A Survey of Task-driven Heterogeneous Feature Embedding and Selection

Hedong YAN

Ph.D. QE Survey

Hong Kong Baptist University

Department of Computer Science

A SURVEY OF TASK-DRIVEN HETEROGENEOUS FEATURE EMBEDDING AND SELECTION

Hedong YAN

Hong Kong Baptist University

ABSTRACT

As the complexity of real-world tasks and data sources continues to grow, the need for heterogeneous feature processing techniques becomes increasingly apparent. Traditional machine learning algorithms which is designed for homogeneous data can be degraded for heterogeneous data. How to handle heterogeneous features is a very important problem. Our research problem for handling heterogeneous data is divided into two sub-problem: heterogeneous feature embedding and heterogeneous feature selection. This survey explores the existing works for both the heterogeneous embedding algorithms and feature selection algorithms which is a very critical step in data analysis. The survey presents our methodologies and progress in addressing the identified problem, while outlining future plans.

1 BACKGROUND

1.1 INTRODUCTION

In many real world scenarios, the data can be collected from different sensors, devices, or platforms. Heterogeneous data can better reflects the complexity of real-world situations, and collectively provide a more complete and nuanced understanding of a situation. For example, in healthcare field, patient care can be enhanced by combining medical records, imaging data, and genetic information. In finance, making investment decisions may involve analyzing economic data, market sentiment from social media, and historical trends. Utilizing heterogeneous data provides a broader context for prediction and decision-making.

The first research problem that we identified is how to embedding the heterogeneous features for the given target of the task. For tabular data, the categorical feature can be nominal or ordinal and the continuous data can interval or ratio. Utilizing homogeneous models to handle the heterogeneous features can be harmful for the task performance. How to utilize the information inside the het-

erogeneous data is very important. One critical technique is the heterogeneous feature embedding. For instance, target statistic (TS) methods Micci-Barreca (2001) Zhang et al. (2013) Bottou & Cun (2003) He et al. (2014) Langford et al. (2009) Ling et al. (2017) use the expected value of the target variable to replace the original categorical feature or use target statistic as new features. One-hot encoder, rank-hot encoder, and ordinal encoder transform the categorical variables into continuous values. TabTransformer Huang et al. (2020) architecture uses a column embedding layer for categorical features and a layer normalization for continuous features. The input of transformer is only categorical features. Then the coded features will be concatenated for downstream MLP model. FT-Transformer Gorishniy et al. (2021) uses a embedding layer for categorical features and linear layer for continuous features. The embedded data was then processed by a transformer and final linear layer for output prediction. Piece-wise linear encoder and periodic encoder for continuous features introduced by Gorishniy et al. (2022) was used to handle the heterogeneous tabular data. The backbone model is MLP and hyper-parameter is tuned by Optuna. However, the structure information and global information inside the heterogeneous features is not well handled in their embedding works.

The second research problem is how to select heterogeneous features to improve the model's performance in the given task. Feature selection is a very important technique in machine learning. It is helpful to reduce model complexity and avoid over-fitting. It aims to keep the most relevant features and remove the irrelevant and redundant features. In this survey, we are interested in task-driven feature selection for data with heterogeneous structural features.

Structures feature selection is not a new technique. The existed structural feature selection algorithm can be divided into three classes based on the structures assumptions: group-based, tree-based, and graph-based. For example, Group LASSO Yuan & Lin (2006) select or not select a group of features as a whole. While Sparse Group LASSO Simon et al. (2013) introduces sparsity prior with intra-group and inter-group simultaneously. Tree-guided group LASSO Liu & Ye (2010) use tree to represent the structure of features while the leaf nodes are features and the internal node is associated with a weight which represents the height (correlation) of its subtree. Graph LASSO Ye & Liu (2012) add a positive adjacency matrix to measure the pair-wise dependency. GFLasso Kim & Xing (2009) also take the negative correlation into consideration. However, how to automatically infer the dependency and structures is still a problem.

In recent years, fuzzy rough set-based (FRS) methods for heterogeneous feature selection have attain more and more attentions. The difference between the FRS model and tradition models to handle the heterogeneous features is the determination of fuzzy relation matrix, which can calculate

the distance between two data points. Hu et al. (2006) lets fuzzy relation between two points is 1 when value of categorical attribute is the same and 0 otherwise. For continuous attributes, they use a norm to measure the distance between two points and a similarity function (which can map 0 to 1 and map ∞ to 0) to measure the relation of those two points from the calculated distance. Yuan et al. (2018), Yuan et al. (2021a), Yuan et al. (2021b) propose a implementation of the similarity function for continuous features. They use the clip negative linear function which return 0 when the absolute difference between the two values is larger than or equal to adaptive fuzzy radius ϵ_k. The clip linear function return 1 when the absolute difference is 0 for those points. Zhang et al. (2022) use a heterogeneous distance to measure the relationship among data points for heterogeneous features. The distance between continuous features is measured by the absolute value of difference.

In the following part, we will introduce the related works and our methodology and future plan to handle the heterogeneous problem.

2 Related works

2.1 Heterogeneous Feature Embedding

Recently, heterogeneous feature embedding algorithms become more and more important. In order to handle these heterogeneous features, various embedding techniques have been developed to map them into the real number field $\mathbb{R}$ or the interval $[0, 1]$. Existing embedding algorithms for features with heterogeneous scales can be broadly categorized into two classes: unsupervised and supervised (target-aware) methods.

Unsupervised Feature Encoder Unsupervised feature embedding approaches do not rely on label information during the embedding process. Instead, they typically utilize certain priors or assumptions, such as orthogonality, to guide the embedding process. These methods aim to discover inherent patterns or structures within the data itself, without considering the specific labels or target variable. By leveraging these priors, unsupervised embedding techniques can effectively capture the underlying characteristics of the features and represent them in a transformed space. The traditional unsupervised encoder for different type feature is listed in the table 1.

Nominal. One-hot encoder is a widely used embedding algorithm for categorical features. It represents each unique value of a feature as a binary number, indicating whether the corresponding value appeared or not. The resulting mapped vector for a feature has a size equal to the number of unique feature values.

