How to Learn when Data Reacts to Your Model: Performative Gradient Descent

Zachary Izzo [1] **Lexing Ying** [1 2] **James Zou** [3]

Abstract

Performative distribution shift captures the setting where the choice of which ML model is deployed changes the data distribution. For example, a bank which uses the number of open credit lines to determine a customer's risk of default on a loan may induce customers to open more credit lines in order to improve their chances of being approved. Because of the interactions between the model and data distribution, finding the optimal model parameters is challenging. Works in this area have focused on finding stable points, which can be far from optimal. Here we introduce *performative gradient descent* (PerfGD), an algorithm for computing performatively optimal points. Under regularity assumptions on the performative loss, PerfGD is the first algorithm which provably converges to an optimal point. PerfGD explicitly captures how changes in the model affects the data distribution and is simple to use. We support our findings with theory and experiments.

1. Introduction

A common paradigm in machine learning is to assume access to training and test datasets which are drawn independently from a fixed distribution. In practice, however, this is frequently not the case, and changes in the underlying data distribution can lead to suboptimal model performance. This problem is referred to as *distribution shift* or *dataset shift*.

While there is an extensive body of literature on distribution shift (Quionero-Candela et al., 2009), most prior works have focused on exogenous changes in the data distribution due to e.g. temporal or spatial changes. For instance, such changes may occur when a model trained on medical imaging data from one hospital is deployed at a different hospital due to the difference in imaging devices. Time series analysis is another plentiful source of these types of dataset shift. A model trained on stock market data from 50 years ago is unlikely to perform well in the modern market due to changing economic trends; similarly, a weather forecasting model trained on old data will likely have poor performance without accounting for macroscopic changes in climate patterns.

More recently, researchers have sought to address *endogenous* sources of distribution shift, i.e. where the change in distribution is induced by the choice of model. This setting, first explored in (Perdomo et al., 2020), is known as *performative* distribution shift. Such effects can arise for a variety of reasons. The modeled population may try to "game the system," causing individuals to modify some of their features to receive a more favorable classification (e.g. opening more credit lines to improve one's likelihood of being approved for a loan). Performative effects may also arise when viewing model output as a treatment. For instance, if a bank predicts a customer's default risk is high, the bank may assign that customer a higher interest rate, thereby increasing the customer's chance of defaulting (Drusvyatskiy & Xiao, 2020). As ML systems play an ever-increasing role in daily life, accounting for performative effects will naturally become more and more critical for both the development of effective models and understanding the societal impact of ML.

The original paper (Perdomo et al., 2020) and much of the follow-up research (Mendler-Dünner et al., 2020; Drusvyatskiy & Xiao, 2020; Brown et al., 2020) has viewed the performative setting as a dynamical system. The modeler repeatedly observes (samples from) the distribution arising from her choice of model parameters, then, treating this induced distribution as fixed, updates her model by reducing its loss on that fixed distribution. The primary question addressed by these works is under what conditions this process stabilizes, i.e. when will this process converge to a model which is optimal for the distribution it induces? A model with this property is known as a *performatively stable point*.

While performatively stable points may be interesting from a theoretical standpoint, focusing on this objective misses

[1]Department of Mathematics, Stanford University [2]Institute for Computational and Mathematical Engineering, Stanford University [3]Department of Biomedical Data Science. Correspondence to: Zachary Izzo <zizzo@stanford.edu>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

the primary objective of model training: namely, obtaining the minimum *performative loss*, i.e. the loss of the deployed model on the distribution it induces. The aforementioned previous works show that, in certain settings, a performatively stable point is a good proxy for a performatively optimal point, by bounding the distance between these two points in parameter space. In general, however, a performatively stable point may be far from optimal. In other less restrictive settings, a stable point may not even exist, and algorithms designed to find such a point may oscillate or diverge.

1.1. Our contributions

Motivated by these shortcomings, we introduce a new algorithm dubbed *performative gradient descent* (PerfGD) which provably converges to the performatively optimal point under realistic assumptions on the data generating process. We demonstrate, both theoretically and empirically, the advantages of PerfGD over existing algorithms designed for the performative setting.

1.2. Related work

Dataset shift is not a new topic in ML, but earlier works focused primarily on exogenous changes to the data generating distribution. For a comprehensive survey, see (Quionero-Candela et al., 2009).

Performativity in machine learning was first introduced by (Perdomo et al., 2020). The authors introduced two algorithms (repeated risk minimization and repeated gradient descent) as methods for finding a performatively stable point, and showed that under certain smoothness assumptions on the loss and the distribution map, a performatively stable point must lie in a small neighborhood of the performatively optimal point. Their results relied on access to a large-batch or population gradient oracle. In the follow up work (Mendler-Dünner et al., 2020), the authors showed similar results for the stochastic optimization setting. The authors in (Drusvyatskiy & Xiao, 2020) analyze a general class of stochastic optimization methods for finding a performatively stable point. They view these algorithms as performing biased stochastic optimization on the fixed distribution introduced by the performatively stable point, and show that the bias decreases to zero as training proceeds. In (Brown et al., 2020), the authors give results analogous to those in (Perdomo et al., 2020) when the distribution map also depends on the previous distribution. This models situations in which the population adapts to the model parameters slowly. In this case and under certain regularity conditions, RRM still converges to a stable point, and a stable point must lie within a small neighborhood of the optimum. We note that all of these works aim at finding a performatively stable, rather than performatively optimal, point.

Performativity in ML is closely related to the concept of strategic classification (Hardt et al., 2016; Cai et al., 2015; Shavit et al., 2020; Kleinberg & Raghavan, 2019; Khajehnejad et al., 2019). Strategic classification is a specific mechanism by which a population adapts to a choice of model parameters; namely, each member of the population alters their features by optimizing a utility function minus a cost. Performativity includes strategic classification as a special case, as we make no assumptions on the specific mechanism by which the distribution changes.

At the time of writing, the only other work which computed the performatively optimal point is (Munro, 2020). However, this work differs from ours in several important ways. First, in (Munro, 2020), the planner may deploy a different model on each individual from the sample at each time step. In our setting, as in (Perdomo et al., 2020), the model deployment must be uniform across all agents in each time step; testing different models constitutes different deployments, and we also seek the optimal uniform model. Second, (Munro, 2020) assumes that the performative shift results from strategic classification on the part of the agents. We trade these assumptions for parametric assumptions on the data generating process, but allow for a more general change in the data distribution (i.e. the change need not arise from a utility maximization problem.) In short, while superficially similar, our papers address unique settings and the results are in fact complementary.

Concurrently with our work, (Miller et al., 2021) studied necessary and sufficient conditions for a convex performative loss, as well as some specific families of performative shifts where these conditions hold. They then propose using black-box derivative-free optimization (DFO) methods to find the performative optimum.

