On Classification of Strategic Agents Who Can Both Game and Improve

Saba Ahmadi
Toyota Technological Institute at Chicago, IL, USA

Hedyeh Beyhaghi
Carnegie Mellon University, Pittsburgh, PA, USA

Avrim Blum
Toyota Technological Institute at Chicago, IL, USA

Keziah Naggita
Toyota Technological Institute at Chicago, IL, USA

Abstract

In this work, we consider classification of agents who can both game and improve. For example, people wishing to get a loan may be able to take some actions that increase their perceived credit-worthiness and others that also increase their true credit-worthiness. A decision-maker would like to define a classification rule with few false-positives (does not give out many bad loans) while yielding many true positives (giving out many good loans), which includes encouraging agents to improve to become true positives if possible. We consider two models for this problem, a general discrete model and a linear model, and prove algorithmic, learning, and hardness results for each.

For the general discrete model, we give an efficient algorithm for the problem of maximizing the number of true positives subject to no false positives, and show how to extend this to a partial-information learning setting. We also show hardness for the problem of maximizing the number of true positives subject to a nonzero bound on the number of false positives, and that this hardness holds even for a finite-point version of our linear model. We also show that maximizing the number of true positives subject to no false positive is NP-hard in our full linear model. We additionally provide an algorithm that determines whether there exists a linear classifier that classifies all agents accurately and causes all improvable agents to become qualified, and give additional results for low-dimensional data.

2012 ACM Subject Classification Theory of computation → Algorithmic mechanism design; Theory of computation → Sample complexity and generalization bounds

Keywords and phrases Strategic Classification, Social Welfare, Learning

Digital Object Identifier 10.4230/LIPIcs.FORC.2022.3

Related Version *Extended Version*: `https://arxiv.org/pdf/2203.00124.pdf` [2]

Funding This work was supported in part by the National Science Foundation under grants CCF-1733556 and CCF-1815011 and by the Simons Foundation under the Simons Collaboration on the Theory of Algorithmic Fairness.

1 Introduction

Consider a bank offering loans. Based on observable information about applicants, it must decide which of them are loan-worthy and which are not. For example, it might compute a credit score based on some (perhaps linear) function of observable features and then compare the result to a cutoff value. So far, this looks like a standard binary classification problem. However, there is an additional wrinkle: individuals have agency and may be able to modify their observable features somewhat if it will help them get approved for a loan. This wrinkle brings both challenges and opportunities. A challenge is that some of these actions may involve “gaming” the system: performing activities that do not affect their true loan-worthiness such as changing how they spend on different credit cards. An opportunity is that other actions, such as taking a money-management course, may truly help them become more loan-worthy, increasing the number of good loans the bank can give out. How can the bank best set its loan criteria in such settings to maximize the number of loans given out subject to not giving loans to unqualified applicants?

Or, consider a school that would like to prepare students for the workforce. There are many different career paths a student might take, so the school would like to have multiple different criteria for graduation (multiple tracks or majors) such that satisfying any one of them will earn the student a diploma. Imagine there is a limited set of options the school can choose from, and once the school chooses some subset of them as criteria, every student selects the easiest of those criteria to fulfill (or none, if all are too hard) and then may or may not become truly qualified for the workforce, depending perhaps on the extent to which satisfying that criterion involved gaming versus true improvement. How can the school best select criteria to maximize the number of students who become truly qualified for the workforce while minimizing the number of diplomas given to unqualified students?

In this work we consider algorithmic and learning-theoretic formulations of such scenarios, where a binary classification must be made in the presence of both gaming and improvement actions with a goal of maximizing true-positive predictions while keeping false-positives to a minimum. Specifically, we consider the following two formulations (given in more detail in Section 2).

General Discrete Model: In this formulation, we are given a weighted, colored bipartite graph with n nodes on the left representing agents, and m nodes on the right representing distinct possible ways agents could be considered *qualified* for the prize at hand (the loan, the diploma, etc.). For example, the nodes on the right could represent different possible definitions of “credit-worthy” or could represent different bundles of activities sufficient to receive a diploma. Each edge has both a *weight* representing the amount of effort the agent would need to achieve the given qualification and a *color* blue or red indicating whether the agent would indeed be truly qualified or not (respectively) if it did so. The goal of the classifier is to select a subset $\mathcal{P}^{\text{final}}$ of points on the right such that if each agent in the neighborhood of $\mathcal{P}^{\text{final}}$ takes its least-cost edge into $\mathcal{P}^{\text{final}}$, then a large number of blue edges and very few red edges are taken (many good loans and few bad loans are given out); more specific objectives will be detailed in Section 3.
In the learning-theoretic version of this problem, the left-hand-side of the graph is replaced with a probability distribution $\mathcal{D}$ over nodes (where a node is given by its neighborhood and the weights and colors of its edges). We have sampling access to $\mathcal{D}$ and our goal is to find a subset $\mathcal{P}^{\text{final}}$ of points on the right-hand-side with good performance under $\mathcal{D}$. In a partial-information version, when we sample a point from $\mathcal{D}$ we do not get to observe its edges, only where the agent goes to and whether it was qualified. That is, learning proceeds in rounds, where in each round we choose a subset $\mathcal{P}'$ of points on the right, and then for a random draw $x \sim \mathcal{D}$ we observe what point $p \in \mathcal{P}'$ (if any) was selected and the color of the edge taken.

Linear Model: In this formulation, we assume agents are points $\mathbf{x} \in \mathbb{R}^d$ (they have d real-valued features) and there is a linear separator $f^* : \mathbf{a}^*\mathbf{x} \geq b^*$ with non-negative weights that separates the truly qualified individuals from the unqualified ones. Agents have the ability to increase their jth feature at cost $\mathbf{c}[j]$ (decreasing is free) and receive value 1 for being classified as positive. However, only some features correspond to true improvement and others involve just gaming. That is, if an agent begins at $\mathbf{x}^{\text{init}}$ and moves to a point $\mathbf{x}^{\text{perc}}$, their true qualification is not $f^*(\mathbf{x}^{\text{perc}})$ but rather $f^*(\mathbf{x}^{\text{true}})$, where $\mathbf{x}^{\text{true}}$ agrees with $\mathbf{x}^{\text{init}}$ in the gaming directions and with $\mathbf{x}^{\text{perc}}$ in the improvement directions. Movement costs and which features are improvement versus gaming are assumed to be the same for all agents. The goal is to find a classifier that produces a large number of true positives and few false positives. Note that using f^* itself will be optimal if the coordinate j maximizing $\mathbf{a}^*[j]/\mathbf{c}[j]$ (having the most "bang per buck") is an improvement direction, so the interesting case is when this is a gaming direction. Also note that shifting f^* in this direction (adding $\mathbf{a}^*[j]/\mathbf{c}[j]$ to b^*) will be a perfect classifier but may not be optimal because it does not take advantage of the ability to encourage agents to improve. We consider settings where (a) the mechanism designer must use a linear classifier, (b) arbitrary classifiers are allowed, and (c) a polynomial-sized set $\mathcal{P}$ of "target points" is given and the mechanism designer must select some subset $\mathcal{P}^{\text{final}} \subseteq \mathcal{P}$ as its classifier – this is a special case of our General Discrete Model.

In this work, we consider both models. We give an efficient algorithm for the general discrete model for the problem of maximizing the number of blue edges taken subject to no red edges taken (maximizing the number of good loans given out subject to no bad loans) and show how to extend this to the partial-information learning setting. We also show hardness for the problem of maximizing the number of blue edges subject to a nonzero bound on the number of red edges, and show that this hardness holds even for the simplest finite-point linear model. Furthermore, we show the problem of maximizing the number of true positives subject to no false positives is NP-hard in the linear model when we are not given a polynomial-sized set of target points. We additionally give algorithms for the linear model. We provide an algorithm that determines whether there exists a linear classifier which classifies all agents accurately and causes all improvable agents to become qualified. In the special two-dimensional case, we design a linear classifier maximizing the number of true positives minus false positives; and a general (not necessarily linear) classifier that maximizes true positives subject to no false positives.

1.1 Related Work

There is an exciting and growing literature on decision-making in the presence of strategic agents. Much of this work considers agents whose actions are only gaming and do not change their true label (see [11, 7, 13, 16, 1, 6, 9, 5] among others) but researchers have also been investigating mechanism design in the presence of agents who can both game and improve [14, 12, 3, 18, 15, 10, 4, 17].

Kleinberg and Raghavan [14] consider a single agent with a variety of gaming and improvement actions available, that are then converted into observable features through an effort-conversion matrix. They then examine mechanisms for incentivizing desired action vectors, showing among other things that any vector that can be incentivized by a monotone mechanism can also be incentivized by a linear mechanism. Harris et al. [12] consider a multi-round version of the Kleinberg and Raghavan [14] model in which true improvements carry over to future rounds whereas gaming effort do not; they show that in this model, the principal (the decision-maker) can incentivize the agent to produce a greater range of desirable behaviors.

Alon et al. [3] consider a multi-agent extension of the Kleinberg and Raghavan [14] model, where agents all begin at the same place (the origin) but each have their own effort-conversion matrix. The goal of the designer is to choose an evaluation mechanism – mapping observable features to payoffs – that encourages all agents to take *admissible* actions, assuming that agents will maximize payoff subject to budget constraints. They specifically consider the case (1) that there is a single admissible action vector, and (2) that individual actions are either improvement or gaming actions and no agent should take a gaming action. Among other results they show that unlike in [14], nonlinear evaluation mechanisms can now be more powerful than linear ones; they also analyze the complexity of a variety of associated optimization problems. We can think of our setting to some extent in this language by viewing any action that makes an agent truly qualified as "admissible" (and specifically the blue edges in our general discrete model). However, two key distinctions are (1) in our setting we can only give the loan/diploma or not – we do not have the flexibility to choose arbitrary payoffs, and (2) we assume agents may begin at different starting locations (but have the same costs for movement in our linear model).

Xiao et al. [18] define a problem they call the *Multiple Agents Contract Problem* which is very similar to our General Discrete Model, except instead of binary (red/blue) colors, the edges have different values to the principal, and instead of producing a classification, the principal can assign an arbitrary payment profile to the right-hand-side nodes. They prove that maximizing payoff to the principal is NP-hard, and give an algorithm for a case of related agents in which there is a certain strict ordering among agents and costs.

Shavit et al. [17], building on Miller et al. [15], consider the goal of getting agents to improve without loss of predictive accuracy. As in our setting, they assume agents begin a different starting locations, and then modify their profiles from there, and they also consider a learning formulation. However, their focus is on a regression model in which agents' payoffs are an inner product of their observable features with a decision vector; this means that the incentives are basically the same no matter what the initial location of an agent is. In contrast, in our binary classification setting, even in the linear model the effect of a proposed classifier on an agent may depend greatly (and in a non-convex manner) on the initial location of the agent. Bechavod et al. [4] also consider a linear regression learning setting: agents arrive one at a time iid from a fixed distribution and then modify their state by changing a single variable based on the current regression vector. As in our linear model, some directions are improvement and some are gaming. They consider a limited feedback setting where the learner sees only the dot-product of the agent's true position with the true regression function, plus noise, and the learner's goal is to recover the true regression function.

