Causal Strategic Classification: A Tale of Two Shifts

Guy Horowitz [1] **Nir Rosenfeld** [1]

Abstract

When users can benefit from certain predictive outcomes, they may be prone to act to achieve those outcome, e.g., by strategically modifying their features. The goal in strategic classification is therefore to train predictive models that are robust to such behavior. However, the conventional framework assumes that changing features does not change actual outcomes, which depicts users as 'gaming' the system. Here we remove this assumption, and study learning in a causal strategic setting where true outcomes do change. Focusing on accuracy as our primary objective, we show how strategic behavior and causal effects underlie two complementing forms of distribution shift. We characterize these shifts, and propose a learning algorithm that balances between these two forces and over time, and permits end-to-end training. Experiments on synthetic and semi-synthetic data demonstrate the utility of our approach.

1. Introduction

The field of strategic classification (Brückner et al., 2012; Hardt et al., 2016) studies learning in a setting where users can strategically respond to a learned classifier by modifying their features, at some cost, to obtain favorable predictive outcomes. Such behavior can be expected when predictions are used to inform decisions about users, and from which users stand to gain (or lose); common examples include loans approval, university admissions, and job hiring. The framework of strategic classification succinctly captures a widespread form of tension that naturally arises between a classifier and the users it targets, and which applies broadly. This has made it the target of much recent interest (Dong et al., 2018; Miller et al., 2020; Tsirtsis & Gomez Rodriguez, 2020; Jagadeesan et al., 2021; Ghalme et al., 2021; Zrnic et al., 2021; Levanon & Rosenfeld, 2021; 2022; Estornell et al., 2021; Lechner & Urner, 2021; Ahmadi et al., 2022; Nair et al., 2022; Eilat et al., 2022; Barsotti et al., 2022).

As a learning problem, strategic classification is appealing in that its simple and clean formulation permits and feasible practical challenges. But simplicity comes at a price, and the framework's general applicability is hindered by its reliance on a set of strong assumptions. As part of a growing community effort to extend strategic classification beyond its original narrow form, our goal in this work is to take one step towards making strategic classification more flexible. In particular, here we target one of the key assumptions in strategic classification, which is the assumption that true outcomes y do *not* change when features x are modified. Under this assumption, strategic behavior amounts to gaming, and users are depicted as acting to 'fool' the classifier. But in reality, this assumption rarely holds, since actions taken by users to change predictions can also change true outcomes.

The observation that changes in x can causally affect y has been made by several authors (Miller et al., 2020; Shavit et al., 2020; Bechavod et al., 2021; Harris et al., 2022). But to date, works that have addressed this point focus primarily on the question of *improvement*, i.e., whether (and how) learning can incentivize users to change x in ways that improve outcomes y. While this is an important goal, here we argue that the current perspective conflates (i) the mere fact that y can change, with (ii) the desire for y to change favorably. But from a purely predictive point of view, *any* changes to y—whether for better or for worse—may deteriorate performance. Hence, and given that the implications of causal strategic behavior on learning are not yet well-understood, here we choose to focus entirely on the conventional goal of optimizing predictive accuracy, and study appropriate notions of robustness. We view this as an essential first step, intended to set the ground for more elaborate learning tasks such as incentivizing for improvement.

Towards this, and aiming to remain as true as possible to the original formulation, we seek to take the minimal necessary step beyond strategic classification for introducing meaningful causal relations. Our proposed setting, which generalizes vanilla strategic classification, can be described succinctly by a simple causal graph depicting the relations between different variable types: *causal*, *non-causal* (or 'correlative'), and *unobserved*. The graph's structure defines

[1]Technion – Israel Institute of Technology, Haifa, Israel. Correspondence to: Nir Rosenfeld <nirr@cs.technion.ac.il>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

the learning objective, which in turn determines the precise form in which learning must be strategically-robust. This formulation reveals where and how causality can impede learning, and hints at how these challenges can be addressed.

In essence, strategic behavior can be viewed as entailing a certain form of distribution shift—with the key property that *how* the distribution shifts depends on the choice of classifier, indirectly through how it shapes user responses (Drusvyatskiy & Xiao, 2022; Maheshwari et al., 2022). Our first contribution is characterizing the role causality plays in this process. When causality is absent, strategic updates $x \mapsto x'$ change $p(x)$, but also $p(y|x)$; this is since the induced $p'(y|x')$ must 'remember' the original y. Conversely, we show that in a fully causal setting, learning reduces to a particular instance of *decision-dependent covariate shift*, in which strategic behavior affects only the marginal $p(x)$; In other words, causality 'cancels out' the strategic effect on $p(y|x)$. Thus, the challenge in learning lies in correctly balancing between two distinct notions of robustness.

Based on these insights, our second contribution is a learning algorithm for strategic causal classification. We focus on the setting where users respond rationally and under a predetermined feature partition; this places emphasis on coping with the uncertainty in y introduced by the causal structure. Here the challenge is that learning must simultaneously account for (i) *strategic* changes in x, in response to the learned classifier f; and (ii) *causal* changes in y, which result from changes in x. The key to effective learning therefore lies in correctly decoupling strategic and causal effects; towards this, and relying on our theoretical analysis, our algorithm makes use of an estimated marginal density model $\hat{p}(x)$, which is novel in this space. As we show, our approach effectively separates informational uncertainty, which is irreducible, from statistical uncertainty—which our approach efficiently reduces by making use of additional strategically-modified (i.e., 'dirty') data.

Our approach becomes especially effective over *time*: here we make connections to the literature on *performative prediction* (Perdomo et al., 2020), and study causal strategic learning in a temporal setting and under retraining dynamics. In standard strategic classification, learning is known to converge after a single time-step (Hardt et al., 2016); but this notion breaks once causal effects are introduced. The fact that both $p(y|x)$ and $p(x)$ can now temporally change poses a challenge, but also an opportunity: using an appropriate form of regularization, we show how learning can be made to incentivize feature updates that reveal labeled information from under-represented areas of $p(x)$, which contribute to an improved estimation of $p(y|x)$.

Finally, we conduct a series of experiments that empirically validate our approach. First, using synthetic data, we design experiments aimed at showcasing the challenges, pitfalls, and opportunities that arise when learning in causal strategic environments. Then, we use real data (augmented with simulated responses) to compare our approach to several baselines. We report both quantitative and qualitative results, and perform sensitivity analysis regarding to our structural assumptions. Overall, our results shed light on the importance of accounting for causal effects in strategic setting, and the need to correctly balance between these forces. All code is made publicly available and can be found at: `https://github.com/guyhorowitz/CSC`.

1.1. Related work

Strategic classification. Since its introduction (Brückner et al., 2012; Hardt et al., 2016), the literature on strategic classification has been growing rapidly. Efficient learning algorithms have been proposed for the original batch setting (Levanon & Rosenfeld, 2021; 2022), as well as online formulations (Chen et al., 2020; Ahmadi et al., 2021). On the theoretical front, Zhang & Conitzer (2021); Sundaram et al. (2021) extend PAC theory via strategic VC analysis. Ongoing efforts aim to extend the original setting to handle utilities that are unknown (Dong et al., 2018), noisy (Jagadeesan et al., 2021), estimated (Ghalme et al., 2021; Bechavod et al., 2022; Barsotti et al., 2022), allow for arbitrary preferences (Levanon & Rosenfeld, 2022), or are linked by a graph (Eilat et al., 2022). Other works break or relax some core assumptions, such as the order of play (Nair et al., 2022) or the role of time (Zrnic et al., 2021). Our work joins these efforts, with the aim of allowing true outcomes to causally change when features are modified.

Causal strategic learning. Several works blend causality with strategic learning, but these focus almost exclusively on improvement. Kleinberg & Raghavan (2020) study the problem of incentivizing agents to improve, and Alon et al. (2020); Haghtalab et al. (2020) generalize their setting to multiple agents; however, neither of these works directly consider learning. Miller et al. (2020) show that learning to incentivize improvement inevitably requires solving a non-trivial causal inference problem; thus, coping with causality requires making some assumptions about the underlying causal structure. Some works make assumptions that permit causal inference, acting either through indirect experimentation in online learning (Bechavod et al., 2021) or by using the published classifiers as instruments in an offline setting (Harris et al., 2022); these works, however, are restricted to regression. Other works consider particular causal relations, such as Mendler-Dünner et al. (2022) who study predictions as interventions, or Chen et al. (2021) who decouple gaming and improving effects in both learning and evaluation. Closest to ours is Shavit et al. (2020), who provide learning algorithms for improvement, estimation, and (to some extent) accuracy; however, they focus on linear regression (in which strategic responses are invertible),

consider a realizable (linear) setting, and make assumptions that permit causal discovery. Our work studies (agnostic) classification and focuses predominantly on accuracy.

2. Problem setup

We start by briefly describing standard strategic classification, and continue with our proposal for injecting causality.

2.1. Standard strategic classification

In the original formulation of strategic classification (Hardt et al., 2016), users have feature representations $x \in \mathcal{X} = \mathbb{R}^d$ and binary labels $y \in \mathcal{Y} = \{\pm 1\}$. Let $p(x, y)$ be a joint distribution over (nonstrategic) features and labels. The primary goal in learning is to find a classifier $f : \mathcal{X} \to \mathcal{Y}$ from a class F that achieves high expected accuracy, given a train set $S = \{(x_i, y_i)\}_{i=1}^m$ with $(x_i, y_i) \overset{iid}{\sim} p(x, y)$. At test time, however, f is evaluated on strategically-modified data, where users update features via the *best-response mapping*:

$$x^f = \Delta_f(x) \triangleq \operatorname*{argmax}_{x' \in \mathcal{X}} f(x') - c(x, x') \tag{1}$$

where $c(x, x')$ is a cost function that determines the cost of changing x to x', and is assumed to be known to all. The goal of learning is to minimize the expected 0-1 loss, but under the strategically-induced distribution:

$$\min_{f \in F} \mathbb{E}_{p(x,y)} \left[\mathbb{1}\{f(x^f) \neq y\} \right] \tag{2}$$

We focus on the common choice of linear classifiers $\hat{y} = f(x) = \operatorname{sign}(w^\top x + b)$ and generalized quadratic costs $c_Q(x, x') = (x' - x)^\top Q(x' - x) = \|x' - x\|_Q^2$ for PSD Q.

2.2. Causal strategic classification

A key assumption in standard strategic classification is that changes in x (via Δ_f) do *not* affect y; this is encoded directly in Eq. (2). We will be interested in breaking this assumption by allowing changes in x to causally affect y. To account for causal effects, we require a concrete structure that determines how changes in x translate to changes in y. We seek to take the minimally-necessary step for generalizing the standard setting to include causal effects.

