Proceedings of the 3rd International Workshop on Social Influence Analysis (SocInf 2017)
August 19th, 2017 - Melbourne, Australia

Explanation Systems for Influence Maximization Algorithms

Amulya Yadav[1], Aida Rahmattalabi[1], Ece Kamar[2], Phebe Vayanos[1], Milind Tambe[1], Venil Loyd Noronha[1]

[1]Center for Artificial Intelligence in Society, University of Southern California, LA, CA, 90089
[2]Microsoft Research, Redmond, WA 98052
[1]{amulyaya, rahmatta, phebe.vayanos, tambe, vnoronha}@usc.edu
[2]{eckamar}@microsoft.com

Abstract. The field of influence maximization (IM) has made rapid advances, resulting in many sophisticated algorithms for identifying "influential" members in social networks. However, in order to engender trust in IM algorithms, the rationale behind their choice of "influential" nodes needs to be explained to its users. This is a challenging open problem that needs to be solved before these algorithms can be deployed on a large scale. This paper attempts to tackle this open problem via four major contributions: (i) we propose a general paradigm for designing explanation systems for IM algorithms by exploiting the tradeoff between explanation accuracy and interpretability; our paradigm treats IM algorithms as black boxes, and is flexible enough to be used with any algorithm; (ii) we utilize this paradigm to build XplainIM, a suite of explanation systems; (iii) we illustrate the usability of XplainIM by explaining solutions of HEALER (a recent IM algorithm) among $\sim$200 human subjects on Amazon Mechanical Turk (AMT); and (iv) we provide extensive evaluation of our AMT results, which shows the effectiveness of XplainIM.

1 Introduction

The field of influence maximization (IM) deals with finding a set of "influential" nodes in a social network, which can optimally spread influence throughout the network. Significant progress in this field over the last few years has led to a variety of sophisticated algorithms [1] for finding influential nodes. Recently, [10] evidenced the real-world usability of IM algorithms by deploying three such algorithms in the real world.

As with any technology, evidence of real-world usability of IM algorithms brings to the fore other important questions that need to be tackled before their widespread adoption. One of the most important open questions for IM algorithms (which are used as decision support tools by domain experts such as viral marketers, social workers, etc.) is how to engender trust by *explaining* their recommendations to domain experts [7]. This is important because explaining the rationale behind these algorithms decisions helps the domain experts decision making process: they can then choose to either use the

algorithm's recommendation or discard it. Indeed, [9] noted the challenge of explaining the solutions of HEALER (an IM algorithm) to homeless shelter officials, who wanted explanations for why HEALER's solutions were better than their preferred solutions. This raises the following important question: How to explain solutions generated by IM algorithms to its intended users? To the best of our knowledge, this problem has not received any attention in the literature so far.

Designing such an explanation system for IM algorithms is a challenging task for two reasons. First, instead of building different explanation systems for different IM algorithms, we want to build a single explanation system which is *generalizable*, i.e., it should be flexible enough to explain the solutions of every possible IM algorithm. This makes our task challenging because we can no longer rely on "*explaining the algorithm to explain the solution*" (since there is no fixed algorithm). Second, most human end-users have cognitive limitations and therefore, in order to avoid overloading them with information, we need to ensure that our system's explanations are relatively *interpretable*. However, this involves trading off explanation accuracy (i.e., correctness of explanation) with its interpretability (as we explain later).

This paper attempts to address this open problem via four major contributions. First, we propose a machine learning based paradigm for designing generalizable explanation systems for IM algorithms. Our paradigm achieves *generalizability* by using an algorithm independent classification dataset to build a machine learning classifier, which generates natural language explanations (in terms of this dataset's features) for IM solutions. In order to aid *interpretability* of the generated explanations, our classification dataset does not contain features for un-interpretable complicated terms (e.g., marginal gain). This transforms the problem of explaining IM algorithm solutions into a problem of explaining the predictions made by that machine learning classifier (as we explain later). Further, our paradigm exploits a fundamental tradeoff between explanation accuracy and interpretability (i.e., more accurate explanations are less interpretable, and vice versa) to generate a Pareto frontier of machine learning classifiers, which is then used to generate natural language explanations of varying accuracy and interpretability. Second, we utilize our paradigm to build XplainIM, a suite of explanation systems. Third, we illustrate the usability of XplainIM by explaining the solutions of HEALER (a recent IM algorithm) to $\sim$200 human subjects on Amazon Mechanical Turk (AMT). Fourth, we provide extensive evaluation of our AMT results, which shows the effectivenes of XplainIM.

