Propositionalization

Nicolas Lachiche

Propositionalisation

Nicolas Lachiche, University of Strasbourg

October 8, 2014

Definition

Propositionalisation is the process of explicitly transforming a relational dataset into a propositional dataset.

The input data consists of examples represented by structured terms (cf. Learning from Structured D several predicates in first-order logic, or several tables in a relational database. We will jointly refer to these as *relational representations*. The output is an attribute-value representation in a single table, where each example corresponds to one row and is described by its values for a fixed set of attributes. New attributes are often called features to emphasize that they are built from the original attributes. The aim of propositionalisation is to pre-process relational data for subsequent analysis by attribute-value learners. There are several reasons for doing this, the most important of which are: to reduce the complexity and speed up the learning; to separate modeling the data from hypothesis construction; or to use familiar attribute-value (or propositional) learners.

Motivation and Background

Most domains are naturally modeled by several tables in a relational database or several classes in an object-oriented language, for example: customers and their transactions; molecules, their atoms and bonds; or patients and their examinations. A proper relational dataset involves at least two tables linked together. Typically, one table of the relational representation corresponds to the individuals of interest for the machine learning task, and the other tables contain related information that could be useful. The first table is the individual, or the primary table, the other tables are complementary tables.

Example 1 *Let us consider a simplified medical domain as an example. This is inspired by a real medical dataset [10]. It consists of four tables.*

The patient table is the primary table. It contains data on each patient such as the patient identifier (pid), name, date of birth, height, job, the identifier of the company where the patient works, etc.:

Patient

pid	*name*	*birth*	*height*	*job*	*company*	...
I	*Smith*	*15/06/1956*	*1.67*	*manager*	*a*	...
II	*Blake*	*13/02/1968*	*1.82*	*salesman*	*a*	...
⋮	⋮	⋮	⋮	⋮	⋮	...

The company table contains its name, its location, and so on. There is a many-to-one relationship from the patient table to the company table: A patient works for a single company, but a company may have several employees.

Company

cid	*name*	*location*	...
a	*Eiffel*	*Paris*	...
⋮	⋮	⋮	...

The examination table contains the information on all examinations of all patients. For each examination, its identifier (eid), the patient identifier (pid), the date, the patient's weight, whether the patient smokes, his or her blood pressure, etc. are recorded. Of course, each examination corresponds to a single patient, and a given patient can have several examinations, i.e., there is a one-to-many relationship from the patient table to the examination table.

Examination

eid	*pid*	*date*	*weight*	*smokes*	*BP*	...
1	*I*	*10/10/1991*	*60*	*yes*	*10*	...
2	*I*	*04/06/1992*	*64*	*yes*	*12*	...
⋮	⋮	⋮	⋮	⋮	⋮	...
23	*II*	*20/12/1992*	*80*	*yes*	*10*	...
24	*II*	*15/11/1993*	*78*	*no*	*11*	...
⋮	⋮	⋮	⋮	⋮	⋮	...

Additional tests can be prescribed at each examination. Their identifiers (tid), *corresponding examinations (eid), names, values and interpretations are recorded in the additional_test table:*

Additional_test

tid	*eid*	*date*	*name*	*value*	*interpretation*
t237	*1*	*19/10/1991*	*red blood cells*	*35*	*bad*
t238	*1*	*23/10/1991*	*radiography*	*nothing*	*good*
⋮	⋮	⋮	⋮	⋮	⋮
t574	*2*	*07/06/1992*	*red blood cells*	*43*	*good*
⋮	⋮	⋮	⋮	⋮	⋮

Several approaches exist to deal directly with relational data, e.g., inductive logic programming, relational data mining [2], or statistical relational learning. However relational hypotheses can be transformed into propositional expressions.

Generally, a richer representation language permits the description of more complex concepts, however, the cost of this representational power is that the search space for learning greatly increases. Therefore, mapping a relational representation into a propositional one generally reduces search complexity.

A second motivation of propositionalisation is to focus on the construction of features before combining them into an hypothesis [9]. This is related to feature construction, and to the use of background knowledge. One could say that propositionalisation aims at building an intermediate representation of the data in order to simplify the hypothesis subsequently found by a propositional learner.

A third motivation is pragmatic. Most available machine learning systems deal with propositional data only, but tend to include a range of algorithms in a single environment, whereas relational learning systems tend to concentrate on a single algorithm. Propositional systems are therefore often more versatile and give users the possibility to work with the algorithms they are used to.

Solutions

There are various ways to propositionalise relational data consisting of at least two tables linked together through a relationship. We will first focus on a single relationship between two tables. Most approaches can then iteratively deal with several relationships as explained below.

