this paper proposes a new class of utility models that rewards how well the discovered UUs cover (or "explain") a sample distribution of expected queries. Although choosing an optimal cover is intractable, even if the UUs were known, our utility model is monotone submodular, affording a greedy discovery strategy. Experimental results on four datasets show that our method outperforms bandit-based approaches and achieves within 60.9% utility of an omniscient, tractable upper bound.

Introduction

With the rise of machine learning, pre-trained classifiers are often available as a Web service accessible via an API, e.g., Microsoft Cognitive Services, Amazon Rekognition and IBM Watson. Since users of these services will apply them to their own data, which likely differs from the service provider's training data, the classification performance may deviate from expectations.

? (2015) found that a classifier may be prone to systematically making mistakes on inputs, despite being highly confident of its prediction; they referred to such high-confidence mistakes as "unknown unknowns." While many reasons can account for poor classifier performance, **?** (**?**) discusses an interesting one: unmodeled aspects of the world. He also used the term, unknown unknowns, for the resultant errors, because the system is unaware of its mistake. For a classifier, such situations may arise because of bias in the training data, a difference in the training and test set distribution, insufficient classifier expressiveness, and others.



(a) (b)

Figure 1: Two payoff sets of seven unknown unknowns (UUs). The green region denotes the space of true positives; the straight black line shows the decision boundary; red regions are false positives. Red areas x, y, and z are low confidence predictions near the decision boundary. Red interior regions 1 and 2 contain UUs. The stars indicate the discovered UUs. (a) Lakkaraju *et al.*'s utility function yields a tightly clustered set and does not discover the upper blind spot. (b) our utility function rewards better coverage.

Identifying a classifier's UUs is an important task for many reasons: for understanding the classifier's limitations, for debugging it (Amershi et al. 2015), for attacking (Szegedy et al. 2014) or preventing attacks against it (Goodfellow, Shlens, and Szegedy 2015), or for deploying a high performing human-machine hybrid system (Werling et al. 2015; **?**). Understanding a machine learner's limitations is especially important when developing safer AI systems for use in high-stakes domains, such as health care and transportation, where errors could be catastrophic (**?**; **?**; **?**).

? (2015) found that many UUs, particularly high-confidence mistakes, may be present as systematic errors in a specific region of a classifier's feature space, a kind of "blind spot." They developed a novel crowdsourcing workflow that provides human workers monetary incentives for discovering UUs. Subsequently, Lakkaraju et al. (2017) developed an *automatic* method to find UUs for black-box classifiers using an innovative clustering and reinforcement-learning-based algorithm. However, their method assumes that discovering a UU results in a fixed reward *independent* of any observation that the system might have made previously. While this independence affords an efficient algo-

rithm for finding UUs, their utility model has an important limitation — it provides the same reward for finding a novel UU as it does for finding another UU that is indistinguishable from previous finds. For example, consider the two sets of UUs shown in Figure 1. While these sets are of equal size, the second set (b) is more informative since it: 1) documents the presence of multiple blind spots in the feature space, and 2) better characterizes their extent. However, Lakkaraju *et al.*'s utility model assigns the same reward to both sets. Furthermore, their bandit algorithm is *more likely* to return set (a) than (b) because it recognizes that it is more likely to find additional UUs in the vicinity of the first one it found (see, e.g., Figure 2). Finally, the utility model rewards discovery of an unlikely UU in the same way as it does for one found in a dense area of feature space.

This paper corrects these problems by offering a more expressive class of utility models and a new algorithm for identifying UUs. We assume a setting that has access to a classifier's predictions and its confidence in them. At some expense, the system can query a human oracle to check the true label for any given input, determining if it is a correct prediction or yields an error. Since querying the oracle is expensive, we assume a budget constraint that specifies the maximum number of oracle calls. The utility model specifies the usefulness of querying a set of inputs for the task. The objective is to select a set of inputs from the the test set that maximizes the achieved utility. In summary, we make the following contributions:

- We identify three issues with the utility model used by previous work to identify unknown unknowns that leads to an incomplete understanding of the machine learning classifier's blind spots.
- As a solution, we propose a new class of utility models for the task that evaluates the *coverage* of a set of discovered unknown unknowns over a sample distribution of expected test time queries.
- We show that our coverage-based utility model is monotone submodular; hence, an omniscient greedy strategy will find a cover within a constant factor of optimal, which forms a tractable upper bound for evaluating solutions. We further implement two online approximation algorithms, Greedy-conf and Greedy-uniform, that utilize different probabilistic priors.
- Finally, we present experiments showing that our Greedy-conf algorithm performs substantially better than several baselines and within 60.9% of the tractable upper bound.

