A Technique for Advanced Dynamic Integration of Multiple Classifiers

Alexey Tsymbal *, Seppo Puuronen **, Vagan Terziyan *

* Meta-Intelligence Lab., Kharkov State Technical University of Radio-electronics,
14 Lenina Av., 310726 Kharkov, Ukraine,
e-mail: alexey@milab.kharkov.ua, vagan@milab.kharkov.ua

** Department of Computer Science and Information Systems, University of Jyväskylä,
P.O.Box 35, SF-40351 Jyväskylä, Finland,
e-mail: sepi@jytko.jyu.fi

Abstract

Currently electronic data repositories are growing quickly and contain huge amount of data from commercial, scientific, and other domain areas. Knowledge discovery in databases (KDD) is an emerging area that considers the process of finding previously unknown and potentially interesting patterns and relations in large databases. Most current KDD systems offer only isolated discovery techniques, and very few systems use a combination of the available discovery techniques. Our goal is to design an architecture of an integrated knowledge discovery management system (IKDMS), which enables integration of multiple discovery techniques forming a platform upon which different KDD applications can be build. In this paper our focus is on the method evaluation/selection subsystem of an IKDMS. This subsystem is very important in any IKDMS because it helps a user to select an appropriate data mining method among the supported ones. We present and evaluate a technique for advanced dynamic integration of multiple classifiers that is based on the assumption that each classifier is the best only inside certain sub domains of the whole application domain. We have made experiments using three databases included in the University of California Machine Learning Repository achieving promising results either in diagnosis accuracy or in the time requirements of diagnostics or both.

1 Introduction

Currently electronic data repositories are growing quickly and contain huge amount of data from commercial, medical, scientific, and other domain areas. The capabilities for collecting and storing all kinds of data totally exceed the development in abilities to analyze, summarize, and extract knowledge from this data. Knowledge discovery in databases (KDD) is a combination of data warehousing, decision support, and data mining and it is an innovative new approach to information management. KDD is an emerging area that considers the process of finding previously unknown and potentially interesting patterns and relations in large databases [5].

KDD is a multidisciplinary field that uses the last achievements from the disciplines of statistics, artificial intelligence, expert systems, pattern recognition, machine learning, databases, high performance computing, and visualization. KDD is not only a theoretical research field. Nowadays KDD has many applications that bring significant gains to organizations, for example through better targeted marketing and enhanced internal performance [2], [3]. KDD is used not only in economics – databases with medical, geological, and astronomical data as data from many other fields are also rich sources of new knowledge.

KDD is usually defined as "the nontrivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data" [6, p.84]. KDD is a process comprising many steps, which involve data selection, data preprocessing, data transformation, data mining (search for patterns), and interpretation and evaluation of patterns. The set of data mining tasks used to extract and verify patterns in data is the core of the process. Data mining (DM) consists of applying data analysis and discovery algorithms for producing a particular enumeration of patterns (or models) over the data [5]. Most current KDD researches are dedicated to the DM step of the whole KDD process. However, this core typically takes only a small part (estimated to be 15%-25%) of the effort of the overall process. The additional steps of the KDD process, such as incorporating appropriate prior knowledge, data selection, data preparation, data cleaning, and proper interpretation of the results of mining, are also essential to derive useful knowledge from data [6].

The current generation of database systems is mainly based on a small number of primitives of Structured Query Language (SQL) which are sufficient to support a vast number of business applications, but not sufficient to capture the emerging family of new applications dealing with knowledge discovery [10]. Most current KDD systems offer only isolated discovery techniques (tree inducers, neural nets, or rule discovery algorithms). Very few systems use a combination of available discovery techniques. Most current KDD systems are based on a loosely coupled architecture, where the database and the data mining subsystems are realized as separate independent parts. This architecture demands continuous context switching between the two subsystems. Such KDD systems are called first-generation database mining systems [10].

Nowadays there is a growing need for second generation database mining systems that manage KDD applications just as SQL-based systems successfully manage database applications. These systems should integrate the data mining and database subsystems and automate all steps of the whole KDD process as much as possible. In this approach, the data mining subsystem is usually combined with the database subsystem, avoiding context switching between the database and data mining parts. These systems should be able to discover knowledge by combining several available KDD techniques.

