Ensemble Techniques for Lazy Classification Based on Pattern Structures

Ilya Semenkov[1] and Sergei O. Kuznetsov[1]

Higher School of Economics – National Research University, Moscow, Myasnitskaya street 20, 101000, Russia

Abstract. This paper presents different versions of classification ensemble methods based on pattern structures. Each of these methods is described and tested on multiple datasets (including datasets with exclusively numerical and exclusively nominal features). As a baseline model Random Forest generation is used. For some classification tasks the classification algorithms based on pattern structures showed better performance than Random Forest. The quality of the algorithms is noticeably dependent on ensemble aggregation function and on boosting weighting scheme.

Keywords: Formal Concept Analysis (FCA) · pattern structures · boosting algorithms · ensemble algorithms

1 Introduction

Pattern structures were introduced in [1] for the analysis of data with complex structure. In [6] [5] a model of lazy (query-based) classification using pattern structures was proposed. The model shows quite good performance, which, however, is lower than that of ensemble classifiers that employ boosting techniques. The main goal of this paper is to study various ensemble approaches based on pattern structures, boosting, and different aggregation functions. The model is known for good interpretability, so we would try to use ensemble techniques to improve its prediction quality.

2 Model description

2.1 Pattern structures

The main idea of the pattern structures is the use of intersection (similarity) operation, with the properties of a lower semilattice, defined on object descriptions to avoid binarization (discretization) prone to creating artifacts [1]. An operation of this kind allows one to define Galois connection and closure operator, which can be used to extend standard FCA-based tools of knowledge discovery to non-binary data without scaling (binarizing) them.

At the first step of the model we transform features in the following way: consider $x \in \mathbb{R}^n$ to be an observation, then we transform it using the following transformation $T : (x_1, x_2, ..., x_n) \to ((x_1, x_1), (x_2, x_2), ..., (x_n, x_n))$. Basically, for each feature j, its value in the observation x gets transformed from a number x_i into a 2-dimensional vector (x_i, x_i) with the same value repeated twice. This is used later in the definition of similarity operation.

After the transformation each observation x has 2 values for each feature i, namely $x_{i,1}$ and $x_{i,2}$. In this paper, we define the similarity of two transformed observations x, y in the standard way of interval pattern structures [4] [5] [6]:

$$x \sqcap y = ((min(x_{1,1}, y_{1,1}), max(x_{1,2}, y_{1,2})), ..., (min(x_{n,1}, y_{n,1}), max(x_{n,2}, y_{n,2})))$$

where in $x_{i,j}$ and $y_{i,j}$ i indicates a feature and j indicates one of two values for a feature. In other words for each feature i there were $x_i = (x_{i,1}, x_{i,2}), y_i = (y_{i,1}, y_{i,2})$. After similarity operation $(x \sqcap y)_i = (min(x_{i,1}, y_{i,1}), max(x_{i,2}, y_{i,2}))$. Below, for simplicity this operation will be called intersection.

The subsumption relation is defined as the natural order of the semilattice: $x \sqsubset y \equiv x \sqcap y = x$.

A hypothesis for a description x is a description $x_h \in C_j : \nexists y \in C_i (i \neq j) : (x \sqcap x_h) \sqsubset y$, where C_i stands for a set of elements from class i. So, if a description x_h does not fit any observation from classes C_j $(j \neq i)$, but fits at least 1 observation of the class C_i, then it is considered to be a hypothesis for the class C_i.

The aggregation function is applied either to the whole set of all intersections between a test observation and every element of the training sample, or to the set of hypotheses (extracted from the training set).