Finally, training under performative distribution shift can be seen as a special instance of a zeroth-order optimization problem (Flaxman et al., 2005; Duchi et al., 2015; Lattimore, 2020), and our use of finite differences to approximate a gradient is a technique also employed by these works. However, the additional structure of our problem leads to algorithms better suited for the particular case of performative distribution shift. Generic zeroth-order optimization algorithms are sensitive to noise in the function value oracle, but PerfGD is capable of handling the noise resulting from evaluating the performative loss on a finite sample.

The rest of the paper is structured as follows. In Section 2, we introduce the problem framework as well as notation that we will use throughout the paper. We also discuss previous algorithms for performative ML and explore their shortcomings. In Section 3, we introduce our algorithm, performative gradient descent (PerfGD). In Section 4, we prove quantitative results on the accuracy and convergence of PerfGD. Section 5 considers several specific applications

of our method and verifies its performance empirically. We conclude in Section 6 and introduce possible directions for future work.

2. Setup and notation

We introduce notation which will be used throughout the rest of the paper.

- $\mathcal{Z} \subseteq \mathbb{R}^d$ denotes the sample space of our data.
- $\Theta \subseteq \mathbb{R}^p$ denotes the space of model parameters, which we assume is closed and convex. For notational convenience, we also assume that $p = d$, though this assumption is not necessary.
- $\mathcal{D} : \Theta \to \mathcal{P}(\mathcal{Z})$ denotes the performative distribution map. That is, when we deploy a model with parameters θ, we receive data drawn iid from $\mathcal{D}(\theta)$. We will assume that $\mathcal{D}$ is unknown; we only observe it indirectly from the data.
- $\ell(z; \theta)$ denotes the loss of the model with parameters θ on the point z. For regression problems, this will typically be the (regularized) square loss; for (binary) classification problems, this will typically be the (regularized) cross-entropy loss.
- $L(\theta_1, \theta_2)$ denotes the decoupled performative loss:

$$L(\theta_1, \theta_2) = \mathbb{E}_{\mathcal{D}(\theta_2)}[\ell(z; \theta_1)].$$

  Note that θ_1 denotes the *model's* parameters, while θ_2 denote's the *distribution's* parameters.
- $\mathcal{L}(\theta) = L(\theta, \theta)$ denotes the performative loss.
- It will be convenient to distinguish the two components of the *performative gradient* $\nabla_\theta \mathcal{L}(\theta)$. We denote $\nabla_1 \mathcal{L}(\theta) = \nabla_{\theta_1} L(\theta_1, \theta_2)|_{\theta_i = \theta}$ and $\nabla_2 \mathcal{L}(\theta) = \nabla_{\theta_2} L(\theta_1, \theta_2)|_{\theta_i = \theta}$, so $\nabla \mathcal{L} = \nabla_1 \mathcal{L} + \nabla_2 \mathcal{L}$.
- We denote $\theta_{\mathrm{OPT}} = \mathrm{argmin}_{\theta \in \Theta} \mathcal{L}(\theta)$.
- θ_{ALG} denotes the final output of the algorithm ALG. The three algorithms we will consider in this paper are repeated risk minimization (RRM), repeated gradient descent (RGD), and our algorithm, performative gradient descent (PerfGD).

Using the above notation, our interaction model is as follows. Start with some initial model parameters θ_0 and observe data $(z_i^t)_{i=1}^n \overset{\text{iid}}{\sim} \mathcal{D}(\theta_0)$. Then for $t = 0, 1 \ldots T-1$, compute θ_{t+1} using only information from the previous model parameters $\theta_s, s \leq t$ and datasets $(z_i^s)_{i=1}^n$, $s \leq t$. The goal of performative ML is to efficiently compute model parameters $\hat{\theta} \approx \theta_{\mathrm{OPT}}$. For our purposes, we will mainly consider the number of model deployments T as our measure of efficiency, and our goal is to keep this number of deployments low. This corresponds to a setting where deploying a new model is costly, but once the model has been deployed the marginal cost of obtaining more data and performing computations is low.

2.1. Previous algorithms

The authors of (Perdomo et al., 2020) formalized the performative prediction problem and introduced two algorithms—repeated risk minimization (RRM) and repeated gradient descent (RGD)—for computing a near-optimal point. We introduce these algorithms below.

Algorithm 1 Repeated gradient descent (RGD) (Perdomo et al., 2020)

while not converged **do**
  Draw $z_i^{(t)} \overset{\text{iid}}{\sim} \mathcal{D}(\theta_t)$, $i = 1, \ldots, n_t$.
  $\hat{\nabla}_1 \mathcal{L}(\theta_t) \leftarrow \frac{1}{n_t} \sum_{i=1}^{n_t} \nabla \ell(z_i^{(t)}; \theta_t)$
  $\theta_{t+1} \leftarrow \theta_t - \eta_t \hat{\nabla}_1 \mathcal{L}(\theta_t)$
  $t \leftarrow t + 1$
end while

Algorithm 2 Repeated risk minimization (RRM) (Perdomo et al., 2020)

while not converged **do**
  Draw $z_i^{(t)} \overset{\text{iid}}{\sim} \mathcal{D}(\theta_t)$, $i = 1, \ldots, n_t$.
  $\theta_{t+1} \leftarrow \mathrm{argmin}_\theta \sum_{i=1}^{n_t} \ell(z_i^{(t)}; \theta)$
  $t \leftarrow t + 1$
end while

The authors show that under certain assumptions on the loss and distribution shift, RRM and RGD converge to a *stable* point (i.e. model parameters θ_{STAB} such that $\theta_{\mathrm{STAB}} = \mathrm{argmin}_\theta L(\theta, \theta_{\mathrm{STAB}})$), and that $\theta_{\mathrm{STAB}} \approx \theta_{\mathrm{OPT}}$. When these assumptions fail, however, RRM and RGD may converge to a point very far from θ_{OPT}, or may even fail to converge at all.