Previous Work

While Attenberg *et al.* (2015) experimentally showed the presence of UUs using human workers to manually *craft* these inputs, Lakkaraju et al. proposed an *automatic* approach for identifying UUs. Their seminal paper formulated the task as an optimization problem, where the system seeks a sequence of queries to an oracle requesting an input's label that maximizes a utility function. (We discuss their utility function later in this paper.)

Lakkaraju et al. presented a two-phase algorithm that relies on clustering and reinforcement learning to maximize their utility function. The first phase clusters test points so that inputs in the same cluster have *both* similar features and similar prediction-confidence values from the classifier. The rationale is that the systematic occurrence of a classifier's UUs implies that certain clusters will have a higher concentration of UUs than others. In the second phase, a bandit algorithm, which treats each cluster as an arm, picks inputs from D_{test}; this action includes 1) choosing a cluster, and then 2) randomly drawing an input from the cluster without replacement. Lakkaraju et al. showed that their approach discovered more UUs than simpler methods, such as querying the input on which the classifier made the least confident prediction.

This paper extends Lakkaraju et al.'s investigation of automatic methods for identifying UUs by proposing a novel class of utility models that lead to alternative methods for solving the resulting optimization problem.

Problem Statement

We now formally describe the task of identifying unknown unknowns. Let M denote a classifier that maps an input x into one of the possible classes in Y. Let y_x denote the true label and $M(x)$ denote the predicted label of x. Let $c_{M,x}$ denote the classifier's confidence in this prediction. Let D_{train} denote the data on which the classifier was trained. We assume that we do not have access to D_{train}. Following Lakkaraju et al., we assume, without loss of generality, that the positive class is the critical class, i.e., false positives are costly and must be discovered. Let D_{test} denote a set of inputs on which the classifier makes a positive prediction with confidence greater than a threshold τ. Since the true labels of points in D_{test} are not known, the system can query the oracle for this information. Let B denote the maximum number of inputs the system may query. Let $U_M(Q, y_Q)$ denote a utility model that describes the usefulness of querying the oracle for the true label of a set of inputs $Q \subset D_{\text{test}}$. The task of identifying UUs can be formulated as an optimization problem of selecting a set of inputs in D_{test} that maximize the utility model U_M subject to the budget constraint. Equation 1 describes the resulting optimization problem. Here, $Q^* \subset D_{\text{test}}$ denotes the optimal set of inputs to query.

$$Q^* = \underset{Q \subset D_{\text{test}}, |Q| < B}{\arg\max} U_M(Q, y_Q) \tag{1}$$

Following Lakkaraju et al., we consider the sequential decision problem where a solution Q is constructed by choosing the inputs in D_{test} to query one by one. The system is assumed to observe the true label returned by the oracle after each query and uses this information when deciding the next query to make. We next discuss utility models appropriate for identifying UUs.

Utility Models for UUs

Intuitively, a utility model describes the benefit received by querying a subset of test inputs. Formally, a utility model is

a set function $U_M : 2^{D_{\text{test}} \times Y} \rightarrow \mathcal{R}$ that maps a set of queries and their true labels to a real number indicating the resulting utility. Queries that are more useful are mapped to a higher value.

A good utility function depends, of course, on how the UUs will be used. There are many possible objectives, *e.g.*, for attacking (Szegedy et al. 2014) or defending against attacks on the classifier (Goodfellow, Shlens, and Szegedy 2015), or for deploying a high performing human-machine hybrid system (**?**). One might also want to directly improve a classifier's performance, but here, active learning methods are likely more appropriate (Settles 2012). We focus on the case where a user seeks to understand the limitations, specifically high-confidence false positives, of an externally provided classifier, such as a pre-trained, Web-based ML API.

Let $S \subset Q$ indicate the subset of queries that resulted in the discovery of a UU, i.e., whether the classifier M made a *high confidence* mistake.

$$S \coloneqq \{q \mid q \in Q, y_q \neq M(q), c_{M,q} > \tau\} \tag{2}$$

Equation 3 describes the independent UU utility model used by Lakkaraju et al.[1] Note that each UU produces unit utility, and all other outcomes yield no value. Since the rewards earned by the discovery of each UU are independent of each other, the total utility simply sums the utilities from each query.

$$U_M(Q, y_Q) = \sum_{q \in Q} \mathbb{1}_S(q) \tag{3}$$

Here, $\mathbb{1}_S(q)$ is an indicator function, which denotes whether element q is a member of the set S. An optimal solution for this utility model is any set where every element is a UU: the independent utility model assigns the same value to two set of queries if they have the same number of UUs. For example, Lakkaraju *et al.*'s utility model assigns same value to the UUs in Figures 1a and 1b. The independence assumption makes their utility model fast to evaluate and amenable to a multi-armed bandit framework, in which each UU produces a unit reward.

