Copula for Instance-wise Feature Selection and Ranking

Abstract

Instance-wise feature selection and ranking methods can achieve a good selection of task-friendly features for each sample in the context of neural networks. However, existing approaches that assume feature subsets to be independent are imperfect when considering the dependency between features. To address this limitation, we propose to incorporate the Gaussian copula, a powerful mathematical technique for capturing correlations between variables, into the current feature selection framework with no additional changes needed. Experimental results on both synthetic and real datasets, in terms of performance comparison and interpretability, demonstrate that our method is capable of capturing meaningful correlations.

1 INTRODUCTION

The primary goal of feature selection is to select the most relevant features, and simultaneously help benefit the downstream tasks like recognition [Nilsson et al., 2007, Stańczyk and Jain, 2015], clustering [Dash and Liu, 2000, Liu and Yu, 2005, Witten and Tibshirani, 2010]. Traditional approaches view feature selection and downstream tasks as two separate entities. However, recent methods based on neural networks have achieved greater performance by combining the two, resulting in a differentiable optimization. Furthermore, the importance of features can vary from sample to sample; one set of features may be useful for recognizing one thing, while a completely different set of features may be necessary to identify something else. Therefore, it is necessary for feature selection to be done on a case-by-case basis, which is known as instance-wise feature selection [Yoon et al., 2019, Chen et al., 2018, Masoomi et al., 2020], it *selects unique features* for each sample.

Characteristics in reality are often correlated, such as a person's height and weight, economic status and wealth, etc. The identification of feature correlations can *minimize the redundancy of features*. Yet, in the literature of instance-wise feature selection and ranking methods [Chen et al., 2018, Yoon et al., 2019, Abid et al., 2019, Masoomi et al., 2020, Wu and Liu, 2018] that *follow the context of neural networks*, the dependencies between features has not been considered manifestly. For instance, L2X [Chen et al., 2018] performs a feature selection for maximizing the mutual information between selected feature subsets and corresponding outputs. Lowering the KL distance between the selected features and all features, is the guideline of INVASE [Yoon et al., 2019]. However, little or even no consideration in modeling correlations can pose a challenge to tasks.

To address this matter, we *underscore the importance of capturing the pairwise relationship between features*, and explicitly model it through a probabilistic framework. This paper examines both binary feature selection and top-k feature ranking. Specifically, for the top-k feature ranking problem, the number of features chosen is assumed to be known and unchanging. However, for binary feature selection, the number of features varies across samples, as different samples may have different feature significance. For both scenarios, two sampling policies are presented in the paper, along with a corresponding neural network implementation to unite feature selection and task learning.

Our contributions are summarized as follows:

- We explore the explicit dependence between features through copula, such as instance-wise feature selection and top-k ranking, as the initial step.
- Two sampling schemes, implemented via neural networks, have been carefully crafted to ensure accuracy and efficiency. Moreover, they have been rigorously tested and verified to guarantee their efficacy.
- The experimental results have been found to be highly indicative and of superior quality, as evidenced by metrics such as accuracy, true positive rate (TPR), and false

Submitted to the 39th Conference on Uncertainty in Artificial Intelligence (UAI 2023). **To be used for reviewing only**.

discovery rate (FDR). Indeed, these results demonstrate a remarkable explanatory power.

2 PRELIMINARIES AND RELATED WORK

Notation: Lowercase typeface letters (x) represent scalar, lowercase bold typeface letters ($\boldsymbol{x}$) represent vectors, uppercase typeface letters (X) stand for random variable, uppercase bold typeface letters ($\boldsymbol{X}$) represent matrix.

2.1 FEATURE SELECTION

In recent decades, feature selection has evolved from a specialized field to a commonplace technology in the machine learning community [Guyon and Elisseeff, 2003, Koller and Sahami, 1996, Liu and Motoda, 2007], and has been employed to simplify systems and make them more interpretable to researchers [Guyon and Aliferis, 2007, Brown et al., 2012]. Existing works on instance-wise feature selection can be divided into two main categories: binary feature selection [Chen et al., 2018, Yoon et al., 2019] and top-k feature ranking [Abid et al., 2019, Yamada et al., 2020]. The distinction between them is that the number of binary feature selection is dependent on the instance, while the number of top-k feature ranking is user-defined and known. Our work is akin to Yoon et al. [2019] and [Chen et al., 2018], in that all have a 3-stage network structure. Nevertheless, none of the aforementioned works have explicitly modeled the dependency; instead, we propose incorporating a Gaussian copula into binary feature selection and top-k feature ranking.

2.2 REMINDERS ON COPULA

Copula [Cherubini et al., 2004, Trivedi and Zimmer, 2007, Jaworski et al., 2010] is a powerful tool for describing the correlation between variables by modelling their joint distributions, given the known marginal distributions where each variable follows the uniform distribution $[0, 1]$. It can be combined with a variational auto-encoder (VAE) to address the issue of posterior collapse in the latent space [Wang and Wang, 2019]. Furthermore, Suh and Choi [2016] proposed to employ Gaussian copula to model the local dependency. Additionally, copula has been extensively applied in finance applications [Cherubini et al., 2004, 2011].

We briefly review the copula theory. For any continuous random variables, $Y_1, \ldots, Y_d$, let their marginal distributions be $F_1(y_1) = P(Y_1 \leq y_1), \ldots, F_d(y_d) = P(Y_d \leq y_d)$ correspondingly. Then it is easy to see that $U_i := F_i(Y_i)$ follows the uniform distribution on $[0, 1]$ for $i \in [d]$. And the copula of $U_1, \ldots, U_d$ is defined as the joint cumulative distribution function (C.D.F) of $(U_1, \ldots, U_d)$.

$$C(u_1, \ldots, u_d) := P(U_1 \leq u_1, \ldots, U_d \leq u_d), \;\; U_i = F(Y_i)$$

is a function from $[0, 1]^d$ to $[0, 1]$. We can rewrite the above formula as:

$$C(u_1, \ldots, u_d) := P(Y_1 \leq y_1, \ldots, Y_d \leq y_d),$$

One of the most helpful copula is known as the Gaussian copula, which is constructed from a multivariate normal distribution over $\mathbb{R}^d$ by using the probability integral transform. Its explicit formula is given as

$$C_{gaussian}(u_1, \ldots, u_d; \boldsymbol{R}) = \Phi_{\boldsymbol{R}}(\Phi^{-1}(u_1), \ldots, \Phi^{-1}(u_d)),$$

where $\Phi_{\boldsymbol{R}}(\cdot)$ is the joint cdf of multivariate normal distribution with mean zero and correlation/covariance matrix $\boldsymbol{R} \in \mathbb{R}^{d \times d}$, $\Phi^{-1}(\cdot)$ is the inverse of a set of marginal Gaussian cdf. Off-diagonal elements in $\boldsymbol{R}$ capture pairwise relation between different marginals.

