NeuralFCA_Kolotov

December 13, 2022

1 Dataset

1.1 Dataset description

Let me introduce the dataset "Credit Card Approvals" - small dataset for binary classification task. Dataset is openly available: - UCI Machine Learning Repository - original dataset - kaggle - the cleared version (this one is used)

This dataset has good mix of attributes - continuous, categorical with small numbers of values, and categorical with larger numbers of values.

The dataset contains 690 objects, each described by 15 attributes:

- binary: `Gender`, `Married`, `BankCustomer`, `PriorDefault`, `Employed`, `DriversLicense`
- numeric: `Age`, `Debt`, `YearsEmployed`, `Income`, `CreditScore`
- categorical: `Industry`, `Ethnicity`, `Citizen`
- target: `Approved`

1.2 Suitable quality metric

Firstly, note that the dataset is balanced (~44% positive objects):

[5]:
```
0    383
1    307
Name: Approved, dtype: int64
```

For binary classification tasks , the following metrics are commonly used: (in terms of confusion matrix)

$$Accuracy = \frac{TP}{TP + TN + FP + FN}$$
$$Precision = \frac{TP}{TP + FP}$$
$$Recall = \frac{TP}{TP + FN}$$
$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$

Since the dataset is balanced we are free to use Accuracy metric, but in tasks such as credit scoring we usually pay more attention to the positive class, so let's use F1 metric instead.

1.3 Binarization strategy

Let's binarize data is such way:

- For categorical and binary features let's use one-hot encoding
- For numeric features let's select bins (using quantiles), then use interval $[b_i, b_j]$ $\forall b_i, b_j \in bins$
- Binary features - let's keep the same

Now we have 49 binary attributes :)

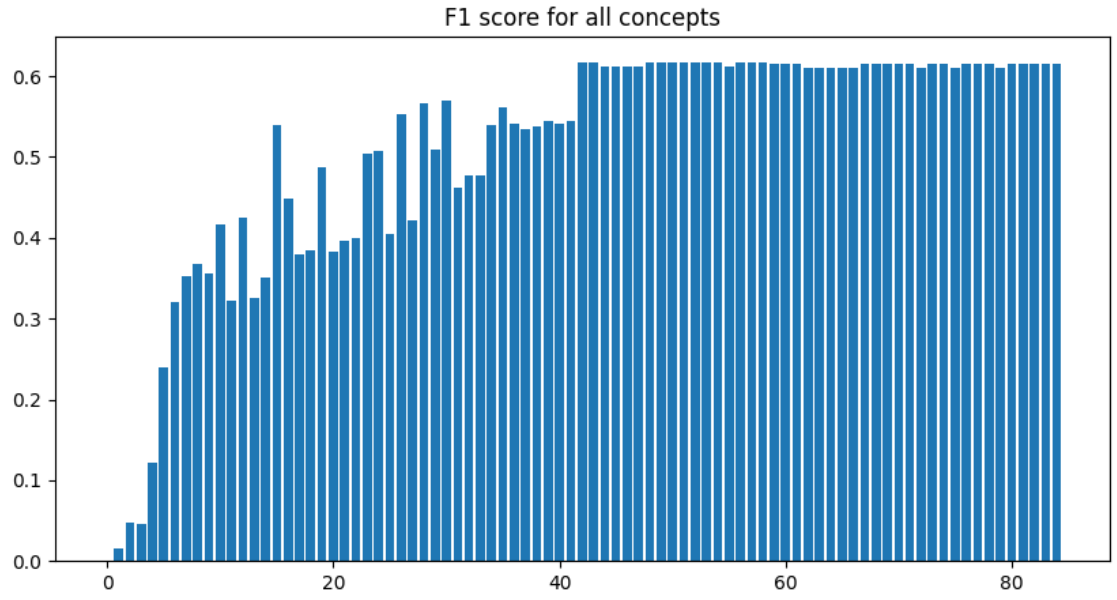
1.4 Splitting data

Let's split the dataset to train (40%), validation(20%) and test (40%) groups of data. We will use train data for computing formal concepts and fitting network. Validation data - for tuning parameters (such as the number of concepts) and test data - for quality measurment.