Aggregation functions There are many reasonable aggregation functions. In our experiments we have used the following ones:

1. *avglengths*: $C = arg \min_{C_i} \frac{1}{\sum_{k \in A_{C_i}} w_k} \sum_{k \in A_{C_i}} dist\,(k)$, where $dist\,(k) = \sum_{j=1}^{n} (k_{j,2} - k_{j,1})$, w_k, the weight of k-th observation in a training set;
2. *k_per_class_closest_avg*: $C = arg \min_{C_i} \frac{1}{\sum_{k \in L_{m,C}} w_k} \sum_{k \in L_{m,C}} dist\,(k)$, where $dist\,(k) = \sum_{j=1}^{n} (k_{j,2} - k_{j,1})$, $L_{m,C}$ is the set of m elements from class C which are closest to the prediction observation, m is a hyperparameter;
3. *k_closest*: $C = arg \max_{C_i} \sum_{k \in L_m} w_k \cdot \mathbb{1}\,(k = C_i)$, where $dist\,(k) = \sum_{j=1}^{n} (k_{j,2} - k_{j,1})$, L_m, the set of m elements which are closest to the prediction observation regardless of their class, m is a hyperparameter;
4. *count_with_threshold_t*: $C = arg \max_{C_i} \sum_{k \in A_{C_i}} \mathbb{1} \left(\frac{dist(k)}{w_k} < t \right)$, where $dist\,(k) = \sum_{j=1}^{n} (k_{j,2} - k_{j,1})$, t, a threshold.

Note: in each case A_{C_i} is a set of intersections of the test observation with each observation in a class C_i, $A_{C_i} = \{x_{test} \sqcap x : (x \in X_{train}) \wedge (c(x) = C_i)\}$ if hypotheses are not used and $A_{C_i} = \{x_{test} \sqcap x_h : (x_h \in X_{train}) \wedge (c(x_h) = C_i)\}$ if hypotheses are used (specified in tables with results if they are used), $c(x)$ is a function which returns class of training object.

2.2 SAMME pattern structures

It is not possible to directly use the gradient boosting techniques as there is no loss function which gets directly optimized. Thus, an analog of AdaBoost is used as we know how to implement weighted datasets in pattern structures.

The general scheme called Stagewise Additive Modeling using a Multi-Class Exponential loss function or SAMME is presented in [3]. After adapting it to pattern structures it looks as follows:

```
Algorithm 1: SAMME for pattern structures analysis, training
  Input: $(X, y)$, training dataset, $M$ is the number of models in ensemble
    ;
  initialize $w_i = \frac{1}{n}$, $n$ - number of objects in $X$;
  for $m \in 1, ..., M$ do
      initialize model $T^{(m)}$;
      classify each observation in $X$ using weighted dataset with a new
        model;
      calculate error:
          $$err^m = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \mathbb{1} \left( y_i \neq T^{(m)}(x_i) \right)$$
        calculate model weight:
          $$\alpha^m = \log\left(\frac{1 - err^m}{err^m}\right) + \log(K - 1)$$
        where $K$ is the amount of classes;
      recalculate object weights:
          $$w_i \leftarrow w_i \cdot e^{\alpha^m \cdot \mathbb{1}\left(y_i \neq T^{(m)}(x_i)\right)}, \ i \in \{1, \ldots, n\}$$
  end
```

Prediction of ensemble:

$$C(x) = arg \max_k \sum_{m=1}^{M} \alpha^m \cdot \mathbb{1}\left(T^{(m)}(x) = k\right)$$

Methods of weighting models Additionally to the original method of calculation of α, others were used such as:

1. Uniform: $\alpha^m = \frac{1}{M}$
2. Linear: $\alpha^m = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i \mathbb{1}\left(y_i = T^{(m)}(x_i)\right)$
3. Exponential: $\alpha^m = \frac{\sum_{i=1}^{n} e^{w_i \mathbb{1}\left(y_i = T^{(m)}(x_i)\right)}}{\sum_{i=1}^{n} e^{w_i}}$
4. Logarithmic: $\alpha^m = \log(1 + (1 - err^m)) = \log(2 - err^m)$
5. Iternum: $\alpha^m = \frac{1}{m}$

3 Datasets description

3.1 Cardiotocography Data Set

The dataset consists of preprocessed 2126 fetal cardiotocograms (CTGs). It contains 23 numerical attributes with no missing values. It could be used for 3 class prediction as well as a 10 class classification (classes are more specific). The dataset is available here [7]. The dataset is unbalanced. Before the classification, data was standartized.