Weaknesses of the Independent UU Utility Model

We argue that the independent UU utility model suffers from three major weaknesses (Ws).

W1 First, it assigns the same incremental utility for finding a novel UU as it does for finding a UU that is nearly indistinguishable from those already discovered. For example, suppose we have already observed five UUs in region 2 of Figure 1b, and those are the only UUs we have discovered. Would it be more useful to find another UU in region 2 or to find a novel UU in region 1? It seems clear that finding a UU in region 1, with the attendant recognition of a new classifier blind spot, should be preferred with higher utility, but the independent utility model has no such preference.

[1]In fact, we have slightly simplified their model. Lakkaraju et al. also included a term to penalize for the time or monetary cost of querying the oracle. Since this term is orthogonal to our argument, we omit it for simplicity.

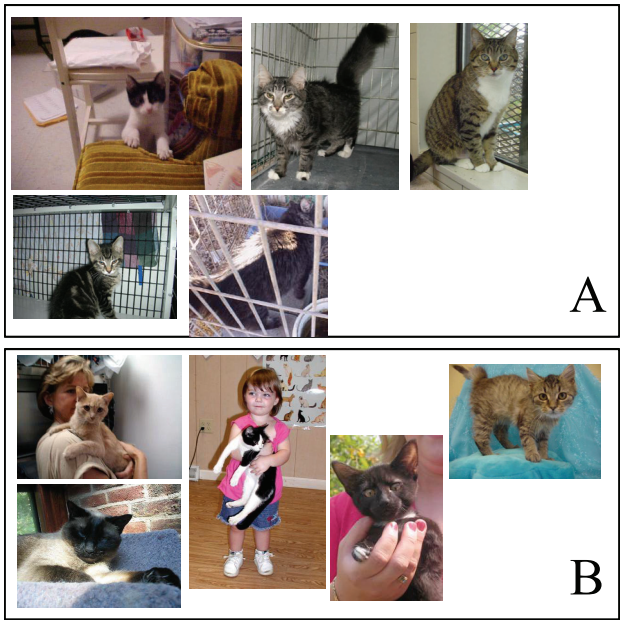


Figure 2: Inspection of the first five UUs discovered by the (A) optimal bandit policy and (B) Greedy-conf on the Kaggle13 dataset. Results show that Greedy-conf discovers a much more diverse set of UUs. The optimal bandit policy returns images that are very similar — most of the discovered false positives are Mackerel tabbies near a "cage".

W2 Second, this utility model assigns the same utility to a set of UUs irrespective of the extent to which these UUs *characterize* the extent of a blind spot. For example, consider any set of five UUs identified in region 2 of Figure 1a and those in region 2 of Figure 1b. Clearly, the second set of UUs better characterize the extent of the blind spot represented by region 2, yet this utility model will be indifferent to these two UU sets.

W3 Third, the utility model assigns the same utility to an extremely rare UU as it does to a UU that lies in a dense region of the feature space, signifying a potentially ruinous classifier error.

Furthermore, since Lakkaraju et al. used a clustering and bandit-based approach, the optimal policy (after exploration) is to choose the cluster with the highest density of UUs. As a result, even though the independent utility model assigns equal utility to the set of UUs in situations (a) and (b) of Figure 1, their algorithm will more likely return the set of UUs shown in Figure 1a because it realizes that it's more likely to find new UUs in the vicinity of the first one that it finds.

For example, Figure 2 shows that for an image classification task, the optimal bandit policy discovers UUs that look very similar, whereas our algorithm (described shortly) finds a more diverse set of UUs.

Together, these issues imply that optimizing the independent utility model will neither reward finding novel UUs nor reward finding UUs that characterize the extent of a systematic blind spot. However, doing both seems crucial for gain-

ing a complete understanding of the spectrum of UUs. We next discuss a new utility model to address these issues.

Coverage-Based Utility Model

To remedy the weaknesses just discussed, we propose a utility model that measures not just how *many* UUs have been found, but the *degree to which discovered UUs "cover" a sample distribution of inputs expected to be seen at test time*. Inspired by the work on optimal sensor placement (Krause, Singh, and Guestrin 2008), the main assumption our utility model makes is that a detected UU "covers" its neighboring inputs by providing an understanding of the classifier's behavior in its neighborhood. Thus, any individual UU explains only a small part, containing its neighbors, of the complete space of UUs that must be analyzed. The utility of a set of UUs can then be evaluated in terms of *total effective coverage*. In this model, discovering a UU in close proximity to an already known UU results in only a small reward because of a large overlap in the regions being explained by these two UUs. And, as a result, the reward obtained for detecting a UU is now no longer independent of the observations made on past queries.

