Learning Set Functions with Implicit Differentiation

Gözde Özcan [1] **Chengzhi Shi** [1] **Stratis Ioannidis** [1]

Abstract

Ou et al. [1] introduce the problem of learning set functions from data generated by a so-called optimal subset oracle. Their approach approximates the underlying utility function with an energy-based model. This approximation yields iterations of fixed-point updates during mean-field variational inference. However, as the number of iterations increases, automatic differentiation becomes computationally prohibitive due to the size of the Jacobians that are stacked during backpropagation. We address this challenge with implicit differentiation and examine the convergence conditions for the fixed-point iterations. We empirically demonstrate the efficiency of our method on subset selection applications including product recommendation and anomaly detection tasks.

1. Introduction

Many interesting applications operate with set-valued outputs and/or inputs. Examples include product recommendation [2], [3], compound selection [4], set matching [5], set retrieval [6], point cloud processing [7], [8], and set anomaly detection [9], to name a few. Motivated by this, several recent works [10]–[12] apply neural networks to learn set functions from input/function value pairs, assuming access to a dataset generated by a *function value* oracle (see App. A for a review of related literature).

Recently, Ou et al. [1] proposed an approximate maximum likelihood estimation framework under the supervision of a so-called *optimal subset* oracle. In contrast to traditional function value oracles, a label produced by an optimal subset oracle is the subset that maximizes an (implicit) utility set function, in the face of several alternatives. The goal of inference is to learn, in a parametric form, this utility function, under which observed oracle selections are optimal. As MLE is intractable in this setting, Ou et al. [1] propose performing variational inference instead. In turn, they show that approximating the distribution of oracle selections requires solving a fixed-point equation per sample. However, fixed-point iterations may diverge in practice. In addition, Ou et al. implement these iterations by stacking up neural network layers; this makes backpropagation expensive, limiting their experiments to only a handful of iterations.

In this work, we establish a condition under which the fixed-point iterations proposed by Ou et al. [1] are guaranteed to converge. We also propose a more effective gradient computation utilizing the recent advances in implicit differentiation [13]–[15], instead of unrolling the fixed-point iterations via automatic differentiation [16]. This corresponds to differentiating after infinite fixed point iterations, while remaining tractable; we experimentally show that this improves the predictive performance of the inferred models.

2. Problem Setup

In the setting introduced by Ou et al. [1], the aim is to learn set functions from a dataset generated by a so-called *optimal subset* oracle. The dataset $\mathcal{D}$ consists of sample pairs of the form (S^*, V), where $V \subseteq \Omega$ is a set of *options*, i.e., items from a universe Ω, and S^* is the *optimal subset* of V, as selected by an oracle. The goal is to learn a set function $F_{\boldsymbol{\theta}} : 2^\Omega \times 2^\Omega, \to \mathbb{R}$, parameterized by $\boldsymbol{\theta} \in \mathbb{R}^d$, modeling the utility of the oracle, so that

$$S^* = \arg\max_{S \subseteq V} F_{\boldsymbol{\theta}}(S, V), \tag{1}$$

for all pairs $(S^*, V) \in \mathcal{D}$. As a motivating example, consider the case of product recommendations. Given a ground set V of possible products to recommend, a recommender selects an optimal subset $S^* \subseteq V$ and suggests these to a user. In this setting, the function $F_{\boldsymbol{\theta}}(S, V)$ captures, e.g., the recommender objective, the utility of the user, etc. Having access to a dataset of pairs, the goal is to learn $F_{\boldsymbol{\theta}}$, effectively reverse-engineering the objective of the recommender engine, inferring the user's preferences, etc.

2.1. MLE with Energy-Based Modeling

Ou et al. [1] propose an approximate maximum likelihood estimation (MLE) by modeling oracle behavior via a Boltz-

[1]Northeastern University, Department of Electrical and Computer Engineering, Boston, MA 02115, USA. Correspondence to: Gözde Özcan <gozcan@ece.neu.edu>.

Published at the 2^{nd} Differentiable Almost Everything Workshop at the 41^{st} International Conference on Machine Learning, Vienna, Austria. July 2024.

mann energy (i.e., soft-max) model [17]–[20]. They assume that the oracle selection is probabilistic, and the probability that S is selected given options V is given by:

$$p_{\boldsymbol{\theta}}(S \mid V) = \frac{\exp\left(F_{\boldsymbol{\theta}}(S,\, V)\right)}{\sum_{S' \subseteq V} \exp\left(F_{\boldsymbol{\theta}}(S',\, V)\right)}. \tag{2}$$

This is equivalent to Eq. (1), presuming that the utility $F_\theta(\cdot)$ is distorted by Gumbel noise [21]. Then, given a dataset $\mathcal{D} = \{(S_i^*, V_i,)\}_{i=1}^N$, MLE amounts to:

$$\arg\max_{\boldsymbol{\theta}} \textstyle\sum_{i=1}^N \left[\log p_{\boldsymbol{\theta}}(S_i^* \mid V_i)\right]. \tag{3}$$

Notice that multiplying $F_{\boldsymbol{\theta}}$ with a constant $c > 0$ makes no difference in the behavior of the optimal subset oracle in Eq. (1): the oracle would return the same decision under arbitrary re-scaling. However, using $c \cdot F_{\boldsymbol{\theta}}(\cdot)$ in the energy-based model of Eq. (2) corresponds to setting a temperature parameter c in the Boltzmann distribution [17], [21], interpolating between the deterministic selection ($c \to \infty$) in Eq. (1) and the uniform distribution ($c \to 0$).[1]

2.2. Variational Approximation of Energy-Based Models

Learning $\boldsymbol{\theta}$ by MLE is challenging precisely due to the exponential number of terms in the denominator of Eq. (2). Instead, Ou et al. [1] construct an alternative optimization objective via mean-field variational inference as follows. First, they introduce a mean field variational approximation of the density $p_{\boldsymbol{\theta}}$ given by $q(S, V, \boldsymbol{\psi}) = \prod_{i \in S} \psi_i \prod_{i \in V \setminus S} (1 - \psi_i)$, parameterized by $\boldsymbol{\psi}$, where $\boldsymbol{\psi}$ is the probability vector representing the probability of each element $s \in V$ being in the optimal subset S^*. Then, estimation via variational inference amounts to:

$$\begin{aligned}\mathcal{L}(\boldsymbol{\psi}^*, \boldsymbol{\theta}) &= \mathbb{E}_{\mathbb{P}(V,\, S)}[-\log q(S,\, V\,, \boldsymbol{\psi}^*)] \\ &\approx \frac{1}{N} \sum_{i=1}^{N} \left(- \sum_{j \in S_i^*} \log \psi_j^* - \sum_{j \in V_i \setminus S_i^*} \log\left(1 - \psi_j^*\right) \right),\end{aligned} \tag{4}$$

where $\boldsymbol{\psi}^* \in [0, 1]^{|V|}$ is the probability vector obtained by solving:

$$\boldsymbol{\psi}^* = \arg\min_{\boldsymbol{\psi}} \mathbb{KL}(q(S,\, V,\, \boldsymbol{\psi}) \,||\, p_{\boldsymbol{\theta}}(S \mid V)), \tag{5}$$

where $\mathbb{KL}(\cdot||\cdot)$ is the Kullback-Leibler divergence, and $p_{\boldsymbol{\theta}}(S)$ is the energy-based model defined in Eq. (2). In turn, this is found through the ELBO maximization process we discuss in the next section.

[1] From a Bayesian point of view, multiplying $F_{\boldsymbol{\theta}}(\cdot)$ with $c > 0$ yields the posterior predictive distribution under an uninformative Dirichlet conjugate prior per set with parameter $\alpha = e^c$ [17].