The causal structure. Our main structural assumption is that observable features x can be partitioned into *causal* features $x_c \in \mathcal{X}_c = \mathbb{R}^{d_c}$ that affect y, and *correlative* features $x_r \in \mathcal{X}_r = \mathbb{R}^{d_r}$ that do not. Together, we denote $x = (x_c, x_r)$. To enable both x_c and x_r to be distinctly important in prediction, we allow for additional *unobserved* causal features, $u \in \mathcal{U} = \mathbb{R}^{d_u}$, with which x_r correlates. Thus, x_r can be informative of y beyond what is conveyed by x_c, and therefore complementarily useful in learning. This mimics a setting in which some known causes are observed (x_c), but alone cannot fully explain y, and so are complemented by additional features (x_r) which relate to other possible causes of y, but are themselves non-causal. We assume $(x, u) \sim p(x, u)$ for some unknown distribution $p(x, u)$. For labels, we consider y as determined jointly by x_c and u via $y \sim h^*(x_c, u)$, for some stochastic ground-truth labeling function h^*. Figure 1 compactly describes our proposed causal structure using a simple causal graph.[1]

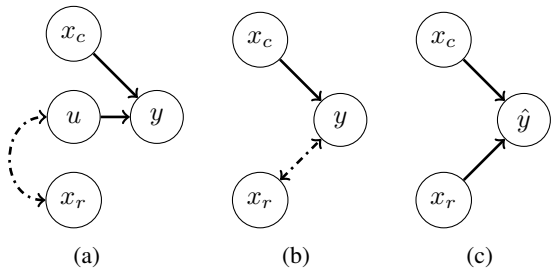

Figure 1: **(a)** The true causal graph over user features and labels. Solid lines represent direct causal effects, dashed lines depict correlation (whose source is abstracted away). **(b)** The causal graph, as perceived by the system. Note that u is unobserved, but its relation to y carries over to x_r, making it predictively informative of y. **(c)** The causal graph, as perceived by users, who seek positive predictions $\hat{y} = 1$.

Implications on learning. Since only (x_c, x_r) are observed, the classifier f can only be a function of these, i.e., $f(x_c, x_r)$, and users respond just as in Eq. (1), i.e., via:

$$x^f = (x_c^f, x_r^f) = \Delta_f(x_c, x_r) \tag{3}$$

Note this implies that users are incentivized to change only x_c and x_r, but *not* u: once Δ_f has been applied, the underlying features become (x_c^f, x_r^f, u), and the updated label—which is the true target of prediction—is $y^f = h^*(x_c^f, u)$. Given this, our causal strategic learning objective is:

$$\min_{f \in F} \mathbb{E}_{p(x,u)} \left[\mathbb{1}\{f(x^f) \neq h^*(x_c^f, u)\} \right] \tag{4}$$

In the simple case where $x_r = u$ (but noting $x_r \mapsto x_r^f$ does *not* change u), y^f can be interpreted as $h^*(\bar{x})$, where $\bar{x} = (x_c^f, x_r)$ is the 'projection' of x^f onto the causal subspace (see Figure 2). In the special case where there are no causal features (i.e., $x = x_r$), Eq. (4) reduces to the standard strategic classification objective in Eq. (2) with $y = h^*(u)$, and any discrepancies between x_r and u manifest as noise.

[1] Despite our structural assumptions, our setting remains quite flexible. First, relations between x_r and u can be arbitrary; e.g., x_r can be a causal child of u, or x_r and u have a common parent z. Second, h^* can be any stochastic function of x_c and u, and we make no assumptions on its form or relation to F. Third, we allow u and x to be dependent (this is abstracted away in Fig. 1). Fourth, we assume u includes *some* variables that correlate with x_r, but make no assumptions on, nor require knowledge of, their nature.

Challenges and prospects. The main challenge in optimizing Eq. (4) is that in addition to accounting for strategic responses $x \mapsto x^f$, learning must now also anticipate how such changes affect labels via $y^f = h^*(x_c^f, u)$. Intuitively, since points move to obtain positive predictions $\hat{y} = 1$, correctly estimating y^f is important for (i) *avoiding* negative post-strategic labels $y^f = -1$ (on which f errs), as well as for (ii) *encouraging* positive post-strategic labels $y^f = 1$ (on which f is correct). This reveals how user interests ($\hat{y} = 1$) align system goals ($\hat{y} = y^f$) with the general aim of improvement ($y^f = 1$). Nonetheless, these notions remain distinct, and optimizing for one criterion does *not* imply optimality for the other (see Appendix B.1).

Since h^* is unknown, our approach for optimizing Eq. (4) will be to replace h^* with some estimated $\hat{h}$. However, since training data S includes only 'clean' points (x, y), the challenge in this is twofold: (i) due to strategic behavior, h^* might need to be queried on points that lie outside the data distribution, and for which S (on which $\hat{h}$ is trained) may not be representative, and (ii) even though x_c^f can be computed, u remains to be unobserved. As we will show, allowing learning to make use of additional 'dirty' data (x^f, y^f), collected *over time* and under different deployed classifiers f, can enable learning to contend with these challenges.

2.3. Learning over time

To study temporal aspects of causal strategic learning, we adopt the general formulation of *performative prediction* (Perdomo et al., 2020). Here, learning proceeds in discrete rounds, where at each round $t > 0$ the currently deployed model f_t determines the data distribution in the next round, $p_{t+1} = p^{f_t} = D(f_t; p_0)$, for some initial p_0 and distribution mapping D. The overall goal is to optimize f on the distribution it induces, namely minimize the *performative risk*:

$$\min_{f \in F} \mathbb{E}_{(x,y)\sim p^f}[\mathbb{1}\{f(x) \neq y\}] \tag{5}$$

In our setting, D corresponds to feature updates via Δ_f and label updates via h^*, and Eq. (4) is a special case of Eq. (5).[2]

Similarly to Miller et al. (2021), we allow T rounds of retraining and deployment. At each round $t \leq T$, the system observes new data $S_t \sim p_t$, and (re)-trains f_t; then, it deploys f_t, which induces a distribution shift:

$$S_t \sim p_t \overset{train}{\longrightarrow} f_t \overset{induce}{\longrightarrow} p^{f_t} = p_{t+1}, \qquad p_0 = p$$

where in our setting $p_0 = p$ is the 'clean' distribution, and $S = S_0$. In retraining dynamics, each f_t is optimized for accuracy using currently available data: this relates to settings where deployed models are used throughout the dynamics, and so are required to perform well at each point in time. Note rounds $t > 0$ includes fresh samples, which consist purely of 'dirty' inputs $(x^{f_t}, y^{f_t}) \sim p_{t+1}$; importantly, for these points, their corresponding original 'clean' (x, y) remain unknown. Finally, at time T, the system commits to some final $f \in \{f_t\}_{t<T}$, to be used henceforth, and on which performative risk is evaluated.

[2]Note Eq. (2) is also a special case of Eq. (5), but which is known to converge after one round when Δ is known.

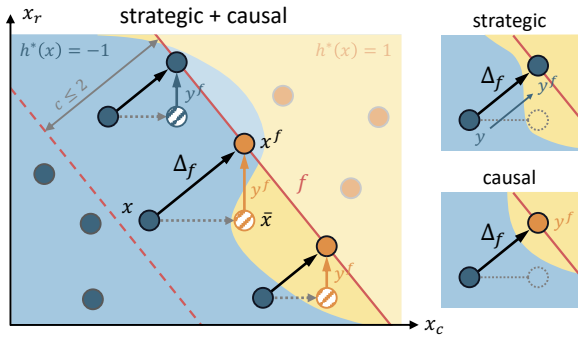

Figure 2: **An example of causal strategic classification.** Given f, strategic users move clean points $x = (x_c, x_r)$ onto the decision boundary, $x^f = \Delta_f(x)$, when costs permit. Labels, however, are affected only by the updated causal component, x_c^f, via $y^f = h^*(x_c^f, u)$ (c.f. strategic-only or causal-only cases). For $x_r = u$, labels y^f are given by projecting x^f onto the causal subspace, $\bar{x} = (x_c^f, x_r)$.

3. Analysis

To learn well in causal strategic settings, we must first understand how strategic behavior and causal effects translate into distribution shifts. In this section we characterize such shifts by analyzing the induced marginal $p^f(x^f)$ and conditional $p^f(y^f|x)$ for different cases. Throughout we use capital letters to denote random variables (e.g., X, U, Y) and lowercase for their realizations (e.g., x, u, y). Our analysis makes use of an 'inverse' response mapping operator:

$$\Delta_f^{-1}(x') \triangleq \{x \mid \Delta_f(x) = x'\} \tag{6}$$

which for any 'shifted' point x' returns the set of points x from which x' could have originated. For simplicity here we present results for deterministic h^*, but these also hold in the stochastic case. Proofs are deferred to Appendix A.

3.1. Case #1: Correlative-only features

Using x_r to predict y can be useful due to its correlation with u, which is a direct (and potentially distinct) cause of y. When f relies only on x_r, all of the features that are used for learning are non-causal, hence changes in x do not affect y.

Observation 1. *When using only x_r, causal strategic classification reduces to standard strategic classification.*

Denote $x_r' = \Delta_f(x_r)$. Our first result shows the connection

between the original and induced distributions.

Lemma 1. *Let $p(x_r, y)$ be some base distribution. Then for any classifier f, the induced $p^f(x'_r, y)$ can be expressed using the base marginal $p(x_r)$ and conditional $p(y|x_r)$ as:*

$$p^f(x'_r) = \int_{x_r \in \Delta_f^{-1}(x'_r)} p(x_r)\,\mathrm{d}x_r, \tag{7}$$

$$p^f(y|x'_r) = \int_{x_r \in \Delta_f^{-1}(x'_r)} \frac{p(x_r)}{p^f(x'_r)} p(y|x_r)\,\mathrm{d}x_r \tag{8}$$

Eq. (7) simply states that the probability of observing some modified x'_r derives from all points x_r that map to it via Δ_f. Eq. (8) then shows that predicting for x'_r its corresponding y (which remains unmodified) requires reasoning about the possible labels of all points in $\Delta_f^{-1}(x'_r)$; since this is a set, the implication is inherent (informational) uncertainty in y, which cannot be reduced through statistical means (i.e., observing more data from p^f). This reveals the mechanism through which strategic behavior can hinder accuracy, where $\frac{p(x_r)}{p^f(x'_r)}$ expresses how strategic behavior 'distorts' the base probability $p(y|x_r)$ (which already includes any uncertainty due to u, and to x_c if it exists).

Lemma 1 shows that the case of $x = x_r$ entails *full distribution shift*, i.e., both $p(x)$ and $p(y|x)$ can vary. Nonetheless, it provides useful insight, which is immediate from Eq. (8):

Corollary 1. *When using only x_r, knowing the base distribution suffices for constructing the Bayes-optimal classifier.*

Corollary 1 highlights why clean samples are useful; it also suggests that learning might benefit from incorporating a marginal density estimator, $\hat{p}(x)$, into the training procedure—a notion we adopt in our algorithm in Sec. 4.

3.2. Case #2: Causal-only features

Using x_c is useful for learning as it is a direct cause of y in itself. We now analyze the case of using only x_c for prediction, which requires us to directly account for u. In this case, the base conditional has the following form:

$$\begin{aligned} p(y|x_c) &= \int_u p(u|x_c)p(y|x_c, u)\,\mathrm{d}u \\ &= \int_u p(u) \cdot \mathbb{1}\{y = h^*(x_c, u)\}\,\mathrm{d}u \\ &= \int_{u:\, y = h^*(x_c, u)} p(u)\,\mathrm{d}u \end{aligned} \tag{9}$$

which is simply the uncertainty in y due to u. Further assuming that X_c and U are independent reveals a tight connection. Denote $x'_c = \Delta_f(x_c)$ and $y' = h^*(x'_c, u)$.

Lemma 2. *Let $p(x_c, y)$ be some base distribution, and assume $x_c \perp u$. Then for any classifier f, we have:*

$$p^f(y'|x'_c) = \int_{u:\, y' = h^*(x'_c, u)} p(u)\,\mathrm{d}u \tag{10}$$

and $p^f(x'_c)$ is as in Eq. (7) (with x_c replacing x_r).

Because the induced marginal is susceptible only to strategic effects, its form remains the same regardless of which features are used. More interestingly, Eq. (10) states that the induced conditional $p^f(y'|x'_c)$ remains exactly the same as the *original* base $p(y|x)$ (Eq. (9)). Thus, causal effects 'cancel out' the strategic effect of f on $p(y|x_c)$, and only $p^f(x_c)$ remains susceptible to strategic shifts. Thus, for *any* f, it holds that $P(Y' = y'|X'_c = x'_c) = P(Y = y'|X_c = x'_c)$, which is a special case of *covariate shift* (Shimodaira, 2000).

