Toward Knowledge-Rich Data Mining

Pedro Domingos
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
pedrod@cs.washington.edu

1 The Knowledge Gap

Data mining has made tremendous progress in the last ten years. However, a large gap remains between the results a data mining system can provide and taking actions based on them. This gap must be filled by human work, which greatly limits the efficiency of the overall process and the scope of applicability of data mining. For example, data mining can reveal that a purchase of diapers at a supermarket is often accompanied by a purchase of beer, but it cannot hypothesize that the buyer is probably a new father, and propose other products appropriate to this demographic. In general, decision-making involves a chain of inferences, and, while we often have enormous quantities of data relevant to some inference steps, allowing us to automate them by learning from the data, we often have no data at all relevant to other steps, and these become the labor-intensive bottleneck. Clearly, the potential benefits of shrinking this "knowledge gap" could greatly exceed the benefits obtainable from further improving the parts of the process that are already efficient.

Another area where lack of portable, machine-understandable knowledge has a high cost is data pre-processing. It is by now a truism in the data mining community that most of the cost of a mining project is in data gathering, integration, cleaning, transformation, etc.; more generally, in setting up the problem in a way that can be solved by our data mining tools (classification, clustering, etc.). Despite this, progress in this area continues to be sparser than progress in further refining the modeling tools. A large part of this is due to the fact that modeling tools can be very general, making them attractive targets for research, while pre-processing tends to be very domain-specific. In effect, large quantities of precious human knowledge are incorporated into the pre-processing in a data mining project, but they are encoded in opaque scripts and SQL commands, and in the the next project we have to start again from scratch. Ideally, this knowledge would be encoded in a transparent, modular form that could be readily reused.

Even in the modeling phase, where the state of the art is most advanced, the use of knowledge remains effectively the bottleneck. Data mining efforts are seldom successful on the first attempt; a cycle of mining, examining the results and re-doing the mining with an improved model is required. Effectively, the role of this cycle is to incorporate the knowledge of the human data miner into the model. The process is iterative because it is much easier for humans to call up knowledge "on demand," in response to the results and failures of the data mining system, than to provide all the necessary knowledge *a priori*. However, as advanced as many tools are from a statistical and computational perspective, they do not provide humans with an easy interface to do this. Typically, knowledge is implicitly incorporated by trying out many different data mining algorithms, variations of them, parameter settings, combinations of techniques, etc. As a result, data mining requires experts with advanced degrees, which limits its diffusion. Allowing users to instead state their knowledge (or hypotheses) explicitly, for example by correcting the output and providing a justification, would improve ease of use, productivity, and reusability.

In sum, data mining as practiced today is mostly knowledge-poor; current tools do not facilitate the incorporation and reuse of knowledge, and this is perhaps the single greatest barrier to progress. While a number of previous research strands provide promising starting points to address this problem, much remains to be done. At the opposite end of the spectrum from data mining (and preceding it in time), expert systems use *only* manually encoded knowledge. They have not found widespread use because the cost of manually acquiring all the knowledge needed for useful results is too large, and even then the resulting inference is typically too brittle (Scott et al., 1991; Marcus, 1989; Henrion, 1987). Work on knowledge-intensive learning and theory revision seeks to strike a balance between the two, using data to refine an initial knowledge base, but still suffers from using logic as the representation language and the resulting brittleness (e.g., Bergadano and Giordana (1988); Pazzani and Kibler (1992); Ourston and Mooney (1994); Towell and Shavlik (1994)). Work on using knowledge to constrain association rule mining is useful, but limited in scope, as is work on inputting knowledge into the data mining system to discover interesting ways in which the data contradicts it (e.g., Srikant et al. (1997); Padmanabhan and Tuzhilin (1998); see also Section 7.1 in Domingos (1999)).

It is well known that purely empirical induction is impossible; some amount of knowledge must be combined with the data to produce non-trivial results. It is a remarkable fact that very weak knowledge, of the kind implicitly incorporated into standard machine learning algorithms, suffices to obtain useful results in many domains, when combined with substantial human work. But further automating the data mining process requires making it more knowledge-rich. Induction is effectively a way to leverage knowledge into more knowledge. It is a much more powerful lever than deduction, which can only make explicit the knowledge that is already implicitly in the data and knowledge base. But it is still the case that the more and better knowledge we leverage, the more and better knowledge we should be able to obtain as a result. It also follows that we should focus on obtaining the knowledge that gives us the greatest leverage for the least acquisition cost. In particular, the most useful knowledge complements what is easily mined from the available data, rather than repeating it; for example, knowledge relating observed variables to unobserved but important ones, or knowledge that is difficult to construct by greedy search.