2.2. Why aren't previous algorithms sufficient?

As an example, let $\mathcal{Z} = \mathbb{R}$, $\Theta = [-R, R]$ for some $0 < R < \infty$, and $\ell(z; \theta) = z\theta$. Define the distribution map $\mathcal{D}(\theta) = \mathcal{N}(a_1\theta + a_0, \sigma^2)$ for some fixed σ^2. The performative loss is then given by

$$\min_{\theta \in [-1,1]} \mathbb{E}_{\mathcal{N}(a_1\theta + a_0, \sigma^2)}[\theta z] = \min_{\theta \in [-1,1]} a_1\theta^2 + a_0\theta.$$

The optimal solution is at $\theta_{\mathrm{OPT}} = -a_0/2a_1$. Let us analyze the behavior of RRM. Letting $(z_i^t)_{i=1}^n$ denote the data sampled from $\mathcal{D}(\theta_t)$ and $\bar{z}_i^t = \frac{1}{n} \sum_{i=1}^n z_i^t$, RRM will set

$\theta_{t+1} = \mathbb{1}\{\bar{z}_i^t < 0\} - \mathbb{1}\{\bar{z}_i^t \geq 0\}$. If $a_1 > a_0 \geq 0$ and a sufficient number of samples are drawn at each deployment, and assuming σ^2 is small, with high probability when $\theta_t = R$, we will have $\bar{z}_i^t \approx a_1 + a_0 > 0 \Rightarrow \theta_{t+1} = -R$, and when $\theta_t = -R$, we will have $\bar{z}_i^t \approx -a_1 + a_0 < 0 \Rightarrow \theta_{t+1} = R$. That is, RRM will oscillate between $\theta = \pm R$ and fail to converge even to a stable point.

Next we analyze RGD. At each step, we update $\theta_{t+1} = \theta_t - \eta \mathbb{E}_{\mathcal{D}(\theta_t)} \nabla[\theta_t z]$. If this procedure converges, it will converge to a point θ_{STAB} such that $\mathbb{E}_{\mathcal{D}(\theta_{\mathrm{STAB}})} \nabla[\theta_{\mathrm{STAB}} z] = 0$. We can evaluate this expectation explicitly, and we see that $\theta_{\mathrm{STAB}} = -a_0/a_1 = 2\theta_{\mathrm{OPT}}$. Indeed, if R and $|\theta_{\mathrm{OPT}}|$ are large enough, this example shows that θ_{STAB} can be arbitrarily far from θ_{OPT}. Thus we see that in this simple case, RRM and RGD fail to find the optimal point, motivating our search for improved algorithms. In Section 5.2, we will return to a more general version of the problem introduced above and verify that our method, PerfGD, does indeed converge to θ_{OPT}.

3. General formulation of PerfGD

Our main goal is to devise a more accurate estimate for the true performative gradient $\nabla\mathcal{L} = \nabla_1\mathcal{L} + \nabla_2\mathcal{L}$. We already have a good stochastic estimate for $\nabla_1\mathcal{L}$ (this is just the gradient used by RGD), so we just need to estimate $\nabla_2\mathcal{L}$, i.e. the part of the gradient which actually accounts for the shift in the distribution.

In order to accomplish this, we make some parametric assumptions on $\mathcal{D}(\theta)$. Namely, we will assume that $\mathcal{D}(\theta)$ has a continuously differentiable density $p(z; f(\theta))$, where the functional form of $p(z; w)$ is known and the quantity $f(\theta)$ is easily estimatable from a sample drawn from $\mathcal{D}(\theta)$. For instance, if $\mathcal{D}(\theta)$ is in an exponential family, it has a density of the form $\frac{h(z)\exp[\eta(\theta)^\top T(z)]}{\int h(y)\exp[\eta(\theta)^\top T(y)]\,dy}$, which corresponds to the known function

$$p(z; w) = \frac{h(z)\exp[w^\top T(z)]}{\int h(y)\exp[w^\top T(y)]\,dy}$$

and unknown function $f(\theta) = \eta(\theta)$. For standard exponential families, there is a simple method of estimating the natural parameters $\eta(\theta)$ from a sample from $\mathcal{D}(\theta)$. Thus any continuous exponential family fits within this framework.

For concreteness, for the majority of the paper we will assume that $\mathcal{D}(\theta) = \sum_{i=1}^K \gamma_i \mathcal{N}(f_i(\theta), \Sigma_i)$, $\sum_{i=1}^k \gamma_i = 1$, $\gamma_i \geq 0$ is a mixture of normal distributions with varying means and fixed covariances. As any probability distribution with a smooth density can be approximated to arbitrary precision via a mixture of Gaussians, this parametric assumption on $\mathcal{D}(\theta)$ still gives rise to a powerful method.

3.1. Algorithm description

To describe the algorithm, it will be convenient to introduce some notation. For any collection of vectors $v_0, v_1, \ldots \in \mathbb{R}^d$ and any two indices $i < j$, we will denote by $v_{i:j}$ the matrix whose columns consist of $v_i, v_{i+1}, \ldots, v_j$, i.e.

$$v_{i:j} = \begin{bmatrix} | & | & & | \\ v_i & v_{i+1} & \cdots & v_j \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{d\times(j-i+1)}. \quad (1)$$

We also define $\mathbf{1}_H \in \mathbb{R}^H$ to be the vector consisting of H ones. Recalling that the space of model parameters Θ is assumed to be closed and convex, we define $\mathrm{proj}_\Theta(\theta)$ to be the Euclidean projection of θ onto Θ. Using this notation the pseudocode for PerfGD is given by Algorithm 3.

Algorithm 3 PerfGD

Input: Learning rate η; gradient estimation horizon H; parametric estimator function $\hat{f}$; gradient estimator function $\hat{\nabla}\mathcal{L}_2$

// Take first H updates via RGD
for $t = 0$ **to** $H - 1$ **do**
  // Draw a new sample and compute estimate for $f(\theta_t)$
  $(z_i)_{i=1}^n \overset{\text{iid}}{\sim} \mathcal{D}(\theta_t)$
  $f_t \leftarrow \hat{f}((z_i)_{i=1}^n)$

  // Compute naive gradient estimate and update parameters
  $\nabla_1\mathcal{L} \leftarrow \frac{1}{n}\sum_{i=1}^n \nabla\ell(z_i; \theta_t)$
  $\theta_{t+1} \leftarrow \mathrm{proj}_\Theta(\theta_t - \eta\nabla_1\mathcal{L})$
end for

// Run gradient descent with full gradient estimate
while not converged **do**
  // Draw a new sample and compute estimate for $f(\theta_t)$
  $(z_i)_{i=1}^n \overset{\text{iid}}{\sim} \mathcal{D}(\theta_t)$
  $f_t \leftarrow \hat{f}((z_i)_{i=1}^n)$

  // Estimate the first part of the performative gradient
  $\nabla_1\mathcal{L} \leftarrow \frac{1}{n}\sum_{i=1}^n \nabla\ell(z_i; \theta_t)$

  // Estimate the second part of the performative gradient

  $\Delta\theta \leftarrow \theta_{t-H:t-1} - \theta_t \mathbf{1}_H^\top$
  $\Delta f \leftarrow f_{t-H:t-1} - f_t \mathbf{1}_H^\top$
  $\frac{\Delta f}{\Delta\theta} \leftarrow (\Delta f)(\Delta\theta)^\dagger$
  $\nabla_2\mathcal{L} \leftarrow \hat{\nabla}\mathcal{L}_2(f_t, \frac{\Delta f}{\Delta\theta})$

