The Strategic Perceptron

SABA AHMADI*, University of Maryland
HEDYEH BEYHAGHI†, Toyota Technological Institute at Chicago
AVRIM BLUM‡, Toyota Technological Institute at Chicago
KEZIAH NAGGITA§, Toyota Technological Institute at Chicago

The classical *Perceptron algorithm* provides a simple and elegant procedure for learning a linear classifier. In each step, the algorithm observes the sample's *position* and *label* and updates the current predictor accordingly if it makes a mistake. However, in presence of strategic agents that desire to be classified as positive and that are able to modify their position by a limited amount, the classifier may not be able to observe the true position of agents but rather a position where the agent pretends to be. Unlike the original setting with perfect knowledge of positions, in this situation the Perceptron algorithm fails to achieve its guarantees, and we illustrate examples with the predictor oscillating between two solutions forever, making an unbounded number of mistakes even though a perfect large-margin linear classifier exists. Our main contribution is providing a modified Perceptron-style algorithm which makes a bounded number of mistakes in presence of strategic agents with both ℓ_2 and weighted ℓ_1 manipulation costs. In our baseline model, knowledge of the manipulation costs (i.e., the extent to which an agent may manipulate) is assumed. In our most general model, we relax this assumption and provide an algorithm which learns and refines both the classifier and its cost estimates to achieve good mistake bounds even when manipulation costs are unknown.

CCS Concepts: • **Theory of computation → Online learning theory**; **Algorithmic mechanism design**.

Additional Key Words and Phrases: perceptron algorithm, strategic classification, online learning

ACM Reference Format:
Saba Ahmadi, Hedyeh Beyhaghi, Avrim Blum, and Keziah Naggita. 2021. The Strategic Perceptron. In *Proceedings of the 22nd ACM Conference on Economics and Computation (EC '21), July 18–23, 2021, Budapest, Hungary.* ACM, New York, NY, USA, 20 pages. https://doi.org/10.1145/3465456.3467629

1 INTRODUCTION

In machine learning, *strategic classification* deals with the problem of learning a classifier when the learner relies on data that is provided by strategic agents [4, 10]. For example, consider deciding eligibility of individuals for employment or education. In order to be considered eligible, individuals may engage in activities that do not truly change their qualifications, but affect the decision made.

*Research initiated during author's visit to Northwestern University. Author was supported in part by a research award from Amazon, and NSF CCF-1733556. Part of the research was done when author was visiting Toyota Technological Institute at Chicago.
†This work was done in part while the author was a Postdoctoral Researcher at Northwestern University, supported in part by the National Science Foundation under grant CCF-1618502.
‡This work was supported in part by the National Science Foundation under grants CCF-1815011 and CCF-1733556.
§This work was supported in part by the National Science Foundation under grant CCF-1815011.

EC '21, July 18–23, 2021, Budapest, Hungary

ACM ISBN 978-1-4503-8554-1/21/07…$15.00
https://doi.org/10.1145/3465456.3467629

In the aforementioned settings, these activities include job or college applicants carefully crafting their application materials and investing in interview or test preparations. In these scenarios, by using information about the classifier, individuals alter their features artificially by a limited amount to achieve their desirable outcome.

Strategic classification is particularly challenging in the online setting, where data points arrive in an arbitrary sequence, because the way that points manipulate may depend (in a discontinuous way) on the current classifier, and there is no useful source of unmanipulated data. More specifically, consider a standard online learning setting as follows. Individuals arrive one at a time, and based on the individual's features, the classifier predicts the individual as positive or negative. The learner is then told the correct classification and may update its classifier for the next round. The learner's goal is to minimize the number of mistakes made. Performing the same procedure in the strategic setting brings in several challenges. First, since the learner does not observe the *true* features, the update is done based on the individual's *manipulated* features. Therefore, at each point in time, the current classifier is built from manipulated data the learner has observed in the past. Second, each individual reacts to the *current* classifier. This means that the individuals' behaviors change over time and may be different from behavior of previous individuals with similar features. Moreover, because data arrives in an arbitrary order, there is no way to collect a "representative sample" of unmanipulated data by, say, classifying all examples as negative for an initial period. Finally, manipulation behavior may be a discontinuous function of the classifier's parameters: if an individual's cost to manipulate is slightly less than the benefit of being classified as positive then it will do so, but if it is slightly greater then it will not. Due of these issues, as we will show, standard learning algorithms that would make a limited number of mistakes in non-strategic settings may end up cycling and making unbounded number of mistakes; even if there exists a perfect classifier they may not find one.

Another challenge in online strategic classification is when the learner is unaware of the manipulation costs, which determine the extent to which agents will manipulate their features to achieve a positive classification. In this case, on top of estimating the individuals' real attributes based on the observed data, the learner also needs to estimate the costs. Unreasonable estimate of costs may lead to poor performance by the learner as the learner may not be able to distinguish if a classification mistake is due to an improper classifier or improper estimate of costs. This failure to distinguish correctly may lead to deterioration of the classifier and divergence from the optimal solution.

We study an online linear classification problem when the individuals are strategic. To isolate the effect of manipulation, we focus on finding a linear classifier when the unmanipulated data is linearly separable; i.e., the feature space is divided into two half spaces: with *positive* data points in one and *negative* data points in the other, and a nonzero margin between them. When individuals can manipulate, in each step, the arriving individual wishes to be classified positively. If the individual's feature vector $\mathbf{z}$ is not classified as positive with the true attributes, they may choose to suffer a cost and pretend to have a feature vector $\mathbf{x}$. More specifically, we consider utility-maximizing individuals, where utility is defined as value minus cost, who receive value 1 for being classified as positive and 0 for being classified as negative. We then consider two classes of cost functions: ℓ_2 costs (where cost is proportional to the Euclidean distance moved) and weighted ℓ_1 costs (where the cost of reaching a destination is the sum of separate costs paid in each coordinate direction). The ℓ_2 case represents settings where individuals when manipulating can take actions that affect multiple attributes. The ℓ_1 case represents settings where there is a specific action associated with each attribute. Note that in both cases, even though the *unmanipulated* data is linearly separable, the observed *manipulated* data points may no longer be separable.

Our Techniques and Results. The main contribution of this paper is solving the problem of online learning of linear separators in the strategic setting, making a bounded number of mistakes when the unmanipulated data is linearly separable by a nonzero margin. To do this, we build on and adapt the classic Perceptron algorithm [16], redesigning it to work in various strategic settings. This classic algorithm makes a bounded number of mistakes in the nonstrategic case when positive and negative data points are linearly separable. However, as mentioned earlier, in the strategic case it may cycle indefinitely (much like gradient descent for finding a Nash equilibrium) and make an unbounded number of mistakes; see Examples 1 and 2. Our main technique is to carefully design *surrogate* data points and feed them as the observed data to the algorithm. The role of the surrogate is to ensure that the algorithm is able to make positive progress each time it makes a mistake; however, defining it requires extra care. In particular, while it is not hard to show we can compute the *direction* that data points may have manipulated in, we can never be sure exactly how far (and we are particularly interested in the case that the amount by which data points can manipulate is large compared to the margin of separation). Another adaptation is to use a positive threshold for the dot product with the classifier's weight vector for a point to be classified positive.

Making use of the Perceptron algorithm, surrogate data points, and a positive linear threshold is central in all the algorithms designed in this paper. However, additional ideas are needed to handle subtleties of each specific setting. For example, for weighted ℓ_1 costs we need to take extra steps to make the manipulation direction unique and in line with the true classifier's weight vector, and in the unknown costs setting, we need to distinguish if the cost estimates are above or below the true costs. Another case is when the separating hyperplane does not cross the origin. In this case, the classic approach is to just add a fake coordinate in which each example has value 1, and then apply the Perceptron algorithm to those extended data points. However, when data is given by strategic agents, this reduction breaks down and we need to apply different ideas. There are also some results that hold for the non-strategic case that we do not know how to achieve, such as obtaining a mistake bound proportional to the hinge-loss of the best separator when data is not perfectly separable; for this setting we show examples where our algorithm fails and propose it as an open problem.

The main contributions of this paper are:

- We give an online learning algorithm robust to manipulation that finds a linear classifier in a bounded number of mistakes with the knowledge of costs. The number of mistakes is not much larger than the standard Perceptron bound in the non-strategic case for ℓ_2 costs and is reasonably bounded in other settings as well, see Theorems 1, 2 and 4.
- We give an online learning algorithm that generalizes the previous algorithm to unknown costs with a bounded number of mistakes. See Theorem 3.
- We generalize the algorithm for known ℓ_2 costs to the case of heterogeneous agents whose utility functions differ by a limited amount and give an online learning algorithm with bounded number of mistakes. See Corollary 1.

Related Work. The first studies on strategic classification focused on the offline setting; i.e., where the agents' true features come from a distribution. Brückner and Scheffer [4] and later Hardt et al. [10], formalized the strategic classification problem as a Stackelberg competition between a learner and an agent. They assume the learner has access to the distribution of agents' true features and their cost functions; and use this information to design near-optimal classifiers.

Dong et al. [8] initiated the study of strategic classification in the online setting where the learner does not know the distribution of agents' true features or their cost functions. A key difference between [8] and this paper is the assumption on the objective of the agents: we consider agents that wish to be classified as positive, whereas [8] considers agents that wish to increase their dot-product

with the hypothesis vector no matter how they are classified. Based on this assumption, in [8] the agents' behaviors are continuous in the hypothesis vector. However, in our model, a small change in the hypothesis vector can cause a drastic (discontinuous) change in agents' behavior. More particularly, as a consequence of agents' objective and utility structure, each agent can manipulate by a limited amount. If the classification hyperplane is closer than this amount, the agent would manipulate to be classified as positive; however, if it is slightly farther, the agent stays stationary. This discontinuity in the agent's behavior is common in mechanism design and occurs in other problems such as pricing and auction design.

Chen et al. [5] also study an online learning problem where agents can manipulate by a bounded distance. However, while there are similarities between the setting studied in [5] and our ℓ_2 cost model, explained in more detail in Section 2, [5] does not consider a fixed utility model and instead considers a regret term that is worst-case over agents that can manipulate by some bounded distance. As a result, their regret term may be arbitrarily high when the observed positions of positive and negative data points are inseparable, even if the unmanipulated points are linearly separable. Our algorithms, in contrast, can handle this inseparability during the learning procedure and make a bounded number of mistakes.

The goal of the papers mentioned so far is accuracy or minimizing loss. There are also papers that consider other objectives. Hu et al. [11] focus on a fairness objective and raise the issue that different populations of agents may have different manipulation costs. Braverman and Garg [3], by introducing noise in their classification, design algorithms where agents with different costs are better off not manipulating which tackles the fairness issue. Milli et al. [15] state that the accuracy that strategic classification seeks leads to a raised bar for agents who naturally are qualified and puts a burden on them to prove themselves. Kleinberg and Raghavan [12], Haghtalab et al. [9], Alon et al. [1], Bechavod et al. [2], Shavit et al. [17], and Miller et al. [14] focus on models in which the policy maker is interested in choosing a rule which incentivizes agent(s) to invest their effort into features that truly improve their qualification. Other papers considering incentives in machine learning include [6, 7, 13].