Haghtalab et al. [10] consider a similar setting to ours in which there are improvement and gaming actions, and the designer is limited to binary classification, where agents receive value 1 for being classified as positive. Among other results, they give approximation algorithms for the goal of maximizing the total amount of true improvement that occurs when the allowed mechanisms are linear separators and agents have ℓ_2 movement costs. In contrast, our goal is to maximize true positive classifications while minimizing false positives, and in the linear case our movement cost assumptions are somewhat different.

Organization of the Paper

Section 2 introduces the general discrete model and linear model more formally. In Section 3, we give an efficient algorithm for the problem of maximizing the number of true positives subject to no false positives in the general discrete model, and provide hardness results for the problem of maximizing the number of true positives subject to a nonzero bound on false positives (in either the general discrete model or the linear model when arbitrary classifiers are allowed) and hardness for the problem of maximizing the number of true positives subject to no false positives in the linear model when arbitrary classifiers are allowed. In Section 4, we consider a learning-theoretic version of the problem of maximizing true positives subject to no false positives, and provide efficient learning algorithms as well as upper and lower bounds on the number of samples needed. In Section 5, we focus on the linear model and provide algorithms specific to this setting. We provide an algorithm that determines whether there exists a linear classifier which classifies all agents accurately and causes all improvable agents to become qualified. In the special two-dimensional case, we design a general (not necessarily linear) classifier that maximizes true positives subject to no false positives. In the full version of this work, we show how to provide a linear classifier maximizing the number of true positives minus false positives in the two-dimensional case.

2 Model

We study a binary classification problem. As the mechanism designer or classifier, we would like to maximize the number of agents we correctly classify as positive (true positives), and minimize the number of unqualified agents we misclassify as positive (false positives).

Agents are assumed to be utility maximizers and wish to be classified as positive. Each agent $i \in \{1, \ldots, n\}$ has a set of actions it can perform, and it will choose the cheapest of these that causes it to be classified as positive if that cost is less than its value on receiving a positive classification. We use $\mathcal{Q}$ to denote the set of truly qualified agents. If an agent is initially not qualified (not in $\mathcal{Q}$), some of its actions may cause it to become truly qualified, whereas others may not. However, the classifier cannot see which action was taken, only the observable result of that action. Therefore, the challenge of the mechanism designer is to determine which observable results to classify as positive to maximize correct positive classifications while minimizing false positives.

2.1 General Discrete Model

In this model, we assume that as a mechanism designer we are given a polynomial-sized set $\mathcal{P}$ of criteria we may select from (e.g., graduation criteria or criteria for being approved for a loan), and are limited to choosing some subset $\mathcal{P}^{\text{final}} \subseteq \mathcal{P}$ as the criteria we will use. We then will classify as positive any agent that meets any one of these criteria, and as negative any agent who does not. Specifically, we are given a weighted, colored bipartite graph with the n agents on the left and the set $\mathcal{P}$ of criteria on the right. Edge (i, j) corresponds to agent i taking an action to satisfy criteria j and is colored blue or red depending on whether that action would make the agent truly qualified or not, respectively. Each edge also has a weight representing its cost to that agent, and only actions whose costs are less than the value to the agent of being classified as positive are shown. Given a set $\mathcal{P}^{\text{final}} \subseteq \mathcal{P}$ chosen by the mechanism designer, each agent in the neighborhood of $\mathcal{P}^{\text{final}}$ will choose its cheapest edge into $\mathcal{P}^{\text{final}}$ as the action it will take, and will be classified as positive by the mechanism; agents not in the neighborhood of $\mathcal{P}^{\text{final}}$ will be classified as negative.

We also consider a learning-theoretic version of this problem, where the left-hand-side of the graph is replaced with a probability distribution $\mathcal{D}$ over nodes. We have sampling access to $\mathcal{D}$ and our goal is to find a subset $\mathcal{P}^{\text{final}}$ of points on the right-hand-side with good performance under $\mathcal{D}$. In a partial-information (bandit-style) version, when we sample a point from $\mathcal{D}$ we do not get to observe its edges, only where it goes to and whether it was qualified. That is, learning proceeds in rounds, where in each round we choose a subset $\mathcal{P}'$ of points on the right, and then for a random draw $x \sim \mathcal{D}$ we observe what point $p \in \mathcal{P}'$ (if any) was selected and the color of the edge taken.

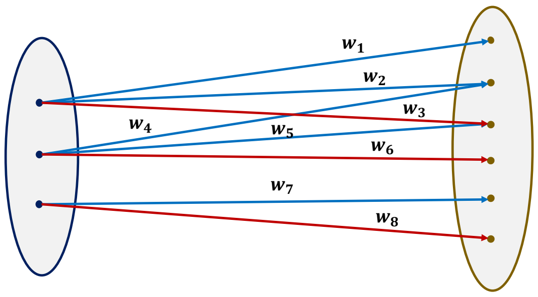

Figure 1 *Points on the left are the agents, and those on the right are the set $\mathcal{P}$ of possible criteria; w_i is the cost of satisfying the criterion. A red edge means the agent taking that action would not truly be qualified. A blue edge means that the agent taking that action would be qualified.*

2.2 Linear Model

In the linear model, agents have d real-valued features. Each agent i begins at an initial point $\mathbf{x}_i^{\text{init}} \in \mathbb{R}^d$, and there is assumed to be a linear threshold function $f^* : \mathbf{a}^*\mathbf{x} \geq b^*$ with non-negative weights that separates the truly qualified individuals from the unqualified ones. Agents have the ability to increase their jth feature at cost $\mathbf{c}[j]$ (decreasing is free) and receive value 1 for being classified as positive. However, only some features correspond to true improvement and others involve just gaming. That is, if an agent begins at $\mathbf{x}^{\text{init}}$ and moves to a point $\mathbf{x}^{\text{perc}}$, their true qualification is not $f^*(\mathbf{x}^{\text{perc}})$ but rather $f^*(\mathbf{x}^{\text{true}})$, where $\mathbf{x}^{\text{true}}$ agrees with $\mathbf{x}^{\text{init}}$ in the gaming directions and with $\mathbf{x}^{\text{perc}}$ in the improvement directions. On the other hand, the classification rule can only be based only on $\mathbf{x}^{\text{perc}}$ and not $\mathbf{x}^{\text{true}}$ (or $\mathbf{x}^{\text{init}}$). Movement costs and which features are improvement versus gaming are assumed to be the same for all agents. So, for any agent i, $\mathit{cost}(\mathbf{x}_i^{\text{init}}, \mathbf{x}_i^{\text{perc}}) = \sum_{j=1}^{d} \mathbf{c}[j]\left(\mathbf{x}_i^{\text{perc}}[j] - \mathbf{x}_i^{\text{init}}[j]\right)^+$, where $x^+ = \max\{x, 0\}$ and $\mathbf{c}[j]$ is the cost per unit of movement in the positive direction of dimension j. An example is given in Figure 2.

We consider settings where (a) the mechanism designer must use a linear classifier (a linear threshold function), (b) arbitrary classifiers are allowed, and (c) a polynomial-sized set $\mathcal{P}$ of "target points" is given and the mechanism designer must select some subset $\mathcal{P}^{\text{final}} \subseteq \mathcal{P}$ as its classifier. Notice that this last case is a special case of the general discrete model because given each initial state $\mathbf{x}_i^{\text{init}}$, we can compute the costs to move to each $p \in \mathcal{P}$ and whether doing so will make the agent truly qualified, to produce the desired weighted, colored bipartite graph.

3 Algorithmic and Hardness Results

In this section we first provide an algorithm for the problem of maximizing the number of true positives subject to no false positives in the general discrete model. Then, we provide hardness results for the problem of maximizing the number of true positives subject to a nonzero bound on false positives (in either the general discrete model or the linear model when arbitrary classifiers are allowed) and hardness for the problem of maximizing the number of true positives subject to no false positives in the linear model when arbitrary classifiers are allowed. Later in Section 4 we extend our algorithmic results to the learning model and in Section 5 we give algorithms for learning linear classifiers in the linear model.

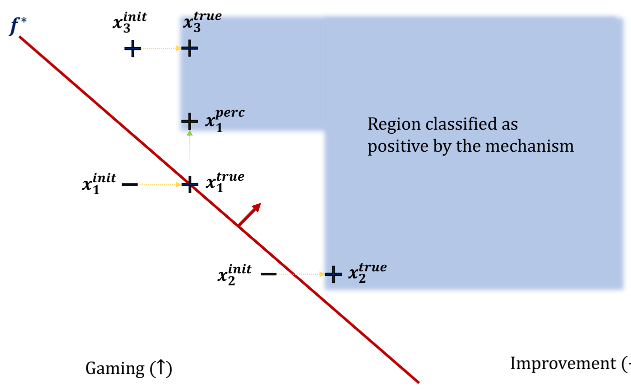

Figure 2 *An example of the linear model (the horizontal axis is an improvement direction and the vertical axis is a gaming direction) with a mechanism using a non-linear classifier. There are three agents, two of whom are initially not qualified. All three become qualified and are correctly classified as positive by the mechanism.*

3.1 Maximize True Positives Subject to No False Positives

The main result of this section is an algorithm that given a weighted, colored bipartite graph $\mathcal{G}$ with agents, $\mathcal{X}$, on the left and potential criteria, $\mathcal{P}$, on the right, finds $\mathcal{P}^{\text{final}} \subseteq \mathcal{P}$ such that using $\mathcal{P}^{\text{final}}$ as the criteria maximizes the number of agents taking a blue edge (true positive) subject to no agent taking a red edge (false positive). We call the agents that take a blue edge *improving agents* and the agents taking a red edge *gaming agents*. The algorithm, although simple in structure, satisfies strong properties noted afterwards; and serves as the building block of the learning algorithms in Section 4. Furthermore, as shown in the following subsection, natural generalizations of the objective function make the problem computationally hard. Therefore, the algorithm together with the hardness results tightly characterize the settings for which there is an efficient algorithm, or the problem is NP-hard.

Overview of Algorithm 1. The algorithm takes in a weighted, colored bipartite graph $\mathcal{G} = (\mathcal{X} \cup \mathcal{P}, E)$ and outputs $\mathcal{P}^{\text{final}}$, a subset of $\mathcal{P}$ that specifies the final criteria. Initially, $\mathcal{P}^{\text{final}}$ is set to $\mathcal{P}$. The algorithm proceeds in rounds. In each round, it visits all the nodes (agents) in $\mathcal{X}$ to determine whether there is an agent who takes a red edge to its lowest cost neighbor $p \in \mathcal{P}^{\text{final}}$. If there is such a gaming agent, its corresponding criteria, p, is removed from $\mathcal{P}^{\text{final}}$. These rounds continue until there is no gaming agent and therefore no removal of criteria in a single round, or the current set of criteria is empty.