3 METHOD

In this section, we shall commence by presenting the problem statement. Thereafter, we shall introduce two sampling schemes for binary feature selection and top-k feature ranking, respectively. Subsequently, we shall provide the overall algorithm workflow. Lastly, we shall discuss the details of implementing neural networks in greater detail.

3.1 PROBLEM FORMULATION

Given a collection of samples $\{(\boldsymbol{x}^n, y^n)\}_{n=1}^N$, where $\boldsymbol{x}^n = (x_1^n, \ldots, x_d^n)$ is a d-dimension feature vector, y^n is the corresponding output. Feature selection seeks to select a subset of features and simultaneously learn a task-specific objective function under certain loss metrics as follows:

$$\min_{\boldsymbol{\theta}} \frac{1}{N} \sum_{n=1}^{N} \mathcal{L}\left[\Phi\left(\boldsymbol{\theta}; \boldsymbol{x}^n \odot \boldsymbol{z}^n\right), y^n\right],$$
$$\text{subject to} \quad \boldsymbol{z}^n = \mathcal{S}\left(\boldsymbol{\alpha}^n\right); \boldsymbol{\alpha}^n = f\left(\boldsymbol{w}, \boldsymbol{x}^n\right),$$

where $\boldsymbol{\theta}, \boldsymbol{w}$ denote the learnable parameters, $\boldsymbol{\alpha}^n$ represents the learned score, $\odot$ represents the element-wise product, $\boldsymbol{z}^n = (z_1^n, \ldots, z_d^n) \in \{0, 1\}^d$ stands for the feature indicator, $\mathcal{L}$ is the objective function, $\mathcal{S}(\cdot)$ represents the sampling function, $f\left(\boldsymbol{w}, \cdot\right)$ and $\Phi\left(\boldsymbol{\theta}, \cdot\right)$ stand for the mapping function to learn $\boldsymbol{\alpha}^n$ and infer prediction, respectively. For top-k feature ranking, since the specific size of active features is established, we add an *auxiliary constraint* $\left\|\boldsymbol{z}^n\right\|_0 = k, \forall n = 1 \cdots N$ to loss function. In contrast, the number of binary feature selection is sample-dependent, the *regularization term* $\frac{\lambda}{N} \sum_{n=1}^{N} \left\|\boldsymbol{z}^n\right\|_1$ that controls selected features number, where λ is the trade-off parameter.

The above expression shows that the selection of features is dictated by the variable $\boldsymbol{z}^n$. Also on the other hand, $\boldsymbol{z}^n$ is depended on $\boldsymbol{\alpha}^n$. Thus the final performance is greatly affected by the way how we model $\boldsymbol{\alpha}^n$. Due to expressive

power and complexity of neural network, we naturally introduce neural network e.g. multilayer perceptron (MLP) to learn the mapping $\boldsymbol{\alpha}^n = f(\boldsymbol{w}, \boldsymbol{x}^n)$.

Despite we have known how to parameterize $\boldsymbol{\alpha}^n$, sampling function $\mathcal{S}(\cdot)$ usually is non-differentiable, as non-differentiable methods may have a large variance. It will hinder the usage of standard back-propagation algorithm and has large variance. Besides, $\mathcal{S}(\cdot)$ mostly assumes elements of $\boldsymbol{z}^n$ to be independent, like Gumbel-Softmax [Jang et al., 2017]. To address the first issue, we develop two sampling schemes as a continuous differentiable approximation to Bernoulli distribution and top-k ranking without replacement. To address the second issue, we incorporate the copula function to model the dependency between features explicitly. In what follows, *we drop the superscript n for simplicity*.

3.2 SAMPLING SCHEME VIA BINARY MASK

When the number of features is unknown and sample-dependent, for each $i \in \{1, \ldots, d\}$, we define a binary-valued random variable z_i which indicates whether the i-th feature should be included in $\mathcal{S}_a$. Specifically, z_i follows this distribution: $P(z_i = 1) = \frac{\exp\{\alpha_i\}}{1+\exp\{\alpha_i\}}$ and $p(z_i = 0) = \frac{1}{1+\exp\{\alpha_i\}}$. Here α_i's are obtained from weight layer.

By Gumbel-Max trick, we know that z_i and $\hat{z}_i$ have the same distribution, where $\hat{z}_i = 1\{g_i + \alpha_i > 0\}$ and $g_i = \log \frac{u_i}{1-u_i}$ follow a standard logistic distribution. u_i's follow the uniform distribution $[0, 1]$, and each element is independent to the other during the generation procedure in the classical Gumbel-Max algorithm. Also, as we can see, $\hat{z}_i$ is an indicator function and hence is not differentiable with respect to α_i. To circumvent this issue, we consider to replace $\hat{z}_i$ by its soft counterpart, that is, $\tilde{z}_i = \frac{1}{1+\exp\{-(g_i+\alpha_i)/t\}}$. Here t is a tuning parameter. When $t \to 0$, it is not hard to see that $\tilde{z}_i \to \hat{z}_i$.

In addition to characterizing the marginal distribution, we need to take into account the dependence among different z_i's. It suffices to model the joint distribution of g_i's. To do so, we incorporate a copula to accommodate its dependence structure. We write the joint C.D.F. of $(g_1, \ldots, g_d)$ as $G(x_1, \ldots, x_d) = P(g_1 \leq x_1, \ldots, g_d \leq x_d)$. We assume the following parameterized form, $G(x_1, \ldots, x_d) = C_{gaussian}(L(x_1), \ldots, L(x_d); \boldsymbol{R})$, where $C_{gaussian}$ is a Gaussian copula with $\boldsymbol{R}$ as its correlation matrix and $L(x) = \frac{1}{1+\exp\{-x\}}$. In a summary, $\tilde{\boldsymbol{z}} \sim G(x_1, \ldots, x_d)$ ($\tilde{\boldsymbol{z}} = (\tilde{z}_1, \ldots, \tilde{z}_d)$) is the output of the feature selection layer.

