Mining Several Databases with an Ensemble of Classifiers

Seppo Puuronen

University of Jyväskylä
P. O. Box 35, FIN-40351 Jyväskylä Finland
phone: + 358 14 603 028;
fax +358 14 603 011
e-mail: sepi@jytko.jyu.fi

Vagan Terziyan

Kharkov State Technical University of
Radioelectronics, 14 Lenin Avenue,
310166 Kharkov, Ukraine
e-mail: vagan@kture.cit-ua.net

Alexander Logvinovsky

Kharkov State Technical University of
Radioelectronics, 14 Lenin Avenue,
310166 Kharkov, Ukraine;
e-mail: vagan@kture.cit-ua.net

Abstract

The results of knowledge discovery in databases could vary depending on the data mining method. There are several ways to select the most appropriate data mining method dynamically. One proposed method clusters the whole domain area into competence areas of the methods. A metamethod is then used to decide which data mining method should be used with each database instance. However, when knowledge is extracted from several databases knowledge discovery may produce conflicting results even if the separate databases are consistent. At least two types of conflicts may arise. The first type is created by data inconsistency within the area of the intersection of the databases. The second type of conflicts is created when the metamethod selects different data mining methods with inconsistent competence maps for the objects of the intersected part. We analyze these two types of conflicts and their combinations and suggest ways to handle them.

1. Introduction

The modern database technology enables the storage of huge amounts of data, but it does not yet offer high level support to analyze, understand, or visualize the stored data in intelligent ways. Data mining is a nontrivial knowledge extracting process which reveals valid, previously unknown, and comprehensive information from databases discovering useful patterns that are not directly obvious to the user [4]. The research in the field of knowledge discovery in databases has rapidly emerged recently producing several new data mining methods and techniques of their evaluation, learning and integration [3,5,6,9,11,10]. Some of these methods are static and do not analyze a new instance in its context. Dynamic data mining methods, as [8,14,13], take into account the context of a new instance and are even able to take benefit from an ensemble of classifiers. Most of the data mining algorithms assume a single data set, but real world application practitioners have usually

to discover knowledge from several databases [7]. Thus knowledge discovery can be seen as a task to apply several data mining methods with several data bases.

The brute-force application of statistical methods does not help, because they are unable to take into account their own context and the context of the knowledge they are processing. We develop methods that can take into account inconsistencies of numerical data when the most appropriate method(s) is chosen. One technique to handle a single database using dynamic integration of multiple classifiers is proposed in [12] and developed further in [13]. The technique consists of two phases: the training phase and the classification phase. During the training phase the characteristics and classifications of the training instances are collected using the Jackknife method into the performance matrix $Q_{n \times m}$, where n is the number of the training instances, m is the number of the classification methods, and q_{ji} is equal to 1 if the classification produced by the method i for the instance j is incorrect and 0 otherwise. In the classification phase a new instance is classified using the weighted k -nearest neighbor algorithm. The weights of the k -nearest neighbors are calculated as a function of the distances between the neighbors and the new instance and the most appropriate method is selected based on the weights and the values of the matrix $Q_{n \times m}$. In a way the performance matrix is used as a competence map of the classifiers included in the ensemble.

The above technique is planned to be used with a single data base and it cannot be applied as such with several data bases because it cannot handle conflicts arising within intersecting areas of data bases. There are at least two types of problematic conflicts. The first type of conflicts arises when there exist different classification results caused by data inconsistency within the intersecting part of the databases. The second type of conflicts arises when the technique selects different data mining methods (classifiers) for the instances of the intersected part and there is inconsistency between the competence maps of the classification methods.

Chan [1,2] has proposed the arbiter meta-learning approach which uses conflicting intersection area of two subsets of the learning set to generate more accurate classification result. It is obvious that when the arbiter (classifier) is derived only a subset of the whole data is taken into account. Thus a classifier of the lowest level in the classifier structure handles the context of a single subset only. The classifiers of the higher levels of the structure are built using the union of the subsets of the lower levels. Thus the higher a classifier is in the structure, the bigger amount of the contexts of the subsets it handles. This is very similar with the above technique the main difference between those two ones being that a group of samples not a single sample is considered. In this paper we generalize both the above approach to take into account the contexts of the data bases during classification.