▶ **Proposition 1.** *Algorithm 1 has running time of $O(|\mathcal{P}|n)$.*

Proof. Proof in Appendix A. ◀

▶ **Theorem 2.** *Algorithm 1 finds the set of criteria, $\mathcal{P}^{final}$, that maximizes the number of true positives subject to no false positive.*

Proof. Proof in Appendix A. ◀

Algorithm 1 satisfies the following strong properties.

(a) *point-wise optimality*: For any agent i, if there exists a solution in which i takes a blue edge and no agent takes a red edge, then the algorithm finds such a solution.

Algorithm 1 Maximize true positives subject to no false positives.

```
Input  : A bipartite graph $\mathcal{G} = (\mathcal{X} \cup \mathcal{P}, E)$ with edge weights $w_e$. Outgoing edges
         assumed sorted by weight. Red edges $E_R \subseteq E$. Blue edges $E_B \subseteq E$.
Output : $\mathcal{P}^{\text{final}}$
1  $\mathcal{P}^{\text{final}} \leftarrow \mathcal{P}$ // Initialization of the set
2  while $\mathcal{P}^{final} \neq \emptyset$ do
3      $flag = 0$
       /* Loop through all $x_i \in \mathcal{X}$ */
4      for $i = 1, 2, \cdots$ do
5          Let $e = (x_i, p \in \mathcal{P}^{\text{final}})$ be the outgoing edge from $x_i$ with lowest weight
6          if $e \in E_R$ then
7              $flag = 1$ // at least one agent is gaming
8              $\mathcal{P}^{\text{final}} \leftarrow \mathcal{P}^{\text{final}} \setminus \{p\}$
9      if $flag$ is 0 then
10         return $\mathcal{P}^{\text{final}}$
11 return $\emptyset$ // When 0 false positive is not possible
```

(b) *general for weighted setting*: The algorithm works optimally in the more general setting that each agent has a weight and the objective is to maximize the sum of weights of improving agents subject to the constraint of no gaming agent. This is a direct implication of property a.

(c) *max-min fairness:* Suppose the agents are from different populations and the objective is to maximize the minimum number of agents improving from each population subject to no gaming. By property a, the algorithm satisfies this max-min fairness notion.

(d) *heterogeneous utilities*: The algorithm works optimally in the more general setting that agents have different values for being classified positive.

(e) *minimizing the total cost of improvement*: Since the algorithm only removes $p \in \mathcal{P}$ that causes an agent to game, with $\mathcal{P}^{\text{final}}$ each agent incurs the minimal cost subject to no agent gaming.

▶ Remark 3. The sets of criteria satisfying the no false positive constraint is not downward closed. In other words, a subset of a set of criteria that satisfies the no false positives property does not necessarily satisfy this property.

3.2 Hardness Results

In this part, we prove hardness results for maximizing the number of true positives when the constraints in the previous subsection are relaxed. First, we show that if we relax the no false positives constraint to a bounded number of false positives, the problem becomes NP-hard; moreover, this holds even for the simpler linear model. Then, for the linear model, we show if we are not given a finite set of potential criteria $\mathcal{P}$, it is NP-hard to find criteria that maximize true positives subject to no false positives.

▶ **Theorem 4.** *Given the initial feature vectors of agents $\mathbf{x}_1^{init}, \mathbf{x}_2^{init}, \ldots, \mathbf{x}_n^{init} \in \mathbb{R}^d$ and a set $\mathcal{P}$ of potential criteria, the problem of finding a subset $\mathcal{P}^{final} \subseteq \mathcal{P}$ that maximizes the number of true positives subject to at most k false positives is NP-hard.*

Proof sketch. The proof is done by a reduction from the Max-k-Cover problem with n elements where the goal is to choose k sets covering the most elements. For every element e_i in the Max-k-Cover, we consider agent i, and for every set S_j in the Max-k-Cover problem we consider agent $n+j$ and a target point $\mathbf{p}_j$. The coordinates of the initial points and the target points are set such that agent i corresponding to element e_i can only move to target point $\mathbf{p}_j$ such that $e_i \in S_j$ and become a true positive; moreover, agent $n+j$ corresponding to set S_j can only move to target point $\mathbf{p}_j$ and become a false positive. On the one hand, since including each $\mathbf{p}_j$ in the final set of criteria, $\mathcal{P}^{\text{final}}$, causes exactly one agent to be a false positive, $\mathcal{P}^{\text{final}}$ must contain at most k target points. On the other hand, to maximize the number of true positives a set of k target points that the maximum number of agents can reach to it must be selected. This is equivalent to the Max-k-Cover solution. A formal proof is included in Appendix A. ◀

▶ **Theorem 5.** *Suppose we are given a set of n agents where $\mathbf{x}_1^{init}, \mathbf{x}_2^{init}, \ldots, \mathbf{x}_n^{init}$ denote their initial feature vectors. Deciding whether there exists a set of target points $\mathcal{P}^{final} \subseteq \mathbb{R}^d$ for which all the agents become true positives is NP-hard.*

Proof sketch. The proof is done by a reduction from the approximate version of the hitting set problem where given a set of elements, $\mathcal{E} = \{e_1, \ldots, e_n\}$ and a family of sets of elements, $\mathcal{F} = \{S_1, S_2, \ldots, S_m\}$, the goal is to find a minimum size set S^* that intersects all S_i. We construct an $n+1$-dimensional space, where the first n dimensions are improvement dimensions and correspond to the n elements, and the last dimension is gaming. We consider two sets of agents. For each S_i, we consider a corresponding agent i; these are the *usual* agents. We also consider agent $m+1$, a *special* agent that does not correspond to any particular set. The construction is such that each agent needs to move $2k$ units along the improvement dimensions to become truly qualified. Further details of the construction can be found in the full proof. The proof includes two directions. (1) If all the agents can become true positives by reaching to a set of target points $\mathcal{P}^{\text{final}} \subseteq \mathbb{R}^d$, then we can construct a hitting set of size at most $2k$; and (2) if it is not possible, then there does not exist a hitting set of size k.

We briefly cover the key ideas in each direction. To show the first direction, suppose all the agents can become true positives when presented with target points $\mathcal{P}^{\text{final}} \subseteq \mathbb{R}^d$. Consider the target point that each agent selects. Using our construction, we show the special agent does not afford to reach to the target points of the usual agents. Also, for each usual agent i, there exists element e_j in their corresponding set such that the target point of the special agent has value more than 1 in coordinate j. In order for the special agent to afford to reach to its target point, the number of improvement coordinates with value at least 1 must be at most $2k$. The elements corresponding to these coordinates constitute a hitting set of size at most $2k$. To prove the reverse direction we argue: if there exists a hitting set S^* of size k, there is a set of target points that encourages all the agents to become true positives. To do so, we construct a set of target points $\mathcal{P}^{\text{final}} = \{\mathbf{p}_1, \ldots, \mathbf{p}_{m+1}\}$, using the elements in the hitting set, that when the size of the hitting set is k makes every agent become true positive. A formal proof is included in Appendix A. ◀

The following is a direct corollary of Theorem 5.

▶ **Corollary 6.** *Given the initial feature vectors of agents, $\mathbf{x}_1^{init}, \mathbf{x}_2^{init}, \ldots, \mathbf{x}_n^{init} \in \mathbb{R}^d$, finding a set of target points $\mathcal{P}^{final} \subseteq \mathbb{R}^d$ that maximizes the number of true positives subject to no false positives is NP-hard.*

4 Learning Results

In this section we consider a learning-theoretic version of our problem, where the left-hand-side of the graph is replaced with a probability distribution $\mathcal{D}$ over nodes. We have sampling access to $\mathcal{D}$ and our goal is to find a subset $\mathcal{P}^{\text{final}}$ of points on the right-hand-side with good performance under $\mathcal{D}$. We provide two different algorithmic results and upper bounds on the number of samples for producing a good solution, depending on the information each sample reveals. The first upper bound works for the case where by sampling an agent, its neighborhood (neighboring edges, their colors and weights) is revealed. The second upper bound works in a partial-information (bandit-style) setting, where when we sample a point from $\mathcal{D}$ we do not get to observe its edges, only where it goes to and whether it was qualified. Finally, we provide a lower bound on the necessary number of samples for any algorithm. The lower bound holds even for the simpler linear model.

The following definition is crucial in this section.

▶ **Definition 7** (OPT, performance, and error). *Let OPT be the maximum probability mass of true positives achievable subject to zero false positives. We denote the probability mass of true positives of an algorithm as its* performance *and the probability mass of false positives as its* error. *A hypothesis is desired if it has comparable performance to OPT and small error.*

4.1 Sufficient Number of Samples in the Full Information Setting

The main result of this section is that a number of samples linear in $|\mathcal{P}|$ and $1/\varepsilon$ is sufficient for Algorithm 1 to learn a desired hypothesis with high probability. Specifically, suppose the learner has access to a weighted, colored bipartite graph $\mathcal{G} = (\mathcal{X} \cup \mathcal{P}, E)$, where $\mathcal{X}$ are sampled from $\mathcal{D}$, and $\mathcal{P}$ is the set of the potential criteria. The learner runs Algorithm 1 with the graph as the input and uses the algorithm output, $\mathcal{P}^{\text{final}} \subseteq \mathcal{P}$, as its hypothesis, i.e., after the training phase it classifies any agent with an edge to $\mathcal{P}^{\text{final}}$ as positive and any other agent as negative. We show that a linear number of samples is sufficient so that with high probability, the probability mass of true positives classified by $\mathcal{P}^{\text{final}}$ is close to OPT and the probability mass of false positives is small.

▶ **Theorem 8.** *Consider $\mathcal{P}^{final}$ as the outcome of Algorithm 1 on $\mathcal{G} = (\mathcal{X} \cup \mathcal{P}, E)$, where $\mathcal{X}$ contains samples from $\mathcal{D}$. For any $0 < \varepsilon, \delta \leq 1$, if $|\mathcal{X}| \geq \varepsilon^{-1}(\ln(2)|\mathcal{P}| + \ln(1/\delta))$ then with probability at least $1 - \delta$ the set $\mathcal{P}^{final}$ achieves performance at least $OPT - \varepsilon$ (i.e., at least $OPT - \varepsilon$ probability mass of true positives) subject to at most ε error (ε probability mass of false positives).*

4.2 Sufficient Number of Samples in the Partial Information Setting

In this section we consider a partial information (bandit-style) setting. Similar to before, the learner has access to a sample set $\mathcal{X}$ drawn from D and a set of potential criteria $\mathcal{P}$. However, observing a sample in $\mathcal{X}$ does not reveal its edges, and the learner can only observe the criterion that the sample selects and whether it becomes truly qualified. The main result of this section is an algorithm, Algorithm 2, for this setting and a guarantee on the number of samples sufficient for it to achieve performance at least $\text{OPT} - \varepsilon$ and error at most ε with high probability.