2.3. ELBO Maximization

To compute $\boldsymbol{\psi}^*$, Ou et al. [1] show that minimizing the KL divergence in Eq. (5) via maximizing the corresponding evidence lower bound (ELBO) reduces to solving a fixed point equation. In particular, Eq. (5) is equivalent to the following ELBO maximization [22], [23]:

$$\max_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta}) + \mathbb{H}(q(S, V, \boldsymbol{\psi})), \tag{6}$$

where $\mathbb{H}(\cdot)$ is the entropy and $\tilde{F} : [0, 1]^{|V|} \times \mathbb{R}^d \to \mathbb{R}$ is the so-called multilinear extension of $F_{\boldsymbol{\theta}}(S, V)$ [24], given by:

$$\tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta}) = \sum_{S \subseteq V} F_{\boldsymbol{\theta}}(S, V) \prod_{i \in S} \psi_i \prod_{i \in V \setminus S} (1 - \psi_i). \tag{7}$$

Ou et al. show that a stationary point maximizing the ELBO in Eq. (6) must satisfy:

$$\boldsymbol{\psi} - \boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})) = 0, \tag{8}$$

where the function $\boldsymbol{\sigma} : \mathbb{R}^{|V|} \to \mathbb{R}^{|V|}$ is defined as $\boldsymbol{\sigma}(\mathbf{x}) = \left[\sigma(x_i)\right]_{i=1}^{|V|}$ and $\sigma : \mathbb{R} \to \mathbb{R}$ is the sigmoid function, i.e., $\sigma(x) = (1 + \exp(-x))^{-1}$. The detailed derivation of this condition can be found in App. B.1.

Observing that the stationary condition in Eq. (8) is a fixed point equation, Ou et al. propose solving it via the following fixed-point iterations. Given $\boldsymbol{\theta} \in \mathbb{R}^d$,

$$\boldsymbol{\psi}^{(k)} \leftarrow \boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi}^{(k-1)}, \boldsymbol{\theta})), \tag{9}$$

where $k \in \mathbb{N}$, starting from $\boldsymbol{\psi}^{(0)} \in [0, 1]^{|V|}$. Exact computation of the multilinear relaxation defined in Eq. (7) requires an exponential number of terms in the size of V. However, it is possible to efficiently estimate both the multilinear relaxation and its gradient $\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})$ via Monte Carlo sampling (see App. B.2 for details).

2.4. `DiffMF` and Variants

Putting everything together yields the `DiffMF` algorithm introduced by Ou et al. [1]. In particular, they implement the fixed-point iterative update steps in Eq. (9) by executing a fixed number of iterations K, given $\boldsymbol{\theta}$. In their implementation, this amounts to stacking up K layers, each involving an estimate of the gradient of the multilinear relaxation via sampling, and thereby multiple copies of a neural network representing $F_{\boldsymbol{\theta}}(\cdot)$. Subsequently, this extended network is entered in the loss (4), which is minimized w.r.t. $\boldsymbol{\theta}$ via SGD.

They also introduce two variants of this algorithm, regressing $\boldsymbol{\psi}^{(0)}$ as a function of the item features via an extra recognition network, assuming the latter are independent (terming inference in this setting as $\texttt{EquiVSet}_{\text{ind}}$) or correlated by a Gaussian copula [25], [26] (termed $\texttt{EquiVSet}_{\text{copula}}$). Compared to `DiffMF`, both translate to additional initial layers/steps per epoch.

2.5. Challenges

The above approach by Ou et al. [1], and constituent algorithm variants, have two drawbacks. First, the fixed-point iterative updates given in Eq. (9) are not guaranteed to converge to an optimal solution. We indeed observed divergence experimentaly in practice. Without convergence and uniqueness guarantees, the quality of the output, $\boldsymbol{\psi}^{(K)}$, is heavily dependent on the selection of the starting point, $\boldsymbol{\psi}^{(0)}$. Moreover, as these iterations correspond to stacking up layers, each containing multiple copies of $F_{\boldsymbol{\theta}}(\cdot)$ due to sampling, backpropagation is computationally prohibitive both in terms of time as well as space complexity. In fact, poor performance due to lack of convergence, as well as computational considerations, led Ou et al. to set the number of iterations $K \leq 5$ (even $K = 1$) in their experiments. We address both of these challenges in the next section.

3. Our Approach

We (a) establish conditions under which iterations of Eq. (8) converge to a unique solution, by utilizing Banach fixed-point theorem and (b) establish a way to efficiently compute the gradient of the loss at the fixed point by using the implicit function theorem. Our results pave the way to utilize recent tools developed in the context of implicit differentiation [13], [14], [27] to the setting of Ou et al. [1].

3.1. Convergence Condition for the Fixed-Point

Fixed-points can be attracting, repelling, or neutral [28], [29]. We characterize the condition under which the convergence is guaranteed in the following assumption.

Assumption 3.1. Consider the multilinear relaxation $\tilde{F} : [0,1]^{|V|} \times \mathbb{R}^d \to \mathbb{R}$ of $F_{\boldsymbol{\theta}}(\cdot)$, as defined in Eq. (7). For all $\boldsymbol{\theta} \in \mathbb{R}^d$, $\sup_{\boldsymbol{\psi} \in [\mathbf{0},\mathbf{1}]} |\tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})| < \frac{1}{|V|}$.

As discussed in Sec. 2, scaling $F_{\boldsymbol{\theta}}(S, V)$ by a positive scalar amounts to setting the temperature of a Boltzmann distribution. Moreover, neural networks are often Lipschitz-regularized for bounded inputs and weights [30]–[32]. Therefore, for any such Lipschitz neural network, we can satisfy Asm. 3.1 by appropriately setting the temperature parameter of the EBM in Eq. (2). Most importantly, satisfying this condition guarantees convergence:

Theorem 3.2. *Assume a set function $F_{\boldsymbol{\theta}} : 2^V \to \mathbb{R}$ satisfies Asm. 3.1. Then, the fixed-point given in Eq. (8) has a unique solution $\boldsymbol{\psi}^* \in [0,1]^{|V|}$ where $\boldsymbol{\psi}^* = \boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi}^*, \boldsymbol{\theta}))$. Moreover, starting with an arbitrary point $\boldsymbol{\psi}^{(0)} \in [0,1]^{|V|}$, $\boldsymbol{\psi}^*$ can be found via the fixed-point iterative sequence described in Eq. (9) where $\lim_{k\to\infty} \boldsymbol{\psi}^{(k)} = \boldsymbol{\psi}^*$.*

The proof can be found in App. D, and relies on the Banach fixed-point theorem [33]. Theorem 3.2 implies that as long as $\tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})$ is bounded and this bound is inversely correlated with the size of the ground set, we can find a unique solution to Eq. (8), no matter where we start the iterations in Eq. (9).

3.2. Efficient Differentiation through Implicit Layers

Our second contribution is to disentangle gradient computation from stacking layers together, by using the implicit function theorem [34]. This allows us to use the recent work on deep equilibrium models (DEQs) [13], [14].

Define $\boldsymbol{\psi}^*(\cdot)$ to be the map $\boldsymbol{\theta} \mapsto \boldsymbol{\psi}^*(\boldsymbol{\theta})$ induced by Eq. (8); equivalently, given $\boldsymbol{\theta}$, $\boldsymbol{\psi}^*(\boldsymbol{\theta})$ is the (unique by Thm. 3.2) limit point of iterations (9). Observe that, by the chain rule:

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\psi}^*(\boldsymbol{\theta}), \boldsymbol{\theta}) = \nabla_{\boldsymbol{\psi}^*} \mathcal{L}(\boldsymbol{\psi}^*(\boldsymbol{\theta}), \boldsymbol{\theta}) \cdot \partial_{\boldsymbol{\theta}} \boldsymbol{\psi}^*(\boldsymbol{\theta}). \quad (10)$$

The term that is difficult to compute here via backpropagation, that required stacking in Ou et al. [1], is the Jacobian $\partial_{\boldsymbol{\theta}} \boldsymbol{\psi}^*(\boldsymbol{\theta})$, as we do not have the map $\boldsymbol{\psi}^*(\cdot)$ in a closed form. Nevertheless, we can use the implicit function theorem (see Thm. C.4 in App. C) to compute this quantity.