At the University of Washington, we have recently begun to develop representations, algorithms and software tools to address this problem. What follows is a brief overview of them, an agenda for further work, and some discussion.

2 Markov Logic

Knowledge-rich data mining begins with representations that facilitate communication of knowledge between human and computer, and manipulation of knowledge by the computer. Unfortunately, there is a tension between these two desiderata. Languages for human use should be as rich as possible; the ideal language is natural language. But rich languages are difficult and inefficient to process automatically; reliable extraction of knowledge from natural language is beyond the state of the art. Most data mining systems use representations at the level of propositional logic, but this is clearly too limited, requiring extremely cumbersome representation of even simple regularities. (For example, in a domain with n objects, the statement that a relation is transitive requires n^3 propositions to encode, and does not generalize to domains with different objects.) First-order logic would seem to provide a good compromise: it is expressive enough to compactly represent much of what can be said in natural language, inference in it is a well-researched subject, and it is the basis for most expert systems, theory revision systems, and inductive logic programming. Unfortunately, it is too brittle. Knowledge, both hand-coded and mined from data, is imperfect, uncertain, and often contradictory; first-order logic is incompatible with all of these. Clearly, at a minimum we require a combination of the expressiveness of first-order logic with the robustness of probabilistic representations like Bayesian networks and Markov networks.

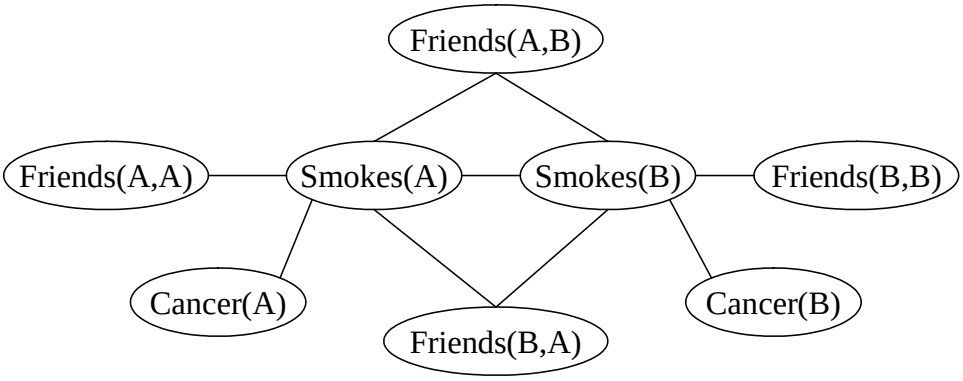

Figure 1: Markov network obtained by applying the formulas $\forall \mathtt{x}\ \mathtt{Smokes}(\mathtt{x}) \Rightarrow \mathtt{Cancer}(\mathtt{x})$ and $\forall \mathtt{x} \forall \mathtt{y}\ \mathtt{Friends}(\mathtt{x}, \mathtt{y}) \Rightarrow (\mathtt{Smokes}(\mathtt{x}) \Leftrightarrow \mathtt{Smokes}(\mathtt{y}))$ to the constants $\mathtt{Anna}(\mathtt{A})$ and $\mathtt{Bob}(\mathtt{B})$.

While combining logic and probability is a subject with a long history, it is only recently that practical realizations of this have begun to appear (e.g., Wellman et al. (1992); Muggleton (1996); Kersting and De Raedt (2001); Friedman et al. (1999); Taskar et al. (2002), etc.). One of the most powerful approaches to date is Markov logic, a simple but general combination of first-order logic and Markov networks (Domingos et al., 2006). A first-order knowledge base can be viewed as a set of hard constraints on the possible states of the world: if a state violates even one formula, it has zero probability. The basic idea in Markov logic is to soften these constraints: when a world violates one formula it is less probable, but not impossible. The fewer formulas a world violates, the more probable it is. Each formula has an associated weight that reflects its strength as a constraint: the higher the weight, the less likely is a state that violates the formula, other things being equal.