Overview of Algorithm 2. In each iteration, a set of examples of size $\varepsilon^{-1}\ln(|\mathcal{P}|/\delta)$ is sampled. After agents select points in $\mathcal{P}$ (if any), we observe the points selected and whether they became truly qualified (in a real-world application, one can think of performing a test to check if each agent is truly qualified). If some agent does not become truly qualified (fails the test), the algorithm deletes the point they have selected. If a set $\mathcal{P}^{\text{final}}$, survives for $\varepsilon^{-1}\ln(|\mathcal{P}|/\delta)$ subsequent examples, the algorithm terminates and returns $\mathcal{P}^{\text{final}}$ as the

Algorithm 2 Learning a high performance low error $\mathcal{P}^{\text{final}}$ in partial-information setting.

```
  Input  : $\mathcal{P}$
  Output : $\mathcal{P}^{final}$
1 $\mathcal{P}^{\text{final}} \leftarrow \mathcal{P}$;
2 while $\mathcal{P}^{final} \neq \emptyset$ do
3     Sample $\mathcal{X} \sim \mathcal{D}$ of size $\frac{1}{\varepsilon} \ln \frac{|\mathcal{P}|}{\delta}$ ;
4     if $\exists x \in \mathcal{X}$ such that $x$ takes a red edge to $p \in \mathcal{P}^{final}$ then
5         $\mathcal{P}^{\text{final}} \leftarrow \mathcal{P}^{\text{final}} \setminus \{p\}$;
6         continue;
      /* if no one from $\mathcal{X}$ takes a red edge: */
7     return $\mathcal{P}^{\text{final}}$;
8 return $\emptyset$;
```

the final set of criteria of the algorithm. Since the number of false positives (agents taking red edges) is bounded by $|\mathcal{P}|$, the algorithm will terminate after at most $\varepsilon^{-1}|\mathcal{P}|\ln(|\mathcal{P}|/\delta)$ samples.

The following theorem proves that with a high probability, Algorithm 2 outputs $\mathcal{P}^{\text{final}}$ with a high performance and a low error.

▶ **Theorem 9.** *For any $0 < \varepsilon, \delta \leq 1$, Algorithm 2 by using at most $\varepsilon^{-1}|\mathcal{P}|\ln(|\mathcal{P}|/\delta)$ total samples outputs a set of criteria $\mathcal{P}^{final}$ that with probability at least $1-\delta$ achieves performance at least $OPT - \varepsilon$ (i.e., at least $OPT - \varepsilon$ probability mass of true positives) subject to at most ε error (ε probability mass of false positives).*

4.3 Necessary Number of Samples

The main result of this section is a lower bound on the necessary number of samples for learning a desired hypothesis. The lower bound provided holds even for the simpler linear model. To restate the setup, suppose the learner has access to a set of initial positions of agents $\mathcal{X}$ and a set of potential criteria (also called target points in the linear model) $\mathcal{P}$ where $\mathcal{X}$ are sampled from distribution $\mathcal{D}$. We lower-bound the required number of samples for any learning algorithm that with probability at least 1/2 achieves high performance and low error.

▶ **Theorem 10.** *Any algorithm for PAC learning a set $\mathcal{P}^{final}$ that with probability at least $1/2$ achieves performance at least $(3/4) \cdot OPT$ (i.e., at least $(3/4) \cdot OPT$ probability mass of true positives) subject to at most ε error (ε probability mass of false positives) must use $\Omega(|\mathcal{P}|/\varepsilon)$ examples in the worst case.*

5 Algorithmic Results Specific to the Linear Model

The algorithmic results provided so far work in both the general discrete and the linear discrete models. In this section we focus on the linear model and provide algorithmic results for various problems. These algorithms do not follow the greedy structure of the previous algorithms, and use novel technical ideas. First, we consider the problem of designing *linear classifiers*. Section 5.1 provides introductory observations and definitions about linear classifiers. Section 5.2 presents the main result of this section which determines whether there exists a linear classifier that classifies all agents accurately and causes all improvable agents to become qualified. Then, we shift focus to general (not necessarily linear) classifiers in a two-dimensional space and in Section 5.3 provide an algorithm for maximizing true positives subject to no false positives. In the full version of this work, we provide results for finding a linear classifier that maximizes the number of true positives minus false positives in the two-dimensional case.

5.1 Properties of Linear Classifiers

Before diving into discussion of the algorithmic results, we provide observations about linear classifiers to set the context. We also provide optimal classifiers in special cases.

For the following discussion, consider linear classifier $f^* : \mathbf{a}^*\mathbf{x} \geq b^*$ that separates the truly qualified agents from unqualified agents.

▶ **Observation 11.** *With linear classifier $f : \mathbf{a}\mathbf{x} \geq b$, any utility maximizing agent that achieves non-negative utility by changing their features moves in dimension $\arg\max_j \mathbf{a}[j]/\mathbf{c}[j]$.*

▶ **Definition 12** (movement dimension)**.** *The movement dimension of linear classifier $f : \mathbf{a}\mathbf{x} \geq b$ is the utility maximizing dimension $\arg\max_j \mathbf{a}[j]/\mathbf{c}[j]$ discussed in Observation 11. If there are multiple such dimensions the ties are broken in favor of improvement dimensions and then lexicographically.*

▶ **Definition 13** (encourage improvement/gaming)**.** *A classifier encourages improvement if its movement dimension is an improvement dimension. It encourages gaming otherwise.*

▶ **Definition 14** (dim-j improving)**.** *A linear classifier is dim-j improving if it encourages improvement and its movement dimension is along dimension j.*

The following definition captures the set of agents that potentially can improve to become truly qualified.

▶ **Definition 15** (improvement margin, improvable agents)**.** *The improvement margin includes all the agents that can afford (do not have to incur a cost of more than 1) to move in an improvement dimension and become truly qualified. Formally, any initially unqualified agent i, i.e., $\mathbf{a}^*\mathbf{x}_i^{init} < b^*$, that has distance $\leq 1/\mathbf{c}[j]$ along an improvement dimension j to f^* is in the improvement margin.*

▶ **Lemma 16.** *If $f^* : \mathbf{a}^*x \geq b^*$ encourages improvement, the optimal classifier is f^* – among all linear or nonlinear classifiers.*

Proof. f^* classifies initially qualified agents and unqualified unimprovable agents accurately. Also, all the agents in the improvement margin improve, become qualified, and are accurately classified as positive. ◀

▶ **Lemma 17.** *Let j be the movement dimension of classifier f^*. The classifier $g : \mathbf{a}^*\mathbf{x} \geq b^* + \mathbf{a}^*[j]/\mathbf{c}[j]$ classifies all the initially qualified agents as positive and the rest as negative.*

Proof. Initially unqualified agents, $\mathbf{a}^*\mathbf{x}_i^{\text{init}} < b^*$, can move at most $1/\mathbf{c}[j]$ in dimension j which is not enough to reach to g. Therefore, these agents are classified as negative by g. On the other hand, initially qualified agents, $\mathbf{a}^*\mathbf{x}_i^{\text{init}} \geq b^*$, afford to reach to g and receive nonnegative utility. Therefore, they will be classified as positive. ◀

▶ **Corollary 18.** *If all the dimensions are gaming dimensions, $g : \mathbf{a}^*\mathbf{x} \geq b^* + \mathbf{a}^*[j]/\mathbf{c}[j]$ is the optimal classifier, where j is the movement dimension of f^*.*

Proof. If all dimensions are gaming dimensions, there are no improvable agents. Therefore, all agents are either initially qualified or unimprovable and unqualified. By Lemma 17, g classifies all such agents accurately. ◀

By Lemma 17, $g : \mathbf{a}^*\mathbf{x} \geq b^* + \mathbf{a}^*[j]/\mathbf{c}[j]$ may be a "reasonable" solution because it classifies all the initially qualified as positive and does not result in any false positive classifications. However, it misses out on any new true positives resulting from encouraging agents to become qualified. From this point on, we aim to study other classifiers (not necessarily parallel to f^*) with the hope of encouraging other agents to become qualified.

5.2 Linear Classifier for Improvable Agents

In this subsection, we study a problem that takes as input three disjoint subsets of the agents, $\mathcal{S}^{\text{yes}}$, $\mathcal{S}^{\text{no}}$, and $\mathcal{S}^{\text{imp}}$, and outputs a linear classifier (if one exists) that satisfies the following properties.

i) Classifies agent i such that $\mathbf{x}_i^{\text{init}} \in \mathcal{S}^{\text{yes}}$ as positive.
ii) Classifies agent i such that $\mathbf{x}_i^{\text{init}} \in \mathcal{S}^{\text{no}}$ as negative.
iii) Encourages agent i such that $\mathbf{x}_i^{\text{init}} \in \mathcal{S}^{\text{imp}}$ to improve and become truly qualified, i.e., $\mathbf{x}_i^{\text{true}} \in \mathcal{Q}$, and classifies i as positive.

The main result of the section is solving this problem in polynomial time. When $\mathcal{S}^{\text{yes}}$ is the set of initially qualified agents, $\mathcal{S}^{\text{no}}$ is the set of unqualified and unimprovable, and $\mathcal{S}^{\text{imp}}$ is the set of improvable agents, this problem determines whether there exists a linear classifier that classifies $\mathcal{S}^{\text{yes}}$ and $\mathcal{S}^{\text{no}}$ accurately and makes all the improvable agents qualified.

To solve this problem, we divide it into subproblems as following: Does there exist a linear classifier with movement direction in *dimension* j that satisfies properties i, ii, and iii? If the answer is "yes" for some dimension j, then the answer to the main problem is "yes". If the answer is "no" for all $1 \leq j \leq d$, no linear classifier satisfying the three properties exists.

Note that if $\mathcal{S}^{\text{imp}}$ is nonempty, in order to satisfy property iii, dimension j must be an improvement dimension. Therefore, we study the following problem.

▶ **Problem 1.** *Does there exist a dim-j improving classifier (a linear classifier encouraging improvement* in dimension j*) that satisfies properties i, ii, and iii?*

We propose a linear program that solves Problem 1. The following definition and observations illustrate the conditions under which a dim-j improving classifier satisfies each property for agent i.

▶ **Definition 19.** *For a fixed improvement dimension j and classifiers $f^* : \mathbf{a}^*\mathbf{x} \geq b^*$ and $f : \mathbf{a}\mathbf{x} \geq b$, the points $\mathbf{x}_{i,f^*}$, $\mathbf{x}_{i,f}$, $\mathbf{x}_{i,max}$ are defined as follows (depicted in Figure 3.):*

- *$\mathbf{x}_{i,f^*}$ is the projection of $\mathbf{x}_i^{init}$ on the separating hyperplane of classifier f^* along dimension j.*
- *$\mathbf{x}_{i,f}$ is the projection of $\mathbf{x}_i^{init}$ on the separating hyperplane of classifier f along dimension j.*
- *$\mathbf{x}_{i,max}$ is the shifted $\mathbf{x}_i^{init}$ along dimension j by $1/\mathbf{c}[j]$.*

More formally, for all coordinates $k \neq j$, we have $\mathbf{x}_{i,f^*}[k] = \mathbf{x}_{i,f}[k] = \mathbf{x}_{i,max}[k] = \mathbf{x}_i^{init}[k]$. Also, since $\mathbf{a}^*\mathbf{x}_{i,f^*} = b^*$, we have $\mathbf{x}_{i,f^*}[j] = \left(b^* - \sum_{k\neq j} \mathbf{a}^*[k]\mathbf{x}_i^{init}[k]\right)/\mathbf{a}^*[j]$. Similarly, since $\mathbf{a}\mathbf{x}_{i,f} = b$, we have $\mathbf{x}_{i,f}[j] = \left(b - \sum_{k\neq j} \mathbf{a}^*[k]\mathbf{x}_i^{init}[k]\right)/\mathbf{a}[j]$. Finally, $\mathbf{x}_{i,max}[j] = \mathbf{x}_i^{init}[j] + 1/\mathbf{c}[j]$.