Organization of the Paper. Section 2 introduces the model and provides examples where the original Perceptron algorithm makes an unbounded number of mistakes. Sections 3 and 4 study the case where the cost of manipulation is known: Section 3 focuses on ℓ_2 costs and Section 4 on weighted ℓ_1 costs. Section 5 studies the unknown costs model. Section 6 studies the generalization of known manipulation costs to heterogeneous agents that have slightly different costs. Section 7 extends the results of Sections 3 and 4 where the separator of the unmanipulated data points crosses the origin to the general case. Finally, conclusions and some open problems are presented in Section 8.

2 MODEL AND PRELIMINARIES

In Section 2.1, we formally define our model. In Section 2.2, we overview the non-strategic setting. In Section 2.3, we provide examples where the original Perceptron algorithm makes an unbounded number of mistakes.

2.1 Model

We study an online classification problem in which a series of examples in $\mathbb{R}^d$ arrive one at a time. We think of examples as corresponding to d observable features of individuals who wish to be classified as positive. They have the ability to manipulate their observable features at some cost. Let $\mathbf{z}_t$ denote the t^{th} example before manipulation, and $\mathbf{x}_t$ denote the observed t^{th} example. We assume there exists a vector $\mathbf{w}^*$, such that for each unmanipulated positive example $\mathbf{z}_t$ we have $\mathbf{z}_t^T \mathbf{w}^* \geq 1$,

and for each unmanipulated negative example $\mathbf{z}_t$ we have $\mathbf{z}_t^T \mathbf{w}^* \leq -1$; i.e., a linear separator of margin $\gamma = 1/|\mathbf{w}^*|$. We use $|\mathbf{w}|$ to indicate the ℓ_2 norm of $\mathbf{w}$.

We assume individuals are utility maximizers, where utility is defined as value minus cost. Individuals have value 1 for being classified as positive, and 0 for being classified as negative. More formally, an agent with true coordinates $\mathbf{z}_t$ will move to $\mathbf{x}_t = \arg\max_x [value(\mathbf{x}) - cost(\mathbf{z}_t, \mathbf{x})]$ where $value(\mathbf{x}) = 1$ if $\mathbf{x}$ is classified positive by the current classifier and $value(\mathbf{x}) = 0$ if $\mathbf{x}$ is classified negative, and $cost(\mathbf{z}_t, \mathbf{x})$ refers to the cost of manipulation from $\mathbf{z}_t$ to $\mathbf{x}$. This implies if the agent can manipulate their features at cost at most 1 to change their classification from negative to positive, then they will do so in the cheapest way possible, otherwise they will not.

We consider two settings for cost of manipulation. In the first setting, $cost(\mathbf{z}_t, \mathbf{x})$ is proportional to the ℓ_2 distance of the two points $\mathbf{z}_t$ and $\mathbf{x}$; i.e., $cost(\mathbf{z}_t, \mathbf{x}) = c\sqrt{\sum_{i=1}^{d} (\mathbf{x}_i - \mathbf{z}_{t,i})^2}$, where c is the cost per unit of movement. We define $\alpha = 1/c$ as the maximum amount data points would be willing to move to achieve a positive classification.[1] We assume $0 \leq \alpha \leq R$ where $R = \max_t |\mathbf{z}_t|$. In the second setting, $cost(\mathbf{z}_t, \mathbf{x})$ is a weighted ℓ_1 metric, such that $cost(\mathbf{z}_t, \mathbf{x}) = \sum_{i=1}^{d} c_i |\mathbf{x}_i - \mathbf{z}_{t,i}|$. Similarly we define $\alpha_i = 1/c_i$ as the maximum amount data points would move along the i^{th} coordinate vector $\mathbf{e}_i$, where $0 \leq \alpha_i \leq R$. We consider both scenarios of known and unknown costs. In the unknown ℓ_2 costs we don't assume knowledge of c, and in unknown weighted ℓ_1 costs we do not assume knowledge of $c_1, \ldots, c_d$.

Generalizations. For the majority of the paper we study the above model. In Sections 6 and 7, we then present and analyze several generalizations. Section 6 studies heterogeneous agents with ℓ_2 costs where the costs per unit of movement (defined previously as c) for agents are slightly different. More particularly, we study the case where the maximum amount the agent arriving at time t can move, α_t (i.e. $1/c_t$, where c_t is the cost per unit of movement for agent t), is in the interval $[\alpha_{\min}, \alpha_{\max}]$ where $0 \leq \alpha_{\max} - \alpha_{\min} \leq \gamma/2$. The algorithm does not have access to α_t but knows the interval. Section 7 studies the case where the separating hyperplane of the unmanipulated data does not cross the origin. More particularly, there exists a separator $\mathbf{z}^T \mathbf{w}^* + b = 0$, such that for a positive example $\mathbf{z}_t$, $\mathbf{z}_t^T \mathbf{w}^* + b \geq 1$ and for a negative example $\mathbf{z}_t$, $\mathbf{w}^{*T} \mathbf{z}_t + b \leq -1$.

2.2 Non-Strategic Setting and the Perceptron Algorithm

As a reminder for the reader we provide the classical Perceptron algorithm here. This algorithm classifies all points with $\mathbf{x}_t^T \mathbf{w} \geq 0$ as positive, and the rest as negative; updating $\mathbf{w}$ when it makes a mistake. The total number of mistakes made by the algorithm is upper bounded by $R^2 |\mathbf{w}^*|^2$.

Algorithm 1: Perceptron Algorithm

$\mathbf{w} \leftarrow \mathbf{0}$;
for $t = 1, 2, \cdots$ **do**
  Given example $\mathbf{x_t}$, predict $sgn(\mathbf{x}_t^T \mathbf{w})$;
  if *the prediction was a mistake* **then**
    if $\mathbf{x_t}$ *was* $+$ **then** $\mathbf{w} \leftarrow \mathbf{w} + \mathbf{x}_t$;
    if $\mathbf{x}_t$ *was* $-$ **then** $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$;

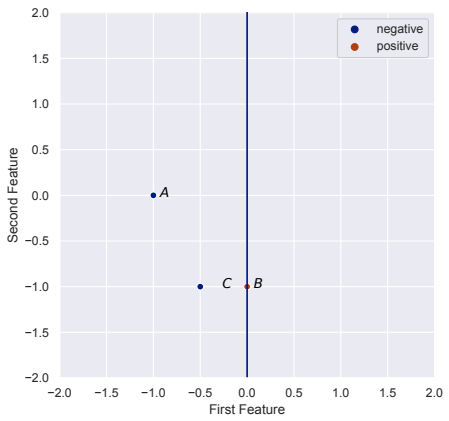
[1] For convenience we assume that if an agent is indifferent, i.e., its distance to the decision boundary is exactly α, then it will manipulate. Note that Chen et al. [5] also consider a model where individuals can move in a ball of fixed radius from their real position. However, they do not focus on a specific utility model.

Extension 1 (Perceptron with separator not crossing the origin)**.** A classic extension of Algorithm 1 to the case where examples are linearly separable, but not by a separator passing through the origin, is to create an extra "fake" coordinate. Specifically, assume there exists a separator $\mathbf{x}^T\mathbf{w}^* + b = 0$, such that for a positive example $\mathbf{x}_t$, $\mathbf{x}_t^T\mathbf{w}^* + b \geq 1$ and for a negative example $\mathbf{x}_t$, $\mathbf{w}^{*T}\mathbf{x}_t + b \leq -1$. Then Algorithm 1 is extended by adding an extra coordinate of value 1 to each example $\mathbf{x}_t$, replacing $\mathbf{x}_t$ with $(\mathbf{x}_t, 1)$. The bias term b is absorbed into $\mathbf{w}^*$ by adding an additional coordinate to $\mathbf{w}^*$, i.e. replacing $\mathbf{w}^*$ with $(\mathbf{w}^*, b)$. Now, for the positive examples, $\mathbf{x}_t^T\mathbf{w}^* \geq 1$, and for the negative examples $\mathbf{x}_t^T\mathbf{w}^* \leq -1$, and Algorithm 1 can be used as before.

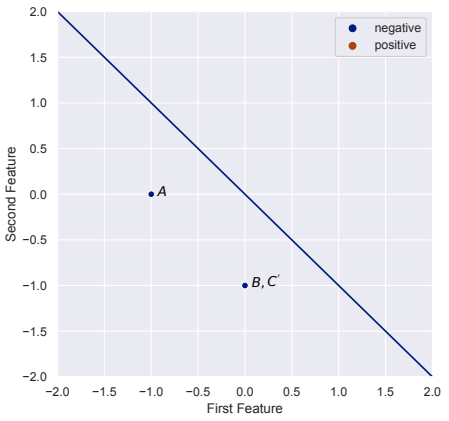
2.3 Failure of the Perceptron Algorithm in Strategic Settings

The Perceptron algorithm may make unbounded number of mistakes in the models considered in this paper even when a perfect classifier exists. The following example illustrates this in a setting with ℓ_2 cost.

Example 1. Consider three examples $A = (-1, 0)$, $B = (0, -1)$, and $C = (-0.5, -1)$ where A is negative, B is positive, and C is negative. Suppose that $\alpha = 0.5$. The following scenario of arrival of these examples makes the standard Perceptron algorithm (Algorithm 1) cycle between two classifiers and make an unbounded number of mistakes. Suppose A is the first example to arrive, then individuals B and C arrive respectively and repeatedly. After arrival of A, $\mathbf{w} = (1, 0)$. B does not need to manipulate as it is classified positive with the current classifier. However C manipulates to point $(0, -1)$ and the algorithm mistakenly classifies it as positive. As a consequence, $\mathbf{w}$ will be updated to $(1, 0) - (0, -1) = (1, 1)$. With the new classifier, B cannot manipulate to be classified positive because it has distance $\sqrt{2}/2$ from the decision boundary. So, B is misclassified as negative, causing an update to $\mathbf{w} = (1, 1) + (0, -1) = (1, 0)$ and the scenario repeats.



(a) Classic Perceptron correctly classifies B after update of $\mathbf{w}$ to $(1, 0)$. However, C now manipulates to the same location as B, which will cause a mistake and an update.



(b) After C manipulates to $C' = (0, -1)$, it is misclassified as positive and $\mathbf{w}$ is updated to $(1, 1)$. B cannot manipulate with current classifier because it is at distance $\sqrt{2}/2$ from the boundary.

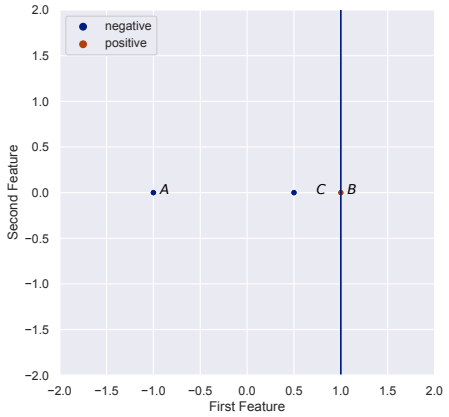
Fig. 1. Example 1 shows the classic Perceptron algorithm can make an unbounded number of mistakes in the strategic setting.