Dummy encoding is similar to the one-hot encoder, but it reduces the vector size by one $(n - 1)$ by designating one value as the reference category and representing it with a zero vector. The other

Feature Scale	Encoder	Example
Nominal	One-hot	[1,2,3]→[[1,0,0],[0,1,0],[0,0,1]]
	Binary	$[1,2,3] \rightarrow [[0,0],[0,1],[1,0]]$
	Dummpy	$[1,2,3] \rightarrow [[1,0],[0,1],[0,0]]$
	Count	$[1,1,3] \rightarrow [[2],[2],[1]]$
	Simple	$[1,2,3] \rightarrow [[\frac{2}{3},-\frac{1}{3},-\frac{1}{3}],[-\frac{1}{3},\frac{2}{3},-\frac{1}{3}],[-\frac{1}{3},-\frac{1}{3},\frac{2}{3}]]$
Ordinal	Ordinal	$[1,2,3] \rightarrow [1,2,3]$
	Rank-hot	$[1,2,3] \rightarrow [[1,0,0],[1,1,0],[1,1,1]]$
	Gray	$[1,2,3] \rightarrow [[0,0],[0,1],[1,1]]$
Continuous	Bins+One-hot	$[0.11,0.22,0.31] \rightarrow [[1,0,0],[0,1,0],[0,0,1]]$
	Piece-wise linear (PLE)	$[0.11,0.22,0.31] \rightarrow [[0.1,0,0],[1,0.2,0],[1,1,0.1]]$
Other	Hashing	/

Table 1: Traditional unsupervised feature encoders.

values are then encoded using binary vectors. For example, a feature with three values would be represented as [0,0] for one value and [1,0] or [0,1] for the other two values.

Binary encoder maps the original feature values into a binary representation using a fixed number of binary bits. The number of bits required is determined by the formula $\lceil \log_2(n) \rceil$, where n is the number of unique feature values.

Frequency encoder, also known as Count encoder, maps each feature value to its frequency within the dataset. This encoding technique replaces the original value with its corresponding frequency, effectively representing the value by its occurrence count.

Simple encoder is similar to dummy encoding, but it replaces the binary values 0 and 1 with continuous values. Specifically, it substitutes 0 with $-\frac{1}{n}$ and 1 with $\frac{n-1}{n}$, where n represents the number of unique feature values. This encoding approach retains the ordinal information of the feature values.

Ordinal. Ordinal encoder is used to map a feature with ordinal scale into an integer value. It assigns a unique integer to each distinct value of the feature, considering the order or ranking among the values.

Rank-hot encoder, also known as thermometer encoder, is similar to one-hot encoding. However, instead of having only one value as hot (1) and the others as cold (0), it sets all values up to and including the current rank as hot. This encoding method captures the ordinal nature of the feature values.

Gray encoder, a type of binary encoder, ensures that adjacent values in the encoded representation differ by only a single bit. This helps in reducing errors or noise during the encoding process.

Continuous. The common practice is to use the continuous value of a feature directly as input for the backbone model. Another approach involves discretizing the feature values and applying categorical encoders.

One of discretization approach is the piece-wise linear (PLE) encoder, introduced by Gorishniy et al. (2022). The PLE encoder is inspired by the cumulative distribution function of a value. It first discretizes the continuous values of the feature and then applies the rank-hot encoder. Within each bin, the PLE encoder replaces the value $f(x)$ (which is initially set to 1) with a linear transformation $f(x) = \frac{x-b_{t-1}}{b_t-b_{t-1}}$, where b_{t-1} and b_t represent the lower and upper boundaries of the bin, respectively.

By discretizing the feature values and applying the rank-hot encoder with this modified transformation, the PLE encoder captures the relative position or rank of the values within each bin, enabling the model to learn and leverage this ordinal information during training.

Others. Base-N encoder is an encoding method that maps feature values into their base-N representation. In this encoding scheme, the base-N refers to the numerical base used for the representation, where base-1 corresponds to the one-hot encoder, base-2 corresponds to the binary encoder, and base-N corresponds to the ordinal encoder, with N being the number of unique values for the specific feature. This encoding approach leverages the inherent ordinality of the feature values by assigning them integer values based on their order or rank.

Hashing encoder is a technique that maps the original feature values into hash values. This encoding method involves applying a hash function to transform the values into a new representation. However, finding an appropriate hash function that yields good results for downstream models can be a non-trivial task. The effectiveness of the hashing encoder depends on the quality of the chosen hash function and its compatibility with the specific downstream models being used.

Supervised Feature Embedding Supervised embedding methods have the potential to enhance model performance by leveraging the supervised label information during the embedding process. These techniques incorporate the target variable or label into the embedding algorithm, allowing the model to learn informative representations that are directly aligned with the prediction task.

However, it is important to be cautious when applying supervised embedding, as it can inadvertently introduce target leakage. Target leakage occurs when the embedding process unintentionally incorporates information from the target variable that would not be available in a real-world prediction scenario. This leakage can lead to inflated performance during training but can severely degrade the model's generalization ability and performance on unseen data.

To mitigate target leakage and ensure reliable performance, careful consideration should be given to the design and implementation of supervised embedding methods. It is crucial to ensure that

the embedding process only utilizes information that would be available at the time of prediction, preventing any inadvertent incorporation of future or otherwise unavailable information. Thorough validation and evaluation on separate test sets can help detect and address any potential target leakage issues, allowing for more robust and reliable model performance.

Categorical feature. The greedy target statistic Micci-Barreca (2001) estimates the expected value of the target variable, denoted as $E(y|x = x_k)$, based on the training dataset. To mitigate potential noise or variability in the estimates, it is common to apply smoothing using parameters a and p Zhang et al. (2013). The smoothed estimate can be calculated using the following formula:

$$x_k^i = \frac{\sum_{j=1}^{n} \mathbb{I}_{x_j^i = x_k^i} * y_j + ap}{\sum_{j=1}^{n} \mathbb{I}_{x_j^i = x_k^i} + a} \tag{1}$$

Holdout target statistic Chen & Guestrin (2016) use the subset of instances excluding x_k,

$$x_k^i = \frac{\sum_{x_j \neq x_k} \mathbb{I}_{x_j^i = x_k^i} * y_j + ap}{\sum_{x_j \neq x_k} \mathbb{I}_{x_j^i = x_k^i} + a} \tag{2}$$

Ordered target statistic Prokhorenkova et al. (2018) creates a virtual 'time', and use its all available history to calculate the target statistic.

Continuous feature. The one-blob encoding method, proposed by Müller et al. (2019), assumes that the feature values follow a Gaussian or Laplace distribution. Each value is represented as a "blob" with a probability distribution centered at that value. The adjacent bins or intervals around the value correspond to the probabilities associated with that value. This encoding approach captures the uncertainty or variability in the feature values by modeling their distributions.