Propositionalisation mechanisms depend on whether that relationship is functional or non-determinate. This distinction explains most common mistakes made by newcomers.

Functional relationship (many-to-one, one-to-one)

When the primary table has a many-to-one or one-to-one relationship to the complementary table, each row of the primary table links to one row of the complementary table. A simple join of the two tables results in a single table where each row of the primary table is completed with the information derived from the complementary table.

Example 2 *In our simplified medical domain, there is a many-to-one relationship from each patient to his or her company. Let us focus on those two tables only. A join of the two tables results in a single table where each row describes a single patient and the company he or she works for:*

Patient and his/her company

pid	*name*	*birth*	*height*	*job*	*cid*	*company*	*location*	…
I	*Smith*	*15/06/1956*	*1.67*	*manager*	*a*	*Eiffel*	*Paris*	…
II	*Blake*	*13/02/1968*	*1.82*	*salesman*	*a*	*Eiffel*	*Paris*	…
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	…

The resulting table is suitable for any attribute-value learner.

Non-determinate relationship (one-to-many, many-to-many)

Propositionalisation is less trivial in a non-determinate context, when there is a one-to-many or many-to-many relationship from the primary table to the complementary table, i.e., when one individual of the primary table is associated to a set of rows of the complementary table.

A propositional attribute is built by applying an aggregation function to a column of the complementary table over a selection of rows. Of course a lot of conditions can be used to select the rows. Those conditions can involve other columns than the agregated column. Any aggregation function can be used, e.g., to check whether the set is not empty, to count how many elements there are, to find the mean (for numerical) or the mode (for categorical) values, etc.

Example 3 *In our simplified medical domain, there is a one-to-many relationship from the patient to his or her examinations. Let us focus on those two tables only. Many features can be constructed. Simple features are aggregation functions applied to a scalar (numerical or categorical) column. The number of occurrences of the different values of every categorical attributes can be counted. For instance, the f60 feature in the table below counts in how many examinations the patient stated he or she smoked. The maximum, minimum, average and standard deviation of every numerical columns can be estimated, e.g., the f84 and f85 features in* *the table below respectively estimates the average and the maximum blood pressure of the patient over his or her examinations. The aggregation functions can be applied to any selection of rows, e.g., the f135 feature in the table below estimates the average blood pressure over the examinations when the patient smoked.*

Patient and his/her examinations

pid	*name*	...	*f60*	...	*f84*	*f85*	...	*f135*	...
I	*Smith*	...	*2*	...	*11*	*12*	...	*11*	...
II	*Blake*	...	*1*	...	*10.5*	*11*	...	*10*	...
⋮	⋮	...	⋮	...	⋮	⋮	...	⋮	...

From this example it is clear that non-determinate relationships can easily lead to a combinatorial explosion of the number of features.

Common mistakes and key rules to avoid them

Two mistakes are frequent when machine learning practitioners face a propositionalisation problem, i.e., when they want to want to apply a propositional learner to an existing relational dataset [7].

The first mistake is to misuse the (universal) join. Join is valid in a functional context, as explained earlier. When applied to a non-determinate relationship, it produces a table where several rows correspond to a single individual, leading to a multiple instance problem [1] (cf. Multi-instance Learning).

Example 4 *In our simplified medical domain, there is a one-to-many relationship from the patient table to the examination table. If a join is performed, each row of the examination table is completed with the information on the examined patient, i.e., there are as many rows as examinations.*

Examination and its patient

eid	*date*	*weight*	*smokes*	*BP*	...	*pid*	*name*	...
1	*10/10/1991*	*60*	*yes*	*10*	...	*I*	*Smith*	...
2	*04/06/1992*	*64*	*yes*	*12*	...	*I*	*Smith*	...
⋮	⋮	⋮	⋮	⋮	...	⋮	⋮	...
23	*20/12/1992*	*80*	*yes*	*10*	...	*II*	*Blake*	...
24	*15/11/1993*	*78*	*no*	*11*	...	*II*	*Blake*	...
⋮	⋮	⋮	⋮	⋮	...	⋮	⋮	...

In this example, the joined table deals with the examinations rather than with the patients. An attribute-value learner could be used to learn hypotheses about the examinations, not about the patients.

This example reinforces a key representation rule in attribute-value learning: "Each row corresponds to a single individual, and vice-versa".

The second mistake is a meaningless column concatenation. This is more likely when a one-to-many relationship can be misinterpreted as several one-to-one relationships, i.e., when the practitioner is led to think that a non-determinate relationship is actually functional.

