CoLe: A Cooperative Data Mining Approach and Its Application to Early Diabetes Detection

Jie Gao and Jörg Denzinger
Dept of Computer Science
University of Calgary, AB, Canada
{gaoj,denzinge}@cpsc.ucalgary.ca

Robert C. James
Aechidna Health Informatics
Winnipeg, MB, Canada
rob@aetiologic.ca

Abstract

We present CoLe*, a cooperative data mining approach for discovering hybrid knowledge. It employs multiple different data mining algorithms, and combines results from them to enhance the mined knowledge. For our medical application area, we analyse several focusing strategies that allowed us to gain medically significant results.*

1. Introduction

Data mining is facing a challenging situation where large, heterogeneous and complex data sets are to be mined. We have developed *CoLe* (*Co*operative *Le*arning) to handle data of this kind. It uses a multi-agent framework to run multiple data mining algorithms simultaneously and cooperatively. Results from these algorithms are combined into hybrid knowledge. Partial results are exchanged during the process of mining to maximize the synergetic effects in the cooperation. Other existing distributed/cooperative mining approaches, like [4, 6] and others, emphasize handling already distributed data sources flexibly and efficiently, while *CoLe* concentrates more on getting hybrid knowledge that cannot be generated by a single data mining algorithm.

2. Our Cooperative Mining Approach

The *CoLe* model employs a multi-agent system framework (see Figure 1). It works on a data set D using a mining agent (miner) set M and a combination agent $\mathrm{Ag_{CBN}}$. An individual miner m_i contains a data mining algorithm. Each m_i works on a dedicated sub data set D_i split from D to fit m_i's algorithm. Miners are synchronized to work in iterations. In each iteration, m_i creates a knowledge set K_i and sends it to $\mathrm{Ag_{CBN}}$ for combination. $\mathrm{Ag_{CBN}}$ puts good combined hybrid results in a final knowledge set K, and sends feedback (summary of discoveries in this iteration) to each m_i to help their later work. $\mathrm{Ag_{CBN}}$ also controls how D_i's are created for the next iteration. [3] described the *CoLe* model and its cooperation schemes in detail.

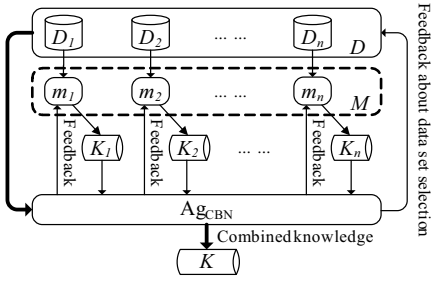

Figure 1. *CoLe* model

The most important aspect of *CoLe* is the cooperation of miners, coordinated by $\mathrm{Ag_{CBN}}$: it instructs how D_i's are generated from D; it receives results K_i from the miners; the final result K is produced by $\mathrm{Ag_{CBN}}$. $\mathrm{Ag_{CBN}}$ also synchronizes the iterative mining process. $\mathrm{Ag_{CBN}}$ collects useful information from each miner's results in one iteration and sends feedback to all miners to help them improve their mining in the next iteration — so, miners influence each others' work indirectly via $\mathrm{Ag_{CBN}}$.

3. A Mining Problem and Our Mining Agents

We applied *CoLe* to a problem occurring at the Calgary Health Region: Patients have their health harmed before laboratory tests can detect diabetes. Data mining may help to diagnose diabetes earlier. The diabetes data set contains patients' (both diabetics, called *cases*, and non-diabetics, called *controls*) basic information and their time-stamped medical record (diagnostic codes). The records are windowed by a 5-year monitoring period. Medical researchers are expecting rules showing both static conditions influencing diabetes and the development of the disease.

We instantiate *CoLe* for this medical mining problem with two miners and one $\mathrm{Ag_{CBN}}$. One miner ($\mathrm{m_S}$) uses a sequence mining algorithm to observe the development of diagnoses. The other ($\mathrm{m_C}$) uses a classification algorithm to get rules with conjunctions of diagnoses. Their discovered rule sets are R_S and R_C respectively. Correspondingly,

we split D into two sub sets. They are D_S, which contains temporal data suitable for m_S, and D_C, a flattened D with timestamps removed, for m_C. The Ag_{CBN} takes R_S and R_C, validates them against D, and uses combination strategies to combine them into *hybrid rules* (as rule set R) that contain both sequence and conjunctive conditions.

3.1. Sequence Mining by m_S

Our m_S uses a genetic algorithm for sequence mining. A sequence contains ordered *events* (diagnosis sets), which do not need to be consecutive in a matched record. The goal is to discover sequences to discriminate *cases* from *controls*. We do not use the well-known *Apriori*-based algorithms from [1] because our data amount is huge and the sequences can be long. With a GA, we also gain easy control of iterative mining and integration of feedbacks.