The periodic encoder, introduced by Gorishniy et al. (2022), maps a feature into its Fourier forms. The encoding is represented as a vector $f(x) = [\sin(v), \cos(v)]$, where $v = [2\pi c_1 x, .., 2\pi c_k x]$. The frequency vector c_i can be learned from the data. This encoding technique is particularly useful for handling periodic or cyclical features, where the relationship between values wraps around in a circular fashion (e.g., time of day or day of the week). By representing the feature values in their Fourier forms, the periodic encoder captures the underlying cyclical patterns and relationships within the data.

Deep learning In recent years, transformer-based heterogeneous feature embedding techniques for tabular data appeared, such as TabTransformer Huang et al. (2020), and FTTransformer Gorishniy et al. (2021).

As shown in the figure 1, TabTransformer Huang et al. (2020) architecture uses a column embedding layer for categorical features and a layer normalization for continuous features. The input of transformer is only categorical features. Then the coded features will be concatenated for downstream MLP model. As shown in the figure 2 3, FTTransformer Gorishniy et al. (2021) uses a embedding layer for categorical features and linear layer for continuous features. The embedded data was then processed by a transformer and a linear layer for output prediction.

Piece-wise linear encoder and periodic encoder for continuous features introduced by Gorishniy et al. (2022) was used to handle the heterogeneous tabular data. The backbone model is MLP and hyper-parameter is tuned by Optuna. The experiment result showed the tuned MLP model by Optuna with heterogeneous embedding module is better than transformers on the given tabular data.

However, the structure information and global information inside the heterogeneous features is not well handled in existing embedding works for heterogeneous data.

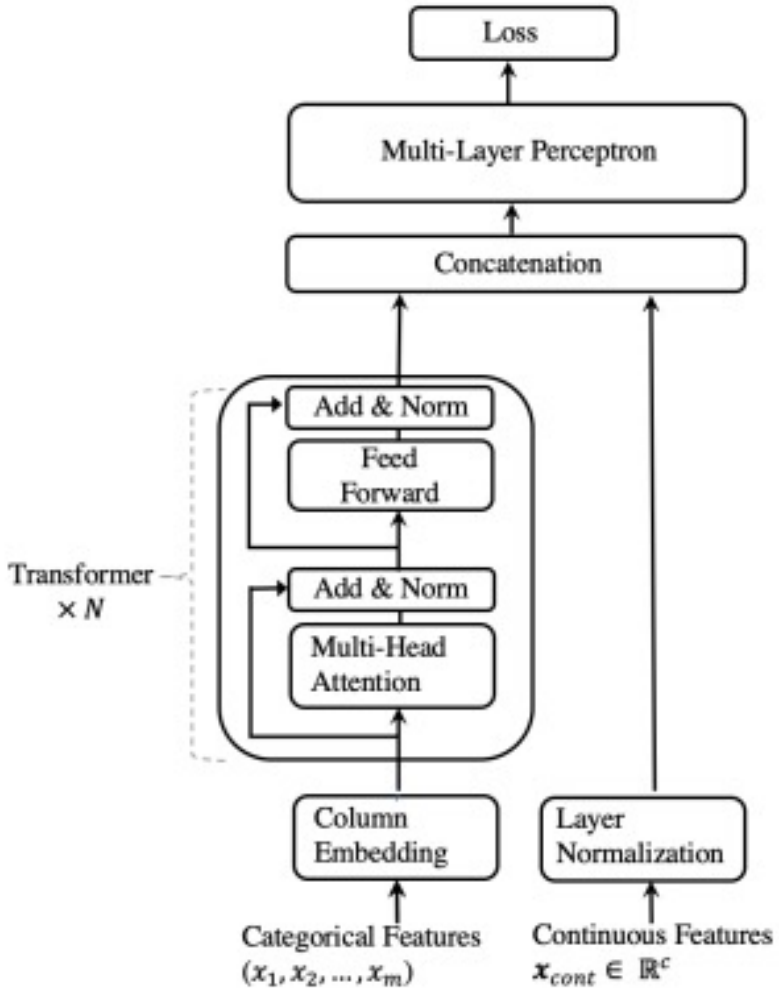


Figure 1: TabTransformer.

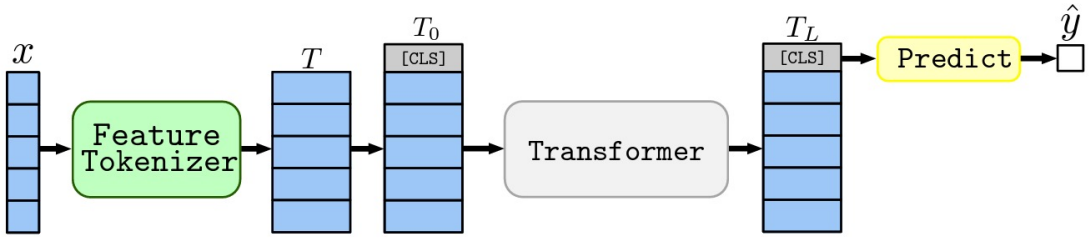


Figure 2: FTTRansformer.

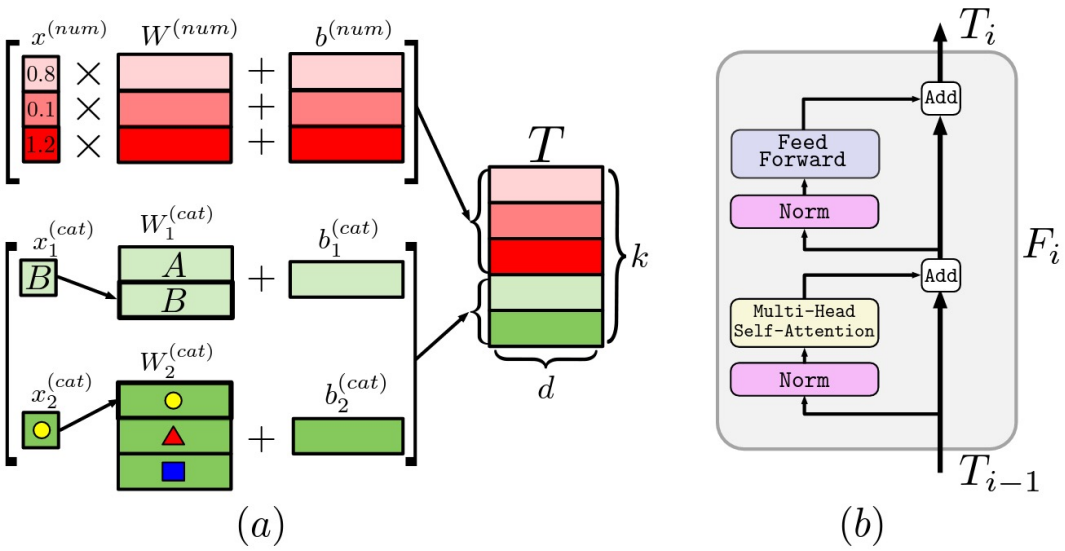


Figure 3: Feature Tokenizer of FTTransformer.