▶ **Observation 20.** *A dim-j improving classifier $f : \mathbf{ax} \geq b$ classifies agent i as positive (property i) if $\mathbf{ax}_{i,max} \geq b$. It classifies agent i as negative (property ii) if $\mathbf{ax}_{i,max} < b$.*

▶ **Observation 21.** *Using a dim-j improving classifier f, agent i becomes qualified and is classified as positive (property iii) if and only if $\mathbf{x}_{i,f^*}[j] \leq \mathbf{x}_{i,f}[j] \leq \mathbf{x}_{i,max}[j]$. See Figure 3.*

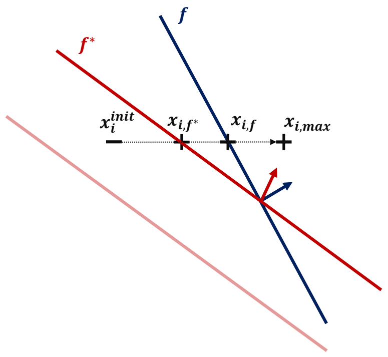

Figure 3 Depicting $\mathbf{x}_i^{\text{init}}, \mathbf{x}_{i,f^*}, \mathbf{x}_{i,f}, \mathbf{x}_{i,\text{max}}$ in Definition 19 and Observation 21. The horizontal axis shows dimension j in the definition.

▶ **Proposition 22.** *The following LP captures Problem 1, where the variables are $\mathbf{a}$ and b.*

$$\frac{\mathbf{a}[k]}{\mathbf{c}[k]} \leq \frac{\mathbf{a}[j]}{\mathbf{c}[j]} \qquad \forall k \neq j \tag{1}$$

$$b \leq \mathbf{ax}_{i,max} \qquad \forall \mathbf{x}_i^{init} \in \mathcal{S}^{yes} \tag{2}$$

$$\mathbf{ax}_{i,max} < b \qquad \forall \mathbf{x}_i^{init} \in \mathcal{S}^{no} \tag{3}$$

$$\mathbf{x}_{i,f^*}[j] \leq \mathbf{x}_{i,f}[j] \qquad \forall \mathbf{x}_i^{init} \in \mathcal{S}^{imp} \tag{4}$$

$$\mathbf{x}_{i,f}[j] \leq \mathbf{x}_{i,max}[j] \qquad \forall \mathbf{x}_i^{init} \in \mathcal{S}^{imp} \tag{5}$$

Constraint 1 asserts that the movement direction of the classifier is along dimension j. Constraint 2 asserts property i. Constraint 3 asserts property ii. Finally, constraints 4 and 5 assert property iii.

▶ **Theorem 23.** *Given the sets $\mathcal{S}^{yes}$, $\mathcal{S}^{no}$, and $\mathcal{S}^{imp}$, there is a polynomial-time algorithm that outputs a linear classifier (if one exists) that satisfies Properties i, ii,iii, or declares non-existence of such a classifier.*

Proof. If $\mathcal{S}^{\text{imp}} \neq \emptyset$, run LP 1-5 for all improvement dimensions j. If $\mathcal{S}^{\text{imp}} = \emptyset$, run the LP for $1 \leq j \leq n$. By Proposition 22, if there exist feasible solution $\mathbf{a}$ and b for one of these LPs, $f : \mathbf{ax} \geq b$ is a classifier satisfying properties i, ii, and iii. ◀

▶ **Corollary 24.** *There is a polynomial-time algorithm that determines whether there exists a linear classifier that classifies the initially qualified as positive, unqualified unimprovable agents as negative, encourages the agents in the improvement margin to improve to become qualified, and classifies them as positive. If such a classifier exists, it maximizes true positives subject to no false positives.*

▶ Remark 25. Theorem 5 asserts that given the initial feature vectors of agents, $\mathbf{x}_1^{\text{init}}, \mathbf{x}_2^{\text{init}}, \ldots, \mathbf{x}_n^{\text{init}} \in \mathbb{R}^d$, deciding whether there exists a classifier for which all the agents become true positives is NP-hard. However, when limiting to linear classifiers this problem is no longer NP-Hard. Using Theorem 23, by setting $\mathcal{S}^{\text{yes}}$ to the set of initially qualified agents, and $\mathcal{S}^{\text{imp}}$ to the rest of the agents, this problem is solvable in polynomial time.

5.3 Optimal General Classifier in Two-Dimensional Space

In this subsection, we consider the problem of maximizing true positives subject to no false positives in a 2-dimensional space, where the horizontal dimension is improvement, and the vertical dimension is gaming. We provide an algorithm in the linear model that given a set of agents, returns a set of target points $\mathcal{P}^{\text{final}} \subset \mathbb{R}^2$ that maximizes true positives subject to no false positives. Note that unlike Algorithm 1, our algorithm in this subsection does not take a finite set of target points $\mathcal{P}$ as input. For simplicity, by scaling we may assume wlog that $c = \mathbf{c}[1] = \mathbf{c}[2]$.

Overview of Algorithm 3. First, all the points $\mathbf{x}_i^{\text{init}}$ for $1 \le i \le m$ are sorted along the gaming dimension in a descending order, such that $\mathbf{x}_n^{\text{init}}$ has the smallest value in the gaming dimension. Our goal is to find designated points, $\mathbf{x'}_i$, for each $\mathbf{x}_i^{\text{init}}$. Starting with $\mathbf{x}_n^{\text{init}}$, for each point $\mathbf{x}_i^{\text{init}}$, move $\mathbf{x}_i^{\text{init}}$ along the improvement dimension until it crosses the line $\mathbf{a}^*\mathbf{x} = b^*$ at $\mathbf{x}_{i,min}$ (See Figure 4). Let $\mathbf{x'}_i$, the designated point of $\mathbf{x}_i^{\text{init}}$, be initially $\mathbf{x'}_i = \mathbf{x}_{i,min}$. If given the current set of designated points for agents $n, n-1, \ldots, i$, another point $\mathbf{x}_j^{\text{init}}$ for $j > i$ maximizes utility by moving to $\mathbf{x'}_i$ and becomes false positive, push $\mathbf{x'}_i$ upward along the gaming dimension, until $\mathbf{x}_j^{\text{init}}$ no longer picks $\mathbf{x'}_i$. When pushing $\mathbf{x'}_i$ along the gaming dimension, let $\mathbf{x}_{i,max}$ denote the furthest point that $\mathbf{x}_i^{\text{init}}$ can afford to reach to it. If the final point $\mathbf{x'}_i$ is such that $\mathbf{x}_i^{\text{init}}$ cannot afford to move to it, i.e. $\mathbf{x'}_i[2] > \mathbf{x}_{i,max}[2]$, discard $\mathbf{x'}_i$. Otherwise, $\mathbf{x'}_i$ is added to $\mathcal{P}^{\text{final}}$.

Note that we assume that if a point $\mathbf{x}_j^{\text{init}}$ can improve to $\mathbf{x'}_j$ and game to $\mathbf{x'}_i$ with the same cost, it would pick the improvement option.

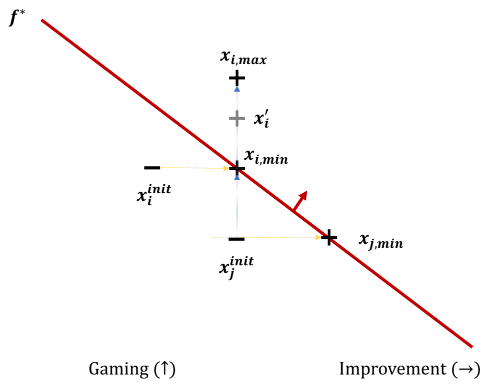

Figure 4 In Algorithm 3, $\mathbf{x'}_i$ is pushed along the gaming dimension so $\mathbf{x}_j^{\text{init}}$ no longer moves to it.

▶ **Theorem 26.** *Given initial feature vectors of agents, $\mathbf{x}_1^{init}, \mathbf{x}_2^{init}, \ldots, \mathbf{x}_n^{init} \in \mathbb{R}^2$, Algorithm 3 maximizes the number of true positives subject to no false positives.*

Proof. Proof is deferred to Appendix B. ◀

▶ Remark 27. By Corollary 6, this problem is NP-hard when $\mathcal{X} \subset \mathbb{R}^d$ for general (not constant) d.

Algorithm 3 Maximizing the number of true positives in 2-dimensions.

```
Input  : $\mathcal{X}$, $f^* : \mathbf{a}^*\mathbf{x} \geq b^*$
Output : $\mathcal{P}^{\text{final}}$
1  Sort $\mathbf{x}_i \in \mathcal{X}$ in a descending order of $\mathbf{x}_i[2]$;
2  for $i = n, \cdots, 1$ do
       /* Let $\mathbf{x}_{i,min}$ be the projection of $\mathbf{x}_i$ on $\mathbf{a}^*\mathbf{x} = b^*$ along the
          improvement dimension */
3      $\mathbf{x}_{i,min} = \left(\frac{b^* - \mathbf{a}^*[2]\mathbf{x}_i[2]}{\mathbf{a}^*[1]}, \mathbf{x}_i[2]\right)$;
4      if $\mathbf{x}_{i,min}[1] - \mathbf{x}_i[1] > 1/c$ then
           /* $\mathbf{x}_i$ cannot become true positive.  */
5          continue;
6      $\mathbf{x'}_i \leftarrow \mathbf{x}_{i,min}$;
7      for $j = n, \cdots, i+1$ do
8          if $cost(\mathbf{x}_j, \mathbf{x}'_j) > cost(\mathbf{x}_j, \mathbf{x}'_i)$ then
9              $\mathbf{x}'_i \leftarrow (\mathbf{x}'_i[1], \mathbf{x}'_i[2] + cost(\mathbf{x}_j, \mathbf{x}'_j) - cost(\mathbf{x}_j, \mathbf{x}'_i))$;
10     if $\mathbf{x'}_i[2] > \mathbf{x}_{i,max}[2]$ then
           /* $\mathbf{x}_i$ cannot become true positive without another point becoming
              false positive.  */
11         $\mathbf{x'}_i = (\mathbf{x'}_i[1], \infty)$;
12     $\mathcal{P}^{\text{final}} \leftarrow \mathcal{P}^{\text{final}} \cup \mathbf{x'}_i$;
13     return $\mathcal{P}^{\text{final}}$;
```

References

1 Saba Ahmadi, Hedyeh Beyhaghi, Avrim Blum, and Keziah Naggita. The strategic perceptron. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 6–25, New York, NY, USA, 2021. Association for Computing Machinery. doi:10.1145/3465456.3467629.

