Learning Neural Set Functions Under the Optimal Subset Oracle

Anonymous Author(s)
Affiliation
Address
email

Abstract

Learning set functions becomes increasingly more important in many applications like product recommendation and compound selection in AI-aided drug discovery. The majority of existing works study methodologies of set function learning under the function value oracle, which, however, requires expensive supervision signals. This renders it impractical for applications with only weak supervisions under the Optimal Subset (OS) oracle, the study of which is surprisingly overlooked. In this work, we present a principled yet practical maximum likelihood learning framework, termed as `EquiVSet`,[1] that simultaneously meets the following desiderata of learning set functions under the OS oracle: i) permutation invariance of the set mass function being modeled; ii) permission of varying ground set; iii) minimum prior; and iv) scalability. The main components of our framework involve: an energy-based treatment of the set mass function, DeepSet-style architectures to handle permutation invariance, mean-field variational inference, and its amortized variants. Thanks to the delicate combination of these advanced architectures, empirical studies on three real-world applications (including Amazon product recommendation, set anomaly detection and compound selection for virtual screening) demonstrate that `EquiVSet` outperforms the baselines by a large margin.

1 Introduction

Many real-world applications involve prediction of set-value outputs, such as recommender systems which output a set of products to customers, anomaly detection that predicts the outliers from the majority of data (Zhang et al., 2020), and compound selection for virtual screening in drug discovery aims at extracting the most effective compounds from a given compound database (Gimeno et al., 2019). All of these applications implicitly learn a set function (Rezatofighi et al., 2017; Zaheer et al., 2017) that measures the utility of a given set input, such that the most desirable set output has the highest (or lowest *w.l.o.g*) utility value.

More formally, consider a recommender system: given a set of product candidates V, it is expected to recommend a subset of products $S^* \subseteq V$ to the user, which would satisfy the user most, *i.e.*, offering the maximum utility to the user. We assume the underlying process of determining S^* can be modelled by a utility function $F_\theta(S; V)$ parameterized by θ, and the following criteria:

$$S^* = \operatorname*{argmax}_{S \in 2^V} F_\theta(S; V). \tag{1}$$

There are mainly two settings for learning the utility function. The first one, namely function value (FV) oracle, targets at learning $F_\theta(S; V)$ to fit the utility explicitly, under the supervision of data in the form of $\{(S_i, f_i)\}$ for a fixed ground set V, where f_i is the true utility function value of the

[1] Code is available at: https://anonymous.4open.science/r/EquiVSet-DDD0

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

subset S_i. However, training in this way is prohibitively expensive, since one needs to construct large amounts of supervision signals for a specific ground set V (Balcan & Harvey, 2018). Here we consider an alternative setting, which learns $F_\theta(S; V)$ in an implicit way. More formally, with the data in form of $\{(V_i, S_i^*)\}_{i=1}^N$, where S_i^* is the optimal subset (OS) corresponding to V_i, our goal is to estimate θ such that for all possible (V_i, S_i^*), it satisfies equation (1). The OS oracle is arguably more practical than the FV oracle, which alleviates the need for explicitly labeling utility values for a large amount of subsets.[2]

Though being critical for practical success, related study on set utility function learning under the OS supervision oracle is surprisingly lacked. The most relevant work is the probabilistic greedy model (PGM), which solves optimization (1) with a greedy maximization algorithm (Tschiatschek et al., 2018). Specifically, PGM interprets the maximization algorithm as to construct differentiable distributions over sequences of items in an auto-regressive manner. However, such construction of distributions is problematic for defining distributions on sets due to the dependency on the sampling order. Therefore, they alleviate this issue by enumerating all possible permutations of the sampling sequence (detailed discussion is given in Appendix A). Such enumerations scale poorly due to the combinatorial cost $\mathcal{O}(|V|!)$, which hinders PGM's applicability to real-world applications.

To learn set functions under the OS oracle, we advocate the maximum likelihood paradigm (Stigler, 1986). Specifically, this learning problem can be viewed from a probabilistic perspective

$$\begin{aligned} &\operatorname*{argmax}_{\theta} \ \mathbb{E}_{\mathbb{P}(V,S)}[\log p_\theta(S|V)] \\ &\text{s.t. } \ p_\theta(S|V) \propto F_\theta(S; V), \forall S \in 2^V, \end{aligned} \tag{2}$$

where the constraint admits the learned set function to obey the objective defined in (1). Given limited data $\{(V_i, S_i^*)\}_{i=1}^N$ sampled from the underlying data distribution $\mathbb{P}(V, S)$, one would maximize the empirical log likelihood: $\sum_{i=1}^N [\log p_\theta(S_i^*|V_i)]$. The most important step is to construct a proper set distribution $p_\theta(S|V)$ whose probability mass monotonically grows with the utility function $F_\theta(S; V)$ and satisfy the following additive requirements: (i) *permutation invariance*: the probability mass should not change under any permutation of the elements in S; (ii) *varying ground set*: the function should be able to process input sets of variable size; iii) *minimum prior*: we should make no assumptions of the set probability, *i.e.,* with maximum entropy, which is equivalent to the uninformative prior (Jeffreys, 1946); and iv) *scalibility*: the learning algorithm should be scalable to large-scale datasets and run in polynomial time.

In this paper, we propose **Equi**variant **V**ariational inference for **Set** function learning (`EquiVSet`), a new method for learning set functions under the OS oracle, which satisfies all the requirements. Specifically, we use an energy-based model (EBM) to construct the set mass function. EBMs are maximum entropy distributions, which satisfies the *minimum prior* requirement. Moreover, by modeling the energy function with DeepSet-style architectures (Zaheer et al., 2017; Lee et al., 2019), the two requirements, *i.e., permutation invariance* and *varying ground set* are naturally satisfied. Unfortunately, the flexibility of EBMs exacerbates the difficulties of learning and inference, since the inputs of set are discrete and lie in an exponentially-large space. To remedy this issue, we develop an approximate maximum likelihood approach which estimates the marginals via the mean-field variational inference, resulting in an efficient training manner under the supervision of OS oracles. In order to ensure *scalability*, an amortized inference network with permutation equivariance is proposed, which allows the model to be trained on large-scale datasets.

Although it may be seen as combining existing components in approximate inference, the proposed framework addresses a surprisingly overlooked problem in the set function learning communities using an intuitive yet effective method. Our main contributions are summarized below:

- We formulate set functions learning problems under the OS supervision oracle using the maximum likelihood principle;
- We present a delicate framework based on EBMs which satisfies the four desirable requirements and is efficient both at training and inference stages;

[2]Notably, learning set functions under the OS oracle is distinct to that under the FV oracle; the two settings are not comparable in general. To illustrate this, one can easily obtain the FV oracle of maximum cut set functions, but fail to specify the OS oracle since it is NP-complete to solve the maximum cut problem (Garey & Johnson, 1979, Appendix A2.2). Besides, even though the OS oracle naturally shows up in the product recommendation scenario, one cannot identify its FV oracle since the true utility values are hard to obtain.

- Real-world experiments demonstrate effectiveness of the proposed OS learning framework.

2 Energy-Based Modeling for Set Function Learning

The first step to solve problem (2) is to construct a proper set mass function $p_\theta(S|V)$ monotonically growing with the utility function $F_\theta(S;V)$. There exits countless ways to construct such a probability mass function, such as the sequential modeling in PGM (Tschiatschek et al., 2018, Section 4). Here we resort to the energy-based treatment:

$$p_\theta(S|V) = \frac{\exp(F_\theta(S;V))}{Z},\ Z := \sum_{S' \subseteq V} \exp(F_\theta(S';V)), \tag{3}$$

where the utility function $F_\theta(S;V)$ stands for the negative energy, with higher utility representing lower energy. The energy-based treatment is attractive, partially due to its maximum entropy (*i.e.*, minimum prior) property. That is, it assumes nothing about what is unknown, which is known as the "noninformative prior" principle in Bayesian modeling (Jeffreys, 1946). This basic principle is, however, violated by the set mass function defined in PGM. We refer detailed motivation of the energy-based modeling to Appendix B.1.

In addition to the *minimum prior*, the energy-based treatment also enables the set mass function $p_\theta(S|V)$ to meet the other two requirements, *i.e. permutation invariance* and *varying ground set*, by deliberately designing a suitable set function $F_\theta(S;V)$. However, modeling such a proper function is nontrivial, since classical feed-forward neural networks (e.g., the ones designed for submodular set functions (Bilmes & Bai, 2017)) violate both two criteria, which restricts their applicability to the problems involving a set of objects. Fortunately, Zaheer et al. (2017) sidestep this issue by introducing a novel architecture, namely DeepSet. They theoretically prove the following Proposition.

Proposition 1. *All permutation invariant set functions can be decomposed in the form* $f(S) = \rho\left(\sum_{s\in S}\kappa(s)\right)$*, for suitable transformations* κ *and* ρ*.*

By combining the energy-based model in (3) with this architecture, we could construct a valid set mass function to meet two important criteria: *permutation invaraince* and *varying ground set*. However, the flexibility of EBMs exacerbates the difficulties of learning and inference, since the partition function Z is typically intractable and the input of sets is undesirably discrete.

3 Approximate Maximum Likelihood Learning with OS Supervision Oracle

In this section, we explore an effective framework for learning set functions under the supervision of optimal subset oracles. We start with discussing the principles for learning parameter θ, followed by discussing the detailed inference method for discrete EBMs.

3.1 Training Discrete EBMs Under the Guidance of Variational Approximation

For discrete data, *e.g.,* set, learning the parameter θ in (3) via maximum likelihood is notoriously difficult. Although one could apply techniques, such as ratio matching (Lyu, 2012), noise contrastive estimation (Tschiatschek et al., 2016), and contrastive divergence (Carreira-Perpinan & Hinton, 2005), they generally suffer from instability on high dimensional data, especially when facing very large ground set in real-world applications. Instead of directly maximizing the log likelihood, we consider an alternative optimization objective that is computationally preferable. Specifically, we first fit a variational approximation to the EBM by solving

$$\boldsymbol{\psi}^* = \operatorname*{argmin}_{\boldsymbol{\psi}} D(q(S;\boldsymbol{\psi})||p_\theta(S)), \tag{4}$$

where $D(\cdot||\cdot)$ is a discrepancy measure between two distributions, $p_\theta(S)$[3] is the EBM defined in (3), and $q(S;\boldsymbol{\psi})$ denotes the mean-field variational distribution with the parameter $\boldsymbol{\psi} \in [0,1]^{|V|}$ standing for the odds that each item $s \in V$ shall be selected in the optimal subset S^*. Note that the optimal

[3]Here we omit the condition V for brevity. In some specific context, it would be helpful to regard subset S as a binary vector, *i.e.*, $S := \{0,1\}^{|V|}$ with the i-th element equal to 1 meaning $i \in S$ and 0 meaning $i \notin S$.

Algorithm 1 MFVI($\boldsymbol{\psi}, V, K$)

for $k \leftarrow 1, \ldots, K$ **do**
  for $i \leftarrow 1, \ldots, |V|$ in parallel **do**
    sample m subsets $S_n \sim q(S; (\boldsymbol{\psi}^{(k-1)} | \psi_i^{(k-1)} \leftarrow 0))$
    update variational parameter $\boldsymbol{\psi}_i^{(k)} \leftarrow \sigma(\frac{1}{m}\sum_{n=1}^{m}[F_\theta(S_n + i) - F_\theta(S_n)])$
  end for
end for

Algorithm 2 DiffMF(V, S^*)

initialize variational parameter $\boldsymbol{\psi}^{(0)} \leftarrow 0.5 * \mathbf{1}$
compute the marginals $\boldsymbol{\psi}^* \leftarrow \mathrm{MFVI}(\boldsymbol{\psi}^{(0)}, V, K)$
update parameter θ using (5) $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta; \boldsymbol{\psi}^*)$

Algorithm 3 EquiVSet(V, S^*)

update parameter ϕ using (6) $\phi \leftarrow \phi + \eta \nabla_\phi \mathrm{ELBO}(\phi)$
initialize variational parameter $\boldsymbol{\psi}^{(0)} \leftarrow \mathrm{EquiNet}(V; \phi)$
one step fixed point iteration $\boldsymbol{\psi}^* \leftarrow \mathrm{MFVI}(\boldsymbol{\psi}^{(0)}, V, K = 1)$
update parameter θ using (5) $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta; \boldsymbol{\psi}^*)$

Figure 1: The main components and algorithms in our framework. Note that `DiffMF` and `EquiVSet` are for one training sample only. Detailed and self-contained descriptions of each component of these algorithms are presented in Appendix D.

parameter $\boldsymbol{\psi}^*$ of (4) can be viewed as a function of θ. In this regard, we can optimize the parameter θ by minimizing the following cross entropy loss,[4] which is well-known to be implementing the maximum likelihood estimation (Goodfellow et al., 2016) *w.r.t.* the surrogate distribution $q(S; \boldsymbol{\psi}^*)$,

$$\mathcal{L}(\theta; \boldsymbol{\psi}^*) = \mathbb{E}_{\mathbb{P}(V,S)}[-\log q(S; \boldsymbol{\psi}^*)] \approx \frac{1}{N}\sum_{i=1}^{N}\left(-\sum_{j \in S_i^*} \log \psi_j^* - \sum_{j \in V_i \setminus S_i^*} \log(1 - \psi_j^*)\right). \quad (5)$$

This is also known as the marginal-based loss (Domke, 2013), which trains probabilistic models by evaluating them using the marginals approximated by an inference algorithm. Despite not exactly bounding the log-likelihood of (3), this objective, as pointed out by Domke (2013), benefits from taking the approximation errors of inference algorithm into account while learning. However, minimizing (5) requires the variational parameter $\boldsymbol{\psi}^*$ being differentiable *w.r.t.* θ. Inspired by the differentiable variational approximation to the Markov Random Fields (Krähenbühl & Koltun, 2013; Zheng et al., 2015; Dai et al., 2016), below, we extend this method to the deep energy-based formulation, which admits an end-to-end training paradigm with the back-propagation algorithm.