2.2 Heterogeneous feature selection

Feature selection is one of the most important critical technique in machine learning. In this section, we will introduce the structural feature selection, multi-view feature selection, and fuzzy rough set-based feature selection for heterogeneous data respectively.

2.2.1 Structural feature selection

LASSO Tibshirani (1996) is the basic feature selection algorithm with l_1-norm sparsity-induced penalty term, which can force some feature coefficients to be close to zero. Let $\mathbf{w}$ denotes feature coefficient, then the objective function of LASSO is:

$$min_w loss(w; X, y) + \alpha||w||_1 \tag{3}$$

where $loss(\cdot)$ is the loss function of the task, such as cross-entropy or least square loss. The LASSO approach usually assumes independence among features. Group LASSO Yuan & Lin (2006) treats

whole groups of features as a single entity, with the selection or exclusion unit being a group rather than an individual feature. The objective function of Group LASSO is:

$$min_w loss(w; X, y) + \alpha \sum_{i=1}^{g} h_i ||w_{G_i}||_2 \tag{4}$$

where h_i is the weight of the i-th group w_{G_i} which is given before optimization. In order to address the miss-selection problem in Group LASSO, Sparse Group LASSO Simon et al. (2013) add a l-1 penalty term to balance the intra-group selection and inter-group selection. The objective function of Group LASSO can be formulated as the following,

$$min_w loss(w; X, y) + \alpha ||w||_1 + (1 - \alpha) \sum_{i=1}^{g} h_i ||w_{G_i}||_2 \tag{5}$$

where α is a balancing parameter between zero and one. In Sparse Group LASSO, the features in the selected group can also be ignored. Tree-guided Group LASSO Liu & Ye (2010) is used to handle groups that can be represented by an index tree, where the leaf nodes represent features, and the internal nodes represent groups and subgroups. The objective function of Tree-guided Group LASSO is:

$$min_w loss(w; X, y) + \alpha \sum_{i=0}^{d} \sum_{j=1}^{n_i} h_j^i ||w_{G_j^i}||_2 \tag{6}$$

where α is a regularization parameter and h_j^i is the prior for group G_j^i. Graph LASSO Ye & Liu (2012) uses a undirected graph to represent the strong pairwise dependencies between features. The formulation of Graph LASSO is:

$$\min_w loss(w; X, y) + \alpha ||w||_1 + (1 - \alpha) \sum_{i,j} M(i, j)(w_i - w_j)^2 \tag{7}$$

where the second regularization term is to ensure the feature's dependencies. Features with a large $A_{i,j}$ will be similar to each others. GFLASSO Kim & Xing (2009) models the relationships for both positive and negative correlations. The objective function of GFLASSO is:

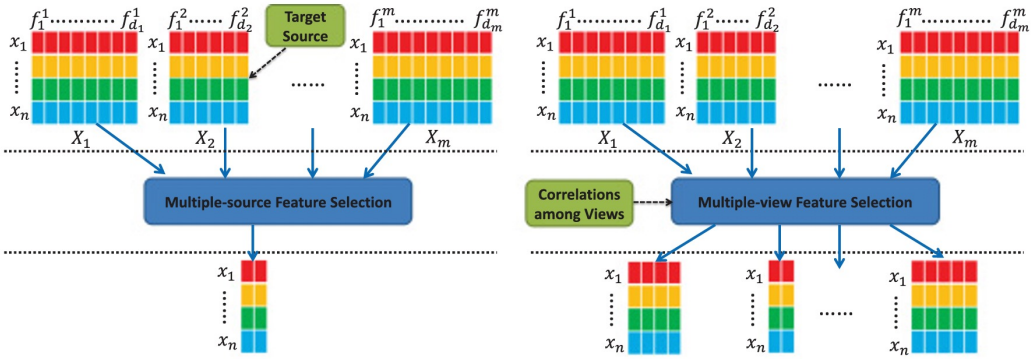
$$\min_w loss(w; X, y) + \alpha ||w||_1 + (1 - \alpha) \sum_{i,j} A(i, j)|w_i - sign(r_{i,j})w_j| \tag{8}$$

However, two challenges exist in existing works. First, learning the structures from data instead of relying on a prior for feature selection remains a problem. Second, for heterogeneous features, measuring the distance between them becomes more difficult, making learning the structures even more challenging.

2.2.2 MULTI-VIEW AND MULTI-SOURCE FEATURE SELECTION

In reality, the heterogeneous data from different source and different scales (such as nominal, ordinal) has been increased. For example, data from genes, electronic health records, clinical diagnosis, and images, have increased with the development of medical devices and computers science. How to fuse heterogeneous features together and find more efficient feature selection algorithm is still a problem. The current researches can be divided into two classes: traditional approaches and fuzzy rough set-based approaches.

The traditional heterogeneous feature selection approaches can be divided into two classes: multi-source feature selection and multi-view feature selection. The difference between multi-source and multi-view is illustrated in the figure 4.



(a) Multi-Source Feature Selection (b) Multi-View Feature Selection

Figure 4: The difference between multi-source feature selection and multi-view feature selection.

Multi-source feature selection is aim to select the most relevant features from the target source where each sources is constructed by many features. The difference between multi-source feature selection and traditional methods is that it utilize all heterogeneous features while only select features from one target source. For example, GPCOV algorithms use linear combination of local structures inside the sources as global pattern. In the feature selection step, Zhao & Liu (2008) selects features with highest variances greedily in the target source while d'Aspremont et al. (2004) takes the feature dependency into consideration and select features that retain maximum total variance.

