Adversarial Representation Learning With Closed-Form Solvers

Bashir Sadeghi, Lan Wang, Vishnu Boddeti
Michigan State University
sadeghib,wanglan3,vishnu@msu.edu

Abstract

Adversarial representation learning aims to learn data representations for a target task while removing unwanted sensitive information at the same time. Existing methods learn model parameters iteratively through stochastic gradient descent-ascent which is often unstable in practice. To overcome this challenge, we adopt closed-form solvers for the adversary and target predictors by modeling them as kernel ridge regressors, resulting in a more stable one-shot optimization, dubbed OptNet-ARL. Numerical experiment on CelebA dataset demonstrates the utility of our approach for mitigating leakage of private information from learned representation.

1. Introduction

Adversarial Representation Learning (ARL) is a promising framework that affords explicit control over unwanted information in learned data representations. The basic idea of ARL [16] is to introduce a proxy adversary and limit its ability to extract sensitive information during training. The adversary acts as a proxy for the inductive biases necessary for controlling unwanted information in the representation. Such an approach has practically been employed in various applications, such as, learning unbiased and fair representations [11], learning controllable representations invariant to sensitive attributes [16], mitigating leakage of sensitive information [12, 13], unsupervised domain adaption [3], learning flexibly fair representation [15] etc.

The ARL framework is a three-player minimax game between an encoder E, a predictor T, and an adversary A, as illustrated in Figure 1.Target predictor T seeks to extract target information and make correct prediction on target task; Adversary A seeks to extract sensitive information from learned representation; Encoder E seeks to learn a data representation that aids the target predictor and hinders the adversary at the same time. In most ARL settings, while the encoder is a deep neural network, the target predictor and adversary are typically shallow neural networks, all of which are difficult to optimize simultaneously.

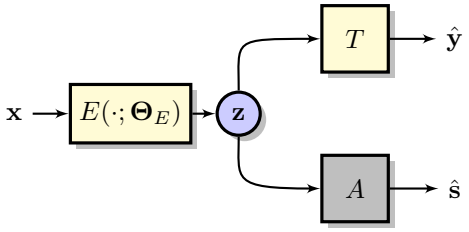


Figure 1: **Adversarial Representation Learning** consists of three players, an encoder E that obtains a compact representation $\mathbf{z}$ of input data $\mathbf{x}$, predictors T and S that seek to extract a desired target $\mathbf{y}$ and sensitive $\mathbf{s}$ attribute, respectively from the embedding.

In such problems the solution of interest is the Nash equilibrium rather than the local minima of each objective. Unfortunately, finding the Nash equilibrium for general games where the players are modeled as neural networks is an intractable problem [1]. The vanilla algorithm for learning the parameters of the encoder, target and adversary networks is gradient descent-ascent (GDA) [12, 16], where the players take a gradient step simultaneously. However, applying gradient descent, including its stochastic version, is not an optimal strategy for ARL and is known to suffer from many drawbacks. Firstly, GDA has undesirable convergence properties; it fails to converge to a local minimax and can converge to fixed points that are not local minimax, while being very unstable and slow in practice [7]. Secondly, GDA exhibits strong rotation around fixed points, which requires using very small learning rates [1] to converge. Numerous solutions have been proposed recently to address the aforementioned challenges and improve the optimization dynamics of multi-player games. These approaches, however, seek to obtain solutions to the minimax optimization problem in the general case, where each player is modeled as a complex neural network.

In this paper we take a different perspective and propose an alternative solution for adversarial representation learning. Our key insight is to replace the shallow neural networks with other analytically tractable models with similar or higher capacity. We propose to adopt simple learning

algorithms that admit closed-form solutions, such as linear or kernel ridge regression for the target and adversary, while modeling the encoder as a deep neural network. Note that, universal kernels (e.g., Gaussian) can potentially be of infinite capacity. Crucially, such models are particularly suitable for ARL and afford numerous advantages, including, (1) closed-form solution allows learning problems to be optimized globally and efficiently, (2) the simplicity and differentiability allows us to backpropagate through the learning process, (3) practically it resolves the notorious rotational behaviour of iterative minimax gradient dynamics, resulting in an optimization that is empirically stable, reliable, and more effective while also converging faster. We refer to our proposed algorithm as OptNet-ARL.

