Meeting Discussions

1 Learner

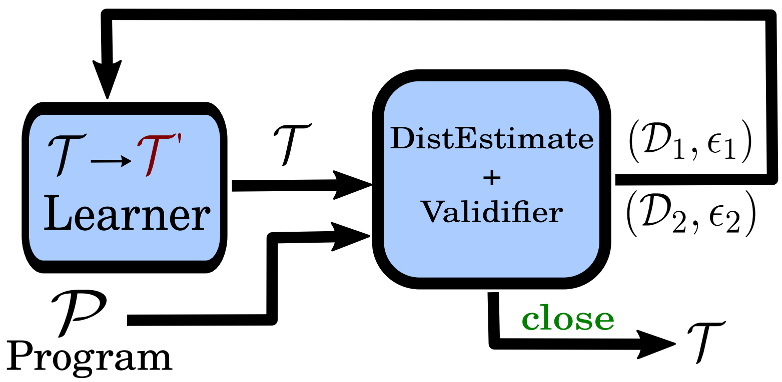


Fig. 1: Overview

The *DistEstimate* and *Validifier* modules will return the pairs ($\mathcal{D}_1$, ϵ_1) and ($\mathcal{D}_2$, ϵ_2) respectively (see fig. 1).

Assuming dataset $\mathcal{D}_1$ contains n_1 data points $\{(s_i, w_i, l_i)\}_{i=1}^{n_1}$, where s_i denotes a program state, w_i denotes its weight and $l_i \in \{0, 1\}$ is a label; l_i is 0 if the state $s_i \not\models \mathcal{T}$ and l_i is 1 if $s_i \models \mathcal{T}$. We also use l_i^{ex} to denote the expected labels of the states, for $s_i \in \mathcal{D}_1$, l_i^{ex} is 1.

Error Probability ϵ_1
Any random sampled state from S, must reach $\mathcal{T}$ with probability $(1 - \epsilon_1)$ in K-steps

Similarly, assume dataset $\mathcal{D}_2$ contains n_2 data points $\{(s_i, l_i)\}_{i=1}^{n_2}$, where s_i denotes a program state and $l_i \in \{0, 1\}$. In this case, $l_i = 0$ if s_i is not reachable from S otherwise $l_i = 1$. We also use l_i^{ex} to denote the expected labels of the states, for $s_i \in \mathcal{D}_2$, $l_i^{ex} = l_i$.

Error Probability ϵ_2
Any random sampled state from $\mathcal{T}$, must be reachable from S in K-steps with probability $(1 - \epsilon_2)$.

Cases to consider while mutating:

1. $\forall s_i \in \mathcal{D}_1$ with $l_i = 1$, we want to preserve the label of such states in $\mathcal{T}'$.
2. $\forall s_i \in \mathcal{D}_1$ with $l_i = 0$, we want to include these states in $\mathcal{T}'$ i.e. flip the labels of such states.
3. $\forall s_i \in \mathcal{D}_2$ with $l_i = 1$, we want to preserve the label of such states in $\mathcal{T}'$.
4. $\forall s_i \in \mathcal{D}_2$ with $l_i = 0$, we want to exclude such states from $\mathcal{T}'$.

Mutations. If $\mathcal{T}$ has x leaves then maximum possible mutations are $2 \cdot x$. For each leaf following two mutations are possible:

1. Splitting a leaf node.
2. Pruning a leaf node.

Cost of Mutations. Pruning a leaf node is preferred over splitting therefore cost of pruning should be less than the cost of splitting the node. For both types of mutations, mutating a leaf at higher depth is preferred than a leaf at lower depth therefore cost of applying any mutation on a leaf at higher depth is lesser than applying mutation on a leaf at lower depth.