Motivating IM Algorithm Our proposed explanation system is flexible enough to explain the solutions of any IM algorithm. However, we motivate our discussion in the rest of the paper by focusing on HEALER [9], a recent IM algorithm. We choose HEALER primarily because its paper [9] was the first to pose the question of explaining the solutions of an IM algorithm. HEALER was designed to select multiple sets of "influential" nodes sequentially from a homeless youth social network, who could optimally spread awareness about HIV in their social network. HEALER solves this problem by modeling it as a Partially Observable Markov Decision Process (POMDP) [5] and uses hierarchical ensembling techniques to scale up to real-world network sizes [9].

Thus, HEALER is a specialized algorithm for solving a more general problem than standard influence maximization (defined below). However, in order to ensure the generalizability of our explanation system to other IM algorithms, we ignore the POMDP style sequential planning aspects of HEALER, and use it as an algorithm for solving the standard single-shot influence maximization problem: *Given a social network graph G, select a set of K "influential" nodes which optimally spread influence in the network after T time steps*.

2 Related Work

The problem of explaining solutions of IM algorithms was first posed by [9] in the context of HEALER, but their main focus was on justifying the need for such explanations, as opposed to providing any practical solutions to this problem. While explanation systems for POMDPs have been proposed [8], they cannot be applied to explain solutions of every IM algorithm. To the best of our knowledge, no other previous work has looked at this problem so far.

3 Machine Learning based Paradigm for Explanation Systems

We now define an *explanation system* formally. Given a social network graph $\boldsymbol{G}$ as input, an IM algorithm outputs an optimal set of K nodes $\boldsymbol{O_G}$. Moreover, the end-user of the IM algorithm has a preferred set of K nodes $\boldsymbol{U_G}$, which he/she *wrongly* thinks is the optimal solution. Then, an explanation system is defined as follows:

Definition 1. *An explanation system for IM algorithms is a function $\boldsymbol{\mathcal{E}} : \boldsymbol{G} \times \boldsymbol{O_G} \times \boldsymbol{U_G} \rightarrow S$ which takes a graph $\boldsymbol{G}$, an IM algorithm's solution on $\boldsymbol{G}$ (i.e., $\boldsymbol{O_G}$) , and an end-user's preferred solution on $\boldsymbol{G}$ (i.e., $\boldsymbol{U_G}$) as input, and produces as output a natural language string S, which explains why $\boldsymbol{O_G}$ is a better solution than $\boldsymbol{U_G}$ on graph $\boldsymbol{G}$.*

We believe that Definition 1 accounts for most practical use cases of explanation systems. For example, [9] noted that homeless shelter officials (i.e., end-users) wanted explanations for why HEALER's solutions were better than their preferred solutions. Next, we explain our novel paradigm for designing explanation systems (for IM algorithms) in two stages.

First, we transform our original *problem of explaining IM solutions* into an alternate problem: "*how to explain predictions made by ML classifiers?*" Second, having made this transformation, we provide novel solutions for explaining predictions made by ML classifiers.

Step 1: Transformation Consider the following two problems for any given IM algorithm Δ:

Problem 1. Given as input an IM algorithm Δ, output an explanation system $\boldsymbol{\mathcal{E}}_{\Delta}$ which explains why Δ's solutions ($\boldsymbol{O_G^{\Delta}}$) are better than $\boldsymbol{U_G}$ (for all possible $\boldsymbol{G}$ and $\boldsymbol{U_G}$).

Problem 2. Given as input (i) an IM algorithm Δ, and (ii) a binary classifier M, which perfectly (i.e., without any misclassifications) differentiates Δ's solutions $\boldsymbol{O}_{\boldsymbol{G}}^{\boldsymbol{\Delta}}$ from all other possible solutions $\boldsymbol{U}_{\boldsymbol{G}}$ (for all possible $\boldsymbol{U}_{\boldsymbol{G}}$ and $\boldsymbol{G}$), output an explanation system $\boldsymbol{\mathcal{E}}_M$ which explains how classifier M differentiates $\boldsymbol{O}_{\boldsymbol{G}}^{\boldsymbol{\Delta}}$ from $\boldsymbol{U}_{\boldsymbol{G}}$.

We claim that solving Problem 2 is sufficient to solve Problem 1, as formally stated below (proof in appendix[1]):

Theorem 1. *Problem 2 solution* $\implies$ *Problem 1 solution*

As a result, we transform the *problem of explaining IM solutions* (Problem 1) into the alternate problem: "*how to explain predictions made by ML classifiers?*" (Problem 2). This completes the transformation step of our approach. Next, we provide a novel solution method for solving Problem 2.

Step 2: Solution Method First, we specify the *input to Problem 2*: an ML classifier M which can differentiate between $\boldsymbol{O}_{\boldsymbol{G}}$ and $\boldsymbol{U}_{\boldsymbol{G}}$. Second, we need to use this classifier M to generate *output for Problem 2*: the process of explaining the predictions made by classifier M needs to be specified. We focus on the input specification in this section (output generation is explained in Section 5).