Notation: Scalars are denoted by regular lower case or Greek letters, e.g., n, λ. Vectors are boldface lowercase letters, e.g., $\mathbf{x}$, $\mathbf{y}$; Matrices are uppercase boldface letters, e.g., $\mathbf{X}$. The pseudo inverse of $\mathbf{X}$ is denoted by $\mathbf{X}^\dagger$. A $n \times n$ identity matrix is denoted by $\mathbf{I}$, sometimes with a subscript indicating its size, e.g., $\mathbf{I}_n$. Centered (mean subtracted w.r.t columns) data matrix is indicated by "~", e.g., $\tilde{\mathbf{X}}$. Assume that $\mathbf{X}$ contains n columns, then $\tilde{\mathbf{X}} = \mathbf{X}\mathbf{D}$ where $\mathbf{D} = \mathbf{I}_n - \frac{1}{n}\mathbf{1}\mathbf{1}^T$ and $\mathbf{1}$ denotes a vector of ones with length of n. Given matrix $\mathbf{M} \in \mathbb{R}^{m\times m}$, we use $\mathrm{Tr}[\mathbf{M}]$ to denote its trace; its Frobenius norm is denoted by $\|\mathbf{M}\|_F$, which is related to the trace as $\|\mathbf{M}\|_F^2 = \mathrm{Tr}[\mathbf{M}\mathbf{M}^T]$. The subspace spanned by the columns of $\mathbf{M}$ is denoted by $\mathcal{R}(M)$ or simply $\mathcal{M}$ (in calligraphy). The orthogonal complement of $\mathcal{M}$ is denoted by $\mathcal{M}^\perp$ and the orthogonal projectors onto $\mathcal{M}$ and $\mathcal{M}^\perp$ are denoted by $P_{\mathcal{M}}$ and $P_{\mathcal{M}^\perp}$, respectively.

2. Problem Setting

Let the data matrix $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_n] \in \mathbb{R}^{d\times n}$ be n realizations of d-dimensional data, $\mathbf{x} \in \mathbb{R}^d$. Similarly, we denote n realizations of sensitive attribute vector $\mathbf{s} \in \mathbb{R}^q$ and target attribute vector $\mathbf{y} \in \mathbb{R}^p$ by matrices $\mathbf{S} = [\mathbf{s}_1, \cdots, \mathbf{s}_n]$ and $\mathbf{Y} = [\mathbf{y}_1, \cdots, \mathbf{y}_n]$, respectively. Treating the attributes as vectors enables us to consider both multi-class classification and regression under the same formulation. Each data sample $\mathbf{x}_k$ is associated with the sensitive attribute $\mathbf{s}_k$ and the target attribute $\mathbf{y}_k$, respectively.

The ARL problem is formulated with the goal of learning parameters of an embedding function $E(\cdot; \mathbf{\Theta}_E)$ that maps a data sample $\mathbf{x}$ to $\mathbf{z} \in \mathbb{R}^r$ with two objectives: (i) aiding a target predictor $T(\cdot; \mathbf{\Theta}_y)$ to accurately infer the target attribute $\mathbf{y}$ from $\mathbf{z}$, and (ii) preventing an adversary $A(\cdot; \mathbf{\Theta}_s)$ from inferring the sensitive attribute $\mathbf{s}$ from $\mathbf{z}$. The ARL problem can be formulated as a bi-level optimization,

$$\begin{aligned} &\min_{\mathbf{\Theta}_E} \min_{\mathbf{\Theta}_y} \mathcal{L}_y\left(T(E(\mathbf{x}; \mathbf{\Theta}_E); \mathbf{\Theta}_y), \mathbf{y}\right) \\ &\quad \text{s.t.} \quad \min_{\mathbf{\Theta}_s} \mathcal{L}_s\left(A(E(\mathbf{x}; \mathbf{\Theta}_E); \mathbf{\Theta}_s), \mathbf{s}\right) \geq \alpha \end{aligned} \tag{1}$$

where $\mathcal{L}_y$ and $\mathcal{L}_s$ are the loss functions (averaged over the training dataset) for the target predictor and the adversary, respectively; $\alpha \in [0, \infty)$ is a user defined value that determines the minimum tolerable loss α for the adversary on the sensitive attribute; and the minimization in the constraint is equivalent to the encoder operating against an optimal adversary. Existing instances of this problem adopt deep neural networks to represent E, T and A and learn their respective parameters $\{\mathbf{\Theta}_E, \mathbf{\Theta}_y, \mathbf{\Theta}_s\}$ through stochastic gradient descent-ascent (SGDA).