Note that Example 1 shows that the standard Perceptron algorithm can fail even if there exists a classifier that is perfect in the presence of manipulation.ÂăIn this example, the classifier given by $\mathbf{w} = (1, 0.5)$ works perfectly for the three points as B can manipulate to be classified positive but A and C cannot. The main reason the algorithm fails despite existence of a perfect classifier, is that the behavior of individuals *depends on the classifier* we are currently using and this can cause the algorithm to cycle indefinitely.

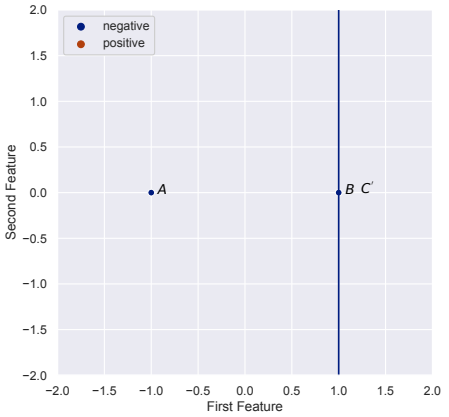
The failure of the Perceptron algorithm is not restricted to the ℓ_2 costs model. Example 1 with $\alpha = (0.6, 0)$ makes an unbounded number of mistakes in the ℓ_1 costs model as well.

The Perceptron algorithm as described above uses a threshold of 0. One may wonder if the usual extension to non-zero thresholds (Extension 1) might solve the strategic learning problem. In particular, any linearly separable dataset is still linearly separable in the presence of manipulation, by simply shifting the target separator by α. However, the example below shows that this extension also fails when the data points are strategic.

Example 2. Consider three examples $A = (-1, 0)$, $B = (1, 0)$, and $C = (0.5, 0)$ where A and C are negative and B is positive. Let $\alpha = 0.5$. Suppose A is the first example to arrive, then individuals B and C arrive respectively and repeatedly. After arrival of A, the separator is $1x_1 + 0x_2 - 1 \geq 0$. B does not need to manipulate as it is classified positive with the current classifier as shown in Figure 2a. However $C = (0.5, 0)$ manipulates to $(1, 0)$ and the algorithm mistakenly classifies it as positive as shown in Figure 2b. As a consequence, the separator is updated to $0x_1 + 0x_2 - 2 \geq 0$. With the new classifier, B is misclassified as negative but does not manipulate. The separator is then updated to $1x_1 + 0x_2 - 1 \geq 0$, and the process repeats indefinitely. Therefore the classifier keeps cycling and never correctly classifies the data even when a linear separator exists.



(a) The Perceptron algorithm updates $\mathbf{w}$ after misclassifying A. When B arrives it correctly classifies it.



(b) After C manipulates to $C' = (1, 0)$, it is now misclassified as positive by the current classifier and $\mathbf{w}$ and *bias* are updated accordingly.

Fig. 2. Example 2 shows the non-zero threshold Perceptron algorithm can make an unbounded number of mistakes in the strategic setting.

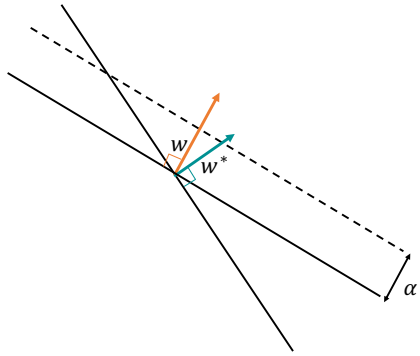


Fig. 3. Strategic Perceptron with known manipulation cost. The dashed line represents the manipulation hyperplane discussed in Observation 1. The margin of width α is the forbidden region, discussed in Observation 2.

3 KNOWN ℓ_2 COSTS

In this section, we provide an algorithm for the ℓ_2 costs setting. At a high level, there are two main ideas to modify and generalize the Perceptron algorithm for this setting. The first modification is raising the bar for a point to be classified as positive. Previously, a nonnegative dot product with the current classifier (a threshold of 0), sufficed for positive classification. However, in the new algorithm, the threshold is a strictly positive value depending on the cost of manipulation. The second modification is using a *surrogate* for the data points when the classifier updates. Interestingly, we only need to use a surrogate for negative points, and in this case the surrogate is a projection of the point in the opposite direction of manipulation, detected by the algorithm.

Overview of Algorithm 2. This algorithm is a generalization of the Perceptron algorithm which we call *strategic Perceptron*. The algorithm starts by predicting all points as positive until it makes a mistake. Note that during this period, individuals do not have incentive to manipulate. From that point on, the algorithm classifies all points with $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| - \alpha \geq 0$ as positive, and the rest as negative. Whenever the algorithm makes a mistake, the predictor $\mathbf{w}$ is updated with a surrogate value, $\tilde{\mathbf{x}}_t$, defined below.

Definition 1 ($\tilde{\mathbf{x}}_t$, surrogate data point in ℓ_2 setting)**.** We define surrogate data point, $\tilde{\mathbf{x}}_t$, as follows.

$$\tilde{\mathbf{x}}_t = \begin{cases} \mathbf{x}_t - \alpha \frac{\mathbf{w}}{|\mathbf{w}|}, & \text{if } \mathbf{x}_t \text{ is } - \text{ and } \frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} = \alpha; \\ \mathbf{x}_t, & \text{if } \mathbf{x}_t \text{ is } + \text{ and } \frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} = \alpha; \\ \mathbf{x}_t, & \text{if } \frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} > \alpha \text{ or } \frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} \leq 0. \end{cases}$$

Observation 1 (manipulation hyperplane)**.** In Algorithm 2, $\mathbf{x}_t$ is a manipulated example only if $\mathbf{x}^T \mathbf{w}/|\mathbf{w}| = \alpha$. The reason is as follows. In order to maximize utility, individuals move data points in direction of $\mathbf{w}$ and move the point the minimum amount to be classified as positive. Therefore, if with true features they are classified as negative, they only need to move to the line with dot product equal to α and moving to any other location contradicts with utility maximizing. In other words:

$$\mathbf{x}_t = \begin{cases} \mathbf{z}_t + \left(\alpha - \frac{\mathbf{z}_t^T \mathbf{w}}{|\mathbf{w}|}\right) \frac{\mathbf{w}}{|\mathbf{w}|}, & \text{if } 0 \leq \frac{\mathbf{z}_t^T \mathbf{w}}{|\mathbf{w}|} \leq \alpha; \\ \mathbf{z}_t, & \text{otherwise.} \end{cases}$$

Algorithm 2: Strategic Perceptron for ℓ_2 costs

$\mathbf{w} \leftarrow \mathbf{0}$;
for $t = 1, 2, \cdots$ **do**
  Given example $\mathbf{x}_t$:
  if $|\mathbf{w}|$ *is* 0 **then**
    predict +;
    if *the prediction was a mistake* **then** $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$;
  else
    predict $sgn(\frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} - \alpha)$;
    if *the prediction was a mistake and* $\mathbf{x}_t$ *was* + **then** $\mathbf{w} \leftarrow \mathbf{w} + \tilde{\mathbf{x}}_t$;
    if *the prediction was a mistake and* $\mathbf{x}_t$ *was* − **then** $\mathbf{w} \leftarrow \mathbf{w} - \tilde{\mathbf{x}}_t$;

Observation 2 (forbidden region)**.** No observed data point $\mathbf{x}_t$ will satisfy $0 < \mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| < \alpha$, and therefore $\tilde{\mathbf{x}}_t$ does not need to be defined for $0 < \mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| < \alpha$. The reason is that any such data point must either have manipulated to that position or not. If it manipulated, the manipulation was not rational since it did not help the data point to get classified as positive. If it did not manipulate, this was not rational either since the data point has a distance less than α from the classifier.

We show Algorithm 2 makes at most $(R+\alpha)^2|\mathbf{w}^*|^2$ mistakes. First, we need to prove the following lemmas hold.

Lemma 1. For any positive data point $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \geq 1$, and for any negative data point $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \leq -1$. Also, throughout the execution of Algorithm 2, $\mathbf{w}^T \mathbf{w}^* \geq 0$.

Proof. The proof uses induction. First, we show after the first update of the algorithm $\mathbf{w}^T \mathbf{w}^* > 0$. Second, we show if at the end of step $t-1$, $\mathbf{w}^T \mathbf{w}^* \geq 0$, then at step t, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \geq 1$ for positive points, and $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \leq -1$ for negative points. Finally, we show if $\mathbf{w}^T \mathbf{w}^* \geq 0$ at the end of step $t-1$, and $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \geq 0$ for positive points, and $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \leq 0$ for negative points, then $\mathbf{w}^T \mathbf{w}^* \geq 0$ at the end of step t.

The first step is straight-forward. Initially, $\mathbf{w} = 0$. While $\mathbf{w} = 0$, we have $\mathbf{x}_t^T \mathbf{w} = 0$, and arriving examples get classified positively. The first mistake occurs when a negative example $\mathbf{x}_t$ arrives and gets classified as positive. In this case, $\mathbf{w}$ gets updated to $\mathbf{w} - \mathbf{x}_t$. Since $\mathbf{x}_t^T \mathbf{w}^* \leq -1$, we conclude $(\mathbf{w} - \mathbf{x}_t)\mathbf{w}^* > 0$.

The second step is more involved. By definition of the surrogate values, for any points such that $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| \neq \alpha$, we have $\tilde{\mathbf{x}}_t = \mathbf{x}_t$. By Observation 1, these points are not manipulated, i.e., $\mathbf{x}_t = \mathbf{z}_t$. This implies $\tilde{\mathbf{x}}_t = \mathbf{z}_t$ and therefore the claim holds. Thus, we only need to argue for the points on the hyperplane $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| = \alpha$. Consider such data points. For the positive data points, we have, $\tilde{\mathbf{x}}_t = \mathbf{x}_t = \mathbf{z}_t + \beta \cdot \mathbf{w}/|\mathbf{w}|$, where $0 \leq \beta \leq \alpha$. Therefore, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* = \mathbf{z}_t^T \mathbf{w}^* + \beta \cdot \mathbf{w}^T \mathbf{w}^*/|\mathbf{w}| \geq \mathbf{z}_t^T \mathbf{w}^* \geq 1$. The first inequality holds since by assumption of this step, $\mathbf{w}^T \mathbf{w}^* \geq 0$. On the other hand, for the negative data points we have $\tilde{\mathbf{x}}_t = \mathbf{x}_t - \alpha \cdot \mathbf{w}/|\mathbf{w}|$, where $\mathbf{x}_t = \mathbf{z}_t + \beta \cdot \mathbf{w}/|\mathbf{w}|$ and $0 \leq \beta \leq \alpha$. This implies $\tilde{\mathbf{x}}_t = \mathbf{z}_t + (\beta - \alpha) \cdot \mathbf{w}/|\mathbf{w}|$. By multiplying with $\mathbf{w}^*$, we get $\tilde{\mathbf{x}}_t^T \mathbf{w}^* = \mathbf{z}_t^T \mathbf{w}^* + (\beta - \alpha) \cdot \mathbf{w}^T \mathbf{w}^*/|\mathbf{w}| \leq \mathbf{z}_t^T \mathbf{w}^* \leq -1$.