3.2 Differentiable Mean Field Variational Inference

To solve the optimization problem (4), we need to specify the variational distribution $q(S; \boldsymbol{\psi})$ and the divergence measure $D(\cdot||\cdot)$, such that the optimum marginal $\boldsymbol{\psi}^*$ is differentiable *w.r.t.* the model parameter θ. A natural choice is to restrain $q(S; \boldsymbol{\psi})$ to be fully factorizable, which leads to a mean-field approximation of $p_\theta(S)$. The simplest form of $q(S; \boldsymbol{\psi})$ would be a $|V|$ independent Bernoulli distribution, *i.e.*, $q(S; \boldsymbol{\psi}) = \prod_{i \in S} \psi_i \prod_{i \notin S}(1 - \psi_i), \boldsymbol{\psi} \in [0, 1]^{|V|}$. Further restricting the discrepancy measure $D(q||p)$ to be the Kullback-Leibler divergence, we recover the well-known mean-field variational inference method. It turns out that minimizing the KL divergence amounts to maximizing the evidence lower bound (ELBO)

$$\min_{\boldsymbol{\psi}} \mathbb{KL}(q(S, \boldsymbol{\psi})||p_\theta(S)) \quad \Leftrightarrow \quad \max_{\boldsymbol{\psi}} f_{\mathrm{mt}}^{F_\theta}(\boldsymbol{\psi}) + \mathbb{H}(q(S; \boldsymbol{\psi})) =: \mathrm{ELBO}, \quad (6)$$

where $f_{\mathrm{mt}}^{F_\theta}(\boldsymbol{\psi})$ is the multilinear extension of $F_\theta(S)$ (Calinescu et al., 2007), which is defined as

$$f_{\mathrm{mt}}^{F_\theta}(\boldsymbol{\psi}) := \sum_{S \subseteq V} F_\theta(S) \prod_{i \in S} \psi_i \prod_{i \notin S}(1 - \psi_i), \boldsymbol{\psi} \in [0, 1]^{|V|}. \quad (7)$$

To maximize the ELBO in (6), one can apply the fixed point iteration algorithm. Specifically, for coordinate ψ_i, the partial derivative of the multilinear extension is $\nabla_{\psi_i} f_{\mathrm{mt}}^{F_\theta}(\boldsymbol{\psi})$, and for the entropy term, it is $\nabla_{\psi_i}\mathbb{H}(q) = \log \frac{1-\psi_i}{\psi_i}$. Thus, the stationary condition of maximizing ELBO is $\psi_i = \sigma(\nabla_{\psi_i} f_{\mathrm{mt}}^{F_\theta}(\boldsymbol{\psi})), i = 1, \ldots, |V|$, where σ is the sigmoid function, which means ψ_i should be updated as $\psi_i \leftarrow \sigma(\nabla_{\psi_i} f_{\mathrm{mt}}^{F_\theta}(\boldsymbol{\psi}))$. This analysis leads to the traditional mean field iteration, which updates each coordinate one by one (detailed derivation in Appendix B.2). In this paper, we suggest

[4] This objective would suffer from label-imbalanced problem when the size of OS is too small. In practice, we can apply negative sampling to overcome this problem: we randomly select a negative set $N_i \subseteq V_i \setminus S_i^*$ with the size of $|S^*|$, and train the model with an alternative objective $\sum_i -\sum_{j \in S_i^*} \log \psi_j^* - \sum_{j \in N_i} \log(1 - \psi_j^*)$.

to update $\boldsymbol{\psi}$ in a batch manner, which is more efficient in practice. More specifically, we summerize the mean field approximation as the following fixed-point iterative update steps

$$\boldsymbol{\psi}^{(0)} \leftarrow \text{Initialize in } [0,1]^{|V|}, \tag{8}$$

$$\boldsymbol{\psi}^{(k)} \leftarrow (1+\exp(-\nabla_{\psi^{(k-1)}} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi}^{(k-1)})))^{-1}, \tag{9}$$

$$\boldsymbol{\psi}^{*} \leftarrow \boldsymbol{\psi}^{(K)}. \tag{10}$$

We denote the above iterative steps as a function termed as $\text{MFVI}(\boldsymbol{\psi}, V, K)$, which takes initial vairational parameter $\boldsymbol{\psi}$, ground set V, and number of iteration steps K as input, and outputs the parameter $\boldsymbol{\psi}^*$ after K steps. Note that, $\text{MFVI}(\boldsymbol{\psi}, V, K)$ is differentiable *w.r.t.* the parameter θ, since each fixed-point iterative update step is differentiable. Thereby, one could learn θ by minimizing the cross entropy loss in (5). However, the computation complexity raises from the derivative of multilinear extension $f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi})$ defined in (7), which sums up all the possible subsets in the space of size $2^{|V|}$. Fortunately, the gradient $\nabla_\psi f_{\text{mt}}^{F_\theta}$ can be estimated efficiently via Monte Carlo approximation methods, since the following equation holds.

$$\nabla_{\psi_i} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi}) = \mathbb{E}_{q(S;(\boldsymbol{\psi}|\psi_i \leftarrow 0))}\left[F_\theta(S+i) - F_\theta(S)\right], \tag{11}$$

in which we use $S+i$ to denote the set union $S \cup \{i\}$. Detailed derivation is provided in Appendix B.3. According to (11), we can estimate the partial derivative $\nabla_{\psi_i} f_{\text{mt}}^{F_\theta}$ via Monte Carlo approximation: i) sample m subsets $S_n, n = 1, \ldots, m$ from the surrogate distribution $q(S;(\boldsymbol{\psi}|\psi_i \leftarrow 0))$; ii) approximate the expectation by the average $\frac{1}{m}\sum_{k=1}^{n}[F_\theta(S_n+i) - F_\theta(S_n)]$. After training, the OS for a given ground set can be sampled via

$$S = \text{topN}(\boldsymbol{\psi}^*), \tag{12}$$

where the function $\text{topN}(\mathbf{x})$ returns the indexes of top N values in the vector $\mathbf{x}$, and $\boldsymbol{\psi}^* = \text{MFVI}(\boldsymbol{\psi}, V, K)$ denotes the optimal variational parameter after K-step mean-field iteration. We term this method as **Diff**erentiable **M**ean **F**ield (`DiffMF`) and summarize the complete training and inference process in Algorithm 2 and 1, respectively.

4 Amortizing Inference with Equivariant Neural Networks

Although `DiffMF` can learn set function F_θ in an effective way, it undesirably has two notorious issues: i) the computation is in general prohibitively expensive, since `DiffMF` involves a typically expensive sampling loop per data point; ii) some information regarding interactions between elements is discarded, since `DiffMF` assumes a fully fatorizable variational distribution. In this section, we first propose to amortize the inference process with an additional recognition neural network, and then extend it to considering correlation for more accurate approximations.

4.1 Equivariant Amortized Variational Inference

To enable training the proposed model on a large-scale dataset, we propose to amortize the approximate inference process with an additional recognition neural network which outputs parameter $\boldsymbol{\psi}$ for the variational distribution $q_\phi(S;\boldsymbol{\psi})$,[5] where ϕ denotes the parameter of neural networks. A proper recognition network involving set objects shall satisfy *permutation equivariance*.

Definition 1. *A function $f : \mathcal{X}^d \to \mathcal{Y}^d$ is called permutation equivalent when upon permutation of the input instances permutes the output labels, i.e., for any permutation π: $f(\pi([x_1, \ldots, x_d])) = \pi(f([x_1, \ldots, x_d]))$.*

Zaheer et al. (2017) propose to formulate the *permutation equivariant* architecture as :

$$f_i(S) = \rho\left(\lambda\kappa(s_i) + \gamma\sum\nolimits_{s\in S}\kappa(s)\right), \tag{13}$$

where s_i denotes the ith element in the set S, λ, γ are learnable scalar variables, and ρ, κ are any proper transformations. Note that the output value of $f_i : 2^V \to [0,1]$ is relative to the ith coordinate, but not the order of the elements in S. Thus the equivariant recognition network, denoted as $\boldsymbol{\psi} = \text{EquiNet}(V;\phi) : 2^V \to [0,1]^{|V|}$, can be defined as $\text{EquiNet}_i := f_i$, which takes the ground set V as input and outputs the distribution parameter $\boldsymbol{\psi}$ for $q_\phi(S;\boldsymbol{\psi})$.

[5] With a slight abuse of notations, we use the same symbol here as in (6).

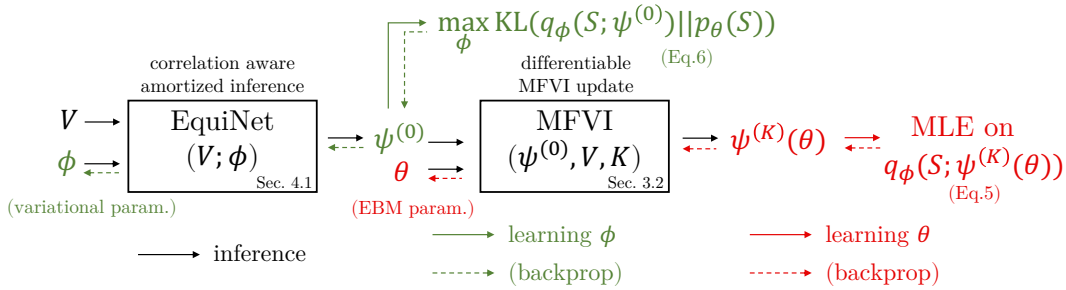

Figure 2: Overview of the training and inference processes of `EquiVSet`.

4.2 Correlation-aware Inference with Gaussian Copula

Due to the mean-field assumption, the proposed variational distribution cannot model the interactions among elements in the input set. We address this issue by introducing Gaussian copula (Nelsen, 2007), which is a cumulative distribution function (CDF) of random variables $(u_1, \ldots, u_{|V|})$ over the unit cube $[0,1]^{|V|}$, with $u_i \sim \mathrm{Uniform}(0,1)$. More formally, given a covariance matrix $\mathbf{\Sigma}$, the Gaussian copula $C_{\mathbf{\Sigma}}$ with parameter $\mathbf{\Sigma}$ is defined as

$$C_{\mathbf{\Sigma}}(u_1, \cdots, u_{|V|}) = \Phi_{\mathbf{\Sigma}}\left(\Phi^{-1}(u_1), \cdots, \Phi^{-1}(u_{|V|})\right),$$

where $\Phi_{\mathbf{\Sigma}}$ stands for the joint CDF of a Gaussian distribution with zero mean and covariance matrix $\mathbf{\Sigma}$, and Φ^{-1} is the inverse CDF of standard Gaussian. With the location parameter $\boldsymbol{\psi}$ output by $\mathrm{EquiNet}(V;\phi)$, we can induce correlation into the Bernoulli distribution via the following way: i) sample an auxiliary noise $\boldsymbol{g} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$; ii) apply element-wise Gaussian CDF $\boldsymbol{u} = \mathbf{\Phi}_{\mathrm{diag}(\mathbf{\Sigma})}(\boldsymbol{g})$; iii) obtain binary sample via $\boldsymbol{s} = \mathbb{I}(\boldsymbol{\psi} \leq \boldsymbol{u})$,[6] where $\boldsymbol{\psi} \leq \boldsymbol{u}$ means $\forall i, \psi_i \leq u_i$, $\mathbb{I}(\cdot)$ is the indicator function, and $\mathrm{diag}(\mathbf{\Sigma})$ returns the diagonal matrix of $\mathbf{\Sigma}$. In practice, the covariance matrix $\mathbf{\Sigma}$ could be generated by another neural network with the input ground set. We refer the discussion on it to Appendix C, and demonstrate how to efficiently construct and sample from a non-diagonal Gaussian distribution, while retaining a *permutation equivariant* sampling process.

To learn the parameters of the variational distribution, one can maximize the ELBO objective in (6). However, the ELBO has no differentiable closed-form expression *w.r.t.* ϕ.[7] To remedy this, we relax the binary variable $\boldsymbol{s}$ to a continuous one by applying the Gumbel-Softmax trick (Jang et al., 2016), resulting in an end-to-end training process with backpropagation.

4.3 Details of Training and Inference

Our model consists of two components: the EBM $p_\theta(S)$and the variational distribution $q_\phi(S;\boldsymbol{\psi})$. As shown in Fig. 2, these two components are trained in a cooperative learning scheme (Xie et al., 2018). Specifically, we train the variational distribution q_ϕ with fixed θ firstly by maximizing the ELBO in (6). To train the energy model p_θ, we first initialize the variational parameter $\boldsymbol{\psi}^{(0)}$ with the output of equivariant recognition network $\mathrm{EquiNet}(V;\phi)$. This enables us to get a more accurate variational approximate, since q_ϕ has modeled the correlation among the elements in the set. Notice that $\boldsymbol{\psi}^{(0)}$ does not depend on θ directly. To learn θ, we take one further step of mean-field iteration $\mathrm{MFVI}(\boldsymbol{\psi}^{(0)}, V, 1)$, which flows the gradient through θ and enables to optimize θ using the cross entropy loss in (5) (*i.e.,* if we skip step 3 in Alg. 3, and feed $\boldsymbol{\psi}^{(0)}$ to step 4, the gradient would not flow through θ). However, if we take multiple steps, it inclines to converge to the local optima that is the same as the original mean-field iteration. As a result, the benefit of correlation-aware inference provided by the Gaussian copula would be diminished. Detailed analysis is provided in Appendix F.4. The training procedure is summarized in Alg. 3 (the complete version is given in Appendix D).