Modeling the target predictor and adversary by exact solvers would enable us to learn a better embedding function. The machine learning literature offers a wealth of methods with exact solutions that are appropriate for modeling both the adversary and target predictors. In this paper, we argue for and adopt simple, fast and differentiable methods such as kernel ridge regressors. On one hand, such modeling allows us to obtain the optimal estimators globally for any given encoder $E(\cdot; \mathbf{\Theta}_E)$. On the other hand, kernelized ridge regressors can be stronger than the shallow neural networks that are used in many ARL-based solutions (e.g., [11,12,16]).

Denote the global minimums of the adversary and target estimators as,

$$\begin{aligned} J_y(\mathbf{\Theta}_E) &:= \min_{\mathbf{\Theta}_y} \mathcal{L}_y\left(T(E(\mathbf{x}; \mathbf{\Theta}_E); \mathbf{\Theta}_y), \mathbf{y}\right) \\ J_s(\mathbf{\Theta}_E) &:= \min_{\mathbf{\Theta}_s} \mathcal{L}_s\left(A(E(\mathbf{x}; \mathbf{\Theta}_E); \mathbf{\Theta}_s), \mathbf{s}\right). \end{aligned} \tag{2}$$

The constrained optimization problem in (1) can be alternately solved through its Lagrangian version:

$$\min_{\mathbf{\Theta}_E} \Big\{(1-\lambda) J_y(\mathbf{\Theta}_E) - \lambda J_s(\mathbf{\Theta}_E)\Big\}, 0 \leq \lambda \leq 1 \tag{3}$$

2.1. Closed-Form Adversary and Target Predictor

Consider two reproducing kernel Hilbert spaces (RKHS) $\mathcal{H}_s$ and $\mathcal{H}_y$ of functions from $\mathbb{R}^r$ to $\mathbb{R}$ for adversary and target regressors, respectively. Let a possible corresponding pair of feature maps be $\phi_s(\cdot) \in \mathbb{R}^{r_s}$ and $\phi_y(\cdot) \in \mathbb{R}^{r_y}$ where r_s and r_y are the dimensionality of the resulting features and can potentially approach infinity. The respective kernels for $\mathcal{H}_s$ and $\mathcal{H}_y$ can be represented as $k_s(\mathbf{z}_1, \mathbf{z}_2) = \langle \phi_s(\mathbf{z}_1), \phi_s(\mathbf{z}_2)\rangle_{\mathcal{H}_s}$ and $k_y(\mathbf{z}_1, \mathbf{z}_2) = \langle \phi_y(\mathbf{z}_1), \phi_y(\mathbf{z}_2)\rangle_{\mathcal{H}_y}$. Under this setting (see Fig. 2 for an illustration), we can relate the target and sensitive attributes to any given embedding $\mathbf{z}$ as,

$$\hat{\mathbf{y}} = \mathbf{W}_y \phi_y(\mathbf{z}) + \mathbf{b}_y, \qquad \hat{\mathbf{s}} = \mathbf{W}_s \phi_s(\mathbf{z}) + \mathbf{b}_s \tag{4}$$

where $\mathbf{\Theta}_y = \{\mathbf{W}_y, \mathbf{b}_y\}$ and $\mathbf{\Theta}_s = \{\mathbf{W}_s, \mathbf{b}_s\}$ are the regression parameters, $\mathbf{W}_y \in \mathbb{R}^{p\times r_y}$ and $\mathbf{W}_s \in \mathbb{R}^{q\times r_s}$, $\mathbf{b}_y \in \mathbb{R}^p$ and $\mathbf{b}_s \in \mathbb{R}^q$ respectively.

Let the embeddings of all the data be denoted as $\mathbf{Z} := [\mathbf{z}_1, \cdots, \mathbf{z}_n]$ and the corresponding features maps as $\mathbf{\Phi}_y := [\phi_y(\mathbf{z}_1), \cdots, \phi_y(\mathbf{z}_n)]$ and $\mathbf{\Phi}_s := [\phi_s(\mathbf{z}_1), \cdots, \phi_s(\mathbf{z}_n)]$, respectively. Furthermore, we denote the associated Gram