The final step is again straight-forward. Whenever $\mathbf{w}$ is updated, for positive points, $\mathbf{w}$ gets updated to $\mathbf{w} + \tilde{\mathbf{x}}_t$, where both $\mathbf{w}$ and $\tilde{\mathbf{x}}_t$ have nonnegative dot product with $\mathbf{w}^*$. For negative points, $\mathbf{w}$ gets updated to $\mathbf{w} - \tilde{\mathbf{x}}_t$, where $\mathbf{w}$ has a nonnegative and $\tilde{\mathbf{x}}_t$ has a negative dot product with $\mathbf{w}^*$. □

Lemma 2. When Algorithm 2 makes a mistake on a positive example $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T\mathbf{w} \leq 0$; and when it makes a mistake on a negative example $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T\mathbf{w} \geq 0$.

Proof. The algorithm makes a mistake on a positive example only if $\mathbf{x}^T\mathbf{w}/|\mathbf{w}| < \alpha$. By Observation 2, for no points, $0 < \mathbf{x}^T\mathbf{w}/|\mathbf{w}| < \alpha$. Therefore, for any positive example that the algorithm makes a mistake on, $\mathbf{x}^T\mathbf{w} \leq 0$. By Definition 1, $\tilde{\mathbf{x}}_t = \mathbf{x}_t$ for all positive examples. Therefore, $\mathbf{x}^T\mathbf{w} \leq 0$ implies $\tilde{\mathbf{x}}^T\mathbf{w} \leq 0$. For negative examples, the algorithm makes a mistake only if $\mathbf{x}^T\mathbf{w}/|\mathbf{w}| \geq \alpha$. If the inequality is strict, i.e., $\mathbf{x}^T\mathbf{w}/|\mathbf{w}| > \alpha$, by Definition 1, $\tilde{\mathbf{x}}_t = \mathbf{x}_t$, and therefore $\tilde{\mathbf{x}}_t^T\mathbf{w} \geq 0$. If $\mathbf{x}^T\mathbf{w}/|\mathbf{w}| = \alpha$, again using Definition 1, we have $\tilde{\mathbf{x}}^T\mathbf{w} = 0$. □

Next, we show the following theorem holds which gives a bound on the number of mistakes. Proof of the following theorem is along the lines of the proof of the classic Perceptron algorithm.

Theorem 1. Algorithm 2 makes at most $(R+\alpha)^2|\mathbf{w}^*|^2$ mistakes in the strategic setting with known ℓ_2 costs, when the unmanipulated data points $\mathbf{z}_t$ satisfy $\mathbf{z}_t^T\mathbf{w}^* \geq 1$ for positive examples and $\mathbf{z}_t^T\mathbf{w}^* \leq -1$ for negative examples, and $R = \max_t |\mathbf{z}_t|$.

Proof. We keep track of two quantities, $\mathbf{w}^T\mathbf{w}^*$ and $|\mathbf{w}|^2$. First, we show that each time we make a mistake, $\mathbf{w}^T\mathbf{w}^*$ increases by at least 1. If we make a mistake on a positive example then,

$$(\mathbf{w} + \tilde{\mathbf{x}}_t)^T\mathbf{w}^* = \mathbf{w}^T\mathbf{w}^* + \tilde{\mathbf{x}}_t^T\mathbf{w}^* \geq \mathbf{w}^T\mathbf{w}^* + 1;$$

where the last inequality holds by Lemma 1. Similarly, if we make a mistake on a negative example,

$$(\mathbf{w} - \tilde{\mathbf{x}}_t)^T\mathbf{w}^* = \mathbf{w}^T\mathbf{w}^* - \tilde{\mathbf{x}}_t^T\mathbf{w}^* \geq \mathbf{w}^T\mathbf{w}^* + 1.$$

Next, on each mistake we claim that $|\mathbf{w}|^2$ increases by at most $(R+\alpha)^2$. If we make a mistake on a positive example $\mathbf{x}_t$, then we have:

$$(\mathbf{w} + \tilde{\mathbf{x}}_t)^T(\mathbf{w} + \tilde{\mathbf{x}}_t) = |\mathbf{w}|^2 + 2\tilde{\mathbf{x}}_t^T\mathbf{w} + |\tilde{\mathbf{x}}_t|^2 \leq |\mathbf{w}|^2 + |\tilde{\mathbf{x}}_t|^2 \leq |\mathbf{w}|^2 + (R+\alpha)^2.$$

To understand the middle inequality note that by Lemma 2, when a mistake is made on a positive example $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T\mathbf{w} \leq 0$. The last inequality comes from $R = \max_t |\mathbf{z}_t|$ implies $\max_t |\tilde{\mathbf{x}}_t| \leq R + \alpha$.

Similarly, if we make a mistake on a negative example $\mathbf{x}_t$, then we have:

$$(\mathbf{w} - \tilde{\mathbf{x}}_t)^T(\mathbf{w} - \tilde{\mathbf{x}}_t) = |\mathbf{w}|^2 - 2\tilde{\mathbf{x}}_t^T\mathbf{w} + |\tilde{\mathbf{x}}_t|^2 \leq |\mathbf{w}|^2 + |\tilde{\mathbf{x}}_t|^2 \leq |\mathbf{w}|^2 + (R+\alpha)^2.$$

By Lemma 2, when a mistake is made on a negative example $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T\mathbf{w} \geq 0$, which implies the middle inequality.

Finally, if the algorithm makes M mistakes, then $\mathbf{w}^T\mathbf{w}^* \geq M$ and $|\mathbf{w}|^2 \leq M(R+\alpha)^2$, or equivalently, $|\mathbf{w}| \leq (R+\alpha)\sqrt{M}$. Using Cauchy-Schwartz inequality, $\mathbf{w}^T\mathbf{w}^*/|\mathbf{w}^*| \leq |\mathbf{w}|$, we have

$$M/|\mathbf{w}^*| \leq (R+\alpha)\sqrt{M} \implies \sqrt{M} \leq (R+\alpha)|\mathbf{w}^*| \implies M \leq (R+\alpha)^2|\mathbf{w}^*|^2.$$

□

4 KNOWN WEIGHTED ℓ_1 COSTS

In this section, we provide an algorithm for the weighted ℓ_1 costs setting. Unlike the ℓ_2 case, the modifications to the classical Perceptron algorithm in Algorithm 2 do not suffice; and our algorithm for this setting is more involved. Here is the key difference: In the ℓ_2 costs setting, the individuals always manipulate in direction of the current classifier $\mathbf{w}$. However, in the weighted ℓ_1 setting this is no longer the case. This brings up two challenges to our approach. First, there may be multiple utility maximizing manipulation directions. Second, the manipulation direction may have a negative dot product with $\mathbf{w}^*$. We overcome these two challenges and provide an algorithm for this setting.

As a reminder, in the weighted ℓ_1 costs setting, there are coordinate unit vectors $\{\mathbf{e}_1, \cdots, \mathbf{e}_d\}$ with cost of manipulation $1/\alpha_i$ along $\mathbf{e}_i$. We need to make one further assumption for this setting. We assume for all $1 \leq i \leq d$, $\mathbf{e}_i^T \mathbf{w}^* \geq 0$. In other words, we assume that each feature is defined so that larger is better. This is natural for settings such as hiring, admissions, loan applications, etc.

Overview of Algorithm 3. The algorithm starts by predicting all points as positive until it makes a mistake. Note that during this period, individuals do not have incentive to manipulate. From that point on, the algorithm classifies all points $\mathbf{x}_t$ such that $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| - \alpha_i \mathbf{w}^T \mathbf{e}_i/|\mathbf{w}| \geq 0$ as positive, and the rest as negative; where $\mathbf{e}_i$ is the manipulation direction which will be defined later. Similar to Algorithm 2, whenever the algorithm makes a mistake, the predictor $\mathbf{w}$ is updated with a surrogate value, $\tilde{\mathbf{x}}_t$, in Definition 2.

Compared to Algorithm 2, we have two further steps. As discussed above, the first challenge with weighted ℓ_1 costs is that with an arbitrary $\mathbf{w}$, there may be multiple utility maximizing manipulation directions, and we may not be able to distinguish along which vector individuals manipulated. Since in the weighted ℓ_1 costs setting, the cost of manipulation can be written as a convex combination of costs in coordinate vectors, there always exists a coordinate vector, $\mathbf{e}_i$, such that manipulating along that is utility maximizing. Consider all the coordinate vectors like $\mathbf{e}_j$ that are utility maximizing, i.e., have the highest $\alpha_j \cdot \mathbf{w}^T \mathbf{e}_j/|\mathbf{w}|$. To make the manipulation direction unique, we add a *tie-breaking step* to the algorithm. This step adds a small multiple $\eta > 0$, of an arbitrary utility maximization coordinate vector $\mathbf{e}_i$, to $\mathbf{w}$ to break the tie. Note that any positive value of η breaks the tie. We set this value in our analysis purposes in Theorem 2 in a way to make sure the number of mistakes our algorithm makes does not increase much.

We need to add another step to address the second challenge: With an arbitrary $\mathbf{w}$ the direction that the individuals manipulate along may not have a positive dot product with $\mathbf{w}^*$, i.e., the individuals may choose to move along one of the vectors $\{-\mathbf{e}_1, \cdots, -\mathbf{e}_d\}$. In order to incentivize individuals to only manipulate along $\{\mathbf{e}_1, \cdots, \mathbf{e}_d\}$, and not $\{-\mathbf{e}_1, \cdots, -\mathbf{e}_d\}$, we do the following *correction step* after each update. If $\mathbf{e}_j^T \mathbf{w} < 0$ for any $\mathbf{e}_j \in \{\mathbf{e}_1, \cdots, \mathbf{e}_d\}$, we set the j^{th} coordinate of $\mathbf{w}$ to 0 by adding the smallest multiple of $\mathbf{e}_j$, denoted by μ_j, to $\mathbf{w}$ to make $\mathbf{e}_j^T \mathbf{w}$ nonnegative. Therefore, $\mu_j = 0$ if $\mathbf{e}_j^T \mathbf{w} \geq 0$, and $\mu_j = -\mathbf{e}_j^T \mathbf{w}$, otherwise; implying $\forall j \; \mu_j \geq 0$.

With the unique manipulation direction, similar to the ℓ_2 costs setting, we are now able to choose a surrogate value along the manipulation direction.

Definition 2 ($\tilde{\mathbf{x}}_t$, surrogate data point in weighted ℓ_1 setting)**.** Let $\mathbf{e}_i$ be the *unique* utility maximizing coordinate vector, i.e., $i = \arg\max_j \alpha_j \mathbf{w}^T \mathbf{e}_j/|\mathbf{w}|$. We define surrogate data point, $\tilde{\mathbf{x}}_t$, as follows.

$$\tilde{\mathbf{x}}_t = \begin{cases} \mathbf{x}_t - \mathbf{e}_i \cdot \alpha_i, & \text{if } \mathbf{x}_t \text{ is } - \text{ and } \frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} = \alpha_i \cdot \frac{\mathbf{w}^T \mathbf{e}_i}{|\mathbf{w}|}; \\ \mathbf{x}_t, & \text{otherwise.} \end{cases}$$

Lemma 3. $\mu_j \leq R + \alpha_j$.