Equation 4 describes the new *coverage-based utility model*. Here, $P(\text{explain}_x \mid Q, y_Q)$ denotes the degree to which an input x has been explained as a result of querying a set of points Q. The model computes a weighted sum of this achieved coverage for all inputs in D_{test}, where the weights indicate the classifier's confidence in the prediction on that input. This is done to encourage the explaining and identifying of UUs in regions on which the classifier makes a higher confidence prediction.

$$U_M(Q, y_Q) = \sum_{x \in D_{\text{test}}} c_{M,x} \cdot P(\text{explain}_x \mid Q, y_Q) \quad (4)$$

To define the degree to which an individual input x in the test has been explained, we assume access to a similarity function $\text{sim}(\cdot, \cdot)$. Further, we assume that if Q contains no UUs, then this value is zero. However, when at least one UU has been identified, we assume that, for an input x, this value depends only on the most similar UU. The following equation describes this term after making these assumptions. Here, $S \subset Q$ denotes the subset of queries that are UUs.

$$P(\text{explain}_x \mid Q, y_Q) \coloneqq \max_{q \in Q} \mathbb{1}_S(q) \cdot \text{sim}(x, q) \quad (5)$$

We now explain how this utility model overcomes the limitations of the independent utility model used by Lakkaraju et al. This utility model addresses W1 by preferring the discovery of a novel UU over a UU that is similar to previously identified ones. This is because a large part of the region explained by the second UU will overlap with the previously identified UUs. For example, if we have already identified the UUs in region 2 of Figure 1b, and these are the only UUs we have discovered, then this utility model will assign a higher reward for identifying a UU in region 1 over another UU in region 2.

Further, this utility model corrects W2 by assigning a higher utility to a diverse set of UUs, which may better characterize a blind spot than a set of very similar UUs. For example, it will assign a higher utility to the five UUs in region 2 of Figure 1b than to the same number of closely clustered UUs in region 2 of Figure 1a.

Additionally, it may be more important to determine UUs in regions with a higher density of inputs. Since this utility model takes into account the coverage of neighboring inputs and hence the density of the distribution, it rewards finding UUs in such a region, addressing W3.

It is important to note that the coverage-based utility model produces rewards that are very different from those assumed in the classical, multi-armed bandit literature. The rewards associated with arms (clusters of test points in the model used by Lakkaraju et al.) decrease over time, diminishing the value of exploring that arm.[2]

Proposition 1. *Optimizing the coverage-based utility model defined in Equation 4 is intractable.*

Proof. Proof by reduction to set cover. □

We now state an important property satisfied by the coverage based utility model. This property motivates the greedy algorithm used for solving the resultant optimization problem presented in the next section.

Proposition 2. *The coverage-based utility model satisfies the diminishing returns property. If $\Delta(x \mid Q)$ denotes an increment in utility as result of querying a new input x when a set Q has already been queried, then $\Delta(x \mid Q_1) \leq \Delta(x \mid Q_2)$ for all $Q_1, Q_2 \subset D_{test}$ whenever $Q_2 \subset Q_1$.*

Proposition 2 (proved in the Appendix) implies that the coverage-based utility model is submodular, which guarantees that an omniscient, linear-time, greedy algorithm with access to the locations of all the UUs will find a solution within a constant factor of the optimum.

Optimization Algorithm

Since our proposed coverage-based utility model is submodular, it is natural to consider a greedy algorithm for choosing data points to query. The idea is to greedily pick the input with the highest expected reward while conditioning on the observations made thus far. For example, starting with a uniform prior over the likelihood of an input being a UU, a greedy solution will first pick the input with the highest expected gain in coverage of the test set. It would then observe the true label of this input and pick the next input from the test set with the highest expected reward taking into account the previous observation. As noted previously, since this utility model is submodular; if the complete set of UUs were known, then such an omniscient greedy strategy would find a solution within constant factor approximation of the optimal solution. However, in practice, the complete set of UUs will not be known. Instead, an alternative is a greedy strategy that starts with a prior (e.g., a uniform prior, one based on classifier confidence, or one based on clustering) over the likelihood of an input being a UU. The system would then

[2]This also implies that cumulative regret is an inappropriate measure of performance; as the number of queries goes to $|D_{\text{test}}|$, all algorithms have the same regret.

Algorithm 1 Greedy Search

Input: Test set D_{test}, prior ϕ and budget B
$Q = \{\}$ {inputs that have been queried}
$y_Q = \{\}$ {true label acquired from the oracle}
$t = 1$
while $t \leq B$ **do**
  $q' = \arg\max_{q \in D_{\text{test}} - Q} \phi(q) \cdot E[\Delta(q \mid Q)]$
  $y_{q'} = \text{Query\_oracle}(q')$
  $Q \leftarrow Q + q'$
  $y_Q \leftarrow y_Q + y_{q'}$
  $t \leftarrow t + 1$
  $\phi \leftarrow \text{Update\_prior}(Q, y_Q)$
end while
return S

update its belief as it makes observations and use the current prior to greedily pick the next input. It is important to note, however, that this strategy does not guarantee a solution with a worst case bound: the algorithm may query an input that results in *zero* reward at any point.

