Machine learning is one of the most important applications of computers today, but, like the Internet in the mid-90's, its full potential is far from realized. Most approaches to date only learn from sets of independent examples, where each example is a simple vector of feature values. However, most real-world problems are interdependent and highly structured. Consider the task of labeling web pages: in addition to the words on the page, the links to other web pages are highly informative, as are the labels of the linked pages. Flattening such a problem into vectors throws away key information. The ever increasing availability and variety of data leads to more and more structured information that is poorly handled by flat approaches. My work has led to more powerful representations and more efficient algorithms for statistical relational learning, helping the field move to richer solutions for these rich problems.

When moving beyond simple classification, one of the greatest hurdles for statistical machine learning is the problem of inference. For most models of moderate size and complexity, exact inference is intractable, while approximate inference may be prohibitively slow or inaccurate. My work has shown that by taking inference cost into account while learning, you can learn accurate models that still guarantee efficient, exact inference. I plan to extend these results to the relational setting, where large, complex models lead to particularly difficult inference problems.

My overall approach is to find general solutions to important, real-world problems by bridging theory and practice. I believe in thorough empirical evaluation, along with releasing code and data whenever possible so that the experiments are reproducible.

Statistical Relational Learning

Most real-world problems are both complex and uncertain. Rich relational structure is traditionally handled with first-order logic; uncertain data (such as that generated by stochastic processes, corrupted by noise, or incompletely understood) is typically modeled using probability. The goal of statistical relational learning is to combine first-order logic and probability, providing a more powerful and general framework for machine learning. Instead of reducing complex problems to vectors, a statistical relational approach allows us to work directly with the natural problem structure, often yielding better results with less work.

I am interested in developing general-purpose statistical relational representations and algorithms. While it may always be possible to create specially engineered solutions that outperform general-purpose methods, we are likely to see the biggest gains from improving the performance and reliability of our best general-purpose methods. A good analogy is microprocessors: while specialized chips still have their purpose, general-purpose microprocessors (x86, ARM, etc.) are sufficiently cheap and effective for the vast majority of applications. One of the most promising general-purpose solutions to date is Markov logic, a language for statistical relational representation, learning, and reasoning. In its simplest form, Markov logic just attaches a weight to each formula in a first-order knowledge base. Given a set of constants for a domain, the sum of the weights of satisfied formulas determines the relative probability of a possible world. Weights and formulas can be learned or revised from data using ideas from convex optimization and inductive logic programming. Markov logic is also one of the most popular general-purpose solutions. According to CiteSeer, the journal paper that introduced Markov logic is the most cited computer science paper

of 2006. Alchemy, the open-source implementation of Markov logic to which I have contributed, has been downloaded over 3300 times. Markov logic has been used to win the LLL-2005 information extraction competition and two best paper awards (CIKM-2007 and ILP-2008). Applications of Markov logic include state-of-the-art solutions for natural language processing, computational biology, robot mapping, and more.

My work to date has improved and extended Markov logic in two key ways. First, I developed a multi-layer generalization of Markov logic called recursive random fields (RRFs). RRFs resolve a key inconsistency in the Markov logic representation, represent many probability distributions exponentially more compactly than MLNs, and allow for new, more flexible approaches to learning. One can view a logical knowledge base as a single formula tree, in which each leaf is an atom and each interior node is a conjunction or disjunction, or an existential or universal quantifier. Markov logic "softens" this knowledge base by replacing the top-level universal quantifiers and conjunctions with a log linear model akin to a noisy AND, but all lower levels remain purely logical. Recursive random fields make every level of the formula tree probabilistic, creating softened disjunctions and existentials. An important application of RRFs is detecting and repairing database errors through probabilistic integrity constraints. For example, consider the inclusion constraint "every person who manages a project supervises an employee", or in first-order logic, $\texttt{Manages}(\texttt{x}, \texttt{y}) \Rightarrow \exists \texttt{z}.\texttt{Supervises}(\texttt{x}, \texttt{z})$. In an RRF, we can use a noisy existential to learn and reason about the number of employees each manager is expected to supervise, while MLNs crudely treat the entire rule as true or false.

Recursive random fields have many nice theoretic properties, but under current methods, learning is very sensitive to the initial weights and inference is significantly slower than for MLNs. The key to making RRFs practical is making the learning and inference robust and efficient. So my next work was on making MLNs more practical, in hopes of eventually applying those ideas to RRFs. At the time, the weight learning algorithms for Markov logic were rather crude and frequently failed on hard problems. Other researchers had been forced to simplify their models or set weights via various hacks. I adapted several convex optimization methods to cope with the particular challenges of Markov logic. The algorithms I developed are now the standard approach to Markov logic weight learning.

Learning for Efficient Inference

One of the important remaining challenges for Markov logic, and in fact, for graphical models in general, is efficient inference. Exact inference computes the most accurate probabilities, but is often intractable since probabilistic inference is #P-complete. Approximation algorithms range from Markov chain Monte Carlo (MCMC) sampling to belief propagation, each with its distinct set of advantages and disadvantages.

My work addresses the problem of efficient inference for propositional graphical models, with the goal of eventually extending these results to Markov logic and similar representations. Rather than trying to solve the hard inference problem directly, I looked for accurate representations that still allow for efficient, exact inference. Through experiments on 50 datasets, I showed that naive Bayes mixture models offer similar accuracy to Bayesian networks while providing predictable, linear-time inference instead of exponential.

