Neural-FCA

Nadir Mohammed

December 2023

Neural-FCA Overview:
Neural-FCA represents a hybrid methodology that combines the strengths of neural network architectures with the formal, mathematical underpinnings of Formal Concept Analysis. FCA, originally rooted in mathematical lattice theory, provides a structured framework for analyzing and organizing complex data relationships. By integrating FCA with neural networks, Neural-FCA aims to capitalize on the expressive and adaptive nature of neural architectures, while benefiting from the interpretability and structure offered by formal concepts.

1 Answer:

1.1 Zoo Dataset:

1.1.1 An overview:

The "Zoo" dataset contains information about various animals and their classification as mammals, birds, fishes, etc. The original dataset is comprised of 101 instances, each representing a different animal, and it includes 16 features describing various attributes of the animals.

Objective:
The primary goal is to classify animals into predefined classes (birds.) based on their features using Neural FCA.

1.1.2 Data Preprocessing:

In zoo datasets, there are no missing values. The dataset is prepared for a binary classification task focused on predicting whether an object belongs to the 'birds' class or not, passing by many steps, which are mentioned below:

One-Hot Encoding:
To handle categorical variables in both the features and target, by using the function **pd.getdummies** is used to perform one-hot encoding on the 'target' and 'feature' variables. For the target variable (y), one-hot encoding is done for the column named 'type', and the new columns are prefixed with 'class'. For the feature variable (X), one-hot encoding is done for the 'legs' column, and the new columns are prefixed with 'legs'.

Conversion to Boolean:
The binary numerical values in both X and y (0 and 1) are converted to boolean values (False and True) using replace.

Index Conversion:
The indices of both X and y are converted to strings.

Attribute Removal:
Columns catsize, legs5, legs6, and legs8 are dropped from X. This is done to focus only on the target $class_2$ (birds) and eliminate unnecessary attributes for animals with 4, 5, 6, and 8 legs, because we are dealing with birds class.

Renaming Index:
The indices of both X and y are renamed to Object.

Target Variable Isolation:
A new variable $y - one - target$ is created, containing only the $class_2$ column from y. This is done to avoid potential issues with a multilabel-indicator.

1.1.3 Model Building:

The dataset is split into training and testing sets with an 80-20 ratio. random-state is set to 42 for reproducibility. The resulting sets (Xtrain, Xtest, ytrain, ytest) are ready to be used for training and evaluating a machine learning model for the binary classification task.

In this step, a set of classification algorithms is selected, including decision tree, random forest, xGboost, catboost, k-NN, Naive Bayes, logistic regression, and also Neural-FCA. These algorithms will be trained on the training set to learn patterns and relationships within the data.

1.1.4 Model Evaluation:

In this step, we assess the model's performance on the testing set using metrics such as accuracy and F1 score. Subsequently, various methods will be explored to adjust the model parameters, as elaborated further in this report. The objective is to enhance the model's performance.

After performing classification on all three selected datasets using these standard ML tools:
- Decision tree
- Random forest
- xGboost
- Catboost
- k-NN
- Naive Bayes
- logistic regression.

and also the Neural-FCA, We obtained classification reports for each the first Zoo dataset in tabular form.

1.1.5 Overall Performance:

. **K-folds:**
K-fold cross-validation provides a more robust estimate of a model's performance compared to a single train-test split. It helps ensure that the model's performance is not overly dependent on a specific subset of the data. The model is then trained and evaluated "k" times, each time using a different fold as the test set and the remaining folds as the training set. Common choices for "k" include 5-fold and 10-fold cross-validation.

Table 1: A report for the Zoo dataset.

$StandardMLtools$	$Accuracy$	$F1Score$
Decision Tree	1.0000	1.0000
Random Forest	1.0000	1.0000
Naive Bayes	1.0000	1.0000
k-NN	1.0000	1.0000
CatBoost	1.0000	1.0000
XGBoost	1.0000	1.0000
Logistic Regression	1.0000	1.0000

Neural-FCA (K-fold 5 and 10):

Table 2: Neural-FCA report Zoo dataset.