In the GA implementation, we use single sequences as individuals. The fitness is calculated by:

$$fitness = 10 \times \left(\frac{tp}{tp+fp}\right)^x \times \frac{\ln(tp)}{\ln(case\_num)} \qquad (1)$$

Here tp and fp are the counts of true (*case*) and false (*control*) positives respectively, $case\_num$ is the total number of *cases* in the data set, and x is a real-number parameter to control the weight of the two factors. This considers both the accuracy and significance of a sequence. It is also used globally as an evaluation of rules' quality.

In our GA, we use not only standard *mutation* and *crossover* but also two knowledge-based genetic operators: knowledge-based mutation and *IntelliCut*. In knowledge-based mutation, new events are chosen from sequence segments that frequently occur in *cases*. IntelliCut cuts off the "bad tail" of a sequence: for a sequence, each possible tail-cut is checked to find the "right" cut-point to maximize its fitness. This can remove low-quality parts from a potentially good sequence.

3.2. Conjunctive Rule Mining by m_C

In m_C we use a classification algorithm to discover conjunctive rules. There are quite a few efficient algorithms for this. We use an existing implementation, namely *PART* (see [2]). The *PART* algorithm is very suitable for m_C because it output results in our targeted format directly. And *PART* does not have global optimization. So we can interrupt it to suit out in our iterative *CoLe* mining process.

3.3. Combination Strategies in Ag_{CBN}

Ag_{CBN} uses both the data set and the already-discovered rules by the two miners in its tasks. The output is a hybrid rule set R. "Byproducts" are the feedbacks to the miners and instructions for generating the next D_S and D_C. The combination is done in 3 stages, namely *direct combination*, *cross combination* and *rule pruning*:

In direct combination we try to put a rule from R_S and one from R_C directly together. The new rule's condition is the conjunction of the two old ones. If the new one is a good rule according to Equation (1), we will put it into R.

The cross combination is to check if we can use segments of sequence (classification) rules to combine with classification (sequence) rules and get good hybrid rules. For each event e in a sequence rule, we convert it to a predicate $e = true$, and add it to each existing rule's condition to see if we can gain good new rules. This process is done repeatedly until no more new good rules can be obtained. We also try converting predicates to events.

We also prune rules in R to remove redundant parts. The following is done: 1) convert a single-event sequence condition to a predicate; 2) remove a predicate that also occurs in the sequence condition; 3) remove duplicate rules.

Ag_{CBN} also generates feedback for m_S and m_C, according to the results of the combination. For m_S, Ag_{CBN} mainly gathers diagnoses from R_C, and enumerates their permutations. These permutations can then be used by m_S as good sequence segments. For m_C, events from R_S are sent, to be used for m_C to focus its D_C (see Section 4.3).

4. Data Reduction and Focusing of the Mining

Due to the large size and sparseness of the diabetes data, we have used several focusing strategies, including *feature aggregation*, *feature selection* and *instance reduction*.

4.1. Feature Aggregation

Feature aggregation is a static step before our *CoLe* system starts its work. We reduce the number of possible diagnostic codes in the aggregation. The over 17000 possible diagnostic codes are aggregated into 307 disease groups. We also have a temporal aggregation to put all diagnoses in a given period (e.g. a week) into the same event.

4.2. Instance Reduction

To make the miners focus on the instances we are interested in and to prevent miners from discovering the same set of rules over and over again in their iterations, we dynamically do *instance reduction* to get D_S and D_C, to give the miners a smaller and focused work data set. In each iteration, D_S and D_C use the same group of patients, so that it is meaningful to combine R_S and R_C.

The core work is to decide on a set of patients I in each iteration. For an instance set I_i in iteration i, it is generated based on I_{i-1} and the results of iteration $(i-1)$. We take from I_{i-1} patients that are not covered by any rules in iter-

ation $(i-1)$, together with randomly chosen patients from D, to form I_i. This makes miners focus more on patients without covering rules.

4.3. Feature Selection

In m_C, we also do a *feature selection* to further reduce the search space. The features are selected by their *relevance factors* against the *case*/*control* class label. We use Equation 2, inspired by the work in [5], to calculate each feature's *relevance factor*.

$$RF(A) = \Pr(A) \times \log\left(\frac{\Pr(A|case)}{\Pr(A|control)}\right) \qquad (2)$$

A is a feature, and probabilities are estimated by frequencies. The absolute value of $RF(A)$ reflects the relevance of A. Positive value means an indication of *cases*, while an indicator of *controls* gets a negative value. When we select the relevant features, a dynamic threshold t_{RF} is used. This suits our situation better because D_C is dynamically generated, thus we can not presume an arbitrary threshold. The use of feedbacks in m_S can also be taken as a *feature selection* strategy.