Multi-view feature selection is aim to select the most relevant features from multi-sources simultaneously. For example, the pixels, the tabular terms, and text terms that is associated with the instance. For supervised task, Sparse Group is often used where the different view is regarded as different groups. For unsupervised task, the loss function is build by a pseudo class label which is learned from spectral clustering prior in m views. AUMFS's Feng et al. (2013) objective function is

as follows,

$$\begin{aligned} &min\ tr(F' \sum_{i=1}^{m} \lambda_i L_i F) + \beta(||XW - F||_{2,1} + \alpha||W||_{2,1}) \\ &s.t. F'F = I_c, F \geq 0, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i > 0 \end{aligned} \tag{9}$$

where λ_i is used to balancing the contribution of different hidden spectral clusters and F is the pseudo class label. Compared with AUMFS, MVFS Tang et al. (2013) learns a weight matrix for each view. The objective function of MVFS is as follows,

$$\begin{aligned} &min\ tr(F' \sum_{i=1}^{m} \lambda_i L_i F) + \sum_{i=1}^{m} \beta(||XW - F||_{2,1} + \alpha||W||_{2,1}) \\ &s.t. F'F = I_c, F \geq 0, \sum_{i=1}^{m} \lambda_i = 1, \lambda_i > 0 \end{aligned} \tag{10}$$

Although the input structure is heterogeneous, those feature selection methods requires continuous homogeneous inputs. The transformation of data scales can be harmful to models' performance. For example, there are usually two methods to convert the features into homogeneous data. First, using discretization methods to transform the continuous features into discrete nominal features. Second, using the integer number to replace the discrete feature. However, discretization methods will lead to information loss, such as ordering of numerical value, and geometry on the real space; replace the nominal features by integers can introduce meaningless structures, such as ordering.

2.2.3 FUZZY ROUGH SET-BASED HETEROGENEOUS FEATURE SELECTION

Recently, fuzzy rough set (FRS) model have attained more and more attention in the feature selection field. It can be applied to the heterogeneous features and overcome the information loss problem of discretization methods. The fuzzy similarity relation is defined to measure the similarity between data objects. The key of fuzzy rough set-based approaches is to select suitable similarity matrix.

The fuzzy uncertainty function is defined as following for heterogeneous features Hu et al. (2006),

$$r_{ij}^{k} = \begin{cases} 1, \text{if} f(x_i, a) = f(x_j, a) \text{ and A is discrete}, \forall a \in A \\ 0, \text{if} f(x_i, a) \neq f(x_j, a) \text{ and A is discrete}, \forall a \in A \\ f(||x_i - x_j||), \text{if A is continuous} \end{cases} \tag{11}$$

where f is a similarity function which satisfies

$$f(0) = 1,\ f(\infty) = 0, f(*) \in [0, 1] \tag{12}$$

Wang et al. (2019) introduces a similarity measure for categorical features, which is defined as following,

$$r_{ij}^{B} = \frac{1}{|A|} card(k \in B : c_k(x_i) = c_k(x_j)) \tag{13}$$

where c_k is the measured value of feature k and $|A|$ is the feature number.

In Yuan et al. (2018),Yuan et al. (2021a), Yuan et al. (2021b), the fuzzy similarity degree between x_i and x_j for feature c_k is defineed as following,

$$r_{ij}^{k} = \begin{cases} 1, \text{if} c_k(x_i) = c_k(x_j) \text{ and } c_k \text{is discrete} \\ 0, \text{if} c_k(x_i) \neq c_k(x_j) \text{ and} c_k \text{is discrete} \\ 1 - |c_k(x_i) - c_k(x_j)|, \text{if} |c_k(x_i) - c_k(x_j)| \leq \epsilon_{c_k} \text{ and } c_k \text{ is continuous} \\ 0, \text{if} |c_k(x_i) - c_k(x_j)| > \epsilon_{c_k} \text{ and } c_k \text{ is continuous} \end{cases} \tag{14}$$

where c_k is the measured value of data point x for feature c_k and ϵ_{c_k} a adaptive fuzzy radius. The ϵ_{c_k} is calculated as following,

$$\epsilon_{c_k} = \frac{std(c_k)}{\lambda} \tag{15}$$

where $std(c_k)$ is standard deviation of the feature values c_k and λ is a hyper-parameter that is fine-tuned with step 0.1 in the range [0.1,2.0].

In Zhang et al. (2022), the heterogeneous distance between two data points, which is contrast to similarity, is defined as following,

$$d_{ij}^{k} = \begin{cases} 0, \text{if} f(x_i, a) = f(x_j, a) \text{ and} a \text{is discrete} \\ 1, \text{if} f(x_i, a) \neq f(x_j, a) \text{ and } a \text{is discrete} \\ |f(x_i, a) - f(x_j, a)|, \text{if } a \text{ is continuous} \end{cases} \tag{16}$$

2.2.4 SELECTION POLICY

In Yuan et al. (2021b), Zhang et al. (2022), the features are sorted by sequence and the features are selected by a greedy algorithms when the selected number K is given and the K is determined by exhaustive methods.

3 METHODOLOGY FOR HETEROGENEOUS EMBEDDING

3.1 NOTATION AND PROBLEM

Given dataset $D = \{x_i\}_{i=1}^n$ with m heterogeneous features $\mathcal{F} = \{f_1, ..f_m\}$, we aim to learn a low-dimensional representation h_i for each instance, such the effective information in original space X can be preserved in the united space H. The input of the heterogeneous embedding modules is features x_i of data points i and the output is hidden representation h_i of this data point i. The preserved effective information of the embedding module is evaluated by the task performance in real world, such as classification for supervised task and clustering for unsupervised task.

3.2 HETEROGENEOUS FEATURE EMBEDDING

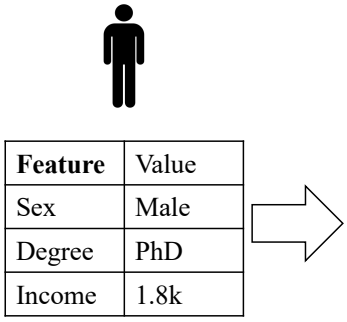
For any heterogeneous features, a key characteristic shared by those features is the occurrence probability of features' measurements. We introduce the occurrence space to handle the heterogeneous problem.

Definition 1 *Occurrence Space H. Every data point h in the occurrence space H represent the probability of the current measurement x_i where axis is denoted by a feature's specific value. The occurrence space's dimension is $\sum_{i=1}^n |f_i|$ where $|f_i|$ is the potential assignments number of the feature f_i. The range of occurrence space is* $[0, 1]$.

Regardless of the feature scales (nominal, ordinal, interval, or ratio) and feature sources (such as text, images, tabular data), these occurrences hold significant meaning for the instance. For example, the occurrence of male for an individual, or the occurrence of face of an individual is in the same hidden space.