  // Update the model parameters
  $\theta_{t+1} \leftarrow \mathrm{proj}_\Theta(\theta_t - \eta(\nabla_1\mathcal{L} + \nabla_2\mathcal{L}))$
  $t \leftarrow t + 1$
end while

3.2. Derivation

Assume that $\mathcal{D}(\theta)$ has density $p(z; f(\theta))$ with $p(z; w)$ known for arbitrary w. The performative loss is given by $\mathcal{L}(\theta) = \int \ell(z;\theta)p(z; f(\theta))\, dz$. Assuming that p and f are continuously differentiable, we can compute the performative gradient:

$$\nabla\mathcal{L}(\theta) = \overbrace{\int \nabla\ell(z;\theta)p(z; f(\theta))\, dz}^{\nabla_1\mathcal{L}} + \underbrace{\int \ell(z;\theta)\frac{df}{d\theta}^\top \partial_2 p(z; f(\theta))\, dz}_{\nabla_2\mathcal{L}}. \tag{2}$$

Note that $\nabla_1\mathcal{L} = \mathbb{E}_{\mathcal{D}(\theta)}[\nabla\ell(z;\theta)]$ and we can obtain an estimate for this quantity by simply averaging $\nabla\ell$ over our sample from $\mathcal{D}(\theta)$. For $\nabla_2\mathcal{L}$, the only unknown quantities are $f(\theta)$ and $df/d\theta$. By assumption, $f(\theta)$ should be easily estimatable from our sample, i.e. there exists an estimator function $\hat{f}$ which, given a sample $(z_i)_{i=1}^n \overset{\text{iid}}{\sim} \mathcal{D}(\theta)$ returns $\hat{f}((z_i)_{i=1}^n) \approx f(\theta)$.

To estimate $df/d\theta$, we use a finite difference approximation. By Taylor's theorem, we have $\Delta f \approx \frac{df}{d\theta}\Delta\theta$. By taking a pseudoinverse of $\Delta\theta$, we obtain an estimate for the derivative: $\frac{df}{d\theta} \approx \Delta f(\Delta\theta)^\dagger$. We require that this this system is overdetermined, i.e. $H \geq p$, to avoid overfitting to noise in the estimates of f and bias from the finite difference approximation to the derivative. (Recall that H is the number of previous finite differences used to estimate $df/d\theta$, and p is the dimension of θ.)

Substituting these approximations for $f(\theta)$ and $df/d\theta$ into the expression for $\nabla_2\mathcal{L}$, we can then evaluate or approximate the integral using our method of choice. One universally applicable option is to use a REINFORCE-style approximation (Williams, 1992):

$$\begin{aligned}\nabla_2\mathcal{L} &= \int \ell(z;\theta)\frac{df}{d\theta}^\top \partial_2[\log p(z; f(\theta))]p(z; f(\theta))\, dz \\ &= \mathbb{E}_{\mathcal{D}(\theta)}\left[\ell(z;\theta)\frac{df}{d\theta}^\top \partial_2[\log p(z; f(\theta))]\right]. \end{aligned} \tag{3}$$

Since p is known, $\partial_2 \log p$ is known as well, and we can approximate equation (3) by averaging the expression in the expectation over our sample $(z_i)_{i=1}^n$, substituting our approximations for $f(\theta)$ and $df/d\theta$. Any technique which gives an accurate estimate for $\nabla_2\mathcal{L}$ is also acceptable, and we will see in the case of a Gaussian distribution that a REINFORCE estimator of the gradient is unnecessary. We refer to the approximation of the full gradient $\nabla\mathcal{L} = \nabla_1\mathcal{L} + \nabla_2\mathcal{L}$ obtained by this procedure as $\hat{\nabla}\mathcal{L}$.

4. Theoretical results

In this section, we quantify the performance of PerfGD theoretically. For simplicity, we focus on the specific case where $\mathcal{D}(\theta) = \mathcal{N}(f(\theta), \sigma^2)$ is a one-dimensional Gaussian with fixed variance, and our model also has a single parameter $\theta \in \mathbb{R}$. We also use a single previous step to estimate $df/d\theta$ (i.e. $H = 1$). For results with longer estimation horizon ($H > 1$) and stochastic errors on $\hat{f}$, see Appendix E.

Below we state our assumptions on the mean function f, the loss function ℓ, and the errors on our estimator $\hat{f}$ of f.

1. The mean function f has bounded first and second derivatives: $|f'(\theta)| \leq F$ and $|f''(\theta)| \leq M \;\forall\, \theta \in \mathbb{R}$.
2. The estimator $\hat{f}$ for f has bounded error: $\hat{f}(\theta) = f(\theta) + \varepsilon(\theta)$ and $|\varepsilon(\theta)| \leq \delta$.
3. The loss is bounded: $|\ell(z;\theta)| \leq \ell_{\max}$.
4. The gradient estimator $\hat{\nabla}\mathcal{L}$ is bounded from below and above: $g \leq |\hat{\nabla}\mathcal{L}| \leq G$.
5. The true performative gradient is upper bounded by G: $|\nabla\mathcal{L}| \leq G$.
6. The true performative gradient is L_{Lip}-Lipschitz: $|\nabla\mathcal{L}(\theta) - \nabla\mathcal{L}(\theta')| \leq L_{\text{Lip}}|\theta - \theta'|$.
7. The performative loss is convex.

Lastly, we assume that all of the integrals and expectations involved in computing $\hat{\nabla}\mathcal{L}$ are computed exactly, so the error comes only from the estimate $\hat{f}$ and the finite difference used to approximate $df/d\theta$.

We remark that Assumption 7 is nontrivial. Indeed, (Perdomo et al., 2020) give an example where the point loss ℓ is strongly convex, but the performative loss resulting from a particular distribution map is concave. (Miller et al., 2021) studied conditions under which the performative loss is convex. However, their work makes a similarly strong assumption that the decoupled performative loss is convex in the second (distributional) argument. Finding easily empirically verifiable conditions under which the performative loss is convex is an interesting open problem.

We will prove that PerfGD converges to an approximate critical point, i.e. a point where $\nabla\mathcal{L} \approx 0$. The lower bound in Assumption 4 can therefore be thought of as a stopping criterion for PerfGD, i.e. when the gradient norm drops below the threshold g, we terminate. As a corollary to our main theorem, we will show that this criterion can be taken to be $g \propto \delta^{1/5}$. We begin by bounding the error of our approximation $\hat{\nabla}\mathcal{L}_t$. In what follows, $\nabla\mathcal{L}_t = \nabla\mathcal{L}(\theta_t)$ and $\hat{\nabla}\mathcal{L}_t = \hat{\nabla}\mathcal{L}(\theta_t)$.

