Structured Machine Learning: Ten Problems for the Next Ten Years

Pedro Domingos
Department of Computer Science and Engineering
University of Washington

1 Statistical Predicate Invention

Predicate invention in ILP and hidden variable discovery in statistical learning are really two faces of the same problem. Researchers in both communities generally agree that this is a key (if not the key) problem for machine learning. Without predicate invention, learning will always be shallow. In essence, every word in the dictionary is an invented predicate, with many layers of invention between it and the sensory percepts on which it is ultimately based. Unfortunately, progress to date has been limited. The consensus seems to be that the problem is just too hard, and it's not clear what to do about it. However, combining predicate invention and latent variable discovery into the single problem of statistical predicate invention may lead to new breakthroughs. (One is reminded of Eisenhower's saying: "If you can't solve a problem, magnify it.") Considering statistical and logical aspects simultaneously gives us both more opportunities and more constraints to work with. It's also a natural continuation of the statistical relational learning agenda. For some preliminary ideas, see our paper "Statistical Predicate Invention," in ICML-2007.

2 Generalizing Across Domains

Machine learning has traditionally been defined as generalizing across tasks from the same domain, and in the last few decades we've learned to do this quite successfully. However, the glaring difference between machine learners and people is that people can generalize across domains with great ease. For example, Wall Street hires lots of physicists who know nothing about finance, but they know a lot about particle physics and the math it requires, and somehow this transfers quite well to pricing options and predicting the stock market. Machine learners can do nothing of the kind. If the predicates describing two domains are different, there's just nothing the learner can do in the new domain given what it learned in the old one. The key insight that seems to be missing is that domains have structural similarities, and we can detect them and exploit them. For example, two domains might be described by the same formula(s), but over different predicates, and having learned the formula in one domain it should be easier to rediscover it in another. This seems like an ideal challenge for relational learning, since in some sense it's "extreme relational learning": we're not just using relations to generalize, we're using relations between relations. DARPA has recently started a project in this area, but so far we've only scratched the surface. (For a good example, see "Mapping and Revising Markov Logic Networks for Transfer Learning," by Lily Mihalkova and Ray Mooney, in AAAI-2007.)

3 Learning Many Levels of Structure

So far, in statistical relational learning (SRL) we have developed algorithms for learning from structured inputs and structured outputs, but not for learning structured internal representations. In both ILP and statistical learning, models typically have only two levels of structure. For example, in support vector machines the two levels are the kernel and the linear combination, and in ILP the two levels are the clauses and their conjunction. While two levels are in principle sufficient to represent any function of interest, they are an extremely inefficient way to represent most functions. By having many levels and reusing structure we can often obtain representations that are exponentially more compact. For example, a BDD (Boolean decision diagram) can represent parity with a linear number of operations, while clausal form requires an exponential number. This compactness should also be good for learning, but nobody really knows how to learn models with many levels. The human brain has many layers of neurons, but backpropagation seldom works with more than a few. Hinton and others have begun to work on learning "deep networks," but they are not very deep yet, and they are only for unstructured data. Recursive random fields, proposed in our IJCAI-2007 paper, are a potentially "deep" SRL representation, but learning them suffers from the limitations of backpropagation. Clearly this an area where there is much to be discovered, and where progress is essential if are to ever achieve something resembling human learning.

4 Deep Combination of Learning and Inference

Inference is crucial in structured learning, but research on the two has been largely separate to date. This has led to a paradoxical state of affairs where we spend a lot of data and CPU time learning powerful models, but then we have to do approximate inference over them, losing some (possibly much) of that power. Learners need biases and inference needs to be efficient, so efficient inference should be the bias. We should design our learners from scratch to learn the most powerful models they can, subject to the constraint that inference over them should always be efficient (ideally real-time). For example, in our ICML-2005 paper "Naive Bayes Models for Probability Estimation," we learned models that were as accurate as Bayesian networks, but where inference was always linear-time, as opposed to worst-case exponential. In SRL efficient inference is even more important, and there is even more to be gained by making it a goal of learning.

