Neural FCA

Yuriy Sosnin

December 13, 2022

1 Dataset & Binarination Strategy

The data I chose for this homework is a somewhat classic Credit Scoring binary classification dataset from Germany. It has 1000 objects and 9 attributes: one already binary ("Sex"), three numeric ("Duration", "Age", "Credit Amount") and 5 categorical ("Job", "Housing", "Saving accounts", "Checking account", "Purpose").

Target variable "Risk" represents risk of default and is encoded 1 for safe clients, and 0 for risky. The classification task is unbalanced: 70% of observations is 1. Considering this, I chose F1 Score as main classification metric. It is a common in such settings, where simple accuracy fails because of non-equal distribution of targets.

The other major desision was the choise of binarization strategy. Initally I thought of three approaches: manual, statistical and neural. Manual scaling would require me to appropriately divide numeric attributes into categories according to some "domain knowledge", e.g. age into "young", "middle", "elderly", etc. I decided to not consider this approach, to test Neural FCA algorithm in a more general setting, where such strategies would not be possible, for instance with a more numerous or non-interpretable attributes.

Secondly, I thought of utilising Neural Networks for this task. As main algorithm is a Neural Network itself, another module mapping continuous numeric attributes to binary features could in theory be made trainable with backpropagation as well. However, main Network requires a lattice, constructed from already prepared features, which makes training binarization module at the same time non-trivial. Perhaps useful concepts (and their connections) could be recalculated on-the-go as well, but it is too complex and defeats the purpose of designing interpretable NN algorithm.

In the end I decided to settle for statistical approach, simply dividing numerical attributes into bins with either uniform or quantile statistics, based on training sample.

2 Model Description

The idea of this Neural FCA algorithm is to incorporate FCA lattice into the structure of a Neural Network, making it sparse, by providing an inductive bias for which connections are useful and which are not before training. In a sense, all major NN architectures are designed to reduce the number of trainable parameters in some meaningful way and make the network aware of some structure of the data. As I understand this would be an attempt to do it in general classification task.

The idea provided in base pipeline is as follows. First, Formal Context is created based on training part of the data. Second, a Concept Lattice is constructed from it. Then, a small number of concepts is selected based on a criterion (say, F1 score). They from a partially ordered set, which connections are transferred into a sparse linear layers of an MLP.

I encountered some problems during development of the project: the provided code for constructing layers of a network did not work as intended, so I had to rewrite it. Moreover, I realized that computing Concept Lattice from full Formal Concept with several hundred objects takes very long to be compured. Fortunately, there is another algorithm in FCApy package called Sofia, which does not compute full lattice, but only predefined number of 'best' concepts according to stability measure. This way not only is lattice construction much quicker, but also the next step is redundant, as the small number of concepts is already chosen (according to stability).

3 Experiments

Neural FCA model was tested with different nonlinearities, number of concepts, and binarization strategies. Table 1 depicts F1 Scores with ReLU, Tanh and Sigmoid as activation functions; ReLU and Sigmoid show the same result, while Tanh ends up with lower metric score. All the rest experiments are done with ReLU as nonlinearity.

Table 1: F1-Score per nonlinearity

	F1 Score
ReLU	0.8201
Tanh	0.8097
Sigmoid	0.8201

As for binarization strategies, quantile and uniform modes were tested, but did not show any difference between them. The same, surprisingly, goes for number of best concepts provided by Sofia algorithm. 10, 20, 30, 40, and 50 concepts all yielded the same result, while taking progressively more time to train. For comparison with other models, the smallest network with 10 concepts was chosen.

Table 2: F1-Score on test data

	Binary	Numeric
KNN	**0.8520**	0.8455
Random Forest	0.8333	0.8339
Logistic Regression	0.8143	0.8212
MLP	0.8200	0.7579
Neural FCA	0.8439	-

In Table 2 there are scores for different ML models typically applied to binary classification on table data, compared to Neural FCA. As can be seen, Neural FCA does quite well, outperforming Random Forest, Logistic Regression and fully-connected MLP. However, simplest KNN with 50 neighbours outperforms them all on both binarized dataset and original dataset with numeric features.

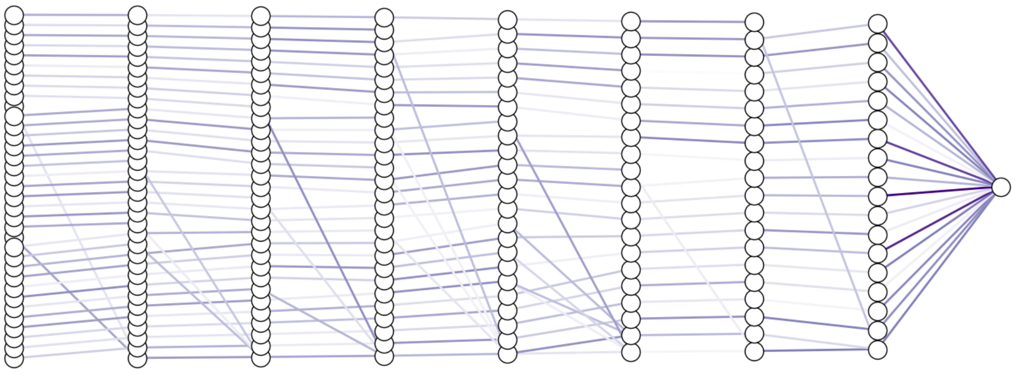

Figure 1: Weigths of fitted network, absolute values