Indeed, to simplify the notation for clarity, we define a function $G : [0,1]^{|V|} \times \mathbb{R}^d \to [0,1]^{|V|}$, where

$$G(\boldsymbol{\psi}(\boldsymbol{\theta}), \boldsymbol{\theta}) \triangleq \boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})) - \boldsymbol{\psi}$$

and rewrite Eq. (8) as $G(\boldsymbol{\psi}(\boldsymbol{\theta}), \boldsymbol{\theta}) = 0$. Using the implicit function theorem, given in App. C, we obtain

$$\underbrace{-\partial_{\boldsymbol{\psi}} G(\boldsymbol{\psi}^*(\boldsymbol{\theta}), \boldsymbol{\theta})}_{A \in \mathbb{R}^{|V| \times |V|}} \underbrace{\partial_{\boldsymbol{\theta}} \boldsymbol{\psi}^*(\boldsymbol{\theta})}_{J \in \mathbb{R}^{|V| \times d}} = \underbrace{\partial_{\boldsymbol{\theta}} G(\boldsymbol{\psi}^*(\boldsymbol{\theta}), \boldsymbol{\theta})}_{B \in \mathbb{R}^{|V| \times d}}. \quad (11)$$

This yields the following way of computing the Jacobian via implicit differentiation:

Theorem 3.3. *Computing $\partial_{\boldsymbol{\theta}} \boldsymbol{\psi}^*(\boldsymbol{\theta})$ is the equivalent of solving a linear system of equations, i.e., $\partial_{\boldsymbol{\theta}} \boldsymbol{\psi}^*(\boldsymbol{\theta}) = A^{-1} B$,*

$$\begin{aligned} A &= I - \Sigma'(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})) \cdot \nabla^2_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta}), \text{ and} \\ B &= \Sigma'(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})) \cdot \partial_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta}), \end{aligned} \quad (12)$$

where $\Sigma'(\boldsymbol{x}) = \operatorname{diag}\left(\left[\sigma'(x_i)\right]_{i=1}^{|V|}\right)$, and $\sigma'(x) = (1 + \exp(-x))^{-2} \cdot \exp(-x)$.

The proof is in App. E. Eq. (11) shows that the Jacobian $\partial_{\boldsymbol{\theta}} \boldsymbol{\psi}^*(\boldsymbol{\theta})$, can be expressed in terms of Jacobians of G at the solution point. This means implicit differentiation only needs the final fixed point value, whereas automatic differentiation via the approach by Ou et al. [1] required all the iterates (see also [14]). In practice, we use JAXopt [27] for its out-of-the-box implicit differentiation support.

4. Experiments

We evaluate our proposed method on two real-world, two synthetic datasets for set anomaly set detection and product

	Datasets	EquiVSet$_{\text{ind}}$		EquiVSet$_{\text{copula}}$		DiffMF		iDiffMF$_2$		iDiffMF$_*$	
		Test JC	**Time** (s)	**Test JC**	**Time** (s)	**Test JC**	**Time** (s)	**Test JC**	**Time** (s)	**Test JC**	**Time** (s)
AD	CelebA	$\underline{53.16 \pm 0.68}$	1822.33 ± 208.36	$\mathbf{54.31 \pm 1.06}$	1889.18 ± 163.30	51.98 ± 0.64	1837.20 ± 221.18	51.17 ± 0.74	2075.74 ± 225.35	51.10 ± 0.83	2206.81 ± 455.08
	Gaussian	90.87 ± 0.16	20.76 ± 1.89	$\underline{90.98 \pm 0.04}$	27.24 ± 5.92	90.96 ± 0.05	59.63 ± 18.64	90.81 ± 0.08	37.76 ± 0.73	$\mathbf{91.05 \pm 0.02}$	43.22 ± 5.50
	Moons	56.47 ± 0.52	17.98 ± 0.97	58.46 ± 0.20	17.70 ± 0.59	$\underline{58.54 \pm 0.10}$	44.39 ± 6.42	58.47 ± 0.17	36.12 ± 1.95	$\mathbf{58.56 \pm 0.26}$	35.88 ± 1.39
Product Recommendation (Amazon)	apparel	66.29 ± 1.66	37.13 ± 12.32	$\mathbf{76.28 \pm 0.61}$	71.23 ± 6.31	69.65 ± 0.73	106.66 ± 19.01	$\underline{69.66 \pm 1.62}$	83.55 ± 21.35	69.31 ± 1.77	75.07 ± 20.58
	bath	68.00 ± 0.69	32.42 ± 5.22	75.82 ± 1.45	46.78 ± 7.114	71.75 ± 0.33	92.89 ± 3.37	$\underline{76.59 \pm 0.82}$	62.74 ± 9.24	$\mathbf{76.78 \pm 0.77}$	63.52 ± 11.36
	bedding	65.36 ± 0.29	40.73 ± 4.24	75.81 ± 0.95	58.77 ± 11.75	66.32 ± 1.00	94.47 ± 12.73	75.99 ± 0.88	67.55 ± 16.24	$\mathbf{75.99 \pm 0.88}$	65.44 ± 14.89
	carseats	20.64 ± 1.15	13.31 ± 5.43	20.30 ± 1.26	12.06 ± 5.87	20.46 ± 1.08	19.31 ± 10.48	$\mathbf{21.25 \pm 0.94}$	39.84 ± 12.28	$\underline{21.23 \pm 0.91}$	34.78 ± 6.55
	diaper	72.23 ± 1.75	52.11 ± 17.47	$\mathbf{82.52 \pm 0.69}$	82.23 ± 21.69	81.10 ± 0.78	173.69 ± 31.34	$\underline{82.12 \pm 1.42}$	99.11 ± 36.21	82.11 ± 1.65	101.38 ± 56.45
	feeding	71.36 ± 1.13	83.91 ± 25.05	$\mathbf{81.75 \pm 1.44}$	116.54 ± 41.32	77.34 ± 0.35	239.76 ± 57.10	$\underline{80.30 \pm 0.72}$	109.27 ± 24.42	$\underline{80.30 \pm 0.72}$	108.00 ± 22.30
	furniture	14.60 ± 1.92	5.64 ± 1.48	15.87 ± 1.75	7.69 ± 3.62	15.78 ± 2.01	6.65 ± 1.06	$\mathbf{21.97 \pm 3.46}$	42.62 ± 9.42	$\mathbf{21.97 \pm 3.46}$	38.52 ± 9.15
	gear	$\underline{64.06 \pm 1.07}$	42.58 ± 3.29	$\mathbf{72.34 \pm 1.36}$	61.06 ± 15.40	59.44 ± 1.19	95.18 ± 13.29	58.47 ± 2.39	42.61 ± 8.56	58.47 ± 2.39	41.35 ± 8.39
	health	63.35 ± 1.14	34.34 ± 4.36	70.99 ± 0.93	60.25 ± 10.77	58.69 ± 0.49	73.97 ± 11.68	$\underline{72.08 \pm 1.22}$	78.65 ± 13.01	$\mathbf{72.46 \pm 0.87}$	82.37 ± 12.84
	media	56.21 ± 1.17	46.36 ± 6.71	55.29 ± 1.78	52.13 ± 15.07	51.45 ± 1.23	55.07 ± 11.51	57.88 ± 0.74	68.61 ± 7.66	$\underline{57.74 \pm 0.74}$	64.60 ± 5.04
	safety	14.51 ± 0.57	4.95 ± 1.04	13.94 ± 1.77	4.43 ± 0.30	13.77 ± 0.45	5.27 ± 0.76	$\mathbf{30.36 \pm 1.46}$	43.57 ± 3.95	$\mathbf{30.36 \pm 1.46}$	40.60 ± 4.20
	toys	63.08 ± 0.68	32.32 ± 4.43	$\mathbf{67.97 \pm 0.98}$	48.00 ± 6.45	63.19 ± 1.42	78.08 ± 18.49	$\underline{67.49 \pm 1.08}$	73.69 ± 26.15	67.10 ± 1.00	52.22 ± 10.02

Table 1. Test Jaccard coefficient and training time for set anomaly detection (AD) and product recommendation tasks, across all five algorithms. iDiffMF$_2$ and iDiffMF$_*$ correspond to our algorithm with Frobenius and nuclear norm rescaling, respectively. **Bold** and underline indicate the best and second best performance results, respectively.

recommendation tasks (see App. F). We closely follow the experimental setup of Ou et al. [1]. We report the mean and standard variation for all experiment results based on 5 runs with different seeds.

4.1. Algorithms

DiffMF [1]: Differentiable mean field variational inference algorithm described in Section 2.3 where, as per Ou et al., the number of iterations is set to $K = 5$ for all datasets.

EquiVSet$_{\text{ind}}$ [1]: Equivariant variational inference algorithm proposed by Ou et al. [1]. It is a variation of the DiffMF algorithm where the parameter $\boldsymbol{\psi}$ is predicted by an additional recognition network as a function of the features. As per Ou et al., we set $K = 1$ for all datasets.