To train classifier M, each possible IM solution (i.e., set of K nodes) is mapped to a set of network-centric feature values (e.g., degree, mutual connections, etc.), and is labeled as either being "*optimal*" (if it is the IM algorithm's solution) or "*sub-optimal*" (for any other solution). *These network-centric features generalize across all IM algorithms, as they only depend on the network based characteristics of any possible IM solution*. This creates a labeled dataset (details in Section 6) which is then used to train the ML classifier M which differentiates an optimal solution from a sub-optimal solution.

We use decision trees [6] as our ML classifier of choice. The key advantage of decision trees over other classifiers is their interpretability, as their predictions can be explained to end-users as a conjunction of decision rules, where each rule involves a single feature in the dataset. In many applications, including our own, this interpretability is preferred over other methods that may have higher accuracy but are relatively un-interpretable [9].

Unfortunately, while decision trees are more interpretable than other ML classifiers (e.g., neural nets), the explanations for their predictions (i.e., a conjunction of decision rules) can overload human users with too much information to process (as we show in our experiments). Thus, there is a need to ensure interpretability of the generated explanations.

We address this challenge by introducing "*interpretable decision trees*", which allows us to trade off the prediction accuracy of a decision tree with its interpretability in a quantifiable manner. Further, we exploit the competing objectives of prediction accuracy and interpretability to generate a Pareto frontier of decision trees. This Pareto frontier allows us to correctly differentiate between $\boldsymbol{O}_{\boldsymbol{G}}$ and $\boldsymbol{U}_{\boldsymbol{G}}$, while maintaining different levels of interpretability. We then utilize trees from this Pareto frontier to generate explanations for the IM algorithm's solutions (see Section 4). The novelty of our

[1] See http://bit.ly/2kXDhE4 for proofs

work lies in the versatility of our explanation system, which is able to provide explanations of varying interpretability upon demand. Specifically, our explanation system is flexible enough to dynamically increase the interpretability of the generated explanation (by using more "interpretable" decision trees to generate the explanation), if the end-user is unconvinced about previously supplied explanations. We now formally define interpretable decision trees.

4 Interpretable Decision Trees

Decision trees take an input a labeled dataset with N datapoints $(X^i, Y^i)\ \forall i \in \{1, N\}$, where X^i is a p-dimensional feature vector for the i^{th} datapoint, and Y^i is the label associated with the datapoint. In our domain, X^i represents the vector of network-centric feature values corresponding to the i^{th} IM solution (i.e., set of K nodes) in our dataset. Similarly, $Y^i = 1$ if the i^{th} solution in our dataset is the IM algorithm's solution, and it is 0 otherwise.

While a decision tree's prediction for X^i can be explained as a conjunction of all the feature splits that X^i traverses on its way to a leaf node, this explanation is uninterpretable, as it overloads end-users with too much information (we validate this in our experiments). Thus, in order to ensure interpretability of the generated explanations, we introduce the notion of *interpretable decision trees*, which allows us to tradeoff the explanation accuracy with its interpretability in a quantifiable manner. Next, we define the interpretability score, which quantifies the interpretability of a decision tree.

Interpretability Score We associate with each feature $t \in \{1, p\}$ an interpretability weight $w_t \in [0, 1]$ which quantifies the interpretability of feature t. We estimate w_t values $\forall t \in \{1, p\}$ by surveying AMT workers (see Section 6). Further, let $x_t^{\mathcal{D}}\ \forall t \in \{1, p\}$ denote a binary variable which indicates whether a split with feature t is used at any branch node in the decision tree $\mathcal{D}$ or not. Then, the interpretability score of decision tree $\mathcal{D}$ is defined as: $\mathcal{I}_{\mathcal{D}} = (\sum_{t=1}^{p} x_t^{\mathcal{D}} w_t)/(\sum_{t=1}^{p} x_t^{\mathcal{D}})$.

Thus, trees which use fewer number of features to split (the denominator in $\mathcal{I}_{\mathcal{D}}$'s definition) have higher $\mathcal{I}_{\mathcal{D}}$ scores (as it is easier to explain the tree's predictions with fewer features). Moreover, trees which use features having more interpretable weights have higher $\mathcal{I}_{\mathcal{D}}$ scores. Next, we define the A&I score of a decision tree, which captures the tradeoff between its prediction accuracy and interpretability.

A&I Score For decision trees (and most other classification algorithms), interpretability and prediction accuracy (measured by $\mathcal{I}_{\mathcal{D}}$ and F_1 score[2], respectively) are two competing objectives. Increasing the prediction accuracy of a decision tree leads to a loss of its interpretability, and vice versa.