Recently numerous KDD systems have been developed. There are mainly two basic approaches in modern KDD systems. First includes domain-specific tools that support discovery in a single domain only. Such systems commonly use the language of a user and he needs very little knowledge about the analysis process itself. The second approach includes technique-oriented systems that use one or several knowledge discovery techniques which can be applied in different domains. Most such systems use only single technique (e.g. decision tree, neural network, case based, or rule discovery technique). Very few of these systems propose the use of a combination of the available techniques. Examples of systems which combine several discovery techniques are considered for example in [2], [3].

Our goal is to design an architecture of an integrated knowledge discovery management system (IKDMS), which enables integration of multiple discovery techniques forming a platform upon which different KDD applications can be build. In this paper our focus is on the method evaluation/selection subsystem of an IKDMS. This subsystem is very important in any IKDMS because it helps a user to select an appropriate data mining method among the supported ones. We present and evaluate a technique for advanced dynamic integration of multiple classifiers that is based on the

assumption that each classifier is the best only inside certain sub domains of the whole application domain.

The rest of the paper is structured as follows. In Chapter 2 we present an architecture of an IKDMS. In Chapter 3 we focus the method evaluation/selection subsystem and present and evaluate a technique for advanced dynamic integration of multiple classifiers with some experiments. We conclude with some future research topics in Chapter 4.

2 An architecture of an IKDMS

In this chapter we present an architecture of an IKDMS and discuss the main parts os it. The proposed structure of an IKDMS is presented in Figure 1.

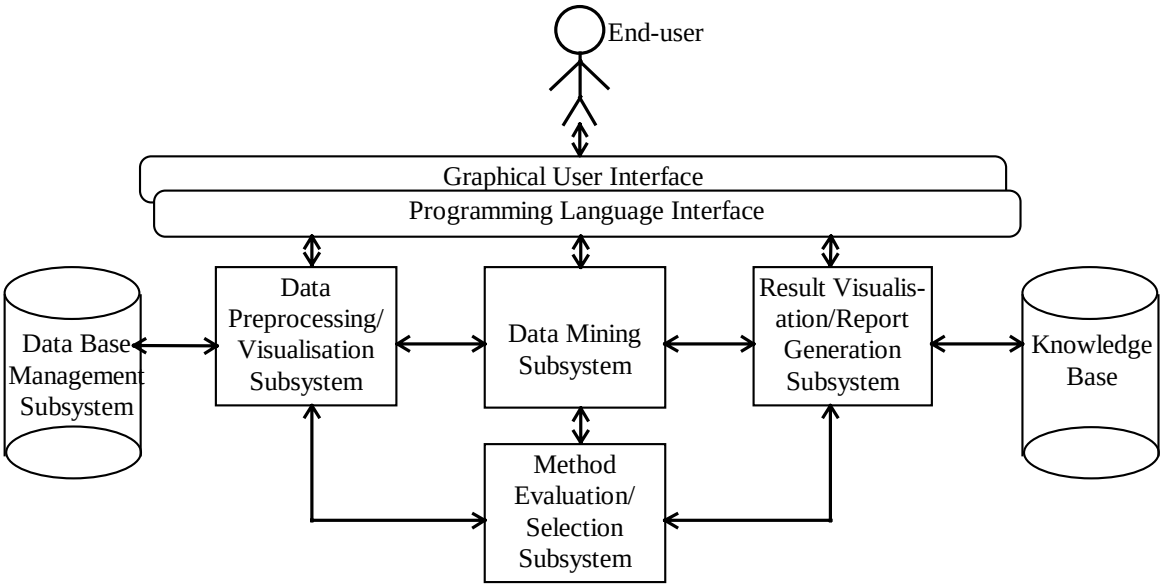

Figure 1: An architecture of an IKDMS

Each subsystem of the IKDMS presented in Figure 1 is intended to automate one basic step of the KDD process and together the whole KDD process as much as possible. The data base management subsystem performs for data storage, retrieval, and manipulation using the standard SQL operations. It is important that this subsystem supports distributed and heterogeneous data and complex data types, adopts a client/server architecture, and can access databases of standard wide-used formats, e.g. ODBC-compatible databases. For example, the ODMG-93 standard can be used as it is proposed in [2].

