Discovering Blind Spots of Predictive Models: Representations and Policies for Guided Exploration

Himabindu Lakkaraju, Stanford University
himalv@cs.stanford.edu
Exciting Times
ML Applied to Critical Domains
Biases in ML

Google apologises for Photos app's racist blunder

7 hours ago | Technology

[Google Photos, y'all. My friend's not a gorilla.]

Mr Alcine tweeted Google about the fact its app had misclassified his photo

[Lakkaraju, Caruana, Horvitz; AAAI 2017]
Outline

- **Blind spots: Overview**
- **Problem Formulation**
- **Our Approach**
- **Experimental Results**
Focus: Detection of unknown unknowns

- **Unknown unknowns:** Instances with highly-confident but incorrect predictions

- **Blind-spots:** Feature subspaces with high concentration of unknown unknowns

- Unknown unknowns and blind spots occur due to a variety of reasons.
  - mismatch between training and execution data.
Common Assumption in ML

real-world concepts

M

cats

dogs

training data
Biases in Training Data

(real-world concepts)

$x = (f_1, ..., f_k)$

training data

wrong label
high confidence

M
Biases in Training Data

cats

dogs

M

(\textit{cat} (\textit{conf} = 0.96))
Discovery of unknown unknowns in the Wild

- **Goal**: Discover unknown unknowns
  - The predictive model is a black box
  - No access to the training data
- **Exploration space**: Execution data
- **Assumptions**
  - Unknown unknowns do not occur at random (Attenberg et. al., 2015)
  - There exist features in the data that can characterize unknown unknowns (No free lunch theorem)
Inputs

- A set of N instances $X = \{x_1, x_2, \ldots, x_N\}$ which were confidently assigned to a class of interest by the black box predictive model $M$ and the corresponding confidence scores $S = \{s_1, s_2, \ldots, s_N\}$
- An oracle $o$ which takes as input a datapoint $x$ and returns its true label $o(x)$ as well as the cost incurred to determine the true label $\text{cost}(x)$
- A budget, $B$, on the number of times the oracle can be queried
Problem Definition

Set $X$ of high confidence instances

$x(t) = (f_1, ..., f_k)$

Utility function: $u(x(t)) = 1_{\{o(x_t) \neq c\}} - \gamma \times \text{cost}(x(t))$

Problem statement: Find $\{x(1), x(2) \cdots x(B)\} \subseteq X$ s.t. $\sum_{t=1}^{B} u(x(t))$ is maximized.
Problem Definition

Set X of high confidence instances

\[ x(t) = (f_1, ..., f_k) \]

How to search the data space?
How to guide future discoveries with oracle feedback?
How to trade-off exploration with exploitation?
How to interpret regions of unknown unknowns?
Our Framework

**Input:**
Execution data points with high confidence

**Step 1:**
Descriptive Space Partitioning

- White Dogs
- White Cats
- Brown Cats
- Brown Dogs

**Step 2:**
Multi-armed bandits for unknown unknowns
Descriptive Space Partitioning

- Partition the instances such that those with similar feature values and confidence scores are grouped together.

- Each group must be associated with a descriptive pattern highlighting the characteristics of the instances in the group.
Descriptive Space Partitioning

- Obtain candidate patterns using frequent itemset mining algorithms [E.g., Apriori]
- Choose a set of patterns to ‘group’ the instances in the set $X$ such that:
  - Intra-group feature distance is minimized
  - Confidence scores of instances assigned to the same group are similar
  - Inter-group feature distances are maximized
  - Confidence scores of instances assigned to different group are dissimilar
Descriptive Space Partitioning

Intra-partition feature distance:
\[ g_1(q) = \sum_{\{x \in \mathcal{X} : x \in \text{covered by}(q)\}} d(x, \bar{x}_q) \]

Inter-partition feature distance:
\[ g_2(q) = \sum_{\{x \in \mathcal{X} : x \in \text{covered by}(q)\}} \sum_{q' \in Q : q' \neq q} d(x, \bar{x}_{q'}) \]

Intra-partition confidence score distance:
\[ g_3(q) = \sum_{\{s_i : x_i \in \mathcal{X} \land x_i \in \text{covered by}(q)\}} d'(s_i, \bar{s}_q) \]

Inter-partition confidence score distance:
\[ g_4(q) = \sum_{\{s_i : x_i \in \mathcal{X} \land x_i \in \text{covered by}(q)\}} \sum_{q' \in Q : q' \neq q} d'(s_i, \bar{s}_{q'}) \]

Pattern Length: \( g_5(q) = \text{size}(q) \)