Example 5 *In our simplified medical domain, let us assume that the physician numbered the successive examinations (1, 2, 3, and so on) of each patient. Then given that each patient has a first examination, it is tempting to consider that there is a functional relationship from the patient to his or her "first" examination, "second" examination, and so on . This would result in a new patient table with concatenated columns: weight at the first examination, whether he or she smoked at the first examination, . . . , weight at the second examination, etc.*

Patient and his/her examinations (incorrect representation!)

pid	*name*	. . .	*"first" examination*			*"second" examination*			. . .
			weight	*smokes*	. . .	*weight*	*smokes*	. . .	. . .
I	*Smith*	. . .	*60*	*yes*	. . .	*64*	*yes*	. . .	. . .
II	*Blake*	. . .	*80*	*yes*	. . .	*78*	*no*	. . .	. . .
⋮	⋮	. . .	⋮	. . .	⋮	⋮	⋮	. . .	. . .

This could easily lead to an attribute-value learner generalising over a patient's weight at their i-th examination, which is very unlikely to be meaningful.

Two aspects should warn the user of such a representation problem: first, the number of columns depends on the dataset, and as a consequence, lots of columns are not defined for all individuals. Moreover, when the absolute numbering does not make sense, there is no functional relationship. Such a misunderstanding can be avoided by remembering that in an attribute-value representation, "each column is uniquely defined for each row".

Further relationships

The first complementary table can itself have a non-determinate relationship with another complementary table, and so on. Two approaches are available.

A first approach is to consider the first complementary table, the one having a one-to-many relationship, as a new primary table in a recursive propositionalisation.

Example 6 *In our simplified medical domain, the examination table has a one-to-many relationship with the additional_test table. The propositionalisation of* *the examination and additional test tables will lead to a new examination table completed with new features, such as a count of how many tests were bad:*

Examination and its additional_tests

eid	*pid*	*date*	*weight*	*smokes*	*BP*	...	*bad tests*	...
1	*I*	*10/10/1991*	*60*	*yes*	*10*	...	*1*	...
2	*I*	*04/06/1992*	*64*	*yes*	*12*	...	*0*	...
⋮	⋮	⋮	⋮	⋮	⋮	...	⋮	...

Then the propositionalisation of the patient table and the already propositionalised examination tables is performed, producing a new patient table completed with new features such as the mean value for each patient of the number of bad tests among all his or her examinations (f248):

Patient, his/her examinations and additional_tests

pid	*name*	...	*f60*	...	*f248*	...
I	*Smith*	...	2	...	*1*	...
⋮	⋮	...	⋮	...	⋮	...

It is not necessarily meaningful to aggregate at an intermediate level. An alternative is to join complementary tables first, and apply the aggregation at the individual level only. A variant consists in replacing the join by a propagation of the identifier, *i.e.* adding the identifier of the individual into all related tables. Both lead to a kind of "star schema" where the individual is directly linked to all complementary tables.

Example 7 *In our simplified medical domain, it is perhaps more interesting to first relate all additional tests to their patients, then aggregate on similar tests. First the complementary tables are joined:*

Additional_test and its examination

tid	*name*	*value*	*interpretation*	*eid*	*pid*	*weight*	...
t237	*red blood cells*	*35*	*bad*	*1*	*I*	*60*	...
t238	*radiography*	*nothing*	*good*	*1*	*I*	*60*	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	...
t574	*red blood cells*	*43*	*good*	2	*I*	*64*	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	...

Let us emphasize the difference with the propositionalised examination and its additional_tests table of example 6.

There is a one-to-many relationship from the patient table to that new additional_test and its examination table. Aggregation functions can be used to build *features such as the minimum percentage of red blood cells (f352):*

Patient, his/her additional_tests and examinations

pid	*name*	...	*f60*	...	*f352*	...
1	*Smith*	...	*2*	...	*35*	...
⋮	⋮	...	⋮	...	⋮	...

Finally, different propositionalisation approaches can be combined, by a simple join.

Future Directions

Propositionalisation explicitly aims at leaving attribute selection to the propositional learner applied afterward. The number of potential features is large. No existing propositionalisation system is able to enumerate all imaginable features. Historically existing approaches have focused on a subset of potential features, e.g., numerical aggregation functions without selection [4], selection based on a single elementary condition and existential aggregation [3, 5]. Most approaches can be combined to provide more features. The propositionalisation should be guided by the user.