In Chapter 2 we analyze the inconsistencies of the competence maps and the data and present ways to handle them for three cases: one database and an ensemble of classifiers, several databases and one classifier, and an ensemble of classifiers and several databases. In Chapter 3 we describe one approach to handle the inconsistency of data by decontextualizing it. We end up with conclusions in Chapter 4.

2. Data mining with an ensemble of classifiers and several databases

Let there be n databases and a training set for each database including k instances with the attributes x_j , $j=1, \dots, k$ and the known classification result $c_j = \text{class}(x_j)$. Let there also be m classifiers that can be applied to the training sets. The classification results of the classifiers may differ over the same training set and the results of each classifier may differ over the training sets of different databases. In the general case all m classifiers can be applied over n databases. Let there be a new unclassified instance. The primary problem is to develop a method which classifies the new instance taking into account all the training sets and all the classifiers (Figure 1).

Figure 1. The data mining problem when several classifiers are used with several databases

We discuss first the case when there are m classifiers and one database. Then we discuss the case when there is one classifier and n databases. Last we discuss the general case with several classifiers and several databases. The trivial case of one classifier and one training set is not discussed.

2.1 An ensemble of classifiers and one database

Let there be m classifiers and one database (Figure 2).

Figure 2. A case when there are an ensemble of classifiers and one database

We apply the approach suggested in [12] to integrate classifiers based on the performance matrix of the classifiers. The integration technique selects the most appropriate component classifier from the ensemble. The advanced dynamic integration of multiple classifiers in [13] is one variation of the stacked generalization method using the assumption that each component classifier is best inside certain areas of the application domain. In the learning phase a performance matrix of each component classifier is derived and these matrixes are used in the application phase to predict the performances of each component classifiers with a new instance. We present here another variation that takes into account the degree of the error made by the classification method in numerical classification cases.

We change only the step 4 of the above mentioned Jackknife method and after change the learning phase consists of the following five steps:

- 1) Unmark all the instances of the training set.
- 2) Take an unmarked instance j from the training set. If there is no unmarked instance then stop.
- 3) Derive the classification result for the instance j using the remaining training set (without instance j) as the

input to the method i for each statistical method ($i=1, \dots, m$).

- 4) Give q_{ji} value e_{ji} where the value e_{ji} is an absolute difference between the classification result of the step 3 for method i and the value of the classification result c_j given for the training instance j in the training set.
- 5) Continue from the step 2.

Let us denote the weight of the classification method i over the instance j as ω_{ji} . There are many different possibilities to define the weights. The only requirement is that the weight ω_{ji} should satisfy the following conditions: $\omega_{ji} \approx 1$ when $e_{ji} \approx 0$ and $\omega_{ji} \approx 0$ when $e_{ji} \approx \pm \infty$. We propose the following function to calculate the weights:

$$w = \frac{1}{1 + e^2} \quad (0.1)$$

The weight of a classification method for a new instance is calculated by inter- or extrapolating the errors of the training set instances and then the weight is evaluated using this error value.

Let us denote the error function of the classification method i for the instance x as $e^i(x)$ and the weight function of the classification method i for the instance x as $\omega^i(x)$. The function $\omega^i(x)$ can be interpreted as a competence map of the method i . The competence maps of all the classification methods together form a competence map of the database.

An example of the competence map for one-dimensional instance x is shown in Figure 3. The curves are obtained calculating first the error values for all the instances in the training set and interpolating these error values to the whole space of the x parameter.

Figure 3. An example of the competence map for three classifiers with one-dimensional instance database

The most appropriate classification method for a new instance y is selected in the following way. First the values of the functions $\omega^i(x)$, $i=1, m$ are calculated and then the classification method giving the highest value is selected. When there is a tie one of the classification methods with the highest value is selected randomly. In the above example the intervals AB and CD are the competence areas of the method 2 and for example in the point C both the method 2 and the method 3 are as competent.

This selection mechanism can be easily adapted to take into account all the classification methods so that each method effects according its weight using the following formula to calculate the final result y :

$$y = \frac{\sum_i y^i w^i}{\sum_i w^i}, \quad (0.2);$$

where y^i is the value obtained by the method i and w^i is the weight of the classification method i .

Thus the weights of the classification methods over the instances can be either used to select the most appropriate classifier or to calculate the final result taking into account the predictions of all the available classifiers.