3.2 Male Fertility dataset

The dataset [8] consists of 9 features about patient health, injuries, lifestyle and bad habits. All features are nominal. There are 100 observations in the dataset. The final goal is binary classification: normal semen sample or sample with deviations.

3.3 Divorces

The dataset [9] consists of 54 features which are the answers to questions about relationship experience. All features are nominal. There are 170 observations in the dataset. The final goal is binary classification: got divorced or not.

4 Random Forest

For each dataset Random Forest was chosen as a baseline, since this algorithm is an efficient ensemble of decision trees (which are also good in explanation) [2].

In this algorithm the ensemble of Decision trees is built, where each tree is tuned using random subsample from the training sample (Bagging) and random subsample of features.

Gridsearch with crossvalidation was used to tune it.

5 Results

Accuracy, precision, recall and F1-score were chosen as quality metrics. For datasets with more than 2 target classes, precision, recall and F1-score were calculated for each class separately and averaged afterwards.

Metrics are measured on a randomly chosen test set.

SAMME was also run with the aggregation function k_per_class_closest_avg, since it has shown good performance.

Due to space limitations, we present only results of the most interesting experiments, together with the baseline model Random Forest.

In the table pattern structures are referred to as FCA and SAMME pattern structures is referred to as SAMME FCA.

Table 1. Cardiotocography Data Set (3 classes)

Algorithm	Accuracy	Precision	Recall	F1-score	Ensemble size
SAMME FCA ('k_per_class_closest_avg' original method)	0.934	0.892	0.833	0.858	5
FCA ('k_per_class_closest_avg')	0.934	0.892	0.833	0.858	1
Random Forest	0.925	0.867	0.839	0.852	100
SAMME FCA ('k_per_class_closest_avg' uniform method)	0.925	0.877	0.829	0.848	5
SAMME FCA ('k_per_class_closest_avg' linear method)	0.906	0.838	0.804	0.819	5
SAMME FCA ('k_per_class_closest_avg' exponential method)	0.761	0.582	0.630	0.602	5
SAMME FCA ('k_per_class_closest_avg' logarithmic method)	0.864	0.760	0.735	0.747	5
SAMME FCA ('k_per_class_closest_avg' iternum method)	0.897	0.828	0.784	0.803	5

Table 2. Cardiotocography Data Set (10 classes)

Algorithm	Accuracy	Precision	Recall	F1-score	Ensemble size
SAMME FCA ('k_per_class_closest_avg' original method)	0.784	0.803	0.680	0.712	5
FCA ('k_per_class_closest_avg' original method)	0.784	0.803	0.680	0.712	1
Random Forest	0.793	0.720	0.723	0.704	100
SAMME FCA ('k_per_class_closest_avg' uniform method)	0.765	0.727	0.636	0.66	5
SAMME FCA ('k_per_class_closest_avg' linear method)	0.681	0.609	0.591	0.594	5
SAMME FCA ('k_per_class_closest_avg' exponential method)	0.465	0.423	0.412	0.409	5
SAMME FCA ('k_per_class_closest_avg' logarithmic method)	0.728	0.652	0.623	0.629	5
SAMME FCA ('k_per_class_closest_avg' iternum method)	0.643	0.602	0.575	0.582	5

As it can be seen in Table 1 and Table 2, with original weighting the metrics are identical to the simple FCA one-model. This happens because the first model has a significantly bigger α^1 and dominates others in ensemble. That is why other model weightings are tested.

However, in both cases best SAMME FCA and FCA models have higher average F1-score than the tuned baseline and for the first case they also win in terms of accuracy. Comparing SAMME FCA models it can be seen that the original weighting is better in terms of the presented metrics even though it effectively uses the first model only. In both SAMME and FCA the ensemble size is smaller than in the random forest.

On the divorces dataset (Table 3) due to its size and simplicity Random Forest manages to come out as an absolute winner having 100% in every score. A lot of SAMME FCA and FCA models show the same relatively high scores. SAMME again uses only the first model with original weights. However, a lot of other weighting techniques have similar metric values on this dataset. Because of that even though random forest uses 5 models, FCA can use less models.