2 Saba Ahmadi, Hedyeh Beyhaghi, Avrim Blum, and Keziah Naggita. On classification of strategic agents who can both game and improve. *arXiv preprint*, 2022. arXiv:2203.00124.

3 Tal Alon, Magdalen Dobson, Ariel Procaccia, Inbal Talgam-Cohen, and Jamie Tucker-Foltz. Multiagent evaluation mechanisms. *In Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):1774–1781, April 2020. doi:10.1609/aaai.v34i02.5543.

4 Yahav Bechavod, Katrina Ligett, Zhiwei Steven Wu, and Juba Ziani. Causal feature discovery through strategic modification. *ArXiv*, abs/2002.07024, 2020. arXiv:2002.07024.

5 Mark Braverman and Sumegha Garg. The role of randomness and noise in strategic classification. In *Proceedings of the 1st Symposium on Foundations of Responsible Computing, FORC 2020, June 1-3, 2020, Harvard University, Cambridge, MA, USA (virtual conference)*, volume 156 of *LIPIcs*, pages 9:1–9:20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.FORC.2020.9.

6 Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 547–555, New York, NY, USA, 2011. Association for Computing Machinery. doi:10.1145/2020408.2020495.

7 Jinshuo Dong, Aaron Roth, Zachary Schutzman, Bo Waggoner, and Zhiwei Steven Wu. Strategic classification from revealed preferences. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, EC '18, pages 55–70, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3219166.3219193.

8 Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998. doi:10.1145/285055.285059.

9 Alex M. Frankel and Navin Kartik. Improving information from manipulable data. *arXiv: Theoretical Economics*, June 2019. doi:10.1093/jeea/jvab017.

10 Nika Haghtalab, Nicole Immorlica, Brendan Lucier, and Jack Z. Wang. Maximizing welfare with incentive-aware evaluation mechanisms. In Christian Bessiere, editor, *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pages 160–166. International Joint Conferences on Artificial Intelligence Organization, July 2020. Main track. doi:10.24963/ijcai.2020/23.

11 Moritz Hardt, Nimrod Megiddo, Christos Papadimitriou, and Mary Wootters. Strategic classification. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, pages 111–122, New York, NY, USA, 2016. Association for Computing Machinery. doi:10.1145/2840728.2840730.

12 Keegan Harris, Hoda Heidari, and Zhiwei Steven Wu. Stateful strategic regression. *CoRR*, abs/2106.03827, 2021. arXiv:2106.03827.

13 Lily Hu, Nicole Immorlica, and Jennifer Wortman Vaughan. The disparate effects of strategic manipulation. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 259–268, New York, NY, USA, 2019. ACM. doi:10.1145/3287560.3287597.

14 Jon Kleinberg and Manish Raghavan. How do classifiers induce agents to invest effort strategically? In *Proceedings of the 2019 ACM Conference on Economics and Computation*, EC '19, pages 825–844, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3328526.3329584.

15 John Miller, Smitha Milli, and Moritz Hardt. Strategic classification is causal modeling in disguise. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 6917–6926. PMLR, 2020. URL: http://proceedings.mlr.press/v119/miller20b.html.

16 Smitha Milli, John Miller, Anca D. Dragan, and Moritz Hardt. The social cost of strategic classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 230–239, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3287560.3287576.

17 Yonadav Shavit, Benjamin Edelman, and Brian Axelrod. Learning from strategic agents: Accuracy, improvement, and causality. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume abs/2002.10066 of *Proceedings of Machine Learning Research*, pages 8676–8686. PMLR, 13–18 July 2020. URL: http://proceedings.mlr.press/v119/shavit20a.html, arXiv:2002.10066.

18 Shenke Xiao, Zihe Wang, Mengjing Chen, Pingzhong Tang, and Xiwang Yang. Optimal common contract with heterogeneous agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7309–7316, April 2020. doi:10.1609/aaai.v34i05.6224.

A Missing Proofs of Section 3

Proof of Proposition 1. The size of $\mathcal{X}$ is n, and within the for loop each computation takes $O(1)$ time since the edges for each x_i are already sorted. When the flag is set to 1, at least one point in $\mathcal{P}$ is removed, and when the flag is 0 at the end of the inner loop, the algorithm returns. Therefore, the outer loop is run at most $|\mathcal{P}|$ times while the inner loop is run n times; resulting in a running time of $O(|\mathcal{P}|n)$. ◀

Proof of Theorem 2. Let A be the improving agents (agents taking blue edges) associated with the set of criteria $\mathcal{P}^{\text{final}}$. We show that having any other set $Q \subseteq \mathcal{P}$ as the criteria, either causes an agent to take a red edge, or no more than $|A|$ agents to take blue edges. To do so, consider partitioning Q into two subsets Q^F and $Q^{\bar{F}}$, where $Q^F \subseteq \mathcal{P}^{\text{final}}$ and $Q^{\bar{F}} \subseteq \mathcal{P} \setminus \mathcal{P}^{\text{final}}$.

First, we show that if $Q^{\bar{F}} \neq \emptyset$, an agent takes a red edge. To prove this claim, suppose by contradiction that $Q^{\bar{F}}$ is nonempty and consider the first time the algorithm deletes an element $p \in Q^{\bar{F}}$. At this stage, the set of criteria in the algorithm $\mathcal{P}'$ is a superset of

$Q^{\bar{F}} \cup \mathcal{P}^{\text{final}}$. By definition, p is the lowest-weight neighbor of a gaming agent, a, in $\mathcal{P}'$. This implies that p is also the lowest-weight neighbor of a in $Q \subseteq Q^{\bar{F}} \cup \mathcal{P}^{\text{final}} \subseteq \mathcal{P}'$, and a is a gaming agent given the criteria set Q. This implies the claim.

Secondly, we show that among the sets of criteria with no gaming agent, $\mathcal{P}^{\text{final}}$ has the highest number of improving agents. The previous claim implies that any set of criteria with no gaming agent is a subset of $\mathcal{P}^{\text{final}}$. Now, we need to show that among $Q \subseteq \mathcal{P}^{\text{final}}$, $\mathcal{P}^{\text{final}}$ has the largest set of improving agents. This is trivial, since by considering a subset we may only lose on agents in A that do not have a neighbor in Q or their lowest-weight edge is red. Therefore, any $Q \subseteq \mathcal{P}^{\text{final}}$ has at most $|A|$ improving agents. ◀

Proof of Theorem 4. We show the following problem is NP-hard.

► **Problem 2.** *Suppose we are given a set of n agents where $\mathbf{x}_1^{init}, \mathbf{x}_2^{init}, \ldots, \mathbf{x}_n^{init}$ denote their initial feature vectors, and a set $\mathcal{P}$ of potential criteria also called target points in the linear model. Find a subset $\mathcal{P}^{final} \subseteq \mathcal{P}$ that maximizes the number of true positives subject to at most k false positives.*

We prove the NP-hardness by reducing the Max-k-Cover problem with equal-sized sets of size 3 to this problem. In the Max-k-Cover problem, we are given a set $\mathcal{E}$ of elements e_i, and sets $S_j \subseteq \mathcal{E}$, and the goal is to select at most k sets out of S_j that maximize the number of elements they cover.

First, we show how to construct an instance of Problem 2 from an instance of the Max-k-Cover problem. To do so, we determine the number of dimensions, initial positions of the agents, the target points, and the movement costs. Let n be the number of elements of the Max-k-Cover instance, we construct an $n+1$-dimensional space where the first n dimensions are improvement and the last dimension is gaming. Consider elements $e_1, e_2, \ldots, e_n$ in the Max-k-Cover instance. For every element, we consider an agent; and for every set, we consider an agent and a target point. For e_i, the corresponding agent is at initial point $\mathbf{x}_i^{\text{init}}$, an $n+1$-dimensional vector whose i^{th} and $n+1^{st}$ coordinates are 1 and the other coordinates are 0. For every set S_j, we consider a target point $\mathbf{p}_j$ and an agent with initial point $\mathbf{x}_{n+j}^{\text{init}}$. In $\mathbf{p}_j$, the coordinates corresponding to the elements in S_j and the $n+1^{st}$ coordinate are set to 1 and the rest of the coordinates are 0. In $\mathbf{x}_{n+j}^{\text{init}}$, the coordinates corresponding to the elements in S_j are set to 1, the $n+1^{st}$ coordinate is set to -1, and the rest of the coordinates are 0. Finally, let the movement cost in any dimension be 1/2. Note that this construction fits into the framework of a linear model and $f^* : \sum_{j=1}^{n+1} \mathbf{x}[j] \geq 4$ is the linear threshold function for the truly qualified agents. All the target points $\mathbf{p}_j$ satisfy the threshold and all the agents are initially unqualified and do not meet the threshold.

Next, we discuss what target point each agent selects and whether they become truly qualified (true positive) or not (false positive). Because the cost per unit of movement equals 1/2, each agent can only afford to reach to target points with distance at most 2. Agents $\mathbf{x}_i^{\text{init}}$ for $i \in \{1, \ldots, n\}$ can only afford to reach a target point whose i^{th} coordinate is 1 since they are at distance 2. They are at distance 3 to any other target points. Since all dimensions $1, \ldots, n$ are improving dimensions these agents become truly qualified when they reach such target points. Agents $\mathbf{x}_i^{\text{init}}$ for $i > n$ can only afford to reach $\mathbf{p}_i$ since they have distance 2. They have distance more than 2 to any other target points. Agents $\mathbf{x}_i^{\text{init}}$ for $i > n$ can only reach to $\mathbf{p}_i$. To do so, these agents move in a gaming dimension and do not become truly qualified.

Finally, we show how the solutions of these two problems coincide. Consider the problem of maximizing the true positives subject to including at most k false positives. Including each $\mathbf{p}_j$ in the final set of target points, $\mathcal{P}^{\text{final}}$, causes exactly one agent, $\mathbf{x}_j^{\text{init}}$, to be a false positive. Therefore, having at most k false positive is equivalent to including at most k target points. Maximizing the true positives subject to at most k target points is exactly equivalent to selecting at most k sets that maximize the elements they cover. This completes the reduction. ◀

Proof of Theorem 5. We show the following problem is NP-hard.