A natural question is how to guarantee the correlation matrix to be positive definite. We consider the following parameterization scheme. From the perspective of factor analysis [Harman, 1976, Kline, 2014], suppose a random vector $\boldsymbol{x} = (x_1, \ldots, x_d)$ lies in a lower-dimensional manifold:

$$\boldsymbol{x} = \boldsymbol{L}\boldsymbol{\xi} + \boldsymbol{\epsilon},$$

where $\boldsymbol{\xi}$ is a latent random p-vector and $\boldsymbol{L} = (\boldsymbol{l}_1, \ldots, \boldsymbol{l}_p)$ is the coefficient/loading matrix with size being $d \times p$. Suppose $\boldsymbol{\xi}$ and $\boldsymbol{\epsilon}$ are mutually uncorrelated, covariance matrix of $\boldsymbol{\xi}$ is an identity and the noise level of $\boldsymbol{\epsilon}$ is σ^2, then we know

$$\boldsymbol{\Sigma} = \boldsymbol{L}\boldsymbol{L}^T + \sigma^2 \boldsymbol{I}.$$

In other words, we can parameterize the correlation matrix as $\boldsymbol{R} = \text{Norm}(\boldsymbol{L}\boldsymbol{L}^T + \sigma^2 \boldsymbol{I})$, where function $\text{Norm}(\cdot)$ maps a covariance matrix to a correlation matrix and satisfies $(\text{Norm}(\boldsymbol{\Sigma}))_{ij} = \boldsymbol{\Sigma}_{ij}/(\boldsymbol{\Sigma}_{ii}\boldsymbol{\Sigma}_{jj})^{1/2}$. By adopting this parametrization, it's guaranteed that the correlation matrix is positive definite. Procedure to obtain correlated noise via Gaussian copula is summarized in Algorithm 1, algorithm flow for binary feature selection in Algorithm 2.

Algorithm 1: Generate Correlated Uniform Noise via Gaussian Copula

Input: Full-rank or low-rank matrix $\boldsymbol{L}$, identity matrix $\boldsymbol{I}$ and noise level σ^2.
Output: Correlated noise $\boldsymbol{u}$.

1. Obtain the covariance matrix via low-rank approximation $\boldsymbol{\Sigma} = \boldsymbol{L}^{\boldsymbol{L}+\sigma^2\boldsymbol{I}}$ or full-rank approximation $\boldsymbol{\Sigma} = \boldsymbol{L}^T\boldsymbol{L}$;
2. Perform Cholesky factorization on $\boldsymbol{\Sigma}$ to get Cholesky factor $\boldsymbol{V}$;
3. Generate a Gaussian noise vector $\boldsymbol{\zeta}$ from standard normal distribution $\boldsymbol{\zeta} \sim \mathcal{N}(0, \boldsymbol{I}_d)$;
4. Get the correlation matrix $\boldsymbol{R} = \text{Norm}(\sigma^2\boldsymbol{I} + \boldsymbol{L}\boldsymbol{L}^T)$;
5. Calculate the Gaussian vector $\boldsymbol{q} = \boldsymbol{V}\boldsymbol{\zeta}$;
6. Apply Gaussian copula to obtain $\boldsymbol{u}$ as $u_i = \Phi_{\boldsymbol{R}}(q_i), \forall i = 1, \ldots, d$;

Algorithm 2: Sampling Scheme via Binary Mask.

Input: Feature vector $\boldsymbol{x} \in \mathbb{R}^d$ with its corresponding weight $\boldsymbol{\alpha} \in \mathbb{R}^d$, full-rank or low-rank matrix $\boldsymbol{L}$ and noise level σ^2, identity matrix $\boldsymbol{I}$, tuning parameter t, round denotes the round operator.
Output: Binary mask vector $\boldsymbol{z} \in \{0, 1\}^d$.

1. Apply Algorithm 1 to obtain correlated noise vector $\boldsymbol{u}$;
2. Compute each element in the logit $\boldsymbol{g}$ as $g_i = \log \frac{u_i}{1-u_i}, \forall i = 1, \ldots, d$;
3. Calculate the probability $\tilde{z}_i = \frac{1}{1+\exp\{-(g_i+\alpha_i)/t\}}, \forall i = 1, \ldots, d$;
4. Obtain indicator vector $\boldsymbol{z}$ via discretizing the probability to binary variable $z_i = \text{round}(\tilde{z}_i), \forall i = 1, \ldots, d$;

3.3 SAMPLING SCHEME VIA TOP-k RANKING

When the number of features is known, without loss of generality, we let $k := |\mathcal{S}_a|$ ($k \leq d$). We aim to find the top-k features with the most predictive power. Suppose we have obtained the weights $\{\alpha_1, \alpha_2, \ldots \alpha_d\}$ from weight layer. A straightforward way to select the k features randomly proportional to its weights, which is also known as the weighted random sampling (WRS) [Xie and Ermon, 2019]. This procedure can be realized through the following ways: (i) For each $i \in [d]$, sample $u_i \sim U(0,1)$ independently and compute keys $v_i = u_i^{1/\alpha_i}$. (ii) Select k features with the largest keys v_i.

Firstly, we can observe that u_i are sampled independently in above procedure. We extend this by adding correlations between u_i's. That is, their joint distribution satisfies

$$F(u_1, \ldots, u_d) = C_{gaussian}(u_1, \ldots, u_d; \boldsymbol{R}), \tag{1}$$

where correlation matrix $\boldsymbol{R}$ should be learned through the network. Here, we assume $\boldsymbol{R}$ admits the structure $\text{Norm}(\boldsymbol{I} + \tau \boldsymbol{L}\boldsymbol{L}^T)$ with τ as a hyper parameter controlling the magnitude of correlations. *Observe to ourselves that certain equivalence of τ here and σ in* (1) *by giving* $\tau = \frac{1}{\sigma^2}$.

Secondly, suppose the top-k active feature set based on keys $\{v_1, \ldots, v_d\}$ is $\mathcal{S}_a = \{i_1, \ldots, i_k\}$. We then define the indicator $z_i = \mathbf{1}\{i \in \mathcal{S}_a\}$ to represent whether feature i is selected or not. By such construction, we know that z_i is not a differentiable function of α's. This may bring difficulty in computing the backward gradients. To avoid this problem, we consider a continuous deterministic relaxation to approximate z_i's. The specific procedure is described as follows. We define

$$p_i^s = \frac{\exp\{v_i^s/t\}}{\sum_{l=1}^d \exp\{v_l^s/t\}}$$

for $s \in \{1, \ldots, k\}$, where v_j^s is defined recursively by

$$\begin{aligned} v_i^s &= v_i^{s-1} + t^\delta \log(1 - p_i^{s-1}) \quad \text{for } s \in \{2, \ldots, k\}; \\ v_i^s &= v_i \quad \text{for } s = 1. \end{aligned}$$

Here t is a tuning parameter that determines the approximation level and $\delta \in [0, 1)$ is a hyper-parameter that adjusts the step size. We then define the relaxed value of z_i which is $\tilde{z}_i := \sum_{s=1}^k p_i^s$ and let $\tilde{\boldsymbol{z}} = (\tilde{z}_1, \ldots, \tilde{z}_d), \boldsymbol{z} = (z_1, \ldots, z_d)$. By writing into the vector form, we have

$$\tilde{\boldsymbol{z}} = \sum_{s=1}^k \boldsymbol{p}^s,$$

with $\boldsymbol{p}^s = (p_1^s, \ldots, p_d^s)$. Via using this relaxation scheme, our method contains the classical weighted random sampling as special case. The algorithm workflow for top-k ranking method is listed in Algorithm 3. On top of the above, we offer the following two theorems to shed light on role of σ in the convergence of the sampling scheme, with the proofs deferred to the Appendix.