For *inference* in the test time, given a ground set V, we initialize the variational parameter via $\boldsymbol{\psi}^{(0)} = \mathrm{EquiNet}(V)$, then run one step mean-field iteration $\boldsymbol{\psi}^* \leftarrow \mathrm{MFVI}(\boldsymbol{\psi}^{(0)}, V, 1)$. Finally, the corresponding OS is obtained by applying the rounding method in (12). We term our method as **Equi**variant **V**ariational Inference for **Set** Function Learning (`EquiVSet`), and respectively use $\texttt{EquiVSet}_{\text{ind}}$ and $\texttt{EquiVSet}_{\text{copula}}$ to represent two variants with independent and copula variational posterior.

[6] Here $\boldsymbol{s}$ is a binary vector $\{0,1\}^{|V|}$ with the i-th element equal to 1 meaning $i \in S$ and 0 meaning $i \notin S$.

[7] For correlation-aware inference, the variational parameter ϕ consists of two parts: i) ϕ of the $\mathrm{EquiNet}(V;\phi)$ and ii) $\mathbf{\Sigma}$ of the Gaussian copula.

5 Related Work

Set function learning. There is a growing literature on learning set functions with deep neural networks. Zaheer et al. (2017) designed the DeepSet architecture to create permutation invariant and equivariant function for set prediction. Lee et al. (2019) enhanced model ability of DeepSet by employing transformer layer to introduce correlation among instances of set, and Horn et al. (2020) extended this framework for time series. It is worth to be noted that they all learned set functions under the function value oralce and can be employed as the backbone of the utility function $F_\theta(S; V)$ in our model. Dolhansky & Bilmes (2016); Bilmes & Bai (2017); Ghadimi & Beigy (2020) have also designed deep architectures for submodular set functions, however, these designs can not handle the varying ground set requirement. There are papers studying the learnability of specific set functions (e.g., submodular functions and subadditive functions) in a distributional learning setting (Balcan et al., 2012; Badanidiyuru et al., 2012; Balcan & Harvey, 2018) under the function value oracle, they mainly provide sample complexity with inapproximability results under the probably mostly approximately correct (PMAC) learning model. Other methods relevant to our setting are TSPN (Kosiorek et al., 2020) and DESP (Zhang et al., 2020). However they both focused on generating set objects under a given condition. While we aim at predicting under the optimal subset oracle.

Energy-based modeling. Energy based learning (LeCun et al., 2006) is a classical framework to model the underlying distribution over data. Since it makes no assumption of data, energy-based models are extremely flexible and have been applied to wide ranges of domains, such as data generation (Nijkamp et al., 2019), out-of-distribution detection (Liu et al., 2020), game-theoretic valuation algorithms (Bian et al., 2022) and biological structure prediction (Shi et al., 2021). Learning EBMs can be done by applying some principled methods, like contrastive divergence (Hinton, 2002), score matching (Hyvärinen & Dayan, 2005), and ratio matching (Lyu, 2012). For inference, it is widely exploited gradient-based MCMC methods (Welling & Teh, 2011; Grathwohl et al., 2021). In this paper, we train discrete EBMs under the supervision of OS oracle by running mean-field inference. This new training paradigm may have some implicit connections with conventional methods, like noise contrastive estimation. We leave it for future works.

Amortized and Copula variational inference. Instead of approximating separate variables for each data point, amortized variational inference (VI) (Kingma & Welling, 2013) assumes that the variational parameters can be predicted by a parameterized function of the data (Zhang et al., 2018). The idea of amortized VI has been widely applied in deep probabilistic models (Hoffman et al., 2013; Garnelo et al., 2018). Although this procedure would introduce an amortization gap (Cremer et al., 2018), which refers to the suboptimality of variational parameters, amortized VI enables significant speedups and combines probabilistic modeling with the representational power of deep learning. Copula is the other method to improve the representational power for VI. Tran et al. (2015) used copula to augment the mean-field VI for better posterior approximation. Suh & Choi (2016) adopted Gaussian copula in VI to model the dependency structure of observed data. However, none of them can be directly applied to our setting involving discrete latent variables.

6 Empirical Studies

We evaluate the proposed methods on various tasks: set anomaly detection, product recommendation, and compound selection in drug discovery. All experiments are repeated five times with different random seeds. The model architectures and training details are deferred to Appendix E. Additional experiments on synthetic datasets and varying ground set are given in Appendix F.1 and Appendix F.2, respectively. Ablation studies on hyper-parameter choices (e.g. MFVI iteration step number, number of MC samples, and rank of perturbation) are provided in Appendix F.4.

Evaluations. We evaluate the methods using the mean Jaccard coefficient (MJC). Specifically, for each sample (V, S^*), denoting the corresponding model predict as S', the Jaccard coefficient is defined as $\mathrm{JC}(S, S') = \frac{|S' \cap S|}{|S' \cup S|}$. Then the MJC metric can be computed by averaging over all samples in the test set: $\mathrm{MJC} = \frac{1}{|\mathcal{D}_t|} \sum_{(V,S^*) \in \mathcal{D}_t} \mathrm{JC}(S^*, S')$.

Baselines. We compare our solution variants, *i.e.,* `DiffMF`, $\texttt{EquiVSet}_{\text{ind}}$, and $\texttt{EquiVSet}_{\text{copula}}$ to the following three baselines:

- Random: The expected performance of random guess, which estimates how difficult the task is.

Table 1: Product recommendation results on the Amazon dataset with different categories.

Categories	Random	PGM	DeepSet (NoSetFn)	`DiffMF` (ours)	`EquiVSet`$_{\text{ind}}$ (ours)	`EquiVSet`$_{\text{copula}}$ (ours)
Toys	0.0832	0.4414 ± 0.0036	0.4287 ± 0.0047	0.6147 ± 0.0102	0.6491 ± 0.0152	$\mathbf{0.6762 \pm 0.0221}$
Furniture	0.0651	0.1746 ± 0.0069	0.1758 ± 0.0072	0.1744 ± 0.0121	$\mathbf{0.1775 \pm 0.0108}$	0.1724 ± 0.0091
Gear	0.0771	0.4712 ± 0.0037	0.3806 ± 0.0019	0.5622 ± 0.0171	0.6103 ± 0.0193	$\mathbf{0.6973 \pm 0.0119}$
Carseats	0.0659	$\mathbf{0.2330 \pm 0.0115}$	0.2121 ± 0.0096	0.2229 ± 0.0104	0.2141 ± 0.0073	0.2149 ± 0.0123
Bath	0.0763	0.5638 ± 0.0077	0.4241 ± 0.0058	0.6901 ± 0.0061	0.6457 ± 0.0200	$\mathbf{0.7567 \pm 0.0095}$
Health	0.0758	0.4493 ± 0.0024	0.4481 ± 0.0041	0.5650 ± 0.0092	0.6315 ± 0.0153	$\mathbf{0.7003 \pm 0.0159}$
Diaper	0.0839	0.5802 ± 0.0092	0.4572 ± 0.0050	0.7011 ± 0.0112	0.7344 ± 0.0199	$\mathbf{0.8275 \pm 0.0136}$
Bedding	0.0791	0.4799 ± 0.0061	0.4824 ± 0.0081	0.6408 ± 0.0093	0.6287 ± 0.0195	$\mathbf{0.7688 \pm 0.0121}$
Safety	0.0648	0.2495 ± 0.0060	0.2211 ± 0.0044	0.2007 ± 0.0527	0.2250 ± 0.0287	$\mathbf{0.2524 \pm 0.0285}$
Feeding	0.0925	0.5596 ± 0.0081	0.4295 ± 0.0021	0.7496 ± 0.0114	0.6955 ± 0.0063	$\mathbf{0.8101 \pm 0.0074}$
Apparel	0.0918	0.5333 ± 0.0050	0.5074 ± 0.0036	0.6708 ± 0.0225	0.6465 ± 0.0150	$\mathbf{0.7521 \pm 0.0114}$
Media	0.0944	0.4406 ± 0.0092	0.4241 ± 0.0105	0.5145 ± 0.0105	0.5506 ± 0.0072	$\mathbf{0.5694 \pm 0.0105}$

- PGM (Tschiatschek et al. (2018), see Appendix A): The probabilistic greedy model, which is permutation invariant but computationally prohibitive.

- DeepSet (NoSetFn) (Zaheer et al., 2017): The deepset architecture, satisfying permutation invariant, is the backbone of our models. We train the deepset model: $2^{|V|} \to [0,1]^{|V|}$ with cross entropy loss and sample the subset via topN rounding in (12). The term "NoSetFn" is used to emphasize that although DeepSet can be applied to our empirical studies, it does not learn a set function explicitly.

Product Recommendation. In this experiment, we use the Amazon baby registry dataset (Gillenwater et al., 2014), which contains numerous subsets of products selected by different customers. Amazon characterizes each product in a baby registry as belonging to a specific category, such as "toys" and "furniture". Each product is characterized by a short textual description and we represent it as a 768 dimensional vector using the pre-trained BERT model (Devlin et al., 2018).

For each category, we generate samples (V, S^*) as follows. Firstly, we filter out those subsets selected by customers whose size is equal to 1 or larger than 30. Then we split the remaining subset collection $\mathcal{S}$ into training, validation and test folds with a $1:1:1$ ratio. Finally for each OS oracle $S^* \in \mathcal{S}$, we randomly sample additional $30 - |S^*|$ products from the same category to construct $V \backslash S^*$. In this way, we construct one data point (V, S^*) for each customer, which reflects this real world scenario: V contains 30 products displayed to the customer, and the customer is interested in checking $|S^*|$ of them. Note that this curation process is different from that of (Tschiatschek et al., 2018, Section 5.3), which is deviated from the real world scenario (Detailed discussion in Appendix E.4.).

The performance of all the models on different categories are shown in Table 1. Evidently, our models perform favorably to the baselines. Compared with PGM, which learns the set function via a probabilistic greedy algorithm, we can observe that our models, which model the the set functions with energy-based treatments, achieves better results on all settings. Although DeepSet is also permutation invariant, our model still outperforms it by a substantial margin, indicating the superiority of learning the set function explicitly.

Set Anomaly Detection. In this experiment, we evaluate our methods on two real-world datasets: the double MNIST (Sun, 2019) and the CelebA (Liu et al., 2015b). For each dataset, we randomly split the training, validation, and test set to the size of $10,000$, $1,000$, and $1,000$, respectively.

Double MNIST: The dataset consists of 1000 images for each digit ranging from 00 to 99. For each sample (V, S^*), we randomly sample $n \in \{2, \dots, 5\}$ images with the same digit to construct the OS oracle S^*, and then select $20 - |S^*|$ images with different digits to construct the set $V \backslash S^*$. **CelebA:** The CelebA dataset contains $202,599$ images with 40 attributes. We select two attributes at random and construct the set with the size of 8. For each ground set V, we randomly select $n \in \{2, 3\}$ images as the OS oracle S^*, in which neither of the two attributes is present. See Fig. 3 and Fig. 4 in Appendix E.5 for illustrations of sampled data.

From Table 2, we see that the variants of our model consistently outperform bot baseline methods strongly. Furthermore, we observe that by introducing the correlation to the variational distribution, significant performance gains can be obtained, demonstrating the benefits of relaxing the independent assumption by using Gaussian copula.

Compound Selection in AI-aided Drug Discovery. A critical step in drug discovery is to select compounds with high biological activity (Wallach et al., 2015; Li et al., 2021; Ji et al., 2022), diversity and satisfactory ADME (absorption, distribution, metabolism, and excretion) properties (Gimeno et al., 2019). As a result, virtual screening is typically a hierarchical filtering process with several necessary filters, e.g., first choosing the highly active compounds, then selecting diverse subsets from

Table 2: Set anomaly detection results.

Method	Double MNIST	CelebA
Random	0.0816	0.2187
PGM	0.3031 $\pm$ 0.0118	0.4812 $\pm$ 0.0064
DeepSet (NoSetFn)	0.1108 $\pm$ 0.0031	0.3915 $\pm$ 0.0133
`DiffMF` (ours)	**0.6064** $\pm$ **0.0133**	0.5455 $\pm$ 0.0079
`EquiVSet`$_{\text{ind}}$ (ours)	0.4054 $\pm$ 0.0122	0.5310 $\pm$ 0.0123
`EquiVSet`$_{\text{copula}}$ (ours)	0.5878 $\pm$ 0.0068	**0.5549** $\pm$ **0.0053**

Table 3: Compound selection results.