▶ **Problem 3.** *Suppose we are given a set of n agents where $\mathbf{x}_1^{init}, \mathbf{x}_2^{init}, \ldots, \mathbf{x}_n^{init}$ denote their initial feature vectors. Does there exist a set of target points $\mathcal{P}^{final} \subseteq \mathbb{R}^d$ for which all the agents become truly qualified?*

We prove the NP-hardness by a reduction from the approximate version of the hitting set with equal-sized sets problem. As an instance of the hitting set problem we are given, $(\mathcal{F}, \mathcal{E})$ where $\mathcal{F} = \{S_1, \cdots, S_m\}$ is a collection of the subsets of $\mathcal{E} = \{e_1, e_2, \cdots, e_n\}$, and each set S_i has a size of $0 < s < n$, and our goal is to find a minimum size set $S^* \subseteq \mathcal{E}$ that intersects every set in $\mathcal{F}$. In order to show NP-hardness, we construct an instance of Problem 3 and prove: (1) If all the agents can become true positives by reaching to a set of target points $\mathcal{P}^{\text{final}} \subseteq \mathbb{R}^d$ that the mechanism designer selects, then there exists a hitting set of size at most $2k$. (2) If there exists a hitting set of size k then the mechanism designer can select a set of target points that encourages all the agents to become true positives. Since hitting set and set cover problems are equivalent and approximating set cover within a constant factor is NP-hard [8], this implies that Problem 3 is NP-hard.

First, we show how to construct an instance of Problem 3 from an instance of the Hitting Set problem. To do so, we determine the number of dimensions, initial positions of the agents, the movement costs, and a linear threshold function for the truly qualified. Let n be the number of elements of the Hitting Set instance, we construct an $n+1$-dimensional space where the first n dimensions are improvement and the last dimension is gaming. Consider sets $S_1, S_2, \ldots, S_m$ in the Hitting Set instance. For every set S_i, we consider agent i at initial point $\mathbf{x}_i^{\text{init}}$. In $\mathbf{x}_i^{\text{init}}$, the j^{th} coordinates such that $e_j \in S_i$ is set to 1. The rest of the first n coordinates are set to $2k$ and the last coordinate is 0. Also consider an extra agent $m+1$ at initial point $\mathbf{x}_{m+1}^{\text{init}}$ where all the first n coordinates are 0 and the last coordinate is $2k(n-s)+s$. Note that for all the agents $\sum_{j=1}^{n+1} \mathbf{x}_i^{\text{init}}[j] = 2k(n-s)+s$. Let the movement cost in all the dimensions $1 \le j \le n$ be $\frac{1}{2k}$ and in dimension $n+1$ be c such that $\frac{1}{2k(n-s)+s+1} < c < \frac{1}{2k(n-s)+s}$. Let $f^* : \sum_{j=1}^{n+1} \mathbf{x}[j] \ge 2k(n-s)+s+2k$. Therefore, all the agents are initially unqualified and at ℓ_1 distance of $2k$ from f^*.

Now we prove the first direction, i.e., if all the agents can become true positives by reaching to a set of target points $\mathcal{P}^{\text{final}} \subseteq \mathbb{R}^d$ that the mechanism designer selects, then there exists a hitting set of size at most $2k$. For all $1 \le i \le m+1$, let $\mathbf{p}_i \in \mathcal{P}^{\text{final}}$ denote the target point that $\mathbf{x}_i^{\text{init}}$ moves to and becomes true positive.

It consists of the following arguments: (i) For all $1 \le i \le m+1$, agent i receives utility 0 by reaching to $\mathbf{p}_i$. (ii) For all $1 \le i \le m$, agent $m+1$ does not afford to reach to $\mathbf{p}_i$. (iii) If $\mathbf{p}_{m+1}[j] \le 1$ for all $e_j \in S_i$, agent i moves to $\mathbf{p}_{m+1}$ and becomes a false positive. Therefore if all agents improve, for each $1 \le i \le m$, there exists $e_j \in S_i$ such that $\mathbf{p}_{m+1}[j] > 1$. (iv) In order for agent $m+1$ to afford to reach to target point $\mathbf{p}_{m+1}$, the number of coordinates $1 \le j \le m$ with value at least 1 must be at most $2k$. (v) These elements constitute a hitting set of size at most $2k$.

First, we prove argument (i). Each agent $1 \le i \le m+1$, is at ℓ_1 distance of $2k$ to f^*. To become qualified it needs to move $2k$ in the improvement dimensions. Since moving for a distance of $2k$ along the improvement dimensions costs a value of $(2k) \times (\frac{1}{2k}) = 1$, agent i makes a utility of 0.

Now, we move to argument (ii). Following up on the previous claim, to reach $\mathbf{p}_i$, agent $1 \leq i \leq m$ spends all of their movement budget in the improvement dimensions and cannot move a positive amount in the gaming dimension $n+1$. Therefore, $\mathbf{p}_i[n+1] = 0$ and $\sum_{j=1}^{n} \mathbf{p}_i[j] = 2k(n-s) + s + 2k$. In order for agent $m+1$ to reach such a target point, it needs to move a total of $2k(n-s) + s + 2k > 2k$ in the improvement dimensions, which costs more than 1 and it cannot afford.

Next, we prove argument (iii). Since $\mathbf{x}_{m+1}^{\text{init}}$ has an ℓ_1 distance of $2k$ from f^* and costs exactly a value of 1 to reach there, it can only afford to move along the improvement dimensions. Therefore, $\mathbf{p}_{m+1}[n+1] \leq 2k(n-s)+s$. Additionally, for $1 \leq j \leq n$, $\mathbf{p}_{m+1}[j] \leq 2k$; otherwise, agent $m+1$ cannot afford to reach to $\mathbf{p}_{m+1}$. Suppose $\mathbf{p}_{m+1}[j] \leq 1$ for all $e_j \in S_i$. Using this assumption, for agent i to reach $\mathbf{p}_{m+1}$ it only needs to pay cost of movement in dimension $n+1$, moving $2k(n-s)+s$ units and paying c per unit of movement. Since $(2k(n-s)+s) \times c < 1$, agent i makes a strictly positive utility. Therefore agent i prefers $\mathbf{p}_{m+1}$ over any other target point that makes it true positive which by argument (i) achieves utility 0.

Argument (iv) is straight-forward. To achieve non-negative utility each agent can afford to move at most $2k$ units along the improvement dimensions. Therefore, for the target point $\mathbf{p}_{m+1}$, the number of coordinates $1 \leq j \leq n$ with value at least 1 must be at most $2k$.

Argument (v) is a direct implication of the two previous arguments. By argument (iii), for each $1 \leq i \leq m$ there is an element $e_j \in S_i$ such that $\mathbf{p}_{m+1}[j] > 1$. By argument (iv), the number of coordinates $j \leq n$ such that $\mathbf{p}_{m+1}[j] > 1$ is at most $2k$ since otherwise agent $m+1$ cannot afford to reach to $\mathbf{p}_{m+1}$. Therefore, elements e_j such that $\mathbf{p}_{m+1}[j] > 1$ constitute a hitting set of size at most $2k$.

Now, we prove the reverse direction: if there exists a hitting set S^* of size k, the mechanism designer can select a set of target points that encourages all the agents to become true positives. To do so, we construct a set of target points $\mathcal{P}^{\text{final}} = \{\mathbf{p}_1, \ldots, \mathbf{p}_{m+1}\}$ that makes every agent to become true positive. For each agent i, $1 \leq i \leq m$, put a target point $\mathbf{p}_i$ whose first coordinate is $2k$ more than $\mathbf{x}_i^{\text{init}}$. For agent $m+1$, put a target point $\mathbf{p}_{m+1}$ whose coordinates j where $e_j \in S^*$ are set to 2 and the remaining agree with $\mathbf{x}_{m+1}^{\text{init}}$. Each target point $\mathbf{x}_i^{\text{init}}$ is set such that $\sum_{j=1}^{n+1} \mathbf{x}_i^{\text{init}}[j] = 2k(n-s) + s + 2k$. In order to show that every agent is able to improve, we argue that: (i) For all $1 \leq i \leq m$, agent i can afford to move to $\mathbf{p}_i$. Additionally, if agent i moves to any of the target points $\mathbf{p}_j$ where $1 \leq j \leq m$, it becomes true positive. (ii) For all $1 \leq i \leq m$, agent i cannot reach to $\mathbf{p}_{m+1}$. (iii) Agent $m+1$ moves to $\mathbf{p}_{m+1}$ and becomes true positive.

First, we prove argument (i): Agent i is at a distance of $2k$ from $\mathbf{p}_i$. It can afford to reach to $\mathbf{p}_i$ by paying a cost of $(2k) \times (\frac{1}{2k}) = 1$ and become true positive. In addition, if it moves to any of the other target points $\mathbf{p}_j$ where $1 \leq j \leq m$, since it has only moved along the improvement dimensions, it would become true positive.

Next, we prove argument (ii): We know that for each S_i, there exists an element $e_j \in S_i$ such that $\mathbf{p}_{m+1}[j] = 2$. As a result, the ℓ_1 distance of $\mathbf{x}_i^{\text{init}}$ and $\mathbf{p}_{m+1}$ is at least $(2k(n-s)+s+1) \times c > 1$. Therefore, for each $1 \leq i \leq m$, $\mathbf{x}_i^{\text{init}}$ cannot afford to reach to $\mathbf{p}_{m+1}$.

Finally, we prove argument (iii): First, we argue that agent $m+1$ cannot afford to reach to any of the target points $\mathbf{p}_i$ where $1 \leq i \leq m$. For each target point $\mathbf{p}_i$ where $1 \leq i \leq m$, $\mathbf{p}_i[n+1] = 0$ and $\sum_{j=1}^{n} \mathbf{p}_i[j] = 2k(n-s) + s + 2k$. In order for agent $m+1$ to reach such a target point, it needs to move a total of $2k(n-s) + s + 2k > 2k$ units in the improvement dimensions, which costs more than 1 and it cannot afford. In addition, agent $m+1$ can afford to move to $\mathbf{p}_{m+1}$, and by reaching there it becomes true positive.

As a result of the above arguments, given a hitting set of size k, the mechanism designer can select a set of target points that encourages all the agents to become true positives.

Combining the above two directions, shows that the problem of selecting a set of target points for which all the agents become truly qualified is NP-hard. ◀

B Proof of Theorem 26

In order to prove Theorem 26, we need to first show that the following observation and lemma hold.

▶ **Observation 28.** *Line $\mathbf{a}^*\mathbf{x} = b^*$ has a negative slope, i.e., each feature is defined so that larger is better. Therefore, after the points in $\mathcal{X}$ are sorted, if an agent $\mathbf{x}_j^{init}$ where $j < i$ reaches to any point $\mathbf{x'}_i \in [\mathbf{x}_{i,min}, \mathbf{x}_{i,max}]$, then $\mathbf{x}_j^{init}$ becomes true positive. On the other hand, for $j > i$, if $\mathbf{x}_j^{init}$ moves to any point $\mathbf{x'}_i \in [\mathbf{x}_{i,min}, \mathbf{x}_{i,max}]$, then $\mathbf{x}_j$ becomes false positive.*

▶ **Lemma 29.** *Consider a point $\mathbf{p}$ such that $\mathbf{p}[1] \geq \mathbf{x}_{i,min}[1]$, and another point $\mathbf{q} \in [\mathbf{x}_{i,min}, \mathbf{x}_{i,max}]$. Suppose $cost(\mathbf{x}_i^{init}, \mathbf{p}) = cost(\mathbf{x}_i^{init}, \mathbf{q})$. Then, for any $j > i$, it is the case that $cost(\mathbf{x}_j^{init}, \mathbf{p}) \leq cost(\mathbf{x}_j^{init}, \mathbf{q})$.*

Proof. Initially, if $\mathbf{p}[2] < \mathbf{x}_i^{\text{init}}[2]$, $\mathbf{p}$ is replaced with $(\mathbf{p}[1], \mathbf{x}_i^{\text{init}}[2])$. By doing so, $cost(\mathbf{x}_j^{\text{init}}, \mathbf{p})$ would not decrease. Hence, without loss of generality, we can assume $\mathbf{p}[2] \geq \mathbf{x}_i^{\text{init}}[2]$.

