데이터베이스로부터의 선형계획모형 추출방법에 대한 연구

Linear Programming Model Discovery from Databases

권오병*, 김윤호
한동대학교 경영경제학부
kob@handong.edu

Abstract

Knowledge discovery refers to the overall process of discovering useful knowledge from data. The linear programming model is a special form of useful knowledge that is embedded in a database. Since formulating models from scratch requires knowledge-intensive efforts, knowledge-based formulation support systems have been proposed in the DSS area. However, they rely on the strict assumption that sufficient domain knowledge should already be captured as a specific knowledge representation form. Hence, the purpose of this paper is to propose a methodology that finds useful knowledge on building linear programming models from a database. The methodology consists of two parts. The first part is to find s first-cut model based on a data dictionary. To do so, we applied the GPS algorithm. The second part is to discover a second-cut model by applying neural network technique. An illustrative example is described to show the feasibility of the proposed methodology.

1. Introduction

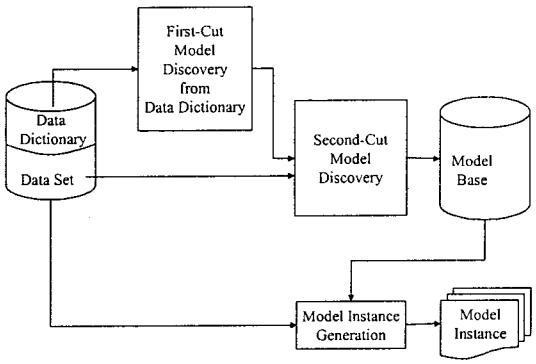
Linear programming model formulation from scratch requires knowledge-intensive efforts and is therefore very costly. The formulation efforts include selecting decision variables, and finding functional and linear dependencies between decision variables and other coefficients. In this paper we selected the Structured Modelling formatted model instance as a final output since the format is very rigorous and well known in DSS research area. Moreover, the SML (Structured Modelling Language) compiler and executor are already prepared [2]. The Structured Modelling schema is composed of elements and their calling sequences. Hence, formulating SML schema formatted models from database is equivalent to finding elements from database and then enumerating them according to the SML syntax. Among them, finding functions and tests are more difficult because they can be defined after discovering numerical and functional dependencies among other attributes. This kind of work can be performed by data mining or knowledge discovery from database (KDD).

Data mining is the exploration and analysis, by automatic or semiautomatic means, of large quantities of data in order to discover meaningful patterns and rules [3]. Knowledge discovery, on the other hand, refers to the overall process of discovering useful knowledge from data. Frawley *et. al.* defined knowledge discovery as the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data. Standard statistical method, Market Basket Analysis, Memory-Based Reasoning, Genetic Algorithms, Cluster Detection, Link Analysis, Decision Trees, and Artificial Neural Networks are popularly used techniques for data programming model discovery method. The first phase is to exhaustively find well-formed model structures from a data dictionary. To do so, we apply the GPS algorithm. The second phase is to choose semantically meaningful models among well-formed model structures. Back propagation method, one of the Artificial Neural Network methods, is adopted as a selection tool.

2. Overall Framework

An overall structure of the two-phase model formulation from data model is shown in Figure 1. The framework is divided into the three components: (1) first-cut model discovery from data dictionary, (2) second-cut model discovery, and (3) model instance generation. Data dictionary embeds information of modelling constructs. Most of the domain knowledge for model formulation is directly imported from database; sets, parameters, and decision variables are imported from relations, indices are deduced from keys of relations in data model [1]. In addition, model constructs that are compound of relations, key of relations, and operators (arithmetic, sum, product, and comparison) can be incorporated by data manipulations. However, the database does not contain knowledge on constraints and objective function. To achieve them from another model constructs, additional efforts must be committed. To increase the speed of finding models from database, rather than investigating all possible linear combination of elements, the first-cut model discovery involves some useful knowledge about mining a "syntactically verified" set of Structured Modeling elements from original data dictionary. We will call the knowledge metaknowledge in this paper. The metaknowledge is based on the means-ends analysis. Means-ends analysis, which is applied to General Problem