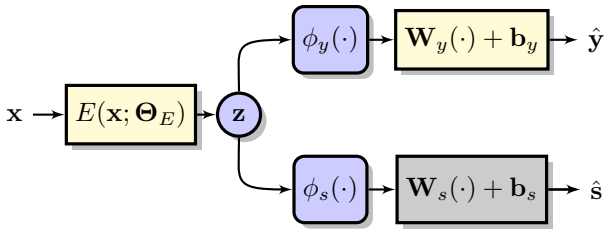


Figure 2: **OptNet-ARL** consists of an encoder E modeled as a deep neural network to obtain a compact representation $\mathbf{z}$ of input data $\mathbf{x}$ and kernelized ridge regressors T and A to predict the target $\mathbf{y}$ and sensitive attributes $\mathbf{s}$, respectively from the embedding $\mathbf{z}$.

matrices by $\mathbf{K}_y = {\mathbf{\Phi}_y}^T\mathbf{\Phi}_y$ and $\mathbf{K}_s = {\mathbf{\Phi}_s}^T\mathbf{\Phi}_s$. A centered (i.e., mean subtracted) kernel matrix $\tilde{\mathbf{K}}$ corresponding to the kernel matrix $\mathbf{K}$ can be obtained [6] as,

$$\tilde{\mathbf{K}} = \tilde{\mathbf{\Phi}}^T\tilde{\mathbf{\Phi}} = (\mathbf{\Phi}\mathbf{D})^T(\mathbf{\Phi}\mathbf{D}) = \mathbf{D}^T\mathbf{K}\mathbf{D}. \tag{5}$$

From the representer theorem [14], the regression parameters can be decomposed as $\mathbf{W}_y = \mathbf{\Lambda}_y\tilde{\mathbf{\Phi}}_y^T$ and $\mathbf{W}_s = \mathbf{\Lambda}_s\tilde{\mathbf{\Phi}}_s^T$ for target and adversary respectively, where $\mathbf{\Lambda}_y \in \mathbb{R}^{p\times n}$ and $\mathbf{\Lambda}_s \in \mathbb{R}^{n\times q}$ are free parameters matrices. As a result, the kernelized regressors in (4) would be equivalent to

$$\hat{\mathbf{y}} = \mathbf{\Lambda}_y\tilde{\mathbf{\Phi}}_y^T\phi_y(\mathbf{z}) + \mathbf{b}_y, \quad \hat{\mathbf{s}} = \mathbf{\Lambda}_s\tilde{\mathbf{\Phi}}_s^T\phi_s(\mathbf{z}) + \mathbf{b}_s \tag{6}$$

Let $J_y(\mathbf{Z})$ and $J_s(\mathbf{Z})$ be regularized minimum MSEs for adversary and target:

$$\begin{aligned} J_y(\mathbf{\Theta}_E) &= \min_{\mathbf{\Lambda}_y,\mathbf{b}_y}\left\{\mathbb{E}\{\|\hat{\mathbf{y}}-\mathbf{y}\|^2\} + \gamma_y\|\mathbf{\Lambda}_y\|_F^2\right\} \\ J_s(\mathbf{\Theta}_E) &= \min_{\mathbf{\Lambda}_s,\mathbf{b}_s}\left\{\mathbb{E}\{\|\hat{\mathbf{s}}-\mathbf{s}\|^2\} + \gamma_s\|\mathbf{\Lambda}_s\|_F^2\right\} \end{aligned}$$

where γ_y and γ_s are regularization parameters for target and adversary regressors, respectively. Then, for any given embedding matrix $\mathbf{Z}$, the minimum MSE for kernelized adversary and target can be obtained as

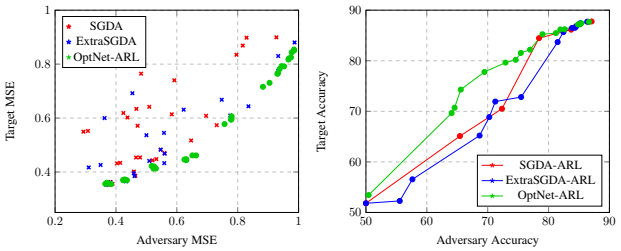
$$\begin{aligned} J_y(\mathbf{\Theta}_E) &= \frac{1}{n}\|\tilde{\mathbf{Y}}\|_F^2 - \frac{1}{n}\left\|P_{\mathcal{M}_y}\begin{bmatrix}\tilde{\mathbf{Y}}^T\\ \mathbf{0}_n\end{bmatrix}\right\|_F^2 \\ J_s(\mathbf{\Theta}_E) &= \frac{1}{n}\|\tilde{\mathbf{S}}\|_F^2 - \frac{1}{n}\left\|P_{\mathcal{M}_s}\begin{bmatrix}\tilde{\mathbf{S}}^T\\ \mathbf{0}_n\end{bmatrix}\right\|_F^2 \end{aligned} \tag{7}$$