Lemma 1. *With step size η, the error of the performative gradient is bounded by*

$$|\hat{\nabla}\mathcal{L}_t - \nabla\mathcal{L}_t| = \mathcal{O}\left(\ell_{\max}\left(MG\eta + \frac{\delta}{g}\frac{1}{\eta} + F\delta\sqrt{\log\frac{1}{\delta}}\right)\right).$$

Next, we quantify the convergence rate of PerfGD as well as the error of the final point to which it converges.

Theorem 2. *With step size*

$$\eta = \sqrt{\frac{1}{MG^2T} + \frac{\delta}{MGg}},$$

the iterates of PerfGD satisfy

$$\min_{1\le t\le T}|\nabla\mathcal{L}_t|^2 = \max\left\{\mathcal{O}\left(\ell_{\max}\sqrt{\frac{MG^2}{T} + \frac{MG^3\delta}{g}}\right), \mathcal{O}\left(g^2 + \ell_{\max}^2\left(\frac{M}{T} + \frac{MG\delta}{g}\right)\right)\right\}.$$

Theorem 2 shows that PerfGD converges to an approximate critical point. A guarantee on the gradient norm can easily be translated into a bound on the distance of θ_t to θ_{OPT} with additional regularity assumptions on the performative loss. For instance, if $\mathcal{L}$ is α-strongly convex, then a standard result from convex analysis implies that $|\theta_t - \theta_{\mathrm{OPT}}| \le \alpha^{-1}|\nabla\mathcal{L}_t|$. The proof of Theorem 2 amounts to combining the error bound from Lemma 1 with a careful analysis of gradient descent for L_{Lip}-smooth functions. For details, see the appendix.

As as a corollary to Theorem 2, we see that we can choose the stopping criterion to be $g \propto \delta^{1/5}$.

Corollary 3. *With stopping criterion $g \propto \delta^{1/5}$, the iterates of PerfGD satisfy*

$$\min_{1\le t\le T}|\nabla\mathcal{L}_t|^2 = \mathcal{O}\left(\ell_{\max}\sqrt{\frac{MG^2}{T} + MG^3\delta^{4/5}}\right).$$

In particular, this suggests that the error in PerfGD will stop decaying after approximately $T \propto \delta^{-4/5}$ iterations.

The corollary follows by matching the leading order behavior in δ of the two terms in the max in Theorem 2. Note that the bound in Lemma 1 still holds for the iterate in which we have $|\hat{\nabla}\mathcal{L}_t| < g$ since we estimate $\hat{\nabla}\mathcal{L}_t$ using the *previous* iterate. If we stop at the first violation of the gradient lower bound criterion, then we had $|\hat{\nabla}\mathcal{L}_{t-1}| \ge g$, and Lemma 1 still applies.

The above results are stated with deterministically bounded errors on our estimates for $f(\theta)$. The following corollary shows that the guarantees of PerfGD still hold with high probability when the error in f comes from a finite sample (and is therefore not deterministically bounded).

Corollary 4. *Suppose at each time t, we collect $n_t \ge n$ samples from $\mathcal{D}(\theta_t)$ and compute $\hat{f}_t$ using the sample average. If $n \ge \frac{2\sigma^2}{\delta^2}\log\frac{2T}{\gamma}$, then with probability at least $1-\gamma$, we have $|\hat{f}_t - f_t| < \delta$ for all $1 \le t \le T$.*

4.1. Generalization to higher dimensions

Our results also hold in dimensions $d > 1$ with a mild dependence on the dimension. In this case, in order to ensure that we have sufficiently damped the effect of errors in $\hat{f}_t$ on our finite difference approximation for $df/d\theta$, we require a lower bound on the minimum singular value of the "step matrix" $\Delta\theta$. (We denote this singular value by $\sigma_{\min}(\Delta\theta)$.) Namely, we require that $\sigma_{\min}(\Delta\theta) \ge \eta g$. This is analogous to Assumption 4. The following result holds for any estimation horizon $H \ge d$.

Theorem 5. *With step size*

$$\eta = \sqrt{\frac{g}{Md^{3/2}G^3H^3}\frac{1}{T} + \frac{1}{MdG^2H^{5/2}}\delta},$$

the iterates of PerfGD satisfy

$$\min_{1\le t\le T}|\nabla\mathcal{L}_t|^2 = \mathcal{O}\left(\ell_{\max}\sqrt{\frac{Md^{3/2}G^2H^3}{g}\frac{1}{T} + \frac{Md^2G^2H^4}{g^2}\delta}\right).$$

Note that we do not exactly recover the one dimensional result by setting $d = H = 1$. This is due to some cancellation in the analysis unique to the $d = H = 1$ setting.

We verify empirically $\sigma_{\min}(\Delta\theta)$ is bounded away from 0 in our high-dimensional experiments (see the appendix). In practice, if this singular value becomes to small, one may need to manually enforce movement in other directions, e.g. by deploying some perturbations of the current θ along a deterministic frame or by adding e.g. Gaussian or uniform spherical perturbations. Relaxing this above assumption along the lines of the "stopping criterion" in one dimension is an interesting direction for future work.

Interpretation of main results In general, PerfGD should be thought of as simply mimicking gradient descent (GD), and we should expect PerfGD and GD to exhibit similar behavior even in the non-convex case. In particular, the results of (Lee et al., 2016) suggest that PerfGD should converge to a local minimizer even with a non-convex performative loss. Furthermore, if Assumptions 1-7 are satisfied, then PerfGD converges to a minimum as $\delta \to 0$ and $T \to \infty$.