The A&I score of decision tree $\mathcal{D}$ captures the tradeoff between these competing objectives and is defined as follows: $A\&I_{\mathcal{D}}(\alpha) = {F_1}^{\mathcal{D}} + \alpha\mathcal{I}_{\mathcal{D}}$. Here, α is a parameter which controls the tradeoff between the F_1 and $\mathcal{I}_{\mathcal{D}}$ scores. Higher α values lead to highly interpretable decision trees with low prediction accuracy, and vice versa. Next, we explain an algorithm which finds tree $\mathcal{D}_\alpha$ which maximizes the $A\&I(\alpha)$ score (i.e., it optimally trades off between F_1 and $\mathcal{I}_{\mathcal{D}}$ scores for the tradeoff level specified by α).

[2] F_1 score = $2TP/(2TP + FP + FN)$

Algorithm 1: Maximize A&I score

Input: α, Dataset $(X^i, Y^i)\ \forall i \in \{1, N\}$
Output: $\mathcal{D}_\alpha$, the A&I score maximizing decision tree

1 $(Train, Validation, Test) = Split(Dataset)$;
2 $Tree = CART(Train)$;
3 **for** $level \in PruningLevels(Tree)$ **do**
4 $\quad PrunedTree = Prune(Tree, level)$;
5 $\quad r_{level} = A\&I(\alpha, PrunedTree, Validation)$;
6 $OptLevel = argmax_j r_j$;
7 $\mathcal{D}_\alpha = Prune(Tree, OptLevel)$;
8 return $\mathcal{D}_\alpha$;

Maximizing A&I score For each α value, we use Algorithm 1 to build a decision tree (on the dataset from Section 3) which maximizes $A\&I(\alpha)$. First, we randomly split our dataset into a training, validation, and test set in Step 1(in our experiments, we use a 80:10:10 split). Next, we use CART, a state-of-the-art decision tree algorithm [4], to learn a decision tree on our training set in Step 2. Normally, CART outputs deep decision trees, which have high prediction accuracies on the training set. However, deeper trees have lower $\mathcal{I}_\mathcal{D}$ scores, as they have more number of features. As a result, we use our validation set to find the optimal pruning level of CART's decision tree, which maximizes $A\&I(\alpha)$. Specifically, for each possible pruning level , we prune CART's decision tree to that level in Step 4. We use this pruned decision tree's F_1 score on the validation set to calculate the pruned tree's $A\&I(\alpha)$ score on the validation set (Step 5). Finally, we select the optimal pruning level which maximizes $A\&I(\alpha)$ on the validation set (Step 6), and prune CART's decision tree to this level (Step 7). To account for randomization in splitting of the dataset into the training, validation and test set, we average over 50 such splits.

Decision Tree Pareto Frontier Instead of generating a single decision tree (using Algorithm 1), we exploit the accuracy-interpretability tradeoff by varying the value of α to generate multiple decision trees. We do this because of two reasons. First, end-users may find explanations generated from one decision tree hard to understand, and thus, we may need a more interpretable tree (measured by $\mathcal{I}_\mathcal{D}$) to generate the explanations. Second, explanations can only be generated using decision trees which correctly differentiate between the IM algorithm's optimal solution ($\boldsymbol{O_G}$) and the end-user's sub-optimal solution ($\boldsymbol{U_G}$) (else we don't have a valid classifier M in Problem 2). Thus, if more interpretable (less accurate) trees fail to differentiate between $\boldsymbol{O_G}$ and $\boldsymbol{U_G}$, we need more accurate (less interpretable) trees to generate explanations.

Therefore, for each α value, we find the optimal $\mathcal{D}_\alpha$ tree using Algorithm 1. Thus, we get a set of trees Θ, each of which has different F_1 and $\mathcal{I}_\mathcal{D}$ scores. Next, we remove all trees $\mathcal{D} \in \Theta$ which are dominated by other trees $\mathcal{D}'$ in terms of both F_1 and $\mathcal{I}_\mathcal{D}$ scores (i.e., there exist trees $\mathcal{D}' \in \Theta$ for which either $\{F_1^{\mathcal{D}'} \geqslant F_1^{\mathcal{D}} \wedge \mathcal{I}_{\mathcal{D}'} > \mathcal{I}_\mathcal{D}\}$ or $\{F_1^{\mathcal{D}'} > F_1^{\mathcal{D}} \wedge \mathcal{I}_{\mathcal{D}'} \geqslant \mathcal{I}_\mathcal{D}\}$ holds). This gives us our Pareto frontier of trees.

