Neural Set Function Extensions: Learning with Discrete Functions in High Dimensions

NeurIPS 2022

Nikolaos Karalias* EPFL nikolaos.karalias@epfl.ch

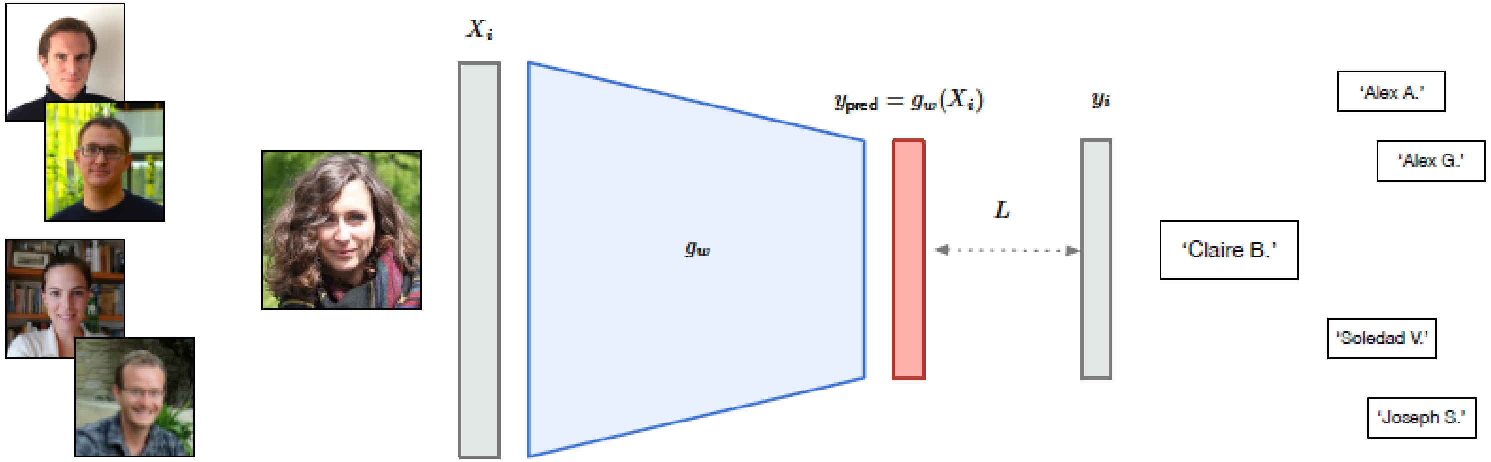
Joshua Robinson* MIT CSAIL joshrob@mit.edu