The subsequent aspect involves the integration of information from the occurrence space and the original value representations. Nominal features are encoded using a one-hot encoder, facilitating the capture of occurrence probabilities. For ordinal features, we employ a rank-hot encoder, as the occurrence of higher rank values implies the occurrence of lower rank values. In order to apply to continuous features, we transform the continuous features into nominal features by Fourier transformation with learnable parameters or ordinal features by linear-piece-wise encoder. The Fourier-based form enables the modeling of occurrence patterns across different frequencies.

Figure 5 is an example of the representation in occurrence space. There are three feature in the original feature space: sex, educational background, income per month. Then the occurrence space is [male, female, high school, Bachelor, PhD, 0-10K, 10k-20k, 20-30k, >30k]. The patient A (male, PhD, 1.8k) in the original space is encoded as [male: 1, female: 0, high school: 1, Bachelor: 1, PhD:



Feature	Value
Sex	Male
Degree	PhD
Income	1.8k

Occurrence	Observation	Frequency
Male	1	0.5
Female	0	0.5
High School	1	0.6
Bachelor	1	0.3
PhD	1	0.1
0-10K	1	0.1
10k-20k	0.8	0.4
20-30k	0	0.4
>30k	0	0.1

Figure 5: An example of the occurrence space

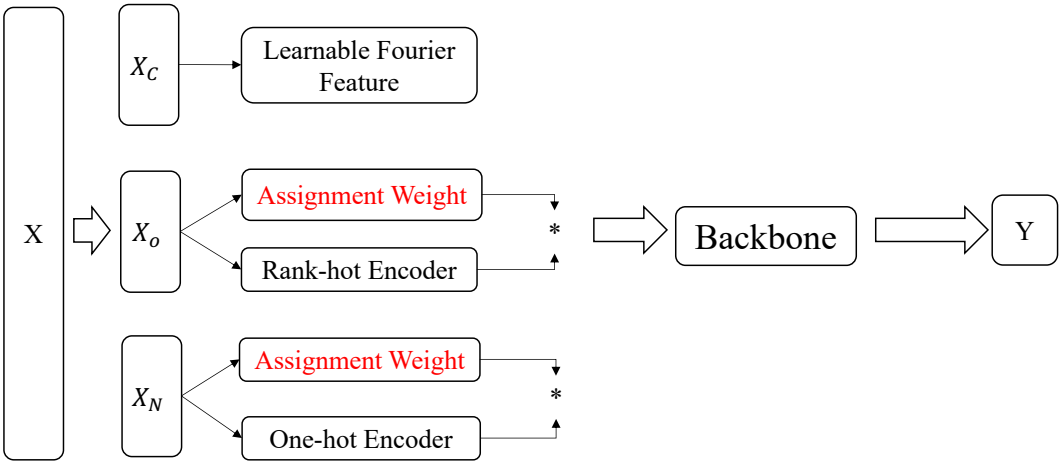


Figure 6: Proposed embedding module. The proposed embedding module incorporates the information of occurrence probabilities, as indicated by the red text. The module handles three types of features: continuous (X_C), ordinal (X_O), and nominal (X_N).

1, 0-10k: 1, 10k-20k: 0.8, 20-30k: 0, >30k: 0] in the occurrence space when we use piece-wise linear encoder for continuous features.

In order to introduce the global frequency information of each feature into the embedding module, we use the feature value's frequency to re-weight the occurrence probability. The proposed methodology involves assigning lower values to features with higher frequencies after the embedding process, while assigning higher values to features with lower frequencies. This approach is based on the rationale that neural units may experience fatigue when exposed to high-frequency values, resulting in reduced input. Figure 6 provides a visual representation of our proposed inverse probability weighting encoder.

3.3 EXPERIMENT

3.3.1 TASK: PREDICTING OUTCOME IN RANDOMIZED TRIAL

Predicting individual outcomes in randomized trials is a fundamental objective in causal inference. In datasets derived from randomized trials, instances are characterized by three classes of features: treatment, pre-treatment, and post-treatment. The values of the treatment feature are not determined by the data collector but are instead assigned by a random generator controlled by the researchers. Subsequently, the treatment is implemented according to the assigned value by the trial executor. The pre-treatment features are measured prior to the assignment of the treatment value, while the post-treatment features are measured subsequent to the treatment assignment. Among the post-treatment features, three categories can be distinguished: main outcome, secondary outcome, and additional post-treatment variables.

3.3.2 DATASET

We have curated a comprehensive collection of heterogeneous randomized trial datasets and made them publicly accessible on the website: https://github.com/herdonyan/RandomizedTrialDataset. The datasets encompass a wide range of randomized trials, representing a valuable resource for researchers in various fields. Despite the significant cost associated with designing and executing randomized trials, there is a growing trend among funding agencies and journals to mandate the availability of these datasets, while ensuring privacy protection measures are in place. This initiative aims to promote transparency, reproducibility, and collaboration in the scientific community by facilitating access to randomized trial data and fostering further research advancements.

The AKIAlert dataset Wilson et al. (2021) is a valuable resource derived from a randomized trial that recorded electronic health record (EHR) data of patients. This dataset follows a double-blinded, multicenter, and parallel design. The primary focus of the trial is to evaluate the impact of an acute kidney injury (AKI) alert provided by the electronic system compared to usual care without an alert. The participants were identified electronically and randomized using a simple randomization approach with allocation concealment.

The dataset comprises a total of 6,030 adult inpatients with AKI, which is defined based on the Kidney Disease: Improving Global Outcomes (KDIGO) creatinine criteria. It includes 49 pre-treatment variables, 1 main outcome variable, and additional post-treatment variables. Among the 49 pre-treatment variables, there are 9 nominal features, 19 ordinal features, 3 interval features, and 20 ratio features.

Within the cohort of 6,030 patients, 948 individuals experienced AKI progression within a 14-day period, while 5,082 patients did not exhibit such progression. The dataset provides a valuable resource for conducting analyses and exploring the impact of the AKI alert on various post-treatment variables, aiding researchers in gaining insights into the management and outcomes of AKI in the context of electronic health records.

The primary task of interest in the AKIAlert dataset is to predict the occurrence of AKI progression within 14 days of randomization based on the available pre-treatment variables. This task can be formulated as a classical binary classification problem, where the objective is to distinguish between patients who will experience AKI progression within the specified timeframe and those who will not.