5. Applying PerfGD

In this section we will show by way of several examples that this simple framework can easily handle performative effects in many practical contexts. We again focus on Gaussian distributions with fixed covariance for concreteness, i.e. $\mathcal{D}(\theta) = \mathcal{N}(f(\theta), \Sigma)$. Using the terminology from Section 3.1, for a d-dimensional Gaussian we have $p(z; w) = \frac{1}{\sqrt{(2\pi)^d \det \Sigma}} e^{-\frac{1}{2}(z-w)^\top \Sigma^{-1}(z-w)}$ and $f(\theta)$ is the mean of the Gaussian. Our estimator $\hat{f}$ for $f(\theta)$ is just the sample average: $\hat{f}((z_i)_{i=1}^n) = \frac{1}{n}\sum_{i=1}^n z_i$. Of particular note is the form that $\nabla_2 \mathcal{L}$ takes in this case. An elementary calculation yields

$$\begin{aligned}\nabla_2 \mathcal{L} &= \int \ell(z;\theta) \frac{df}{d\theta}^\top \Sigma^{-1}(z - f(\theta)) p(z; f(th))\, dz \\ &= \mathbb{E}_{\mathcal{D}(\theta)}\left[\ell(z;\theta)\frac{df}{d\theta}^\top \Sigma^{-1}(z - f(\theta))\right]. \qquad (4)\end{aligned}$$

Equation (4) shows that we can approximate $\nabla_2 \mathcal{L}$ by averaging the expression inside the expectation over our sample from $\mathcal{D}(\theta)$ without the need for the REINFORCE trick or other more complicated methods of numerically evaluating the integral. Specifically, we have

$$\hat{\nabla}_2 \mathcal{L}\left(f, \frac{df}{d\theta}\right) = \frac{1}{n}\sum_{i=1}^n \ell(z_i;\theta)\frac{df}{d\theta}^\top \Sigma^{-1}(z_i - f).$$

We present each of the following experiments in a fairly general form. For all of the specific constants we used for both data generation and training, see the appendix. In all of the figures below, the shaded region denotes the standard error of the mean over 10 trials for the associated curve. We also note that in all cases except for the high-dimensional pricing experiment (Figure 3), RRM exhibited highly oscillatory behavior and failed to converge. We have therefore omitted it from the plots for visual clarity.

In addition to RRM and RGD, we also compare to the black-box DFO algorithm of (Flaxman et al., 2005), denoted FLX in the plots. We note that FLX's current *internal* estimate for the optimal point is not the same as the points which it actually *queries*. In our case, a query corresponds to actually deploying a model, so the performative loss actually incurred by using FLX corresponds to the queries rather than the internal estimate. This is significant because FLX is sensitive to noise in the function value oracle, and to compensate it needs to use larger perturbations. This means that FLX may deploy highly suboptimal models en route to finding one with low performative loss.

5.1. Toy examples: Mixture of Gaussians and nonlinear mean

Here we verify that PerfGD converges to the performatively optimal point for some simple problems similar (but slightly more difficult than) to the one introduced in Section 2.2. In both cases we take $\ell(z;\theta) = \theta z$ and $\Theta = [-1, 1]$.

For the first example, we set $\mathcal{D}(\theta) = \mathcal{N}(f(\theta), \sigma^2)$ with $f(\theta) = \sqrt{a_1\theta + a_0}$. Since f is nonlinear, estimating $df/d\theta$ is more challenging. Despite this fact, PerfGD still finds the optimal point. The results are shown in Figure 1 below.

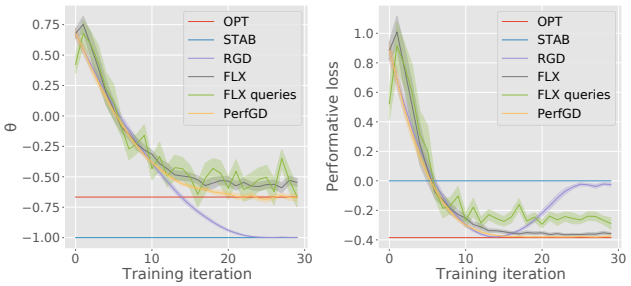

Figure 1. Results for a modified version of the toy example introduced in Section 2.2. OPT denotes the performatively optimal point θ_{OPT}, and STAB denotes the performatively stable point θ_{STAB}. Since the mean is nonlinear in θ, estimating $df/d\theta$ with finite differences is more challenging. In spite of this, PerfGD still converges to the optimal point. FLX also finds a decent point, though its query points are worse than those of PerfGD.

For the second example, we set $\mathcal{D}(\theta) = \gamma\mathcal{N}(f_1(\theta), \sigma_1^2) + (1-\gamma)\mathcal{N}(f_2(\theta), \sigma_2^2)$. Here both of the means are linear in θ, i.e. $f_i(\theta) = a_{i,1}\theta + a_{i,0}$. We apply PerfGD where the true cluster assignment for each point is known; in this case, PerfGD converges to θ_{OPT} exactly and achieves optimal performative loss. The results are shown in Figure 2 below.

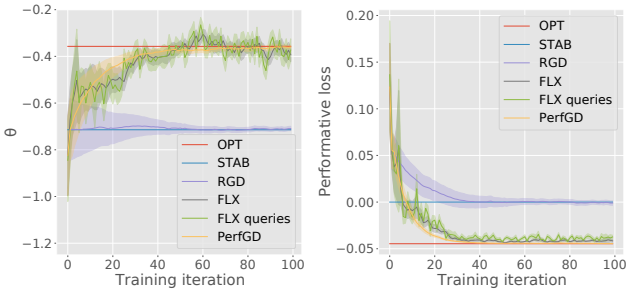

Figure 2. Results for a modified version of the toy example introduced in Section 2.2 with a mixture of Gaussians. We supply the cluster label for each point. PerfGD is again able to converge to the minimum, while RGD converges to a suboptimal point. FLX also does well in this setting, though its convergence is noisier than PerfGD's.

5.2. Pricing

We next examine a generalized version of the problem introduced in Section 2.2. Let θ denote a vector of prices for various goods which we, the distributor, set. A vector z denotes a customer's demand for each good. Our goal is to maximize our expected revenue $\mathbb{E}_{\mathcal{D}(\theta)}[\theta^\top z]$. (In other words, we set the loss function $\ell(z;\theta) = -\theta^\top z$.) Assuming $\mathcal{D}(\theta) = \mathcal{N}(f(\theta), \Sigma)$, we can directly apply Algorithm 3 with the functions $\hat{f}$ and $\hat{\nabla}\mathcal{L}_2$ defined at the beginning of the section to compute the optimal prices.

Experiments For this experiment, we work in a higher dimensional setting with $d = 5$. We define $\Theta = [0, 5]^d$ and $f(\theta) = \mu_0 - \varepsilon\theta$. (That is, the mean demand for each good decreases linearly as the price increases.)

Our results are shown in Figure 3. For this case, we can compute θ_{OPT} and θ_{STAB} analytically. The performative revenue for each of these points is shown on the right side of the figure. As expected, PerfGD converges smoothly to the optimal prices, while RGD converges to the only fixed point which produces suboptimal revenue. In this case, RRM (not shown) stays fixed at $\theta_{\text{RRM}} = [5, 5, \ldots, 5]^\top$.

We remark that our theoretical results for PerfGD in dimensions $d > 1$ require a lower bound assumption on the minimum singular value of the matrix $\Delta\theta$. We empirically verified that this minimum singular value stays bounded away from 0, so our theory still holds for this high dimensional experiment. A plot of the minimum singular value throughout training can be found in the appendix.