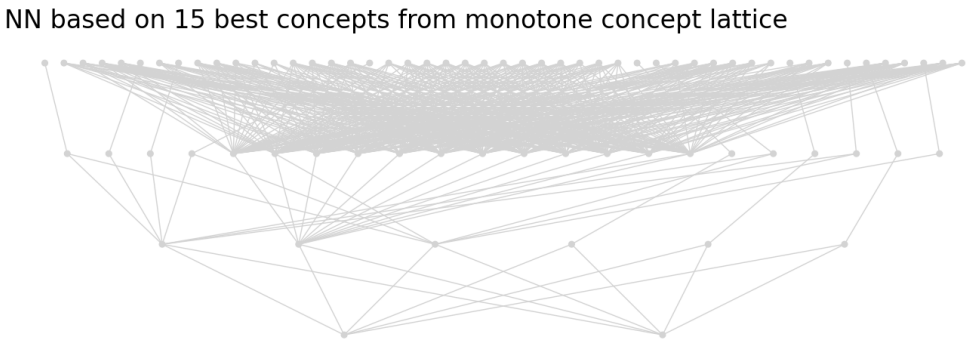
2 FCA

2.1 Find best formal concepts

Use train data to compute monotone ConceptLattice. Since `CbO` algorithm is too slow, let;s use `Sofia` algo instead to select only few interesting concepts. Then, compute F1 score for each formal concept (assuming that an object is predicted True if it is in the extent of the concept)



The concepts at the lasts levels have the maximum F1 score. Let's choose 15 best concepts (we have to cover all train objects).



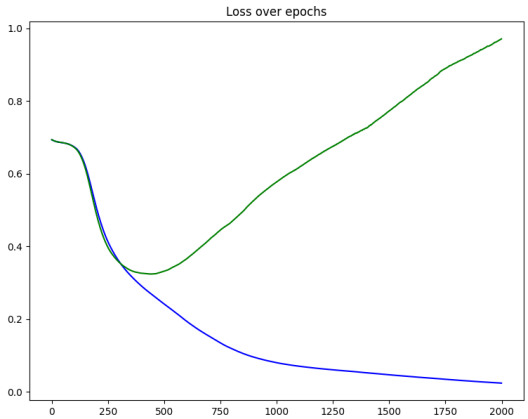
3 Neural Network

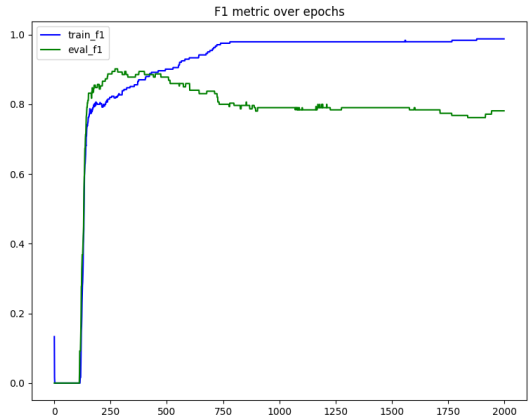
Note, the `neural_lib` file has been modified in such way:

- Add history for `fit` function
- Add eval dataset to `fit` fucntion (to control overfitting)
- Drop `Sigmoid` activation function on the last layer - it's already implemented in loss function - `CrossEntropyLoss`
- Prettify `fit` and `predict` functions with `model.train()/eval()` modes and `torch.no_grad()` context manager

3.1 Optimization

Let's fit our NN and see what happened. We use optimizer Adam with a huge number of epochs = 2000 and learning rate = 0.001. As nonlinearity we use `ReLU` as default.





As we can see - the max quality is acheved with a small (250) number of epochs, then the network begins to overfitting. Let's fix the number of epochs on 250 and measure quality on dataset.

```
Test Quality :  0.8050847457627119
Val Quality :  0.823529411764706
Train Quality :  0.8305084745762712
```

We have achieved 0.80 F1 score on test dataset - it's good but let's start experiments to improve it!

4 Experiments

4.1 Comparison of different binarization strategies

Firstly, let's improve our binarization strategy. Studying lattices, i noticed that associative rules do not support negation natively: for example, if there is a attribute "yellow", then we cannot get a rule: if not "yellow" then .... So, after getting the binary context, let's add a negation for each attribute.