Corollary 2. *If x_c and u are independent, then irrespective of f, using only x_c reduces to learning under covariate shift.*

3.3. Case #3: Using all features

We now consider the most general case when all feature types are used (and with no assumptions on independence).

Lemma 3. *For any $p(x, u)$ and any classifier f, we have:*

$$p^f(y'|x') = \int_{u:\, y' = h^*(x'_c, u)} \nu_f(u; x')p(u)\,\mathrm{d}u \tag{11}$$

where:

$$\nu_f(u; x') = \int_{x \in \Delta_f^{-1}(x')} \frac{p(x|u)}{p^f(x')}\,\mathrm{d}x \tag{12}$$

and $p^f(x')$ is as in Eq. (7) (with x replacing x_r).

While covariate shift no longer holds, note that Eq. (11) matches Eq. (10) up to the term ν_f. Hence, ν_f quantifies the deviation from covariate shift due to f: when $\nu_f(u; x')$ takes values close to one across u, then covariate shift 'approximately' holds; otherwise, we have a particular form of full distribution shift. Note that in itself, ν_f relates to Eq. (8), in that the strategic effects also express as a distorted probability term integrated over the inverse response set.

Interpretation. Our analysis thus far reveals a tradeoff: Correlative features x_r are susceptible to gaming—which manifests as full distribution shift, but requires only clean data to accommodate. Conversely, causal features x_c (and their relation to u) bring learning closer to covariate shift, which is simpler, but introduces larger uncertainty in y. Note this uncertainty stems from points x^f moving to regions of low density under p; hence, in principle, dirty data gathered over time and in response to different models f may aid in decreasing uncertainty and improving performance. Our approach, presented next, aims to balance these two forces.

4. Method

Recall that our goal is to optimize the causal strategic learning objective in Eq. (4). Given a finite sample S, we adopt the conventional ERM approach and aim to minimize the empirical risk. Ideally, we would like to solve:

$$\operatorname*{argmin}_{f\in F} \sum\nolimits_{(x,y)\in S} \mathbb{1}\{f(x^f) \neq h^*(x_c^f, u)\} \tag{13}$$

However, this introduces several challenges: (i) the 0-1 loss is non-differentiable; (ii) x^f is the output of $\Delta_f(x)$, which is an argmax operator that is also non-differentiable; (iii) h^* is unknown, and (iv) u is unobserved, which together prevent us from computing updated labels $y^f = h^*(x_c^f, u)$. Also note that Eq. (13) makes no use of the observed clean y.

Our first step is to replace $\mathbb{1}\{\cdot\}$ with an appropriate proxy loss; for this, we adopt the *strategic hinge* $\ell_{\text{s-hinge}}$ from Levanon & Rosenfeld (2022), which accounts for strategic behavior and provides favorable generalization guarantees. Importantly, it does not explicitly rely on Δ_f, and is entirely differentiable. Next, for handling h^* and u, our general approach will be to replace h^* with a learned h, and use x_r as a surrogate for u.[3] We first describe our approach for clean data, and then extend it to utilize additional dirty data. Pseudocode for our entire procedure is given in Algorithm (1).

Algorithm 1 CSERM

Input: clean data S, regularization schedule λ_t
$S_0 \leftarrow S$
$\hat{p} \leftarrow \text{KDE}(S)$
for $t = 0, \dots T-1$ **do**
  update $S_{\leq t}$ to include S_t
  $\hat{q} \leftarrow \text{KDE}(S_{\leq t})$
  $h_t \leftarrow \operatorname{argmin}_{h\in\mathcal{H}} \sum_{(x,y)\in S_{\leq t}} \ell\left(h(x), y\right)$
  $f_t \leftarrow \operatorname{argmin}_{f\in F} \sum_{(x,y)\in S} \ell_{\text{s-hinge}}\left(f(x^f), \tilde{y}_{h_t}(x, x^f)\right) + \lambda_t R(f; S, \hat{q})$
  publish f_t and collect dirty samples $S_{t+1} \sim p^{f_t}$
  for $(x_c^{f_t}, x_r^{f_t}, y^{f_t}) \in S_{t+1}$ **do**
    $\tilde{x}_r \leftarrow \mathbb{E}_{x_r\sim\hat{p}(x_r|x^{f_t})}[x_r]$
    replace $x_r^{f_t}$ with $\tilde{x}_r$
  end for
end for

4.1. Learning with clean data

We propose to replace h^* in Eq. (13) with a differentiable estimate $h : \mathcal{X}_c \times \mathcal{X}_r \to [-1, 1]$, learned from data over some chosen function class H. Here the goal is to exploit the correlation between u and the observed (pre-strategic) x_r; i.e. h uses x_r as a "substitute" for how h^* uses u, and as a complement to x_c^f.[4] Since clean data includes clean labels $y = h^*(x_c, u)$, we can use S to optimize h via:

$$h = \operatorname*{argmin}_{h'\in H} \sum\nolimits_{(x,y)\in S} \ell\left(h'(x_c, x_r), y\right) \tag{14}$$

for some standard proxy loss ℓ (e.g., hinge loss or log loss). Note that h is learned on (x_c, x_r), but used on (x_c^f, x_r). In principle, h^* is needed only for points that move, since if $x^f \neq x$ then $y^f = h^*(x_c^f, u)$, and otherwise $y^f = y$. To prevent h from needlessly erring in such cases, we implement y^f as a differentiable 'soft if', $\tilde{y}_h$, as follows. First, note that x moves iff it is necessary *and* cost-effective, i.e.:

$$\mathbb{1}\{x \neq x^f\} = \mathbb{1}\{f(x) = -1\} \cdot \mathbb{1}\{c(x, x^f) < 2\}$$

Next, we relax this and define a soft movement indicator:

$$\mu(x, x^f) = 2\left(\sigma_\tau(c(x, x')) - 1/2\right) \cdot \sigma_\tau\left(2 - c(x, x^f)\right)$$

where σ is a sigmoid with temperature τ. Then, we express y^f using h and μ, which gives our soft updated label:

$$\tilde{y}_h(x, x^f) = y + \left(h(x_c^f, x_r) - y\right) \cdot \mu(x, x^f) \tag{15}$$

Finally, given h, our proposed objective for clean data is:

$$\operatorname*{argmin}_{f\in F} \sum\nolimits_{(x,y)\in S} \ell_{\text{s-hinge}}\left(f(x^f), \tilde{y}_h(x, x^f)\right) \tag{16}$$

4.2. Utilizing additional dirty data

The limitation in using only clean data for training h is that h is tailored to p, and may not approximate h^* well outside it—a likely scenario when points move strategically. Towards this, we propose to use temporally-gathered dirty data, sampled from induced distributions p^f, to iteratively improve h by retraining it at each round t on all available data, namely training h_t on $S_{\leq t} = \cup_{t'\leq t} S_{t'}$ where $S_{t'} \sim p^{t'} = p^{f_{t'-1}}$. Unfortunately, dirty data isn't immediately useful, and naïvely training h_t on it may introduce bias. To see why, note that dirty data includes inputs (x_c^f, x_r^f, y^f); in contrast, the observed y^f depends via h^* on x_c^f and on u. Whereas x_c^f is useful, what we require is the original x_r, which is informative of u—instead, we observe the modified x_r^f, which is inappropriate. Ideally, we would like to train h_t on 'mixed' pairs (x_c^f, x_r), but these are unavailable. As a solution, we propose to reconstruct $\tilde{x}_r \approx x_r$ using a density model $\hat{p}(x) \approx p(x)$, trained once at the onset on clean data. Since x^f is the strategic response to some x, we can estimate the likelihood for any given x_r as $\hat{p}(x_r|x^f)$; by considering all points in $\Delta_f^{-1}(x^f)$, we define:

$$\hat{p}(x_r|x^f) \triangleq \frac{1}{\hat{p}^f(x^f)} \int_{x_c:(x_c,x_r)\in\Delta_f^{-1}(x^f)} \hat{p}(x_c, x_r)\, \mathrm{d}x_c$$

To obtain a single entry, we compute the expected value:

$$\tilde{x} = (x_c^f, \tilde{x}_r), \text{ where } \tilde{x}_r = \mathbb{E}_{x_r\sim\hat{p}(x_r|x^f)}[x_r] \tag{17}$$

[3] Our approach requires to operationally define a feature partition as input to the learning algorithm. In Sec. 6.4 we empirically demonstrate its robustness to misspecified partitionings.

[4] Note h^* takes inputs (x_c, u), whereas h operates on (x_c, x_r).

and use these for training h. Appendix B.2 shows how to efficiently compute Eq. (17) , using the fact that $\Delta_f^{-1}(x^f)$ can be expressed as a closed interval of points in $\mathbb{R}^d$.

Regularization for exploration. Although dirty data can be helpful in extending the regions of data on which h_t is trained, *what* those regions are is determined entirely by the set of previous f_t. To promote variation in dirty data, we propose to augment Eq. (16) with a regularization term that encourages f to push points x^f to regions of low density:

$$R(f; S, \hat{q}) = \frac{1}{|S|} \sum_{x \in S} \log \hat{q}(x_c^f, x_r) \tag{18}$$

Here, $\hat{q}$ is a density model that is (re)-trained on aggregate data $S_{\leq t}$ at each round t, since its role is to inform us of uncertainty in h_t. Our final regularized learning objective is:

$$\underset{f \in F}{\operatorname{argmin}} \sum_{(x,y) \in S} \ell_{\text{s-hinge}}\left(f(x^f), \tilde{y}_h(x, x^f)\right) + \lambda R(f; S, \hat{q}) \tag{19}$$

This equips our approach with a mild form of exploration, whose degree is determined by λ. Practically we found it useful to use a gradually decaying λ_t: this places initial emphasis on exploration, which gradually shifts towards exploitation as more data is collected. In our experiments we use a kernel density estimator (KDE) for $\hat{q}$, which is differentiable; hence, the entire objective can be trained end-to-end.

5. Experiments using synthetic data

We begin with a series of synthetic experiments, each designed to demonstrate a different aspect of our setting and approach. We consider $x \in \mathbb{R}^2$ (which can be visualized), and fix $x_1 = x_c$ and $x_2 = x_r = u$. We compare learning using our strategic causal approach (`CSERM`) to (i) a naïve `ERM` approach, and (ii) a strategically-aware (but causally-oblivious) baseline that optimizes Eq. (2) using the approach in Levanon & Rosenfeld (2022) (`SERM`). We also consider a non-strategic benchmark (`ns-bench`) in which `ERM` is evaluated on clean (i.e., non-strategic) data.

Utilizing improvement. Our first experiment studies the ability of our approach to identify and make use of regions where h^* is *positive* to increase predictive performance (Fig. 3 A). We construct $p(x_c, u)$ to include two clusters that are separable by a linear h^*, but inject noise so that it is no longer separable by any $f(x)$ (while preserving the majority class in each cluster). As expected, `SERM` (80% accuracy) operates by taking the optimal `ERM` solution (54%) and making it more strict to prevent negative points from crossing; from its own perspective, this is sensible, since if y does not change, then negative points that move cause f to err. In contrast, `CSERM` (89%) utilizes its knowledge of h^* (via h) to push negative points to positive regions; once these points move, they obtain both positive predictions *and* positive labels, and accuracy increases—surpassing `ns-bench` (80%).