EquiVSet$_{\text{copula}}$ [1]: A correlation-aware version of the EquiVSet$_{\text{ind}}$ algorithm where the relations among the input elements are modeled by a Gaussian copula. As per Ou et al., we set $K = 1$ for all datasets.

iDiffMF: Our proposed implicit alternative to the DiffMF algorithm where we solve the fixed-point condition in Eq. (8) with a very low tolerance threshold, instead of running the fixed-point iterations in Eq. (9) for only a fixed number of times. Although DNNs are bounded, the exact computation of their Lipschitz constant is NP-hard [31]. In our implementation, we satisfy the condition in Asm. 3.1 by normalizing the gradient of the multilinear relaxation by its norm and scaling it with the size of the ground set, i.e., we multiply $\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})$ with $2/(|V| \|\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})\|)$. For the L_2 ($\|\cdot\|_2$) and nuclear ($\|\cdot\|_*$) norms, we observe that this heuristic works well in practice and propose two variations as iDiffMF$_2$ and iDiffMF$_*$, respectively.

We train all models with an Adam optimizer [35] with a learning rate $\eta = 10^{-4}$ and a batch size of 128.

Metrics: We measure the performance of the algorithms by their mean Jaccard Coefficient (JC) score. The mean JC over all optimal oracle subset pairs is $\frac{1}{N} \sum_{i=1}^{N} \frac{|S_i^* \cap \hat{S}_i^*|}{|S_i^* \cup \hat{S}_i^*|}$, where $\hat{S}_i^*$ is the prediction of the model.

4.2. Results

We report the performance of our proposed iDiffMF methods against the baselines on Tab. 1. For the majority of the cases, iDiffMF$_*$ achieves either the best or the second-best JC score. While the next best competitor, EquiVSet$_{\text{copula}}$, performs the best on some of the datasets its performance is sub-optimal on the remaining datasets. For the Amazon furniture and safety datasets, iDiffMF variants give significantly better results than EquiVSet$_{\text{copula}}$, even though EquiVSet$_{\text{copula}}$ is faster. This is probably because EquiVSet$_{\text{copula}}$ converges to a local optimum and finishes training earlier. It is also important to highlight that we evaluate iDiffMF using JAX+Flax [36]–[39] while we use PyTorch to evaluate the baselines. Therefore, the differences in running time can also be explained with the framework differences. Finally, even though iDiffMF executes fixed-point iterations until convergence, as opposed to $K = 1$ or 5 in remaining methods [1], the average running times are comparable across datasets.

5. Conclusion

We improve upon an existing learning set functions with an optimal subset oracle setting by characterizing the convergence condition of the fixed point iterations resulting during MLE approximation and by using implicit differentiation over automatic differentiation. Our results perform better than or comparable to the baselines for the majority of the cases without the need of an additional recognition network.

6. Acknowledgements

We gratefully acknowledge support from the NSF grant CCF-1750539.

References

[1] Z. Ou, T. Xu, Q. Su, Y. Li, P. Zhao, and Y. Bian, "Learning neural set functions under the optimal subset oracle," *Advances in Neural Information Processing Systems*, vol. 35, pp. 35 021–35 034, 2022.

[2] H. Bonab, M. Aliannejadi, A. Vardasbi, E. Kanoulas, and J. Allan, "Cross-market product recommendation," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 110–119.

[3] J. B. Schafer, J. Konstan, and J. Riedl, "Recommender systems in e-commerce," in *Proceedings of the 1st ACM conference on Electronic commerce*, 1999, pp. 158–166.

[4] X. Ning, M. Walters, and G. Karypis, "Improved machine learning models for predicting selective compounds," in *Proceedings of the 2nd ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, 2011, pp. 106–115.

[5] Y. Saito, T. Nakamura, H. Hachiya, and K. Fukumizu, "Exchangeable deep neural networks for set-to-set matching and learning," in *European Conference on Computer Vision*, Springer, 2020, pp. 626–646.

[6] Q. Feng, Y. Zhou, and R. Lan, "Pairwise linear regression classification for image set retrieval," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4865–4872.

[7] H. Zhao, L. Jiang, C.-W. Fu, and J. Jia, "Pointweb: Enhancing local neighborhood features for point cloud processing," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[8] A. Gionis, D. Gunopulos, and N. Koudas, "Efficient and tumble similar set retrieval," in *Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, 2001, pp. 247–258.

[9] M. Mašková, M. Zorek, T. Pevnỳ, and V. Šmídl, "Deep anomaly detection on set data: Survey and comparison," *Pattern Recognition*, p. 110 381, 2024.

[10] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep sets," *Advances in neural information processing systems*, vol. 30, 2017.

[11] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *International conference on machine learning*, PMLR, 2019, pp. 3744–3753.

[12] S. Tschiatschek, A. Sahin, and A. Krause, "Differentiable submodular maximization," in *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, 2018, pp. 2731–2738.

[13] S. Bai, J. Z. Kolter, and V. Koltun, "Deep Equilibrium Models," *Advances in neural information processing systems*, vol. 32, 2019.

[14] Z. Kolter, D. Duvenaud, and M. Johnson, "Deep Implicit Layers - Neural ODEs, Deep Equilibirum Models, and Beyond," NeurIPS, 2020. [Online]. Available: https://implicit-layers-tutorial.org.

[15] Z. Huang, S. Bai, and J. Z. Kolter, "$(\text{Implicit})^2$: Implicit layers for implicit representations," *NeurIPS*, 2021.

[16] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[17] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT press, 2012.

[18] A. Mnih and G. Hinton, "Learning nonlinear constraints with contrastive backpropagation," in *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, IEEE, vol. 2, 2005, pp. 1302–1307.

[19] G. Hinton, S. Osindero, M. Welling, and Y.-W. Teh, "Unsupervised discovery of nonlinear structure using contrastive backpropagation," *Cognitive science*, vol. 30, no. 4, pp. 725–731, 2006.

[20] Y. LeCun, S. Chopra, and R. Hadsell, "A tutorial on energy-based learning," 2006.

[21] A. Kirsch, S. Farquhar, P. Atighehchian, A. Jesson, F. Branchaud-Charron, and Y. Gal, "Stochastic batch acquisition: A simple baseline for deep active learning," *Transactions on Machine Learning Research*, 2023.

[22] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[23] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[24] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrák, "Maximizing a monotone submodular function subject to a matroid constraint," *SIAM Journal on Computing*, vol. 40, no. 6, pp. 1740–1766, 2011.

[25] A. Sklar, "Random variables, joint distribution functions, and copulas," *Kybernetika*, vol. 9, no. 6, pp. 449–460, 1973.

[26] R. B. Nelsen, *An Introduction to Copulas*, second. New York, NY, USA: Springer, 2006.

[27] M. Blondel, Q. Berthet, M. Cuturi, R. Frostig, S. Hoyer, F. Llinares-López, F. Pedregosa, and J.-P. Vert, "Efficient and modular implicit differentiation," *Advances in neural information processing systems*, vol. 35, pp. 5230–5242, 2022.

[28] B. Davies, *Exploring chaos: Theory and experiment*. CRC Press, 2018.

[29] A. Rechnitzer, *Fixed points - summary [lecture notes]. dynamical systems and chaos — 620341*. 2003. [Online]. Available: https://personal.math.ubc.ca/~andrewr/620341/pdfs/fp_sum.pdf.

[30] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[31] A. Virmaux and K. Scaman, "Lipschitz regularity of deep neural networks: Analysis and efficient estimation," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[32] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree, "Regularisation of neural networks by enforcing lipschitz continuity," *Machine Learning*, vol. 110, pp. 393–416, 2021.

[33] S. Banach, "Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales," *Fundamenta mathematicae*, vol. 3, no. 1, pp. 133–181, 1922.

[34] S. G. Krantz and H. R. Parks, *The implicit function theorem: history, theory, and applications*. Springer Science & Business Media, 2002.

[35] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *ICLR*, 2015.

[36] J. Bradbury, R. Frostig, P. Hawkins, M. J. Johnson, C. Leary, D. Maclaurin, G. Necula, A. Paszke, J. VanderPlas, S. Wanderman-Milne, and Q. Zhang, *JAX: Composable transformations of Python+NumPy programs*, version 0.3.13, 2018. [Online]. Available: http://github.com/google/jax.