2.2 One classifier and several databases

Let there be one classifier to be applied over n databases (Figure 4).

Let us denote the classification result of the method for the instance x over the database j as $y_j(x)$. The weight of the classifier for the instance x over the database j is denoted $\omega_j(x)$. The integral weight of the classifier over the database j can be taken as the mean weight:

$$n_j = \frac{1}{b-a} \int_a^b \omega_j(x) \quad (0.3)$$

where a and b are the boundaries of the instances, $\omega_j(x)$ is the weight of the classifier over the instance x of the database j , n_j is the integral weight of the classifier over the database j , and the integral sign means the sum of the weights of all training instances of the database. This integral weight is used to estimate the performance of the classifier over each database separately (Figure 5).

Figure 4. A case when there is one classifier and several databases

This mechanism can be easily adapted to take into account all the databases in the evaluation of a classifier using the following formula to calculate the final weight y :

$$y = \frac{\sum_j y_j \mathbf{n}_j}{\sum_j \mathbf{n}_j}, \quad (0.4)$$

where y_j is the value obtained by the classifier over the instance x of the database j and $\mathbf{n}_j$ is the integral weight of the classifier over the database j .

Figure 5. A classifier's performance estimation with multiple databases

This is one possible approach to resolve data inconsistency between separate databases either using the mean result or the using the condition $\mathbf{n} = \max \mathbf{n}_j$ to select the most representative one.

2.3 Several classifiers and several databases

Let there be m classifiers to be applied over n databases (Figure 6).

Figure 6. General case when there is an ensemble of classifiers and several databases

Let us denote the classification result of the classifier i for the instance x over the database j as $y_j^i(x)$. The weight of the value and the integral weight of the classifier are denoted $\mathbf{w}_j^i(x)$ and $\mathbf{n}_j^i$ respectively.

The variants of the application of the classifiers to the databases and the appropriate weights are shown in the Table 1.

Table 1. Weighting classifiers in the general case

	Classifier ₁	...	Classifier _m	
DB ₁	$y_1^1, \mathbf{w}_1^1(\mathbf{n}_1^1)$	...	$y_1^m, \mathbf{w}_1^m(\mathbf{n}_1^m)$	$y_1, \mathbf{w}_1$
DB ₂	$y_2^1, \mathbf{w}_2^1(\mathbf{n}_2^1)$	...	$y_2^m, \mathbf{w}_2^m(\mathbf{n}_2^m)$	$y_2, \mathbf{w}_2$
...	...	...	...	...
DB _n	$y_n^1, \mathbf{w}_n^1(\mathbf{n}_n^1)$	...	$y_n^m, \mathbf{w}_n^m(\mathbf{n}_n^m)$	$y_n, \mathbf{w}_n$
	$y^1, \mathbf{w}^1$	...	$y^m, \mathbf{w}^m$	$y, \mathbf{w}$

Each row of the table describes the results and weights of all classifiers over a single database. The total classification result y_j and the weight $\mathbf{w}_j$ of the database j can be calculated by the following formulas:

$$y_j = \frac{\sum_{i=1}^m y_j^i \mathbf{w}_j^i}{\sum_{i=1}^m \mathbf{w}_j^i}, \quad \mathbf{w}_j = \frac{\sum_{i=1}^m \mathbf{w}_j^i \mathbf{n}_j^i}{\sum_{i=1}^m \mathbf{n}_j^i}, \quad (0.5)$$

where $y_j, \mathbf{w}_j$ are the total classification result and the weight of the database j and $y_j^i, \mathbf{w}_j^i, \mathbf{n}_j^i$ are the classification result, the weight and the integral weight of the classifier i over the database j respectively.

Each column of the table describes the results and the weights obtained by one classifier over all the databases. The classification result y^i and the weight $\mathbf{w}^i$ obtained by the classifier i over all the databases can be calculated as follows:

$$y^i = \frac{\sum_{j=1}^n y_j^i \mathbf{n}_j^i}{\sum_{j=1}^n \mathbf{n}_j^i}, \quad \mathbf{w}^i = \frac{\sum_{j=1}^n \mathbf{w}_j^i \mathbf{n}_j^i}{\sum_{j=1}^n \mathbf{n}_j^i}, \quad (0.6)$$

where $y^i, \mathbf{w}^i$ are the total classification result and the weight evaluated by the classifier i over all the databases respectively and $y_j^i, \mathbf{w}_j^i, \mathbf{n}_j^i$ are the classification result, the weight and the integral weight of the classifier i over the database j respectively.