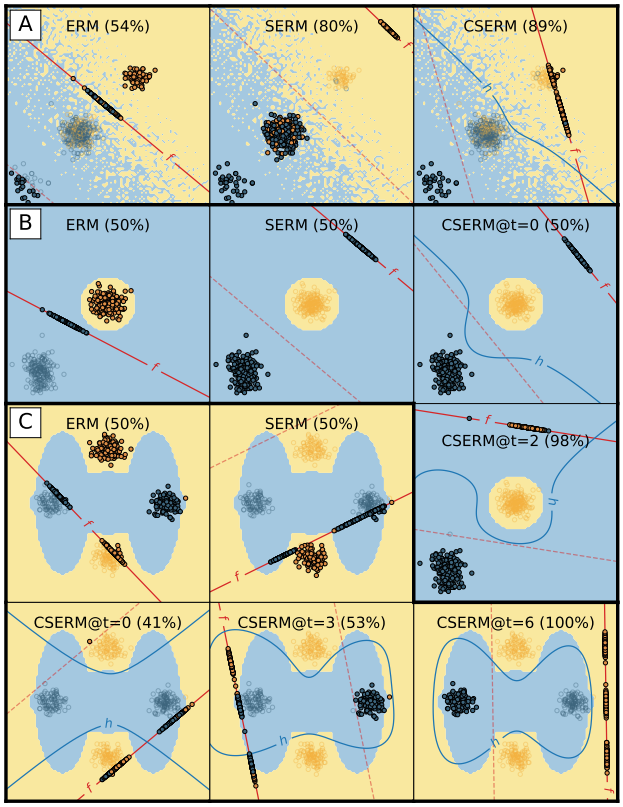

Figure 3: Results on synthetic experiments (A,B,C) for `ERM`, strategic `ERM` (`SERM`), and our approach (`CSERM`) (acc. in parentheses). Here $x_1{=}x_c$ (x-axis) and $x_2{=}x_r{=}u$ (y-axis). Hollow circles show pre-modified ('clean') data, filled circles show strategically-modified ('dirty') data. Colored regions depict h^*; red lines show learned f, dashed lines mark regions of movement for Δ_f, blue lines show learned h.

Avoiding pitfalls. We next experiment in a setting in which knowledge about where h^* is *negative* is crucial for preserving accuracy (Fig. 3 B). Here the clean $p(x_c, u)$ also defines two clusters, but which can now be separated by a learned f. However, outside $p(x_c, u)$, we define h^* to be positive precisely on the positive cluster, and negative elsewhere. Here the optimal solution is to use only x_r since it preserves the original y. `ERM` (50%) fails due to strategic behavior; `SERM` (50%) anticipates strategic responses, but is oblivious to h^*, and so inadvertently pushes positive points to become negative, and errs. `CSERM`, by estimating h^* with h, is able to find the optimal solution, though this takes time.

The role of exploration. Our last synthetic experiment considers the canonical XOR classification task, which is well-known to be non-linearly separable (Fig. 3 C). Indeed, `ERM` fails catastrophically (50%), as does `SERM` (50%). Nonetheless, our approach can obtain perfect accuracy—by utilizing causal knowledge to incentivize strategic behavior that *makes* the data separable. This, however, requires

Table 1: Results on real data.

	card fraud					spam				
	accuracy	%improve	%move	%neg↦pos	welfare	accuracy	perceived	%improve	%move	%pos↦neg
$\texttt{CSERM}_\lambda$	**87.8** $\pm$0.2	**12.2**	60.1	**13.8**	-0.65	**92.7** $\pm$0.5	97.0	3.1	37.5	**0.1**
CSERM	**86.6** $\pm$0.5	**10.2**	58.9	**11.8**	-0.48	**92.4** $\pm$0.4	97.1	2.4	36.7	**0.1**
SERM	78.4 $\pm$0.2	0.8	45.9	1.6	-0.16	84.0 $\pm$0.3	**91.2**	**-7.2**	41.3	7.2
RRM	75.8 $\pm$0.5	0.5	24.7	0.7	-0.06	77.2 $\pm$2.4	76.6	-2.1	30.5	2.8
ERM	66.7 $\pm$0.6	0.3	19.8	0.4	0.25	75.4 $\pm$0.3	91.2	0.4	17.1	0.0
oracle	87.0 $\pm$0.2	10.1	57.9	11.8	-0.60	93.5 $\pm$0.1	93.5	4.5	41.7	0.0

exploration: without regularization, CSERM is unable to improve, since newly collected dirty points do not improve h; however, by encouraging f to uncover uncertain regions, h improves over time, until it is sufficiently informative of h^* for learning to find the optimal f (100%), which pushes one cluster of negative points to a positive region.

6. Experiments using real data

We now turn to experiments based on real data using two public datasets: (i) spam, used originally in Hardt et al. (2016), and (ii) card fraud, used in Levanon & Rosenfeld (2021). Appendix D includes further details on data, methods, and optimization.

Procedure. Experimenting in a causal setting requires us to be able to query labels for arbitrary (modified) points (x'_c, u). Towards this, we begin each experiment by determining a partition of the original features into x_c and u, and use points (x_c, u) to train a ground-truth labeling function h^* using original labels y^*. For consistency we use h^* to generate labels y for both clean and dirty examples. We then define a mapping $u \mapsto x_r$, which can be lossy and noisy.

Next, we split the data roughly 60-10-30 into train, validation, and test sets. The train set is then further partitioned into a clean set, and an inventory from which dirty data is sampled (see Appendix E.2 for additional results on different ratios of clean vs. dirty data). Validation data is used for early stopping and model selection, and held-out test data is used for final evaluation. In line with our temporal setup in Sec. 2.3, we consider $T = 10$ rounds of retraining, where at each round t, we generate on the basis of f_t dirty samples S_t. These are obtained by taking a $1/T$-portion of the reserved inventory, and simulating strategic responses via $x^{f_t} = \Delta_{f_t}(x)$ and label updates $y^{f_t} = h^*(x_c^{f_t}, u)$. Once S_t is obtained, at round $t+1$ it can be used for training f_{t+1}. Non-temporal methods are given access to the full (clean) train set. Costs are $c_\alpha(x, x') = \alpha\|x - x'\|_2^2$, where for consistency across datasets we set α so that $\sim 50\%$ of points move on round $t = 1$. Appendix E.2 includes results on additional α-s and for all methods, exhibiting qualitatively similar performance trends.

Finally, we run all methods and compare performance. For methods that make use of dirty data over time, we report results for each round, as well as for the best model (chosen on validation data) to which the method commits. We report average results and standard errors over 15 random splits.

Methods. In addition to (i) ERM, (ii) SERM, and (iii) our CSERM, here we consider the following additional methods: (iv) repeated risk minimization (RRM) (Perdomo et al., 2020), which applies ERM independently at each round; (v) $\texttt{RRM}_{\leq t}$, which uses all previous data; and (vi) RRMc, which uses only causal features to avoid dealing with 'gaming' behavior. For our approach, we distinguish between non-regularized (CSERM) and regularized ($\texttt{CSERM}_\lambda$) variants (by default $\lambda_0 = 1$). We include the non-strategic ns-bench, and an oracle-like benchmark which combines h^* within our approach.

6.1. Utilizing improvement on card fraud

For fraud, we set h^* be 3-layer MLP, on top of which we add noise, so that some regions of x-space include a mixture of positive and negative labels. Potentially, if f can incentivize such points to move to areas where h^* is more positive, then this should entail better accuracy. Table 1 (left) shows results. As can be seen, CSERM improves significantly over other methods, gaining +8.2% by accounting for changes in y (vs. SERM) in addition to strategic effects (+19.9% vs. ERM). which here RRM is also able to achieve using time. Regularization adds +1.2%. To gain insight as to why, notice that CSERM incentivizes more movement (14% vs. SERM); of this, 14% of points shift from $y = -1$ to $y^f = 1$, giving an overall improvement rate of 12%. In comparison, other methods improve by $< 1\%$. Improvement, however, does not imply that users necessarily benefit: when considering welfare, defined as average utility minus costs (and so in $[-1, 1]$), the predictive success of CSERM comes at the price of reduced welfare, *despite* improvement.

6.2. Avoiding pitfalls on spam

For spam, we set h^* to be linear, except for one 'tricky' causal feature c_i which we concavify by preserving its positive slope *in-domain*, but reversing its slope to negative *out-of-domain*. This mimics a setting where having some

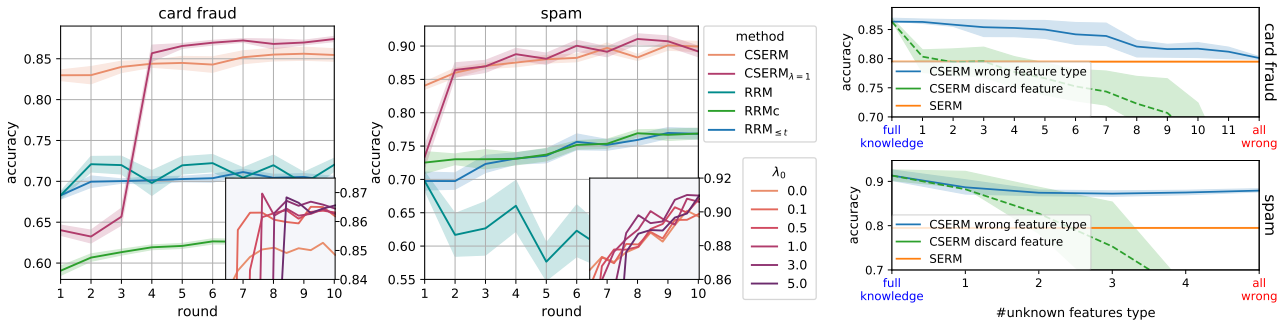

Figure 4: **(Left:)** Results for temporal methods over rounds. Inlay shows accuracy of $\texttt{CSERM}_\lambda$ for different values of λ. **(Right:)** Sensitivity analysis, showing accuracy for increasingly 'wrong' feature type attribution (causal vs. correlative).

amount of c_i is helpful for obtaining $y = 1$, but having 'too much' is not. Table 1 (right) shows results. Here as well, CSERM exhibits significant gains, but this time through different means. To see this, notice first that SERM's failure comes from causing positive points to become negative (7.2%); this occurs since its reliance on c_i—which is predictively-useful in-domain—breaks once points move out-of-domain in that direction (interestingly, SERM is blind to this, as its 'perceived' accuracy 7% higher than its actual accuracy). Conversely, and by correctly identifying its nature, CSERM dodges the c_i 'trap', and diverts movement elsewhere.

6.3. Time and regularization

Fig. 4 (left) shows the performance of temporal methods over time. Over time, and by utilizing additional dirty data, CSERM is able to improve performance (relative to $t = 1$) by $\sim$3% in `card fraud`, and $\sim$6% in `spam`. RRMc also improves over time—but to a significantly lesser degree; this shows the effectiveness of using causally-affected dirty data, but at the same time, reveals the (unutilized) potential of using non-causal features. RRM does use all features, but its performance over time is unstable (in `spam` performance *decreases* over time). $\texttt{RRM}_{\leq t}$ does improve, but is inconsistent across datasets. As for the effect of regularization, results show how $\texttt{CSERM}_\lambda$ initially performs worse than CSERM—but proceeds to outperform it for both datasets. This holds for all λ, and becomes more pronounced (for better and worse) as λ grows (see inlays).