[37] R. Frostig, M. J. Johnson, and C. Leary, "Compiling machine learning programs via high-level tracing," *Systems for Machine Learning*, vol. 4, no. 9, 2018.

[38] J. Heek, A. Levskaya, A. Oliver, M. Ritter, B. Rondepierre, A. Steiner, and M. van Zee, *Flax: A neural network library and ecosystem for JAX*, version 0.8.5, 2023. [Online]. Available: http://github.com/google/flax.

[39] P. Lippe, *UvA Deep Learning Tutorials*, https://uvadlc-notebooks.readthedocs.io/en/latest/, 2024.

[40] M. Kimura, R. Shimizu, Y. Hirakawa, R. Goto, and Y. Saito, "On permutation-invariant neural networks," *arXiv preprint arXiv:2403.17410*, 2024.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[42] L. Zhang, V. Tozzo, J. Higgins, and R. Ranganath, "Set norm and equivariant skip connections: Putting the deep in deep sets," in *International Conference on Machine Learning*, PMLR, 2022, pp. 26 559–26 574.

[43] G. Giannone and O. Winther, "Scha-vae: Hierarchical context aggregation for few-shot generation," in *International Conference on Machine Learning*, PMLR, 2022, pp. 7550–7569.

[44] M. Michalkiewicz, J. K. Pontes, D. Jack, M. Baktashmotlagh, and A. Eriksson, "Implicit surface representations as layers in neural networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[45] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in neural information processing systems*, vol. 33, pp. 7462–7473, 2020.

[46] Y. Strümpler, J. Postels, R. Yang, L. V. Gool, and F. Tombari, "Implicit neural representations for image compression," in *European Conference on Computer Vision*, Springer, 2022, pp. 74–91.

[47] S. W. Fung, H. Heaton, Q. Li, D. McKenzie, S. Osher, and W. Yin, "Jfb: Jacobian-free backpropagation for implicit networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, 2022, pp. 6648–6656.

[48] M. Retchin, B. Amos, S. Brunton, and S. Song, "Koopman constrained policy optimization: A koopman operator theoretic method for differentiable optimal control in robotics," in *ICML 2023 Workshop on Differentiable Almost Everything: Differentiable Relaxations, Algorithms, Operators, and Simulators*, 2023.

[49] E. Dupont, A. Golinski, M. Alizadeh, Y. W. Teh, and A. Doucet, "Coin: Compression with implicit neural representations," in *Neural Compression: From Information Theory to Applications–Workshop@ ICLR 2021*, 2021.

[50] D. Grattarola and P. Vandergheynst, "Generalised implicit neural representations," *Advances in Neural Information Processing Systems*, vol. 35, pp. 30 446–30 458, 2022.

[51] E. Winston and J. Z. Kolter, "Monotone operator equilibrium networks," *Advances in neural information processing systems*, vol. 33, pp. 10 718–10 728, 2020.

[52] R. T. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud, "Neural ordinary differential equations," *Advances in neural information processing systems*, vol. 31, 2018.

[53] X. Li, T.-K. L. Wong, R. T. Chen, and D. Duvenaud, "Scalable gradients for stochastic differential equations," in *International Conference on Artificial Intelligence and Statistics*, PMLR, 2020, pp. 3870–3882.

[54] B. Amos and J. Z. Kolter, "Optnet: Differentiable optimization as a layer in neural networks," in *International Conference on Machine Learning*, PMLR, 2017, pp. 136–145.

[55] T. Apostol, *Mathematical Analysis* (Addison-Wesley series in mathematics). Addison-Wesley, 1974, ISBN: 9780201002881. [Online]. Available: https://books.google.com/books?id=Le5QAAAAMAAJ.

[56] W. Rudin, *Principles of Mathematical Analysis* (International series in pure and applied mathematics). McGraw-Hill, 1976, ISBN: 9780070856134. [Online]. Available: https://books.google.com/books?id=kwqzPAAACAAJ.

[57] J. V. Burke, "Nonlinear optimization," *Lecture Notes, Math*, vol. 408, p. 80, 2014.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[59] Z. Liu, P. Luo, X. Wang, and X. Tang, "Deep learning face attributes in the wild," in *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

[60] J. A. Gillenwater, A. Kulesza, E. Fox, and B. Taskar, "Expectation-maximization for learning determinantal point processes," *Advances in Neural Information Processing Systems*, vol. 27, 2014.

[61] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *North American Chapter of the Association for Computational Linguistics*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:52967399.

Supplementary Material

A. Related Work

Learning Set Functions. There is a line of work that tries to extend the capabilities of neural networks for functions on discrete domains, i.e., *set functions*. Set functions are characterized by their *permutation invariance*. We refer the reader to [40] for a detailed overview of permutation-invariant architectures. One such seminal architecture, Deep Sets [10], characterizes a general family of functions for permutation invariant objectives. Following the effectiveness of transformers [41], set transformers [11] process input sets with an attention-based neural network architecture. Zhang et al. [42] improves over both of these methods by introducing a novel set normalization layer. Ou et al. [1] study the problem of learning set functions under the supervision of optimal subset oracles with a mean-field variational approximation-based method. We build upon their approach by using the implicit differentiation toolbox in the context of set function learning.

There are existing models that use variational auto-encoders (VAE) [22] for set outputs: SetVAE adopts the aforementioned set transformers [11] into VAEs for point cloud generation tasks and SCHA-VAE [43] applies the same idea for few-shot image generation. These are both generative models and not applicable to subset selection tasks.

Implicit Differentiation. Implicit representations [15], [44]–[46], layers [13]–[15], and differentiation [27], [47], [48] have gained a lot of traction in the recent years. Implicit representations [49], [50] refer to the case when high-frequency, often discretely measured data is modeled with a continuous function, whereas implicit layers is a term that characterizes model architectures. Huang et al. [15] study the complementary benefits of combining implicit representations with implicit layers. Deep equilibrium models (DEQ) [13], [51], an example of implicit layers, directly compute the fixed-point resulting from stacking up weight-tied forward sequence models and also directly differentiate through the stacked fixed-point equations via implicit differentiation. A recent tutorial [14] highlights the advantages of implicit differentiation by including other examples from neural ordinary differential equations [52], [53] and differentiable optimization problems [54]. Blondel et al. [27] provide a JAX-based, modular automatic implicit differentiation tool in order to aid practitioners with implementation. For feed-forward networks, Fung et al. [47] eliminate the Jacobian that results from implicit differentiation. It is important to note that they also use Banach fixed-point theorem [33] to prove the convergence of the fixed-point iterations. However, they directly assume the mapping in focus is a contraction where we specify the condition under which the mapping in focus is a contraction (see Asm. 3.1).

B. Derivations for Section 2

B.1. Derivation of the Fixed-Point

Rewriting the ELBO by plugging in the definition of entropy,

$$\tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta}) + \mathbb{H}(q(S, \boldsymbol{\psi})) = \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta}) - \sum_{i=1}^{|V|} \left[\psi_i \log \psi_i + (1 - \psi_i) \log(1 - \psi_i)\right]. \tag{13}$$

Taking the partial derivative of this expression with respect to the i^{th} coordinate and setting it to zero, yields

$$\begin{aligned}
\frac{\partial \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial \psi_i} - \log \frac{\psi_i}{1 - \psi_i} &= 0, \\
\exp \frac{\partial \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial \psi_i} &= \frac{\psi_i}{1 - \psi_i}, \\
\exp \frac{\partial \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial \psi_i} - \psi_i \exp \left(\frac{\partial \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial \psi_i} \right) &= \psi_i, \\
\exp \left(\frac{\partial \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial \psi_i} \right) &= \psi_i \left(1 + \exp \left(\frac{\partial \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial \psi_i} \right) \right), \\
\psi_i = \frac{\exp \left(\frac{\partial \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial \psi_i} \right)}{1 + \exp \left(\frac{\partial \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial \psi_i} \right)} &= \frac{1}{1 + \exp \left(-\frac{\partial \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial \psi_i} \right)} = \sigma \left(\frac{\partial \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial \psi_i} \right),
\end{aligned}$$

where $\sigma(x) = (1 + \exp(-x))^{-1}$ is the sigmoid function.