Method	PDBBind	BindingDB
Random	0.0725	0.0267
PGM	0.3499 $\pm$ 0.0087	0.1760 $\pm$ 0.0055
DeepSet (NoSetFn)	0.3189 $\pm$ 0.0034	0.1615 $\pm$ 0.0074
`DiffMF` (ours)	0.3534 $\pm$ 0.0143	0.1894 $\pm$ 0.0021
`EquiVSet`$_{\text{ind}}$ (ours)	**0.3553** $\pm$ **0.0049**	**0.1904** $\pm$ **0.0034**
`EquiVSet`$_{\text{copula}}$ (ours)	0.3536 $\pm$ 0.0083	0.1875 $\pm$ 0.0032

them, and finally excluding compounds that are bad for ADME. We finally arrive at a compound subset after a series of these steps. Given the OS supervision signals, we can learn to conduct this complicated selection process in an end to end manner. As a result, it will eliminate the need for intermediate supervision signals, which can be very expensive or impossible to obtain due to pharmacy's personal protection policy. For example, measuring the bioactivity and ADME properties of a compound has to be done in wet labs, and pharmaceutical companies are reluctant to disclose the data. Here we simulate the OS oracle of compound selection by applying the *two filters*: high bioactivity and diversity filters, based on the following two datasets.

PDBBind (Liu et al., 2015a): This dataset consists of experimentally measured binding affinities for bio-molecular complexes. We construct our dataset using the "refined" subsets therein, which contains 179 protein-ligand complexes. **BindingDB**[8]**:** It is a public database of measured binding affinities, which consists of $52,273$ drug-targets with small, drug-like molecules. Instead of providing complexes, here only the target amino acid sequence and compound SMILES string are provided.

We apply the same filtering process to construct samples (V, S^*) for these two datasets. Specifically, we first randomly select a number of compounds to construct the ground set V, whose size is 30 and 300 for PDBBind and BindingDB, respectively. Then $\frac{1}{3}$ compounds with the highest bioactivity are filtered out, accompanied by a distance matrix measured by the corresponding fingerprint similarity of molecules. To ensure diversity, the OS oracle S^* is generated by the centers of clusters which are presented by applying the affinity propagation algorithm. We finally obtain the training, validation, and test set with the size of 1,000, 100, and 100, respectively, for both two datasets. Detailed description is provided in Appendix E.6.

From Table 3, one can see that our methods magnificently outperform the random guess. This indicates that the proposed `EquiVSet` framework has great potential for drug discovery to facilitate the virtual screening task by modeling the complicated hierarchical selection process. Besides, improvements of `EquiVSet` can be further observed by comparing with DeepSet, which simply equips the deepset architecture with cross entropy loss, illustrating the superiority of explicit set function learning and energy-based modeling. Although comparable results could be achieved by PGM with sequential modeling, which satisfies permutation invariance and differentiability, our models still outperform it. This is partially because our models additionally maintain the other three desiderata of learning set functions, *i.e.,* varying ground set, minimum prior, and scalability. We also conduct a fairly simple task in Appendix F.3, in which only the bioactivity filter is considered. To simulate the full selection process, we leave it as important future work due to limited labels.

7 Discussion and Conclusions

We proposed a simple yet effective framework for set function learning under the OS oracle. By formulating the set probability with energy-based treatments, the resulting model enjoys the virtues of *permutation invariance*, *varying ground set*, and *minimum prior*. A *scalable* training and inference algorithm is further proposed by applying maximum log likelihood principle with the surrogate of mean-field inference. Real-world applications confirm the effectiveness of our approaches.

Limitations & Broader Impact. The training objective in (5) does not bound the log-likelihood of EBMs. This limitation can be mitigated by applying advanced discrete EBMs trainer, *e.g.,* GFlowNet (Zhang et al., 2022). We leave it as a future work. In addition, the proposed framework has the potential to facilitate learning to select subsets for other applications (Iyer et al., 2021), including active learning (Kothawade et al., 2021), targeted selection of subsets, selection of subsets for robustness (Killamsetty et al., 2020), and selection of subsets for fairness. Though we consider learning generic neural set functions in this work, it is beneficial to consider building useful priors into the neural set function architectures, such as set functions with the diminishing returns prior (Bilmes & Bai, 2017) and the bounded curvature/submodularity ratio prior (Bian et al., 2017).

[8]We take the curated one from https://tdcommons.ai/multi_pred_tasks/dti/

References

Badanidiyuru, A., Dobzinski, S., Fu, H., Kleinberg, R., Nisan, N., and Roughgarden, T. Sketching valuation functions. In Rabani, Y. (ed.), *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pp. 1025–1035. SIAM, 2012. doi: 10.1137/1.9781611973099.81. URL https://doi.org/10.1137/1.9781611973099.81.

Balcan, M. and Harvey, N. J. A. Submodular functions: Learnability, structure, and optimization. *SIAM J. Comput.*, 47(3):703–754, 2018. doi: 10.1137/120888909. URL https://doi.org/10.1137/120888909.

Balcan, M., Constantin, F., Iwata, S., and Wang, L. Learning valuation functions. In Mannor, S., Srebro, N., and Williamson, R. C. (eds.), *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, volume 23 of *JMLR Proceedings*, pp. 4.1–4.24. JMLR.org, 2012. URL http://proceedings.mlr.press/v23/balcan12b/balcan12b.pdf.

Bian, A. A., Buhmann, J. M., Krause, A., and Tschiatschek, S. Guarantees for greedy maximization of non-submodular functions with applications. In *International conference on machine learning*, pp. 498–507. PMLR, 2017.

Bian, Y., Rong, Y., Xu, T., Wu, J., Krause, A., and Huang, J. Energy-based learning for cooperative games, with applications to valuation problems in machine learning. *ICLR*, 2022.

Bilmes, J. A. and Bai, W. Deep submodular functions. *CoRR*, abs/1701.08939, 2017. URL http://arxiv.org/abs/1701.08939.

Calinescu, G., Chekuri, C., Pál, M., and Vondrák, J. Maximizing a submodular set function subject to a matroid constraint. In *International Conference on Integer Programming and Combinatorial Optimization*, pp. 182–196. Springer, 2007.

Carreira-Perpinan, M. A. and Hinton, G. On contrastive divergence learning. In *International workshop on artificial intelligence and statistics*, pp. 33–40. PMLR, 2005.

Cremer, C., Li, X., and Duvenaud, D. Inference suboptimality in variational autoencoders. In *International Conference on Machine Learning*, pp. 1078–1086. PMLR, 2018.

Dai, H., Dai, B., and Song, L. Discriminative embeddings of latent variable models for structured data. In *International conference on machine learning*, pp. 2702–2711. PMLR, 2016.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dolhansky, B. W. and Bilmes, J. A. Deep submodular functions: Definitions and learning. In Lee, D. D., Sugiyama, M., von Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, pp. 3396–3404, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/7fea637fd6d02b8f0adf6f7dc36aed93-Abstract.html.

Domke, J. Learning graphical model parameters with approximate marginal inference. *IEEE transactions on pattern analysis and machine intelligence*, 35(10):2454–2467, 2013.

Garey, M. R. and Johnson, D. S. *Computers and intractability*, volume 174. freeman San Francisco, 1979.

Garnelo, M., Schwarz, J., Rosenbaum, D., Viola, F., Rezende, D. J., Eslami, S., and Teh, Y. W. Neural processes. *arXiv preprint arXiv:1807.01622*, 2018.

Ghadimi, A. and Beigy, H. Deep submodular network: An application to multi-document summarization. *Expert Syst. Appl.*, 152:113392, 2020. doi: 10.1016/j.eswa.2020.113392. URL https://doi.org/10.1016/j.eswa.2020.113392.

Gillenwater, J. A., Kulesza, A., Fox, E., and Taskar, B. Expectation-maximization for learning determinantal point processes. *Advances in Neural Information Processing Systems*, 27:3149–3157, 2014.

Gimeno, A., Ojeda-Montes, M. J., Tomás-Hernández, S., Cereto-Massagué, A., Beltrán-Debón, R., Mulero, M., Pujadas, G., and Garcia-Vallvé, S. The light and dark sides of virtual screening: what is there to know? *International journal of molecular sciences*, 20(6):1375, 2019.

Gomes, J., Ramsundar, B., Feinberg, E. N., and Pande, V. S. Atomic convolutional networks for predicting protein-ligand binding affinity. *arXiv preprint arXiv:1703.10603*, 2017.

Goodfellow, I., Bengio, Y., and Courville, A. *Deep learning*. MIT press, 2016.

Grathwohl, W., Swersky, K., Hashemi, M., Duvenaud, D., and Maddison, C. J. Oops i took a gradient: Scalable sampling for discrete distributions. *arXiv preprint arXiv:2102.04509*, 2021.

Hinton, G. E. Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800, 2002.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

Horn, M., Moor, M., Bock, C., Rieck, B., and Borgwardt, K. Set functions for time series. In *International Conference on Machine Learning*, pp. 4353–4363. PMLR, 2020.

Hyvärinen, A. and Dayan, P. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(4), 2005.

Iyer, R., Khargonkar, N., Bilmes, J., and Asnani, H. Generalized submodular information measures: Theoretical properties, examples, optimization algorithms, and applications. *IEEE Transactions on Information Theory*, 2021.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Jaynes, E. T. Information theory and statistical mechanics. *Physical review*, 106(4):620, 1957a.

Jaynes, E. T. Information theory and statistical mechanics. ii. *Physical review*, 108(2):171, 1957b.

Jeffreys, H. An invariant form for the prior probability in estimation problems. *Proceedings of the Royal Society of London. Series A. Mathematical and Physical Sciences*, 186(1007):453–461, 1946.

Ji, Y., Zhang, L., Wu, J., Wu, B., Huang, L.-K., Xu, T., Rong, Y., Li, L., Ren, J., Xue, D., Lai, H., Xu, S., Feng, J., Liu, W., Luo, P., Zhou, S., Huang, J., Zhao, P., and Bian, Y. DrugOOD: Out-of-Distribution (OOD) Dataset Curator and Benchmark for AI-aided Drug Discovery – A Focus on Affinity Prediction Problems with Noise Annotations. *arXiv e-prints*, art. arXiv:2201.09637, January 2022.

Killamsetty, K., Sivasubramanian, D., Ramakrishnan, G., and Iyer, R. Glister: Generalization based data subset selection for efficient and robust learning. *arXiv preprint arXiv:2012.10630*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

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

Kosiorek, A. R., Kim, H., and Rezende, D. J. Conditional set generation with transformers. *arXiv preprint arXiv:2006.16841*, 2020.

Kothawade, S., Beck, N., Killamsetty, K., and Iyer, R. Similar: Submodular information measures based active learning in realistic scenarios. *Advances in Neural Information Processing Systems*, 34, 2021.

Krähenbühl, P. and Koltun, V. Parameter learning and convergent inference for dense random fields. In *International Conference on Machine Learning*, pp. 513–521. PMLR, 2013.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M., and Huang, F. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, pp. 3744–3753. PMLR, 2019.

Li, S., Zhou, J., Xu, T., Huang, L., Wang, F., Xiong, H., Huang, W., Dou, D., and Xiong, H. Structure-aware interactive graph neural networks for the prediction of protein-ligand binding affinity. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 975–985, 2021.

Liu, W., Wang, X., Owens, J. D., and Li, Y. Energy-based out-of-distribution detection. *arXiv preprint arXiv:2010.03759*, 2020.

Liu, Z., Li, Y., Han, L., Li, J., Liu, J., Zhao, Z., Nie, W., Liu, Y., and Wang, R. Pdb-wide collection of binding data: current status of the pdbbind database. *Bioinformatics*, 31(3):405–412, 2015a.

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

Lyu, S. Interpretation and generalization of score matching. *arXiv preprint arXiv:1205.2629*, 2012.

Nelsen, R. B. *An introduction to copulas*. Springer Science & Business Media, 2007.

Nijkamp, E., Hill, M., Zhu, S.-C., and Wu, Y. N. Learning non-convergent non-persistent short-run mcmc toward energy-based model. *arXiv preprint arXiv:1904.09770*, 2019.

Öztürk, H., Özgür, A., and Ozkirimli, E. Deepdta: deep drug–target binding affinity prediction. *Bioinformatics*, 34(17):i821–i829, 2018.

Rezatofighi, S. H., BG, V. K., Milan, A., Abbasnejad, E., Dick, A., and Reid, I. Deepsetnet: Predicting sets with deep neural networks. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 5257–5266. IEEE, 2017.

Shi, C., Luo, S., Xu, M., and Tang, J. Learning gradient fields for molecular conformation generation. *arXiv preprint arXiv:2105.03902*, 2021.

Stigler, S. M. *The history of statistics: The measurement of uncertainty before 1900*. Harvard University Press, 1986.

Suh, S. and Choi, S. Gaussian copula variational autoencoders for mixed data. *arXiv preprint arXiv:1604.04960*, 2016.

Sun, S.-H. Multi-digit mnist for few-shot learning, 2019. URL https://github.com/shaohua0116/MultiDigitMNIST.

Tran, D., Blei, D., and Airoldi, E. M. Copula variational inference. In *Advances in Neural Information Processing Systems*, pp. 3564–3572, 2015.

Tschiatschek, S., Djolonga, J., and Krause, A. Learning probabilistic submodular diversity models via noise contrastive estimation. In *Artificial Intelligence and Statistics*, pp. 770–779. PMLR, 2016.

Tschiatschek, S., Sahin, A., and Krause, A. Differentiable submodular maximization. *arXiv preprint arXiv:1803.01785*, 2018.

Wallach, I., Dzamba, M., and Heifets, A. Atomnet: a deep convolutional neural network for bioactivity prediction in structure-based drug discovery. *arXiv preprint arXiv:1510.02855*, 2015.

Welling, M. and Teh, Y. W. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 681–688. Citeseer, 2011.