Solver (GPS), is one of the problem solving methodologies in cognitive science area [4]. The key activities of the analysis are (1) to analyse current state to a series of differences, (2) to decompose them into a set of sub-problems, and (3) to select relevant operators to remove the differences. Thus, the metaknowledge consists of identifying the gaps between the current state and goal state, decomposing problems based on 'divide-and-conquer' strategy, and performing problem solving until no gaps are left. As a result, we can acquire a first-cut model by collecting candidate SM elements. The fist-cut model, as structural domain knowledge, may contribute to the problem identification and second-cut model formulation.



< Figure 1 > Overall Framework

The second-cut model discovery is to pick out a "semantically verified" set of Structured Modelling elements from the first-cut model. To do so, an Artificial Neural Network model is designed to evaluate how much the elements in the first-cut model are well fitted to the information in the database. In case of test element, each model elements in a test element are registered as input nodes and the feasibility is registered as output node. The neural network performs learning by the training and testing data that are generated randomly. Then the system acquires interrogating data from real data and analyses error rates. If the error rates from training data and interrogating data are statistically indifferent and also if the average error rates are scored under a threshold level, then the model element will be regarded as semantically meaningful. Thus, the "semantically" verified set of SM elements is stored as a model base.

Model instance generation component firstly transforms the SM elements into an SML compliant programme. Secondly, the component generates SQL statement to substantiate the SML compliant programme.

3. First-Cut Model Discovery from Data Dictionary

Semantics on database is embedded in a data dictionary as metadata. In general, metadata contains name, aliases, classification, description, attributes with data types and units, standard representation, and entity class membership. An attribute has its own data type, index set, and units. The attribute is classified with *primitive attribute* and *derived attribute*. Primitive attribute is a data that has no functional dependencies with any other attributes in a domain. Derived attribute is an attribute that has functional dependencies with another attributes. It is analogous to numeric derivation rather than symbolic derivation in deductive database. For example, if X_{ij} and A_{ij} are attributes, then $D_j = \sum_i A_{ij} * X_{ij}$ is a derived attribute on X_{ij} and A_{ij} since D_j is dependent of A_{ij} and X_{ij}. I will define the specification of the data type, index, and unit of an attribute as a *state S* in a set of all possible *state S*. A *state S* is represented as the following 3-tuble:

$$S =< DT_S I_S U_S >$$

where DT_S : data type of $S, DT_S \in DT := \{DT = \{\forall S \in S, DT_S\}\}$, I_S : index set of $S, I_S \in I := \{I = \{\forall S \in S, I_S\}\}$, U_S : data unit of $S, U_S \in U := \{U = \{\forall S \in S, U_S\}\}$ (1)

Besides, let an attribute A(A∈A, A is a set of attributes of selected problem domain) exist, then a meny-to-one mapping function from an attribute to its state as ρ which satisfies:

$$\rho(A) = S \quad (2)$$

can be devined. From (1), (2):

$$\rho(A) = S =< DT_{\rho(A)}, I_{\rho(A)}, U_{\rho(A)} > \quad (3)$$

is satisfied.

In LP model formulation it is observed that each left hand side (LHS) and right hand side (RHS) in every valid objective function and constraint can be identified as a state and it satisfies: (1) the data type; (2) the index set; and (3) the unit of LHS and RHS are identical. For example, suppose a constraint, $\sum_i A_{ij} * X_{ij} \le B_j$, is valid. Then the data type, index set, and unit of $\sum_i A_{ij} * X_{ij}$ and B_j are the same. On the other hand, $\sum_i A_{ij} * X_{ij} \le D_i$ is not valid since the index set of LHS ({j}) is different from that of RHS({i}). Murphy addressed that the formulation of the model is valid if the index set of each LHS matches that of RHS [4]. In general, if at least one of them is not equivalent, the constraint is no more valid and additional operations to perish the differences should be required. Thus, a rule for model validation is defined as:

[Definition] *Model Vaildation Rule*: If an LP model is valid, then LHS and RHS of the model have the same data type, index set, and unit.