Proof. We can show at the end of each round, $\mathbf{e}_j^T \mathbf{w} \geq 0$. Initially, $\mathbf{w} = 0$, therefore $\mathbf{e}_j^T \mathbf{w} = 0$. Suppose at the end of round $t-1$, $\mathbf{e}_j^T \mathbf{w} \geq 0$. Assume in round t, $\mathbf{w}$ gets updated by adding or subtracting $\tilde{\mathbf{x}}_t$ or $\mathbf{x}_t$. By assumption, the j^{th} coordinate of $\mathbf{x}_t$ is in $[-R, R]$, and therefore the j^{th} coordinate of $\tilde{\mathbf{x}}_t$ is in $[-R - \alpha_j, R + \alpha_j]$. Taken together, $\mu_j \leq R + \alpha_j$. Note that by adding $\eta \mathbf{e}_i$ to $\mathbf{w}$, $\mathbf{e}_j^T \mathbf{w}$ remains nonnegative. □

The following theorem upper bounds the number of mistakes made by Algorithm 3.

Theorem 2. Consider a sequence of examples before manipulation $\mathbf{z}_1, \mathbf{z}_2, \cdots$, which are observed as $\mathbf{x}_1, \mathbf{x}_2, \cdots$. Consider vector $\mathbf{w}^*$ such that $\mathbf{z}_t^T \mathbf{w}^* \geq 1$ for positive examples, and $\mathbf{z}_t^T \mathbf{w}^* \leq -1$

Algorithm 3: Strategic Perceptron for weighted ℓ_1 costs

$\mathbf{w} \leftarrow \mathbf{0}$;
for $t = 1, 2, \cdots$ **do**
  Given example $\mathbf{x}_t$:
  if $|\mathbf{w}|$ *is* 0 **then**
    predict +;
    if *the prediction was a mistake* **then**
      $\mathbf{w} \leftarrow \mathbf{w} - \mathbf{x}_t$;
      /* Correction Step */
      for $j = 1, 2, \cdots, d$ **do**
        $\mathbf{w} \leftarrow \mathbf{w} + \mu_j \mathbf{e}_j$, where $\mu_j = \max(0, -\mathbf{e}_j^T \mathbf{w})$;
      /* Tie-breaking Step */
      $i \leftarrow \arg\max_j \alpha_j \cdot \frac{\mathbf{w}^T \mathbf{e}_j}{|\mathbf{w}|}$;
      $\mathbf{w} \leftarrow \mathbf{w} + \eta \mathbf{e}_i$;
  else
    predict $sgn(\frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} - \alpha_i \cdot \frac{\mathbf{w}^T \mathbf{e}_i}{|\mathbf{w}|})$;
    if *the prediction was a mistake and* $\mathbf{x}_t$ *was* + **then** $\mathbf{w} \leftarrow \mathbf{w} + \tilde{\mathbf{x}}_t$;
    if *the prediction was a mistake and* $\mathbf{x}_t$ *was* − **then** $\mathbf{w} \leftarrow \mathbf{w} - \tilde{\mathbf{x}}_t$;
    /* Correction Step */
    for $j = 1, 2, \cdots, d$ **do**
      $\mathbf{w} \leftarrow \mathbf{w} + \mu_j \mathbf{e}_j$ where $\mu_j = \max(0, -\mathbf{e}_j^T \mathbf{w})$;
    /* Tie-breaking Step */
    $i \leftarrow \arg\max_j \alpha_j \cdot \frac{\mathbf{w}^T \mathbf{e}_j}{|\mathbf{w}|}$;
    $\mathbf{w} \leftarrow \mathbf{w} + \eta \mathbf{e}_i$;

for negative examples. Algorithm 3 makes at most $(1 + (d+1)(R+\alpha)^2)|\mathbf{w}^*|^2$ mistakes, where $R = \max_t |\mathbf{z}_t|$, and $\alpha = \max\{\alpha_1, \cdots, \alpha_d\}$.

Proof. Similar to the proof of Theorem 1, we keep track of two quantities $\mathbf{w}^T\mathbf{w}^*$ and $|\mathbf{w}|^2$. First, we show each time a mistake is made, $\mathbf{w}^T\mathbf{w}^*$ increases by at least 1. Then we find an upper bound on the increase of $|\mathbf{w}|^2$.

Starting from the current $\mathbf{w}$, the algorithm follows three steps to update: addition/subtraction of $\tilde{\mathbf{x}}_t$, the correction step, and the tie-breaking step. As in the algorithm $\mathbf{e}_i$ is the manipulation direction.

If the algorithm makes a mistake on a positive example the new value of $\mathbf{w}$ is $\mathbf{w} + \tilde{\mathbf{x}}_t + \eta\mathbf{e}_i + \sum_j \mu_j \mathbf{e}_j$. Therefore,

$$\left(\mathbf{w} + \tilde{\mathbf{x}}_\mathbf{t} + \eta\mathbf{e}_i + \sum_j \mu_j \mathbf{e}_j\right)^T \mathbf{w}^* = \mathbf{w}^T\mathbf{w}^* + \tilde{\mathbf{x}}_t^T\mathbf{w}^* + \eta\mathbf{e}_i^T\mathbf{w}^* + \sum_j \mu_j \mathbf{e}_j^T\mathbf{w}^* \geq \mathbf{w}^T\mathbf{w}^* + 1;$$

where the inequality holds because first using the ideas from Lemma 1, $\tilde{\mathbf{x}}_t^T\mathbf{w}^* \geq 1$ for the positive examples the algorithm makes a mistake on and $\tilde{\mathbf{x}}_t^T\mathbf{w}^* \leq -1$ for the negative examples the algorithm makes a mistake on, and second, for all j, $\mathbf{e}_j^T\mathbf{w}^* \geq 0$ by assumption, and $\mu_j \geq 0$.

Similarly, If the algorithm makes a mistake on a negative example, we have:

$$\left(\mathbf{w} - \tilde{\mathbf{x}}_t + \eta\mathbf{e}_i + \sum_j \mu_j \mathbf{e}_j\right)^T \mathbf{w}^* = \mathbf{w}^T\mathbf{w}^* - \tilde{\mathbf{x}}_t^T\mathbf{w}^* + \eta\mathbf{e}_i^T\mathbf{w}^* + \sum_j \mu_j \mathbf{e}_j^T\mathbf{w}^* \geq \mathbf{w}^T\mathbf{w}^* + 1.$$

Next, on each mistake we claim $|\mathbf{w}|^2$ increases by at most $(d+1)(R+\alpha)^2 + 1$. If the algorithm makes a mistake on a positive example, we have:

$$\begin{aligned}
&\left|\mathbf{w} + \tilde{\mathbf{x}}_t + \eta\mathbf{e}_i + \sum_j \mu_j \mathbf{e}_j\right|^2 \\
&= |\mathbf{w} + \tilde{\mathbf{x}}_t + \eta\mathbf{e}_i|^2 + \left|\sum_j \mu_j \mathbf{e}_j\right|^2 + 2\left(\sum_j \mu_j \mathbf{e}_j\right)^T (\mathbf{w} + \tilde{\mathbf{x}}_t + \eta\mathbf{e}_i) \\
&= |\mathbf{w} + \tilde{\mathbf{x}}_t + \eta\mathbf{e}_i|^2 + \sum_j |\mu_j \mathbf{e}_j|^2 + 2\sum_j \mu_j \mathbf{e}_j^T(\mathbf{w} + \tilde{\mathbf{x}}_\mathbf{t} + \eta\mathbf{e}_i) \\
&\leq |\mathbf{w} + \tilde{\mathbf{x}}_\mathbf{t} + \eta\mathbf{e}_i|^2 + \sum_j |\mu_j \mathbf{e}_j|^2 + 2\sum_j \eta\mu_j \mathbf{e}_j^T\mathbf{e}_i \\
&= |\mathbf{w} + \tilde{\mathbf{x}}_\mathbf{t} + \eta\mathbf{e}_i|^2 + \sum_j |\mu_j|^2 + 2\eta\mu_i \\
&= |\mathbf{w}|^2 + |\tilde{\mathbf{x}}_t|^2 + |\eta\mathbf{e}_i|^2 + 2\mathbf{w}^T\tilde{\mathbf{x}}_t + 2\eta\mathbf{w}^T\mathbf{e}_i + 2\eta\tilde{\mathbf{x}}_t^T\mathbf{e}_i + \sum_j |\mu_j|^2 + 2\eta\mu_i \\
&\leq |\mathbf{w}|^2 + (R+\alpha)^2 + \eta^2 + 0 + 2\eta|\mathbf{w}| + 2\eta(R+\alpha) + d(R+\alpha)^2 + 2\eta(R+\alpha) \\
&\leq |\mathbf{w}|^2 + (d+1)(R+\alpha)^2 + \eta^2 + \eta(2|\mathbf{w}| + 4(R+\alpha)) \\
&\leq |\mathbf{w}|^2 + (d+1)(R+\alpha)^2 + 1/4 + 1/2 \\
&\leq |\mathbf{w}|^2 + (d+1)(R+\alpha)^2 + 1;
\end{aligned}$$

where the first equality is the result of expansion. The second uses $\mathbf{e}_j^T\mathbf{e}_k = 0$ for $j \neq k$. The inequality in the third row uses $\mu_j = 0$ when $\mathbf{e}_j^T(\mathbf{w} + \tilde{\mathbf{x}}_t) \geq 0$, and $\mu_j > 0$ when $\mathbf{e}_j^T(\mathbf{w} + \tilde{\mathbf{x}}_t) < 0$, implying $\mu\mathbf{e}_j^T(\mathbf{w} + \tilde{\mathbf{x}}_t) \leq 0$. The fourth row uses $\mathbf{e}_j^T\mathbf{e}_k = 0$ for $k \neq j$ and $\mathbf{e_j}^T\mathbf{e_j} = 1$. The fifth row is the result of expansion. The sixth row substitutes each term with an upper bound using $|\tilde{\mathbf{x}}_t| \leq R + \alpha$ and $\mathbf{w}^T\tilde{\mathbf{x}}_t \leq 0$, similar to the arguments from Lemma 2, and $\mu_j \leq R + \alpha$, by Lemma 3. The eighth row results by setting $\eta = \frac{1}{4|\mathbf{w}|+8(R+\alpha)+2}$. The last row sums up and upper bounds similar terms.

Similarly, if the algorithm makes a mistake on a negative example, we have:

$$\left|\mathbf{w} - \tilde{\mathbf{x}}_t + \eta\mathbf{e}_i + \sum_j \mu_j \mathbf{e}_j\right|^2 \leq |\mathbf{w}|^2 + (d+1)(R+\alpha)^2 + 1.$$

Therefore, after each mistake, $|\mathbf{w}|^2$ increases by at most $(d+1)(R+\alpha)^2 + 1$. The rest of the proof is similar to the proof of Theorem 1, concluding that the total number of mistakes is at most $((d+1)(R+\alpha)^2 + 1)|\mathbf{w}^*|^2$. □

5 UNKNOWN COSTS

The main result of this section is generalizing our algorithms to the unknown costs setting. The generalization holds for ℓ_2 costs . However, it does not extend fully to weighted ℓ_1 costs and only

works for a specific case. The algorithm for unknown ℓ_2 costs is presented in Section 5.1. The case of unknown ℓ_1 costs is studied in Section 5.2.