Algorithm 1 defines the greedy algorithm for optimizing the coverage-based utility model. Here, $E[\Delta(x \mid Q)]$ denotes the expected increment in utility, using the probability measure ϕ, as a result of querying x when the labels of points in Q have been observed. The term $\phi(x)$ specifies a prior probability of an input x being a UU, i.e., $\phi(x) = P(M(x) \neq y_x \mid Q)$. While many strategies could be used for updating the prior, we associate probabilities with clusters of examples, revising the cluster's estimate after querying any point inside it. As Equation 6 shows, we compute the new probability by smoothing between the observed frequency and the previous prior.

.

$$\phi_{t+1}(x) \leftarrow \frac{\lambda \cdot \phi_0(x) + N_t(\text{UU}, k_x)}{\lambda + N_t(k_x)} \tag{6}$$

Here, k_x denotes the cluster to which x belongs, $N_t(k_x)$ denotes the number of times a point from this cluster has been queried, and $N_t(\text{UU}, k_x)$ denotes the number of times a UU has been discovered in this cluster.

Experiments

We evaluate our methods on the same four classification datasets used by previous work (Lakkaraju et al. 2017).[3]

- Pang05 (Pang and Lee 2005): This dataset contains 10k sentences from movie reviews on Rotten Tomatoes. The objective is to classify whether an input sentence has positive or negative sentiment.
- Pang04 (Pang and Lee 2004): This dataset contains 10k sentences from IMDb plot summaries and Rotten Tomatoes movie reviews. The objective is to classify whether an input sentence is subjective or objective (positive class).

[3]Since Lakkaraju et al. did not release the code or datasets associated with their paper these datasets and the procedure for creating bias in the training sets are our reimplementation of their methods.

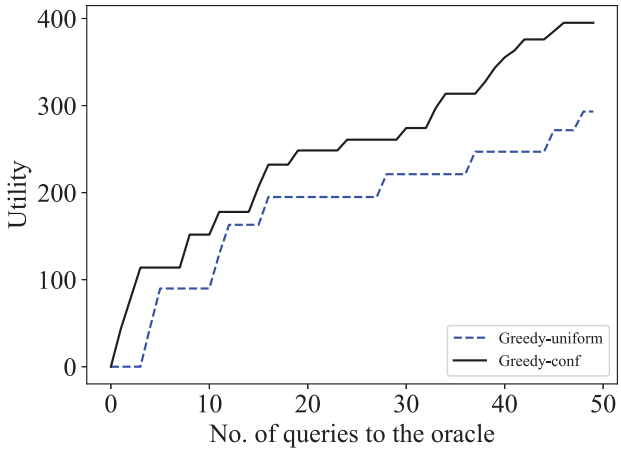


Figure 3: A confidence based prior performs better than using a uniform prior on the Kaggle13 dataset. Relative performance is similar in the other datasets (not shown).

- McAuley15 (McAuley, Pandey, and Leskovec 2015): This dataset contains Amazon reviews for books and electronic items. The objective is to classify whether an input review has positive or negative sentiment. We trained on 50k randomly-selected electronics reviews and tested on reviews of books.
- Kaggle13[4] : This dataset contains 25k images of cats and dogs in total, which were randomly split into a train and test set of equal size. The objective is to classify whether an input image is that of a cat or a dog (positive class). The training set was biased to ensure that it did not contain any black cats.

For all datasets, we limited the size of the test set to 5k. To replicate the bias on Pang05 and Pang04, we trained a decision tree on the training set and removed all data corresponding to a randomly chosen leaf with a the majority class that was negative. To create the "no black cat" bias for the Kaggle13 training dataset, we asked Crowdflower workers to rate the "blackness" of the animal on a scale of 1-5, where 1 refers to an animal with no black patches and 5 to an animal with patches of no color other than black. We then converted the labels into binary judgments wherein the animals with a rating of 5 were considered black. On a sample of 100 such inputs, we got a Fleiss Kappa score of 0.92. To encourage follow-on research, all our code and data sets are available on aiweb.cs.washington.edu/ai/unkunk18.

Systems for Finding Unknown Unknowns

This section describes the algorithms we used to optimize the coverage-based utility model.

- Upper bound (tractable) is the greedy algorithm with omniscience: it knows the location of all UUs.
- Most-uncertain greedily picks the most uncertain example to query.

