Facility Locations Utility for Uncovering Classifier Overconfidence

Karsten Maurer[1] – Walter Bennette[2]

Miami University Dept. of Statistics[1] – Air Force Research Lab[2]

Searching for Unknown Unknowns

Context: You have a classifier that you wish to apply to a domain in which you only have access to a large set of unlabeled test instances.

Problem: You know that the classifier was trained in a different domain, or you suspect a bias in the training set relative to the testing

Goal: Select a set of instances from the unlabeled set that we will look up true labels which we use to evaluate the classifier performance. This is called a *query* of the unlabeled set.

An *oracle* is the person who will find the true labels. We don't want to waste the oracle's time, so we need a smart way to query the unlabeled set.

What could we look for in the query set?
Attenberg et al. (2015) define *Unknown Unknowns* (UUs) as instances, x, from unlabeled test set where the classifier, $M(x)$, is both:
1. Highly confident. Above some threshold $\tau \in (0,1)$ Thus, $c_x \geq \tau$
2. Wrong. Misclassified with $y \neq M(x)$

Why would we care about finding UUs in query set?
High confidence mistakes lead to unmitigated risks in application of classifier
Characteristics of the UUs may help analyst to understand classifier deficiencies

Existing Utility-Based Query Algorithms

Attenberg et al. (2015) - Crowdsource
- "Beat the Machine" game with monetary rewards for finding UUs
- Crowdsourcing as mechanism for learning classifier deficiencies
- Basically like using many oracles with large budget for labelling

Drawback – Can't scale for budgeted oracle queries

Lakkaraju et al. (2017) - Multi-Armed Bandit
- Adaptive query algorithm that updates the optimal recommendation for next oracle query after evaluating the newly labeled instance
- Utility function adds unit value for each UU and penalizes by the cost of labeling

Drawback - Fundamentally place value on finding *any* UU, regardless of confidence

Bansal and Weld (2018) - Coverage-Based
Another greedy query algorithm, but uses coverage-based utility
Goal is to encourage both discovery of UUs and exploration of feature space

$$U(Q) = \sum_{x \in \boldsymbol{X}} c_x \max_{q \in S} \{sim(x, q)\}$$

$\boldsymbol{X} \subset \mathbb{R}^{\boldsymbol{p}}$ is p-dimensional unlabeled test set
$Q \subset \boldsymbol{X}$ is the set of points labeled by an oracle
$S = \{x \mid x \in Q, y_x \neq M(x)\}$ is the set of discovered UUs
c_x is the classifier's confidence in its prediction of x
$sim(x, q)$ is a distance-based similarity metric.

Drawback - The utility-based algorithms err toward instances that are "safe bets"; instances with confidence just above the τ threshold.

Adapting Utility Objectives to Overcome Deficiencies

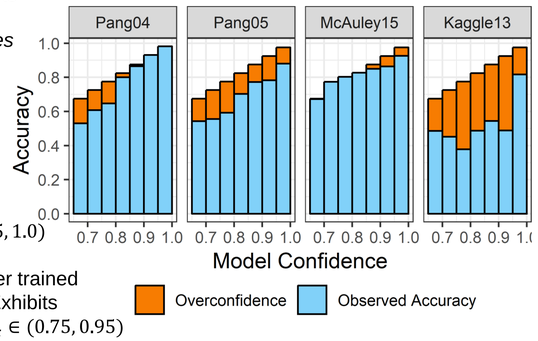
- The focus on discovering Unknown Unknowns, as defined in previous literature, is misdirected.
- High confidence should not be interpreted as an absolute. Misclassifications *should* be occurring.
- The problem is when the rate of misclassification exceeds the rate expected based on confidence.

Adapted Objectives for Query Based on Facility Locations Utility:
1. Our goal should be to query instances that demonstrate *classifier overconfidence*; containing more misclassification than should be expected based on classifier confidence.
2. Our query should also seek to thoroughly explore the feature space.