5.1 ℓ_2 Costs

In this section, we provide an algorithm that makes at most a bounded number of mistakes when the manipulation cost, $1/\alpha$, is unknown. Algorithm 2 is used as a subroutine to evaluate our estimate of α. First, we show Algorithm 2 works efficiently if the estimated value, α', is in proximity of the real value (when α' is in the interval of length $\gamma/2$ below α). Using this idea we can run a linear search for α with step size $\gamma/2$. However, we show we can do better than a linear search. The key ingredient that lets us outperform the linear search is the ability to distinguish whether the estimate is below or above the real value. Using this idea we run a binary search to find a proper estimate and come up with an efficient algorithm.

For convenience, we will present the algorithm assuming γ is known. At the end we show how to remove this assumption. Below, we explain these steps more formally.

Case 1: $0 \leq \alpha - \alpha' \leq \gamma/2$. First, we consider the case of $0 \leq \alpha - \alpha' \leq \gamma/2$. Suppose Algorithm 2 takes α' instead of α as input. Also, suppose $\tilde{\mathbf{x}}_t$ is defined with respect to α' instead of α. In Proposition 1, we show if $0 \leq \alpha - \alpha' \leq \gamma/2$, Algorithm 2 with these modifications, makes at most $4(R + \alpha' + \gamma/2)^2|\mathbf{w}^*|^2$ mistakes. We need the following two lemmas for proving the proposition. Proofs of Lemma 4 Lemma 5 are along the lines of proofs of Lemmas 1 and 2 respectively.

Lemma 4. Consider data points $\tilde{\mathbf{x}}_t$ as defined in Definition 1 w.r.t. α' such that $0 \leq \alpha - \alpha' \leq \gamma/2$. These data points are $1/2$-separable; i.e., for positive data points, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \geq 1/2$; and for negative data points, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \leq -1/2$. Also, throughout the execution of Algorithm 2 with α', $\mathbf{w}^T \mathbf{w}^* \geq 0$.

Proof. The proof uses the same three steps as Lemma 1. The first and the third steps are identical to Lemma 1. Here, we argue for the second step, i.e., if at the end of step $t-1$, $\mathbf{w}^T \mathbf{w}^* \geq 0$, then at step t, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \geq 1/2$ for positive points, and $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \leq -1/2$ for negative points.

When Algorithm 2 is run with α', by Definition 1, for any points such that $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| \neq \alpha'$, we have $\tilde{\mathbf{x}}_t = \mathbf{x}_t$. By Observation 1, these points are not manipulated, i.e., $\mathbf{x}_t = \mathbf{z}_t$. This implies $\tilde{\mathbf{x}}_t = \mathbf{z}_t$ which implies the claim for these points. Thus, we only need to argue for the data points such that $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| = \alpha'$. Consider such data points. For the positive data points, we have, $\tilde{\mathbf{x}}_t = \mathbf{x}_t = \mathbf{z}_t + \beta \cdot \mathbf{w}/|\mathbf{w}|$, where $0 \leq \beta \leq \alpha$. Therefore, $\tilde{\mathbf{x}}_t^T \mathbf{w}^* = \mathbf{z}_t^T \mathbf{w}^* + \beta \cdot \mathbf{w}^T \mathbf{w}^*/|\mathbf{w}| \geq \mathbf{z}_t^T \mathbf{w}^* \geq 1$. The first inequality holds because by the assumption of this step, $\mathbf{w}^T \mathbf{w}^* \geq 0$. On the other hand, for the negative data points we have $\tilde{\mathbf{x}}_t = \mathbf{x}_t - \alpha' \cdot \mathbf{w}/|\mathbf{w}|$, where $\mathbf{x}_t = \mathbf{z}_t + \beta \cdot \mathbf{w}/|\mathbf{w}|$ and

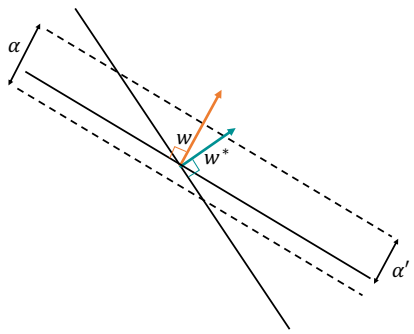


Fig. 4. Strategic Perceptron with unknown manipulation cost, when $\alpha \geq \alpha'$. The top dashed line represents the manipulation hyperplane. The margin between the two dashed lines represents the forbidden region.

$0 \leq \beta \leq \alpha$. This implies $\tilde{\mathbf{x}}_t = \mathbf{z}_t + (\beta - \alpha') \cdot \mathbf{w}/|\mathbf{w}|$. By multiplying with $\mathbf{w}^*$, we get $\tilde{\mathbf{x}}_t^T \mathbf{w}^* = \mathbf{z}_t^T \mathbf{w}^* + (\beta - \alpha') \cdot \mathbf{w}^T \mathbf{w}^*/|\mathbf{w}| \leq \mathbf{z}_t^T \mathbf{w}^* + (\alpha - \alpha') \cdot \mathbf{w}^T \mathbf{w}^*/|\mathbf{w}|$. Using $0 \leq \alpha - \alpha' \leq \gamma/2$ and $\gamma = 1/|\mathbf{w}^*|$, we have $\tilde{\mathbf{x}}_t^T \mathbf{w}^* \leq \mathbf{z}_t^T \mathbf{w}^* + \mathbf{w}^T \mathbf{w}^*/(2|\mathbf{w}^*||\mathbf{w}|) \leq \mathbf{z}_t^T \mathbf{w}^* + 1/2 \leq -1/2$. □

Lemma 5. Suppose Algorithm 2 is run with α' such that $0 \leq \alpha - \alpha' \leq \gamma/2$. When the algorithm makes a mistake on a positive example $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T \mathbf{w} \leq 0$; and when it makes a mistake on a negative example $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T \mathbf{w} \geq 0$.

Proof. First, we consider the positive points. The algorithm makes a mistake on a positive example only if $\mathbf{x}^T \mathbf{w}/|\mathbf{w}| < \alpha'$. Similar to Observation 2, in this case there is a margin without any observed data points. However, as illustrated in Figure 4, this margin is located differently; such that for no points, $\alpha' - \alpha < \mathbf{x}^T \mathbf{w}/|\mathbf{w}| < \alpha'$. Thus, for any positive example that the algorithm makes a mistake on, $\mathbf{x}^T \mathbf{w} \leq 0$. By Definition 1, $\tilde{\mathbf{x}}_t = \mathbf{x}_t$ for all positive examples. Therefore, $\mathbf{x}^T \mathbf{w} \leq 0$ implies $\tilde{\mathbf{x}}^T \mathbf{w} \leq 0$. Second, we consider negative points. For negative examples, the algorithm makes a mistake only if $\mathbf{x}^T \mathbf{w}/|\mathbf{w}| \geq \alpha'$. If the inequality is strict, i.e., $\mathbf{x}^T \mathbf{w}/|\mathbf{w}| > \alpha'$, by Definition 1, $\tilde{\mathbf{x}}_t = \mathbf{x}_t$, and therefore $\tilde{\mathbf{x}}_t^T \mathbf{w} \geq 0$. If $\mathbf{x}^T \mathbf{w}/|\mathbf{w}| = \alpha'$, again using Definition 1, we have $\tilde{\mathbf{x}}^T \mathbf{w} = 0$. □

Proposition 1. When $0 \leq \alpha - \alpha' \leq \gamma/2$, Algorithm 2 makes at most $4(R + \alpha' + \gamma/2)^2 |\mathbf{w}^*|^2$ mistakes.

Proof. Using Lemmas 4 and 5, the rest of the proof is similar to Theorem 1. □

Case 2: $\alpha < \alpha'$. Suppose α' is larger than α. By Observation 2, when Algorithm 2 is run with the real value of α, no data point is observed by algorithm in the margin $0 < \mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| < \alpha$. However, when the estimate is larger, since we overestimate by how far individuals can manipulate, Observation 2 no longer holds. Therefore, if the algorithm observes a point in the margin $0 < \mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| < \alpha'$, we realize that the estimate is large, and we need to refine it. On the other hand, while we have not observed any such points, the algorithm makes at most $(R + \alpha')^2 |\mathbf{w}^*|^2$ mistakes. This statement is summarized and proved below.

Proposition 2. Suppose Algorithm 2 is run with α', such that $\alpha' > \alpha$, and is halted if for a data-point $\mathbf{x}_t$, $0 < \mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| < \alpha'$. This modified algorithm makes at most $(R + \alpha')^2 |\mathbf{w}^*|^2 + 1$ mistakes.

Proof. Similar to the proof of Theorem 1, the maximum number of mistakes Algorithm 2 with estimated manipulation cost $1/\alpha'$ makes on observed data points $\mathbf{x}_t$ where $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| \leq 0$ or $\mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| \geq \alpha'$ is at most $(R + \alpha')^2 |\mathbf{w}^*|^2$. If a data point $\mathbf{x}_t$ is observed such that $0 < \mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| < \alpha'$, it implies $\alpha' > \alpha$ and the algorithm halts, and at most one more mistake is made on this data point. Therefore, the total number of mistakes is at most $(R + \alpha')^2 |\mathbf{w}^*|^2 + 1$. □

Case 3: $\alpha' < \alpha - \gamma/2$. We infer from Propositions 1 and 2 that if the number of mistakes is greater than $\max\{4(R + \alpha' + \gamma/2)^2 |\mathbf{w}^*|^2, (R + \alpha')^2 |\mathbf{w}^*|^2 + 1\} = 4(R + \alpha' + \gamma/2)^2 |\mathbf{w}^*|^2$ then $\alpha' < \alpha - \gamma/2$. Note that the equality holds since the number of mistakes is an integer.

Putting Everything Together. After discussing the three cases, we are now ready to explain Algorithm 4. This algorithm uses a binary search scheme to find a predictor in a bounded number of mistakes. The algorithm starts with $\alpha' = 0$. For each fixed α' we consider $4(R + \alpha' + \gamma/2)^2 |\mathbf{w}^*|^2$ as the maximum number of allowed mistakes. Whenever we exceed this bound using the discussion in Section 5.1 we learn that α' is too small. Also whenever we see a data point $\mathbf{x}_t$ such that $0 \leq \mathbf{x}_t^T \mathbf{w}/|\mathbf{w}| < \alpha$ as explained above we learn that α' is too large. Distinguishing between the cases where α' is too large or too small allows us to refine the upper bound and lower bound on α' until $0 \leq \alpha - \alpha' \leq \gamma/2$. The following theorem shows that the total number of mistakes is bounded during the whole process.