The model validation rule implies that if a modeller elicits a set of model constructs to formulate an objective function or a constraint, then at least the data type, index set, and unit of the LHS and RHs should be identical. Meanwhile, the activities of model formulation usually begin with the decision variable indentification. If RHS is given, then the model formulation work is an activity to determine the right relationship between decision variable and RHS, which includes identifying the differences of data type, index set, and unit of selected decision variable and RHS constants. Suppose a problem P is defined as follows:

$$P =< I(s), G(s), M, S >$$

where

$I(s)$: initial state,
$G(s)$: goal state,
M : operator,
S : a set of state. (4)

Then a problem solving is interpreted as expiring the gaps between $I(s)$ and $G(s)$ by applying appropriate M. In each state of applying operators, the changed states are added to S. In LP formulation problem, the initial state and goal state will correspond to decision variable and RHS constants, respectively.

The first-cut model discovery process involves problem definition, sub-problem generation, and difference resolution shown as follows:

[Step 1] Problem definition
[Step 1.1] Determine a problem domain and identify it as A (a set of attributes in the problem domain) and corresponding state $\rho(A) = S$ (original problem; S = <*DT, I, U*>)
[Step 2] Sub-problem generation
[Step 2.1] Elicit decision variable A in A and identify it as an initial state
($\rho(A) = I(S)$):
$I(S) =< DT_{I(S)}, I_{I(S)}, U_{I(S)} >$.
[Step 2.2] Elicit candidates for RHS in A.
[Step 2.3] For each candidate, decompose the original problem into sub-problems. Each of the sub-problems are represented as goal state (G(S)):
$G(S) =< DT_{G(S)}, I_{G(S)}, U_{G(S)} >$.
[Step 3] Difference resolution. For all sub-problems,
[Step 3.1] Analyse differences: characteristic difference, structural difference, and unit difference.
[Step 3.2] Apply relevent modelling engine.
[Step 3.3] Repeat step 3.1 and step 3.2 until either of the following conditions is satisfied. If no differences are found, then stop. The sub-problem is solved. If any differences are found and no relevant modelling engines can be applied, then stop. The sub-problem is insoluble.
[Step 4] Determine a model that is made up of solved suproblems.

4. Second-Cut Model Discovery

Actually, it is not easy for generic modelling systems, as proposed in this paper, to warrant model validity. One of the typical methods for model validity is to test the feasibility by using past data, and specially test by causal modelling or path analysis. We combine the two methods by applying back propagation method, which is one of the Artificial Neural Network techniques for knowledge discovery. In designing neural network model, input nodes represent model elements in a function or test, and output node indicates the feasibility.

At this time, it is our assumption that it is a valid module if the network model learns with training data and testing data that are artificially prepared, then the correctness level of the network model from the artificially prepared data set and that from real data would be most similar when compared with other modules in the same variation group. As a matter of course, if correctness levels from real data of all modules in a same variation group do not come up to the correctness level from artificially prepared data set, then the modules are judged as invalid.

There are two useful measures: similarity level and usefulness level. Similarity level is a measure how the correctness of learning rule interrogated by real data is similar to that of learning rule interrogated by training data and testing data that are artificially and randomly prepared

Similarity level:

$$S(i,\alpha_1,\alpha_2,...,\alpha_n) = 100 - \frac{}{|2C(i,\alpha_1,\alpha_2,...,\alpha_n,3) - (C(i,\alpha_1,\alpha_2,...,\alpha_n,1) + C(i,\alpha_1,\alpha_2,...,\alpha_n,2))|},$$

where $S(i,\alpha_1,\alpha_2,...,\alpha_n)$ designates the similarity level of module i, and $C(i,\alpha_1,\alpha_2,...,\alpha_n,1)$, $C(i,\alpha_1,\alpha_2,...,\alpha_n,2)$, $C(i,\alpha_1,\alpha_2,...,\alpha_n,3)$ designate correctness level for module i by training data, testing data and real data, respectively.