Thus, syntactically, Markov logic is simply first-order logic with a weight attached to each formula. Semantically, it can be viewed as a template for constructing Markov networks.[1] In a Markov network, the probability of a state is a normalized exponentiated weighted sum of features of the state:

$$P(X = x) = \frac{1}{Z} \exp\left(\sum_i w_i f_i(x)\right) \tag{1}$$

Together with a set of constants representing objects in the domain, a set of formulas in Markov logic defines a Markov network over Boolean variables. Each possible grounding of each predicate appearing in the formulas is one such variable, and each possible grounding of each formula (or clause) is a feature (with value 1 if the ground clause is true in the state, and 0 otherwise). The weight of a feature is the weight of the formula that originated it. For example, the formulas $\forall \mathtt{x}\ \mathtt{Smokes}(\mathtt{x}) \Rightarrow \mathtt{Cancer}(\mathtt{x})$ (smoking causes cancer) and $\forall \mathtt{x} \forall \mathtt{y}\ \mathtt{Friends}(\mathtt{x}, \mathtt{y}) \Rightarrow (\mathtt{Smokes}(\mathtt{x}) \Leftrightarrow \mathtt{Smokes}(\mathtt{y}))$ (friends have similar smoking habits) applied to the constants `Anna` and `Bob` (or `A` and `B` for short) yield the Markov network in Figure 1. Its features include $\mathtt{Smokes}(\mathtt{Anna}) \Rightarrow \mathtt{Cancer}(\mathtt{Anna})$, etc. Two nodes have an arc between them (and thus depend directly on each other) if the corresponding predicates appear together in some formula. Notice that, although the two formulas above are false as universally quantified logical statements, as weighted features of a Markov network they capture valid statistical regularities, and in fact represent a standard social network model.

We have developed a series of efficient algorithms for inference and learning in Markov logic. These algorithms build on the state of the art in satisfiability testing, Markov chain Monte Carlo, inductive logic

[1] Markov networks are also known as, or closely related to, Markov random fields, maximum entropy models, Gibbs distributions, exponential models, and log-linear models; and they have Boltzmann machines, conditional random fields, and logistic regression as special cases.

programming, and numeric optimization. Further details can be found in Domingos et al. (2006). Implementations of the algorithms are available in the open-source Alchemy package (Kok et al., 2006).

We have successfully used Alchemy for knowledge-rich data mining in a number of domains (Domingos et al., 2006). Typically, the process begins by writing down formulas representing known or hypothesized regularities in the domain. We have both encoded knowledge directly in logic and manually converted to logic knowledge contributed by others in natural language. Unlike in a conventional knowledge base, these formulas need not be always true or consistent with each other; to be useful, it suffices that they capture some of the dependency structure of the domain. In fact, many commonly used statistical models are easily represented as simple formulas in Markov logic, which both makes explicit the assumptions they encode and facilitates combining them with further knowledge. Weights can be hand-coded, learned from data, or a combination of the two. Similarly, formulas can be revised by hand, automatically, or both. Alchemy can also be used to perform data integration, by introducing the equality predicate and related axioms. In our experience, the use of Alchemy greatly increases the speed with which data mining applications can be developed.

3 An Agenda for Knowledge-Rich Data Mining

Markov logic and Alchemy form a starting point for a more knowledge-rich approach to data mining. Here we sketch what such an approach might look like, and some of the research issues involved.

Acquiring knowledge. If we accept the premise that starting from explicitly formulated knowledge is useful for data mining, acquiring this knowledge becomes an important goal. We envisage this being pursued in multiple ways:

- **By direct input.** Much knowledge can be acquired by direct entry by members of the relevant organization, the scientific community, or Web users at large. In turn, this knowledge can be vetted, applied and refined by further members of the community. The success of knowledge-sharing Web sites shows the extent to which this is possible, and facilitated by the Internet; the key going forward is to obtain knowledge in more structured form than is typical today. This can be done in a range of ways, from interfaces that directly translate their input into logic, to near-natural language with restricted vocabulary and grammar, to a more-knowledgeable subset of contributors codifying the contributions of others.
- **By extraction from text and the Web.** While text mining has long been a focus of research, text has generally been treated as purely a source of raw, low-level data. However, text is also an excellent direct source of knowledge at all levels of generality. While extracting knowledge from (say) textbooks and manuals is a difficult problem, Markov logic makes it a more realistic prospect, because it allows knowledge to be noisy and imperfect. We are beginning to work on this problem.
- **By use.** Many opportunities for acquiring knowledge arise once the mined knowledge is deployed. The users in the field routinely notice what it does wrong, how it could be corrected, and what knowledge is missing. Unfortunately, today there is generally no simple process by which these insights can be systematically incorporated into the knowledge base, and as a result they are often lost. This can be overcome by having data mining systems that are always online, continually incorporating not only new data but also new hypotheses, for example in the form of Markov logic formulas that explain a mistake made by the system.

Refining knowledge. The core of knowledge-rich data mining is providing a richer interface between the data mining system and its users. By allowing the user to explicitly make new statements or modify

previous ones in response to the results of mining, we can reduce the number of mine-refine loops required to reach good results, and reach better results in less time. Further, by allowing the user to easily reformulate, consolidate and trim the results of mining, these can be better integrated with the existing knowledge, and made more stable, robust, and portable.

Reusing knowledge. Data mining projects today are typically standalone; the results of a project are never reused, and each new project starts from scratch (modulo experience and perhaps some supporting code). This significantly limits the depth and breadth of knowledge that can be acquired. One of our goals with Alchemy is to support the development of reusable knowledge bases in Markov logic. Given such a repository, the first step of a data mining project becomes the selection of relevant knowledge. This may be used as is or manually refined. A new knowledge base is initiated by writing down plausible hypotheses about the new domain. The core process of inducing new knowledge for the task can now start from a much stronger base. Formula weights and structure for the supporting knowledge bases may be adjusted based on data from the new task. Over time, more knowledge becomes available, and existing knowledge is refined and specialized to different (sub)domains.

4 Objections

A number of objections can be raised to the notion that knowledge-rich data mining is necessary or useful. This section briefly addresses the main ones.

With enough data, you don't need knowledge. The amount of data required to sample an instance space with constant density increases exponentially with its dimensionality. Thus, in the high-dimensional problems that are the main focus of data mining, the asymptote in the learning curve may not be reached even with extremely large quantities of data. Even if sufficient data is available, the computational cost of exploiting it will often be too high. In particular, there is a trade-off between fast, greedy search, which is limited in the patterns it can discover, and more exhaustive search, which is typically too expensive. The use of knowledge helps overcome this tradeoff, by focusing the search in promising areas and providing component patterns that would be difficult to find greedily. Most of all, as mentioned earlier, in many applications the data available only covers some parts of the inference chain from evidence to actions, and the only alternative to incorporating knowledge is human intervention.

Knowledge-rich data mining doesn't scale. If knowledge is used to constrain the search for new patterns, it increases rather than reduces the scalability of learning. While inference in a language that combines first-order logic and probability can be expensive, state-of-the-art satisfiability solvers like the one used in Alchemy can solve hard problems with hundreds of thousands to millions of variables in minutes. Weighted satisfiability testing performs probabilistic inference (finding the most likely state given evidence) no less efficiently than the purely logical case (and potentially more, if some previously hard constraints are softened). In many cases, the knowledge relevant to a specific problem is a small subset of the knowledge available, and the problem can be divided into retrieving this knowledge, which can be done in linear time, and using it, where the higher cost of inference is incurred. While there is much work to do in further scaling inference and learning in languages like Markov logic, the state of the art is already sufficient for many applications.

We'll never be able to acquire enough knowledge. The goal of knowledge-rich data mining is not to provide *a priori* all the knowledge that might be required to preprocess data, interpret and refine results, turn them into actions, etc. Rather, it is to support a feedback loop by which a small amount of initial knowledge can be bootstrapped into more knowledge by mining, which can in turn be complemented by more human-supplied knowledge to allow further mining, etc. Neither pure manual acquisition nor

purely empirical induction suffice to obtain all the knowledge required for decision-making in complex domains. Rather, a fine-grained, iterative combination of the two offers the best chances of success.