Algorithm 3: Sampling Scheme via Top-k Ranking.

Input: Feature vector $\boldsymbol{x} \in \mathbb{R}^d$ with its corresponding weight $\boldsymbol{\alpha} \in \mathbb{R}^d$, full-rank or low-rank matrix $\boldsymbol{L}$ and noise level σ^2, identity matrix $\boldsymbol{I}$, tuning parameter t, δ.

Output: Binary mask vector $\boldsymbol{z} \in \{0, 1\}^d$.

Apply Algorithm 1 to obtain correlated noise vector $\boldsymbol{u}$;
Compute keys $v_i = u_i^{1/\alpha_i}, \forall i = 2, \ldots, d, v_1 = u_1$;
Take the log transformation $\log(\cdot)$ to v_i, and rewrite the key $v_i = (1/\alpha_i)\log(u_i)$;
for $s \leftarrow 2$ **to** k **do**
  $v_i^s = v_i^{s-1} + t^\delta \log(1 - p_i^{s-1}), \forall i = 1, \ldots, d$;
  $p_i^s = \frac{\exp\{v_i^s/t\}}{\sum_{l=1}^d \exp\{v_l^s/t\}}, \forall i = 1, \ldots, d$
end
$\tilde{\boldsymbol{z}} = \sum_{s=1}^k \boldsymbol{p}^s$;
Obtain feature indicator vector $\boldsymbol{z}$ via masking the top-k largest elements in $\tilde{\boldsymbol{z}}$ with 1, other elements with 0 ;

Theorem 3.1. *As both $t, \frac{1}{\sigma} \to 0$, we have Trunc$(\tilde{\boldsymbol{z}}, k) \rightsquigarrow \boldsymbol{z}^{wrs}$.*

Here, "$\rightsquigarrow$" represents the convergence in distribution. Function Trunc$(\boldsymbol{x}, k)$ chooses the indices of largest k elements in $\boldsymbol{x}$ provided the length of $\boldsymbol{x}$ is at least k. When $1/\sigma \to 0$, the proposed $\tilde{\boldsymbol{z}}$ converges to $\boldsymbol{z}^{wrs}$ which follows the weighted sampling distribution. On the other hand, if we take $\boldsymbol{L} = (1, \ldots, 1)^T \in \mathbb{R}^d$ and let $\sigma \to 0$, then the proposed method can also recover the situation top k sampling based on weights $\{\alpha_1, \ldots, \alpha_d\}$.

Proof of Theorem 3.1. Let $\check{\boldsymbol{z}}$ be Trunc$(\tilde{\boldsymbol{z}}, k)$ and it suffices to show that

$$\begin{aligned} &\lim_{t, \frac{1}{\sigma} \to 0} \mathbb{P}(\check{\boldsymbol{z}} = (i_1, \ldots, i_k)) \\ =&\frac{\alpha_{i_1}}{\sum_{i=1}^d \alpha_i} \cdot \ldots \cdot \frac{\alpha_{i_k}}{\sum_{i=1}^d \alpha_i - \sum_{j=1}^{k-1} \alpha_{i_j}}. \end{aligned} \tag{2}$$

By recursive formula of $\boldsymbol{p}^s$, it can directly verified that for any $\epsilon > 0$, there exists a constant t_0 such that

$$\begin{aligned} \tilde{z}_i &\geq 1 - \epsilon \qquad \text{for } i \in \mathcal{S}_{v,a}; \\ \tilde{z}_i &\leq \epsilon \qquad \text{for } i \notin \mathcal{S}_{v,a} \end{aligned} \tag{3}$$

holds when $t < t_0$. Here, $\mathcal{S}_{v,a}$ is defined to be the set of indices of k largest keys v_i. (Here $v_i := \log(u_i)/\alpha_i$ is called as the key for i-th feature.) In other words, Trunc$(\tilde{\boldsymbol{z}}, k)$ returns the indices of top k keys when $t \downarrow 0$.

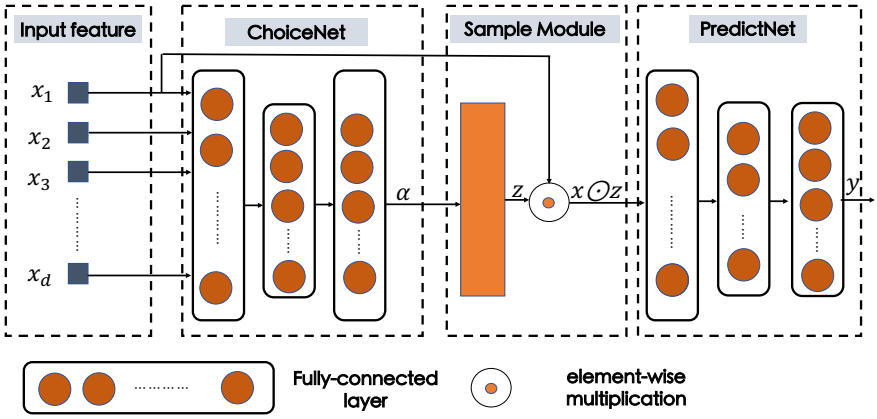

Figure 1: Our feature selection framework, it contains three main parts: ChoiceNet receives the input feature $x_1 \dots x_d$ and determines the score $\boldsymbol{\alpha}$, sampler module then receives $\boldsymbol{\alpha}$ and outputs the indicator variable $\boldsymbol{z}$; at last, the suppressed feature $\boldsymbol{x} \odot \boldsymbol{z}$ is fed into PredictNet for prediction.