The result value y and the weight $\mathbf{w}$ in the bottom-left cell of the table are obtained as a combination of the

classification results of all the classifiers over all the databases.

Thus there exists at least two possibilities to handle the general case as it is shown in Figure 7. The first one uses formulas (2.5) to convert the general case to the case “one database and an ensemble of classifiers”. The second one uses formulas (2.6) to convert the general case to the case “one classifier and several databases”. Both cases lead to the trivial problem “one database and one classifier” (more precisely “one virtual database and one virtual classifier”) as it is shown in Chapters 2.1 and 2.2.

Figure 7. Two ways to handle the general case

There exist some variants of how to determine the total classification result and the weight over all the databases using several classifiers. At least the next four ones can be recognized:

Evaluate classification results y^i and weights $\mathbf{w}^i$, $i=(1,m)$, using formulas (2.6) (i.e. through the columns of the matrix) and then determine the total result with the formula:

$$y = \frac{\sum_{i=1}^m y^i \mathbf{w}^i}{\sum_{i=1}^m \mathbf{w}^i} \quad (0.7)$$

Evaluate classification results y_j and weights $\mathbf{w}_j$, $j=(1, n)$, using formulae (2.5) (i.e. through the rows of the matrix) and then determine the total result with the formula:

$$y = \frac{\sum_{j=1}^n y_j \mathbf{w}_j}{\sum_{j=1}^n \mathbf{w}_j} \quad (0.8)$$

Evaluate the total result based on the values obtained by the two previous methods (2.7, 2.8), for example as a mean value.

Evaluate the result through the whole matrix at once:

$$y = \frac{\sum_{i,j} y_j^i \mathbf{w}_j^i}{\sum_{i,j} \mathbf{w}_j^i} \quad (0.9).$$

3. The Sliding Exam Technique with Decontextualization

In this chapter we present one algorithm to improve the Jackknife (sliding exam or leave one out cross-validation) method used in the learning phase with an ensemble of classifiers (see 1, 2.1 and [12]). Let us denote the classification result y^+ , when all the instances of the training set as are used and y^i when the classification is made excluding the instance i . It is obvious that y^+ depends on the contexts of all the training set instances and that the classification result y^i does not depend on the context of the instance i . Comparison of y^i and y^+ can be used to determine decontextualization trend of the database instances.

For the decontextualization process of the values describing the trend of a context decrease when we use the following formula:

$$y_{res} = \frac{y_i \cdot y_j}{y_i + y_j} \quad (0.1)$$

The main property of (3.1) is that the resulting value of decontextualization y_{res} is smaller than the values y_i and y_j .

Thus when the value y^+ which depends on the contexts of all the instances and the value y^i which depends on the contexts of all the other instances than i are known for an instance y then we can find decontextualized value y' using the above formula:

$$y' = \frac{y^+ \cdot y^{i-}}{y^+ + y^{i-}}. \quad (0.2)$$

It is possible to take into account all the subcontexts that can be formed from the training set using the following formula:

$$y^- = \frac{\prod_i y^{i-}}{\prod_i y^{i-} \cdot \sum_j \left(\frac{1}{y^{j-}} \right)} = \frac{1}{\sum_j \left(\frac{1}{y^{j-}} \right)}; \quad (0.3)$$

$$y' = \frac{y^+ \cdot y^-}{y^+ + y^-} = \frac{1}{\frac{1}{y^+} + \sum_j \left(\frac{1}{y^{j-}} \right)}. \quad (0.4)$$

The difference between the decontextualized value and the precise classification value y is:

$$\mathbf{e} = y - y' \quad (0.5)$$

which can be calculated for all the instances y_i of the training set.

This evaluation of differences between the decontextualized and precise values enables the improvement of the precision of the statistical methods at any instance x applying the following procedure:

Evaluate the decontextualized values y_i and the differences $\mathbf{e}_i$ for every training set instance $i, i=1, \dots, k$, where k is the number of the instances in the training set.