Xie, J., Lu, Y., Gao, R., and Wu, Y. N. Cooperative learning of energy-based model and latent variable model via mcmc teaching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Zaheer, M., Kottur, S., Ravanbakhsh, S., Poczos, B., Salakhutdinov, R., and Smola, A. Deep sets. *arXiv preprint arXiv:1703.06114*, 2017.

Zhang, C., Bütepage, J., Kjellström, H., and Mandt, S. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):2008–2026, 2018.

Zhang, D., Malkin, N., Liu, Z., Volokhova, A., Courville, A., and Bengio, Y. Generative flow networks for discrete probabilistic modeling. *arXiv preprint arXiv:2202.01361*, 2022.

Zhang, D. W., Burghouts, G. J., and Snoek, C. G. Set prediction without imposing structure as conditional density estimation. *arXiv preprint arXiv:2010.04109*, 2020.

Zheng, S., Jayasumana, S., Romera-Paredes, B., Vineet, V., Su, Z., Du, D., Huang, C., and Torr, P. H. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE international conference on computer vision*, pp. 1529–1537, 2015.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes]
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] See Sec. 7.
   (c) Did you discuss any potential negative societal impacts of your work? [No] That is not explicitly pertinent to any negative social impact.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] See Appendix B.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Appendix E.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] See Appendix E.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Appendix E.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See Sec. 6
   (b) Did you mention the license of the assets? [Yes] The assets we use are public.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] Our code is available at a anonymous link.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] All datasets we evaluate on are open source datasets.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

Appendix for "Learning Neural Set Functions Under the Optimal Subset Oracle"

Contents

A Details of the Probabilistic Greedy Model **15**

B Derivations **16**

B.1 Derivations of the Maximum Entropy Distribution 16

B.2 Derivations of the Fixed Point Iteration 17

B.3 Derivations of the Gradient of Multilinear Extension 18

C Low-Rank Perturbation for the Covariance Matrix **18**

D Detailed Pseudo Code of EquiVSet Algorithms **18**

E Experimental Details **20**

E.1 The Architecture of EquiVSet 20

E.2 Implementation Details 21

E.3 Baselines 21

E.4 Detailed Experiment Settings for Product Recommendation 22

E.5 Detailed Experiment Settings for Set Anomaly Detection 23

E.6 Detailed Experiment Settings for Compound Selection 23

F Additional Experiments **24**

F.1 Synthetic Experiments 24

F.2 Experiments on Varying Ground Set 24

F.3 Experiments on Compound Selection with Only the Bioactivity Filter 25

F.4 Sensitivity Analysis of Hyperparameters 26

A Details of the Probabilistic Greedy Model

The probabilistic greedy model (PGM) solves optimization (1) with a differentiable extension of greedy maximization algorithm (Tschiatschek et al., 2018). Specifically, denote the first j chosen elements as $S_j = \{s_1, \ldots, s_j\} \subseteq V$, PGM samples the $(j+1)^{\text{th}}$ element from the candidate set $V \backslash S_j$ with the probability proportional to $\exp(F_\theta(s_{j+1} + S_j)/\gamma)$, which raises the probability of the selected elements in the sequence $\pi = \{s_1, s_2, \ldots, s_k\}$ as

$$p_\theta(\pi|V) = \prod_{j=0}^{k-1} \frac{\exp(F_\theta(s_{j+1} + S_j)/\gamma)}{\sum_{s \in V \backslash S_j} \exp(F_\theta(s + S_j)/\gamma)}, \tag{14}$$

where γ is a temperature parameter, $S_0 = \emptyset$, and $s + S := S \cup \{s\}$. Note that, the computation of $p_\theta(\pi|V)$ depends on the order of sequence π, which would make the learned parameter θ sensitive to the sampling order. To alleviate this problem, Tschiatschek et al. (2018) finally construct the set mass function by enumerating all possible permutations

$$p_\theta(S|V) = \sum_{\pi \in \Pi^S} p_\theta(\pi|V), \tag{15}$$

where Π^S is the permutation space generated from S. After training, the OS oracle S can be sampled via sequential decoupling $p(s_{i+1}|S_i) \propto \exp(F_\theta(s_{i+1} + S_i)/\gamma)$. However, maximizing the log

likelihood of (15) is prohibitively expensive and unscalable due to the exponential time complexity of enumerating all permutations. Although one can apply Monte Carlo approximation to avoid that, *i.e.,* approximating $\log p_\theta(S|V) = \log \sum_{\pi \in \Pi^S} p_\theta(\pi|V)$ with $\log p_\theta(\pi|V), \pi \sim \Pi^S$, such a simple estimator is biased, resulting in a permutation variant model.

B Derivations

B.1 Derivations of the Maximum Entropy Distribution

The first step to solve problem (2) is to construct a proper set mass function $p_\theta(S|V)$ monotonically growing with the utility function $F_\theta(S; V)$. There exits countless ways to construct such a probability mass function, such as $p_\theta(S|V) \propto F_\theta(S; V)$ and the set mass function defined in PGM, *i.e.,* Eq. (15). Here, one would care about what the most appropriate set mass function should be? Generally we prefer the model to assume nothing about what is unknown. More formally, we should choose the most "uniform" distribution, which maximizes the Shannon entropy $\mathbb{H}(p) = -\sum_{S \subseteq V} p(S) \log p(S)$. This principle is known as "noninformative prior" (Jeffreys, 1946), which has been widely applied in many physical systems (Jaynes, 1957a,b). It turns out that the energy-based model is the only distribution with maximum entropy. More specifically, the following theorem holds:

Theorem 1. *Let $\mathcal{P}_\mu := \{p(S) : \mathbb{E}_p[F(S)] = \mu\}$ be a set of distributions satisfying the expectation constraint $\mathbb{E}_p[F(S)] = \mu$, and p_λ have density*

$$p_\lambda(S) = \frac{\exp(\lambda F(S))}{Z}, \quad Z := \sum_{S \subseteq V} \exp(\lambda F(S)).$$

If $\mathbb{E}_{p_\lambda}[F(S)] = \mu$, then p_λ maximizes the entropy $\mathbb{H}(p)$ over $\mathcal{P}_\mu$; moreover, the distribution p_λ is unique.

Proof. The derivation below is adapted from Jaynes (1957a) in the context of set function learning, for completeness. We rewrite the maximum entropy problem in the form of

$$\begin{aligned} &\text{maximize } -\sum_{S \subseteq V} p(S) \log p(S) \\ &\text{subject to } \sum_{S \subseteq V} p(S) F(S) = \mu, \quad \forall S \subseteq V p(S) \geq 0, \quad \sum_{S \subseteq V} p(S) = 1. \end{aligned}$$

Introducing Lagrange multipliers $\alpha(S) > 0$ for the constraint $p(S) > 0$, $\beta \in \mathbb{R}$ for the normalization constraint that $\sum_{S \subseteq V} p(S) = 1$, λ for the constraint that $\mathbb{E}_p[F(S)] = \mu$, and , we obtain the following Lagrangian:

$$\begin{aligned} L(p, \alpha_0, \alpha_1, \beta) = \sum_{S \subseteq V} p(S) \log p(S) + \beta \left(\sum_{S \subseteq V} p(S) - 1 \right) + \\ \lambda \left(\mu - \sum_{S \subseteq V} p(S) F(S) \right) - \sum_{S \subseteq V} \alpha(S) p(S). \end{aligned} \tag{16}$$

Now we take derivatives and obtain

$$\frac{\partial}{\partial p(S)} L(p, \alpha, \beta, \lambda) = 1 + \log p(S) + \beta - \lambda F(S) - \alpha(S). \tag{17}$$

Since this function is convex in p, the minimizing p can be find by setting this equal to zero

$$p(S) = \exp(\lambda F(S) - 1 - \beta + \alpha(S)). \tag{18}$$

Note that in this setting we always have $p(S) > 0$. By complementary slackness, the constraint $p(S) > 0$ is unnecessary and we have $\alpha(S) = 0$. To satisfy the constraint $\sum_{S \subseteq V} p(S) = 1$, we take $\beta = 1 - + \log Z = -1 + \log \sum_{S \subseteq V} \exp(\lambda F(S))$. Then the optimal mass p has the form

$$p_\lambda(S) = \frac{\exp(\lambda F(S))}{\sum_{S \subseteq V} \exp(\lambda F(S))}. \tag{19}$$

So we reach the form of $p(S)$ we would like to have.

Next we show the distribution p_λ is unique. Assume there exists any other distribution $p \in \mathcal{P}_\mu$, such that $p = \operatorname{argmax}_p \mathbb{H}(p)$. In this case, we have

$$\begin{aligned}
\mathbb{H}(p) &= -\sum_{S\subseteq V} p(S)\log p(S) = -\sum_{S\subseteq V} p(S)\log\frac{p(S)}{p_\lambda(S)} - \sum_{S\subseteq V} p(S)\log p_\lambda(S) \\
&= -\mathbb{KL}(p||p_\lambda) - \sum_{S\subseteq V} p(S)\left(\lambda F(S) - Z\right) \\
&= -\mathbb{KL}(p||p_\lambda) - \sum_{S\subseteq V} p_\lambda(S)\left(\lambda F(S) - Z\right) \\
&= -\mathbb{KL}(p||p_\lambda) + \mathbb{H}(p_\lambda).
\end{aligned}$$

As $\mathbb{KL}(p||p_\lambda) \geq 0$ unless $p = p_\lambda$, we have shown that p_λ is the unique distribution maximizing the entropy, as desired. □

Discussion. Theorem 1 shows that EBM is the maximum entropy distribution, which verifies the assertion that energy-based treatments of set function enjoy the *minimum prior* property. It should be noted that the model proposed by Tschiatschek et al. (2018) violates this requirement. They used sequencial modeling to construct $p(S)$ (see (14) and (15)). Although this approach simplifies the sampling process, it introduces undesirable inductive bias.

B.2 Derivations of the Fixed Point Iteration

In this section, we give the detailed derivation for the fixed point iteration (FPI) of MFVI:

$$\psi_i^{(k+1)} \leftarrow (1 + \exp(-\nabla_{\psi_i^{(k)}} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi}^{(k)})))^{-1}. \tag{20}$$

First, recall that we want to maximize the ELBO:

$$\max_{\boldsymbol{\psi}} \underbrace{\sum_{S\subseteq V} F_\theta(S)\prod_{i\in S}\psi_i\prod_{i\notin S}(1-\psi_i)}_{f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi})} - \underbrace{\sum_{i=1}^{|V|}\left[\psi_i\log\psi_i + (1-\psi_i)\log(1-\psi_i))\right]}_{-\mathbb{H}(q(S;\boldsymbol{\psi}))}. \tag{21}$$

The formula (20) is obtained by setting the partial derivative *w.r.t.* coordinate i of ELBO to be 0:

$$\nabla_{\psi_i} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi}) + \nabla_{\psi_i}\mathbb{H}(q(S;\boldsymbol{\psi})) = \nabla_{\psi_i} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi}) + \log\frac{1-\psi_i}{\psi_i} = 0,$$

which implies

$$\psi_i = (1 + \exp(-\nabla_{\psi_i} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi})))^{-1}.$$

This is exactly the formula of FPI used in the mean-field variational inference algorithm. Note that the FPI actually corresponds to the gradient ascent with an adaptive step size vector $\boldsymbol{\alpha}$ as

$$\alpha_i = \frac{\sigma(\nabla_{\psi_i} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi})) - \psi_i}{\nabla_{\psi_i} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi}) + \log[(1-\psi_i)/\psi_i]},$$

where $\sigma(x) = (1+\exp(-x))^{-1}$ denotes the sigmoid function. To verify this, we have

$$\begin{aligned}
\psi_i^{(k+1)} &= \psi_i^{(k)} + \alpha_i^{(k)}\left(\nabla_{\psi_i^{(k)}} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi}^{(k)}) + \nabla_{\psi_i^{(k)}}\mathbb{H}(q(S;\boldsymbol{\psi}^{(k)}))\right) \\
&= \psi_i^{(k)} + \frac{\sigma(\nabla_{\psi_i} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi}^{(k)})) - \psi_i^{(k)}}{\nabla_{\psi_i^{(k)}} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi}^{(k)}) + \log[(1-\psi_i^{(k)})/\psi_i^{(k)}]}\left(\nabla_{\psi_i^{(k)}} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi}^{(k)}) + \log\frac{(1-\psi_i^{(k)})}{\psi_i^{(k)}}\right) \\
&= (1 + \exp(-\nabla_{\psi_i^{(k)}} f_{\text{mt}}^{F_\theta}(\boldsymbol{\psi}^{(k)})))^{-1}.
\end{aligned}$$

The connection to gradient ascent further confirms the soundness of our FPI algorithm.