On the other hand, when $1/\sigma \to 0$, we know that the correlation matrix $\boldsymbol{R}$ converges to $\boldsymbol{I}$. By Fubini's Theorem and following the proof strategy of Proposition 3 in Efraimidis and Spirakis [2006], we know that

$$\lim_{\sigma \to 0} \mathbb{P}(v_1 \leq \dots \leq v_d) = \prod_{i=1}^{d} \frac{\alpha_i}{\alpha_1 + \dots + \alpha_i} \tag{4}$$

Combining (3) and (4), it gives exactly (2). This completes the proof. □

Theorem 3.2. *When $\boldsymbol{L} = (1, \dots, 1)^T \in \mathbb{R}^d$, we have Trunc$(\tilde{\boldsymbol{z}}, k) \rightsquigarrow \boldsymbol{z}^{topk}$ as both t, $\sigma \to 0$ (the same as $\tau \to \infty$) where $\boldsymbol{z}^{topk} := (z_{i_1}, \dots, z_{i_k})$.*

Theorem 3.3. *When $\boldsymbol{L} = (1, \dots, 1)^T \in \mathbb{R}^d$, we have Trunc$(\tilde{\boldsymbol{z}}, k) \rightsquigarrow \boldsymbol{z}^{topk}$ as both $t, \sigma \to 0$ (the same as $\tau \to \infty$), where $\boldsymbol{z}^{topk} := (z_{i_1}, \dots, z_{i_k})$.*

Proof of Theorem 3.3. Similar to the proof of Theorem 3.1, we again know that Trunc$(\tilde{\boldsymbol{z}}, k)$ returns the indices of top k keys when $t \downarrow 0$. It remains to show that the order of v_i's is the same as the order of α_i's when $1/\sigma \to 0$.

We only need to show that the probability that, for any pair i, j, it holds

$$\lim_{\sigma \to 0} \mathbb{P}(v_i \leq v_j) = 0 \tag{5}$$

if $\alpha_i > \alpha_j$. By straightforward calculation, we get

$$\mathbb{P}(v_i \leq v_j) = \mathbb{P}(u_i^{1/\alpha_i} \leq u_j^{1/\alpha_j}) = \mathbb{P}(\frac{\log u_i}{\log u_j} > \frac{\alpha_i}{\alpha_j})$$

The right hand side goes to 0 as $\sigma \to 0$. This leads to the desired result. □

3.4 THE OVERALL ARCHITECTURE

We now turn to the detailed implementation via neural networks, which merges the deep neural networks and the proposed framework of feature selection.

As illustrated in Figure 1, our architecture is composed of three parts: (i) **ChoiceNet** (abbreviation of Choice Network), which models the mapping $\boldsymbol{\alpha} = f(\boldsymbol{w}; \boldsymbol{x})$ and is responsible for selecting features and outputting the learned score $\boldsymbol{\alpha}$ for each sample. (ii) **Sampler Module**, which models the sampling function $\mathcal{S}(\boldsymbol{\alpha})$ and receives the input $(\boldsymbol{\alpha})$ to output $\boldsymbol{z}$, where $z_i = 1$ indicates that the i-th feature is preserved or otherwise removed, and the selected feature can be expressed as $\boldsymbol{x} \odot \boldsymbol{z}$. (iii) **PredictNet** (abbreviation of Predictor Network), which receives the selected feature $\boldsymbol{x} \odot \boldsymbol{z}$ as input and outputs the corresponding prediction. For further details on the practical applications of this architecture in terms of neural networks, please refer to the Appendix.

3.4.1 Architecture of ChoiceNet and PredictNet

Both ChoiceNet and CriticNet are three-layer MLPs with ReLU [Brownlee, 2019] or SeLU [Klambauer et al., 2017] activations. For ChoiceNet, the size of the input layer is d, a fully connected layer with the dimension of h_c is added after the input layer, and the dimension of the final layer is also equivalent to d. PredictNet differs from ChoiceNet in that the number of units in the last layer is k, where k is the number of labels for classification tasks, and h_p is the dimension of the hidden layer. Additionally, a Softmax layer is added to the last layer for prediction, and Batch Normalization is added after the activation function to reduce overfitting.

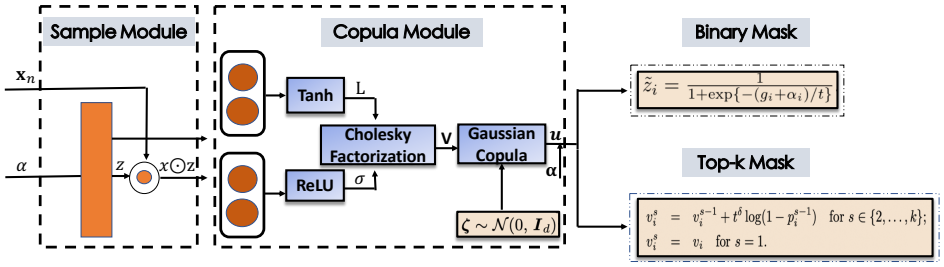

Figure 2: Sampler module $\mathcal{S}(\boldsymbol{\alpha})$. We make correlated uniform noise via Gaussian copula as Algorithm 1, then we apply the two sampling schemes as stated in Algorithm 2 and Algorithm 3.

3.4.2 Architecture of Sampler Module

Architecture of Binary Sampler Module It is not straightforward to implement a copula using a neural network, as the covariance matrix $\boldsymbol{\Sigma}$ must be guaranteed to be positive semi-definite (PSD). To this end, two extra fully connected layers are included to infer $\boldsymbol{L}$ and σ, with tanh and relu activation functions applied after the fully connected layer. After obtaining the covariance function, Cholesky decomposition can be used to obtain the factor matrix $\boldsymbol{V}$, ensuring that the covariance matrix is PSD. During the inference stage, a Bernoulli distribution is sampled for each sample. Algorithm 2 can be leveraged to generate the feature indicator vector, as illustrated in Figure 2.

Architecture of Top-k Sampler Module WRS generates the key v_i for the associated feature x_i as $v_i = u_i^{1/\alpha_i}$. By taking the log transformation, the key can be written as $\tilde{v}_i = (1/\alpha_i)\log(u_i)$, since the log transformation is monotonic, the induced construction of $\tilde{v}_i$ is still consistent with the original key v_i. For concrete realization, we propose to reparameterize the key as a deterministic mapping of the parameters $(1/\alpha_i)$ and $\log(u_i)$ via a neural network. Then, by leveraging the top-k relaxation in Algorithm 3, we can build a differentiable approximation with respect to the key v_i. The architecture of the top-k sampling scheme in the context of neural networks is similar to that of binary feature selection, except for the block representation. We also provide a demonstration of the block in Figure 2.