Evaluate the decontextualized value $y(x)$ for instance x as follows

$$y'(x) = \frac{1}{\frac{1}{y^+(x)} + \sum_j \left(\frac{1}{y^{j-}(x)} \right)}; \quad (0.6)$$

Evaluate the difference $\mathbf{e}(x)$ for the instance x as the result of interpolation on the set of the values $\mathbf{e}_1, \dots, \mathbf{e}_k$, where k is the number of the instances in the training set.

Evaluate $y(x)$ using the formula $y(x) = y'(x) + \mathbf{e}(x)$.

In the case of single database, $\mathbf{e}(x)$ function determines the context of the database. Let us call it as a self-difference. By analogy with the process of finding the self-difference we can find inter-difference between two databases as follows

$$\mathbf{e}_{ij} = y_j - y'_i, \quad (0.7)$$

where y_j is the precise value in the database j , y'_i is the decontextualized value in the database i , and $\mathbf{e}_{ij}$ is the mutual difference between the databases.

This inter-difference $\mathbf{e}_{ij}$ determine the context of a database j in terms of the database i . Hence based on the values of the database i it is possible to derive corresponding values in the context of the database j by:

$$y_i(x) = y'_j(x) + \mathbf{e}_{ij}(x). \quad (0.8)$$

4. Conclusion

This paper describes an approach to handle inconsistencies with multiple databases. We distinguish two types of inconsistencies: data inconsistency and competence map inconsistency. We have considered different cases of applying methods over databases and suggested ways to determine the total classification result based on the results of all methods over each database. We have also shortly considered decontextualization

process which improves the correctness of applying the classification method and to handle the mutual inconsistencies between databases. The results presented here are mostly theoretical ones and good real world examples are needed to verify the usefulness of the suggested methods.

5. Acknowledgment

This research is partly supported by the grant from the Academy of Finland.

6. References

- [1] P. Chan, An Extensible Meta-Learning Approach for Scalable and Accurate Inductive Learning, Ph.D. Thesis, Columbia University, 1996.
- [2] P. Chan and S. Stolfo, On the Accuracy of Meta-Learning for Scalable Data Mining. *Intelligent Information Systems* 8 (1997) 5-28.
- [3] T. Dietterich, Machine Learning Research: Four Current Directions. *AI Magazine* 18 (1997) 97-136.
- [4] U. Fayyad *et al.*, Advances in Knowledge Discovery and Data Mining. AAAI/ MIT Press, 1997.
- [5] R. Kohavi, A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In: Proceedings of IJCAI'95, 1995.
- [6] M. Koppel and S. Engelson, Integrating Multiple Classifiers by Finding their Areas of Expertise. In: AAAI-96 Workshop On Integrating Multiple Learning Models, 1996, pp. 53-58.
- [7] H. Liu, H. Lu, J. Yao, Identifying Relevant Databases for Multidatabase Mining. In: Proceedings of the PAKDD'98, Melbourne, Australia, Springer Verlag, 1998.
- [8] C. Merz, Dynamical Selection of Learning Algorithms. In: D. Fisher, H.-J. Lenz (Eds.), Learning from Data, Artificial Intelligence and Statistics, Springer Verlag, NY, 1996.
- [9] G. Nakhaeizadeh and A. Schnabl, Development of Multi-Criteria Metrics for Evaluation of Data Mining Algorithms. In: Proceedings of the KDD'97, 1997, pp. 37-42.
- [10] R. Schapire, Using Output Codes to Boost Multiclass Learning Problems. In: Machine Learning: Proceedings of the Fourteenth International Conference, 1997, pp. 313-321.
- [11] D. Skalak, Combining Nearest Neighbor Classifiers. Ph.D. Thesis, Dept. of Computer Science, University of Massachusetts, Amherst, MA, 1997.

[12] V. Terziyan, A. Tsymbal and S. Puuronen, The Decision Support System for Telemedicine Based on Multiple Expertise. *International Journal of Medical Informatics* 49 (1998) 217-229.

[13] A. Tsymbal, S. Puuronen and V. Terziyan, Advanced Dynamic Selection of Diagnostic Methods. In: Proceedings of the CBMS'98, IEEE CS Press, Lubbock, Texas, 1998, pp. 50-54.

[14] D. Wolpert, Stacked Generalization. *Neural Networks* 5 (1992) 241-259.