6.4. Sensitivity analysis

Our final experiment tests the sensitivity of our approach to errors in features type attribution (i.e., considering a causal feature as non-causal, and vice versa). Towards this, for each $d' \leq d$, we evaluate a variant of CSERM which wrongly associates the type of a random subset of d' features ($\texttt{CSERM}_{\text{WRONG}}$). We compare this to SERM (which does not use feature type information at all), and to a variant of CSERM which simply discards the wrong features ($\texttt{CSERM}_{\text{DISCARD}}$). Results are shown in Fig. 4 (right), with average and standard deviation over 10 random seeds and $\min\{30, \binom{d}{d'}\}$ feature subsets per experimental condition. As expected, errors in feature type attribution do entail reduced performance for $\texttt{CSERM}_{\text{WRONG}}$; however, performance goes down slowly, either reaching SERM when all features are wrong ($d' = d$; for `card fraud`), or remaining above it (`spam`). In contrast, discarding the same 'wrong' features causes performance to deteriorate quickly and sharply.

7. Discussion

This paper extends the study of strategic classification to causal settings in which changing inputs can also change outputs. By focusing on the fundamental goal of optimizing accuracy, our analysis surfaces the need for learning to accommodate two interwoven forms of distribution shift. These differ in the challenges they present, but are also complementary in their relation to *time*; our approach utilizes these properties to provide a learning algorithm that is effective and efficient. Our choice of remaining true to the original problem formulation permits a clean formulation, and allows us to make connections to existing works. Nonetheless, the current literature on strategic classification remains far from being applicable in real social settings; we view our work as taking one step toward this ultimate goal.

Acknowledgements

This research was supported by the Israel Science Foundation (grant No. 278/22).

References

Ahmadi, S., Beyhaghi, H., Blum, A., and Naggita, K. The strategic perceptron. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pp. 6–25, 2021.

Ahmadi, S., Beyhaghi, H., Blum, A., and Naggita, K. On classification of strategic agents who can both game and improve. *arXiv preprint arXiv:2203.00124*, 2022.

Alon, T., Dobson, M., Procaccia, A., Talgam-Cohen, I., and Tucker-Foltz, J. Multiagent evaluation mechanisms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 1774–1781, 2020.

Barsotti, F., Koçer, R. G., and Santos, F. P. Transparency, detection and imitation in strategic classification. In *Proceedings of the 31st International Joint Conference on Artificial Intelligence, IJCAI 2022*, 2022.

Bechavod, Y., Ligett, K., Wu, S., and Ziani, J. Gaming helps! Learning from strategic interactions in natural dynamics. In *International Conference on Artificial Intelligence and Statistics*, pp. 1234–1242. PMLR, 2021.

Bechavod, Y., Podimata, C., Wu, S., and Ziani, J. Information discrepancy in strategic learning. In *International Conference on Machine Learning*, pp. 1691–1715. PMLR, 2022.

Brückner, M., Kanzow, C., and Scheffer, T. Static prediction games for adversarial learning problems. *The Journal of Machine Learning Research*, 13(1):2617–2654, 2012.

Chen, Y., Liu, Y., and Podimata, C. Learning strategy-aware linear classifiers. *Advances in Neural Information Processing Systems*, 33:15265–15276, 2020.

Chen, Y., Wang, J., and Liu, Y. Linear classifiers that encourage constructive adaptation. In *Algorithmic Recourse workshop at ICML'21*, 2021.

Costa, H., Merschmann, L. H., Barth, F., and Benevenuto, F. Pollution, bad-mouthing, and local marketing: the underground of location-based social networks. *Information Sciences*, 279:123–137, 2014.

Dong, J., Roth, A., Schutzman, Z., Waggoner, B., and Wu, Z. S. Strategic classification from revealed preferences. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pp. 55–70, 2018.

Drusvyatskiy, D. and Xiao, L. Stochastic optimization with decision-dependent distributions. *Mathematics of Operations Research*, 2022.

Eilat, I., Finkelshtein, B., Baskin, C., and Rosenfeld, N. Strategic classification with graph neural networks. *arXiv preprint arXiv:2205.15765*, 2022.

Estornell, A., Das, S., Liu, Y., and Vorobeychik, Y. Unfairness despite awareness: Group-fair classification with strategic agents. In *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS), StratML workshop*, 2021.

Ghalme, G., Nair, V., Eilat, I., Talgam-Cohen, I., and Rosenfeld, N. Strategic classification in the dark. In *International Conference on Machine Learning*, pp. 3672–3681. PMLR, 2021.

Haghtalab, N., Immorlica, N., Lucier, B., and Wang, J. Z. Maximizing welfare with incentive-aware evaluation mechanisms. In Bessiere, C. (ed.), *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, pp. 160–166, 7 2020. Main track.

Hardt, M., Megiddo, N., Papadimitriou, C., and Wootters, M. Strategic classification. In *Proceedings of the 2016 ACM conference on innovations in theoretical computer science*, pp. 111–122, 2016.

Harris, K., Ngo, D. D. T., Stapleton, L., Heidari, H., and Wu, S. Strategic instrumental variable regression: Recovering causal relationships from strategic responses. In *International Conference on Machine Learning*, pp. 8502–8522. PMLR, 2022.

Jagadeesan, M., Mendler-Dünner, C., and Hardt, M. Alternative microfoundations for strategic classification. In *International Conference on Machine Learning*, pp. 4687–4697. PMLR, 2021.

Kleinberg, J. and Raghavan, M. How do classifiers induce agents to invest effort strategically? *ACM Transactions on Economics and Computation (TEAC)*, 8(4):1–23, 2020.

Lechner, T. and Urner, R. Learning losses for strategic classification. In *Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS), Workshop on Learning in Presence of Strategic Behavior*, 2021.

Levanon, S. and Rosenfeld, N. Strategic classification made practical. In *International Conference on Machine Learning*, pp. 6243–6253. PMLR, 2021.

Levanon, S. and Rosenfeld, N. Generalized strategic classification and the case of aligned incentives. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

Maheshwari, C., Chiu, C.-Y., Mazumdar, E., Sastry, S., and Ratliff, L. Zeroth-order methods for convex-concave min-max problems: Applications to decision-dependent risk minimization. In *International Conference on Artificial Intelligence and Statistics*, pp. 6702–6734. PMLR, 2022.

Mendler-Dünner, C., Ding, F., and Wang, Y. Predicting from predictions. In *Advances in neural information processing systems*, 2022.

Miller, J., Milli, S., and Hardt, M. Strategic classification is causal modeling in disguise. In *International Conference on Machine Learning*, pp. 6917–6926. PMLR, 2020.

Miller, J. P., Perdomo, J. C., and Zrnic, T. Outside the echo chamber: Optimizing the performative risk. In *International Conference on Machine Learning*, pp. 7710–7720. PMLR, 2021.

Nair, V., Ghalme, G., Talgam-Cohen, I., and Rosenfeld, N. Strategic representation. In *International Conference on Machine Learning*, pp. 16331–16352. PMLR, 2022.

Perdomo, J., Zrnic, T., Mendler-Dünner, C., and Hardt, M. Performative prediction. In *International Conference on Machine Learning*, pp. 7599–7609. PMLR, 2020.

Shavit, Y., Edelman, B., and Axelrod, B. Causal strategic linear regression. In *International Conference on Machine Learning*, pp. 8676–8686. PMLR, 2020.

Shimodaira, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.

Sundaram, R., Vullikanti, A., Xu, H., and Yao, F. PAC-learning for strategic classification. In *International Conference on Machine Learning*, pp. 9978–9988. PMLR, 2021.

Tsirtsis, S. and Gomez Rodriguez, M. Decisions, counterfactual explanations and strategic behavior. *Advances in Neural Information Processing Systems*, 33:16749–16760, 2020.

Zhang, H. and Conitzer, V. Incentive-aware PAC learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pp. 5797–5804, 2021.

Zrnic, T., Mazumdar, E., Sastry, S., and Jordan, M. Who leads and who follows in strategic classification? *Advances in Neural Information Processing Systems*, 34, 2021.

A. Proofs

A.1. Lemma 1

Proof. Let $p(x_r, y)$ be some base distribution, and let f be a classifier $f : \mathcal{X}_r \to \mathcal{Y}$. Given f, denote $x'_r = \Delta_f(x_r)$, for which we can write the induced joint distribution as $p^f(x'_r y')$. To consider both x_r and x'_r together, we denote their joint distribution with y by $q^f(x_r, x'_r, y')$, whose definition derives immediately from Δ_f. Since Δ_f is deterministic, we get that $q^f(x'_r|x_r) = \mathbb{1}\{\Delta_f(x_r) = x'_r\} = \mathbb{1}\{x_r \in \Delta_f^{-1}(x'_r)\}$.

First, with the law of total probability, we get the following expression for the induced marginal density:

$$p^f(x'_r) = \int_{x_r \in \mathcal{X}_r} p(x_r) q^f(x'_r|x_r)\,\mathrm{d}x_r = \int_{x_r \in \mathcal{X}_r} p(x_r) \cdot \mathbb{1}\{x_r \in \Delta_f^{-1}(x'_r)\}\,\mathrm{d}x_r = \int_{x_r \in \Delta_f^{-1}(x'_r)} p(x_r)\,\mathrm{d}x_r \tag{20}$$

For the induced conditional density, again using the law of total probability we get:

$$p^f(y|x'_r) = \int_{x_r \in \mathcal{X}_r} q^f(x_r|x'_r) q^f(y|x'_r, x_r)\,\mathrm{d}x_r \tag{21}$$

Now, since y is sampled jointly with x_r from p, and since x'_r is a function only of x_r (which in itself is non-causal), we get that $q^f(y|x_r, x'_r) = p(y|x_r)$. Also, from Bayes' theorem, we have $q^f(x_r|x'_r) = q^f(x'_r|x_r)\frac{p(x_r)}{p^f(x'_r)}$. With these we get

$$\begin{aligned} p^f(y|x'_r) &= \int_{x_r \in \mathcal{X}_r} q^f(x'_r|x_r) \frac{p(x_r)}{p^f(x'_r)} p(y|x_r)\,\mathrm{d}x_r \\ &= \int_{x_r \in \mathcal{X}_r} \mathbb{1}\{x_r \in \Delta_f^{-1}(x'_r)\} \cdot \frac{p(x_r)}{p^f(x'_r)} p(y|x_r)\,\mathrm{d}x_r \\ &= \int_{x_r \in \Delta_f^{-1}(x'_r)} \frac{p(x_r)}{p^f(x'_r)} p(y|x_r)\,\mathrm{d}x_r \end{aligned} \tag{22}$$

□

A.2. Lemma 3

Proof. Let $p(x, u, y)$ be some base distribution, and let f be a classifier $f : \mathcal{X} \to \mathcal{Y}$. Given f, we denote $x' = \Delta_f(x)$, and define the joint distributions $p^f(x', u, y')$ and $q^f(x, u, x', y')$. From Eq. (20) and replacing x_r with x, we get:

$$p^f(x') = \int_{x \in \Delta_f^{-1}(x')} p(x)\,\mathrm{d}x \tag{23}$$

For the induced conditional density, with the law of total probability, we get:

$$p^f(y'|x') = \int_{u \in \mathcal{U}} p^f(u|x') p^f(y'|x', u)\,\mathrm{d}u \tag{24}$$

For a stochastic $h^* = p^*$, since x_c, u are both causal (and x_r is not), and since they jointly fully determine y (up to irreducible noise in h^*), we get that:

$$p^f(y'|x', u) = p^f(y'|x'_c, x'_r, u) = p^f(y'|x'_c, u) = p^*(y'|x'_c, u)$$

Plugging in we get:

$$p^f(y'|x') = \int_{u \in \mathcal{U}} p^f(u|x') p^*(y'|x'_c, u)\,\mathrm{d}u \tag{25}$$