Our strategy is to search this Pareto-frontier for a decision tree $\mathcal{D}^*$, which can correctly differentiate between $\boldsymbol{O_G}$ and $\boldsymbol{U_G}$, and then use $\mathcal{D}^*$ to generate explanations.

Next, we complete the discussion of our explanation system by proposing XplainIM, a solution method for Problem 2 which utilizes our Pareto frontier (*recall that solving Problem 2 is sufficient to create an explanation system for IM algorithms*).

5 Solution for Problem 2: XplainIM

Problem 2 takes as input a machine learning classifier M (which can differentiate between $\boldsymbol{O_G}$ and $\boldsymbol{U_G}$), and produces as output an explanation system which explains how classifier M differentiates $\boldsymbol{O_G}$ from $\boldsymbol{U_G}$. We propose XplainIM as a solver for Problem 2. We begin by discussing XplainIM's design.

XplainIM builds an explanation system for a specific IM algorithm in two stages. In the offline stage, it takes as input a classification dataset for that algorithm, and uses it to store a Pareto-frontier of decision trees. This input dataset is created using a fixed set of network-centric features (Figure 1), irrespective of the specific IM algorithm under consideration. In the online stage, it takes as input a network G and user solution U_G and uses them to choose classifier M. We now explain how XplainIM chooses this classifier M.

Choosing Classifier Given a choice of $\boldsymbol{O_G}$ and $\boldsymbol{U_G}$, XplainIM finds the set $\mathcal{F}$ of "*feasible trees*" in its Pareto frontier which can successfully classify $\boldsymbol{O_G}$ as optimal, and $\boldsymbol{U_G}$ as sub-optimal. Each tree in $\mathcal{F}$ is a valid classifier for Problem 2, as it can differentiate between $\boldsymbol{O_G}$ and $\boldsymbol{U_G}$. To choose M, XplainIM simply uses the most interpretable tree in $\mathcal{F}$, i.e., $M = argmax_{\mathcal{D} \in \mathcal{F}} \mathcal{I}_{\mathcal{D}}$.

Note that it is possible that for some choice of $\boldsymbol{O_G}$ and $\boldsymbol{U_G}$, no tree in our Pareto frontier is *feasible*. Then, solving Problem 2 is impossible, as we wouldn't have an appropriate classifier M. However, this case rarely arises in practice (see Section 6).

Explanation System Next, having chosen a feasible decision tree as the classifier M, we now discuss XplainIM's explanation system, which generates explanations for "*how M differentiates $\boldsymbol{O_G}$ from $\boldsymbol{U_G}$*". Xplain relies on a natural language template (described below) to generate explanations, which is populated at runtime (depending on $\boldsymbol{O_G}$ and $\boldsymbol{U_G}$).

Template 1 *Your solution $\boldsymbol{U_G}$ had a f_t feature value of $\boldsymbol{U_G}(f_t)$. On the other hand, the optimal solution $\boldsymbol{O_G}$ had a f_t feature value of $\boldsymbol{O_G}(f_t)$. For solutions to be optimal, the value of f_t feature should be $\{\leqslant, >\} b_t$.*

Given $\boldsymbol{O_G}$ and $\boldsymbol{U_G}$, we propose three different methods for populating Template 1. Each of these methods relies on the observation that the paths taken by $\boldsymbol{O_G}$ and $\boldsymbol{U_G}$ (to leaf nodes) in tree M diverge at some branch node $t^* \in T_B$ (because tree M is feasible). We use features f_t and corresponding thresholds b_t (recall Section 4) starting from this point of divergence t^* to populate Template 1 to generate explanations for the end-user. Our three methods are explained below:

1. XplainIM-A Template 1 is populated using only f_{t^*} and b_{t^*} for branch node t^*, which is the first point of divergence between paths of $\boldsymbol{O_G}$ and $\boldsymbol{U_G}$ in tree M. This populated template is presented as explanation to the end-user.

2. XplainIM-B Template 1 is populated using f_{t^*} and b_{t^*} for branch node t^* (similar to XplainIM-A). However, the choice of classifier M used to generate explanations

is different. Instead of using the feasible tree with the highest $\mathcal{I}_{\mathcal{D}}$ score as our classifier M, we use the feasible tree in which the feature f_{t^*} (present at the branch node t^*) has the highest interpretability weight $w_{f_{t^*}}$. This is because only feature f_{t^*} is used to populate Template 1, and a high value of $w_{f_{t^*}}$ would ensure the interpretability of the generated explanation.

3. XplainIM-Full Instead of providing explanations in terms of a single feature, XplainIM-Full uses multiple features in its explanation. Starting from branch node t^*, XplainIM-Full populates Template 1 for all features along the two divergent paths of $\boldsymbol{O_G}$ and $\boldsymbol{U_G}$. This method produces explanations which are identical in structure to ones generated by decision sets [2].