The data preprocessing/visualization subsystem provides which are needed to prepare data far data mining and visualization facilities to apply exploratory data analysis before data mining techniques are used. The data preprocessing tools are used to remove noise, handle missing data fields, reduce data and make data projections.

The data mining subsystem incorporates different mining techniques for creating models and extracting patterns of data, e.g. classification, clustering, and dependency modeling techniques. It is

important that this subsystem is open allowing easy addition of new techniques as through a "plug-in" interface or implemented as internal procedures using a built-in programming language. This allows rapid extension of the IKDMS without redevelopment the system core.

The method evaluation/selection subsystem is very important, helping a user to select an appropriate data mining method. We have developed already several techniques for the problem of method selection in data mining [14]-[17]. Our results shown that dynamic selection that took into account the expertise areas of each data mining method can lead to better data mining results.

The result visualization/report generation subsystem includes tools for visualization of models being built and patterns discovered and for generation of reports including discovery results. This subsystem helps a user to interpret and evaluate extracted patterns and models which is essential for the process of obtaining new knowledge. This subsystem helps a user to determine which patterns can be considered new knowledge and to make correct conclusions.

The built-in programming language is offered by programming language interface and it is needed to provide extensibility of the IKDMS and to make it configurable to different applications. SQL is extended in [10] to the notion of KDD query language that is proposed to be used in knowledge and data discovery management systems, i.e. in database mining systems of the second generation.

The graphical user interface guides the user through the discovery process helping to reduce the requirements in user's expertise in KDD. This module hides the power and complexity of discovery subsystem deep inside the IKDMS. The graphical user interface should allow an end-user to use the system without any programming abilities.

3 Method evaluation and selection

In this chapter we discuss more deeply about the very important method evaluation/selection subsystem, that helps a user to select an appropriate data mining method among the numerous data mining methods recently suggested. The problem of appropriate method selection has long been solved by static selection approach and only recently effective dynamic selection approaches have been proposed. When the method selection is done dynamically taking into account the characteristics of each instance, then the solution of the selection problem usually gives better results.

In this chapter we present a technique for advanced dynamic integration of multiple classifiers. This technique can be used as a base of the method evaluation/selection subsystem of an IKDMS. The technique is a new variation of the stacked generalization method [18]. We use the basic assumption that each component classifier is the best inside certain sub domains of the whole application domain and try to estimate and present these competence areas in a way that can be used in the dynamic selection of methods.

The problem of integrating multiple classifiers can be defined as follows. Let us suppose that a training set **T** and a set of classifiers **C** are given. The training set **T** is $\{(\mathbf{x}_i, y_i), i = 1,\ldots,n\}$, where n is the number of training instances, $\mathbf{x}_i$ is the i-th training instance presented as a vector of attributes $\{x_j\}, j = 1,\ldots,l$ (values of attributes are numeric, nominal, or symbolic), and $y_i \in \{c_1,\ldots,c_k\}$ is the actual class of the i-th instance, where k is the total number of classes. A set of classifiers **C** is $\{C_1,\ldots,C_m\}$, where m is the number of available classifiers (we shall call them *component classifiers*). Each component classifier is either derived using some learning algorithm or 'hand-crafted' classifier constructed using some heuristic knowledge. A new instance $\mathbf{x}^*$ is an assignment of values

to the vector of attributes $\{x_j\}$. The task is to select from the set **C** one or several classifiers that together make the best classification of the new instance $\mathbf{x}^*$.

In the recent research two basic approaches are used to integrate multiple classifiers. In the first approach the results of component classifiers are combined and in the second approach the best classifier for the new case is selected and applied. We propose a technique that applies the second approach and selects the right classifier dynamically for each new input instance. Our goal is to use each component classifier just in the sub domain where it is the most reliable one, and thus achieve overall results that can be considerably better than those of the best individual classifier alone.