Andreas Loukas Prescient Design, Genentech, Roche andreas.loukas@roche.com

Stefanie Jegelka MIT CSAIL stefje@csail.mit.edu

[A lot of] Machine learning these days

Supervised learning: couples of inputs/responses (X_i, y_i), a model g_w



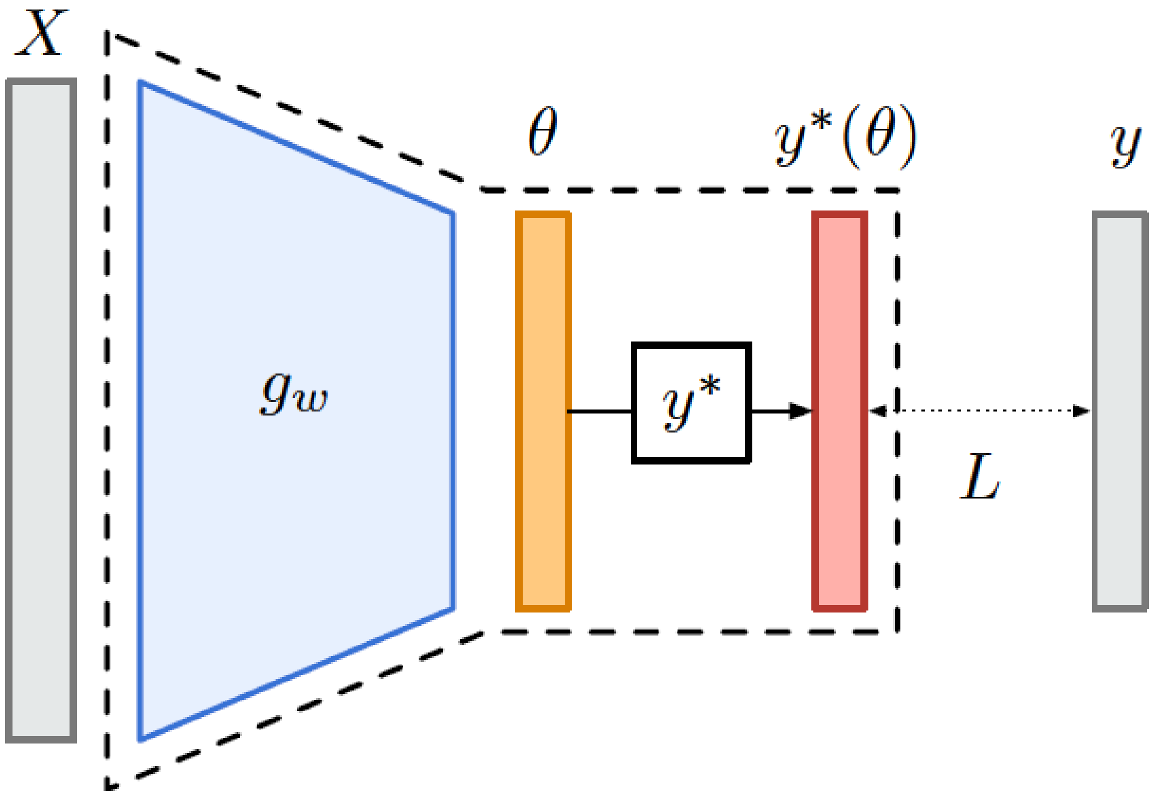
Goal: Optimize parameters $w \in \mathbf{R}^d$ of a function g_w such that $g_w(X_i) \approx y_i$

$$\min_w \sum_i L(g_w(X_i), y_i)\,.$$

Workhorse: first-order methods, based on $\nabla_w L(g_w(X_i), y_i)$, backpropagation

Problem: What if these models contain **nondifferentiable*** operations?

Discrete decisions in Machine learning



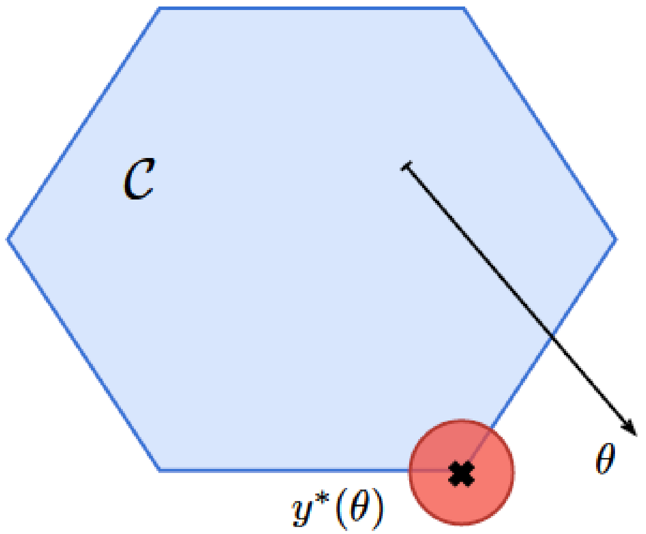
Examples: discrete operations (e.g. max, rankings), break autodifferentiation

- $\theta =$ scores for k products, $y^* =$ vector of ranks e.g. $[5, 2, 4, 3, 1]$
- $\theta =$ edge costs, $y^* =$ shortest path between two points

Perturbed maximizer

Discrete decisions: optimizers of linear program over $\mathcal{C}$, convex hull of $\mathcal{Y} \subseteq \mathbf{R}^d$