On the fertility dataset (Table 4) the tuned Random Forest wins again. Especially the difference is significant in F1-score, precision and recall. The behaviour of the SAMME FCA metrics is similar to the previous dataset: different model weighting methods give similar results. In both SAMME and FCA the ensemble size is smaller than in random forest.

Table 3. Divorce Predictors Data Set

Algorithm	Accuracy	Precision	Recall	F1-score	Ensemble size
FCA ”avglengths”	0.953	0.958	0.952	0.953	1
FCA ”k_per_class_closest_avg”	0.953	0.958	0.952	0.953	1
FCA ”k_closest”	0.953	0.958	0.952	0.953	1
FCA ”count_with_treshold_t”	0.674	0.806	0.667	0.629	1
FCA ”avglengths” (with hypotheses)	0.953	0.958	0.952	0.953	1
FCA ”k_per_class_closest_avg” (with hypotheses)	0.953	0.958	0.952	0.953	1
FCA ”k_closest” (with hypotheses)	0.953	0.958	0.952	0.953	1
FCA ”count_with_treshold_t” (with hypotheses)	0.512	0.256	0.500	0.338	1
SAMME FCA ”avglengths”	0.953	0.958	0.952	0.953	5
SAMME FCA original ”k_per_class_closest_avg”	0.953	0.958	0.952	0.953	5
SAMME FCA ”k_closest”	0.953	0.958	0.952	0.953	5
SAMME FCA ”count_with_treshold_t”	0.674	0.806	0.667	0.629	5
SAMME FCA ”avglengths” (with hypotheses)	0.953	0.958	0.952	0.953	5
SAMME FCA ”k_per_class_closest_avg” (with hypotheses)	0.953	0.958	0.952	0.953	5
SAMME FCA ”k_closest” (with hypotheses)	0.953	0.958	0.952	0.953	5
SAMME FCA ”count_with_treshold_t” (with hypotheses)	0.512	0.256	0.500	0.338	5
Random Forest	1.0	1.0	1.0	1.0	5
SAMME FCA uniform ”k_per_class_closest_avg”	0.953	0.958	0.952	0.953	5
SAMME FCA linear ”k_per_class_closest_avg”	0.953	0.958	0.952	0.953	5
SAMME FCA exponential ”k_per_class_closest_avg”	0.953	0.958	0.952	0.953	5
SAMME FCA logarithmic ”k_per_class_closest_avg”	0.953	0.958	0.952	0.953	5
SAMME FCA iternum ”k_per_class_closest_avg”	0.953	0.958	0.952	0.953	5

6 Conclusion

Even though the model itself seems promising, right now it has multiple issues. First, SAMME boosting technique does not give additional quality. The original SAMME method of calculating α effectively uses only the first classifier, while the other weighting methods do not give stronger metric values than the original method. The problem seems to be in the fact that classifiers with indices > 1 in ensemble are just not good enough. So, there is a reason while original methods stick to the first classifier instead of using all of them. While other weighting might use several parts of the ensemble, it does not improve the metrics, because the classifiers built after the first one have bad metrics most of the time. The potential room for improvement is to make consequent classifiers to produce a better quality results possibly by changing the way dataset gets weighted.

Secondly, we can see that this algorithm performed worse on several datasets. Even though it was better than Random Forest on the complex ones, there is still a room for improvement for simpler ones. This again can be done by improving the quality of the whole ensemble, which Random Forest seems to efficiently perform.