6 Evaluation

We evaluated XplainIM by using it to explain HEALER's solutions to human workers on AMT. We provide results from these experiments in three parts. First, we describe the HEALER specific dataset (which XplainIM uses to build its Pareto-frontier of trees). Second, we provide results from AMT surveys which were used to estimate interpretability weights w_t for features in our dataset. Third, we provide AMT game results to evaluate XplainIM's effectiveness in explaining HEALER's solutions to AMT workers.

Dataset We generate 100 different Watts-Strogatz (WS) networks ($p = 0.25, k = 7$), each of size 20 nodes. The influence probabilities for network edges were randomly sampled from a uniform distribution $U[0, 1]$. We set $K = 2$, i.e., every set of two network nodes is a possible IM solution. For each of these 100 networks, we add HEALER's solution, along with 20 randomly chosen IM solutions to our dataset with the labels "*optimal*" and "*suboptimal*", respectively. Next, we map the i^{th} IM solution in our dataset to a unique feature vector X^i containing the network-centric feature values (see Figure 1) for that IM solution. This gives us a labeled dataset (with 20 features and 2100 datapoints), which serves as XplainIM's input in our experiments.

Feature Name	w_t
Betweenness Centrality	0.0
Clustering Coefficient	0.25
Pagerank	0.28
Closeness Centrality	0.71
Degree	1.0

Fig. 1: Interpretability Weights for Features

Interpretability Weights We deployed a survey among 35 AMT workers to estimate interpretability weights w_t for all 20 features in our dataset. Each worker is (i) provided the definition of each feature; and (ii) given an example of how that feature's

value is calculated for a sample IM solution. Then, the worker is asked to calculate each feature's value for a new IM solution on an unseen network.

For each feature t, we compare the user response distribution M_t (i.e., histogram of feature values calculated by the 21 workers who passed the validation game) with the actual value distribution P_t in order to calculate w_t. We assume that a feature's interpretability correlates heavily with ease of calculating that feature's value. Thus, if M_t is very similar to P_t, it implies that AMT workers find feature t to be *highly interpretable*. However, significant differences between M_t and P_t implies that feature t is *uninterpretable*. Specifically, we set $w_t = 1 - \text{EarthMover}(M_t, P_t)$, where $\text{EarthMover}(M_t, P_t)$ measures the distance between the distributions M_t and P_t [3].

Due to a lack of space, Figure 1 shows normalized w_t values for 5 out of the 20 features. This figure shows that "Degree" and "Betweenness Centrality" are the most and least interpretable features, respectively.

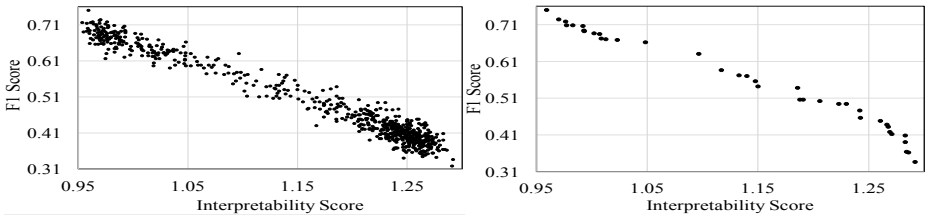
AMT Game Results We now present results from two different game types that we deployed on AMT to evaluate XplainIM's effectiveness at explaining HEALER's solutions. Both games begin by asking AMT workers to pick their preferred IM solution $\boldsymbol{U_G}$ on a randomly chosen WS network ($p = 0.25, k = 7, n = 20$). We begin by explaining the design and motivations of both games.

Game 1 This game has three different versions, one for each variant of XplainIM. This game presents explanations generated by XplainIM to AMT workers, and evaluates the effect of these explanations in improving the worker's IM solutions $\boldsymbol{U_G}$. The game proceeds as follows. After the worker picks solution $\boldsymbol{U_G}$, he/she is given an explanation for why HEALER's solution $\boldsymbol{O_G}$ is better than $\boldsymbol{U_G}$ (this explanation is generated by XplainIM-{A, B or Full} depending on the game version). Then, the worker is asked to improve his solution $\boldsymbol{U_G}$ based on the explanation shown.