[4]https://www.kaggle.com/c/dogs-vs-cats/data

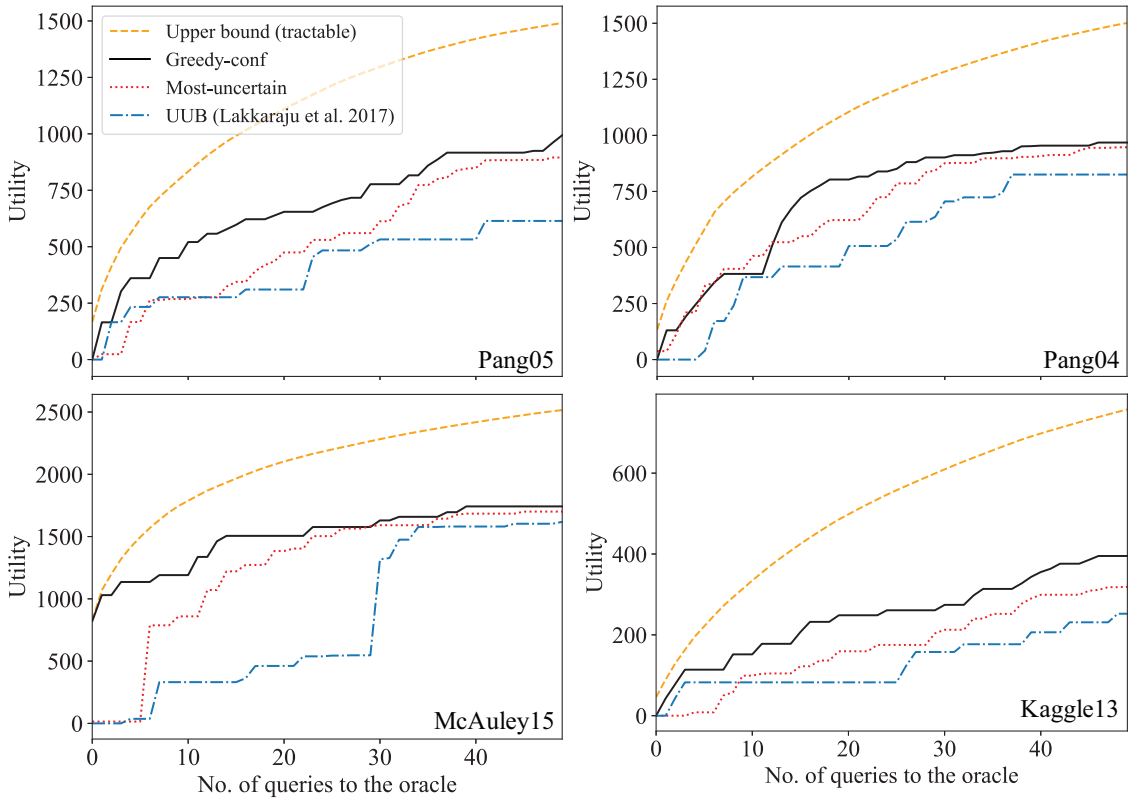


Figure 4: A comparison of the performance of different optimization algorithms for optimizing the coverage based utility model. The y-axis shows the utility received by an algorithm and the x-axis is the number of queries issued to the oracle. Unsurprisingly, UUB performs worse than Greedy-conf because it implicitly tries to optimize a different objective.

- UUB is the bandit algorithm proposed by Lakkaraju et al. Experiments showed that performance was best when it first picks the cluster (arm) using their UUB algorithm and then greedily picks the point in the cluster with the highest expected reward.
- Greedy-conf is a hill-climbing algorithm that uses the classifier's confidence as the initial prior and updates probabilities using the clustering-based procedure described previously.
- Greedy-uniform is a greedy algorithm which uses a fixed uniform prior.

Dataset	% gain over UUB	% of upper bound
Pang04	24.08	63.12
Pang05	42.67	58.75
McAuley15	41.89	73.54
Kaggle13	46.37	48.27

Table 1: A summary of the performance of Greedy-conf on four datasets. The first column shows the average percentage improvement as result of using Greedy-conf over UUB. The second column shows the percentage of the tractably achievable utility achieved by Greedy-conf on average.

Implementation Details

For the text datasets, we used logistic regression with unigram features. For Kaggle13, we used a CNN (two convolution layers and three linear layers). To cluster the inputs, we used the kmean-both algorithm used by Lakkaraju et al. This algorithm first clusters the inputs first by the classifier's confidence and then by the input-data features. The number of clusters were selected using the elbow method. At test time, to compute pairwise distances, we used unigram features for text datasets and raw pixels for image dataset. Note that the representation used at test time does not assume access to the exact representation used by the classifier. For example, for text datasets we do not assume access to the vocabulary used by the logistic regression classifier. For the image dataset, we rely solely on raw pixel values. Additionally, we reduce the dimensions of the representation used at test time to avoid computing pairwise distances in sparse high dimensional space, which can be inefficient. We used the following function as the similarity measure: $\text{sim}(x, s) := e^{-\frac{d(x,s)}{\sigma}}$. We used a gaussian function so that a UU covers only a small neighborhood. Here, $d(x, s)$ denotes Euclidean distance between x and s, and σ controls the width of the gaussian.