$NeuralFCA$	$k-fold=5$	$K-fold=10$
Accuracy	0.9048	1.000
F1 Score	0.8595	1.000

1.2 Titanic Dataset:

1.2.1 An overview:

The dataset contains information about passengers on the Titanic, including features like age, sex, class (ticket class), number of siblings/spouses aboard, number of parents/children aboard, fare, cabin, and embarkation port. The original dataset is comprised of 891 instances, each representing a person, and it includes 15 features. The target variable is whether a passenger survived (1) or did not survive (0).

Objective:
The main goal is to build a predictive model that can accurately predict whether a passenger survived or not based on the given features using Neural FCA.

1.2.2 Data Preprocessing:

In the Titanic dataset, the function isnull().sum() revealed several missing values in the following features: Age (177), Deck (688), and two missing values for the embark-town feature.

In response to these findings, a data cleaning step was performed, and certain columns were removed for various reasons:

Columns age and fare were dropped due to their large variety of values. deck was removed due to a high percentage of missing values (688 out of the total dataset). alive and pclass columns were also removed, potentially because their information was either duplicated or deemed unnecessary for the specific analysis. The resulting dataset, stored in the variable data, has undergone these cleaning operations and is now prepared for further processing, such as binarization.

1.2.3 Data Splitting:

The dataset is split into features (X) and the target variable (y). Features (X) include one-hot encoded columns for categorical variables such as sex, sibsp, parch, embarked, class, who, adult-male, embark-town, and alone. The target variable (y) is assigned the survived column.

Binarization:
Binary numerical values in both X and y are converted to boolean values using replace.

Index Conversion: The indices of both X and y are converted to strings.

Naming Axis:
The axis of the index is named as "Object" for both X and y.

Index Reset:
The index of the original dataset (data) is reset from 0 to 889 using reset-index(drop=True)

1.2.4 Model Building and Model Evaluation:

The same steps have been performed, as in the previous Zoo dataset.

1.2.5 Overall Performance:

Table 3: A report for the Titanic dataset.

$StandardMLtools$	$Accuracy$	$F1Score$
Decision Tree	1.0000	1.0000
Random Forest	1.0000	1.0000
Naive Bayes	1.0000	1.0000
k-NN	0.9663	0.9665
CatBoost	1.0000	1.0000
XGBoost	1.0000	1.0000
Logistic Regression	1.0000	1.0000

Neural-FCA (K-fold 5 and 10):

Table 4: Neural-FCA report Titanic dataset.

$NeuralFCA$	$k-fold=5$	$K-fold=10$
Accuracy	0.6124	0.6124
F1 Score	0.4651	0.4651

1.3 Cars Evaluation Dataset:

1.3.1 An Overview:

The Car Evaluation Dataset consists of features related to different cars, such as the number of doors, the number of persons the car can accommodate, the safety of the car, etc. The

target variable is usually the acceptability of the car, categorized into classes like unacc (unacceptable), acc (acceptable), good, and vgood (very good). The original dataset is comprised of 1728 instances, and it includes 6 input attributes: buying, maint, doors, persons, lug-boot, safety.

Objective:
The primary goal is to classify cars into acceptable classe based on their features. using Neural FCA.

1.3.2 Data Preprocessing:

In Car Evaluation Dataset, there are no missing values. The dataset is prepared for a binary classification task focused on predicting whether an object belongs to the acceptable classe or not, passing by many steps, which are mentioned below:

One-Hot Encoding:
pd.get-dummies is used to perform one-hot encoding on the class column in the target variable (y). For the features (X), one-hot encoding is performed on the columns buying, safety, persons, doors, maint, and lug-boot.

Index Conversion:
The indices of both X and y are converted to strings.

Naming Axis:
The axis of the index is named as "Object" for both X and y.

Target Variable Isolation:
A new variable y-one-target is created, containing only the class-acc column from y. This is done to avoid potential issues with a multilabel-indicator.

1.3.3 Model Building and Model Evaluation:

The same steps have been performed, as in the previous datasets.

Table 5: A report for the Cars dataset.

$StandardMLtools$	$Accuracy$	$F1Score$
Decision Tree	0.8825	0.8833
Random Forest	0.9037	0.8988
Naive Bayes	0.7919	0.8084
k-NN	0.9017	0.8982
CatBoost	0.9750	0.9746
XGBoost	0.9499	0.7941
Logistic Regression	0.7977	0.7941

Neural-FCA (K-fold 5 and 10):

Table 6: Neural-FCA report Cars Evaluation dataset.