Game 2 This game presents explanations generated by all three XplainIM variants to the same AMT worker, and compares the self-assessed interpretabilities of these three explanation schemes. The game proceeds as follows. After the worker picks solution $\boldsymbol{U_G}$, he/she is given three different explanations (generated using XplainIM-{A, B & Full}) for why HEALER's solution $\boldsymbol{O_G}$ is better than $\boldsymbol{U_G}$. Then, the worker is asked to rate these three explanations (each on a scale of 1-10) in terms of their interpretability. Further, we also deploy a "*No Explanation*" game (as a baseline), where workers are asked to improve their suboptimal solutions $\boldsymbol{U_G}$, without being given any explanation for $\boldsymbol{U_G}$'s suboptimality. Each of these game versions was deployed with 50 unique AMT workers (to avoid learning bias), each of whom plays on five different WS networks sequentially. We now discuss evaluation metrics used in this section.

Evaluation Metrics For Game 1, we compare the interpretability of XplainIM-A, B & Full's explanations by measuring the effect of these explanations in improving the worker's IM solutions $\boldsymbol{U_G}$. Specifically, we measure the percentage of workers who (in response to the provided explanation) modified $\boldsymbol{U_G}$ to either (i) match $\boldsymbol{O_G}$ (*Optimal group*); OR (i) correctly followed the provided explanation (*Correct group*); OR (iii) moved in the right direction (*Close group*).

For example, assume that "*Degree*" (Figure 1) of $\boldsymbol{U_G}$ is 4, and the provided explanation is "*For solutions to be optimal, the Degree should be greater than 9*". If the worker modifies $\boldsymbol{U_G}$ to match $\boldsymbol{O_G}$, he/she is counted in the *Optimal group*. Otherwise,



(a) Set of all decision trees (b) Trees forming Pareto frontier

Fig. 2: Plotting the Pareto Frontier of Decision Trees

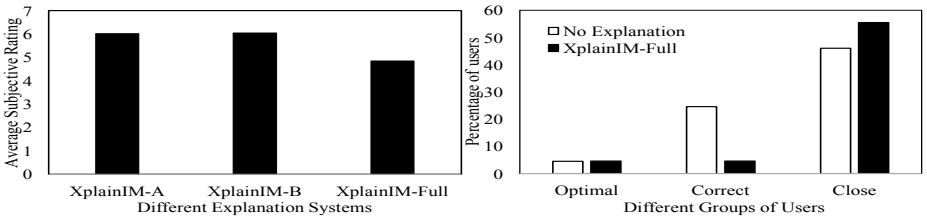
if his/her modified solution's Degree is greater than 9, he/she is counted in the *Correct group*. Finally, if his/her modified solution's Degree is greater than 4 ($\boldsymbol{U_G}$'s original Degree), but is less than 9, then he/she moved $\boldsymbol{U_G}$ in the right direction and is counted in the *Close group*.

Also, we use the *failure rate*, which is defined as the fraction of times an explanation system fails to produce an explanation. This happens when there are no *feasible decision trees* for a given choice of $\boldsymbol{O_G}$ and $\boldsymbol{U_G}$. We calculate an explanation system's failure rate on network G by measuring the fraction of possible user solutions $\boldsymbol{U_G}$ for which no tree is feasible in the Pareto frontier. We now present results comparing XplainIM-A, B & Full with the "*No Explanation*" treatment. All our comparison results are statistically significant under t-test.

Pareto Frontier First, we illustrate the accuracy-interpretability tradeoff exploited by all XplainIM variants to find Pareto-optimal trees. Figure 2a shows all decision trees (depicted by individual points) obtained by varying α values. The X and Y axes show the $\mathcal{I}_\mathcal{D}$ and F_1 scores, respectively. We prune this set of decision trees to find the set of Pareto-optimal trees (shown in Figure 2b). These figures clearly depict the inverse correlation between the two competing objectives of prediction accuracy (F_1) and interpretability ($\mathcal{I}_\mathcal{D}$).

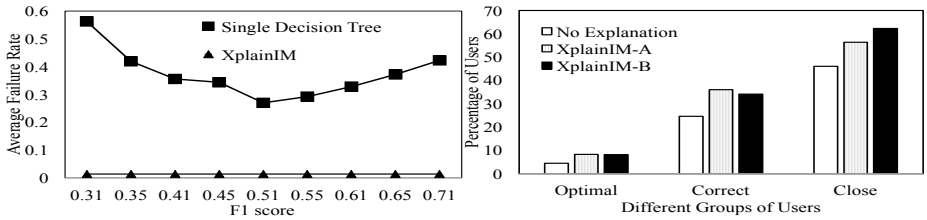
Next, we validate our decision of maintaining this Pareto frontier of trees inside XplainIM (instead of maintaining just a single Pareto-optimal tree). Figure 4a compares the failure rate of XplainIM-A, B & Full to that of single Pareto-optimal trees. The X-axis shows trees of increasing prediction accuracy. The Y-axis shows the average failure rate (across 100 WS nets). This figure shows that our Pareto-frontier based explanation systems (XplainIM-A, B & Full) have an average failure rate of 0.01, i.e., XplainIM generate valid explanations in $\sim$99% of situations. However, the minimum failure rate achieved by a single Pareto-optimal tree is 0.27. This shows that using single trees as our classifier for Problem 2 (instead of maintaining a Pareto frontier) results in failures 27% of the times. This signifies the importance of the Pareto frontier in making XplainIM work.