As we can see below, the F1 score increased to 0.827 on test dataset!

```
Test Quality :  0.8278688524590164
Val Quality :  0.8429752066115702
Train Quality :  0.8389830508474576
```

Also, for numeric features let's compare 2 strategies: `range` - use interval $[b_i, b_j]\ \forall b_i, b_j \in bins$, `semirange` - use interval $[-\infty, b_i]\ \forall b_i \in bins$

For `range`, we have calculations above, for `semirange` calculations are shown below:

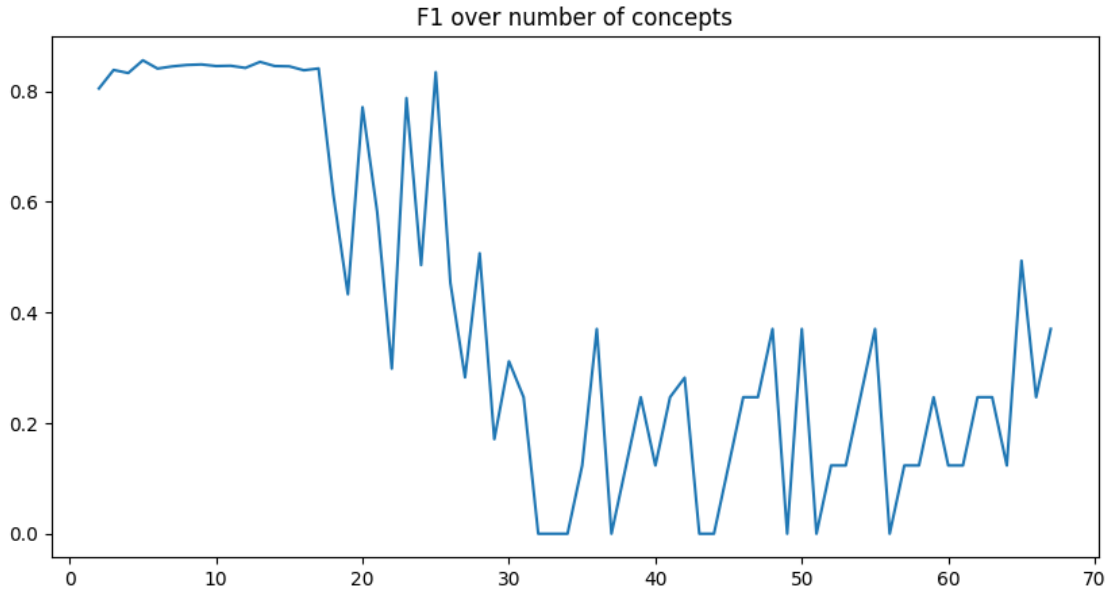
```
Test Quality :  0.8050847457627119
Val Quality :  0.8135593220338982
Train Quality :  0.829059829059829
```

The quality has decreased, so the best binarization is defined as follows: - For categorical and binary features let's use one-hot encoding - For numeric features let's compare (see experiments) 2 strategies: select bins (using quantiles), then use interval $[b_i, b_j]\ \forall b_i, b_j \in bins$ - Binary features let's keep the same - After such binarization for all features let's use both it and its negation

Let's use this binarization further.

4.2 Choosing the right number of concepts

Let's try to find the best number of concepts - using validation dataset and concidering number of concepts as hyperparameter.