Figure 3. Results for performative pricing. RRM (not shown) stays fixed at $\theta_{\text{RRM}} = 5 \cdot \mathbf{1}$, i.e. the vector with all entries equal to 5. Note that PerfGD follows RGD for the first several steps as part of the initialization phase. After this phase, the accurate estimate for the second part of the performative gradient allows PerfGD to reverse trajectory towards θ_{OPT}. While FLX finds a good *internal* estimate for the performatively optimal price vector, it must deploy large perturbations of its internal estimate to overcome the noise from the finite sample. As such, it deploys highly suboptimal price vectors.

5.3. Binary classification

Suppose our goal is to predict a label $y \in \{0, 1\}$ using features $x \in \mathbb{R}^d$. We assume that the label $y \sim \text{Bernoulli}(\gamma)$, and that $x|y \sim \mathcal{N}(f^y(\theta), \Sigma_y)$. The performative loss can then be written as

$$\begin{aligned}\mathcal{L}(\theta) = (1-\gamma)\mathbb{E}_{\mathcal{N}(f^0(\theta),\Sigma_0)}[\ell(x, 0; \theta)] \\ +\gamma\mathbb{E}_{\mathcal{N}(f^1(\theta),\Sigma_1)}[\ell(x, 1; \theta)]\end{aligned} \tag{5}$$

We can apply the general PerfGD method to each of the terms in (5) to obtain an approximate stochastic gradient. (We treat the features of the data with label $y = 0$ as the dataset for the first term, and the features of the data with label $y = 1$ as the dataset for the second term.)

Experiments Here we work with a synthetic model of the spam classification example. We will classify emails with a logistic model, and we will allow a bias term. (That is, our model parameters $\theta = (\theta_0, \theta_1)^\top \in \mathbb{R}^2$. Given a real-valued feature x, our model outputs $h_\theta(x) = 1/(1 + e^{-\theta_0 - \theta_1 x})$.) We let the label $y = \mathbb{1}\{\text{email is spam}\}$. For this case we assume that the distribution of the feature given the label is the performative aspect of the distribution map: spammers will try to alter their emails to slip past the spam filter, while people who use email normally will not alter their behavior according to the spam filter. To this end, we suppose that

$$x|y = 0, \theta \sim \mathcal{N}(\mu_0, \sigma_0^2), \qquad x|y = 1, \theta \sim \mathcal{N}(f(\theta), \sigma_1^2).$$

We note that assuming Gaussian features is in fact a realistic assumption in this case. Indeed, (Li et al., 2020) shows that state-of-the-art performance on various NLP tasks can be achieved by transforming standard BERT embeddings so that they look like a sample from an isotropic Gaussian.

For this experiment, we set $f(\theta) = \mu_1 - \varepsilon\theta_1$. Such a distribution map arises from the strategic classification setting described in (Perdomo et al., 2020) in which the spammers optimize a non-spam classification utility minus a quadratic cost for changing their features. We use ridge-regularized cross-entropy loss for ℓ.

Our results are shown in Figure 4. The improved estimate of the performative gradient given by PerfGD results in roughly a 9% reduction in the performative loss over RGD. In this case, RRM (not shown) oscillates between two values of θ which both give significantly higher performative loss than either RGD or PerfGD. Despite an extensive hyperparameter grid search, FLX (also not shown) was unable to converge for this example.

5.4. Regression

This setting is essentially a generalized version of the performative mean estimation problem in (Perdomo et al., 2020). For simplicity, assume that the marginal distribution of x

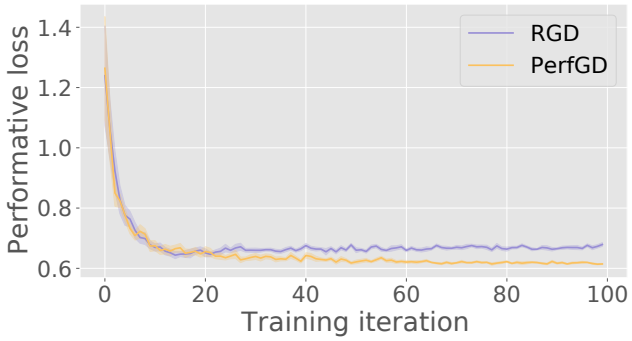

Figure 4. PerfGD vs. RGD for performative logistic regression. By taking into account the change in distribution, PerfGD is able to achieve a lower performative loss than RGD. FLX (not shown) did not converge for this example.

is independent of θ. Assuming that $y|x \sim \mathcal{N}(f(x,\theta), \sigma^2)$, the performative loss becomes

$$\mathcal{L}(\theta) = \mathbb{E}_x[\mathbb{E}_{\mathcal{N}(f(x,\theta),\sigma^2)}[\ell(x,y;\theta)]]. \tag{6}$$

The inner expectation has the required form to apply PerfGD. However, since x takes continuous values, we will in general have only one sample to approximate the inner expectation in (6), leading to heavily biased or innacurate estimates for the required quantities in (2). This leaves us with two options: we can either use techniques for debiasing the required quantities and apply PerfGD directly, or we can use a reparameterization trick and a modified version of PerfGD. Here we present the latter approach.

We assume that the response y follows a linear model, i.e. $y = \beta(\theta)^\top x + \varepsilon, \varepsilon \sim \mathcal{N}(0, \sigma^2)$. The performative loss can then be written as

$$\mathcal{L}(\theta) = \mathbb{E}_{x,\varepsilon}[\ell(x, \beta(\theta)^\top x + \varepsilon; \theta)]. \tag{7}$$

Since we have removed the dependence of the distribution on θ, we can easily compute the gradient:

$$\nabla\mathcal{L}(\theta) = \mathbb{E}_{\mathcal{D}(\theta)}[\nabla_\theta \ell(x,y;\theta)] + \mathbb{E}_{\mathcal{D}(\theta)}\left[\frac{\partial \ell}{\partial y}\frac{d\beta}{d\theta}^\top x\right].$$

We can first estimate β via e.g. regularized ordinary least squares, then estimate $d\beta/d\theta$ via finite differences as in the general setting (2): $\frac{d\beta}{d\theta} \approx \Delta\beta(\Delta\theta)^\dagger$.

Experiments For simplicity, we use one-dimensional linear regression parameters $\theta \in \mathbb{R}$. The feature x is drawn from a fixed distribution $x \sim \mathcal{N}(\mu_x, \sigma_x^2)$, and the performative coefficient $\beta(\theta)$ of $y|x$ has the form $\beta(\theta) = a_0 + a_1\theta$. We use ridge-regularized squared loss for ℓ.