Again using the law of total probability, generally we have:

$$p^f(u|x') = \int_{x \in \mathcal{X}} q^f(x|x') q^f(u|x', x)\,\mathrm{d}x \tag{26}$$

However, this can be simplified using the fact that x' originates from some x, i.e., $x' = \Delta_f(x)$. For the second term, since x' is a function of x alone, we get that $q^f(u|x', x) = p(u|x)$. For the first term, and similarly to the proof in A.1, using Bayes' theorem and the definition of Δ_f^{-1} we get:

$$q^f(x|x') = q^f(x'|x)\frac{p(x)}{p^f(x')} = \mathbb{1}\{x \in \Delta_f^{-1}(x')\} \cdot \frac{p(x)}{p^f(x')}$$

Plugging into the equation above gives:

$$\begin{aligned} p^f(u|x') &= \int_{x\in\mathcal{X}} \mathbb{1}\{x \in \Delta_f^{-1}(x')\} \cdot \frac{p(x)}{p^f(x')} p(u|x)\,\mathrm{d}x \\ &= \int_{x\in\Delta_f^{-1}(x')} \frac{p(x)}{p^f(x')} p(u|x)\,\mathrm{d}x \\ &= \int_{x\in\Delta_f^{-1}(x')} \frac{p(x|u)}{p^f(x')} p(u)\,\mathrm{d}x \end{aligned} \tag{27}$$

With the definition of ν_f as:

$$\nu_f(u; x') = \int_{x\in\Delta_f^{-1}(x')} \frac{p(x|u)}{p^f(x')}\,\mathrm{d}x \tag{28}$$

Taking out $p(u)$ we can write:

$$p^f(u|x') = \nu_f(u; x')p(u) \tag{29}$$

Plug it in back to Eq. (25), we get:

$$p^f(y'|x') = \int_{u\in\mathcal{U}} \nu_f(u; x')p(u)p^*(y'|x'_c, u)\,\mathrm{d}u \tag{30}$$

In the case of a deterministic h^*, i.e. $p^*(y'|x'_c, u) = \mathbb{1}\{h^*(x'_c, u) = y'\}$, this simplifies to:

$$\begin{aligned} p^f(y'|x') &= \int_{u\in\mathcal{U}} \nu_f(u; x')p(u) \cdot \mathbb{1}\{h^*(x'_c, u) = y'\}\,\mathrm{d}u \\ &= \int_{u:\, y'=h^*(x'_c,u)} \nu_f(u; x')p(u)\,\mathrm{d}u \end{aligned} \tag{31}$$

Additionally, in the case where x, u are independent, we get that $p(x|u) = p(x)$, therefore:

$$\begin{aligned} \nu_f(u; x') &= \int_{x\in\Delta_f^{-1}(x')} \frac{p(x|u)}{p^f(x')}\,\mathrm{d}x \\ &= \int_{x\in\Delta_f^{-1}(x')} \frac{p(x)}{p^f(x')}\,\mathrm{d}x = 1 \end{aligned} \tag{32}$$

which gives:

$$p^f(y'|x') = \int_{u:\, y'=h^*(x'_c,u)} p(u)\,\mathrm{d}u \tag{33}$$

□

A.3. Lemma 2

Proof. This is a special case of Lemma 3. Replacing x with x_c in Eq. (33) completes the proof. □

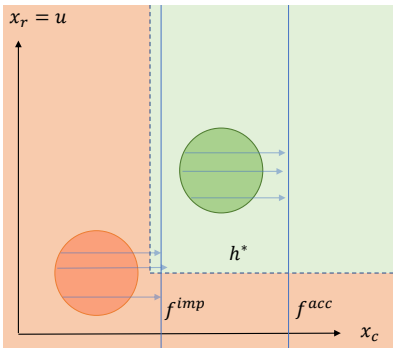

Figure 5: An optimal classifier for improvement is not necessarily optimal for accuracy.

B. Additional results

B.1. Accuracy and improvement can be at odds

Fig. 5 illustrates the idea that an optimal classifier in terms of maximizing improvement is not necessarily optimal for maximizing accuracy. In this example, the red and the green circles represent clusters of negative points ($y = -1$) and positive points ($y = 1$) respectively. The decision boundary of h^* is illustrated by a dashed line, and $x_r = u$. f^{imp} makes all the negative points move from the red circle to its decision boundary; half of the points (the upper half of the circle) become positive ($y' = 1$) since after movement their projection on their original $u = x_r$ lies in the positive region of h^*, and half of the points (the lower half of the circle) remains negative ($y' = -1$) since after movement their projection on their original x_r lies in the negative region of h^*. The points from the lower half of the red circle could never become positive: no matter how they move, their projection on their original x_r will always lie in the negative region of h^*. Therefore, f^{imp} turns all the possibly improvable points into positive and keeps all the originally positive points positive, hence it is optimal for maximizing improvement. However, since the points from the lower half of the red circle move to the decision boundary of f^{imp}, they are classified as positive ($\hat{y} = 1$), which means f^{imp} err ($\hat{y} \neq y'$) on each point from the lower half of the red circle. In contrast, f^{acc} make only the positive points from the green circle to move, and after moving their projection on their original x_r stays in the positive region of h^*, therefore they are classified correctly ($\hat{y} = y' = 1$); since the negative points from the red cluster don't move they are also classified correctly ($\hat{y} = y' = -1$), which means f^{acc} gets 100% accuracy, hence it is an optimal classifier for maximizing accuracy, with higher accuracy than f^{imp}.

B.2. Efficient computation of $\tilde{x}_r$

In this section, we show how to efficiently compute $\tilde{x}_r = \mathbb{E}_{x_r \sim \hat{p}(x_r|x^f)}[x_r]$ for a linear f and a generalized quadratic cost $c_Q(x, x') = (x' - x)^\top Q(x' - x) = \|x' - x\|_Q^2$ for PSD Q. Recall that the idea underlying our definition of $\tilde{x}_r$ is that we'd like to 'reconstruct' x_r, to the best of our ability, given a strategically modified example x^f. This is done by considering its likelihood, $p^f(x_r|x^f)$. We can express this likelihood using the clean marginal density:

$$
\begin{aligned}
p^f(x_r|x^f) &= \frac{p(x_r)}{p^f(x^f)} p^f(x^f|x_r) = \frac{p(x_r)}{p^f(x^f)} \int_{x_c} p(x_c|x_r) p^f(x^f|x_c, x_r)\, \mathrm{d}x_c \\
&= \frac{p(x_r)}{p^f(x^f)} \int_{x_c \in \mathcal{X}_c} p(x_c|x_r) \cdot \mathbb{1}\{(x_c, x_r) \in \Delta_f^{-1}(x^f)\}\, \mathrm{d}x_c \\
&= \frac{p(x_r)}{p^f(x^f)} \int_{x_c:\, (x_c, x_r) \in \Delta_f^{-1}(x^f)} p(x_c|x_r)\, \mathrm{d}x_c \\
&= \frac{1}{p^f(x^f)} \int_{x_c: (x_c, x_r) \in \Delta_f^{-1}(x^f)} p(x_c, x_r)\, \mathrm{d}x_c
\end{aligned}
\tag{34}
$$

where

$$
p^f(x^f) = \int_{x \in \Delta_f^{-1}(x^f)} p(x)\, \mathrm{d}x \tag{35}
$$

Using a model of the clean marginal density $\hat{p}(x) \approx p(x)$ we can estimate $p^f(x_r|x^f)$ for any point by replacing p with $\hat{p}$ in this expression. The expected value of this likelihood is

$$\begin{aligned}\tilde{x}_r &= \mathbb{E}_{x_r \sim \hat{p}(x_r|x^f)}[x_r] = \int_{x_r \in \mathcal{X}_r} \hat{p}(x_r|x^f) \cdot x_r \, \mathrm{d}x_r \\ &= \frac{1}{\hat{p}^f(x^f)} \int_{x_r \in \mathcal{X}_r} \int_{x_c:(x_c,x_r) \in \Delta_f^{-1}(x^f)} \hat{p}(x_c, x_r) \cdot x_r \, \mathrm{d}x_c \, \mathrm{d}x_r \\ &= \frac{1}{\hat{p}^f(x^f)} \int_{x \in \Delta_f^{-1}(x^f)} \hat{p}(x) \cdot x_r \, \mathrm{d}x\end{aligned} \tag{36}$$

Next, we describe the precise structure of Δ_f^{-1} (for a linear classifier $f(x) = \mathrm{sign}(w^\top x + b)$ and $\|x' - x\|_Q^2$ cost) and show how it permits tractable computation. In our setting, points move directly to the hyperplane, in a straight line, defined by w and b, i.e., over a line that is orthogonal to the hyperplane. This means that a point x moves to $x^f = x - z\bar{w}$ such that $w^\top x^f = -b$, where $\bar{w} = \frac{w}{|w|}$ and $z \in \mathbb{R}$ is the movement step. To see which points can afford to move, we can look at the "furthest" points from the hyperplane that can still afford the movement, i.e., points x such that after movement to x^f pay cost of $\|x^f - x\|_Q^2 = 2$. Since Q is PSD, there is an invertible matrix A such that $Q = A^\top A$, so we can rewrite the cost as $\|A(x^f - x)\|_2^2 = \|Ax^f - Ax\|_2^2$. Therefore, the points that pay cost of 2 are points such that $2 = \|Ax^f - Ax\|_2^2 = \|A(x - z\bar{w}) - Ax\|_2^2 = \|Ax - zA\bar{w} - Ax\|_2^2 = \| - zA\bar{w}\|_2^2 = z^2\|A\bar{w}\|_2^2$. With this, we can get the maximal movement step that points can afford to do: $z = \frac{\sqrt{2}}{\|A\bar{w}\|_2}$. Therefore, for a point x^f which lies on the hyperplane, i.e. $w^\top x^f = -b$, we get that $\Delta_f^{-1}(x^f) = \left\{x^f - z\bar{w} \middle| 0 \le z \le \frac{\sqrt{2}}{\|A\bar{w}\|_2}\right\}$. Plugging in this to Eq. (36), we get:

$$\begin{aligned}\tilde{x}_r &= \frac{\int_{x \in \Delta_f^{-1}(x^f)} \hat{p}(x) \cdot x_r \, \mathrm{d}x}{\int_{x' \in \Delta_f^{-1}(x^f)} \hat{p}(x') \, \mathrm{d}x'} \\ &= x_r^f - w_r \cdot \frac{\int_{z=0}^{\frac{\sqrt{2}}{\|A\bar{w}\|_2}} z \cdot \hat{p}(x^f - z\bar{w}) \, \mathrm{d}z}{\int_{z'=0}^{\frac{\sqrt{2}}{\|A\bar{w}\|_2}} \hat{p}(x^f - z'\bar{w}) \, \mathrm{d}z'}\end{aligned} \tag{37}$$

In practice, we compute these integrals numerically.

C. Experimental details - synthetic data

In all of our synthetic experiments, we used 500 samples for clean training data, 150 samples of dirty data collected at each round (out of total $T = 10$ rounds), 100 samples for validation set, and 400 samples for test set. We will now specify experimental details for each experiment.

C.1. Experiment A - utilizing improvement

1. Structure of h^*: in this experiment, h^* is a linear function wrapped with a stochastic mechanism that creates noisy labels near the decision boundary.

2. Structure $p(x_c, u)$: $p(x_c, u)$ is constructed from 3 normal distributed clusters: i) cluster of positive points with $\mu = (2, 2), \sigma^2 = 0.4$ that contains 15% of the total points, ii) cluster of negative points with $\mu = (-5.5, -5.5), \sigma^2 = 0.6$ that contains 10% of the total points, and iii) cluster of a mixture of positive and negative points with $\mu = (-2, -2), \sigma^2 = 0.6$ that contains 75% of the total points.