4 EXPERIMENTAL RESULTS

In this section, we empirically compare our algorithm with the advanced feature selection methods. For binary feature selection, we quantitatively compare our work with several powerful algorithms including methods based on neural networks like INVASE [Yoon et al., 2019], L2X [Chen et al., 2018], LIME [Ribeiro et al., 2016], Shap [Lundberg and Lee, 2017], Knockoff [Barber and Candès, 2015], and classical methods like Tree [Geurts et al., 2006], SCFS [Hall, 2000], LASSO [Tibshirani, 1996]. For top-k feature selection, we also validate our algorithm with several dominant approaches such as STG [Yamada et al., 2020], CAE [Abid et al., 2019], L2X [Chen et al., 2018], Shap [Lundberg and Lee, 2017], LASSO [Tibshirani, 1996], RF [Díaz-Uriarte and De Andres, 2006], boosting [Friedman et al., 2000]. In the comparison of classical methods, the advantages of neural network-based methods in an end-to-end fashion are highlighted. **It should be noted that, although these approaches vary in type, they all serve the purpose of feature selection, which is also widely employed in the preceding methods**. See Appendix for the details of implementation and baseline methods.

4.1 INSTANCE-WISE BINARY FEATURE SELECTION ON SYNTHETIC DATASETS

We begin our experiments with six challenging synthetic datasets as suggested in INVASE [Yoon et al., 2019]. $\boldsymbol{x}$'s are sampled from a multivariate Gaussian distribution with 11 dimensions, and the covariance matrix is an identity matrix. y is dependent on informative and relevant features in $\boldsymbol{x}$, and is set as a Bernoulli random variable that is 1 with probability $P(y=1|x) = \frac{1}{1+\exp(\gamma)}$. By varying γ, we can generate different y values across datasets.

- **Syn1:** $\gamma = x_1 x_2$
- **Syn2:** $\gamma = x_1^2 + x_2^2 + x_3^2 - 4$
- **Syn3:** $\gamma = -10 \times \sin(2x_7) + 2|x_8| + x_9 + \exp(-x_{10})$
- **Syn4:** if $x_{11} < 0$, $\gamma = x_1 x_2$, otherwise, $\gamma = x_3^2 + x_4^2 + x_5^2 + x_6^2 - 4$.
- **Syn5:** if $x_{11} < 0$, $\gamma = x_1 x_2$, otherwise, $\gamma = -10 \times \sin(2x_7) + 2|x_8| + x_9 + \exp(-x_{10})$.
- **Syn6:**if $x_{11} < 0$, $\gamma = x_3^2 + x_4^2 + x_5^2 + x_6^2 - 4$, otherwise, $\gamma = -10 \times \sin(2x_7) + 2|x_8| + x_9 + \exp(-x_{10})$.

We generate 10,000 samples for training and 10,000 samples for testing. The response y in the Syn1, Syn2, and Syn3 datasets is determined by the **identical** subset of features. In contrast, the number of relevant features **differs** across samples in the Syn4, Syn5, and Syn6 datasets; performance on these datasets can demonstrate the ability of methods to detect instance-wise and population-aware features. We evaluate the performance in terms of true positive rate (TPR) and false discovery rate (FDR). As shown in Table 1, our method achieves comparable or even better performance than the previous best method INVASE [Yoon et al., 2019]

Table 1: Experimental results on six synthetic dataset with dimension 11, we compare with a number of classical and advanced algorithms. Better results are marked in bold. Larger TPR and smaller FDR indicate better results. All the results are extracted from the INVASE method except ours.

Dataset	Syn1		Syn2		Syn3		Syn4		Syn5		Syn6	
Metric	TPR ↑	FDR ↓	TPR ↑	FDR ↓	TPR ↑	FDR ↓	TPR ↑	FDR ↓	TPR ↑	FDR ↓	TPR ↑	FDR↓
Ours	**100.0**	**0.0**	86.4	4.8	96.8	2.0	95.5	**2.0**	**89.3**	3.78	**93.8**	**6.6**
INVASE [Yoon et al., 2019]	**100.0**	**0.0**	**100.0**	**0.0**	92.0	**0.0**	**99.8**	10.3	84.8	**1.1**	90.1	7.4
L2X [Chen et al., 2018]	**100.0**	**0.0**	**100.0**	**0.0**	69.4	30.6	79.5	21.8	74.8	26.3	83.3	16.7
Shap [Lundberg and Lee, 2017]	60.4	39.6	93.3	6.7	90.9	9.1	65.2	31.9	62.9	33.7	71.2	28.8
LIME [Ribeiro et al., 2016]	13.8	86.2	**100.0**	**0.0**	98.1	1.9	40.7	49.4	41.1	50.6	50.5	49.5
Knockoff [Barber and Candès, 2015]	10.0	70.0	8.7	36.2	81.2	17.5	38.8	35.1	41.0	51.1	56.6	42.1
Tree [Geurts et al., 2006]	**100.0**	**0.0**	**100.0**	**0.0**	**100.0**	**0.0**	54.7	39.0	56.8	37.5	60.0	40.0
LASSO [Tibshirani, 1996]	19.0	81.0	39.8	60.2	78.3	21.7	49.9	50.9	45.5	48.2	56.4	43.6
SCFS [Hall, 2000]	23.5	76.5	39.5	60.5	78.3	22.0	48.9	52.4	42.4	51.2	56.1	43.9

Table 2: Experimental results on 100-dimension synthetic datasets. Better results are marked in bold. Larger TPR and smaller FDR indicate better results. All the results are extracted from the INVASE method except ours.

Dataset	Syn1		Syn2		Syn3		Syn4		Syn5		Syn6	
Metric	TPR ↑	FDR↓	TPR↑	FDR ↓	TPR↑	FDR ↓	TPR ↑	FDR↓	TPR ↑	FDR ↓	TPR ↑	FDR ↓
Ours	**100.0**	**0.0**	**100.0**	**0.0**	**100.0**	1.2	**94.5**	**40.3**	**88.4**	**9.4**	**99.5**	**14.6**
INVASE [Yoon et al., 2019]	**100.0**	**0.0**	**100.0**	**0.0**	**100.0**	**0.0**	66.3	40.5	73.2	23.7	90.5	15.4
L2X [Chen et al., 2018]	6.1	93.9	81.4	18.6	57.7	42.3	48.5	46.5	35.4	60.8	66.3	33.7
Shap [Lundberg and Lee, 2017]	4.4	85.6	95.1	4.9	88.8	11.2	50.2	43.4	49.9	44.2	62.5	37.5
LIME [Ribeiro et al., 2016]	0.0	100.0	**100.0**	**0.0**	92.7	7.3	43.8	47.4	49.9	44.2	50.1	49.9
Knockoff [Barber and Candès, 2015]	0.0	64.9	3.7	71.2	74.9	24.9	28.2	59.8	33.1	59.4	46.9	53.0
Tree [Geurts et al., 2006]	49.9	50.1	**100.0**	**0.0**	**100.0**	**0.0**	40.7	49.5	56.7	37.5	58.4	41.6
LASSO [Tibshirani, 1996]	2.5	97.5	4.0	96.0	75.3	24.7	28.3	73.2	36.0	56.9	45.9	54.1
SCFS [Hall, 2000]	2.5	97.5	5.3	94.7	74.9	25.1	27.0	74.6	30.6	62.1	38.3	61.7

Table 3: Experimental results on 100-dimension synthetic datasets with correlated features. Better results are marked in bold. Larger TPR and smaller FDR indicate better results.