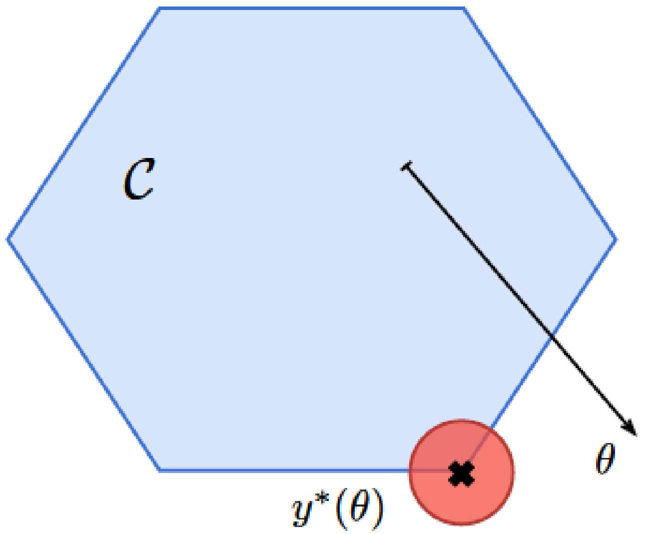
$$F(\theta) = \max_{y \in \mathcal{C}} \langle y, \theta \rangle \,, \quad \text{and} \quad y^*(\theta) = \operatorname*{argmax}_{y \in \mathcal{C}} \langle y, \theta \rangle = \nabla_\theta F(\theta)\,.$$

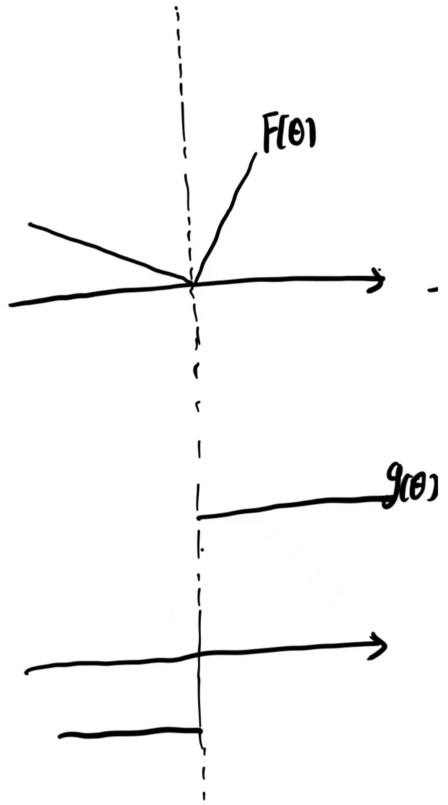


Perturbed maximizer

Discrete decisions: optimizers of linear program over $\mathcal{C}$, convex hull of $\mathcal{Y} \subseteq \mathbf{R}^d$

$$F(\theta) = \max_{y \in \mathcal{C}} \langle y, \theta \rangle \,, \quad \text{and} \quad y^*(\theta) = \operatorname{argmax}_{y \in \mathcal{C}} \langle y, \theta \rangle = \nabla_\theta F(\theta) \,.$$