B.3 Derivations of the Gradient of Multilinear Extension

In this section, we prove that the gradient of multilinear extension can be estimated using Monte Carlo sampling. Specifically we have

$$
\begin{aligned}
\nabla_{\psi_i} f_{\text{mt}}^{F_\theta} &= \nabla_{\psi_i} \sum_{S \subseteq V} F_\theta(S) \prod_{i \in S} \psi_i \prod_{i \notin S} (1 - \psi_i) \\
&= \mathbb{E}_{q(S;(\boldsymbol{\psi}|\psi_i \leftarrow 1))}[F_\theta(S)] - \mathbb{E}_{q(S;(\boldsymbol{\psi}|\psi_i \leftarrow 0))}[F_\theta(S)] \\
&= \sum_{S \subseteq V, i \in S} F_\theta(S) \prod_{j \in S \setminus \{i\}} \psi_j \prod_{j' \notin S} (1 - \psi_{j'}) - \sum_{S \subseteq V \setminus \{i\}} F_\theta(S) \prod_{j \in S} \psi_j \prod_{j' \in S, j' \neq i} (1 - \psi_{j'}) \\
&= \sum_{S \subseteq V \setminus \{i\}} [F_\theta(S + i) - F_\theta(S)] \prod_{j \in S} \psi_j \prod_{j' \in V \setminus S \setminus \{i\}} (1 - \psi_{j'}) \\
&= \mathbb{E}_{q(S;(\boldsymbol{\psi}|\psi_i \leftarrow 0))} \left[F_\theta(S + i) - F_\theta(S) \right]. \qquad (22)
\end{aligned}
$$

Discussion. The Monte Carlo (MC) approximation of $\nabla_{\psi_i} f_{\text{mt}}^{F_\theta}$ is unbiased. Thereby, although exactly calculating (22) has exponential time complexity, we can apply MC sampling to approximate it in a polynomial time, resulting a scalable training algorithm. It is worth to note that the MC approximation used in PGM (see (15)) is biased. That is they approximate $\log p_\theta(S|V) = \log \sum_{\pi \in \Pi^S} p_\theta(\pi|V)$ with $\log p_\theta(\pi|V), \pi \sim \Pi^S$. Although such a biased approximation can be computed in polynomial time, they undesirably introduce permutation variance.

C Low-Rank Perturbation for the Covariance Matrix

In the construction of Gaussian copula $C_{\boldsymbol{\Sigma}}$, we require a positive semi-definite matrix $\boldsymbol{\Sigma} \in \mathbb{R}^{|V| \times |V|}$, whose elements are generally modeled as the output of neural networks. Thereby, if the size of ground size V is large, the number of neural network outputs will be prohibitively large. Meanwhile, based on the definition of set, covariance matrix $\boldsymbol{\Sigma}$ is further required to satisfy *permutation equivariance*. To remedy this issue, we propose to employ a more efficient strategy, namely *Lower-Rank Perturbation*, which restricts the covariance matrix to the form

$$
\boldsymbol{\Sigma} = \boldsymbol{D} + \boldsymbol{P}\boldsymbol{P}^T, \qquad (23)
$$

where $\boldsymbol{\Sigma} \in \mathbb{R}_+^{|V| \times |V|}$ is a diagonal matrix with positive entries and $\boldsymbol{P} = [\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_v]$ is a lower-rank perturbation matrix with $\boldsymbol{p}_i \in \mathbb{R}^{|V|}$ and $v \ll |V|$. In this way, the number of neural network outputs can be dramatically reduced from $|V|^2$ to $v|V|$. Another benefit of constructing $\boldsymbol{\Sigma}$ in this way is that, it is convenient to employ the DeepSet architecture in (13) to output $\boldsymbol{D}$ and $\boldsymbol{p}_i$ for $i = 1, \ldots, v$, such that they are *permutation equivariant*, and the resulting covariance matrix $\boldsymbol{\Sigma} = \boldsymbol{D} + \boldsymbol{P}\boldsymbol{P}^T$ is also *permutation equivariant*. Moreover, the lower-rank perturbation trick permits us to avoid using Cholesky decomposition to sample a Gaussian noise with covariance $\boldsymbol{\Sigma}$, which is prohibitively expensive. Specifically, the Gaussian noise $\boldsymbol{g} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$ can be reparameterized as

$$
\boldsymbol{g} = \boldsymbol{D}^{1/2} \cdot \boldsymbol{\epsilon}_1 + \boldsymbol{P} \cdot \boldsymbol{\epsilon}_2, \qquad (24)
$$

where $\boldsymbol{\epsilon}_1 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_{|V|})$ and $\boldsymbol{\epsilon}_2 \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_v)$. In this way, the sampling complexity can be reduced from $\mathcal{O}(|V|^3)$ to $\mathcal{O}(v^2|V|)$.

D Detailed Pseudo Code of EquiVSet Algorithms

We provide the pseudo-code for `EquiVSet` in Alg. 4. The training procedure consists of two steps: i) train q_ϕ with fixed θ; ii) train p_θ under the guidance of q_ϕ. Specifically, to train q_ϕ, we first fix the parameter θ of the set function and then optimize ϕ by maximizing the ELBO in (6). To train p_θ, we first initialize the variational parameter $\boldsymbol{\psi}$ via EquiNet and then run K steps mean-field iteration to make $\boldsymbol{\psi}$ dependent with θ. Finally, the parameter θ can be optimized by minimizing the cross entropy in (5). Note that, we set K as 1 in our experiments.

Algorithm 4 EquiVSet (complete version)

Input: $\{V_i, S_i^*\}_{i=1}^N$: training dataset; η: learning rate; K : number of mean-field iteration step; m : number of Monte Carlo approximations; v: rank of perturbation; τ: temperature for Gumbel-Softmax
Output: Optimal parameters (θ, ϕ)

$\theta, \phi \leftarrow$ Initialize parameter
repeat
  Sample training data point
  $(V, S^*) \sim \{V_i, S_i^*\}_{i=1}^N$
  Obtain variational parameter $\boldsymbol{\psi}$ via EquiNet
  $\boldsymbol{\psi} \leftarrow \mathrm{EquiNet}(V; \phi)$
  Sample m subsets via $\mathrm{CopulaBernoulli}(V, v, \tau)$ or $\mathrm{IndBernoulli}(V, \tau)$
  $S_n \sim q(S; \boldsymbol{\psi}), n = 1, 2, \ldots, m$
  Update the parameter ϕ by maximizing ELBO in (6)
  $\phi \leftarrow \phi + \eta \nabla_\phi \left(\frac{1}{m} \sum_{n=1}^{m} F_\theta(S_n) + \sum_{i=1}^{|V|} [\psi_i \log \psi_i + (1 - \psi_i) \log(1 - \psi_i)] \right)$
  Optimize ϕ
  Initialize parameter $\boldsymbol{\psi}$ via EquiNet
  $\boldsymbol{\psi}^{(0)} \leftarrow \mathrm{EquiNet}(\mathrm{V}; \phi)$
  for $k \leftarrow 1, \ldots, K$ **do**
    for $i \leftarrow 1, \ldots, |V|$ in parallel **do**
      Sample m subsets via $\mathrm{CopulaBernoulli}(V, v, \tau)$ or $\mathrm{IndBernoulli}(V, \tau)$
      $S_n \sim q(S; \boldsymbol{\psi}^{(0)} | \psi_i^{(0)} \leftarrow 0), n = 1, 2, \ldots, m$
      Update the variational parameter $\boldsymbol{\psi}$
      $\psi_i^* \leftarrow \sigma \left(\frac{1}{m} \sum_{n=1}^{m} [F_\theta(S_n + i) - F_\theta(S_n)] \right)$
    end for
  end for
  Mean-field Iteration $\mathrm{MFVI}(\boldsymbol{\psi}^{(0)}, V, K)$
  Update the parameter θ by minimizing the cross entropy loss in (5)
  $\theta \leftarrow \theta - \eta \nabla_\theta \left(- \sum_{i \in S^*} \log \psi_i^* - \sum_{i \in V \setminus S^*} \log(1 - \psi_i^*) \right)$
  Optimize θ
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$

Algorithm 5 $\mathrm{IndBernoulli}(V, \tau)$

Input: V: ground set; τ: temperature for Gumbel-Softmax
Output: Sampled subset $\boldsymbol{s}$

1: Obtain location parameter $\boldsymbol{\psi}$ via EquiNet: $\boldsymbol{\psi} \leftarrow \mathrm{EquiNet}(V; \phi)$
2: Draw Gaussian noise: $u_i \sim \mathcal{N}(0, I), i = 1, \ldots, |V|$
3: Apply Gumbel-Softmax trick: $\tilde{s}_i = \sigma \left(\frac{1}{\tau} \left(\frac{\psi_i}{1 - \psi_i} + \log \frac{u_i}{1 - u_i} \right) \right), i = 1, \ldots, |V|$
4: Apply Straight-through estimator: $\boldsymbol{s} = \mathrm{stop_gradient}(\mathbb{I}(\boldsymbol{\psi} \leq \boldsymbol{\epsilon}) - \tilde{\boldsymbol{s}}) + \tilde{\boldsymbol{s}}, \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$

Algorithm 6 $\mathrm{CopulaBernoulli}(V, v, \tau)$

Input: V: ground set; v: rank of perturbation; τ: temperature for Gumbel-Softmax
Output: Sampled subset $\boldsymbol{s}$

1: Obtain location parameter $\boldsymbol{\psi}$ via EquiNet: $\boldsymbol{\psi} \leftarrow \mathrm{EquiNet}(V; \phi)$
2: Draw Gaussian noise:
  { In the following, $\boldsymbol{D}$ is a diagonal matrix and $\boldsymbol{P}$ is the lower-rank perturbation matrix. }
  $\boldsymbol{g} = \boldsymbol{D}^{1/2} \cdot \boldsymbol{\epsilon}_1 + \boldsymbol{P} \cdot \boldsymbol{\epsilon}_2, \boldsymbol{P} = [\boldsymbol{p}_1, \boldsymbol{p}_2, \ldots, \boldsymbol{p}_v], \boldsymbol{\epsilon}_1 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_{|V|})$ and $\boldsymbol{\epsilon}_2 \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I}_v)$
3: Apply element-wise Gaussian CDF: $\boldsymbol{u} = \boldsymbol{\Phi}_{\mathrm{diag}(\boldsymbol{D} + \boldsymbol{P}\boldsymbol{P}^T)}(\boldsymbol{g})$
4: Apply Gumbel-Softmax trick: $\tilde{s}_i = \sigma \left(\frac{1}{\tau} \left(\frac{\psi_i}{1 - \psi_i} + \log \frac{u_i}{1 - u_i} \right) \right), i = 1, \ldots, |V|$
5: Apply Straight-through estimator: $\boldsymbol{s} = \mathrm{stop_gradient}(\mathbb{I}(\boldsymbol{\psi} \leq \boldsymbol{u}) - \tilde{\boldsymbol{s}}) + \tilde{\boldsymbol{s}}$

E Experimental Details

E.1 The Architecture of EquiVSet

In this section, we provide a detail architecture description of $\texttt{EquiVSet}_{\text{copula}}$. $\texttt{EquiVSet}_{\text{copula}}$ consists of two different components that are implemented as neural networks: (i) the set function which is permutation invaraint and (ii) the recognition network which is permutation equivariant. We employ the DeepSet architecture to implement these two components, with the detailed architectures are given in Table 4.

Table 4: Detailed architectures of $\texttt{EquiVSet}_{\text{copula}}$.

Set Function	**Recognition Network**	
$\text{InitLayer}(S, 256)$	$\text{InitLayer}(V, 256)$	
SumPooling	$\text{FC}(256, 500, \text{ReLU})$	
$\text{FC}(256, 500, \text{ReLU})$	$\text{FC}(500, 500, \text{ReLU})$	
$\text{FC}(500, 500, \text{ReLU})$	$\boldsymbol{\psi} = \text{FC}(500, 1, \text{sigmoid})$	$\boldsymbol{D} = \text{diag}(\text{FC}(500, 1, \text{softplus}))$
$\text{FC}(256, 1, -)$		$\boldsymbol{P} = [\text{FC}(500, 1, \tanh)]^v$

In Table 4, $\text{InitLayer}(S, d)$ denotes the set transformation function, which encodes the set objects into vector representations. $\text{FC}(d, h, f)$ denotes the fully-connected layer with activation function f. $\text{diag}(\boldsymbol{v})$ is a diagonal matrix with the elements of diagonal being vector $\boldsymbol{v}$. $[\boldsymbol{p}]^v = [\boldsymbol{p}_1, \ldots, \boldsymbol{p}_v]$ denotes a matrix with $\boldsymbol{p}$ representing a column perturbation vector. Note that we also propose two variant methods, *i.e.,* `DiffMF` and $\texttt{EquiVSet}_{\text{ind}}$. For `DiffMF`, we apply the same architecture of the the set function in Table 4. We also exploit the same architecture for $\texttt{EquiVSet}_{\text{ind}}$, but discarding the copula components, *i.e.,* $\boldsymbol{D}$ and $\boldsymbol{P}$. In all experiments, we implement our models following the same architecture with the difference being that we apply various InitLayer to different datasets. The architectures of InitLayer for different datasets are depicted below.

Synthetic datasets. The synthetic datasets consist of the Tow-Moons and Gaussian-Mixture datasets. Each instance of the set is a two-dimensional vector, which represents the corresponding Cartesian coordinates. In this dataset, the InitLayer is a one-layer feed-forward neural network $\text{FC}(2, 256, -)$.

Amazon Baby Registry. The Amazon baby registry dataset consists of a set of products that are characterized by a short textual description. We transform them into vector representations using the pre-trained BERT module (Devlin et al., 2018). Thereby, each instance of the set is a 768 dimensional feature vector. The InitLayer is modelled as $\text{FC}(768, 256, -)$.

Double MNIST. The double MNIST dataset consists of different digit images ranging from 00 to 99. Each image has the shape of $(64, 64)$ and we reshape it into $(4096,)$. Therefore, the InitLayer is designed as $\text{FC}(4096, 256, -)$.

CelebA. The CelebA dataset contains $202,599$ number of face images. Each image is in the shape of $(3, 64, 64)$. We employ convolutional neural networks as InitLayer. Specifically, the architecture of InitLayer is

$$\text{ModuleList}([\text{Conv}(32, 3, 2, \text{ReLU}), \text{Conv}(64, 4, 2, \text{ReLU}),$$
$$\text{Conv}(128, 5, 2, \text{ReLU}), \text{MaxPooling}, \text{FC}(128, 256, -)]),$$

where $\text{Conv}(d, k, s, f)$ is a convolutional layer with d output channels, k kernel size, s stride size, and activation function f.