In the meantime, usefulness level shows how well the learning rule from a neural network model explains the real situation.

Usefulness level:
$U(i) = (C(i,1) + C(i,2) + C(i,3))/3$,
where the U(i) indicates usefuleness level.

Therefore, if we assume that λ is a minimum usefulness level and also satisfies
$\lambda \geq \max(\forall i,\ U(i))$, then the model variation group is decided as invalid.

Guideline for finding valid module (e.g.: $\alpha_1 b_1 + \alpha_2 b_2$, if there are two RHS constants) is as follows:

Step 0: Initialize $\alpha_1 = 0, \alpha_2 = 0$.
Set
$\xi_{\alpha_1+\delta_1,\alpha_2+\delta_2} = S(i,\alpha_1+\delta_1,\alpha_2+\delta_2) - S(i,\alpha_1,\alpha_2)$
where $\delta_1 = \delta_2 = \begin{cases} -\delta \\ 0 \\ \delta \end{cases}$.

Step 1: If $C(i,\alpha_1,\alpha_2,...,\alpha_n,3) < \theta$, where θ is the randomness level, then prune it.
Step 2: If $\xi_{\alpha_1+\delta_1,\alpha_2+\delta_2} > 0$, insert the cell $(\alpha_1+\delta_1,\alpha_2+\delta_2)$ into set Ψ. Then $\alpha_1 = \alpha_1 + \delta_1, \alpha_2 = \alpha_2 + \delta_2$.
Step 3: Until there is no cell that satisfies at least one of the following pruning rules, then go to step 4. Otherwise, go to Step 1.
Rule1: If $U(i) = (C(i,1) + C(i,2) + C(i,3))/3 < \lambda$, then prune it.
Rule2: If $C(i,\alpha_1,\alpha_2,...,\alpha_n,3) \leq \eta$, then prune it, where η designates minimum allowable correctness level.
Rule3: If $\xi_{\alpha_1+\delta_1,\alpha_2+\delta_2} \leq \mu$, then prune it, where μ designates maximum allowable declining amount of similarity level.
Step4: Select optimal solution that has maximum similarity level in set Ψ.

5. An Illustrative Example

To show the feasibility of the idea described in this paper, let us show an illustrative example. The Table 1, Table 2, and Table 3 show data dictionaries and some records of the example. In each table, first row designates data field name. Underlines are candidate key. The second row shows the data type of the fields. The third row means unit of the corresponding fields, if any. Finally, the fourth row indicates example data.

<Table 1> Raw_Material(i)

RAW_ID	MIN_INV_REQ	UNIT_COST	Maximum_Daily_Capacity
String(4)	Numeric	$/Numeric	Numeric
-	Kg	$1000	Tons
M1	5,000	1.2	24
M2	1,500	0.9	6

<Table 2> Product(j)

PROD_ID	PROD_NAME	PROD_QUANTITY	Unit_Profit
String(3)	String(20)	Numeric	$/ Numeric
-	-	1000	1000
X1	Exterior Paint	47.6	5
....	.....	.....	.....
X2	Interior Paint	21.7	4
....	.....	.....	.....