$NeuralFCA$	$k-fold=5$	$K-fold=10$
Accuracy	0.7726	0.7726
F1 Score	0.6735	0.6871

1.4 Improving the performance:

In the context of neural FCA and with the aim of enhancing the basic baseline, I have incorporated three distinct activation functions, in addition to the standard ReLU function, into the neural network. These activation functions are implemented in the 'neural.lib.py' file. Specifically, they are used as follows:

- torch.nn.ReLU
- torch.nn.sigmoid
- torch.nn.softmax
- torch.nn.Tanh.

Note:
I attempted to assess the impact of excluding k-fold cross-validation on a neural network model, with different activation functions.

1.4.1 RELU function:

Commonly used in hidden layers due to its simplicity and efficiency.

$$\boldsymbol{ReLU}\,(\boldsymbol{x}) = \boldsymbol{max}\,(\boldsymbol{0}, \boldsymbol{x})$$

Overall Performance:

Table 7: Neural FCA report for three datasets.

$Dataset$	$Accuracy$	$F1Score$
Titanic Dataset	0.6124	0.4651
Cars Dataset	0.7726	0.6735
Zoo dataset	0.9032	0.8573

1.4.2 Sigmoid function:

Commonly used in the output layer for binary classification problems (0, 1).

$$\boldsymbol{\sigma}\,(\boldsymbol{x}) = \frac{\boldsymbol{1}}{\boldsymbol{1+e^{-x}}}$$

Overall Performance:
we obtained different results from the previous one:

Table 8: Neural FCA report for three datasets.

Dataset	*Accuracy*	*F*1*Score*
Titanic Dataset	0.7809	0.7609
Cars Dataset	0.7726	0.6735
Zoo dataset	0.9032	0.8573

1.4.3 Softmax function:

$$\boldsymbol{softmax}\left(\boldsymbol{z_i}\right) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{x_j}}$$

Overall Performance:

Table 9: Neural FCA report for three datasets.

Dataset	*Accuracy*	*F*1*Score*
Titanic Dataset	0.6124	0.4651
Cars Dataset	0.7726	0.6735
Zoo dataset	0.9032	0.8573

1.4.4 Tanh function:

Commonly used in binary classification problems, especially in the context of recurrent neural networks.

$$\boldsymbol{tanh}\left(\boldsymbol{x}\right) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

Overall Performance:

Table 10: Neural FCA report for three datasets.

Dataset	*Accuracy*	*F*1*Score*
Titanic Dataset	1.0000	1.0000
Cars Dataset	0.7996	0.7935
Zoo dataset	1.0000	1.0000

1.5 Various techniques to select best concepts from the concept lattice:

1.5.1 Chi-square Test:

The Chi-square test is used for categorical features in a dataset. We calculate Chi-square between each feature and the target and select the desired number of features with the best Chi-square scores. In order to correctly apply the chi-squared to test the relation between various features in the dataset and the target variable, the following conditions have to be met: the variables have to be categorical, sampled independently, and values should have an expected frequency greater than 5.

This snippet of code performs feature selection using the chi-squared statistical test. SelectKBest is employed to select the k-best features(the best 10 features by defaults), and the chi-squared test (chi2) is utilized as the scoring function. The selected features are then transformed, for further model training. This step is crucial for reducing dimensionality and selecting features that are most relevant for the given classification task.

```
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
chi2_features = SelectKBest(chi2)
x_kbest_features = chi2_features.fit_transform(X_train, y_train)
x_kbest_features_int = x_kbest_features.astype(int)
x_kbest_features.shape
(1209, 10)
```

This second snippet of code retrieves the indices and names of the features that were deemed most relevant for the model. The printed information includes the column index and name of each selected feature, aiding in the interpretation and understanding of the feature selection results.

```
# Get the selected feature indices
selected_feature_indices = chi2_features.get_support(indices=True)
# Get the names of the selected columns
selected_column_names = X_train.columns[selected_feature_indices]
for i in range(len(selected_column_names)):
    print(" feature with column_index {:}
    = {:}".format(selected_feature_indices[i], selected_column_names[i]))
```