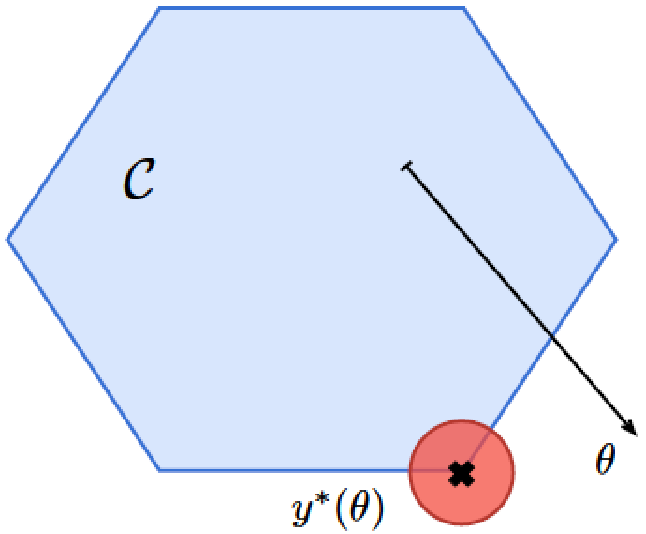


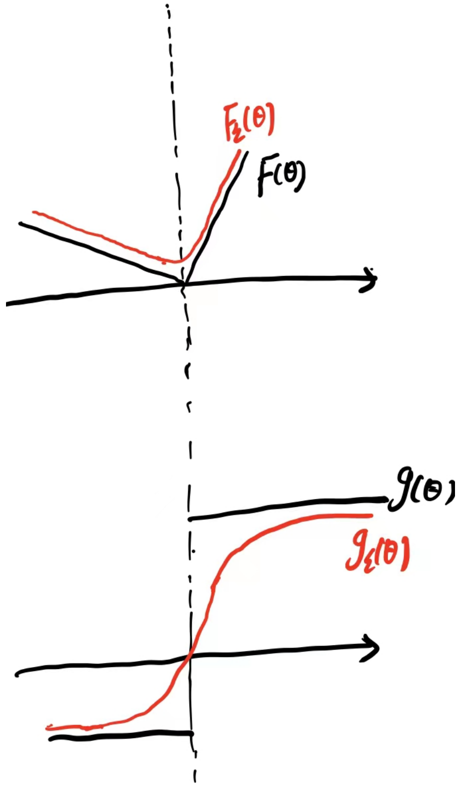


Perturbed maximizer

Discrete decisions: optimizers of linear program over $\mathcal{C}$, convex hull of $\mathcal{Y} \subseteq \mathbf{R}^d$

$$F(\theta) = \max_{y \in \mathcal{C}} \langle y, \theta \rangle\,, \quad \text{and} \quad y^*(\theta) = \operatorname*{argmax}_{y \in \mathcal{C}} \langle y, \theta \rangle = \nabla_\theta F(\theta)\,.$$





Berthet, Quentin, et al. "Learning with differentiable pertubed optimizers." *NIPS*(2020)

Problem Setup

a ground set $[n] = \{1, \ldots, n\}$

an arbitrary function $f : 2^{[n]} \to \mathbb{R} \cup \{\infty\}$

The discrete function breaks the backpropagation process.

an injective map $e : 2^{[n]} \to \mathcal{X}$ $\quad \mathfrak{F} : \mathcal{X} \to \mathbb{R}$

$\mathrm{NN}_2 \circ \mathfrak{F} \circ \mathrm{NN}_1$

automatic differentiation neural network framework

1. $\mathfrak{F}(e(S)) = f(S)$ for all $S \subseteq [n]$ with $f(S) < \infty$

Scalar Set Function Extensions-linear program

$$\widetilde{\mathfrak{F}}(\mathbf{x}) = \max_{\mathbf{z}, b \in \mathbb{R}^n \times \mathbb{R}} \{\mathbf{x}^\top \mathbf{z} + b\} \text{ subject to } \mathbf{1}_S^\top \mathbf{z} + b \leq f(S) \text{ for all } S \subseteq [n]. \qquad \text{(primal LP)}$$

$$\widetilde{\mathfrak{F}}(\mathbf{x}) = \min_{\{y_S \geq 0\}_{S \subseteq [n]}} \sum_{S \subseteq [n]} y_S f(S) \text{ subject to } \sum_{S \subseteq [n]} y_S \mathbf{1}_S = \mathbf{x}, \sum_{S \subseteq [n]} y_S = 1, \text{ for all } S \subseteq [n], \qquad \text{(dual LP)}$$

Definition (Scalar SFE). A scalar SFE $\mathfrak{F}$ of f is defined at a point $\mathbf{x} \in [0,1]^n$ by coefficients $p_{\mathbf{x}}(S)$ such that $y_S = p_{\mathbf{x}}(S)$ is a feasible solution to the dual LP. The extension value is given by

$$\mathfrak{F}(\mathbf{x}) \triangleq \sum_{S \subseteq [n]} p_{\mathbf{x}}(S) f(S)$$

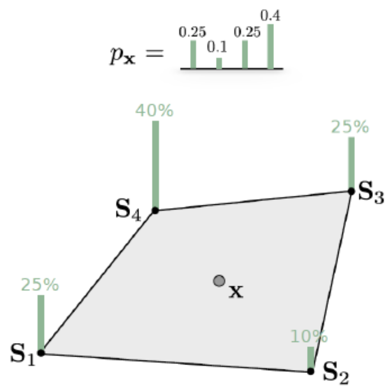
Scalar Set Function Extensions-linear program