Results

To evaluate the effect of priors on the performance of our greedy algorithm, we compared the utility achieved by Greedy-conf and Greedy-uniform. On all four datasets, the confidence-based prior worked better than a uniform prior (e.g., see Figure 3).

We next compared our Greedy-conf method with previous work and the tractable upper bound (Figure 4). On all four datasets, Greedy-conf performed better than both UUB and Most-uncertain. Table 1 presents a quantitative analysis of the performance of Greedy-conf. The bandit based approach performed worse than Most-uncertain, an expected result, because the coverage-based utility model has decreasing rewards; Lakkaraju et al.'s approach was not designed to optimize this type of function.

As mentioned previously, our coverage-based utility function rewards an algorithm that finds a diverse set of UUs. The bandit algorithm, in contrast, keeps probing the vicinity of the first UUs that it finds. Figure 2 illustrates this behavior by showing the first 5 UUs returned by each approach on the Kaggle13 dataset. The bandit algorithm returns photos of three similar-looking cats, while our greedy algorithm returns a more diverse set of classifier failures.

While Greedy-conf achieves a higher utility than UUB, both algorithms have the same time complexity: $O(B^2N^2)$. Most-uncertain, in contrast, is faster and has time complexity of $O(Nlog(N))$. All approaches have the same space complexity: $O(N)$.

Other Related Work

Recently, **?** (2017) argued for development of robust AI systems, listing a variety of approaches for handling both known and unknown unknowns. While we (and Lakkaraju et al.) focus on high-confidence unknown unknowns, **?** (2017) also considers two related problems. First, *open category recognition* submits a query that belongs to a previously unseen class to the system at test time (**?**; **?**). Second, *change-point detection*, detects changes in the query distribution over time (**?**; **?**). Our work on coverage-based utility model differs by rewarding detection of a diverse set of failure modes of a classifier in order to characterize the classifier's limitations in different regions of the feature space. This complements traditional characterizations in terms of (say) precision and recall.

Related to our problem is the task of evaluating in a cost-effective manner the performance of a large-scale machine learning system (**?**). Here, the objective is to estimate the system performance on a test set without having to annotate every test input. While at a high level the objective is similar, i.e., evaluating the performance of a classifier, this work assumes that the precision function is monotonic with respect to the classifier's confidence-based ranking of inputs. This assumption does not hold for UUs by definition.

In data mining, *outlier detection* identifies inputs that are anomalous with respect to a set of inputs (**?**; Eskin 2000). However, unlike our approach, this work assumes access to the training data.

Once a UU has been found, a user may want to better understand *why* the classifier made its prediction. Ribeiro *et al.* (2016) and others have proposed algorithms for generating explanations of black-box classifiers. In the future, we intend to integrate these with UU discovery.

Conclusions

Finding a classifier's unknown unknowns—its high-confidence false positives—is important for understanding the learner's limitations and useful for attacking the classifier, preventing such attacks, and deploying a high performing human-machine hybrid system (**?**). This task is especially timely given the growing popularity of pre-trained (and hence opaque) models, such as Microsoft Cognitive Services. This paper identifies three issues with Lakkaraju et al.'s approach to identifying UUs that cause it to produce an incomplete understanding of a classifier's blind spots. These include W1) the failure to reward discovery of novel UUs, W2) inability to prefer UUs that together characterize the extent of a systematic blind spot, and W3) not distinguishing between rare UUs and those in dense regions of feature space. We observed that their bandit-based approach was especially likely to produce a set of very similar UUs.

To overcome the weaknesses, we define a new, "coverage-based" utility model, which prefers a diverse set of UUs that together "explain" the classifier's blind spots relative to a test distribution. Unfortunately, our coverage-based model sacrifices independence and is, hence, difficult to optimize. However, the model is monotone submodular, which guarantees that an omniscient greedy algorithm with access to the location of the UUs can find a solution within a constant factor of the optimal. Since the location of the UUs is *not* known a priori, we implement versions of the greedy algorithm that use a probabilistic prior. Experimental results show that Greedy-conf outperforms both simple baselines and Lakkaraju et al.'s bandit-based approach on the coverage-based utility, and achieves within 60.9% of the omniscient, tractable upper bound.