B.2. Gradient Computation via Sampling

Lemma B.1. *Given a set function $F : 2^V \to \mathbb{R}$ and a vector of probabilities $\boldsymbol{\psi} \in [0,1]^{|V|}$ where $\psi_i = \mathbb{P}[i \in S]$, the gradient of the multilinear relaxation of the set function $F(S)$ is*

$$\frac{\partial \tilde{F}(\boldsymbol{\psi})}{\partial \psi_i} = \mathbb{E}_{S \sim \boldsymbol{\psi} | \psi_i \leftarrow 0}[F(S+i) - F(S)].$$

Proof.

$$\begin{aligned}
\frac{\partial \tilde{F}(\boldsymbol{\psi})}{\partial \psi_i} &= \sum_{S \subseteq V} F(S) \prod_{i \in S} \psi_i \prod_{i \notin S} (1 - \psi_i), \\
&= \mathbb{E}_{S \sim \boldsymbol{\psi} | \psi_i \leftarrow 1}[F(S)] - \mathbb{E}_{S \sim \boldsymbol{\psi} | \psi_i \leftarrow 0}[F(S)], \\
&= \sum_{S \subseteq V, i \in S} F(S) \prod_{j \in S \setminus \{i\}} \psi_j \prod_{j \notin S} (1 - \psi_j) - \sum_{S \subseteq V \setminus \{i\}} F(S) \prod_{j \in S} \psi_j \prod_{j \notin S, j \neq i} (1 - \psi_j), \\
&= \sum_{S \subseteq V \setminus \{i\}} [F(S+i) - F(S)] \prod_{j \in S} \psi_j \prod_{j \notin S} (1 - \psi_j), \\
&= \mathbb{E}_{S \sim \boldsymbol{\psi} | \psi_i \leftarrow 0}[F(S+i) - F(S)].
\end{aligned}$$

□

Corollary B.2. *Knowing Lemma B.1, the gradient of the multilinear relaxation $\tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})$, is defined as follows*

$$\frac{\partial \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial \psi_i} = \tilde{F}([\boldsymbol{\psi}]_{+i}, \boldsymbol{\theta}) - \tilde{F}([\boldsymbol{\psi}]_{-i}, \boldsymbol{\theta}). \tag{14}$$

Eq. (14) can be computed by producing random samples of S.

C. Technical Preliminaries

Theorem C.1. *(Multivariate Mean Value Theorem [55]–[57]) If $\boldsymbol{f} : \mathbb{R}^n \to \mathbb{R}^m$ is continuously differentiable, then for every $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, there exists a $\boldsymbol{z} \in [\boldsymbol{x}, \boldsymbol{y}]$, such that*

$$\|\boldsymbol{f}(\boldsymbol{x}) - \boldsymbol{f}(\boldsymbol{y})\|_2 \leq \sup_{\boldsymbol{z} \in [\boldsymbol{x}, \boldsymbol{y}]} \|\partial \boldsymbol{f}(\boldsymbol{z})\|_F \|\boldsymbol{x} - \boldsymbol{y}\|_2,$$

where $\|\cdot\|_2$ is the L_2 norm and $\|\cdot\|_F$ is the Frobenius norm.

Definition C.2. A mapping $T : X \to X$ is called a *contraction* on X if there exists a constant $\epsilon \in [0, 1)$ such that for all $x, y \in X$,

$$\|T(x) - T(y)\|_2 \leq \epsilon \|x - y\|_2.$$

Theorem C.3. *(Banach's Fixed Point Theorem [33]) Let $T : X \to X$ be a contraction on X. Then T has a unique fixed point $x^* \in X$ where $T(x^*) = x^*$.*

Theorem C.4. *(Implicit Function Theorem [27], [34]) Given a continuously differentiable function $G : \mathbb{R}^n \times \mathbb{R}^d \to \mathbb{R}^n$, an implicitly defined function $\boldsymbol{x}^* : \mathbb{R}^d \to \mathbb{R}^n$ of $\boldsymbol{\theta} \in \mathbb{R}^d$, and an optimal solution $\boldsymbol{x}^*(\boldsymbol{\theta})$; let*

$$G(\boldsymbol{x}^*(\boldsymbol{\theta}), \boldsymbol{\theta}) = 0. \tag{15}$$

For $(\boldsymbol{x}_0, \boldsymbol{\theta}_0)$ satisfying $G(\boldsymbol{x}_0, \boldsymbol{\theta}_0) = 0$, if the Jacobian $\partial_{\boldsymbol{x}} G$ evaluated at $(\boldsymbol{x}_0, \boldsymbol{\theta}_0)$ is a square invertible matrix, then there exists a function $\boldsymbol{x}^*(\cdot)$ defined on a neighborhood of $\boldsymbol{\theta}_0$ such that $\boldsymbol{x}^*(\boldsymbol{\theta}_0) = x_0$. Furthermore, for all $\boldsymbol{\theta}$ in this neighborhood, we have that $G(x^*(\boldsymbol{\theta}), \boldsymbol{\theta}) = 0$ and $\partial x^*(\boldsymbol{\theta})$ exists. According to the chain rule, the Jacobian $\partial x^*(\boldsymbol{\theta})$ satisfies

$$\partial_{\boldsymbol{x}} G(\boldsymbol{x}^*(\boldsymbol{\theta}), \boldsymbol{\theta}) \partial \boldsymbol{x}^*(\boldsymbol{\theta}) + \partial_{\boldsymbol{\theta}} G(\boldsymbol{x}^*(\boldsymbol{\theta}), \boldsymbol{\theta}) = 0. \tag{16}$$

Therefore, computing $\partial \boldsymbol{x}^*(\boldsymbol{\theta})$ becomes the equivalent of solving the following linear system of equations

$$\underbrace{-\partial_{\boldsymbol{x}} G(\boldsymbol{x}^*(\boldsymbol{\theta}), \boldsymbol{\theta})}_{A \in \mathbb{R}^{n \times n}} \underbrace{\partial \boldsymbol{x}^*(\boldsymbol{\theta})}_{J \in \mathbb{R}^{n \times d}} = \underbrace{\partial_{\boldsymbol{\theta}} G(\boldsymbol{x}^*(\boldsymbol{\theta}), \boldsymbol{\theta})}_{B \in \mathbb{R}^{n \times d}}. \tag{17}$$

D. Proof of Theorem 3.2

Before stating our proof we need to state the following corollary:

Corollary D.1. *Knowing Corollary B.2, we can write the* Hessian *of the multilinear relaxation as*

$$\begin{aligned}\frac{\partial^2 \tilde{F}(\boldsymbol{\psi}, \theta)}{\partial \psi_i \partial \psi_j} &= \left(\tilde{F}([\boldsymbol{\psi}]_{+i,+j}, \boldsymbol{\theta})] - \tilde{F}([\boldsymbol{\psi}]_{-i,+j}, \boldsymbol{\theta})]\right) - \left(\tilde{F}([\boldsymbol{\psi}]_{+i,-j}, \boldsymbol{\theta})] - \tilde{F}([\boldsymbol{\psi}]_{-i,-j}, \boldsymbol{\theta})]\right), \\ &= \tilde{F}([\boldsymbol{\psi}]_{+i,+j}, \boldsymbol{\theta})] - \tilde{F}([\boldsymbol{\psi}]_{-i,+j}, \boldsymbol{\theta}) - \tilde{F}([\boldsymbol{\psi}]_{+i,-j}, \boldsymbol{\theta}) + \tilde{F}([\boldsymbol{\psi}]_{-i,-j}, \boldsymbol{\theta}),\end{aligned} \tag{18}$$

if $i \neq j$*, otherwise* $\frac{\partial^2 \tilde{F}(\boldsymbol{\psi}, \theta)}{\partial \boldsymbol{\psi}_i \partial \boldsymbol{\psi}_j} = 0$.

We proof the following lemma using this corollary:

Lemma D.2. *Elements of the Hessian given in Eq. (18) are bounded with* $4 \sup_{\boldsymbol{\psi} \in [\mathbf{0},\mathbf{1}]} \left|\tilde{F}\left(\boldsymbol{\psi}, \boldsymbol{\theta}\right)\right|$*, i.e.,* $\sup_{\psi_i, \psi_j \in [0,1]} \left|\frac{\partial^2 \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial_{\psi_i} \partial_{\psi_j}}\right| \leq 4 \sup_{\boldsymbol{\psi} \in [\mathbf{0},\mathbf{1}]} \left|\tilde{F}\left(\boldsymbol{\psi}, \boldsymbol{\theta}\right)\right|$.