Dataset	Syn1		Syn2		Syn3		Syn4		Syn5		Syn6	
Metric	TPR ↑	FDR↓	TPR↑	FDR↓	TPR ↑	FDR↓	TPR ↑	FDR↓	TPR ↑	FDR↓	TPR ↑	FDR↓
Ours	**100.0**	**0.0**	**100.0**	**0.0**	**100.0**	**0.0**	**91.95**	**41.6**	**90.9**	**42.0**	**90.1**	**43.9**
NOLA	74.8	18.3	96.4	16.7	99.7	78.4	48.5	84.0	66.8	49.3	50.0	46.2
INVASE [Yoon et al., 2019]	69.3	27.5	93.6	14.5	91.8	72.5	62.5	87.3	46.2	53.8	37.9	41.7

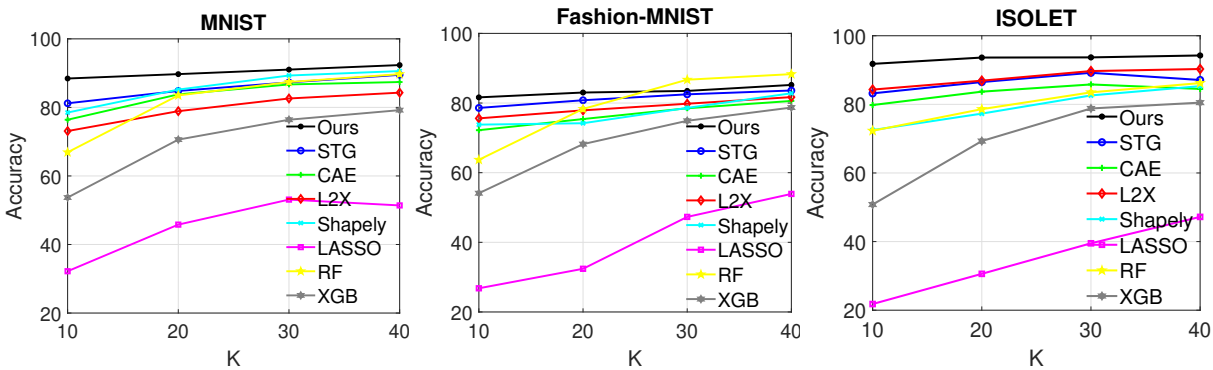

Figure 3: Prediction accuracy vs. the number of selected features k on three real datasets, the value of k varies from 10 to 40. We can discover that our method is superior to most methods.

on Syn1 and Syn3 datasets, demonstrating its capability in selecting relevant features for real data. Moreover, on Syn5 and Syn6 datasets, which are specifically designed for instance-wise feature selection, our algorithm outperforms INVASE [Yoon et al., 2019] in terms of instance-wise feature selection.

4.1.1 100-dimensional Synthetic Dataset

To further demonstrate the generality of our framework, we experimented on the 100-dimensional synthetic dataset, adding 89 auxiliary, unrelated features and increasing the feature dimension to 100, while still keeping the produced features uncorrelated. As illustrated in Table 2, our method remains the best-performing approach across most datasets.

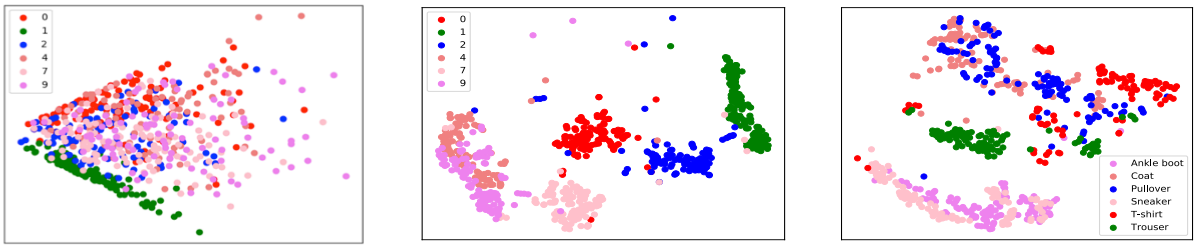

Figure 4: The left image shows the visualization of induced PSD matrix $\mathbf{\Sigma} \in \mathbb{R}^{784\times784}$ on MNIST dataset for binary feature selection, the middle image illustrates the $\mathbf{\Sigma}$ for top-k feature ranking on MNIST dataset, right image demonstrates the $\mathbf{\Sigma}$ for top-k feature ranking on Fashion-MNIST dataset.

4.1.2 Correlated Feature Selection

We use the same dataset in Section 4.1.1, but with a specially designed covariance matrix $\mathbf{\Sigma}_{i,j} = \frac{1}{2}^{|i-j|}$, where i and j denote the indices of features. To validate whether copula can capture the dependence between features, we compare our method with INVASE, the best competing baseline approach, and a variant of our method, NOLA (**N**o c**O**pu**LA**), where only copula is removed from our framework. The results in Table 3 demonstrate that copula performs better than both INVASE and NOLA.

4.2 FEATURE RANKING EXPERIMENTS ON REAL DATASETS

We now turn to instance-wise top-k feature ranking on real datasets, including MNIST [LeCun, 1998], Fashion MNIST [Xiao et al., 2017], and the ISOLET dataset [Cole et al., 1990]. For details of the datasets, ablation studies, and visualization results, please refer to the Appendices.

We measure performance by utilizing accuracy metrics and compare each method varying the number of selected features. Figure 3 shows the resulting prediction accuracy, which mostly outperforms other baselines, indicating that our method is a strong candidate in selecting top-k predictive features. The impact of copula and low-rank approximation on final performance is also investigated in the Appendix.

4.3 VISUALIZATION ON PSD MATRIX

To provide an intuitive illustration of the structure of the PSD matrix, we conducted an experiment on the MNIST dataset using copula to perform instance-wise feature selection. We then utilized t-SNE [van der Maaten and Hinton, 2008] on the induced PSD matrix to visualize the structure. The left image in Figure 4 provides a demonstration; we can observe that the features with label 0 and label 1 are well separable, since these labels are not similar. Additionally, features with labels 4, 7, and 9 are close to each other, due to their similar appearance. The visualization reflects that the PSD matrix indeed captures the intrinsic character of the feature.