By leveraging the 49 pre-treatment variables present in the dataset, researchers can develop predictive models and algorithms to identify patterns and relationships that may contribute to the prediction of AKI progression. These variables, including the 9 nominal features, 19 ordinal features, 3 interval features, and 20 ratio features, offer a rich set of information to analyze and extract relevant predictors for the binary classification task.

The successful development of a predictive model for AKI progression within 14 days in randomized trials can have significant clinical implications, enabling early identification and intervention for alert-benefited patients while avoid the for alert-harmful patients. Moreover, it can contribute to advancing the field of acute kidney injury research and improving patient outcomes in healthcare settings.

In order to address the challenge posed by label imbalance, we employ the average precision score (PR-AUC) as the performance metric for evaluating the models. The PR-AUC, a commonly utilized measure in binary classification tasks, provides a comprehensive evaluation of the model's effectiveness in scenarios where there is a significant disparity in the class distribution.

Unlike conventional evaluation metrics such as accuracy or F1-score, the PR-AUC takes into account both precision and recall, which are particularly relevant in imbalanced datasets. Precision quantifies the proportion of correctly predicted positive instances out of all instances classified as positive, while recall captures the proportion of correctly predicted positive instances out of the total number of actual positive instances.

By calculating the area under the precision-recall curve, the PR-AUC delivers a comprehensive assessment of the model's performance across a range of classification thresholds. This approach proves advantageous when dealing with imbalanced datasets, as it focuses on the performance of

the minority class (AKI progress) and is less influenced by the dominance of the majority class (non-AKI progress).

Employing the PR-AUC as the evaluation metric in the assessment of models on the AKIAlert dataset ensures a robust estimation of their predictive capabilities, mitigating the effects of label imbalance. Higher PR-AUC scores signify superior performance in accurately identifying instances of AKI progression, thereby contributing to enhanced patient management and facilitating informed clinical decision-making.

3.3.3 IMPLEMENTATION DETAILS

To address the potential impact of randomness in dataset splitting, all experiments were conducted five times using different random splits of the dataset. This approach helps mitigate the influence of splitting randomness and provides a more robust evaluation of the models' performance.

Four different models were evaluated on the dataset: HetMLP (our proposed model), Vanilla MLP, and stochastic prediction. Each model was trained and tested using the randomized dataset splits, ensuring a comprehensive assessment of their respective performance.

By employing multiple executions of the experiments and evaluating different models, we aim to obtain reliable and statistically significant results. This approach allows us to analyze the performance of each model across multiple iterations, capturing variations in their predictive capabilities and facilitating a more comprehensive understanding of their strengths and weaknesses.

The VilliaMLP model utilized a multi-layer perceptron (MLP) architecture as its backbone. The MLP consisted of three layers, with hidden units of 1024, 512, and 256, respectively. Dropout regularization was applied with a rate of 0.1 to prevent overfitting.

The learning rate for training the model was set to 0.001, promoting efficient optimization during the learning process. Weight decay regularization with a coefficient of 0.000001 was incorporated to prevent excessive model complexity and enhance generalization.

To assess the model's performance and prevent overfitting, a train/test splitting ratio of 8:2 was employed, with 80% of the data allocated for training and 20% for testing. Additionally, a train/validation splitting ratio of 9:1 was used to further evaluate the model's performance during training. Validation was performed at each epoch, and an early stop policy was implemented to select the best-performing model based on the validation dataset. The early stop epoch was set to 20.

For handling the different types of features, specific encoders were employed. Nominal features were encoded using one-hot encoding, while ordinal features were encoded using an ordinal en-

coder. Continuous features were encoded using a learnable Fourier encoder, allowing the model to effectively capture and represent the underlying patterns in the data. The maximum frequency number for the Fourier encoder was set to 200.

To address label imbalance, the loss function employed a weighting scheme based on the labels. This approach helped mitigate any potential degradation in performance resulting from imbalanced class distributions, ensuring fair and accurate model evaluation.

By leveraging the VilliaMLP model with these configurations and techniques, we aimed to effectively leverage the MLP architecture and appropriate feature encoders to achieve accurate predictions while mitigating common challenges such as overfitting and label imbalance.

In contrast to VilliaMLP, HetMLP shares a similar architecture and configuration, with the exception of the feature encoders used. In HetMLP, we adopted a different approach by assigning weights to the transformed features based on their inverse probabilities.

Specifically, the weight assigned to each transformed feature was determined by its frequency in the dataset. If a feature appeared frequently, it was assigned a smaller weight, whereas features with lower frequencies were given larger weights. This weighting scheme aimed to address the heterogeneity in feature frequencies and ensure that each feature contributed appropriately to the overall model representation.

By incorporating these weighted transformed features, HetMLP aimed to capture the varying importance and impact of different features based on their frequencies. This approach allowed the model to effectively handle the heterogeneous nature of the dataset, providing a more nuanced and informative representation of the input data.

Overall, HetMLP and VilliaMLP shared similar architectural configurations, but their respective feature encoding strategies differed. HetMLP leveraged the inverse probability weighting of transformed features to effectively address the heterogeneity of feature frequencies and enhance the model's predictive performance.

Stochastic prediction was employed as a baseline method to compare against the performance of the proposed models. In this approach, the probability $p(y)$ was utilized to make predictions regarding whether a patient would experience acute kidney progress in the future.

By employing stochastic prediction, we aimed to establish a reference point for evaluating the effectiveness of the other models. The baseline method provided a benchmark against which the performance improvements of the CatBoostTree, VilliaMLP, and HetMLP models could be assessed.

Algorithm	PR-AUC
HetMLP	.2117±.0009
VilliaMLP	.2087±.0164
Baseline (Stochastic)	.1568±.0089

Table 2: PR-AUC of different models

Through this comparative analysis, we sought to highlight the advancements and enhancements achieved by the proposed models over the stochastic prediction baseline. The evaluation of the models against this baseline allowed for a comprehensive understanding of their predictive capabilities and demonstrated the potential improvements that can be achieved in predicting acute kidney progress within the given dataset.

3.4 Result Analysis

In order to assess the effectiveness of our algorithm, we compared its performance with that of VilliaMLP using the average precision score (PR-AUC) metric. The PR-AUC metric was chosen to account for the label imbalance in the dataset.

Upon evaluating the results, it can be observed that our algorithm outperformed VilliaMLP in terms of PR-AUC. The higher PR-AUC score achieved by our algorithm indicates its superior ability to accurately predict the occurrence of acute kidney progress within the specified timeframe.