Proof. By Corollary D.1, we have

$$\left|\frac{\partial^2 \tilde{F}(\boldsymbol{\psi}, \theta)}{\partial \psi_i \partial \psi_j}\right| = \left|\left(\tilde{F}([\boldsymbol{\psi}]_{+i,+j}, \boldsymbol{\theta})] - \tilde{F}([\boldsymbol{\psi}]_{-i,+j}, \boldsymbol{\theta})]\right) - \left(\tilde{F}([\boldsymbol{\psi}]_{+i,-j}, \boldsymbol{\theta})] - \tilde{F}([\boldsymbol{\psi}]_{-i,-j}, \boldsymbol{\theta})]\right)\right|.$$

Using the triangular inequality twice, we get

$$\begin{aligned}\left|\frac{\partial^2 \tilde{F}(\boldsymbol{\psi}, \theta)}{\partial \psi_i \partial \psi_j}\right| &\leq \left|\tilde{F}([\boldsymbol{\psi}]_{+i,+j}, \boldsymbol{\theta})] - \tilde{F}([\boldsymbol{\psi}]_{-i,+j}, \boldsymbol{\theta})]\right| + \left|\tilde{F}([\boldsymbol{\psi}]_{+i,-j}, \boldsymbol{\theta})] - \tilde{F}([\boldsymbol{\psi}]_{-i,-j}, \boldsymbol{\theta})]\right|, \\ &\leq \left|\tilde{F}([\boldsymbol{\psi}]_{+i,+j}, \boldsymbol{\theta})]\right| + \left|\tilde{F}([\boldsymbol{\psi}]_{-i,+j}, \boldsymbol{\theta})]\right| + \left|\tilde{F}([\boldsymbol{\psi}]_{+i,-j}, \boldsymbol{\theta})]\right| + \left|\tilde{F}([\boldsymbol{\psi}]_{-i,-j}, \boldsymbol{\theta})]\right|, \\ &\leq 4 \sup_{\boldsymbol{\psi} \in [\mathbf{0},\mathbf{1}]} \left|\tilde{F}\left(\boldsymbol{\psi}, \boldsymbol{\theta}\right)\right|.\end{aligned}$$

□

Equipped with this lemma, we are ready to proof Thm. 3.2:

Proof. For simplicity, define a mapping $T_{\boldsymbol{\theta}} : [0,1]^{|V|} \to [0,1]^{|V|}$ where $T_{\boldsymbol{\theta}}(\boldsymbol{\psi}) = \boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}} \tilde{F}\left(\boldsymbol{\psi}, \boldsymbol{\theta}\right))$. Given $\tilde{F}$ is a polynomial w.r.t. $\boldsymbol{\psi}$ and the sigmoid is a smooth function, $T_{\boldsymbol{\theta}}$ is continuously differentiable w.r.t. $\boldsymbol{\psi}$ in $[0,1]^{|V|}$. By the multivariate equivalent of the mean-value theorem (see Theorem C.1 in Appendix C), for every $\boldsymbol{x}, \boldsymbol{y} \in |0,1|^{|V|}$, there exists a $\boldsymbol{\psi} \in |0,1|^{|V|}$, such that

$$\begin{aligned}\|T_{\boldsymbol{\theta}}(\boldsymbol{x}) - T_{\boldsymbol{\theta}}(\boldsymbol{y})\|_2 &\leq \sup_{\boldsymbol{\psi} \in [\mathbf{0},\mathbf{1}]} \|\partial T_{\boldsymbol{\theta}}(\boldsymbol{\psi})\|_F \|\boldsymbol{x} - \boldsymbol{y}\|_2, \\ \|\boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}} \tilde{F}\left(\boldsymbol{x}, \boldsymbol{\theta}\right)) - \boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}} \tilde{F}\left(\boldsymbol{y}, \boldsymbol{\theta}\right))\|_2 &\leq \sup_{\boldsymbol{\psi} \in [\mathbf{0},\mathbf{1}]} \|\partial_{\boldsymbol{\psi}} \boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}} \tilde{F}\left(\boldsymbol{\psi}, \boldsymbol{\theta}\right))\|_F \|\boldsymbol{x} - \boldsymbol{y}\|_2.\end{aligned} \tag{19}$$

From Eq. (23), we know that $\partial_{\boldsymbol{\psi}} \boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}} \tilde{F}\left(\boldsymbol{\psi}, \boldsymbol{\theta}\right)) = \left[\sigma'\left(\frac{\partial \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial_{\psi_i}}\right) \frac{\partial^2 \tilde{F}(\boldsymbol{\psi}, \boldsymbol{\theta})}{\partial_{\psi_i} \partial_{\psi_j}}\right]_{1 \leq i,j \leq |V|}$ where $\sigma'(x) = (1 + \exp(-x))^{-2} \cdot \exp(-x)$. Then,

$$\begin{aligned}
\sup_{\boldsymbol{\psi}\in[\mathbf{0},\mathbf{1}]} \|\partial_{\boldsymbol{\psi}}\boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))\|_F &= \sup_{\boldsymbol{\psi}\in[\mathbf{0},\mathbf{1}]} \sqrt{\sum_i^{|V|}\sum_j^{|V|}\left|\sigma'\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial_{\psi_i}}\right)\frac{\partial^2\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial_{\psi_i}\partial_{\psi_j}}\right|^2}, \\
&\leq \sqrt{\sum_i^{|V|}\sum_j^{|V|}\left(\sup_{\psi_i,\psi_j\in[0,1]}\left|\sigma'\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial_{\psi_i}}\right)\frac{\partial^2\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial_{\psi_i}\partial_{\psi_j}}\right|\right)^2}, \\
&= |V|\sup_{\psi_i,\psi_j\in[0,1]}\left|\sigma'\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial_{\psi_i}}\right)\frac{\partial^2\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial_{\psi_i}\partial_{\psi_j}}\right|, \\
&\leq |V|\sup_{\psi_i\in[0,1]}\left|\sigma'\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial_{\psi_i}}\right)\right|\sup_{\psi_i,\psi_j\in[0,1]}\left|\frac{\partial^2\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial_{\psi_i}\partial_{\psi_j}}\right|.
\end{aligned}$$

Since $\arg\max_{x\in\mathbb{R}}\sigma'(x)=0$, we know that $\sup_{\psi_i\in[0,1]}\left|\sigma'\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial_{\psi_i}}\right)\right|\leq\frac{1}{4}$. Moreover, Lemma D.2 gives us a bound for the elements of the Hessian matrix where $\sup_{\psi_i,\psi_j\in[0,1]}\left|\frac{\partial^2\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial_{\psi_i}\partial_{\psi_j}}\right|\leq 4\sup_{\boldsymbol{\psi}\in[\mathbf{0},\mathbf{1}]}\left|\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})\right|$. As a result,

$$\sup_{\boldsymbol{\psi}\in[\mathbf{0},\mathbf{1}]} \|\partial_{\boldsymbol{\psi}}\boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))\|_F \leq |V|\sup_{\boldsymbol{\psi}\in[\mathbf{0},\mathbf{1}]}\left|\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})\right|. \tag{20}$$

According to Assumption 3.1, we know that $\sup_{\boldsymbol{\psi}\in[\mathbf{0},\mathbf{1}]}\left|\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})\right|<\frac{1}{|V|}$. Therefore,

$$\sup_{\boldsymbol{\psi}\in[\mathbf{0},\mathbf{1}]} \|\partial_{\boldsymbol{\psi}}\boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))\|_F < 1. \tag{21}$$

Plugging this in Eq. (19) above, we get

$$\|\boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{x},\boldsymbol{\theta}))-\boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{y},\boldsymbol{\theta}))\|_2 < \|\boldsymbol{x}-\boldsymbol{y}\|_2. \tag{22}$$

This means Eq. (8) is a contraction on $[0,1]^{|V|}$ (see Definition C.2 in Appendix C). Thus, according to Banach fixed-point theorem (see Theorem C.3 in Appendix C) the iterations given in Eq. (9) are bound to converge to a unique solution. □