We envision several avenues for future work. First, we intend to combine discovery with explanation, perhaps using a system such as LIME (Ribeiro, Singh, and Guestrin 2016). Second, our utility model assumes that discovering a non-UU results in zero utility; however, in practice, non-UUs may also help to define the boundary of a blind spot. Finally, it would be useful to develop strategies that can employ multiple representations and clustering algorithms to better isolate UUs.

Appendix

Proof of Proposition 2. Let Q_1 and Q_2 be two subsets of D_{test} where $Q_2 \subset Q_1$. Let s denote an input in D_{test} that is not in Q_1. Let $\Delta(s \mid Q)$ denote the increment in utility obtained as a result of querying s and when $Q \subset D_{\text{test}}$ has already been queried. In order to prove the utility model is submodular we need to show that $\Delta(s \mid Q_1) \leq \Delta(s \mid Q_2)$ for all $Q_1, Q_2 \subset D_{\text{test}}$.

In the case where s is not an UU $\Delta(s \mid Q) = 0$ for all $Q \subset D_{\text{test}}$. In this case $\Delta(s \mid Q_1) - \Delta(s \mid Q_2) = 0$. Now consider the case where s is an UU. Let $S_1 \subset Q_1$ and

$S_2 \subset Q_2$ denote the subset of queries that resulted in an UU.

$$
\begin{aligned}
&\Delta(s \mid Q_2) - \Delta(s \mid Q_1) \\
&= \sum_{x \in D_{\text{test}}} c_{M,x} \cdot ((P(\text{explain}_x \mid Q_2 \cup \{s\}, y_{Q_2 \cup \{s\}}) \\
&\quad - P(\text{explain}_x \mid Q_2, y_{Q_2})) \\
&\quad - (P(\text{explain}_x \mid Q_1 \cup \{s\}, y_{Q_1 \cup \{s\}}) \\
&\quad - P(\text{explain}_x \mid Q_1, y_{Q_1}))) \\
&= \sum_{x \in D_{\text{test}}} c_{M,x} \cdot ((\max_{s' \in S_2 \cup \{s\}} \text{sim}(x, s') - \max_{s' \in S_2} \text{sim}(x, s')) \\
&\quad - (\max_{s' \in S_1 \cup \{s\}} \text{sim}(x, s') - \max_{s' \in S_1} \text{sim}(x, s'))) \\
&\geq \sum_{x \in D_{\text{test}}} c_{M,x} \cdot (\max(0, \text{sim}(x, s) - \max_{s' \in S_1} \text{sim}(x, s')) \\
&\quad - \max(0, \text{sim}(x, s) - \max_{s' \in S_1} \text{sim}(x, s')) = 0
\end{aligned}
$$

The last inequality is obtained using the fact that $S_2 \subseteq S_1$.

Acknowledgements

We appreciate helpful comments and discussions with Himabindu Lakkaraju, Mausam, Jonathan Bragg, Eric Horvitz, Ece Kamar, Rao Kambhampati, Chris Lin, Jim Chen, Marco Tulio Riberio, Mandar Joshi, Eunsol Choi, Phoebe Mulcaire, Sandy Kaplan, and anonymous reviewers. This research was supported as part of the Future of Life Institute (futureoflife.org) FLI-RFP-AI1 program, grant 2015-144577 (5388) with additional support from NSF grant IIS-1420667, ONR grant N00014-15-1-2774, the WRF/Cable Professorship, and a gift from Google.

References

Amershi, S.; Chickering, M.; Drucker, S. M.; Lee, B.; Simard, P.; and Suh, J. 2015. ModelTracker: Redesigning performance analysis tools for machine learning. In *Proc. of CHI*.

Eskin, E. 2000. Anomaly detection over noisy data using learned probability distributions. In *Proc. of ICML*.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2015. Explaining and harnessing adversarial examples. In *Proc. of ICLR*.

Krause, A.; Singh, A.; and Guestrin, C. 2008. Near-optimal sensor placements in gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR* 9:235–284.

Lakkaraju, H.; Kamar, E.; Caruana, R.; and Horvitz, E. 2017. Identifying unknown unknowns in the open world: Representations and policies for guided exploration. In *Proc. of AAAI*.

McAuley, J.; Pandey, R.; and Leskovec, J. 2015. Inferring networks of substitutable and complementary products. In *Proc. of KDD*.

Pang, B., and Lee, L. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proc. of ACL*.

Pang, B., and Lee, L. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proc. of ACL*.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "Why should I trust you?": Explaining the predictions of any classifier. In *Proc. of KDD*.

Settles, B. 2012. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. *ArXiv*.

Werling, K.; Chaganty, A.; Liang, P.; and Manning, C. D. 2015. On-the-job learning with Bayesian decision theory. In *Proc. of NIPS*.