<Table 3> Involves(i,j)

RAW_ID	PROD_ID	Equipments	Usage_Rate
String(2)	String(2)	String(10)	Numeric / Numeric
-	-	-	Tons
M1	X1	Mixer1	6
M2	X2	Mixer2	1

After applying guidelines to acquire first-cut models and model variation rules, the following two modules are deduced.

$$1000 * \sum_i USAGE\_RATE * PROD\_QUANTITY \le \alpha_1 MAXIMUM\_DAILY\_CAPACITY - 0.001 * \alpha_2 MIN\_INV\_REQ$$

where α_1, α_2 *is real.*

Then to make the most adequate model, a neural network model is designed. For back propagation, Neural Planner Ver.4.52 is adopted. Neural Planner is a neural network system for Microsoft Windows. Neural Planner can learn from training files, self-test using testing files and are interrogated by interrogating files. It can produce spreadsheet or hierarchical output files interactively.

The model instance generated from the second-cut model is shown as Figure 2.

&PROD PRODUCTION SECTOR

RAW_MATERIALi /pe/ There is a list of RAW MATERIAL.

MAXIMUM_DAILY_CAPACITY (RAW_MATERIALi) /a/ RAW_MATERIAL : Real+ Every RAW_MATERIAL has a MAXIMUM_DAILY_CAPACITY measured in tons.

MIN_INV_REQ (RAW_MATERIALi) /a/ RAW_MATERIAL: Real+ Every RAW_MATERIAL has a MIN_INV_REQ measured in kilograms.

PROCUTj /pe/ There is a list of PRODUCT.

PROD_QUANTITY (PRODUCTj) /va/ PRODUCT : Real+ Every PRODUCT has a nonnegative PRODUC_QUANTITY measured in thousands.

UNIT_PROFIT (PRODUCTj) /a/ PRODUCT : Real+ Every PRODUCT has a nonnegative UNIT_PROFIT measured in thousands.

INVOLVES (RAW_MATERIALi, PRODUCTj) /ce/ Select RAW_MATERIAL x PRODUCT where i covers RAW_MATERIAL, j covers PRODUCT A PRODUCT INVOLVES RAW_MATERIAL. There must be at least one INVOLVES incident to each RAW_MATERIAL, and at least one INVOLVES incident to each PRODUCT.

USAGE_RATE (INVOLVESij) /a/ INVOLVES : Real+ There can be a nonnegative USAGE_RATE over each INVOLVES.

$ (PRODUCT) /f/ 1 ; @ SUMj (PROD_QUANTITYj * UNIT_PROFITj) There is a TOTAL PROFIT associated with all PRODUCT.

T:SUP (RAW_MATERIALi., RAW_MATERIALi) /t/ RAW_MATERIAL ; @SUMj (USAGE_RATEij * PROD_QUANTITYj) <= MAXIMUM_DAILY_CAPACITYi – MIN_INV_REQi Is the total RAW_MATERIAL used to product PRODUCT to PRO_QUANTITY level less than or equal to MAXIMUMDAILY_CAPACITY minus MIN_INV_REQ? There is called the SUPPLY TEST.

< Figure 2> The example output SML program

6. Conclusion

Finding useful knowledge from database in an efficient way has become critical in the recent competitive and enterprising environment. We drew attention to the fact that the linear programmeming model would be one of the most useful forms of knowledge buried in the corporate database. As a result, model discovery from database is proposed in this paper. We show how a neural network technique can be applied to discover models from database.

References

[1] Choobineh, J., "SQLMP: A Data Sublanguage for Representation and Formulation of Linear Mathematical Models," *ORSA Journal on Computing*, Vol.3, No.4, 1991, pp.358-375.

[2] Geoffrion, A.M., "Reusing Structured Models via Model Integration," *Proceedings of the Twenty Second Hawaii International Conference on System Sciences*, 1989, pp.601-611.

[3] Murphy, F.H., Stohr, E.A., and P.C. Ma, "Composition Rules for Building Linear Programming Models from Component Models," *Management Sciences*, Vol.38, No.7 1992, pp.948-963.

[4] Newell, A., and H. Simon, *Human Problem Solving*, Englewood Cliffs, NJ: Prentice-Hall Inc., 1982.