where

$$\mathbf{M}_y = \begin{bmatrix}\tilde{\mathbf{K}}_y\\ \sqrt{n\gamma_y}\mathbf{I}_n\end{bmatrix}, \qquad \mathbf{M}_s = \begin{bmatrix}\tilde{\mathbf{K}}_s\\ \sqrt{n\gamma_s}\mathbf{I}_n\end{bmatrix}$$

are both full column rank matrices and a projection matrix for any full column rank matrix $\mathbf{M}$ is

$$P_{\mathcal{M}} = \mathbf{M}(\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T$$



(a) (b)

Figure 3: **CelebA:** (a) Target MSE versus adversary MSE for different values of λ and five different weight initialization of encoder networks for SGDA-ARL, ExtraSDGA-ARL and OptNet-ARL. (b) Non-dominant trade-off between target and adversary accuracy for different values of λ for all methods.

3. Closed-Form Solver Gradient

In order to find the gradient of loss function in (3) with J_y and J_s given in (7), we ignore the constant terms, $\|\tilde{\mathbf{Y}}\|_F$ and $\|\tilde{\mathbf{S}}\|_F$ for which $\tilde{\mathbf{Y}}$ and $\tilde{\mathbf{S}}$ are the mean subtracted versions of $\mathbf{Y}$ and $\mathbf{S}$, respectively. Then, the optimization problem in (3) would be equivalent to

$$\begin{aligned} &\min_{\mathbf{\Theta}_E}\left\{(1-\lambda)\left\|P_{\mathcal{M}_s}\begin{bmatrix}\tilde{\mathbf{S}}^T\\ \mathbf{0}_n\end{bmatrix}\right\|_F^2 - \lambda\left\|P_{\mathcal{M}_y}\begin{bmatrix}\tilde{\mathbf{Y}}^T\\ \mathbf{0}_n\end{bmatrix}\right\|_F^2\right\} \\ &= \min_{\mathbf{\Theta}_E}\left\{(1-\lambda)\sum_{k=1}^{p}\|P_{\mathcal{M}_s}\mathbf{u}_s^k\|^2 - \lambda\sum_{m=1}^{q}\|P_{\mathcal{M}_y}\mathbf{u}_y^m\|^2\right\} \end{aligned} \tag{8}$$

where the vectors $\mathbf{u}_s^k$ and $\mathbf{u}_y^m$ are the k-th and m-th columns of $\begin{bmatrix}\tilde{\mathbf{S}}^T\\ \mathbf{0}_n\end{bmatrix}$ and $\begin{bmatrix}\tilde{\mathbf{Y}}^T\\ \mathbf{0}_n\end{bmatrix}$, respectively. Let $\mathbf{M}$ is an arbitrary matrix function of $\mathbf{\Theta}_E$ and θ be arbitrary element of $\mathbf{\Theta}_E$. Then, from [5] we have

$$\frac{1}{2}\frac{\partial\|P_{\mathcal{M}}\mathbf{u}\|^2}{\partial\theta} = \mathbf{u}^T P_{\mathcal{M}^\perp}\frac{\partial\mathbf{M}}{\partial\theta}\mathbf{M}^\dagger\mathbf{u} \tag{9}$$

where

$$\left[\frac{\partial\mathbf{M}}{\partial\theta}\right]_{ij} = \begin{cases}\nabla_{\mathbf{z}_i}^T([\mathbf{M}]_{ij})\nabla_\theta(\mathbf{z}_i) + \nabla_{\mathbf{z}_j}^T([\mathbf{M}]_{ij})\nabla_\theta(\mathbf{z}_j), & i\le n\\ 0, & \text{else.}\end{cases}$$

Note that, the above equation can be directly used to obtain the gradient of objective function in (8).

4. Experiment on CelebA

The CelebA dataset [10] contains $202,599$ face images of $10,177$ celebrities. Each image contains 40 different binary attributes (e.g., gender, emotion, age, etc.). Images are pre-processed and aligned to a fixed size of 112×96 and we use the official train and test splits. The target task is defined as predicting the presence or absence of high cheekbones

(binary) with the sensitive attribute being smiling/not smiling (binary). We adopt ResNet-18 as the encoder for all methods. In the training stage, the encoder is optimized against kernel ridge regressors in the case of OptNet-ARL and two-layer perceptrons for the baselines as adversary and target. At the inference stage, the encoder is frozen, features are extracted and a new target predictor and adversary are trained. At this stage, for both OptNet-ARL and the baselines, the target and adversary have the same model capacity and optimize the cross-entropy loss for classification. Each experiment on each dataset is repeated five times (different random seeds) and for different trade-off parameters $\lambda \in [0, 1]$.