PDBBind. The PDBBind database consists of experimentally measured binding affinities for biomolecular complexes (Liu et al., 2015a). It provides detailed 3D Cartesian coordinates of both ligands and their target proteins derived from experimental (*e.g.*, X-ray crystallography) measurements. The atomic convolutional network (ACNN) (Gomes et al., 2017) provides meaningful vector features for complexes by constructing nearest neighbor graphs based on the 3D coordinates of atoms and predicting binding free energies. In this work, we apply the output of last second layer of the ACNN model followed by feed-forward neural networks to obtain the representations of complexes. More formally, the InitLayer is defined as

$$\text{ModuleList}([\text{ACNN}[:-1], \text{FC}(1922, 2048, \text{ReLU}), \text{FC}(2048, 256, -)]),$$

where $\mathrm{ACNN}[:-1]$ denotes the ACNN module without the last prediction layer, whose output dimensionality is 1922.

BindingDB. The BindingBD dataset containts $52,273$ drug-target pairs. We exploit the DeepDTA model (Öztürk et al., 2018) to encode drug-target pairs as vector representations. Specifically, the DeepDTA model first represents the drug compound and target protein as sequences of one-hot vectors and encodes them as feature vectors using convolutional neural networks. The detailed architecture of $\mathrm{InitLayer}$ used in this dataset is demonstrated in Table 5.

Table 5: Detailed architectures of InitLayer in the BindingDB dataset.

Drug	**Target**
$\mathrm{Conv}(32, 4, 1, \mathrm{ReLU})$	$\mathrm{Conv}(32, 4, 1, \mathrm{ReLU})$
$\mathrm{Conv}(64, 6, 1, \mathrm{ReLU})$	$\mathrm{Conv}(64, 8, 1, \mathrm{ReLU})$
$\mathrm{Conv}(96, 8, 1, \mathrm{ReLU})$	$\mathrm{Conv}(96, 12, 1, \mathrm{ReLU})$
$\mathrm{MaxPooling}$	$\mathrm{MaxPooling}$
$\mathrm{FC}(96, 256, \mathrm{ReLU})$	$\mathrm{FC}(96, 256, \mathrm{ReLU})$
Concat	
$\mathrm{FC}(512, 256, -)$	

E.2 Implementation Details

Here we provide a detailed description of the hyperparameters setup for our model `EquiVSet` and its variants. `EquiVSet` contains four important hyperprameters: the number of Monte Carlo sampling m and mean-field iteration steps K in Alg. 1, and the rank of lower-rank perturbation v in (23). We set $m = 5, v = 5$ throughout the experiments. For the mean-field iteration steps K, we set it as 5 for the variant model `DiffMF`, and 1 for $\texttt{EquiVSet}_{\text{ind}}$ and $\texttt{EquiVSet}_{\text{copula}}$. It is noted that the hyperparameters above are empirically set, and we have detail sensitivity analysis in Appendix F.4. The proposed models are trained using the Adam optimizer (Kingma & Ba, 2014) with a fixed learning rate $1e-4$ and weight decay rate $1e-5$. We choose the batch size from $\{4, 8, 16, 32, 64, 128\}$, since the model sizes for various datasets are different and we choose the largest batch size to enable it can be trained on a single Tesla V100-SXM2-32GB GPU.

We apply the early stopping strategy to train the models, including the baselines and our models. That is if the performances are not improved in continuous 6 epochs, we early stop the training process. Each dataset is trained for maximum 100 epochs. After each epoch, we validate the model and save the model with the best performance on the validation set. After training, we evaluate the performance of saved models on the test set. We repeat all experiments 5 times with different random seeds and the average performance metrics are reported as the final performances.

E.3 Baselines

Throughout the experiments, we compared our models with three conventional approaches: random guess, probabilistic greedy model (PGM) (Tschiatschek et al., 2018) and DeepSet (Zaheer et al., 2017). Further descriptions of the benchmarks and implementation details are as follows.

- Random: We report the expected value of the Jaccard coefficient (JC) of random guess. This baseline provides an estimate of how difficult the task is. Specifically, given a data point (V, S^*), it can be computed as $\mathbb{E}(JC(V, S^*)) = \sum_{k=0}^{|S^*|} \frac{\binom{|S^*|}{k}\binom{|V|-|S^*|}{|S^*|-k}}{\binom{|V|}{|S^*|}} \frac{k}{2|S^*|-k}$.

- PGM Tschiatschek et al. (2018): PGM is the most relevant method to the set functions learning under the OS oracle, that solves optimization (1) using greedy maximization algorithm with the virtues of differentiability and permutation invariance. We employ the same architecture defined in Table 4 to model the set function $F_\theta(S)$ in (14). The temperature parameter γ is empirically set as 1. We use Monte Carlo sampling to estimate (15). That is we randomly sample one permutation $\pi \sim \Pi^S$ and use $p_\theta(\pi|V)$ to approximate $p_\theta(S|V)$. The model is trained using the Adam optimizer, with batch size choosing from $\{4, 8, 16, 32, 64, 128\}$, fixed learning rate $1e-4$, and fixed weight decay rate $1e-5$.

Table 6: The statistics of Amazon product dataset

Categories	$\lvert\mathcal{D}\rvert$	$\lvert V\rvert$	$\sum \lvert S^*\rvert$	$\mathbb{E}[\lvert S^*\rvert]$	$\min_{S^*} \lvert S^*\rvert$	$\max_{S^*} \lvert S^*\rvert$
Toys	2,421	30	9,924	4.09	3	14
Furniture	280	30	892	3.18	3	6
Gear	4,277	30	16,288	3.80	3	10
Carseats	483	30	1,576	3.26	3	6
Bath	3,195	30	12,147	3.80	3	11
Health	2,995	30	11,053	3.69	3	9
Diaper	6,108	30	25,333	4.14	3	15
Bedding	4,524	30	17,509	3.87	3	12
Safety	267	30	846	3.16	3	5
Feeding	8,202	30	37,901	4.62	3	23
Apparel	4,675	30	21,176	4.52	3	21
Media	1,485	30	6,723	4.52	3	19

- DeepSet (NoSetFn) (Zaheer et al., 2017): DeepSet is a neural-network-based architecture that satisfies permutation invariance and varying ground sets. Although the DeepSet architecture can be employed here to sample the optimal subset oracle, it does not learn the set functions explicitly. We exploit the same architecture of $F_\theta(S)$ in Table 4, but drop the $\mathrm{SumPooling}$ operator to ensure the dimensionality of output is $|V|$. This baseline is trained by minimizing the objective in (5) using the Adam optimizer with batch size choosing from $\{4, 8, 16, 32, 64, 128\}$, fixed learning rate $1e-4$, and fixed weight decay rate $1e-5$.

E.4 Detailed Experiment Settings for Product Recommendation

Detailed Descriptions of the Amazon Baby Registry Dataset. The Amazon baby registry data (Gillenwater et al., 2014) consists of baby registry data collected from Amazon and is split into several datasets according to product categories, such as toys, furniture, etc. For each category, which can be considered as the product database, Amazon provides multiple sets of products selected by different customers. Thereby, these subsets of products can be viewed as OS oracles. To ensure that each ground set V only contains one OS oracle S^*, we construct the sample (V, S^*) as follows. For each subset of products selected by an anonymous user, we filter it out if its size is equal to 1 or larger than 30. For each OS oracle S^* in the remaining subsets, we randomly sample $30 - |S^*|$ products in the same category to construct $V \backslash S^*$. We summarize the statistics of the categories in Table 6.

Comparing with the Setting of (Tschiatschek et al., 2018, Section 5.3). In (Tschiatschek et al., 2018, Section 5.3), Tschiatschek et al. (2018) consider an alternative setting which is different from ours. Specifically, they construct the ground set V as all the products in a category, and view the selected subsets of all the customers as the corresponding optimal subsets. That is why they have the data points in the form of $\{V, (S_1^*, \ldots, S_N^*)\}$. This is a bit problematic since it is deviated from the real world scenario: naturally the chosen subset S_i^* shall depend on both V and the i-th customer's personal preference. However, the customer is fully anonymized, so no information can be extracted from this dataset.

In order to be aligned with the real world scenario, we curate the dataset in the following way, in order to make data samples in the OS supervision oracle with the data in the form of $\{V_i, S_i^*\}$.

For each category, we generate samples (V, S^*) as follows. Firstly, we filter out those subsets selected by customers whose size is equal to 1 or larger than 30. Then we split the remaining subset collection $\mathcal{S}$ into training, validation and test folds with a $1:1:1$ ratio. Finally for each OS oracle $S^* \in \mathcal{S}$, we randomly sample additional $30 - |S^*|$ products from the same category to construct $V \backslash S^*$.

In this way, we construct one data point (V, S^*) for each customer, which reflects this real world scenario: V contains 30 products displayed to the customer, and the customer is interested in checking $|S^*|$ of them. This is also consistent with real world recommender system, as users can only browse a small number of products at a time since the screen size of the device is limited, and the user has limited attention.

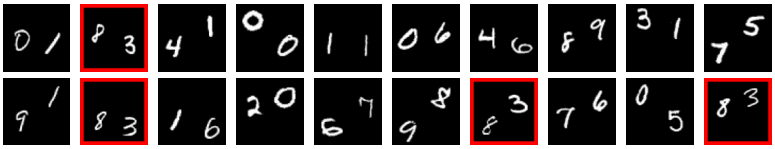

Figure 3: A sampled data for the Double MNIST dataset, which consists of $|S^*|$ images with the same digit (red box, 83 in this case) and $20 - |S^*|$ images with different digits.

Figure 4: Sampled data points for the CelebA dataset. Each row is a sample, consisting of $|S^*|$ anomalies (red box) and $8 - |S^*|$ normal images. In each sample, a normal image has two attributes (rightmost column) while anomalies do not have neither of them.

E.5 Detailed Experiment Settings for Set Anomaly Detection

In this experiment, we evaluate our methods on two real-world datasets:

Double MNIST: The dataset consists of 1000 images for each digit ranging from 00 to 99. For each sample (V, S^*), we randomly sample $n \in \{2, \dots, 5\}$ images with the same digit to construct the OS oracle S^*, and then select $20 - |S^*|$ images with different digits to construct the set $V \backslash S^*$. An example is shown in Fig. 3. **CelebA:** The CelebA dataset contains $202,599$ images with 40 attributes. As shown in Fig. 4, we select two attributes at random and construct the set with the size of 8. For each ground set V, we randomly select $n \in \{2, 3\}$ images as the OS oracle S^*, in which neither of the two attributes is present.

E.6 Detailed Experiment Settings for Compound Selection

Alg. 7 shows the corresponding data generation process of simulating the OS oracle for compound selection. In this algorithm, $\mathrm{random_choose}(\mathcal{C}, n)$ means randomly choosing n compounds from the database $\mathcal{C}$ (*i.e.,* PDBBind or BindingDB), and $\mathrm{topK_bioactivity}(V, m)$ represents selecting the top-m compounds with highest biological activity from the ground set V. These two operators combine together to form the bioactivity filter, in which we set (n, m) as $(30, 10)$, and $(300, 100)$ for PDBBind and BindingDB, respectively. To further apply the diversity filter, we use the RDKit[9] tools to compute the similarity between each molecule pair based on their topological fingerprints. This operator corresponds to the line 3 of Alg. 7, in which $\mathrm{cal_fingerprint_similarity}(S)$ returns the similarity matrix $\boldsymbol{M} \in \mathbb{R}^{|S| \times |S|}$ of the set of compounds S. Since rows (or columns) of the similarity matrix can be regarded as the features of the corresponding molecules, the molecules are clustered based on these similarity features by applying the affinity propagation algorithm. The OS oracle S^* is finally represented by the center of each cluster. Note that, each compound consists of two small molecules, *i.e.,* the protein-ligand molecules in PDBBind, and the drug-target molecules in BindingDB. We use the protein and drug molecules to compute the fingerprint similarity for PDBBind and BindingDB, respectively.

[9] https://github.com/rdkit/rdkit

Algorithm 7 OS Oracle Generation Algorithm for Compound Selection Task

Input: $\mathcal{C}$: compound database; n: size of ground set; m: number of the most active compounds
Output: Data point (V, S^*)

1: Randomly select n compounds to construct the ground set
$V \leftarrow \text{random_choose}(\mathcal{C}, n)$
2: Filter out m compounds with the highest bioactivity
$S \leftarrow \text{topK_bioactivity}(V, m)$
} **bioactivity filter**
3: Calculate the similarity matrix
$\boldsymbol{M} \leftarrow \text{cal_fingerprint_similarity}(S)$
4: Apply the affinity propagation algorithm
$\text{af} \leftarrow \text{affinity_propogation}(\boldsymbol{M})$
5: Assign the OS oracle as cluster centers
$S^* \leftarrow \text{af.cluster_centers_indices}$
} **diversity filter**

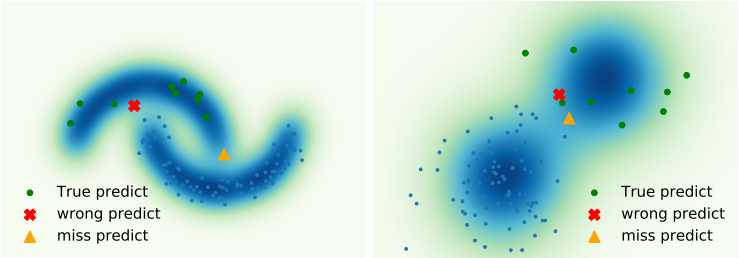

Figure 5: Visualization of the prediction of $\texttt{EquiVSet}_{\text{copula}}$ on the Two-Moons (left) and Gaussian-Mixture (right) datasets.