Definition (Scalar SFE). A scalar SFE $\mathfrak{F}$ of f is defined at a point $\mathbf{x} \in [0,1]^n$ by coefficients $p_{\mathbf{x}}(S)$ such that $y_S = p_{\mathbf{x}}(S)$ is a feasible solution to the dual LP. The extension value is given by

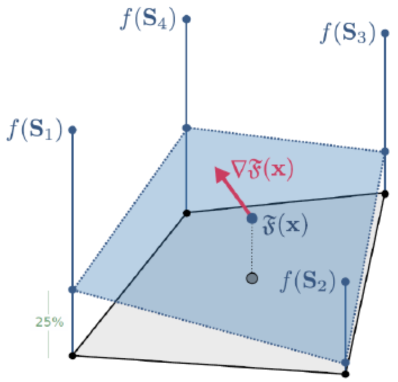
$$\mathfrak{F}(\mathrm{x}) \triangleq \sum_{S \subseteq [n]} p_{\mathbf{x}}(S) f(S)$$

Proposition 1 (Scalar SFEs have no bad minima). If $\mathfrak{F}$ is a scalar SFE of f then:

1. $\min_{\mathbf{x} \in \mathcal{X}} \mathfrak{F}(\mathbf{x}) = \min_{S \subseteq [n]} f(S)$
2. $\arg\min_{\mathbf{x} \in \mathcal{X}} \mathfrak{F}(\mathbf{x}) \subseteq \mathrm{Hull}\big(\arg\min_{\mathbf{1}_S : S \subseteq [n]} f(S)\big)$



$\mathbf{x} = \mathbb{E}_{\mathbf{S} \sim p_{\mathbf{x}}}[\mathbf{1}_{\mathbf{S}}]$



$\mathfrak{F}(\mathbf{x}) = \mathbb{E}_{\mathbf{S} \sim p_{\mathbf{x}}}[f(\mathbf{S})]$

Scalar Set Function Extensions-linear program

$$\mathfrak{F}(\mathbf{x}) \triangleq \sum_{S\subseteq[n]} p_{\mathbf{x}}(S) f(S) \longrightarrow p_{\mathbf{x}}(S) \longrightarrow$$

1) $p_{\mathbf{x}}(S)$ is a continuous function of $\mathbf{x}$

2) $\mathfrak{F}(\mathbf{1}_S) = f(S)$ for all $S \subseteq [n]$

$$\widetilde{\mathfrak{F}}(\mathbf{x}) = \min_{\{y_S \geq 0\}_{S\subseteq[n]}} \sum_{S\subseteq[n]} y_S f(S) \text{ subject to } \boxed{\sum_{S\subseteq[n]} y_S \mathbf{1}_S = \mathbf{x}, \ \sum_{S\subseteq[n]} y_S = 1,} \text{ for all } S \subseteq [n],$$

Scalar Set Function Extensions-linear program

Lovász extension. Re-indexing the coordinates of $\mathbf{x}$ so that $x_1 \geq x_2 \ldots \geq x_n$, we define $p_{\mathbf{x}}$ to be supported on the sets $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_n$ with $S_i = \{1, 2, \ldots, i\}$ for $i = 1, 2, \ldots, n$. The coefficient are defined as $y_{S_i} = p_{\mathbf{x}}(S_i) := x_i - x_{i+1}$ and $p_{\mathbf{x}}(S) = 0$ for all other sets.