We consider two baselines, (1) **SGDA-ARL:** vanilla stochastic gradient descent-ascent that is employed by multiple ARL approaches including [2, 8, 11, 12, 16] etc., and (2) **ExtraSGDA-ARL:** a state-of-the-art variant of stochastic gradient descent-ascent that uses an extra gradient step [9] for optimizing minimax games. Specifically, we use the ExtraAdam algorithm from [4].

Stability: Figure 3 shows experimental results. Observe that the baseline solutions span a large part of the trade-off, but at the same time exhibit large instability and unreliability in the solutions. On the other hand, OptNet-ARL (Figure 3 (a)) solutions are very stable while also spanning a more diverse set of solutions on the trade-off front. Finally, Fig. 3 (b) shows the non-dominated trade-off between target and adversary accuracy. OptNet-ARL achieves a significantly better and more diverse trade-off than the baselines.

Scalability: Instead of kernelizing the entire dataset, we approximate it by kernelizing each batch separately to calculate the loss function and backpropagate.

5. Conclusion

Adversarial representation learning is a minimax game theoretic formulation that affords explicit control over unwanted information in learned data representations. Optimization algorithms for ARL such as stochastic gradient descent-ascent (SGDA) and their variants are sub-optimal, unstable and unreliable in practice. In this paper, we introduced OptNet-ARL to address this challenge by employing differentiable closed-form solvers, such as kernelized ridge regressors, to model the ARL players that are downstream from the representation. OptNet-ARL reduces iterative SGDA to a one-shot optimization, leading to a fast, stable and reliable algorithm that out-performs all existing ARL approaches on CelebA dataset.

References

[1] David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. The mechanics of n-player differentiable games. In *International Conference on Machine Learning*, 2018.

[2] Yanai Elazar and Yoav Goldberg. Adversarial removal of demographic attributes from text data. *Empirical Methods in Natural Language Processing*, 2018.

[3] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(1):2096–2030, 2016.

[4] Gauthier Gidel, Hugo Berard, Gaëtan Vignoud, Pascal Vincent, and Simon Lacoste-Julien. A variational inequality perspective on generative adversarial networks. *International Conference on Learning Representations*, 2019.

[5] Gene H Golub and Victor Pereyra. The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate. *SIAM Journal on Numerical Analysis*, 10(2):413–432, 1973.

[6] Arthur Gretton, Ralf Herbrich, Alexander Smola, Olivier Bousquet, and Bernhard Schölkopf. Kernel methods for measuring independence. *Journal of Machine Learning Research*, 6:2075–2129, 2005.

[7] Chi Jin, Praneeth Netrapalli, and Michael I Jordan. What is local optimality in nonconvex-nonconcave minimax optimization? *arXiv preprint arXiv:1902.00618*, 2019.

[8] Byungju Kim, Hyunwoo Kim, Kyungsu Kim, Sungjin Kim, and Junmo Kim. Learning not to learn: Training deep neural networks with biased data. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[9] GM Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 12:747–756, 1976.

[10] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *IEEE International Conference on Computer Vision*, 2015.

[11] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Learning adversarially fair and transferable representations. In *International Conference on Machine Learning*, 2018.

[12] Proteek Roy and Vishnu Naresh Boddeti. Mitigating information leakage in image representations: A maximum entropy approach. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[13] Bashir Sadeghi, Runyi Yu, and Vishnu Naresh Boddeti. On the global optima of kernelized adversarial representation learning. In *IEEE International Conference on Computer Vision*, 2019.

[14] John Shawe-Taylor, Nello Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[15] Jiaming Song, Pratyusha Kalluri, Aditya Grover, Shengjia Zhao, and Stefano Ermon. Learning controllable fair representations. *International Conference on Artificial Intelligence and Statistics*, 2019.

[16] Qizhe Xie, Zihang Dai, Yulun Du, Eduard Hovy, and Graham Neubig. Controllable invariance through adversarial feature learning. In *Advances in Neural Information Processing Systems*, 2017.