XplainIM Results Figure 3a shows Game 2 results. The X-axis shows the three XplainIM variants. The Y-axis shows average ratings (scale of 1-10) given by workers to different explanation schemes. This figure shows that XplainIM-A & B's explanations average rating was 6.04, as compared to 4.84 for XplainIM-Full (p=0.034). This shows that workers found XplainIM-A & B's explanations to be more interpretable than those generated by XplainIM-Full.



(a) Subjective rating comparison (b) Effect on modified solutions

Fig. 3: Evaluating interpretability of XplainIM-Full



(a) Failure rate comparison (b) Effect on modified solutions

Fig. 4: Frontier Validation & Evaluating XplainIM-A&B

Figure 3b shows Game 1 results for XplainIM-Full and the "*No Explanation*" treatment. The X-axis shows the *Optimal*, *Correct*, and *Close* groups. The Y-axis shows the percentage of workers belonging to these groups. This figure shows that only $\sim$4.6% workers are in the Correct group after seeing XplainIM- Fulls explanations. Surprisingly, even without any explanations, 24.6% workers are in the Correct group (p=$3e^{-9}$), which shows that users perform better without any explanations (compared to XplainIM-Full). This shows that XplainIM-Full's longer explanations (which are based on [2]) are completely uninterpretable to AMT workers (possibly because they overload workers with too much information). As a result, we now focus our attention on XplainIM-A & B.

Figure 4b shows Game 1 results for XplainIM-A & B and the "*No Explanation*" treatment. This figure shows that with XplainIM-A & B's explanations, $\sim$8.5% users are in the Optimal group ($\sim$4% with no explanation), 36% are in the Correct group (24% with no explanation), and $\sim$60% are in the Close group (46% with no explanation) (p=0.05). This establishes that XplainIM-A & B's explanations are far more interpretable, as significantly more AMT workers follow these explanations to improve their solutions. Further, this shows the importance of exploiting the accuracy-interpretability tradeoff (used by XplainIM) in generating explanations to solutions of IM algorithms.

7 Conclusion

This paper solves the open problem of explaining solutions of IM algorithms to its users. This paper shows that the problem of designing an explanation system can be

equivalently viewed as a problem of explaining an ML classifier's predictions. We propose XplainIM, a suite of ML based explanation systems which exploit the accuracy-interpretability tradeoff to generate natural language explanations. Extensive human subject experiments show the effectiveness of XplainIM in explaining IM algorithm solutions to human users.

8 Acknowledgements

This research was supported by MURI Grant W911NF-11-1-0332.

References

1. Borgs, C., Brautbar, M., Chayes, J., Lucier, B.: Maximizing Social Influence in Nearly Optimal Time. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms. pp. 946–957. SODA '14, SIAM (2014)
2. Lakkaraju, H., Bach, S.H., Leskovec, J.: Interpretable decision sets: A joint framework for description and prediction. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 1675–1684. ACM (2016)
3. Ling, H., Okada, K.: An efficient earth mover's distance algorithm for robust histogram comparison. IEEE transactions on pattern analysis and machine intelligence 29(5) (2007)
4. Loh, W.Y.: Classification and regression trees. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery 1(1), 14–23 (2011)
5. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons (2009)
6. Quinlan, J.R.: Induction of decision trees. Machine learning 1(1), 81–106 (1986)
7. Schneier, B.: Trust in man/machine security systems. IEEE Security Privacy 11(5), 96–96 (Sept 2013)
8. Wang, N., Pynadath, D.V., Hill, S.G.: The impact of pomdp-generated explanations on trust and performance in human-robot teams. In: Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems. pp. 997–1005. International Foundation for Autonomous Agents and Multiagent Systems (2016)
9. Yadav, A., Chan, H., Jiang, A.X., Rice, E., Kamar, E., Grosz, B., Tambe, M.: POMDPs for Assisting Homeless Shelters: Computational and Deployment Challenges. In: International Workshop on Issues with Deployment of Emerging Agent-based Systems (IDEAS) (2016)
10. Yadav, A., Wilder, B., Petering, R., Rice, E., Tambe, M.: Influence maximization in the field: The arduous journey from emerging to deployed application. In: In Proceedings of the International Conference on Autonomous Agents and Multi-agent Systems (AAMAS), 2017 (2017)