Satisfy:

1. $\mathfrak{F}(\mathbf{1}_S) = f(S)$

2. $\sum_{i=1} p_{\mathbf{x}}(S_i) \cdot \mathbf{1}_{S_i} = \mathbf{x}.$

3. $\sum_{i=1}^{n} a_i = \sum_{i=1}^{n}(x_i - x_{i+1}) = x_1 \leq 1$

Scalar Set Function Extensions-linear program

Multilinear extension.

$$p_{\mathbf{x}}(S) = \prod_{i=1}^{n} x_i^{y_i}(1 - x_i)^{1-y_i}$$

Satisfy:

1. $\mathfrak{F}(\mathbf{1}_S) = f(S)$

2. $\sum_{S \subseteq [n]} p_{\mathbf{x}}(S) = 1$

3. $\sum_{S \subseteq [n]} p_{\mathbf{x}}(S) \cdot \mathbf{1}_S = \mathbf{x}$

Neural Set Function Extensions-semi-definite programming

$$\max_{\mathbf{Z}\succeq 0, b\in\mathbb{R}} \{\mathrm{Tr}(\mathbf{X}^\top \mathbf{Z}) + b\} \text{ subject to } \mathrm{Tr}((\mathbf{1}_S\mathbf{1}_T^\top + \mathbf{1}_T\mathbf{1}_S^\top)\mathbf{Z}) + 2b \leq 2f(S\cap T) \text{ for } S,T\subseteq [n]. \qquad \text{(primal SDP)}$$

$$\min_{\{y_{S,T}\}} \sum_{S,T\subseteq[n]} y_{S,T} f(S\cap T) \text{ subject to } \mathbf{X} \preceq \sum_{S,T\subseteq[n]} y_{S,T}(\mathbf{1}_S\mathbf{1}_T^\top + \mathbf{1}_T\mathbf{1}_S^\top) \text{ and } \sum_{S,T\subseteq[n]} y_{S,T} = 1 \qquad \text{(dual SDP)}$$

Definition (Neural SFE). A neural set function extension of f at a point $\mathbf{X} \in \mathbb{S}_+^n$ is defined as

$$\mathfrak{F}(\mathbf{X}) \triangleq \sum_{S,T\subseteq[n]} p_{\mathbf{X}}(S,T) f(S\cap T),$$

where $y_{S,T} = p_{\mathbf{X}}(S,T)$ is a feasible solution to the dual SDP and for all $S,T\subseteq[n]$: 1) $p_{\mathbf{X}}(S,T)$ is continuous at $\mathbf{X}$ and 2) it is valid, i.e., $\mathfrak{F}(\mathbf{1}_S\mathbf{1}_S^\top) = f(S)$ for all $S\subseteq[n]$.

Neural Set Function Extensions-semi-definite programming

Proposition 3. Let $p_{\mathbf{X}}$ induce a scalar SFE of f. For $\mathbf{X} \in \mathbb{S}_{+}^{n}$ with distinct eigenvalues, consider the eigendecomposition $\mathbf{X} = \sum_{i=1}^{n} \lambda_i \mathbf{x}_i \mathbf{x}_i^{\top}$ and fix

$$p_{\mathbf{X}}(S,T) = \sum_{i=1}^{n} \lambda_i \, p_{\mathbf{x}_i}(S) p_{\mathbf{x}_i}(T) \text{ for all } S, T \subseteq [n].$$

Then, $p_{\mathbf{X}}$ defines a neural SFE $\mathfrak{F}$ at $\mathbf{X}$.

Algorithms

Algorithm 1: Scalar set function extension

```
def ScalarSFE(setFunction, x):
    # x:  n x 1 tensor of embeddings, the output of a neural network
    # n:  number of items in ground set (e.g.  number of nodes in graph)
    setsScalar = getSupportSetsScalar(x) # n x n, i-th column is S_i.
    coeffsScalar = getCoeffsScalar(x) # 1 x n:  coefficients y_{S_i}.
    extension = (coeffsScalar*setFunction(setsScalar)).sum()
    return extension
```

Algorithm 2: Neural set function extension

```
def NeuralSFE(setFunction, X):
    # X: n x d tensor of embeddings, the output of a neural network
    # n:  number of items in ground set (e.g.  number of nodes in graph)
    # d:  embedding dimension
    X = normalize(X, dim=1)
    Gram = X @ X.T #  n x n
    eigenvalues, eigenvectors = powerMethod(Gram)
    extension = 0 # initialize variable
    for (eigval,eigvec) in zip(eigenvalues,eigenvectors):
        # Compute scalar extension data.
        setsScalar = getSupportSetsScalar(eigvec)
        coeffsScalar = getCoeffsScalar(eigvec)
        # Compute neural extension data from scalar extension data.
        setsNeural = getSupportSetsNeural(setsScalar)
        coeffsNeural = getCoeffsNeural(coeffsScalar)
        extension += eigval*((coeffsNeural*setFunction(setsNeural)).sum())
    return extension
```