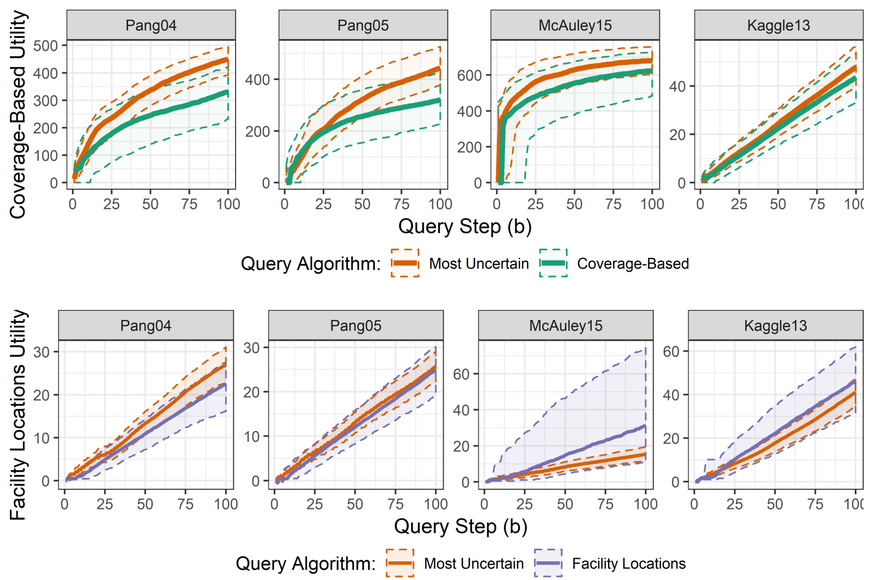
Empirical Experimentation

To compare query algorithms ability to identify overconfident points, we evaluate performance of methods on classifiers for four real datasets

- Pang04 and Pang05 – *Rotten Tomatoes* text classifiers intentionally biased toward subjective reviews. Exhibits strongest overconfidence when $c_x \in (0.65, 0.8)$
- McAuley15 – *Amazon* reviews text Classifier trained on electronic products, then applied to book reviews. Exhibits Strongest overconfidence when $c_x \in (0.85, 1.0)$
- Kaggle13 – Cat vs. Dog image classifier trained without dark fur cats, then applied to all. Exhibits Strongest overconfidence in points with $c_x \in (0.75, 0.95)$



Comparisons of Utility Relative to Baseline Query Algorithm



Facility Locations Search Algorithm

Facility Locations Utility Function

$$W(Q) = \sum_{q \in S} \log\left(\frac{1}{1 - c_q}\right) - \sum_{x \in \boldsymbol{X}} \min_{q \in S}(d(x, q))$$

$\log\left(\frac{1}{1-c_q}\right)$ is the *reward* function for finding a misclassification with confidence c_q
$d(x, q)$ is the Euclidean distance between points x and q.

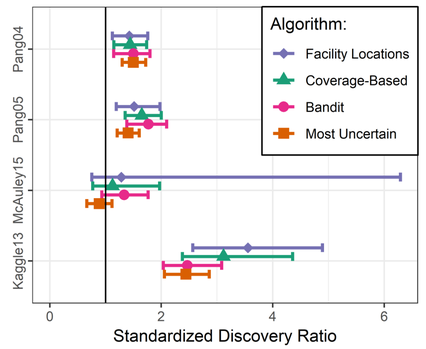
Algorithm for Greedy Facility Location Search
Input:
Test set $\boldsymbol{X}$, prior $\hat{\phi}(x|Q = \emptyset)$, budget B
$Q = \{\}$ inputs that have been queried, $y_Q = \{\}$ oracle defined labels
For: b = 1, 2, ..., B **do:**
$q' = \underset{q' \notin Q}{\operatorname{argmax}}\, E_{\hat{\phi}}[W(Q \cup q')]$
$Q \leftarrow Q \cup q'$
$y_Q \leftarrow y_Q \cup y_{q'}$
$S \leftarrow \{x \mid x \in Q, y_x \neq M(x)\}$
$\hat{\phi} \leftarrow \hat{\phi}(x|Q)$
$b \leftarrow b + 1$
Return: Q, S and y_Q

Efficiency of Discovering Overconfident Points

Standardized Discovery Ratio is a ratio of discovered misclassifications to the expected based on confidence

90% Monte Carlo central intervals and medians from 1000 random initializations of each algorithm

Results
- All methods discovering errors at about same rate as most uncertain for Pang04 and Pang05, where overconfidence is highest just above τ
- Facility Location search efficiently discovers mistakes relative to rate expected based on model confidence for McAuley15 and Kaggle13 datasets



Conclusions

- Facility locations query algorithm performs well across all overconfidence profiles seen in empirical evaluations
- SDR improves as budget increases, but this learning can be slow and inconsistent
- There is the need for an effective unsupervised query algorithm to initialize the utility-based searches

MIAMI UNIVERSITY