5. Experiments

We conducted several experiments with our implementation of *CoLe*. The focus is to evaluate the advantages of our new cooperative model over the individual algorithms, the effect of our focusing strategies, and the significance of the results for medical research. Run time is not a test focus here, because it is acceptable to have a run time as long as 2–3 days in the public health data mining with large data sets (and all our single tests finished within 12 hours, which is quite acceptable).

5.1. Effects of Cooperation

To evaluate the effects of cooperation, we have run *CoLe* with different fitness thresholds, while all other parameters are kept the same. The fitness values listed in Table 1 show that not only the average fitness of hybrid rules is higher, but also the maximum fitness value of hybrid rules is higher. But when we have too high a fitness threshold, the hybrid rule quality decreases, because this results in less candidates from miners for later combination stages, and consequently limits the possibilities for combination.

Table 2 presents the detailed fitness values over iterations in one of the tests. We also list the average fitness of the top 10 fittest rules. Most importantly, the average fitness of the top hybrid rules is significantly higher than the average fitness of the top rules from the individual miners. This shows that combination plays a key role in enhancing the quality of knowledge discovered by individual miners.

Table 1. Individual algorithm vs. cooperation

Test No.	1	2	3	4
Fitness threshold	3.6	3.7	3.8	3.9
Average fitness from m_S	3.10	2.86	3.24	3.05
Average fitness from m_C	2.58	2.51	2.62	2.44
Average hybrid rule fitness	**3.72**	**3.82**	**3.89**	**3.91**
Max fitness from m_S	3.72	3.74	3.74	3.72
Max fitness from m_C	3.72	3.78	3.72	3.72
Max hybrid rule fitness	**4.29**	**4.33**	**4.12**	**3.91**

Table 2. Average fitness over iterations in Test 2

A: Average fitness of all rules in an iteration;
T: Average fitness of top 10 (all if less than 10) rules in an iteration;
(S): Rules from m_S; **(C)**: Rules from m_C; **(H)**: Hybrid rules

Iter.	A(S)	A(C)	A(H)	T(S)	T(C)	T(H)
1	0.78	2.74	3.71	2.40	3.39	3.71
2	3.25	2.45	3.73	3.65	3.22	3.74
3	2.54	2.38	3.72	3.52	3.20	3.72
4	3.49	2.73	3.72	3.65	3.27	3.73
5	3.22	2.20	3.73	3.64	3.42	3.73
6	3.22	2.17	3.72	3.62	3.28	3.72
7	3.08	2.09	3.75	3.59	3.31	3.78
8	2.48	2.48	3.75	3.60	3.44	3.75
9	3.08	2.61	3.72	3.62	3.21	3.72
10	3.41	2.62	3.72	3.65	3.34	3.73
11	3.43	2.57	3.73	3.64	3.33	3.73
12	3.39	2.05	3.76	3.66	3.50	3.81
13	2.78	2.76	3.88	3.64	3.57	4.23
14	2.57	2.93	3.80	3.53	3.57	3.89
15	2.69	2.51	3.72	3.61	3.21	3.72
16	2.81	2.94	3.80	3.60	3.28	3.90
17	2.41	2.46	3.81	3.52	3.20	4.00
18	2.51	2.63	3.72	3.57	3.33	3.72
19	2.60	2.73	3.72	3.56	3.27	3.72
20	3.51	2.20	3.73	3.65	3.30	3.73

5.2. Effects of Focusing

While the help of *feature aggregation* is obvious, we performed detailed tests on the *instance reduction* (IR) and *feature selection* (FS).

Our experiment with IR is a comparison between a normal *CoLe* implementation (with IR) and an implementation without IR, i.e., m_S and m_C run on the entire data set instead of D_S and D_C. All other parameters are the same. The run times in Table 3(a) show that the implementation with IR ran much faster than the one without. The maximum and average fitness values in Table 3(b) are comparable.

In m_S, FS is used to get feedback (hints) from Ag_{CBN}, and use those sequence segments as good materials to construct sequence rule individuals. We ran two experiments, one with FS and the other without it. The average fitness values of the first 10 generations are plotted in Figure 2. It is obvious that the m_S with hints has a faster increase in average fitness.

In m_C, FS is to select relevant features into D_C. Our experiments consist of several test runs with two implementations, one with FS and the other without it. Table 4 reports 5 test runs for each implementation. The m_C without FS runs 30-100 times longer than m_C with it. This proves that our FS strategies help eliminate irrelevant data and let m_C focus on the rules that we are interested in.