The comparison with VilliaMLP serves as empirical evidence supporting the effectiveness and improved performance of our algorithm in addressing the given task. The results demonstrate the potential of our algorithm as a valuable approach for predicting acute kidney progress in randomized trial, surpassing the performance of the previously established VilliaMLP model.

4 Future plan

The first research problem is to find a representation for heterogeneous features for downstream model and given task. The difficulty is how to combine the heterogeneous feature together to improve the models' performance. Our previous work showed the potential development of heterogeneous feature embedding modules for the given task. However, there are some limitation of our work.

- First, the feature structure and instance structure is not considered in our model. If we learned the sparse heterogeneous feature structure and utilize the distance of different data points with heterogeneous features. The performance of our model may can be one of the state-of-the-art for heterogeneous feature embedding.

- Second, the computation of the structures cost too much times. How to find a more efficient algorithm is another question.
- Third, we did not test the heterogeneous feature embedding modules for other backbone models, such as diffusion model, transformers. There should be more datasets in our experiment.

The second research problem is how to automatically infer the heterogeneous feature structures for feature selection. Our preliminary work showed that potential improving space of prediction performance. First, features can be heterogeneity data, such as different data scales (nominal, ordinal, interval, ratio), different data sources (tabular, image, text) in the reality. Second, the features were not all helpful for the given task and we should select some features and ignore others. In order to solve this problem, we are planning to use Wasserstein distance to automatically learn the graph structures between heterogeneous features:

- design a novel encoder to handle the heterogeneous features in a united space which support sparse feature weighting for feature selection;
- balance the intra-attribute structures and inter-attribute structures;
- apply the graph theory to accelerate the computation.

REFERENCES

Léon Bottou and Yann Cun. Large scale online learning. *Advances in neural information processing systems*, 16, 2003.

Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794, 2016.

Alexandre d'Aspremont, Laurent Ghaoui, Michael Jordan, and Gert Lanckriet. A direct formulation for sparse pca using semidefinite programming. *Advances in neural information processing systems*, 17, 2004.

Yinfu Feng, Jun Xiao, Yueting Zhuang, and Xiaoming Liu. Adaptive unsupervised multi-view feature selection for visual concept recognition. In *Computer Vision–ACCV 2012: 11th Asian Conference on Computer Vision, Daejeon, Korea, November 5-9, 2012, Revised Selected Papers, Part I 11*, pp. 343–357. Springer, 2013.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943, 2021.

Yury Gorishniy, Ivan Rubachev, and Artem Babenko. On embeddings for numerical features in tabular deep learning. *Advances in Neural Information Processing Systems*, 35:24991–25004, 2022.

Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, et al. Practical lessons from predicting clicks on ads at facebook. In *Proceedings of the eighth international workshop on data mining for online advertising*, pp. 1–9, 2014.

Qinghua Hu, Daren Yu, and Zongxia Xie. Information-preserving hybrid data reduction based on fuzzy-rough techniques. *Pattern recognition letters*, 27(5):414–423, 2006.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Seyoung Kim and Eric P Xing. Statistical estimation of correlated genome associations to a quantitative trait network. *PLoS genetics*, 5(8):e1000587, 2009.

John Langford, Lihong Li, and Tong Zhang. Sparse online learning via truncated gradient. *Journal of Machine Learning Research*, 10(3), 2009.

Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. Model ensemble for click prediction in bing search ads. In *Proceedings of the 26th international conference on world wide web companion*, pp. 689–698, 2017.

Jun Liu and Jieping Ye. Moreau-yosida regularization for grouped tree structure learning. *Advances in neural information processing systems*, 23, 2010.

Daniele Micci-Barreca. A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD Explorations Newsletter*, 3(1):27–32, 2001.

Thomas Müller, Brian McWilliams, Fabrice Rousselle, Markus Gross, and Jan Novák. Neural importance sampling. *ACM Transactions on Graphics (ToG)*, 38(5):1–19, 2019.

Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in neural information processing systems*, 31, 2018.

Noah Simon, Jerome Friedman, Trevor Hastie, and Robert Tibshirani. A sparse-group lasso. *Journal of computational and graphical statistics*, 22(2):231–245, 2013.

Jiliang Tang, Xia Hu, Huiji Gao, and Huan Liu. Unsupervised feature selection for multi-view data in social media. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, pp. 270–278. SIAM, 2013.

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

Changzhong Wang, Yan Wang, Mingwen Shao, Yuhua Qian, and Degang Chen. Fuzzy rough attribute reduction for categorical data. *IEEE Transactions on Fuzzy Systems*, 28(5):818–830, 2019.

F Perry Wilson, Melissa Martin, Yu Yamamoto, Caitlin Partridge, Erica Moreira, Tanima Arora, Aditya Biswas, Harold Feldman, Amit X Garg, Jason H Greenberg, et al. Electronic health record alerts for acute kidney injury: multicenter, randomized clinical trial. *Bmj*, 372, 2021.

Jieping Ye and Jun Liu. Sparse methods for biomedical data. *ACM Sigkdd Explorations Newsletter*, 14(1):4–15, 2012.

Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 68(1):49–67, 2006.

Zhong Yuan, Xianyong Zhang, and Shan Feng. Hybrid data-driven outlier detection based on neighborhood information entropy and its developmental measures. *Expert Systems with Applications*, 112:243–257, 2018.

Zhong Yuan, Hongmei Chen, Tianrui Li, Jia Liu, and Shu Wang. Fuzzy information entropy-based adaptive approach for hybrid feature outlier detection. *Fuzzy Sets and Systems*, 421:1–28, 2021a.

Zhong Yuan, Hongmei Chen, Pengfei Zhang, Jihong Wan, and Tianrui Li. A novel unsupervised approach to heterogeneous feature selection based on fuzzy mutual information. *IEEE Transactions on Fuzzy Systems*, 30(9):3395–3409, 2021b.

Kun Zhang, Bernhard Schölkopf, Krikamol Muandet, and Zhikun Wang. Domain adaptation under target and conditional shift. In *International conference on machine learning*, pp. 819–827. PMLR, 2013.

Pengfei Zhang, Tianrui Li, Zhong Yuan, Chuan Luo, Keyu Liu, and Xiaoling Yang. Heterogeneous feature selection based on neighborhood combination entropy. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Zheng Zhao and Huan Liu. Multi-source feature selection via geometry-dependent covariance analysis. In *New Challenges for Feature Selection in Data Mining and Knowledge Discovery*, pp. 36–47. PMLR, 2008.