Our results are summarized in Figure 5. In this case, there is a large gap between θ_{OPT} and θ_{STAB}. As expected, PerfGD converges smoothly to θ_{OPT}, while in this case both RGD and RRM converge to θ_{STAB}. The improvement of PerfGD over RGD and RRM results in a factor of more than an order of magnitude in reduction of the performative loss.

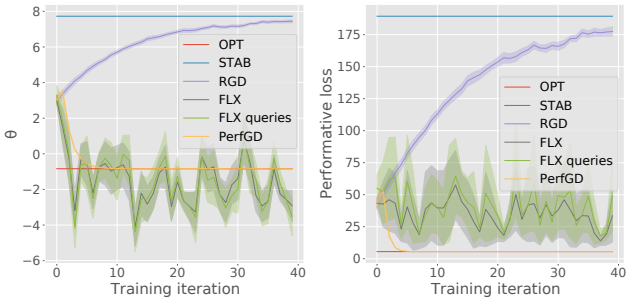

Figure 5. Results for performative linear regression. As expected, RGD converges to the performatively stable point, but in this case the stable point is very far from the performative optimum. PerfGD converges to OPT and incurs a much lower performative loss than RGD. FLX outperforms RGD, but its convergence is noisy and generally worse than PerfGD.

6. Conclusion

We addressed the setting of modeling when the data distribution reacts to the model's parameters, i.e. performative distribution shift. We verified that existing algorithms meant to address this setting in general converge to a suboptimal point in terms of the performative loss. We then introduced a new algorithm, PerfGD, which computes an estimate for the performative gradient under some parametric assumptions on the performative distribution. We proved theoretical results on the convergence of the method, and confirmed via several empirical examples that PerfGD outperforms existing algorithms such as RGD, RRM, and a black-box DFO algorithm (FLX). The accuracy and iteration requirement are both practically feasible, as many ML systems have regular updates every few days.

A natural direction for future work is the extension of our methods to nonparametric distributions. Another direction which may prove fruitful is to improve the estimation of the derivative $df/d\theta$. Finally, methods specifically tailored to deal with high-dimensional data are also of interest.

Acknowledgements

We thank the anonymous reviewers for their detailed and helpful feedback. J.Z. is supported by NSF CAREER 1942926 and grants from the Silicon Valley Foundation and the Chan–Zuckerberg Initiative. L.Y. is supported by the Scientific Discovery through Advanced Computing program, and by the National Science Foundation DMS-1818449. Z.I. is supported by SIGF.

References

Ajalloeian, A. and Stich, S. Analysis of sgd with biased gradient estimators. *arXiv*, 2008.00051, 2020.

Bechavod, Y., Ligett, K., Wu, Z., and Ziani, J. Causal feature discovery through strategic modification. *arXiv*, 2002.07024, 2020.

Bergemann, D. and Morris, S. Robust mechanism design. *Yale: Cowles Foundation Working Papers*, 2003.

Besbes, O., Gur, Y., and Zeevi, A. Non-stationary stochastic optimization. *Operations Research*, 63(5):1227–1244, 2015.

Björkegren, D., Blumenstock, J., and Knight, S. Manipulation-proof machine learning. *arXiv*, 2004.03865, 2020.

Brown, G., Hod, S., and Kalemaj, I. Performative prediction in a stateful world. *arXiv*, 2011.03885, 2020.

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

Cai, Y., Daskalakis, C., and Papadimitriou, C. Optimum statistical estimation with strategic data sources. In *Journal of Machine Learning Research*, volume 40, pp. 1–17, 2015.

Chen, Y., Podimata, C., Procaccia, A. D., and Shah, N. Strategyproof Linear regression in high dimensions. In *ACM EC 2018 - Proceedings of the 2018 ACM Conference on Economics and Computation*, pp. 9–26, 2018.

Chen, Y., Liu, Y., and Podimata, C. Learning strategy-aware linear classifiers. In *NeurIPS*, 2020.

Dalvi, N., Domingos, P., Mausam, Sanghai, S., and Verma, D. Adversarial classification. In *KDD-2004 - Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 99–108, 2004.

Drusvyatskiy, D. and Xiao, L. Stochastic optimization with decision-dependent distributions. *arXiv*, 2011.11173, 2020.

Duchi, J., Jordan, M., Wainwright, M., and Wibisono, A. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 61, 12 2015.

Flaxman, A. D., Kalai, A. T., and McMahan, H. B. Online convex optimization in the bandit setting: Gradient descent without a gradient. *SODA*, 2005.

Hardt, M., Megiddo, N., Papadimitriou, C., and Wootters, M. Strategic classification. In *ITCS 2016 - Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pp. 111–122, 2016.

Khajehnejad, M., Tabibian, B., Schölkopf, B., Singla, A., and Gomez-Rodriguez, M. Optimal decision making under strategic behavior. *arXiv*, 1905.09239, 2019.

Kleinberg, J. and Raghavan, M. How do classifiers induce agents to invest effort strategically? In *ACM EC 2019 - Proceedings of the 2019 ACM Conference on Economics and Computation*, pp. 825–844, 2019.

Kleinberg, J., Ludwig, J., Mullainathan, S., and Obermeyer, Z. Prediction policy problems. In *American Economic Review*, volume 105, pp. 491–495, 2015.

Lattimore, T. Improved regret for zeroth-order adversarial bandit convex optimisation. *arXiv*, 2006.00475, 2020.

Lee, J. D., Simchowitz, M., Jordan, M. I., and Recht, B. Gradient descent only converges to minimizers. In *Conference on learning theory*, pp. 1246–1257. PMLR, 2016.

Li, B., Zhou, H., He, J., Wang, M., Yang, Y., and Li, L. On the sentence embeddings from pre-trained language models. In *EMNLP*, 2020.

Mendler-Dünner, C., Perdomo, J. C., Zrnic, T., and Hardt, M. Stochastic optimization for performative prediction. *NeurIPS*, 2020.

Miller, J., Milli, S., and Hardt, M. Strategic classification is causal modeling in disguise. In *ICML*, 2020.

Miller, J., Perdomo, J. C., and Zrnic, T. Outside the echo chamber: Optimizing the performative risk. *arXiv*, 2102.08570, 2021.

Munro, E. Learning to personalize treatments when agents are strategic. *arXiv*, 2011.06528, 2020.

Perdomo, J., Zrnic, T., Mendler-Dünner, C., and Hardt, M. Performative prediction. In *ICML*, 2020.

Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., and Lawrence, N. D. *Dataset Shift in Machine Learning*. The MIT Press, 2009.

Sassen, S. Do economists make markets? on the performativity of economics. *American Journal of Sociology*, 2008.

Shavit, Y., Edelman, B., and Axelrod, B. Learning from strategic agents: Accuracy, improvement, and causality. *arXiv*, 2002.10066, 2020.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3):229–256, May 1992.