Table 3. Test on *instance reduction* (IR)

(a) Run time comparison (seconds)

	With IR	Without IR
Overall	18098	42606
Average run time per iteration		
m_S	269.85	2334.90
m_C	83.20	1759.25
Ag_{CBN}	622.90	1924.90

(b) Mining result quality comparison (fitness)
S: Rules from m_S; **C**: Rules from m_C; **H**: Hybrid rules

	With IR	Without IR
S rules average	2.86	2.91
C rules average	2.51	2.81
H rules average	3.82	3.85
S rules max	3.74	3.72
C rules max	3.78	3.72
H rules max	4.33	4.36

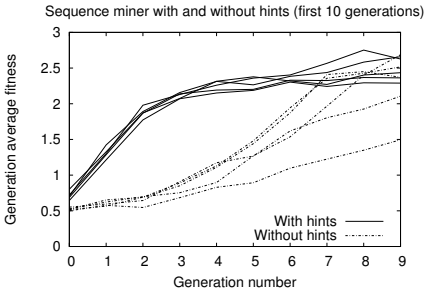

Figure 2. Average fitness of individuals over generations in m_S (with and without hints)

5.3. Medical Significance

An example for a discovered rule is in Table 5: A patient is likely to have diabetes if he/she was born after 1939, has "other diseases of the respiratory system", has "diseases of skin and subcutaneous tissue", and diagnoses in the temporal order: repeatedly hypertension diagnoses, and some general uncomfortable symptoms in between. We presented the results to medical experts for their opinion and got very positive feedbacks. The medical significance is mainly in the following aspects.

Firstly, in our hybrid rules, almost 100% of them contain "hypertensive disease" conditions. This confirms an already known fact that hypertension has a very tight relation with diabetes. There are also many other diabetes-related diagnoses, e.g. diseases of skin and subcutaneous tissue. This is a strong indication that *CoLe* is able to produce knowledge valid to medical research. However, hypertension alone is not a good indicator for diabetes, showing the need for complex rules like our hybrid rules.

Secondly, there are phenomena new to medical experts. The diagnosis "other diseases of the respiratory system" is one example. It appears frequently in the hybrid rules. Medical experts could not recall immediately how it is related to diabetes. But it seems interesting to them, and there seem to be some explanations.

Thirdly, our results urge the public health services to improve their quality. There are quite a few conditions like

Table 4. Run time (seconds) of *PART* algorithm in m_C (with and without *feature selection*(FS))

Test	With FS	Without FS
1	19.416	1855.356
2	49.304	1638.292
3	32.436	1704.105
4	29.663	2205.108
5	40.456	1877.903
Average	34.255	1856.153

Table 5. A representative rule

Part	Condition	Description
Conj.	yofb> 1939	Year of Birth
	{466,480-519}=1	Other diseases of the respiratory system
	{680-686}=1	Diseases of skin and subcutaneous tissue
Seq.	{401-405}	Hypertensive disease
	{780-799}	Signs, symptoms and ill-defined conditions
	{401-405}	Hypertensive disease
	{401-405}	Hypertensive disease

"signs, symptoms and ill-defined conditions". This can only indicate that the patient has uncomfortable feelings, which are indicators of diseases. Therefore we should improve them to be more specific. They also reveal the potential of diagnosing diabetes earlier.

6. Conclusion

We proposed the *CoLe* cooperative data mining approach. It is a multi-agent system framework with multiple miner agents and a combination agent. The main goal of *CoLe* is to get hybrid knowledge that can describe given data from multiple aspects. Our application of *CoLe* was mining medical data on diabetes. The results prove that our *CoLe* approach and our combination and focusing strategies are efficient and promising. And we have discovered some rules that are of interest to the medical researchers. Future work will be aimed at using *CoLe* in other areas and enhancing the diabetes application.

References

[1] R. Agrawal and R. Srikant. Mining sequential patterns. *Proc. 11th Intl. Conf. on Data Eng.*, IEEE, 1995, pp. 3–14.

[2] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. *Proc. 15th Intl. Conf. on Machine Learning*, Morgan Kaufmann, 1998, pp. 144–151.

[3] J. Gao, J. Denzinger, and R. C. James. A cooperative multi-agent data mining model and its application to medical data on diabetes. *Proc. AIS-ADM*, LNAI 3505, 2005, pp. 93–107.

[4] H. Kargupta, I. Hamzaoglu, and B. Stafford. Scalable, distributed data mining - an agent architecture. *Proc. KDD-97*, 1997, pp. 211–214.

[5] H. Liu, H. Lu, and J. Yao. Toward multidatabase mining: Identifying relevant databases. *IEEE Trans. Knowledge Data Eng.*, 13(4):541–553, 2001.

[6] S. J. Stolfo, A. L. Prodromidis, S. Tselepis, W. Lee, D. W. Fan, and P. K. Chan. JAM: Java agents for meta-learning over distributed databases. *Proc. KDD-97*, 1997 pp. 74–81.