First, we show that $cost(\mathbf{x}_j^{\text{init}}, \mathbf{p}) \leq cost(\mathbf{x}_j^{\text{init}}, \mathbf{x}_{i,min}) + cost(\mathbf{x}_{i,min}, \mathbf{p})$, where the inequality holds when $\mathbf{x}_{i,min}[1] < \mathbf{x}_j^{\text{init}}[1] \leq \mathbf{p}[1]$.

$$\begin{aligned}
&cost(\mathbf{x}_j^{\text{init}}, \mathbf{x}_{i,min}) + cost(\mathbf{x}_{i,min}, \mathbf{p})\\
&= max\Big\{\mathbf{x}_{i,min}[1] - \mathbf{x}_j^{\text{init}}[1], 0\Big\} + \Big(\mathbf{x}_{i,min}[2] - \mathbf{x}_j^{\text{init}}[2]\Big) + \Big(\mathbf{p}[1] - \mathbf{x}_{i,min}[1]\Big) + \Big(\mathbf{p}[2] - \mathbf{x}_{i,min}[2]\Big)\\
&= max\Big\{\mathbf{x}_{i,min}[1] - \mathbf{x}_j^{\text{init}}[1], 0\Big\} + \Big(\mathbf{p}[1] - \mathbf{x}_{i,min}[1]\Big) + \Big(\mathbf{p}[2] - \mathbf{x}_j^{\text{init}}[2]\Big)
\end{aligned}$$

If $\mathbf{x}_j^{\text{init}}[1] \leq \mathbf{x}_{i,min}[1]$, the last equation above gets equal to $\Big(\mathbf{p}[1]-\mathbf{x}_j^{\text{init}}[1]\Big)+\Big(\mathbf{p}[2]-\mathbf{x}_j^{\text{init}}[2]\Big) = cost(\mathbf{x}_j^{\text{init}}, \mathbf{p})$. Otherwise, $\mathbf{x}_j^{\text{init}}[1] > \mathbf{x}_{i,min}[1]$ and the last equation above gets equal to $\Big(\mathbf{p}[1] - \mathbf{x}_{i,min}[1]\Big) + \Big(\mathbf{p}[2] - \mathbf{x}_j^{\text{init}}[2]\Big) > \Big(\mathbf{p}[1] - \mathbf{x}_j^{\text{init}}[1]\Big) + \Big(\mathbf{p}[2] - \mathbf{x}_j^{\text{init}}[2]\Big) = cost(\mathbf{x}_j^{\text{init}}, \mathbf{p})$. In any case, $cost(\mathbf{x}_j^{\text{init}}, \mathbf{p}) \leq cost(\mathbf{x}_j^{\text{init}}, \mathbf{x}_{i,min}) + cost(\mathbf{x}_{i,min}, \mathbf{p})$.

Next we argue that $cost(\mathbf{x}_{i,min}, \mathbf{p}) = cost(\mathbf{x}_{i,min}, \mathbf{q})$. First, since $\mathbf{p}[1] \geq \mathbf{x}_{i,min}[1]$ and $\mathbf{p}[2] \geq \mathbf{x}_{i,min}[2]$, then $cost(\mathbf{x}_i, \mathbf{p}) = cost(\mathbf{x}_i, \mathbf{x}_{i,min}) + cost(\mathbf{x}_{i,min}, \mathbf{p})$. Similarly, $cost(\mathbf{x}_i, \mathbf{q}) = cost(\mathbf{x}_i, \mathbf{x}_{i,min}) + cost(\mathbf{x}_{i,min}, \mathbf{q})$. Since $cost(\mathbf{x}_i, \mathbf{p}) = cost(\mathbf{x}_i, \mathbf{q})$, it is the case that $cost(\mathbf{x}_{i,min}, \mathbf{p}) = cost(\mathbf{x}_{i,min}, \mathbf{q})$.

Therefore,

$$\begin{aligned}
&cost(\mathbf{x}_j^{\text{init}}, \mathbf{p}) \leq cost(\mathbf{x}_j^{\text{init}}, \mathbf{x}_{i,min}) + cost(\mathbf{x}_{i,min}, \mathbf{p})\\
&\leq cost(\mathbf{x}_j^{\text{init}}, \mathbf{x}_{i,min}) + cost(\mathbf{x}_{i,min}, \mathbf{q})\\
&= max\Big\{\mathbf{x}_{i,min}[1] - \mathbf{x}_j^{\text{init}}[1], 0\Big\} + max\Big\{\mathbf{x}_{i,min}[2] - \mathbf{x}_j^{\text{init}}[2], 0\Big\} +\\
&max\Big\{\mathbf{q}[1] - \mathbf{x}_{i,min}[1], 0\Big\} + max\Big\{\mathbf{q}[2] - \mathbf{x}_{i,min}[2], 0\Big\}\\
&= max\Big\{\mathbf{x}_{i,min}[1] - \mathbf{x}_j^{\text{init}}[1], 0\Big\} + \Big(\mathbf{x}_{i,min}[2] - \mathbf{x}_j^{\text{init}}[2]\Big) + \Big(\mathbf{q}[2] - \mathbf{x}_{i,min}[2]\Big)\\
&= max\Big\{\mathbf{x}_{i,min}[1] - \mathbf{x}_j^{\text{init}}[1], 0\Big\} + \Big(\mathbf{q}[2] - \mathbf{x}_j^{\text{init}}[2]\Big)\\
&= max\Big\{\mathbf{q}[1] - \mathbf{x}_j^{\text{init}}[1], 0\Big\} + \Big(\mathbf{q}[2] - \mathbf{x}_j^{\text{init}}[2]\Big)\\
&= cost(\mathbf{x}_j^{\text{init}}, \mathbf{q})
\end{aligned}$$
◀

Proof of Theorem 26. Suppose not. Let $\mathbf{x}_1^{\mathrm{OPT}}, \ldots, \mathbf{x}_n^{\mathrm{OPT}}$ be an optimal solution that agrees with $\mathbf{x'}_1, \ldots, \mathbf{x'}_n$ on as large a suffix as possible, and let i be the largest index such that $\mathbf{x}_i^{\mathrm{OPT}} \neq \mathbf{x}'_i$ (so $\mathbf{x}_j^{\mathrm{OPT}} = \mathbf{x}'_j$ for all $j > i$).

First, note that $i \neq n$. This is because $\mathbf{x'}_n = \mathbf{x}_{n,min}$, which is the cheapest point that agent n can reach to become a true positive; moreover, any other point moving to $\mathbf{x}'_n$ is a true improvement. So, replacing $\mathbf{x}_n^{\mathrm{OPT}}$ with $\mathbf{x'}_n$ only helps.

Next, we claim that even if $i < n$, replacing $\mathbf{x}_i^{\mathrm{OPT}}$ with $\mathbf{x'}_i$ can only improve the optimal solution. First, if $cost(\mathbf{x}_i^{\mathrm{init}}, \mathbf{x}_i^{\mathrm{OPT}}) \geq cost(\mathbf{x}_i^{\mathrm{init}}, \mathbf{x'}_i)$ then replacing $\mathbf{x}_i^{\mathrm{OPT}}$ with $\mathbf{x'}_i$ only helps by the same argument as above and the fact that $\mathbf{x'}_i$ was chosen so that no agent $j > i$ manipulates to it; here we are using the fact that the suffixes of the two solutions agree. On the other hand, suppose that $cost(\mathbf{x}_i^{\mathrm{init}}, \mathbf{x}_i^{\mathrm{OPT}}) < cost(\mathbf{x}_i^{\mathrm{init}}, \mathbf{x'}_i)$ and $cost(\mathbf{x}_i^{\mathrm{init}}, \mathbf{x}_i^{\mathrm{OPT}}) \leq 1/c$. Since $\mathbf{x}_i^{\mathrm{init}}$ cannot become a false positive by moving to $\mathbf{x}_i^{\mathrm{OPT}}$, this means that $\mathbf{x}_i^{\mathrm{OPT}}[1] \geq \mathbf{x}_{i,min}[1]$. There exists a point $\mathbf{q} \in [\mathbf{x}_{i,min}, \mathbf{x}_{i,max}]$ such that $cost(\mathbf{x}_i^{\mathrm{init}}, \mathbf{x}_i^{\mathrm{OPT}}) = cost(\mathbf{x}_i^{\mathrm{init}}, \mathbf{q})$, which implies that $cost(\mathbf{x}_i^{\mathrm{init}}, \mathbf{q}) < cost(\mathbf{x}_i^{\mathrm{init}}, \mathbf{x'}_i)$. The reason that $\mathbf{q}$ was not selected as $\mathbf{x'}_i$ is that there exists an agent $\mathbf{x}_j^{\mathrm{init}}$ where $\mathbf{x}_j^{\mathrm{init}}$ moves to $\mathbf{q}$ and becomes false positive. By Observation 28, $j > i$. Hence, $cost(\mathbf{x}_j^{\mathrm{init}}, \mathbf{q}) < cost(\mathbf{x}_j^{\mathrm{init}}, \mathbf{x'}_j)$ and $cost(\mathbf{x}_j^{\mathrm{init}}, \mathbf{q}) \leq 1/c$. By Lemma 29, $cost(\mathbf{x}_j^{\mathrm{init}}, \mathbf{x}_i^{\mathrm{OPT}}) \leq cost(\mathbf{x}_j^{\mathrm{init}}, \mathbf{q})$, so $cost(\mathbf{x}_j^{\mathrm{init}}, \mathbf{x}_i^{\mathrm{OPT}}) < cost(\mathbf{x}_j^{\mathrm{init}}, \mathbf{x'}_j)$ and $cost(\mathbf{x}_j^{\mathrm{init}}, \mathbf{x}_i^{\mathrm{OPT}}) \leq 1/c$. Hence, $\mathbf{x}_j^{\mathrm{init}}$ is closer to $\mathbf{x}_i^{\mathrm{OPT}}$ compared to $\mathbf{x'}_j = \mathbf{x}_j^{\mathrm{OPT}}$ and so agent j would become a false positive under OPT, which contradicts the definition of OPT. So, this second case cannot occur.

Therefore, Algorithm 3 maximizes the number of true positives subject to having no false positives. ◀