Some knowledge is hard to make explicit. Humans can perform many tasks without being able to explain how they do it, and in many cases data mining can learn to perform them by observing the human's input-output behavior. However, even in this case there is often much relevant knowledge that can be easily stated explicitly, and will greatly help the data mining process. For example, digit recognition is greatly facilitated by explicitly incorporating our knowledge that digits are invariant to translation and scaling, instead of requiring the system to learn this at the same time it learns the digits' structure. Also, even when humans have difficulty explaining how they perform a task in general, they often find it quite easy to correct mistakes in specific instances, and to explain what went wrong. Incorporating this advice and generalizing it can greatly aid learning.

5 Conclusion

We can envisage a time when knowledge-rich data mining is the rule rather than the exception. Instead of each new data mining project starting from scratch, it will build on the large repository of knowledge accumulated by previous projects and by direct acquisition from humans. Because of the knowledge it can draw on, a data mining system will be able to discover deeper patterns, and directly connect them to the actions that should result. The new knowledge discovered will in turn be added to the repository, providing a better starting point for future data mining efforts.

References

Bergadano, F., & Giordana, A. (1988). A knowledge-intensive approach to concept induction. *Proceedings of the Fifth International Conference on Machine Learning* (pp. 305–317). Ann Arbor, MI: Morgan Kaufmann.

Domingos, P. (1999). The role of Occam's razor in knowledge discovery. *Data Mining and Knowledge Discovery*, *3*, 409–425.

Domingos, P., Kok, S., Poon, H., Richardson, M., & Singla, P. (2006). Unifying logical and statistical AI. *Proceedings of the Twenty-First National Conference on Artificial Intelligence* (pp. 2–7). Boston, MA: AAAI Press.

Friedman, N., Getoor, L., Koller, D., & Pfeffer, A. (1999). Learning probabilistic relational models. *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence* (pp. 1300–1307). Stockholm, Sweden: Morgan Kaufmann.

Henrion, M. (1987). Some practical issues in constructing belief networks. *Proceedings of the Third Conference on Uncertainty in Artificial Intelligence* (pp. 161–173). New York, NY: Elsevier.

Kersting, K., & De Raedt, L. (2001). Towards combining inductive logic programming with Bayesian networks. *Proceedings of the Eleventh International Conference on Inductive Logic Programming* (pp. 118–131). Strasbourg, France: Springer.

Kok, S., Sumner, M., Richardson, M., Singla, P., Poon, H., & Domingos, P. (2006). *The Alchemy system for statistical relational AI* (Technical Report). Department of Computer Science and Engineering, University of Washington, Seattle, WA. http://alchemy.cs.washington.edu.

Marcus, S. (1989). Special issue on knowledge acquisition. *Machine Learning, 4*.

Muggleton, S. (1996). Stochastic logic programs. In L. De Raedt (Ed.), *Advances in inductive logic programming*, 254–264. Amsterdam, Netherlands: IOS Press.

Ourston, D., & Mooney, R. J. (1994). Theory refinement combining analytical and empirical methods. *Artificial Intelligence*, *66*, 273–309.

Padmanabhan, B., & Tuzhilin, A. (1998). A belief-driven method for discovering unexpected patterns. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining* (pp. 94–100). New York, NY: AAAI Press.

Pazzani, M., & Kibler, D. (1992). The utility of knowledge in inductive learning. *Machine Learning*, *9*, 57–94.

Scott, A. C., Clayton, J. E., & Gibson, E. L. (1991). *A practical guide to knowledge acquisition*. Reading, MA: Addison-Wesley.

Srikant, R., Vu, Q., & Agrawal, R. (1997). Mining association rules with item constraints. *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining* (pp. 67–73). Newport Beach, CA: AAAI Press.

Taskar, B., Abbeel, P., & Koller, D. (2002). Discriminative probabilistic models for relational data. *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence* (pp. 485–492). Edmonton, Canada: Morgan Kaufmann.

Towell, G. G., & Shavlik, J. W. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence*, *70*, 119–165.

Wellman, M., Breese, J. S., & Goldman, R. P. (1992). From knowledge bases to decision models. *Knowledge Engineering Review*, *7*.