SAMME FCA works slower than Random Forest. However, SAMME FCA has much better explainability: it generates only 5 classifiers (in some cases

Table 4. Fertility Data Set

Algorithm	Accuracy	Precision	Recall	F1-score	Ensemble size
FCA "avglengths"	0.680	0.600	0.826	0.561	1
FCA "k_per_class_closest_avg"	0.920	0.460	0.500	0.479	1
FCA "k_closest"	0.920	0.460	0.500	0.479	1
FCA "count_with_treshold_t"	0.560	0.577	0.761	0.476	1
FCA "avglengths" (with hypotheses)	0.760	0.557	0.641	0.554	1
FCA "k_per_class_closest_avg" (with hypotheses)	0.080	0.272	0.272	0.080	1
FCA "k_closest" (with hypotheses)	0.920	0.460	0.500	0.479	1
FCA "count_with_treshold_t" (with hypotheses)	0.520	0.571	0.739	0.449	1
SAMME FCA "avglengths"	0.920	0.460	0.500	0.479	5
SAMME FCA "k_per_class_closest_avg"	0.920	0.460	0.500	0.479	5
SAMME FCA "k_closest"	0.920	0.460	0.500	0.479	5
SAMME FCA "count_with_treshold_t"	0.920	0.460	0.500	0.479	5
SAMME FCA "avglengths" (with hypotheses)	0.760	0.557	0.641	0.554	5
SAMME FCA "k_per_class_closest_avg" (with hypotheses)	0.800	0.455	0.435	0.444	5
SAMME FCA "k_closest" (with hypotheses)	0.920	0.460	0.500	0.479	5
SAMME FCA "count_with_treshold_t" (with hypotheses)	0.920	0.460	0.500	0.479	5
Random Forest	0.960	0.979	0.750	0.823	100
SAMME FCA uniform "k_per_class_closest_avg"	0.920	0.460	0.500	0.479	5
SAMME FCA linear "k_per_class_closest_avg"	0.920	0.460	0.500	0.479	5
SAMME FCA exponential "k_per_class_closest_avg"	0.920	0.460	0.500	0.479	5
SAMME FCA logarithmic "k_per_class_closest_avg"	0.920	0.460	0.500	0.479	5
SAMME FCA iternum "k_per_class_closest_avg"	0.920	0.460	0.500	0.479	5

effectively uses only one of them), while Random Forest consists of 5 trees only on Divorces dataset and consists of 100 trees in every other case.

References

1. Bernhard Ganter, Sergei O. Kuznetsov. Pattern Structures and Their Projections. In: Delugach H.S., Stumme G. (eds) Conceptual Structures: Broadening the Base. ICCS 2001. Lecture Notes in Computer Science, Volume 2120 (2001). Springer, Berlin, Heidelberg.
2. Leo Breiman. Random Forests. Machine Learning, Volume 45 (2001), 5–32.
3. Ji Zhu, Hui Zou, Saharon Rosset, Trevor Hastie. Multi-class AdaBoost. Statistics and Its Interface, Volume 2 (2009), 349–360.
4. Mehdi Kaytoue, Sergei O. Kuznetsov, Amedeo Napoli, Sébastien Duplessis. Mining gene expression data with pattern structures in formal concept analysis. Information Sciences, Volume 181 (2011), Issue 10, 1989-2001.
5. Sergei O. Kuznetsov. Fitting Pattern Structures to Knowledge Discovery in Big Data. In: Cellier P., Distel F., Ganter B. (eds) Formal Concept Analysis. ICFCA 2013. Lecture Notes in Computer Science, Volume 7880 (2013). Springer, Berlin, Heidelberg.
6. Sergei O. Kuznetsov. Scalable Knowledge Discovery in Complex Data with Pattern Structures. In: Maji P., Ghosh A., Murty M.N., Ghosh K., Pal S.K. (eds) Pattern Recognition and Machine Intelligence. PReMI 2013. Lecture Notes in Computer Science, Volume 8251 (2013). Springer, Berlin, Heidelberg.
7. UCI Machine Learning Repository. URL: https://archive.ics.uci.edu/ml/datasets/Cardiotocography [date of access: 27.08.2020].
8. UCI Machine Learning Repository. URL: https://archive.ics.uci.edu/ml/datasets/Divorce+Predictors+data+set [date of access: 21.03.2021].
9. UCI Machine Learning Repository. URL: https://archive.ics.uci.edu/ml/datasets/Fertility [date of access: 21.03.2021].