3. Cost scale = 0.035.

4. Class of h: polynomial model with a degree of 3.

5. Hyper-parameters: f learning-rate = 0.01, h learning-rate = 0.01, batch-size = 64, sigmoid temperature =4 , l_2 regularization coefficient for f = 0, l_2 regularization coefficient for h = 0.

C.2. Experiment B - avoiding pitfalls

1. Structure of h^*: in this experiment, h^* has a circle shape with a center in $(0, 0)$ where points inside the circle are labeled as positive and points outside it are labeled as negative.
2. Structure $p(x_c, u)$: $p(x_c, u)$ is constructed from 2 normal distributed clusters: i) cluster of positive points with $\mu = (0, 0), \sigma^2 = 0.3$ that contains 50% of the total points, ii) cluster of negative points with $\mu = (-5.5, -5.5), \sigma^2 = (0.3, 0.45)$ that contains 50% of the total points.
3. Cost scale = 0.07
4. Class of h: polynomial model with a degree of 3.
5. Hyper-parameters: f learning-rate = 0.1, h learning-rate = 0.1, batch-size = 64, sigmoid temperature = 20, l_2 regularization coefficient for f = 0.1, l_2 regularization coefficient for h = 0.

C.3. Experiment C - XOR

1. Structure of h^*: in this experiment, h^* is constructed from 3 ellipses: 2 vertical ellipses with centers $(2, 0)$ and $(-2, 0)$ and one horizontal ellipse with a canter $(0, 0)$. Together these ellipses create a shape where points inside it are labeled as negative, and points outside it are labeled as negative.
2. Structure $p(x_c, u)$: $p(x_c, u)$ is constructed from 4 normal distributed clusters, each contains 25% of the points and with $\sigma^2 = 0.3$: i) cluster of positive points with $\mu = (0, 2.5)$, ii) cluster of positive points with $\mu = (0, -2.5)$ iii) cluster of negative points with $\mu = (2.5, 0)$, and iiii) cluster of negative points with $\mu = (-2.5, 0)$.
3. Cost scale = 0.08
4. Class of h: polynomial model with a degree of 4.
5. Hyper-parameters: f learning-rate = 0.05, h learning-rate = 0.01, batch-size = 64, sigmoid temperature = 20, l_2 regularization coefficient for f = 1, l_2 regularization coefficient for h = 0.01, exploration regularization coefficient $\lambda = 5$ with decay of 0.4.

D. Experimental details - real data

D.1. Data and preprocessing

D.1.1. CARD FRAUD

Data description. The data is publicly available at https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud. This dataset contains transactions made by credit cards that occurred in two days during September 2013 by European cardholders. This data set is highly unbalanced and contained 492 frauds out of 284,807 transactions. The data contains 31 numerical features: 'Time' which contains the seconds elapsed between each transaction and the first transaction in the dataset, 'Amount' which is the transaction amount, and additional 29 features which are the result of a PCA transformation.

Preprocessing. As preprocessing, we removed the 'Time' feature and then performed Z-score normalization to the data, followed by a division by the square root of the data dimension.

Data augmentation. Since this dataset contains only 492 negative samples, we created synthetic negative samples for the experiment by fitting a KDE model to the negative samples and then sampling generated samples from the model.

Data split. We sampled 5500 balanced samples for the experiment and set 3000 of them ($\sim 54\%$) as training data, 500 ($\sim 9\%$) as validation data, and 2000 ($\sim 36\%$) as test data. The baseline that doesn't use time used all of the training data in a single round. For the methods that do use time, including ours, we split the training data into 1000 clean samples and 2000 assigned to be dirty samples, partitioned into 10 batches of 200 samples each. in the first round, they got access only to the clean samples, and then during 10 rounds, each round t, they got access to additional 200 dirty samples that were created by applying Δ_f on the t-batch of the dirty samples inventory.

Experiment repetition. We repeated the experiment 15 times, each time with a random data split. The reported results are the averages and standard error over these random splits.

D.1.2. SPAM

Data description. The data can be obtained by the authors of Costa et al. (2014). The data includes features describing users of a large social network, some of which are spammers. The data is balanced with a total of 7076 samples and contains 60 numerical features and binary labels (spammer or not).

Preprocessing. As preprocessing, we kept only 15 features: qTips_plc, rating_plc, qEmail_tip, qContacts_tip, qURL_tip, qPhone_tip, qNumeriChar_tip, sentistrength_tip, combined_tip, qWords_tip, followers_followees_gph, qUnigram_avg_tip', qTips_usr, indeg_gph, qCapitalChar_tip. After removing the other features we performed Z-score normalization on the data, followed by a division by the square root of the data dimension.

Data split. Same as in `card fraud`.

Experiment repetition. Same as in `card fraud`.

D.2. Feature partition and labeling function h^*

D.2.1. CARD FRAUD

Feature partition. After preprocessing we selected 6 features to be x_c, and 16 features to be u. We create x_r by takings the first 6 features from u and multiplying them by a random square matrix.

Labeling function. We created h^* by fitting an MLP with 3 hidden layers with hidden dimensions of 10 on a balanced subset of the original data (before augmenting it with a KDE). We then wrapped the MLP model with a stochastic mechanism that given an input x, assigns a probability p as a function of the distance of x from the decision boundary of the model and then multiply the scores MLP(x) by -1 with a probability of p. After assigning score s to a sample, its label is $\text{sign}(s)$. In this way, points from the region near the decision boundary of h^* have noisy labels and they are a mixture of negative and positive points.

D.2.2. SPAM

Feature partition. After preprocessing we selected 3 features to be x_c, and 12 features to be u. We create x_r by takings the first 2 features from u and multiplying them by a random square matrix.

Labeling function. We created h^* by first fitting a linear model g on the data. We then wrapped the g with a 'tricky-feature' mechanism defined for a feature c_i, a threshold γ and a slope β: given an input x, if $x_{c_i} > \gamma$, then replace the score $s = g(x)$ of the linear model with $s \leftarrow g(x) - \beta(x_{c_i} - \gamma)$. After assigning score s to a sample, its label is $\text{sign}(s)$. We used $\gamma = 0.05$ and $\beta = 20$; these values were chosen such that this mechanism will cause label flip to only $\sim 5\%$ of the original data.

D.3. Density estimation

For both usages of a density model in our algorithm, $\hat{p}$ and $\hat{q}$, we used KDE with a Gaussian kernel. The hyper-parameter of the model is the kernel bandwidth, which we choose using a grid search cross-validation.

D.4. Training, tuning, and optimization

For both `card fraud` and `spam` experiments, we used the following parameters, which we choose manually:

1. class of h: MLP with 3 layers with a width of 10
2. f, h learning-rate = 0.01
3. batch size = 64
4. epochs = 100
5. an early stopping mechanism when there are 7 consecutive epochs without accuracy improvement on the validation set
6. sigmoid temperature $\tau = 4$
7. exploration regularization coefficients: in $\texttt{CSERM}_{\lambda\,=\,0.1}$ we used $\lambda_0 = 0.1$ which decays in each round with factor of 0.4. in $\texttt{CSERM}_{\lambda\,=\,1}$ we used $\lambda_0 = 1$ which decays in each round with factor of 0.4.

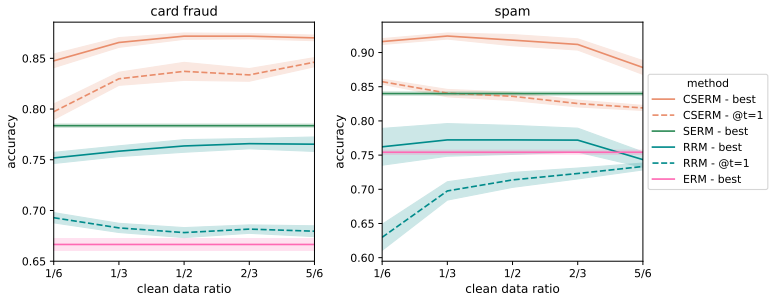

Figure 6: Accuracy across different ratios of clean vs dirty data.

In each experiment, we used a different cost scale α, chosen such that there will $\sim 50\%$ of strategically moving points: at `card fraud` we used $\alpha = 1$, and in `spam` we used $\alpha = 40$.

D.5. Baselines and benchmarks

In our experiments, we used two benchmarks:

1. `ns-bench`: the result of a naïve `ERM` tested on a non-strategic test; this benchmark shows us the maximal possible accuracy when there is no strategic behavior.
2. `oracle`: our method (`CSERM`) with oracle access to h^* and u, therefore in training it can accurately fix $y \mapsto y'$, for moving points; this benchmark shows us the maximal possible accuracy in a causal strategic setting, where there are no information gaps to the learner.

Additionally, we used the following baselines:

1. `ERM`: simulate a naïve learner who doesn't aware to strategic behaviour. The results of this baseline show us how much the learner can lose by not accounting for strategic behavior.
2. `SERM`: a strategically-aware but causally-oblivious baseline that optimizes Eq. (2) using the strategic hinge loss (Levanon & Rosenfeld, 2022). The results of this baseline show us how much the learner can lose by accounting only for the strategic movement of x and not for the possible change in the label y.
3. `RRM`: a baseline that uses time by collecting dirty data at each round, and at each round applies `ERM` using only the last collected dataset. This baseline simulates a learner that is aware of the distribution shift, but either doesn't know the structure of the shift or simply doesn't know how to tackle the problem of the specific distribution shift caused by strategic behavior and causality.
4. $\texttt{RRM}_{\leq t}$: a version of `RRM` that at each round uses the collected data from all previous rounds. This baseline simulates a learner that is aware of the fact that data from various distributions can be useful for learning under the distribution shift.
5. `RRMc`: a version of $\texttt{RRM}_{\leq t}$ that uses only causal features and uses all previous data. This baseline simulates a learner that is aware of the causal strategic structure of the distribution shift, knows the partition of features to x_c and x_r, and chooses to use only x_c to avoid dealing with 'gaming' behavior that the use of x_r causes.
6. `CSERM`: our approach, without regularizing for exploration.
7. $\texttt{CSERM}_{\lambda\,=\,0.1}$: our approach with exploration regularization coefficient of 0.1 in the first round, and decaying with a factor of 0.4 in each round.
8. $\texttt{CSERM}_{\lambda\,=\,1}$: our approach with exploration regularization coefficient of 1 in the first round, and decaying with a factor of 0.4 in each round.

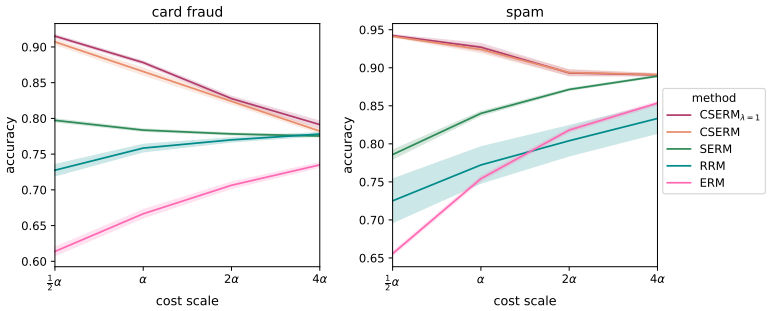

Figure 7: Accuracy across different cost scales.