As illustrated in the geometry structure of $\mathbf{\Sigma}$ on the MNIST dataset for binary feature selection, we provide another visualization on the PSD matrix $\mathbf{\Sigma}$ for top-k feature ranking. Following the same profile in Section 4.2, we set k to 40 while keeping all other experimental parameters unchanged. The middle image of Figure 4 shows a similar experimental phenomenon and leads to the same conclusion, indicating that the learned $\mathbf{\Sigma}$ can effectively capture the correlated relationship.

We also visualize $\mathbf{\Sigma}$ on the Fashion-MNIST dataset, as shown in the right image of Figure 4. We observe that the low-dimensional embeddings of Sneaker and Ankle boot are close, as they share similar characteristics. The same conclusion applies to Coat and Pullover, while the embeddings for Trouser are more concentrated and distant from the other image embeddings. This illustrates that copula can effectively capture the relationship between features. Further visualizations can be found in the Appendix.

5 CONCLUSION

In this paper, we have explored the potential of capturing the relationship between correlated features for binary feature selection and top-k feature ranking. To this end, we have successfully incorporated Gaussian copula into the existing feature selection framework with minimal modifications. Our proposed implementation via neural networks has yielded promising results, outperforming many classical and leading methods. We are confident that our work will inspire further research into more effective methods for mining the correlation between features. Possible future directions include capturing tail dependency in features with more sophisticated copulas, or the correlations can be generated via an implicit generative model [Janke et al., 2021].

References

Abubakar Abid, Muhammad Fatih Balin, and James Zou. Concrete autoencoders for differentiable feature selection and reconstruction. *arXiv preprint arXiv:1901.09346*, 2019.

Rina Foygel Barber and Emmanuel J Candès. Controlling the false discovery rate via knockoffs. *The Annals of Statistics*, 43(5):2055–2085, 2015.

Gavin Brown, Adam Craig Pocock, Ming-Jie Zhao, and Mikel Luján. Conditional likelihood maximisation: A unifying framework for information theoretic feature selection. *J. Mach. Learn. Res.*, 13:27–66, 2012.

Jason Brownlee. A gentle introduction to the rectified linear unit (relu). *Machine learning mastery*, 6, 2019.

Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. Learning to explain: An information-theoretic perspective on model interpretation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 882–891, Stockholmsmässan, Stockholm, Sweden, 2018.

Umberto Cherubini, Elisa Luciano, and Walter Vecchiato. *Copula Methods in Finance*. John Wiley & Sons, 2004.

Umberto Cherubini, Sabrina Mulinacci, Fabio Gobbi, and Silvia Romagnoli. *Dynamic Copula Methods in Finance*. John Wiley & Sons, 2011.

Ron Cole, Yeshwant Muthusamy, and Mark Fanty. The ISOLET spoken letter database, 1990.

Manoranjan Dash and Huan Liu. Feature selection for clustering. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications (PADKK)*, pages 110–121, Kyoto, Japan, 2000.

Ramón Díaz-Uriarte and Sara Alvarez De Andres. Gene selection and classification of microarray data using random forest. *BMC bioinformatics*, 7(1):1–13, 2006.

Pavlos S Efraimidis and Paul G Spirakis. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181–185, 2006.

Jerome H. Friedman, Trevor J. Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 28(2):337–407, 2000.

Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Mach. Learn.*, 63(1):3–42, 2006.

Isabelle Guyon and Constantin Aliferis. Causal feature selection. In *Computational Methods of Feature Selection*, pages 79–102. Chapman and Hall/CRC, 2007.

Isabelle Guyon and André Elisseeff. An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3 (Mar):1157–1182, 2003.

Mark A. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 359–366, Stanford, CA, 2000.

Harry H Harman. *Modern Factor Analysis*. University of Chicago press, 1976.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, Toulon, France, 2017.

Tim Janke, Mohamed Ghanmi, and Florian Steinke. Implicit generative copulas. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 26028–26039, Virtual, 2021.

Piotr Jaworski, Fabrizio Durante, Wolfgang Karl Hardle, and Tomasz Rychlik. *Copula Theory and its Applications*, volume 198. Springer, 2010.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 971–980, Long Beach, CA, 2017.

Paul Kline. *An Easy Guide to Factor Analysis*. Routledge, 2014.

Daphne Koller and Mehran Sahami. Toward optimal feature selection. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML)*, pages 284–292, Bari, Italy, 1996.

Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Huan Liu and Hiroshi Motoda. *Computational Methods of Feature Selection*. CRC Press, 2007.

Huan Liu and Lei Yu. Toward integrating feature selection algorithms for classification and clustering. *IEEE Trans. Knowl. Data Eng.*, 17(4):491–502, 2005.

Scott M. Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4765–4774, Long Beach, CA, 2017.

Aria Masoomi, Chieh Wu, Tingting Zhao, Zifeng Wang, Peter J. Castaldi, and Jennifer G. Dy. Instance-wise feature grouping. In *Advances in Neural Information Processing Systems (NeurIPS)*, virtual, 2020.

Roland Nilsson, José M. Peña, Johan Björkegren, and Jesper Tegnér. Consistent feature selection for pattern recognition in polynomial time. *J. Mach. Learn. Res.*, 8:589–612, 2007.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1135–1144, San Francisco, CA, 2016.

Urszula Stańczyk and Lakhmi C Jain. *Feature Selection for Data and Pattern Recognition*. Springer, 2015.

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

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Pravin K Trivedi and David M Zimmer. *Copula Modeling: an Introduction for Practitioners*. Now Publishers Inc, 2007.

Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *J. Mach. Learn. Res.*, 9(86):2579–2605, 2008.

Prince Zizhuang Wang and William Yang Wang. Neural gaussian copula for variational autoencoder. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 4332–4342, Hong Kong, China, 2019.

Daniela M Witten and Robert Tibshirani. A framework for feature selection in clustering. *Journal of the American Statistical Association*, 105(490):713–726, 2010.

Liang Wu and Huan Liu. Tracing fake-news footprints: Characterizing social media messages by how they propagate. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining (WSDM)*, pages 637–645, Marina Del Rey, CA, 2018.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Sang Michael Xie and Stefano Ermon. Reparameterizable subset sampling via continuous relaxations. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 3919–3925, Macao, China, 2019.

Yutaro Yamada, Ofir Lindenbaum, Sahand Negahban, and Yuval Kluger. Feature selection using stochastic gates. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 10648–10659, Virtual Event, 2020.

Jinsung Yoon, James Jordon, and Mihaela van der Schaar. INVASE: instance-wise variable selection using neural networks. In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.