Experiments-Unsupervised Neural Combinatorial Optimization

	Maximum Clique				
	ENZYMES	PROTEINS	IMDB-Binary	MUTAG	COLLAB
Straight-through (Bengio et al., 2013)	$0.725_{\pm 0.268}$	$0.722_{\pm 0.26}$	$0.917_{\pm 0.253}$	$0.965_{\pm 0.162}$	$0.856_{\pm 0.221}$
Erdős (Karalias & Loukas, 2020)	$0.883_{\pm 0.156}$	$0.905_{\pm 0.133}$	$0.936_{\pm 0.175}$	$1.000_{\pm 0.000}$	$0.852_{\pm 0.212}$
REINFORCE (Williams, 1992)	$0.751_{\pm 0.301}$	$0.725_{\pm 0.285}$	$0.881_{\pm 0.240}$	$1.000_{\pm 0.000}$	$0.781_{\pm 0.316}$
Lovász scalar SFE	$0.723_{\pm 0.272}$	$0.778_{\pm 0.270}$	$0.975_{\pm 0.125}$	$0.977_{\pm 0.125}$	$0.855_{\pm 0.225}$
Lovász neural SFE	$0.933_{\pm 0.148}$	$0.926_{\pm 0.165}$	$0.961_{\pm 0.143}$	$1.000_{\pm 0.000}$	$0.864_{\pm 0.205}$
	Maximum Independent Set				
	ENZYMES	PROTEINS	IMDB-Binary	MUTAG	COLLAB
Straight-through (Bengio et al., 2013)	$0.505_{\pm 0.244}$	$0.430_{\pm 0.252}$	$0.701_{\pm 0.252}$	$0.721_{\pm 0.257}$	$0.331_{\pm 0.260}$
Erdős (Karalias & Loukas, 2020)	$0.821_{\pm 0.124}$	$0.903_{\pm 0.114}$	$0.515_{\pm 0.310}$	$0.939_{\pm 0.069}$	$0.886_{\pm 0.198}$
REINFORCE (Williams, 1992)	$0.617_{\pm 0.214}$	$0.579_{\pm 0.340}$	$0.899_{\pm 0.275}$	$0.744_{\pm 0.121}$	$0.053_{\pm 0.164}$
Lovász scalar SFE	$0.311_{\pm 0.289}$	$0.462_{\pm 0.260}$	$0.716_{\pm 0.269}$	$0.737_{\pm 0.154}$	$0.302_{\pm 0.238}$
Lovász neural SFE	$0.775_{\pm 0.155}$	$0.729_{\pm 0.205}$	$0.679_{\pm 0.287}$	$0.854_{\pm 0.132}$	$0.392_{\pm 0.253}$

Table 1: **Unsupervised neural combinatorial optimization**: Approximation ratios for combinatorial problems. Values closer to 1 are better (↑). Neural SFEs are competitive with other methods, and consistently improve over vector SFEs.