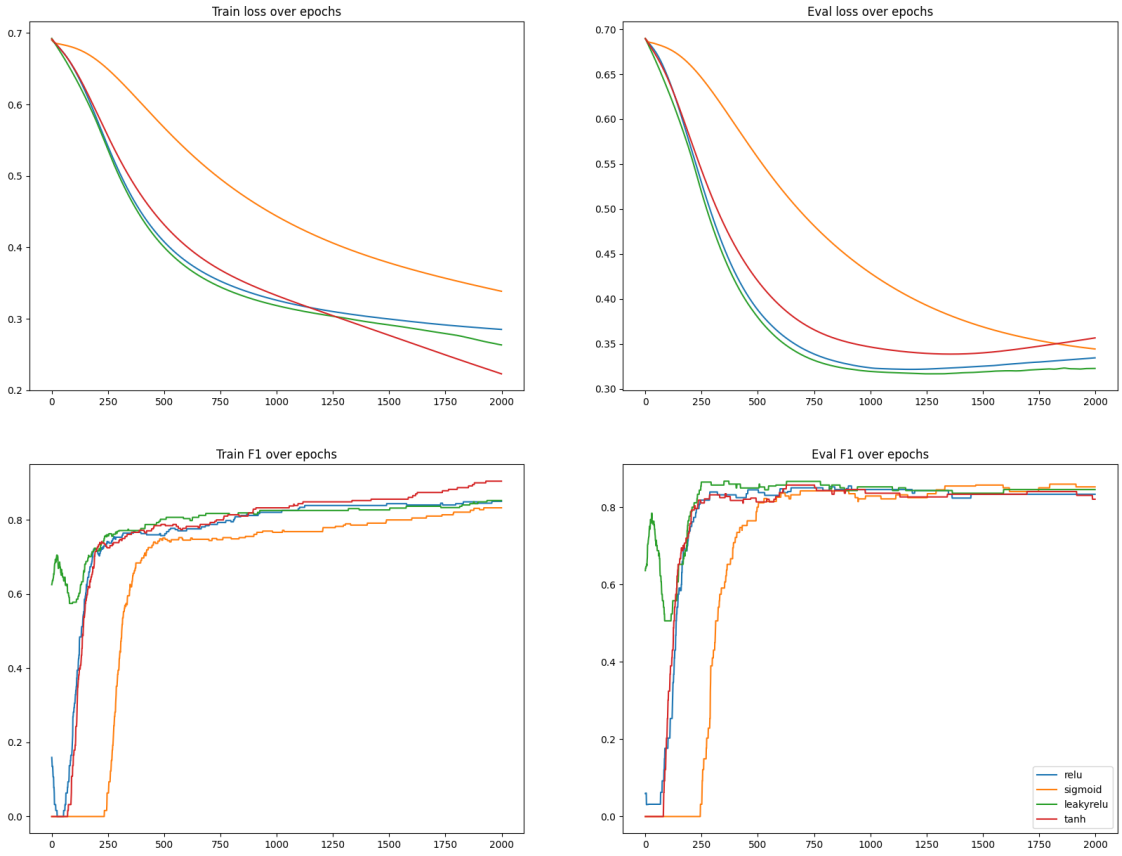


So, the number of concepts that shows the best quality is 5. Let's use it further.

4.3 Comparison of different activation functions (nonlinearities)

Then, let's experiment with the model so that it doesn't overfitting. Let's decrease learning rate to `3e-4` - magic value for training any network. Then, fix the number of epoch at acceptable level (750). Then, try to use the following activation functions:

- `ReLU`
- `Sigmoid`
- `LeakyReLU` with $\alpha = 0.01$
- `Tanh`



[102]:

	train	val	test
relu	0.850000	0.833333	0.827869
sigmoid	0.832618	0.852459	0.811715
leakyrelu	0.852321	0.845528	0.832653
tanh	0.904564	0.820513	0.809917

So, we can see that the `LeakyReLU` shows the best quality - 0.83

5 Vizualization of the best ConceptNetwork

How beautiful it looks...

the best Concept Network

6 The best ConceptNetwork model vs State-of-the-Art approaches

As SotA approaches we will try:

- Decision Tree
- Random Forest
- Catboost - since we have categorical features
- LogReg - linear model
- KNN - k-nearest neighbor algorithm
- MLP - fully-connected neural network with same number of neurons

We will use not binarized dataset for Catboost, with only categorical features binarized for others and full binarized one for ContextNetwork.

[136]:

	train	val	test
DT	0.870690	0.834783	0.808333
Catboost	0.841667	0.934426	0.868526
RF	0.975207	0.847458	0.831933
LogReg	0.834711	0.857143	0.789916
KNN	0.746667	0.789474	0.771930
MLP	0.991803	0.764228	0.762295
ContextNetwork	0.851240	0.845528	0.832653

As we can see, the ContextNetwork achieves results comparable to SotA - it is on the same level with Decision Tree and Random Forest and only Catboost outperforms it.