Our technique contains two phases as the stacked generalization. In the learning phase the performance matrix is formed and stacked. It contains information about the performance of each component classifier in each training instance. In the second phase, application phase, the combining classifier (weighted nearest neighbor - WNN [1]) is used to estimate the performance of each component classifier for the new instance based on the performance information collected during the first phase. In this approach just the classifier with the best predicted performance is then used to make the final classification. In the case of ties, the classifier with the smallest ordinal number is used (the component classifiers are ordered in an advance).

Instead of the component classifier predictions which are used in the stacked generalization, we use information about component classifier performance for each training instance. During the first phase we calculate for each training instance the vector of classifier errors that containing *m* values. These values can be binary (i.e. classifier gives correct/incorrect result) or they can represent corresponding misclassification costs. This information about the component classifier performance is then stacked (as in stacked generalization) and it is further used during the second phase together with the initial training set as the meta-level knowledge for estimating errors in the classification of a new instance.

The jackknife method (also called leave-one-out cross-validation or the sliding exam) [1] can be used to calculate the component classifier errors for the training instances of the training set. In the jackknife method, for each instance in the training set each of the *m* component classifiers are trained using all the other instances of the training set. After training the held-out instance is classified by each of the trained component classifiers. Then, comparing the prediction of each component classifier with the actual class of the held-out instance, a vector of method errors (or misclassification costs) is formed. When this is applied for every instance we receive *n* vectors collected into the matrix of classifier performance $\mathbf{P}_{n\times m}$. This matrix is used with the set of attributes of training instances $\mathbf{T}$ as the training set when the meta-level classifier $\mathbf{T}^*_{n\times(l+m)}$ is formed (*l* is the number of attributes).

The jackknife method is very labor-consuming. In the cases when the training set has many training instances with many attributes it is more reasonable to use the cross-validation technique [1]. In the cross-validation technique all the instances of the training set are divided randomly into *v* approximately equal sized partitions that are usually called *folds*. Each classifier is then trained on *v-1* folds *v* times and tested *v* times on the fold being held-out. Thus the above jackknife method is a particular case of the cross-validation, where the size of each fold is exactly one. In the case we have pre-existent (non-learning) heuristic classifiers we do not need the jackknife or cross-validation; the accuracy of such classifiers is directly evaluated using the initial training instances.

To predict the errors of component classifiers, we propose the use of the weighted nearest neighbor classification (WNN). WNN simplifies the learning phase of the composite classifier. The composite classifier that uses WNN does not need learning *m* referees (meta-level classifiers) as it was proposed in [13]. In the learning phase one needs only to calculate the component classifier performance matrix. In the application phase the nearest neighbors of the new instance are found out among the training instances and the corresponding component classifier performances are used to

predict the performance of each component classifier. In the calculation of performance predictions we sum up the corresponding performance values of a classifier using weights that depend on the distances between the new instance and its nearest neighbors.

The use of WNN as the meta-level classifier is based on the assumption that each classifier has certain sub domains of the space of instance attributes, where it is more reliable than the other ones. This assumption is supported by experiences, that component classifiers usually work well not only in certain points of the domain space, but in certain sub areas of the domain space. The performance of a classifier usually changes gradually from one instance to another near-by instance. Thus if a classifier does not work well with the instances near the new instance, then it is quite probable that it will not work well with the new instance. Of course, the training set should provide a good representation of the whole domain space in order to make such statement highly confident.

This approach was evaluated on several benchmark data sets taken from the UCI Machine Learning Repository. It was also compared with other classifier selection methods. The results of these comparisons were considered also in [16,17]. They show that this dynamic classifier selection very often produces more accurate results than other methods and shortens the time demands of the application phase.

We made experiments with our dynamic integration technique of multiple classifiers using the following three data sets from the University of California Machine Learning Repository: Iris Plants Database, BUPA Liver Disorders database, and Heart Disease Database [12]. The three classification methods that we used are [1,9,11].: DANN (Discriminant Adaptive Nearest Neighbour Classification), K-NN (K-nearest neighbour classification), and LDA (Linear Discriminant Analysis). We first describe shortly the databases and then the results of experiments.