Since naive Bayes mixture models are still limited in what they can represent compactly, I developed an alternate approach that finds Bayesian networks with efficient inference. Instead of

restricting the treewidth, which only gives an upper bound on inference complexity, I used the actual inference cost as determined by compiling the network to an arithmetic circuit. Arithmetic circuits can exploit local structure, such as context-specific independence. I took a greedy algorithm for learning Bayesian networks with context-specific independence and changed the objective function to penalize candidate models by their inference costs. This trades off accuracy and efficiency without artificially restricting the model class. I avoid compiling each candidate model from scratch by incrementally modifying the circuit to match each change in the network. By combining learning and inference, I was able to learn models that appeared very complex (treewidth > 100) but still allowed for very efficient inference (< 100ms).

Future Work

These advances to make machine learning more powerful and more practical are only the beginning. I am currently working on the first approximate inference method to use arithmetic circuits. Compiling probabilistic models into arithmetic circuits for exact inference often fails because the models are too complex or lack sufficient local structure. An alternative is to introduce local structure and other approximations as necessary during the compilation process, trading accuracy for efficiency. Markov logic networks offer a particularly exciting application of such dynamic inference methods, since their structures are determined by highly structured sets of formulas.

For the future, I intend to take rich, statistical relational representations and efficient algorithms to the next level. People who want to apply machine learning should be able to express their knowledge and data naturally, learn a good model automatically, and use the model to make predictions effficiently. Machine learning has come a long way towards this goal for the subproblem of classification using vector data. Statistical relational learning brings greater challenges, but also greater opportunities to use our data to solve problems in bioinformatics, natural language processing, social network analysis, user interface design, and more. One of the things that excites me about machine learning is its power to solve such a wide range of problems in different fields. But to have the greatest impact, machine learning must be more flexible and be usable by people who are not machine learning experts. That is what I want to help accomplish.

Other Research

While interning at Microsoft Research, I looked at the adversarial problem of modifying behavior to avoid classifier detection (e.g., spammers changing their emails to bypass spam filters). To analyze the hardness of attacking different classifiers, I formulated the adversarial classifier reverse engineering (ACRE) learning problem and proved several results for linear classifiers. I also developed simple but novel ways to reverse engineer spam filters, with or without the ability to send test messages through the filter. These practical attacks offer insights into what make spam filters weak, as well as methods for evaluating spam filter vulnerability. This work is already well-cited, and led to invited talks at Oregon State University and the University of Cagliari, Italy.

In addition to spam filtering, I have worked on evaluating collaborative filtering algorithms and developing effective methods for desktop activity recognition. These applications of machine learning have great potential to reduce information overload.

DANIEL LOWD **TEACHING STATEMENT**

My desire to teach stems from a pretty basic drive: I'm excited about computer science, and I love sharing this excitement by helping people learn. (My friends and family will readily attest to this, having learned more about everything from recursion to how spam filters work than they ever wanted to know.) My experiences have prepared me to teach graduate and undergraduate courses, advise students, and contribute to other parts of the educational process. However, in all that I have learned about teaching, the most important lesson has been that there is always more to learn. I look forward, not only to sharing my passion with students, but to further developing my skills and techniques as a teacher.

For my undergraduate studies, I attended a small college that emphasized quality in teaching. I was able to learn a number of techniques from watching the professors there: one math teacher would start every class with a "fun fact"; a physics teacher used near-light-speed rhinoceroses to explain relativity; a computer science teacher worked his students' names into examples to help them stay awake. But for the most part, the professors just gave clear lectures with appropriate examples and in-class questions, provided good supporting materials and problem sets, and were available for questions outside of class. In my senior year, I took an advanced algorithm class which had, as one component, teaching the class for one day about a particular algorithm or data structure. This was my first real introduction to teaching. It was harder than it looked: knowing the material and a few classroom techniques was only the beginning. The real trick seemed to be knowing which techniques to apply when and which explanations would work best. There is no shortcut or gimmick for good communication, just patience, practice, and if you're lucky, talent.

As a graduate student, I have had the opportunity to hone my teaching skills in many different contexts. As a teaching assistant for the University of Washington graduate AI course (CSE 573), I held office hours, designed and graded assignments, and gave several lectures. I gave guest lectures in a Carnegie Mellon class on Markov logic (10-803) and a different section of CSE 573 at UW. I also had the pleasure of tutoring undergraduates for five terms, and I am currently finishing a book on Markov logic (my research area) that has already been used as a course text at Carnegie Mellon and the University of Wisconsin-Madison. These experiences have strengthened my ability to explain concepts in a variety of contexts, from informal, interactive discussions to formal, written prose. I believe that using a variety of formats and styles in teaching is important for teaching students with varied perspectives and learning styles. Furthermore, some concepts simply call for different teaching methods than others. I am open to using whichever approaches work and adapting to the needs of each class. In order to determine what works best, I have sought out student feedback, especially criticism. Continually improving my teaching skills is very important to me; good teaching is an ongoing process.

My perspective on advising has been shaped by my experiences with my own advisor as well as three other academic mentors with whom I have collaborated and authored papers. As I see it, the role of an advisor is not to drive the research, but to provide direction. The level of direction will naturally depend on the student, and should vary through the course of study. For this direction to be well-received, the advisor must respect the student's development into a researcher in her own right.

I would be interested in teaching courses on artificial intelligence and machine learning, both at the graduate and undergraduate level. I would also be interested in teaching a more advanced course on statistical relational learning or Markov logic, based in part on the book I am coauthoring. I am open to teaching other courses, such as introductory programming or data structures. The most basic concepts are often the most fundamental and important, which makes teaching them exciting and challenging.