**Input:** candidate pattern set \( Q = \{q_1, q_2, \ldots\}, X, S, \lambda \)

**Objective:** Find \( P \subseteq Q \) s.t.
\[
\min_{q \in Q} \sum_{q} f_q(\lambda_1 g_1(q) - \lambda_2 g_2(q) + \lambda_3 g_3(q) - \lambda_4 g_4(q) + \lambda_5 g_5(q))
\]
\[
\text{s.t.} \quad \sum_{q : x \in \text{covered by}(q)} f_q \geq 1 \quad \forall x \in \mathcal{X}, \text{ where } f_q \in \{0, 1\} \quad \forall q \in Q
\]

Reduction to weighted set cover \( \rightarrow \) NP-hard
\( \ln N \) approximation with greedy algorithm which picks at each step a pattern with maximum coverage-to-weight ratio
Bandits for Unknown Unknowns

- Each partition $\rightarrow$ an arm
- Pulling an arm $\rightarrow$ sampling a point without replacement
- Various stationary and non-stationary bandit algorithms
  - UCB1
  - Discounted UCB, Sliding window UCB
  - UUB

Partition selection at time $t$

$$\max_i \tilde{u}_t(i) + b_t(i)$$

(discounted) upper confidence bound

(discounted) mean reward

Step 1: space partitions
Multi-Armed Bandit Algorithms

- **UCB1:**
  - Mean reward $\bar{u}_t(i)$: Average reward obtained by pulling arm $i$ till time $t$.
  - Upper confidence bound $b_t(i): \sqrt{\frac{2 \ln N_t}{N_t(i)}}$

- **Sliding window UCB ($\tau$):**
  - Mean reward $\bar{u}_t(i)$: Average reward obtained by pulling arm $i$ over the past $\tau$ plays.
  - Upper confidence bound $b_t(i)$: Same as UCB1 except $N_t$ and $N_t(i)$ are computed over the past $\tau$ plays.
Multi-Armed Bandit Algorithms

- Discounted UCB ($\gamma$):
  - Mean reward $\bar{u}_t(i)$: Average discounted reward obtained by pulling arm $i$ till time $t$
    - reward at time $t - j$ is weighted by $\gamma^{t-j}$
  - Upper confidence bound $b_t(i)$: Similar to UCB1 except:
    - When computing $N_t$ and $N_t(i)$, pull at time $t - j$ is weighted by $\gamma^{t-j}$

Regret $= T \log(T)$
Multi-Armed Bandit Algorithms

■ Our algorithm – UUB:
  □ No need to set discounting factor

□ Mean reward $\bar{u}_t(i)$: Average discounted reward obtained by pulling arm $i$ till time $t$
  ■ reward at time $t - j$ is weighted by $\frac{\text{No. of instances in group } i \text{ at time } t}{\text{No. of instances in group } i \text{ at time } t - j}$

□ Upper confidence bound $b_t(i)$: Similar to UCB1 except:
  ■ When computing $N_t$ and $N_t(i)$, pull at time $t - j$ is weighted by the ratio above

Regret $= T \log(T)$
Experiments

- Sentiment Snippets
  - Bias: Missing subspaces of data
- Subjectivity dataset from Rotten Tomatoes
  - Bias: Missing subspaces of data
- Amazon Reviews
  - Bias: domain adaptation; train on electronics reviews and deploy on book reviews.
- Image Data
  - Bias: Missing subspaces of data; training data comprises of black dogs and non-black cats
Evaluation: Images Data

- Blind spots: non-black dogs, black cats
  - Blind spot: black cats
  - Blind spot: white dogs
  - Blind spot: white cats
Evaluation: Images Data

- Blind spots: non-black dogs, black cats

Blind spot: black cats

Blind spot: white dogs

Blind spot: white cats
Evaluation: Images Data

- Blind spots: non-black dogs, black cats

Blind spots: non-black dogs, black cats

Blind spot: black cats

Blind spot: white dogs

Exploration resources spent heavily on blind spots
Evaluating DSP

Lower entropy $\rightarrow$ Better separation of unknown unknowns
Evaluating Bandits

Lower regret $\rightarrow$ More effective discovery of unknown unknowns
Comparison with Alternative Methods

![Comparison with Alternative Methods](image-url)
From unknown unknowns to blind spots

- Interactively discovering blind spots:
  - The system designer can interactively decrease (or increase) reward for an arm

- Incentivizing diversity:
  - Reward of discovering similar unknown unknowns decreases with each additional discovery

- Our framework is *generic enough to adapt* to either of these extensions
Questions

himalv@cs.stanford.edu
Example