The Iris Plants Database created by R.A. Fisher [7] is perhaps the best known database in the pattern recognition literature. Fisher's paper is considered a classic in the field and it is referenced frequently. The data base contains 150 instances. Each instance has 4 predictive numeric attributes: sepal length in cm, sepal width in cm, petal length in cm, and petal width in cm and one classification attribute with three possible values: Iris Setosa, Iris Versicolour, and Iris Virginica. There are no missing values included. There are equal number of instances in each three class (i.e. type of iris plant) and one of the classes is linearly separable from the other 2 but those two are not linearly separable from each other. This is an exceedingly simple domain and very low misclassification rates have been reached already long ago [4,8] .

The average cross-validation errors for the three classification methods were the following: DANN 0.053, K-NN 0.040, and LDA 0.053. Thus the misclassification rate is very low in this simple domain. Our method did not give any improvement in the accuracy, but the time required was considerably shorter because our dynamic technique applied simpler and thus quicker diagnostic method where it was possible. Our technique required only half of the time taken by the single best method.

The BUPA Liver Disorders database created by BUPA Medical Research Ltd contains 345 instances. Each instance has 7 numeric attributes, the first five ones are results of blood tests of male patients: mean corpuscular volume, alkaline phosphotase, alamine aminotransferase, aspartate aminotransferase, and gamma-glutamyl transpeptidase. These are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. The sixth atribute is the number of half-pint equivalents of alcoholic beverages drunk per day, and the last one is selector field that is used to split the data into two sets. It has previously been discovered that "drinks# > 5" is some sort of selector on this database. There are no missing values in the data set.

The average cross-validation errors for the three classification methods were the following: DANN 0.333, K-NN 0.365, and LDA 0.351. Thus the misclassification rate is clearly higher than with Iris classification domain. Our technique gave average error rate 0.134 that is essentially lower than 0.333 of the best single classification method.

The Heart Disease Database created by European Statlog project contains 270 instances. Each instance has 13 attributes which have been extracted from a larger set of 75 attributes. There are six real number attributes: age, resting blood pressure, serum cholestoral in mg/dl, maximum heart rate achieved, oldpeak = ST depression induced by exercise relative to rest, number of major vessels (0-3) colored by flourosopy, one order type attribute: the slope of the peak exercise ST segment, three binary attribute: sex, fasting blood sugar > 120 mg/dl, exercise induced angina, and three nominal attributes: chest pain type (4 values), resting electrocardiographic results (values 0,1,2), and thal: 3 = normal; 6 = fixed defect; 7 = reversable defect. The variable to be predicted is the absence or presence of heart disease. There are no missing values in the data set.

The average cross-validation errors for the three classification methods were the following: DANN 0.196, K-NN 0.352, and LDA 0.156. Our technique gave average error rate 0.08 that is essentially lower than 0.156 of the best single classification method. The Heart database includes also the cost matrix where the cost of misclassification to the class absence of heart disease in the case heart disease is five times as high as the other possible misclassification. Using the cost matrix frequently changed the decision of our dynamic technique.

4 Conclusions

In this paper we presented an architecture of an IKDMS. The structure of the system was discussed with short discussion of the main parts of it. The system consists of several basic subsystems, helping to automate different KDD steps: the database management subsystem, the data preprocessing/visualization subsystem, the data mining subsystem, the result visualization/report generation subsystem, and the method evaluation/selection subsystem. In data mining one of the key problems is the selection of the most appropriate data mining method. Thus the method evaluation/selection subsystem is very important part of the whole system helping a user to select an appropriate method.

We presented and evaluated a technique for advanced dynamic integration of multiple classifiers that can be used as the base of the method selection subsystem. The technique is based on the assumption that each component classifier gives the best results inside certain sub domains of the whole application domain. It was shown in experiments that this technique performs better than analogous ones. However, this technique considers integration of classifiers only, not taking into account selection of other important methods needed in the whole KDD process (e.g., feature selection methods). Further research is needed to develop the proposed IKDM architecture so that it can be implemented and to extend the features of the method evaluation/ selection subsystem in order to provide a user with a guidance through the whole KDD process.