5 Learning to Map between Representations

An application area where structured learning can have a lot of impact is representation mapping. Three major problems in this area are entity resolution (matching objects), schema matching (matching predicates) and ontology alignment (matching concepts). We have algorithms for solving each of these problems separately, assuming the others have already been solved. But in most real applications they are all present simultaneously, and none of the "one piece" algorithms work. This is a problem of great practical significance because integration is where organizations spend most of their IT budget, and without solving it the "automated Web" (Web services, Semantic Web, etc.) can never really take off. It seems like an ideal problem for joint inference: if two objects are the same, then perhaps the fields they appear in are the same, and in turn the concepts containing those fields may be the same, and vice-versa. And learning for joint inference is what SRL is all about, so this could be a "killer app." Beyond one-to-one mapping lies the deeper challenge of learning to convert from one representation of a problem in logic to another. Humans can recognize when two sets of formulas are essentially saying the same thing, even when they are not logically equivalent.

AI systems should be able to do the same.

6 Learning in the Large

Structured learning is most likely to pay off in large domains, because in small ones it is often not to difficult to hand-engineer a “good enough” set of propositional features. So far, for the most part, we have worked on micro-problems (e.g., identifying promoter regions in DNA); our focus should shift increasingly to macro-problems (e.g., modeling the entire metabolic network in a cell). We need to learn “in the large,” and this does not just mean large datasets. It has many dimensions: learning in rich domains with many interrelated concepts; learning with a lot of knowledge, a lot of data, or both; taking large systems and replacing the traditional pipeline architecture with joint inference and learning; learning models with trillions of parameters instead of millions; continuous, open-ended learning; etc.

7 Structured Prediction with Intractable Inference

Max-margin training of structured models like HMMs and PCFGs has become popular in recent years. One of its attractive features is that, when inference is tractable, learning is also tractable. This contrasts with maximum likelihood and Bayesian methods, which remain intractable. However, most interesting AI problems involve intractable inference. How do we optimize margins when inference is approximate? How does approximate inference interact with the optimizer? Can we adapt current optimization algorithms to make them robust with respect to inference errors, or do we need to develop new ones? We need to answer these questions if max-margin methods are to break out of the narrow range of structures they can currently handle effectively.

8 Reinforcement Learning with Structured Time

The Markov assumption is good for controlling the complexity of sequential decision problems, but it is also a straitjacket. In the real world systems have memory, some interactions are fast and some are slow, and long uneventful periods alternate with bursts of activity. We need to learn at multiple time scales simultaneously, and with a rich structure of events and durations. This is more complex, but it may also help make reinforcement learning more efficient. At coarse scales, rewards are almost instantaneous, and RL is easy. At finer scales, rewards are distant, but by propagating rewards across scales we may be able to greatly speed up learning.

9 Expanding SRL to Statistical Relational AI

We should reach out to other subfields of AI, because they have the same problems we do: they have logical and statistical approaches, each solves only a part of the problem, and what is really needed is a combination of the two. We want to apply learning to larger and larger pieces of a complete AI system. For example, natural language processing involves a large number of subtasks (parsing, coreference resolution, word sense disambiguation, semantic role labeling, etc.). So far, learning has been applied mostly to each one in isolation, ignoring their interactions. We need to drive towards a solution to the complete problem. The same applies to robotics and vision, and other fields. We need to avoid falling into local optima in our research: once a problem is solved “80/20,” we should move on to the next larger one that includes it, not continue to refine our solution with diminishing returns. Our natural tendency to do the latter greatly slows down the progress of research. Moreover, the best solutions to subproblems taken in isolation are often not the best ones in combination. Because of this, refining solutions to subproblems can in fact be counterproductive—digging deeper into the local optimum instead of escaping it.

10 Learning to Debug Programs

Machine learning is making inroads into other areas of computer science: systems, networking, software engineering, databases, architecture, graphics, HCI, etc. This is a great opportunity to have impact, and a great source of rich problems to drive the field. One area that seems ripe for progress is automated debugging. Debugging is extremely time-consuming, and was one of the original applications of ILP. However, in the early days there was no data for learning to debug, and learners could not get very far. Today we have the Internet and huge repositories of open-source code. Even better, we can leverage mass collaboration. Every time a programmer fixes a bug, we potentially have a piece of training data. If programmers let us automatically record their edits, debugging traces, compiler messages, etc., and send them to a central repository, we will soon have a large corpus of bugs and bug fixes. Of course, learning to transform buggy problems into bug-free ones is a very difficult problem, but it's also highly structured, noise-free, and the grammar is known. So this may be a "killer app" for structured learning.