2 Problem Statement

Given $\mathcal{T}$and the pairs ($\mathcal{D}_1$, ϵ_1) and ($\mathcal{D}_2$, ϵ_2), make minimal mutations to $\mathcal{T}$ to obtain a $\mathcal{T}'$ such that following two inequalities hold:

$$\sum_{s_i \in \mathcal{D}_1} (w_i \cdot (|{l_i}^{\mathcal{T}'} - l_i^{ex}|)) \leq u_1 \tag{1}$$

$$\frac{\sum_{s_i \in \mathcal{D}_2} (1 \cdot (|{l_i}^{\mathcal{T}'} - l_i^{ex}|))}{n_2} \leq u_2 \tag{2}$$

3 Algorithm

Given a candidate $\mathcal{T}$, pairs ($\mathcal{D}_1$, ϵ_1) and ($\mathcal{D}_2$, ϵ_2) and user supplied error bounds $u_1$and u_2, algorithm 3 mutates $\mathcal{T}$and outputs $\mathcal{T}'$ such that eq. (1) and eq. (2) are satisfied.

extractLeaves. Returns a mapping of states in $\mathcal{D}_1$ and $\mathcal{D}_2$ to leaves indices of the tree $\mathcal{T}$as a dictionary.

Algorithm 1 getGain($\mathcal{T}, l, m, \epsilon_1, \epsilon_2$)

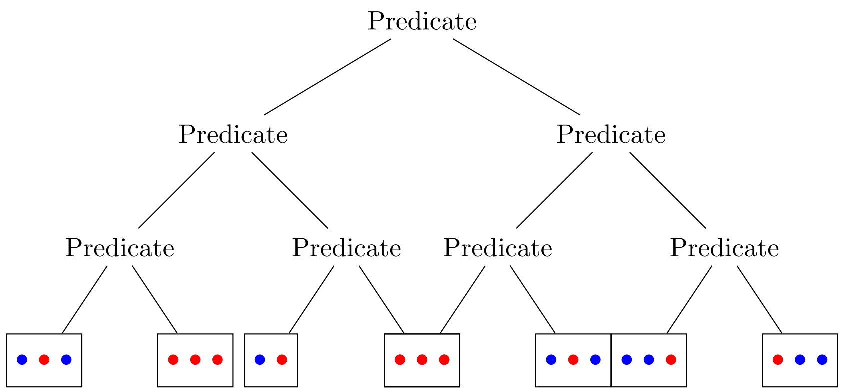
$\mathcal{T}' \leftarrow Mutate(\mathcal{T})$
$\epsilon_1', \epsilon_2' \leftarrow getEstimates(\mathcal{T}')$ //Using eq. (1) and eq. (2)
$gain \leftarrow (\epsilon_1 - \epsilon_1') + (\epsilon_2 - \epsilon_2')$
$return\ gain, \epsilon_1', \epsilon_2'$

Algorithm 2 getCost(dep, m)

$return\ 1$

Algorithm 3 MuteTree($\mathcal{T}$, $\mathcal{D}_1$, $\mathcal{D}_2$, ϵ_1, ϵ_2, u_1, u_2)

$\epsilon_1', \epsilon_2' \leftarrow 0, 0$
while $\epsilon_1' \leq u_1 \land \epsilon_2' \leq u_2$ **do**
    $moves, leaves \leftarrow [], \{\}$
    $leaves \leftarrow extractLeaves(\mathcal{T}, \mathcal{D}_2, \mathcal{D}_1)$
    for $l \in leaves$ **do**
        for $m \in mutations$ **do**
            $gain, \mathcal{T}', \epsilon_1', \epsilon_2' \leftarrow getGain(\mathcal{T}, l, m, \epsilon_1, \epsilon_2)$
            $cost \leftarrow getCost(depth(l), m)$
            $ratio \leftarrow \frac{gain}{cost}$
            $moves.append((ratio, \mathcal{T}', \epsilon_1', \epsilon_2'))$
    $moves \leftarrow sort(moves, key = ratio)[: k]$
    $moves \leftarrow normalize(moves, key = ratio)$
    $\mathcal{T}', \epsilon_1', \epsilon_2' \leftarrow SampleUniform(moves)$
$return\ \mathcal{T}'$


Predicate
Predicate
Predicate
Predicate
Predicate
Predicate
Predicate