Acknowledgments

We thank the UCI Machine Learning Repository of databases, domain theories and data generators for the data sets used in this work. We are grateful to the Centre for International Mobility (CIMO) about the support that made it possible for Alexey Tsymbal to work in Jyväskylä during the preparation of this article and research behind it.

References

[1] Aivazyan, S.A., *Applied Statistics: Classification and Dimension Reduction*, Finance and Statistics, Moscow, 1989 (in Russian).

[2] Anand, S.S., Scotney, B.W., Tan, M. G., McClean, S.I., Bell, D.A., Hughes, J.G., Magill, I.C., Designing a Kernel for Data Mining, In: *IEEE Expert/Intelligent Systems & Their Applications*, Vol.12, No.2, March-April 1997.

[3] Brunk, C. , Kelly, J. , Kohavi, R., MineSet: An Integrated System for Data Mining, In: *Proceedings of Third International Conference on Knowledge Discovery and Data Mining (KDD-97)*, AAAI Press, 1997.

[4] Dasarathy, B.V., Nosing Around the Neighborhood: A New System Structure and Classification Rule for Recognition in Partially Exposed Environments, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-2, No. 1, 1980, pp.67-71.

[5] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R., *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, 1996.

[6] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Knowledge Discovery and Data Mining: Towards a Unifying Framework, In: *Proceedings of Second International Conference on Knowledge Discovery and Data Mining (KDD-96)*, AAAI Press, 1996.

[7] Fisher,R.A. The use of multiple measurements in taxonomic problems, *Annual Eugenics*, 7, Part II, pp. 179-188, 1936. Also in *Contributions to Mathematical Statistics*, John Wiley, NY, 1950.

[8] Gates, G.W., The Reduced Nearest Neighbor Rule. *IEEE Transactions on Information Theory*, May 1972, pp. 431-433.

[9] Hastie, T., Tibshirani, R., *Discriminant Adaptive Nearest Neighbour Classification*, Technical Report, Stanford University, 1995.

[10] Imielinski, T., Mannila, H., A Database Perspective on Knowledge Discovery, *Communications of the ACM*, Vol.39, No.11, Nov. 1996, pp. 58-64.

[11] McLachlan, G.J., *Discriminant Analysis and Statistical Pattern Recognition*, Wiley, New York, 1992.

[12] Merz, C.J., & Murphy, P.M. *UCI Repository of machine learning databases* [http://www.ics.uci.edu/~mlearn/MLRepository.html]. Irvine, CA: University of Califonia, Department of Information and Computer Science. (Read spring 1998).

[13] Ortega, J., Koppel, M., Argamon-Engelson, S., Arbitrating Among Competing Classifiers Using Learned Referees, In: *Machine Learning*, 1998 (to appear).

[14] Terziyan, V., Tsymbal, A., Puuronen, S., The Decision Support System for Telemedicine Based on Multiple Expertise, In: *International Journal of Medical Informatics*, Elsevier, 1998 (to appear).

[15] Terziyan, V., Tsymbal, A., Tkachuk, A., Puuronen, S., Intelligent Medical Diagnostics System Based on Integration of Statistical Methods, In: *Informatica Medica Slovenica, Journal of Slovenian Society of Medical Informatics, Special Issue: Proc. of the 10th IEEE Symp. on Computer-Based Medical Systems*, Vol.3, Ns.1,2,3, 1996, pp.109-114.

[16] Tsymbal, A., Puuronen, S. Arbiter Meta-Learning with Dynamic Selection of Multiple Classifiers, In: *Proc. Workshop On High Performance Data Mining HPDM'98 (Held in conjunction with the IPPS/SPDP'98 conference)*, Orlando, Florida, USA, March 1998, pp. 7-11.

[17] Tsymbal, A., Puuronen, S., Terziyan, V., Advanced Dynamic Selection of Diagnostic Methods, In: *Proc. 11th IEEE Symp. on Computer-Based Medical Systems CBMS'98*, IEEE CS Press, 1998, pp. 50-54.

[18] Wolpert, D., Stacked Generalization, *Neural Networks*, 5, 1992, pp. 241-259.