Experiments-Constraint Satisfaction Problems

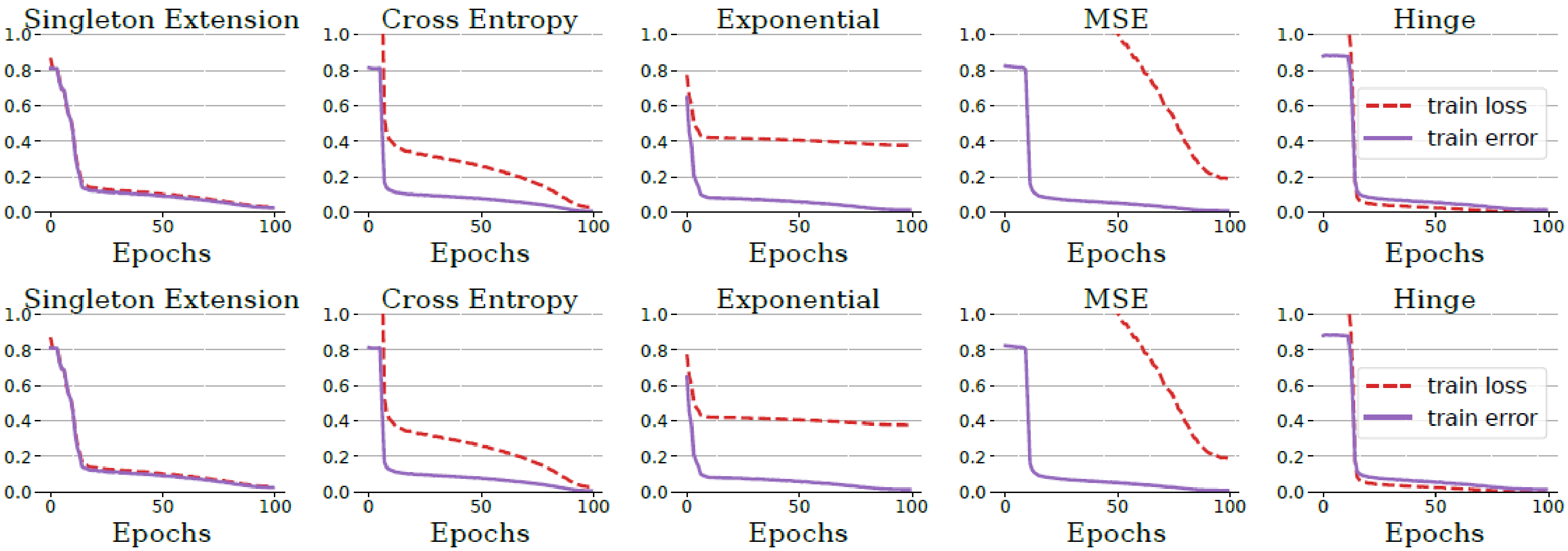


Figure 5: Top: CIFAR10. Bottom: SVHN. The singleton extension loss (left) is the only loss that approximates the true non-differentiable training error at the same numerical scale.

training error

$$\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\{y_i \neq h(x_i)\}.$$

$$\hat{y} \mapsto \mathbf{1}\{y_i \neq \hat{y}\}$$

Experiments-Ablations

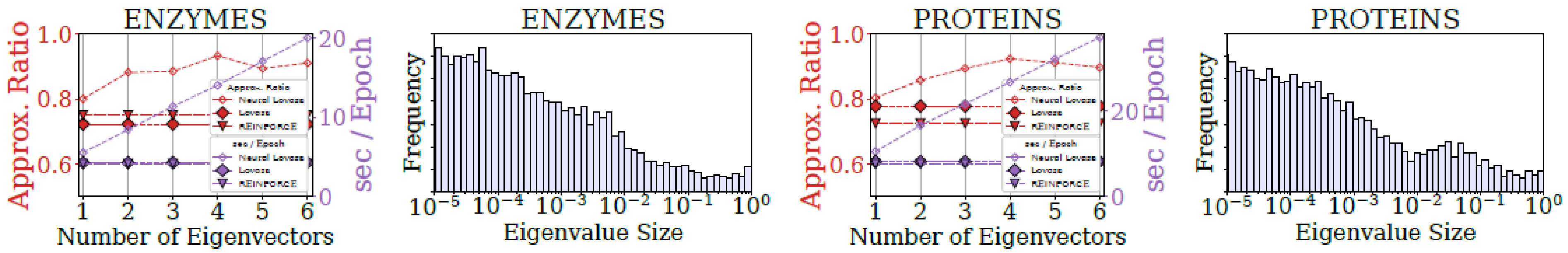


Figure 3: **Left:** Runtime and performance of neural SFEs on MaxClique using different numbers of eigenvectors. **Right:** Histogram of spectrum of matrix $\mathbf{X}$, outputted by a GNN trained on MaxClique.

Conclusion

1. This paper proposes a framework for extending functions on discrete sets to continuous domains.
2. The limit of n.