E. Additional experimental results

E.1. Varying clean data ratio

This experiment tests the effect of the ratio of clean vs. dirty data on the performance of temporal methods that use dirty data over time in addition to clean data. Towards this, for each $r \in \{\frac{1}{6}, \frac{1}{3}, \frac{1}{2}, \frac{2}{3}, \frac{5}{6}\}$ we assigned an r-fraction of the training data to include clean example, and the remaining $1-r$-fraction to include dirty samples, while keeping the total size of training data fixed to 3,000 samples. Figure 6 plots performance as a function of r. As can be seen, the overall trend of the effect of r on accuracy changes across methods and datasets. However, results show that our approach remains effective across the entire spectrum of r, i.e. both when the number of clean samples is relatively small, and when it is relatively large.

E.2. Varying cost scales

In this section we report results for all methods and for multiple cost scales α. Our results in the main paper (Table 1 in Sec. 6) show performance for α chosen such that $\sim 50\%$ of points move (per dataset): in `card fraud` we set $\alpha = 1$, and in `spam` we set $\alpha = 40$. Here we show results for other cost scales, including $\frac{1}{2}\alpha$, α, 2α, and 4α. Figure 7 plots performance as a function of α. As can be seen, in each dataset the relations between the accuracies of the baseline remain similar across different cost scales, but as the cost scale decreases, there is less movement, and the absolute gap between the methods decreases as well. The next pages include tables reporting full results for all considered cost scales, first for `card fraud`, and then for `spam`.

card fraud, cost scale $\frac{1}{2}\alpha$

	accuracy	perceived	%improve	%move	%neg↦pos	%pos↦neg	welfare
CSERM$_{\lambda=1}$	91.5$_{\pm 0.2}$	95.2	15.3	59.9	16.9	1.5	-0.67
CSERM$_{\lambda=0.1}$	91.2$_{\pm 0.3}$	94.9	15.0	59.6	16.4	1.5	-0.7
CSERM	90.7$_{\pm 0.4}$	95.0	14.4	59.7	16.0	1.6	-0.57
SERM	79.7$_{\pm 0.2}$	77.5	2.2	55.5	3.4	1.2	-0.24
RRM	72.7$_{\pm 0.8}$	73.5	0.7	24.6	0.9	0.2	0.05
RRM$_{\leq t}$	69.6$_{\pm 0.2}$	77.7	0.3	16.1	0.3	0.0	0.2
RRMc	60.8$_{\pm 0.3}$	74.8	0.4	22.8	0.5	0.0	0.4
ERM	61.4$_{\pm 0.6}$	77.5	0.6	26.7	0.6	0.0	0.30
oracle	89.8$_{\pm 0.2}$	94.2	13.0	58.5	14.4	1.4	-0.72
ns-bench	77.5$_{\pm 0.2}$	-	-	-	-	-	-

card fraud, cost scale α

	accuracy	perceived	%improve	%move	%neg↦pos	%pos↦neg	welfare
CSERM$_{\lambda=1}$	87.8$_{\pm 0.2}$	93.5	12.2	60.1	13.8	1.7	-0.65
CSERM$_{\lambda=0.1}$	87.7$_{\pm 0.2}$	93.6	11.4	58.9	13.0	1.6	-0.5
CSERM	86.6$_{\pm 0.5}$	93.4	10.2	58.9	11.8	1.5	-0.48
SERM	78.4$_{\pm 0.2}$	77.5	0.8	45.9	1.6	0.7	-0.16
RRM	75.8$_{\pm 0.5}$	70.5	0.5	24.7	0.7	0.2	-0.06
RRM$_{\leq t}$	71.6$_{\pm 0.2}$	77.6	0.2	12.5	0.2	0.0	0.2
RRMc	63.6$_{\pm 0.3}$	74.7	0.3	18.8	0.3	0.0	0.4
ERM	66.7$_{\pm 0.6}$	77.5	0.3	19.8	0.4	0.0	0.25
oracle	87.0$_{\pm 0.2}$	93.3	10.1	57.9	11.8	1.6	-0.60
ns-bench	77.5$_{\pm 0.2}$	-	-	-	-	-	-

card fraud, cost scale 2α

	accuracy	perceived	%improve	%move	%neg↦pos	%pos↦neg	welfare
CSERM$_{\lambda=1}$	82.8$_{\pm 0.3}$	92.4	6.3	57.8	7.5	1.2	-0.58
CSERM$_{\lambda=0.1}$	82.3$_{\pm 0.5}$	92.3	6.5	58.8	7.7	1.2	-0.7
CSERM	82.4$_{\pm 0.4}$	92.8	5.8	57.9	7.0	1.2	-0.52
SERM	77.8$_{\pm 0.1}$	77.6	0.3	22.7	0.7	0.4	-0.05
RRM	77.0$_{\pm 0.3}$	69.9	0.1	22.9	0.3	0.3	-0.11
RRM$_{\leq t}$	73.6$_{\pm 0.1}$	77.5	0.1	8.7	0.1	0.0	0.2
RRMc	64.9$_{\pm 0.3}$	74.2	0.2	15.9	0.2	0.0	0.4
ERM	70.6$_{\pm 0.4}$	77.5	0.1	13.5	0.2	0.0	0.22
oracle	83.6$_{\pm 0.1}$	91.4	6.7	56.9	8.0	1.3	-0.53
ns-bench	77.5$_{\pm 0.2}$	-	-	-	-	-	-

card fraud, cost scale 4α

	accuracy	perceived	%improve	%move	%neg↦pos	%pos↦neg	welfare
CSERM$_{\lambda=1}$	79.1$_{\pm 0.5}$	85.9	2.5	35.4	3.1	0.6	-0.30
CSERM$_{\lambda=0.1}$	79.2$_{\pm 0.4}$	88.2	2.3	42.0	3.0	0.6	-0.4
CSERM	78.2$_{\pm 0.3}$	85.3	0.8	29.1	1.1	0.4	-0.14
SERM	77.5$_{\pm 0.1}$	77.5	0.0	7.3	0.2	0.2	0.06
RRM	77.8$_{\pm 0.2}$	70.9	0.4	18.8	0.5	0.1	-0.08
RRM$_{\leq t}$	74.8$_{\pm 0.2}$	77.7	0.0	6.6	0.1	0.0	0.2
RRMc	66.6$_{\pm 0.3}$	74.0	0.2	12.9	0.2	0.0	0.3
ERM	73.5$_{\pm 0.3}$	77.5	0.1	8.7	0.1	0.0	0.20
oracle	79.6$_{\pm 0.5}$	85.6	2.3	38.4	2.9	0.6	-0.37
ns-bench	77.5$_{\pm 0.2}$	-	-	-	-	-	-

spam, cost scale $\frac{1}{2}\alpha$

	accuracy	perceived	%improve	%move	%neg↦pos	%pos↦neg	welfare
CSERM$_{\lambda=1}$	94.2$_{\pm 0.1}$	97.6	6.4	57.0	6.4	0.0	-0.27
CSERM$_{\lambda=0.1}$	94.0$_{\pm 0.2}$	97.9	5.8	53.9	5.8	0.0	-0.3
CSERM	94.1$_{\pm 0.1}$	97.6	5.8	52.8	5.8	0.0	-0.25
SERM	78.6$_{\pm 0.6}$	91.2	-12.6	56.0	0.0	12.6	-0.36
RRM	72.5$_{\pm 2.9}$	76.9	0.4	36.1	1.3	0.9	0.13
RRM$_{\leq t}$	74.0$_{\pm 0.8}$	90.9	0.1	21.0	0.3	0.2	0.2
RRMc	74.9$_{\pm 0.6}$	87.1	-1.2	21.6	0.2	1.4	0.2
ERM	65.6$_{\pm 0.3}$	91.2	0.4	27.0	0.4	0.0	0.26
oracle	94.4$_{\pm 0.1}$	94.4	6.1	58.6	6.1	0.0	-0.31
ns-bench	91.2$_{\pm 0.1}$	-	-	-	-	-	-

spam, cost scale α

	accuracy	perceived	%improve	%move	%neg↦pos	%pos↦neg	welfare
CSERM$_{\lambda=1}$	92.7$_{\pm 0.5}$	97.0	3.1	37.5	3.2	0.1	-0.23
CSERM$_{\lambda=0.1}$	92.6$_{\pm 0.4}$	97.3	2.9	37.2	2.9	0.0	-0.1
CSERM	92.4$_{\pm 0.4}$	97.1	2.4	36.7	2.5	0.1	-0.21
SERM	84.0$_{\pm 0.3}$	91.2	-7.2	41.3	0.0	7.2	-0.17
RRM	77.2$_{\pm 2.4}$	76.6	-2.1	30.5	0.6	2.8	0.07
RRM$_{\leq t}$	78.7$_{\pm 0.6}$	91.2	0.1	14.8	0.3	0.1	0.2
RRMc	79.1$_{\pm 0.6}$	87.4	-0.8	15.8	0.2	1.1	0.2
ERM	75.4$_{\pm 0.3}$	91.2	0.4	17.1	0.4	0.0	0.23
oracle	93.5$_{\pm 0.1}$	93.5	4.5	41.7	4.5	0.0	-0.17
ns-bench	91.2$_{\pm 0.1}$	-	-	-	-	-	-

spam, cost scale 2α

	accuracy	perceived	%improve	%move	%neg↦pos	%pos↦neg	welfare
CSERM$_{\lambda=1}$	89.3$_{\pm 0.4}$	94.3	-1.3	21.6	0.2	1.5	-0.03
CSERM$_{\lambda=0.1}$	88.9$_{\pm 0.3}$	95.8	-1.8	22.4	0.1	1.9	0.0
CSERM	89.4$_{\pm 0.4}$	95.1	-1.4	22.0	0.2	1.5	-0.05
SERM	87.1$_{\pm 0.1}$	91.2	-4.1	25.7	0.0	4.1	-0.07
RRM	80.4$_{\pm 2.0}$	79.0	-2.3	22.1	0.3	2.6	0.00
RRM$_{\leq t}$	82.6$_{\pm 0.6}$	90.9	0.1	9.7	0.2	0.1	0.2
RRMc	81.7$_{\pm 0.4}$	87.5	-0.4	11.5	0.2	0.6	0.2
ERM	81.8$_{\pm 0.3}$	91.2	0.2	10.5	0.2	0.0	0.22
oracle	89.0$_{\pm 0.3}$	89.0	-1.9	23.2	0.0	1.9	-0.04
ns-bench	91.2$_{\pm 0.1}$	-	-	-	-	-	-

spam, cost scale 4α

	accuracy	perceived	%improve	%move	%neg↦pos	%pos↦neg	welfare
CSERM$_{\lambda=1}$	89.1$_{\pm 0.1}$	93.5	-1.6	11.8	0.0	1.6	0.06
CSERM$_{\lambda=0.1}$	89.0$_{\pm 0.2}$	93.2	-1.6	11.9	0.0	1.6	0.1
CSERM	89.1$_{\pm 0.2}$	93.4	-1.7	12.4	0.0	1.7	0.07
SERM	88.9$_{\pm 0.1}$	91.2	-2.3	15.4	0.0	2.3	0.01
RRM	83.3$_{\pm 1.9}$	81.2	-0.8	16.3	0.2	1.0	0.02
RRM$_{\leq t}$	84.8$_{\pm 0.4}$	91.0	0.0	7.2	0.1	0.1	0.2
RRMc	83.6$_{\pm 0.2}$	87.4	-0.2	8.3	0.1	0.3	0.2
ERM	85.4$_{\pm 0.2}$	91.2	0.0	6.8	0.0	0.0	0.20
oracle	89.1$_{\pm 0.1}$	89.1	-1.9	13.2	0.0	1.9	0.05
ns-bench	91.2$_{\pm 0.1}$	-	-	-	-	-	-