Algorithm 4: Strategic Perceptron with unknown manipulation cost

$\alpha'' \leftarrow 0, \alpha' \leftarrow 0$;
while *examples are arriving* **do**
  Run Algorithm 2 with estimate α' on the sequence of arriving examples, halt if $\#mistakes > 4(R + \alpha' + \gamma/2)^2|\mathbf{w}^*|^2$ or if for an example $\mathbf{x}_t$, $0 < \frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} < \alpha'$;
  if $\#mistakes > 4(R + \alpha' + \gamma/2)^2|\mathbf{w}^*|^2$ **then**
    /* guessed value α' is small. */
    $\alpha'' \leftarrow \alpha'$;
    $\alpha' \leftarrow \min\{\max\{2\alpha', \gamma/2\}, R\}$;
    continue;
  else if *for an example* $\mathbf{x}_t$, $0 < \frac{\mathbf{x}_t^T \mathbf{w}}{|\mathbf{w}|} < \alpha'$ **then**
    /* guessed value α' is large. */
    $\alpha' \leftarrow (\alpha'' + \alpha')/2$;
    continue;

Theorem 3. Algorithm 4 makes at most $O(R^2|\mathbf{w}^*|^2 \log(R|\mathbf{w}^*|))$ mistakes.

Proof. In Algorithm 4, the candidates for α are $\gamma/2$ apart and the number of them is $2R|\mathbf{w}^*|$. Since we are doing a binary search on these candidates, the total number of iterations of binary search is at most $\log(2R|\mathbf{w}^*|)$. Proposition 1, Proposition 2, and Theorem 1, show that in each iteration the total number of mistakes is bounded by $\max\{4(R + \alpha' + \gamma/2)^2|\mathbf{w}^*|^2, (R + \alpha')^2|\mathbf{w}^*|^2 + 1\} \leq 4(R + \alpha' + \gamma/2)^2|\mathbf{w}^*|^2 + 1$. Since we are assuming $\alpha' \leq R$, the total number of mistakes is at most $O(R^2|\mathbf{w}^*|^2 \cdot \log(R|\mathbf{w}^*|))$ and the proof is complete. □

Unknown γ. In the previous steps we assumed knowledge of γ. However, this assumption is not necessary and we can remove it in the following way. Starting from a guess of $|\mathbf{w}^*| = \frac{1}{2R}$ (i.e., a guess of $\gamma = 2R$), repeat the following procedure: for each guessed value of $|\mathbf{w}^*|$, Algorithm 4 is executed and if it makes more than the mistake bound of $(4(R + \alpha' + \gamma/2)^2|\mathbf{w}^*|^2 + 1)\log(2R|\mathbf{w}^*|)$, the guessed value for $|\mathbf{w}^*|$ is doubled (i.e., the guessed value of γ is halved) and the procedure is repeated. By putting this wrapper around Algorithm 4 with at most $O(R^2|\mathbf{w}^*|^2 \log(R|\mathbf{w}^*|))$ number of mistakes, the total number of mistakes remains in the same order of magnitude:

$$\sum_{i=-1}^{\log 2R|\mathbf{w}^*|} R^2 \left(\frac{|\mathbf{w}^*|}{2^i}\right)^2 \log\left(\frac{R|\mathbf{w}^*|}{2^i}\right) = O(R^2|\mathbf{w}^*|^2 \log(R|\mathbf{w}^*|))$$

5.2 Weighted ℓ_1 Costs

As observed in Section 4, in order for the strategic Perceptron algorithm to work in the weighted ℓ_1 costs model, it is necessary to identify in what direction the individuals manipulate. The tie-breaking step in Algorithm 3, ensured that the manipulation direction is unique and identifiable. In the unknown costs model, we need to make a guess for the cost in each direction. Since the guessed values are not accurate, we no longer can use them for a tie-breaking step and determine the manipulation direction. This restrains us from having an efficient algorithm for the general case of ℓ_1 costs. However, for a special case where manipulation is possible in a single direction (finite cost in direction $\mathbf{e}_1$ and infinite in the others), the manipulation direction is known and the ideas of Algorithm 4 extend to this case.

6 DIFFERENT COSTS

In the previous sections, we assumed all individuals have the same utility function. In this section, we show this assumption is not critical for our result and our algorithms still make a bounded number of mistakes and perform almost as well as long as the utility functions are close enough.

More particularly, suppose in the ℓ_2 costs setting, at each time t, the amount that an individual can move, α_t, is upper bounded by $\alpha_{\max}$ and lower bounded by $\alpha_{\min}$ such that $0 \leq \alpha_{\max} - \alpha_{\min} \leq \gamma/2$. Using the ideas presented in Section 5.1, we can show that running Algorithm 2 with $\alpha_{\min}$ as the input and the surrogate data points $\tilde{\mathbf{x}}_t$ defined with respect to $\alpha_{\min}$ makes a bounded number of mistakes.

Corollary 1. Suppose for all t, $\alpha_{\min} \leq \alpha_t \leq \alpha_{\min} + \gamma/2$. Algorithm 2 by using parameter $\alpha_{\min}$ makes at most $4(R + \alpha_{\min} + \gamma/2)|\mathbf{w}^*|^2$ number of mistakes.

Proof Outline. In Section 5.1, the guessed value of α that is used as the input to Algorithm 2, is at most $\gamma/2$ smaller than the real value. Similarly, in this case, $\alpha_{\min}$ is at most $\gamma/2$ smaller than any α_t. With a small difference in the terminology of the proofs of Lemmas 4 and 5, their statements hold and Proposition 1 directly implies this corollary. □

Weighted ℓ_1 Costs. Due to similar reasons explained in Section 5.2, the previous result does not extend to general case of weighted ℓ_1 costs but extends to the special case where manipulation is possible in a single direction.

7 TARGET CLASSIFIER NOT CROSSING THE ORIGIN

In this section, we propose an algorithm for the setting where unmanipulated data points are linearly separable, however not by a linear separator passing through the origin. Ordinarily (in the non-strategic setting), this would be handled by creating an extra fake coordinate, giving each example a value of 1 in that coordinate, and thereby reducing to the case where the separator crosses the origin, as explained in Extension 1. However, in the strategic setting, this reduction breaks down because the condition that $\mathbf{w}^T\mathbf{w}^* \geq 0$ (given in Lemma 1) is no longer sufficient to guarantee the quality of $\tilde{\mathbf{x}}_t$, since agents cannot manipulate in this new coordinate (they are no longer manipulating in the direction of $\mathbf{w}$). Instead, we present a different reduction here that is robust to strategic behavior.

For the case of ℓ_2 costs, we provide Algorithm 5 and show that the number of mistakes it makes is at most $O(R^3|\mathbf{w}^*|^3)$.

Overview of Algorithm 5. Assume the unmanipulated data points are separable by a linear separator $\mathbf{w}^{*T}\mathbf{z} + b = 0$. Suppose we can find an arbitrary point $\mathbf{p}^*$ such that $\mathbf{w}^{*T}\mathbf{p}^* + b = 0$. If we set the point $\mathbf{p}^*$ as the new origin, i.e., replacing each example $\mathbf{z}_t$ with $\mathbf{z}_t - \mathbf{p}^*$, then in the new coordinate system the unmanipulated data points are linearly separable by a separator that crosses the new origin, and we can use our previous algorithms. However, we are not able to necessarily find a point $\mathbf{p}^*$ such that $\mathbf{w}^{*T}\mathbf{p}^* + b = 0$. Instead, we show how to find a point $\mathbf{p}$ that is close enough to $\mathbf{p}^*$.

Initially, we find an unmanipulated positive example $\mathbf{x}^+$, and an unmanipulated negative example $\mathbf{x}^-$ by starting our algorithm in the following way: First, predict positive until the first mistake on a negative example $\mathbf{x}^-$ is made. Next, predict negative until the next mistake is made on some positive example $\mathbf{x}^+$. Consider the line segment between $\mathbf{x}^-$ and $\mathbf{x}^+$. There exists a point $\mathbf{p}^*$ on this line segment where $\mathbf{w}^{*T}\mathbf{p}^* + b = 0$. Consider a series of points on this line segment at distance $\gamma/2$ apart. For one of these points, which we call $\mathbf{p}$, $|\mathbf{p} - \mathbf{p}^*| \leq \gamma/2$. Lemma 6 shows if the origin is set to $\mathbf{p}$, i.e. each data point $\mathbf{z}_t$ is replaced with $\mathbf{z}_t - \mathbf{p}$, there exists a line passing through the

Algorithm 5: Strategic Perceptron with bias for ℓ_2 costs

$\mathbf{x}^+, \mathbf{x}^- \leftarrow \mathbf{0}, \mathbf{0}$;
while *examples are arriving* **do**
  predict +;
  if *the prediction was a mistake on a current example* $\mathbf{x_t}$ **then**
    $\mathbf{x}^- \leftarrow \mathbf{x}_t$;
    break;
while *examples are arriving* **do**
  predict −;
  if *the prediction was a mistake on a current example* $\mathbf{x_t}$ **then**
    $\mathbf{x}^+ \leftarrow \mathbf{x}_t$;
    break;
$\lambda \leftarrow 0$;
while *examples are arriving* **do**
  mistakes $\leftarrow 0$;
  /* choose a point $\mathbf{p}$ on the line segment between $\mathbf{x}^+$ and $\mathbf{x}^-$. */
  $\mathbf{p} \leftarrow (1-\lambda)\mathbf{x}^- + \lambda\mathbf{x}^+$;
  /* set the origin to point $\mathbf{p}$. */
  Run Algorithm 2 on the sequence of arriving examples in the new coordinate system, i.e. replace each example $\mathbf{x}_t$ with $\mathbf{x_t} - \mathbf{p}$, and halt if *mistakes* $> 4(2R+\alpha)^2|\mathbf{w}^*|^2$;
  /* $\mathbf{p}$ is not close enough to $\mathbf{p}^*$, i.e. $|\mathbf{p}-\mathbf{p}^*| > \gamma/2$. Try a different $\mathbf{p}$. */
  $\lambda \leftarrow \lambda + \gamma/2$;
  continue;

new origin $\mathbf{p}$ that separates original data points with a margin of $\gamma/2$, meaning that for each unmanipulated positive example $\mathbf{z}_t$, $(\mathbf{z}_t - \mathbf{p})^T\mathbf{w}^* \geq 1/2$, and for each unmanipulated negative example $\mathbf{z}_t$, $(\mathbf{z}_t - \mathbf{p})^T\mathbf{w}^* \leq -1/2$. When the origin is set to $\mathbf{p}$, Lemma 7 shows that if Algorithm 2 is executed on the arrived examples in the new coordinate system, i.e. replacing each observed example $\mathbf{x}_t$ with $\mathbf{x_t} - \mathbf{p}$, the number of mistakes is at most $4(2R+\alpha)^2|\mathbf{w}^*|^2$. Putting all together, we propose Algorithm 5 that is a generalization of Algorithm 2 for the case that original examples are separable by a linear classifier with non-zero bias. Theorem 4 shows Algorithm 5 makes at most $O(R^3|\mathbf{w}^*|^3)$ mistakes.

Lemma 6. Assume the points $\mathbf{z}_t$ are separable by a linear separator $\mathbf{w}^{*T}\mathbf{z} + b = 0$ of margin $\gamma = 1/|\mathbf{w}^*|$, and let $\mathbf{p}^*$ be a point such that $\mathbf{w}^{*T}\mathbf{p}^* + b = 0$. Then, if $|\mathbf{p}^* - \mathbf{p}| \leq \gamma/2$, the decision boundary $(\mathbf{z} - \mathbf{p})^T\mathbf{w}^*$ has a margin of separation $\gamma/2$.