E. Proof of Theorem 3.3

Proof. For $n=|V|$,

$$\begin{aligned}
A &= I-\partial_{\boldsymbol{\psi}}\boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})) = I-\begin{bmatrix}\frac{\partial\boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))}{\partial\psi_1} & \dots & \frac{\partial\boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))}{\partial\psi_n}\end{bmatrix}, \\
&= I-\begin{bmatrix}\frac{\partial\sigma_1(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))}{\partial\psi_1} & \dots & \frac{\partial\sigma_1(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))}{\partial\psi_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial\sigma_n(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))}{\partial\psi_1} & \dots & \frac{\partial\sigma_n(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))}{\partial\psi_n}\end{bmatrix} = I-\begin{bmatrix}\frac{\partial\sigma\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_1}\right)}{\partial\psi_1} & \dots & \frac{\partial\sigma\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_1}\right)}{\partial\psi_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial\sigma\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_n}\right)}{\partial\psi_1} & \dots & \frac{\partial\sigma\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_n}\right)}{\partial\psi_n}\end{bmatrix}, \\
&= I-\begin{bmatrix}\sigma'\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_1}\right)\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial^2\psi_1} & \dots & \sigma'\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_1}\right)\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_1\partial\psi_n} \\ \vdots & \ddots & \vdots \\ \sigma'\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_n}\right)\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_n\partial\psi_1} & \dots & \sigma'\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_n}\right)\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial^2\psi_n}\end{bmatrix}, \\
&= I-\begin{bmatrix}\sigma'\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_1}\right) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma'\left(\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_n}\right)\end{bmatrix}\begin{bmatrix}\frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial^2\psi_1} & \dots & \frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_1\partial\psi_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial\psi_n\partial\psi_1} & \dots & \frac{\partial\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial^2\psi_n}\end{bmatrix}, \\
&= I-\Sigma'(\nabla_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))\cdot\nabla^2_{\boldsymbol{\psi}}\tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}),
\end{aligned} \tag{23}$$

where the function $\Sigma' : \mathbb{R}^n \to \mathbb{R}^{n\times n}$ defined as $\Sigma'(\mathbf{x}) = \begin{bmatrix} \sigma'(x_1) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma'(x_n) \end{bmatrix}$ and $\sigma' : \mathbb{R} \to \mathbb{R}$ is $\sigma'(x) = (1+\exp(-x))^{-2} \cdot \exp(-x)$.

Similarly,

$$
\begin{aligned}
B = \partial_\theta \boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})) &= \begin{bmatrix} \frac{\partial \boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))}{\partial \theta_1} & \dots & \frac{\partial \boldsymbol{\sigma}(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))}{\partial \theta_d} \end{bmatrix}, \\
&= \begin{bmatrix} \frac{\partial \sigma_1(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))}{\partial \theta_1} & \dots & \frac{\partial \sigma_1(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))}{\partial \theta_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial \sigma_n(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))}{\partial \theta_1} & \dots & \frac{\partial \sigma_n(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}))}{\partial \theta_d} \end{bmatrix} = \begin{bmatrix} \frac{\partial \sigma\left(\frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_1}\right)}{\partial \theta_1} & \dots & \frac{\partial \sigma\left(\frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_1}\right)}{\partial \theta_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial \sigma\left(\frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_n}\right)}{\partial \theta_1} & \dots & \frac{\partial \sigma\left(\frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_n}\right)}{\partial \theta_n} \end{bmatrix}, \\
&= \begin{bmatrix} \sigma'\left(\frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_1}\right) \frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_1 \partial \theta_1} & \dots & \sigma'\left(\frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_1}\right) \frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_1 \partial \theta_d} \\ \vdots & \ddots & \vdots \\ \sigma'\left(\frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_n}\right) \frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_n \partial \theta_1} & \dots & \sigma'\left(\frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_n}\right) \frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_n \partial \theta_d} \end{bmatrix}, \\
&= \begin{bmatrix} \sigma'\left(\frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_1}\right) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma'\left(\frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_n}\right) \end{bmatrix} \begin{bmatrix} \frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_1 \partial \theta_1} & \dots & \frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_1 \partial \theta_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_n \partial \theta_d} & \dots & \frac{\partial \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})}{\partial \psi_n \partial \theta_d} \end{bmatrix}, \\
&= \Sigma'(\nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta})) \cdot \partial_{\boldsymbol{\theta}} \nabla_{\boldsymbol{\psi}} \tilde{F}(\boldsymbol{\psi},\boldsymbol{\theta}).
\end{aligned}
\tag{24}
$$

□

F. Dataset Details

	Dataset	$\lvert\Omega\rvert$	$\lvert\mathcal{D}\rvert$	$\lvert V\rvert$	avg($\lvert S^*\rvert$)	min($\lvert S^*\rvert$)	max($\lvert S^*\rvert$)
	CelebA	202,599	10000	8	2.5	2	3
	Gaussian	100	1000	100	10	10	10
	Moons	100	1000	100	100	10	10
Amazon	apparel	100	4,675	30	4.52	3	19
Amazon	bath	100	3,195	30	3.80	3	11
Amazon	bedding	100	4,524	30	3.87	3	12
Amazon	carseats	34	483	30	3.26	3	6
Amazon	diaper	100	6,108	30	4.14	3	15
Amazon	feeding	100	8,202	30	4.62	3	2
Amazon	furniture	32	280	30	3.18	3	6
Amazon	gear	100	4,277	30	3.8	3	10
Amazon	health	62	2,995	30	3.69	3	9
Amazon	media	58	1,485	30	4.52	3	19
Amazon	safety	36	267	30	3.16	3	5
Amazon	toys	62	2,421	30	4.09	3	14

Table 2. Dataset configurations.

Moons and Gaussian. There are two classes in the synthetic datasets, whose labels are Bernoulli sampled with $p = 0.5$. Based on this label, the optimal subset and ground set pairs are constructed as follows: 1) sampling 10 points within the class as S^*; and 2) sampling 90 points from the other class as $V \backslash S^*$. This process is repeated until $|\mathcal{D}_{\text{training}}| = |\mathcal{D}_{\text{validation}}| = |\mathcal{D}_{\text{test}}| = 1000$.Following the experimental procedure of Ou et al. [1], we use the SCIKIT-LEARN package [58] to generate the Moons dataset, consisting of two interleaving moons with some noise with variance $\sigma^2 = 0.1$. For the Gaussian dataset, we sample data from a mixture of Gaussians $\frac{1}{2}\mathcal{N}(\mu_0, \mathbf{\Sigma}) + \frac{1}{2}\mathcal{N}(\mu_1, \mathbf{\Sigma})$, where $\mu_0 = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right]$, $\mu_1 = -\mu_0$, and $\mathbf{\Sigma} = \frac{1}{4}\mathbf{I}$.

CelebA. The CelebFaces Attributes dataset (CelebA) [59] is a large-scale face dataset used extensively in computer vision research, particularly for tasks such as face detection, face attribute recognition, etc. The dataset contains $202,599$ face images of $10,177$ celebrities with various poses and backgrounds. Besides, each image is annotated with 40 binary attributes, describing facial features and properties (e.g., having a mustache, wearing a hat or glasses, etc.). Following previous work,

we select 2 attributes at random and construct the set V with a size of 8 and the oracle set S^* with a size of 2 or 3, where neither attribute is present (e.g. not wearing glasses and hats).

Amazon. The Amazon Baby Registry dataset [60] includes various subsets of baby registry products chosen by customers. These products are then organized into 18 distinct categories. From these, 12 categories are selected. Each product in the dataset is described by a textual description, which has been transformed into a 768-dimensional vector using a pre-trained BERT model [61]. For each category, the (V, S^*) pairs are sampled using the following process. First, we exclude subsets chosen by customers that contain only one item or more than 30 items. Next, we divide the remaining subsets into training, validation, and test sets equally. For each oracle subset $S^* \in S$, we randomly sample $30 - |S^*|$ additional products from the same category to ensure that V contains exactly 30 products. This method constructs a data point (V, S^*) for each customer, simulating a real-world scenario where V represents 30 products shown to the customer, and S^* represents the subset of products the customer is interested in.