F Additional Experiments

F.1 Synthetic Experiments

We demonstrate the effectiveness of our models on learning set functions with two synthetic datasets: the two-moons dataset with additional noise of variance $\sigma^2 = 0.1$, and mixture of Gaussians $\frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}) + \frac{1}{2}\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma})$, with $\boldsymbol{\mu}_0 = [\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}]^T$, $\boldsymbol{\mu}_1 = -\boldsymbol{\mu}_0$, $\boldsymbol{\Sigma} = \frac{1}{4}\boldsymbol{I}_2$. Take the Gaussian mixture as an example, the data generation procedure $(S^*, V) \sim \mathbb{P}(S^*, V)$ is as follow: i) sample a component: $b \sim Bernoulli(\frac{1}{2})$; ii) sample 10 points from $\mathcal{N}(\boldsymbol{\mu}_b, \boldsymbol{\Sigma})$ to construct S^*; iii) sample 90 points for $V \backslash S^*$ from $\mathcal{N}(\boldsymbol{\mu}_{1-b}, \boldsymbol{\Sigma})$. We collect $1,000$ samples for training, validation, and test, respectively.

A qualitative result of the $\texttt{EquiVSet}_{\text{copula}}$ is shown in Fig. 5, where the green dots represent correct model predictions, the red crosses are incorrect model predictions, and the yellow triangles represent the data points in the subset oracle S^* that are missed by the model. One can see that the most confusing points are located at the intersection of two components. We also illustrate the quantitative results in Table 7. As expected, our methods provide significantly better performance over other methods, with averaged 59.16% and 100.07% improvements compared to PGM. on the Two-Moons and Gaussian-Mixture datasets, respectively.

F.2 Experiments on Varying Ground Set

Thanks to the virtues of DeepSet, our models are able to process input sets of variable sizes, which is termed as *varying ground set* property. To examine the impact of ground set sizes, we care about the following two questions: i) how well the model performs on different sizes of ground set during the test time; and ii) how well does the model train on ground sets of different sizes? To answer these two questions, we conduct experiments on the synthetic datasets using the proposed model $\texttt{EquiVSet}_{\texttt{copula}}$.

Set Size Transferability Analysis We first experiment to understand the pattern of set size transferability. In this experiment, we train the model using fixed sizes of the ground set but test the trained model on different sizes. We present two scenarios: train on a small size but test on a large one, and train on a large size but test on a small one. For the former one, we fix the size of OS oracle S^* to

Table 7: Results in the MJC metric on Two-Moons and Gaussian-Mixture datasets.

Method	Two Moons	Gaussian Mixture
Random	0.0551	0.0551
PGM	0.3646 ± 0.0154	0.4381 ± 0.0086
DeepSet (NoSetFn)	0.4717 ± 0.0031	0.4464 ± 0.0020
`DiffMF` (ours)	0.5842 ± 0.0012	0.9083 ± 0.0022
$\texttt{EquiVSet}_{\text{ind}}$ (ours)	0.5698 ± 0.0030	0.9074 ± 0.0024
$\texttt{EquiVSet}_{\text{copula}}$ (ours)	**0.5869 ± 0.0018**	**0.9085 ± 0.0015**

Table 8: Results in the MJC metric for compound selection with only the bioactivity filter.

Method	PDBBind	BindingDB
Random	0.0992	0.0092
PGM	0.9095 ± 0.0122	0.6878 ± 0.0181
DeepSet (NoSetFn)	0.9120 ± 0.0102	0.6754 ± 0.0119
`DiffMF` (ours)	0.9224 ± 0.0120	0.6943 ± 0.0160
$\texttt{EquiVSet}_{\text{ind}}$ (ours)	**0.9316 ± 0.0103**	0.6965 ± 0.0062
$\texttt{EquiVSet}_{\text{copula}}$ (ours)	0.9311 ± 0.0080	**0.6970 ± 0.0075**

be 10, and train the model with ground set V of size 100. After training, we test it using varying sizes of ground set in the range of $\{200, 400, 600, 800, 1000\}$. For the latter one, we fix the size of OS oracle S^* to be 10, and train the model with ground set V of size 1000. After training, we test it using varying sizes of ground set in the range of $\{100, 200, 400, 600, 800\}$. The former and latter experiments are conducted on the Two-Moons and Gaussian-Mixture datasets, respectively, with the results shown in Fig. 6. As can be seen, the performance would be slightly reduced if tested on a different size. Moreover, increasing the difference would enlarge the reduction.

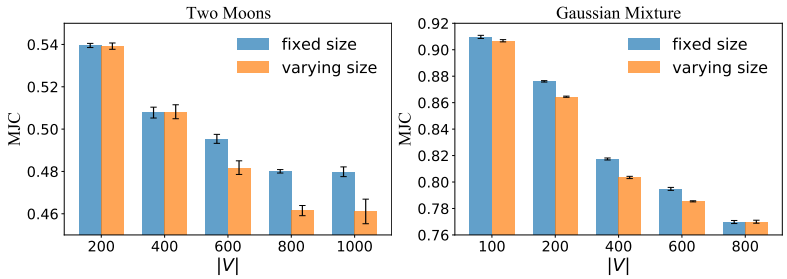

Figure 6: Synthetic results of $\texttt{EquiVSet}_{\texttt{copula}}$ for set size transferability analysis, in which the blue bars represent the performances of using the same sizes of ground set during the training and test time, while the yellow bars mean using different sizes of ground set during the test time. Detailed descriptions are given in the main text.

Selection Ratio Analysis To answer the second question, we fix the size of OS oracle S^* to be 10, and experiment with different selection ratios $\frac{|S^*|}{|V|}$ in the range of $\{0.5, 0.2, 0.1, 0.05, 0.01\}$. Unlike the set size transferability analysis, in this experiment, the selection ratios are the same during training and testing. Fig. 7 shows the performance of different ratios on two synthetic datasets. We observe that increasing the ratio would deteriorate the model performance. This phenomenon makes intuitive sense, since sampling subset from a large collection is more difficult. Moreover, the model performs worst when the ratio is equal to 0.5. This is partly because S^* and $V \backslash S^*$ are randomly sampled from one of two components. When $|S^*| = |V \backslash S^*|$, the model struggles to identify the optimal subset.

F.3 Experiments on Compound Selection with Only the Bioactivity Filter

To further evaluate the potential of `EquiVSet` for drug discovery, we consider an alternative setting here. In contrast to the task in Sec. 6, which aims at selecting the most active compounds while

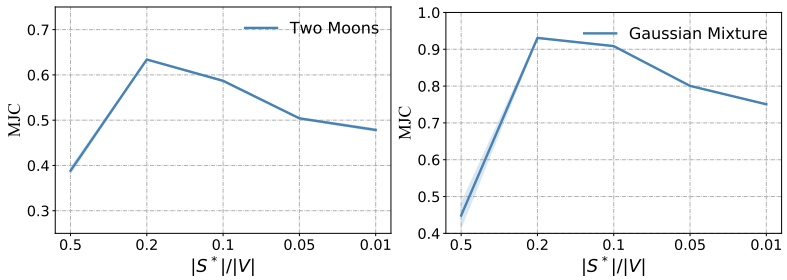

Figure 7: Synthetic results of $\texttt{EquiVSet}_{\text{copula}}$ with varying selection ratios.

preserving diversity, the task defined here only focuses on selecting the compounds with the highest bioactivity, which results a relatively simple selection process. The following is a detailed description.

PDBBind: To construct a data point (V, S^*), we randomly sample 30 complexes as the ground set V from the PDBBind database, and S^* is generated by the five most active complexes in V. Finally, we obtain the training, validation, and test set with the size of $1,000$, 100, 100, respectively. **BindingDB:** We construct the ground set V by randomly sampling 300 drug-targets from the BindingDB database and generate S^* with the 15 most active drug-target pairs. We finally obtain the training, validation, and test set with the size of $10,00$, $1,00$, and $1,00$, respectively.

Table 8 shows that our methods outperform the baselines. Meanwhile, the baselines also show satisfactory results. That is because identifying the most active compounds is a relatively simple task, especially for the PDBBind dataset with complex structures. More specifically, the model could predict the activity value of complexes precisely without considering the interactions between elements in the set, since the structure of complexes has provided sufficient information for this task. It is worth noting that the models in this task perform better than that in Sec. 6 partly because a one-layer filter (i.e. bioactivity) represents an easier way to replicate the OS oracle than a two-layer filter (i.e. bioactivity and diversity). Nevertheless, both experimental results in Sec. 6 and here demonstrate the effectiveness of `EquiVSet` for facilitating the complicated compound selection process.

F.4 Sensitivity Analysis of Hyperparameters

The proposed model $\texttt{EquiVSet}_{\text{copula}}$ has four important hyperparameters: the number of Monte Carlo sampling m and mean-field iteration step K in Alg. 1, and the rank of lower-rank perturbation v in (23). In this section, we discuss the impact of these hyperparameters through a sensitivity analysis on the Amazon product datasets.

Impact of the Mean Field Iteration Step Since iteration step K controls the convergence of mean-field iterative algorithms, this hyperparameter is highly relevant to the final performance of $\texttt{EquiVSet}_{\text{copula}}$. We experiment with different K on the Amazon product dataset. The results are shown in the first row of Fig. 8. We notice that increasing K would degenerate the model's performance. This seems to be embarrassingly surprising at first glance, since a large stride K encourages convergence with guarantee, resulting in a more robust training process. It is worth to be noted that in this method, we apply an amortized variational distribution to initialize the parameters for mean-field iterative algorithms. Since the amortized variational distribution is modeled with Gaussian copula, it can effectively capture the correlation among elements in the set, such that obtaining a better local optimal. However, if the iterative step K is large, the model inclines to converge to the local optimal that is the same as the original mean-field iteration. As a result, the benefit of correlation-aware inference provided by the Gaussian copula would be diminished. This explains why the iterative step K cannot be set too large.

Impact of the Number of MC Sampling The number of Monte Carlo (MC) sampling m plays an important role in the proposed method. It is widely known that increasing number of samples would reduce the variance of MC sampling. Therefore, using larger m would result in a better approximation of the gradient of multilinear extension $\nabla_\psi f_{\text{mt}}^{F_\theta}$ and thus better performance. This

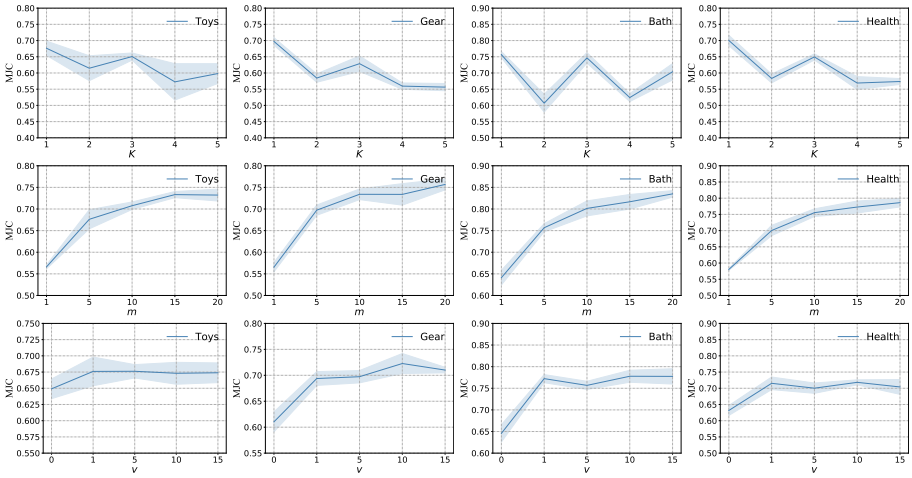

Figure 8: Sensitivity analysis of performance of $\texttt{EquiVSet}_{\text{copula}}$ under different hyperparameters (from top to bottom: the number of mean field iteration step K, the number of MC sampling m, the rank of lower-rank perturbation v, and the temperature of Gumbel-Softmax trick τ).

hypothesis is validated by the empirical results show in the second row of Fig. 8. It can be seen that as the sample number increases, the performance rises steadily at first and then gradually converges into a certain level. Undoubtedly, a large number would increase the computational complexity. In this regard, we uniformly set it as 5 in all experiments.

Impact of the Lower-rank Perturbation Lower-rank perturbed covariance matrix enables the proposed method to model the correlation information of elements in the set. To investigate its impacts, we evaluate the performance of $\texttt{EquiVSet}_{\text{copula}}$ under different values of rank v. The results are demonstrated in the third row of Fig. 8. Notably, the proposed model with $v = 0$ is equivalent to $\texttt{EquiVSet}_{\text{ind}}$. It can be seen that as the number of ranks increases, the performances also increase, indicating the hypothesis that employing the variational distribution with correlations can increase the model's representational capacity and thereby results in a better approximation in turn. It is worth noting that the most significant performance improvement is observed between the models with $v = 0$ and $v = 1$, and then as the value of v continues to increase, the improvement becomes relatively small. This indicates that it is feasible to set the v to a relatively small value to save computational resources while retaining competitive performance.