Proof. First, $|\mathbf{w}^{*T}\mathbf{p} - \mathbf{w}^{*T}\mathbf{p}^*| \leq 1/2$ because $|\mathbf{w}^{*T}\mathbf{p} - \mathbf{w}^{*T}\mathbf{p}^*| \leq |\mathbf{w}^*||\mathbf{p}-\mathbf{p}^*| \leq |\mathbf{w}^*|/2|\mathbf{w}^*| = 1/2$. So, for a positive data point $\mathbf{z}_t$, if $(\mathbf{z}_t - \mathbf{p}^*)^T\mathbf{w}^* \geq 1$, then $(\mathbf{z}_t - \mathbf{p})^T\mathbf{w}^* \geq 1/2$. Similarly, for a negative data point $\mathbf{z}_t$, if $(\mathbf{z}_t - \mathbf{p}^*)^T\mathbf{w}^* \leq -1$ then $(\mathbf{z}_t - \mathbf{p})^T\mathbf{w}^* \leq -1/2$. □

Lemma 7. For a fixed guess $\mathbf{p}$ where $|\mathbf{p}^* - \mathbf{p}| \leq \gamma/2$, when the origin is set to $\mathbf{p}$ (i.e., each example $\mathbf{x}$ is replaced by $\mathbf{x} - \mathbf{p}$), then Algorithm 5 makes at most $4(2R+\alpha)^2|\mathbf{w}^*|^2$ mistakes.

Proof. Proof of this lemma is in the same lines as the proof of Theorem 1 with some modifications. First, Lemma 6 shows there exists a separator with margin of separation $\gamma/2$ passing through $\mathbf{p}$. By following the steps in Lemma 1, and using a margin of separation $\gamma/2$ instead of γ, we can show for any positive data point $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T\mathbf{w}^* \geq 1/2$, and for any negative data point $\mathbf{x}_t$, $\tilde{\mathbf{x}}_t^T\mathbf{w}^* \leq -1/2$. Next,

since each data point $\mathbf{x}_t$ is moved to $\mathbf{x}_t - \mathbf{p}$, $\max_t |\tilde{\mathbf{x}}_t| \leq 2R + \alpha$. Finally, by applying these bounds and following the steps in the proof of Theorem 1, the claim is proved. □

Theorem 4. Algorithm 5 makes at most $16R(2R + \alpha)^2|\mathbf{w}^*|^3$ mistakes in total.

Proof. Since the length of the line segment between $\mathbf{x}^+$ and $\mathbf{x}^-$ is at most $2R$, and guesses tried on this line segment are $\gamma/2$ apart, Algorithm 5 tries at most $4R/\gamma$ guesses for $\mathbf{p}$ in total. For each guess, if the number of mistakes is greater than $4(2R+\alpha)^2|\mathbf{w}^*|^2$, the next guess is tried. By Lemma 7, if for a guess $\mathbf{p}$, $|\mathbf{p} - \mathbf{p}^*| \leq \gamma/2$, on all the examples that will arrive the number of mistakes does not exceed $4(2R + \alpha)^2|\mathbf{w}^*|^2$. Therefore the claim is proved. □

Weighted ℓ_1 Costs. We show a reduction from this setting to the case where unmanipulated examples are separable by a hyperplane passing through the origin. First, similar to the ℓ_2 case, an extra fake coordinate with a value of 1 is added to each example. The bias term b is absorbed into $\mathbf{w}^*$, i.e. replacing $\mathbf{w}^*$ with $(\mathbf{w}^*, b)$. Now for the positive examples $\mathbf{x}_t^T\mathbf{w}^* \geq 1$, and for the negative examples $\mathbf{x}_t^T\mathbf{w}^* \leq -1$. Since agents cannot manipulate in the direction of the fake coordinate, the cost of manipulation along this direction is set to be infinity. Next, we bound the number of mistakes that Algorithm 3 makes in this setting. Since a fake coordinate is added to each example, R increases by a value of at most 1. Since the bias term is absorbed into the $\mathbf{w}^*$, the value of $|\mathbf{w}|$ increases by at most b. As a result, Algorithm 3 makes at most $(1 + (d + 1)(R + \alpha + 1)^2)(|\mathbf{w}^*| + b)^2$ number of mistakes. At the high level, the reason the direct reduction goes through for the weighted ℓ_1 case but not the ℓ_2 case is that in the ℓ_1 case, agents only manipulate in coordinate directions, and we have assumed that $\mathbf{w}^*$ is non-negative in each coordinate direction. So, the algorithm is not hurt if in computing $\tilde{\mathbf{x}}_t$ it overestimates the amount by which the agent has manipulated. This is the property that breaks down in the ℓ_2 case.

8 CONCLUSIONS AND OPEN PROBLEMS

In this work, we showed that if agents have the ability to manipulate their features within an ℓ_2 ball or a weighted ℓ_1 ball in order to be classified as positive, then the classic Perceptron algorithm may fail to achieve a bounded number of mistakes even when a perfect linear classifier exists. We then developed new Perceptron-style algorithms that achieve a finite mistake-bound, not much greater than the classic Perceptron bound in the non-strategic case, in both the ℓ_2 and weighted ℓ_1 manipulation setting. In the case that the manipulation costs are unknown to the learner—i.e., the radius of the ball in which agents can modify their features (or the per-coordinate radius in the weighted ℓ_1 case)—we provide an algorithm for the ℓ_2 costs setting and a specific case of the weighted ℓ_1 costs setting.

Our work suggests several open problems. First, designing an algorithm for the general case of weighted ℓ_1 costs when the costs of manipulation along each coordinate is unknown. This is challenging because given an observed data point, the learner doesn't know which direction it may have manipulated from, and this direction will change as the hypothesis classifier changes.

Second, for the case of inseparable data points, getting a bound in terms of the hinge-loss of the best separator with respect to the original data points $\mathbf{z}_1, \mathbf{z}_2, \cdots$. Our ideas in Section 3 can be extended to get a bound in terms of the hinge-loss of the best separator of surrogate data points $\tilde{\mathbf{x}}_1, \tilde{\mathbf{x}}_2, \cdots$. However, the more interesting question of getting a bound in terms of unmanipulated data points remains open. In the online version of our paper[2], we provide an example where Algorithm 2 makes an unbounded number of mistakes when data points are not perfectly separable, even though there exists a separator with bounded hinge-loss.

[2]https://arxiv.org/abs/2008.01710

Third, when the separator of the original data points $\mathbf{z}_1, \mathbf{z}_2, \cdots$ does not cross the origin, as studied in Section 7, Algorithm 5 makes at most $O(R^3|\mathbf{w}^*|^3)$ mistakes in the ℓ_2 costs setting. Our last open problem is whether it is possible to improve the number of mistakes to $O(R^2|\mathbf{w}^*|^2)$.

REFERENCES

[1] Tal Alon, Magdalen Dobson, Ariel Procaccia, Inbal Talgam-Cohen, and Jamie Tucker-Foltz. 2020. Multiagent Evaluation Mechanisms. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 02 (Apr. 2020), 1774–1781. https://doi.org/10.1609/aaai.v34i02.5543

[2] Yahav Bechavod, Katrina Ligett, Zhiwei Steven Wu, and Juba Ziani. 2020. Causal Feature Discovery through Strategic Modification. *arXiv preprint arXiv:2002.07024* (2020).

[3] Mark Braverman and Sumegha Garg. 2020. The Role of Randomness and Noise in Strategic Classification. In *1st Symposium on Foundations of Responsible Computing, FORC 2020 (LIPIcs)*, Vol. 156. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 9:1–9:20.

[4] Michael Brückner and Tobias Scheffer. 2011. Stackelberg Games for Adversarial Prediction Problems. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '11)*. Association for Computing Machinery, New York, NY, USA, 547–555. https://doi.org/10.1145/2020408.2020495

[5] Yiling Chen, Yang Liu, and Chara Podimata. 2020. Learning Strategy-Aware Linear Classifiers. In *Proceedings of the Thirty-fourth Conference on Neural Information Processing Systems (NeurIPS 2020)*.

[6] Yiling Chen, Chara Podimata, Ariel Procaccia, and Nisarg Shah. 2019. Strategyproof Linear Regression in High Dimensions. *ACM SIGecom Exchanges* 17 (05 2019), 54–60. https://doi.org/10.1145/3331033.3331038

[7] Ofer Dekel, Felix Fischer, and Ariel D. Procaccia. 2008. Incentive compatible regression learning. In *In The ACM-SIAM Symposium on Discrete Algorithms (SODA '08)*. 277–286.

[8] Jinshuo Dong, Aaron Roth, Zachary Schutzman, Bo Waggoner, and Zhiwei Steven Wu. 2018. Strategic Classification from Revealed Preferences. In *Proceedings of the 2018 ACM Conference on Economics and Computation (EC '18)*. Association for Computing Machinery, New York, NY, USA, 55–70. https://doi.org/10.1145/3219166.3219193

[9] Nika Haghtalab, Nicole Immorlica, Brendan Lucier, and Jack Z. Wang. 2020. Maximizing Welfare with Incentive-Aware Evaluation Mechanisms. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, Christian Bessiere (Ed.). International Joint Conferences on Artificial Intelligence Organization, 160–166. https://doi.org/10.24963/ijcai.2020/23 Main track.

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

[11] Lily Hu, Nicole Immorlica, and Jennifer Wortman Vaughan. 2019. The Disparate Effects of Strategic Manipulation. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT* '19)*. ACM, New York, NY, USA, 259–268. https://doi.org/10.1145/3287560.3287597

[12] Jon Kleinberg and Manish Raghavan. 2019. How Do Classifiers Induce Agents to Invest Effort Strategically? *(EC '19)*. Association for Computing Machinery, New York, NY, USA, 825–844. https://doi.org/10.1145/3328526.3329584

[13] Reshef Meir, Ariel D. Procaccia, and Jeffrey S. Rosenschein. 2012. Algorithms for strategyproof classification. *Artificial Intelligence* 186 (2012), 123–156. https://doi.org/10.1016/j.artint.2012.03.008

[14] John Miller, Smitha Milli, and Moritz Hardt. 2020. Strategic Classification is Causal Modeling in Disguise. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Hal Daumé III and Aarti Singh (Eds.), Vol. 119. PMLR, 6917–6926. http://proceedings.mlr.press/v119/miller20b.html

[15] Smitha Milli, John Miller, Anca D. Dragan, and Moritz Hardt. 2019. The Social Cost of Strategic Classification. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT* '19)*. Association for Computing Machinery, New York, NY, USA, 230–239. https://doi.org/10.1145/3287560.3287576

[16] Frank Rosenblatt. 1958. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review* 65, 6 (1958), 386.

[17] Yonadav Shavit, Benjamin Edelman, and Brian Axelrod. 2020. Causal Strategic Linear Regression. In *Proceedings of the 37th International Conference on Machine Learning (Proceedings of Machine Learning Research)*, Hal Daumé III and Aarti Singh (Eds.), Vol. 119. PMLR, 8676–8686. http://proceedings.mlr.press/v119/shavit20a.html