Propositionalisation is closely related to knowledge representation. Specific representational issues require appropriate propositionalisation techniques, e.g., Perlich and Provost [8] introduce new propositionalisation operators to deal with high-cardinality categorical attributes. New data sources, such as geographical or multimedia data, will need an appropriate representation, and perhaps appropriate propositionalisation operators to apply off-the-shelf attribute-value learners.

Propositionalisation raises three fundamental questions. The first question is related to knowledge representation. That question is whether the user should adapt to existing representations, and accept a need to propositionalise, or whether data can be mined from the data sources, requiring the algorithms to be adapted or invented. The second question is whether propositionalisation is needed. Propositionalisation explicitly allows the user to contribute to the feature elaboration and invites him or her to guide the search thanks to that language bias. It separates feature elaboration from model extraction. Conversely, relational data mining techniques automate the elaboration of the relevant attributes during the model extraction, but at the same time leave less opportunity to select the features by hand.

The third issue is one of efficiency. A more expressive representation necessitates a more complex search. Relational learning algorithms face the same dilemma as attribute-value learning in the form of a choice between an intractable search in the complete search space and an ad-hoc heuristic/search bias (cf. Search Bias). They only differ in the size of the search space (cf. Hypothesis Space). Propositionalisation is concerned with generating the search space. Generating all potential features is usually impossible. So practitioners have to constrain the propositionalisation, e.g., by choosing the aggregation functions, the complexity of the selections, etc., by restricting the numbers of operations, and so on. Different operators fit different problems and might lead to differences in performance [6].

See also

Attribute, Feature Construction, Feature Selection, Feature Vector, Inductive Logic Programming, Language Bias, Learning from Structured Data, Multi-instance learning, Relational Learning, Statistical Relational Learning

References and Recommended Reading

[1] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple-instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89(1–2):31–71, 1997.

[2] S. Džeroski and N. Lavrač, editors. *Relational Data Mining*. Springer, 2001.

[3] P. Flach and N. Lachiche. 1BC: a first-order bayesian classifier. In S. Džeroski and P. Flach, editors, *Proceedings of the Ninth International Workshop on Inductive Logic Programming (ILP'99)*, volume 1634 of *Lecture Notes in Computer Science*, pages 92–103. Springer, 1999.

[4] A. J. Knobbe, M. de Haas, and A. Siebes. Propositionalisation and aggregates. In *Proceedings of the Sixth European Conference on Principles of Data Mining and Knowledge Discovery*, volume 2168 of *Lecture Notes in Artificial Intelligence*, pages 277–288. Springer-Verlag, 2001.

[5] S. Kramer, N. Lavrač, and P. Flach. Propositionalization approaches to relational data mining. In Džeroski and Lavrač [2], chapter 11, pages 262–291.

[6] M.-A. Krogel, S. Rawles, F. Železný, P. A. Flach, N. Lavrač, and S. Wrobel. Comparative evaluation of approaches to propositionalization. In T. Horváth and A. Yamamoto, editors, *Proceedings of the Thirteenth International Conference on Inductive Logic Programming*, volume 2835 of *Lecture Notes in Artificial Intelligence*, pages 197–214. Springer-Verlag, 2003.

[7] N. Lachiche. Good and bad practices in propositionalisation. In S. Bandini and S. Manzoni, editors, *Proceedings of Advances In Artificial Intelligence, Ninth Congress of the Italian Association for Artificial Intelligence (AI*IA'05)*, volume 3673 of *Lecture Notes in Computer Science*, pages 50–61. Springer, 2005.

[8] C. Perlich and F. Provost. Distribution-based aggregation for relational learning with identifier attributes. *Machine Learning*, 62:62–105, 2006.

[9] A. Srinivasan, S. Muggleton, R. D. King, and M. Stenberg. Theories for mutagenicity: a study of first-order and feature based induction. *Artificial Intelligence*, 85(1–2):277–299, 1996.

[10] M. Tomečková, J. Rauch, and P. Berka. Stulong data from longitudinal study of atherosclerosis risk factors. In P. Berka, editor, *Discovery Challenge Workshop Notes. ECML/PKDD'02.*, 2002.

Short entry: Relational

The adjective *relational* can have two different meanings in machine learning. The ambiguity comes from an ambiguity in database terminology.

Relational data mining refers to relational database, and is sometimes denoted *multi-relational* data mining. Indeed a relational database typically involves several relations (a relation is the formal name of a table). Those tables are often linked to each other, but the “relational” adjective does not refer to those relationships.

On the other hand, relational learning focuses on those relationships and intends to learn whether a relationship exists between some given entities.

See also

Propositionalisation, Relational Data Mining, Relational Learning