Letter from the Special Issue Editor

This issue explores how machine learning (ML) has become an important application area for data management research and practice over the last few years.

The issue begins with two articles on the topic that has been at the forefront of the convergence of data management and ML research: probabilistic databases. When they wrote their revolutionary 2004 VLDB paper "Efficient Query Evaluation on Probabilistic Databases", I'm not sure whether or not Nilesh Dalvi and Dan Suciu realized how deeply connected their ideas were with the probabilistic structures that underlie much of modern ML, but those connections are becoming clear today. Fittingly, this issue opens with an article written by Eric Gribkoff, Dan Suciu, and Guy Van den Broeck that explains and explores some of these connections. They consider how the idea of lifted inference on probabilistic graphical models is nothing more than the evaluation of an SQL aggregate query in a probabilistic database, and how inference for ML can be performed using a probabilistic database based on this equivalence. The second article, by Dan Olteanu and Sebastiaan J. van Schaik, describes the ENFrame system. ENFrame is a so-called *probabilistic programming* platform that is fundamentally based upon ideas and algorithms that came out of research into probabilistic databases. "Probabilistic programming" is an exciting idea, still in its infancy, that is being pursued by researchers in several communities. The idea is that a user should be able to write code in a high-level programming language that manipulates uncertain or probabilistic data. The platform itself—and not the programmer—should be responsible for drawing inferences about the probability distributions induced.

The next article considers a promising ML-oriented use case for relational database-like engines: as platforms for very large-scale knowledge extraction. Chris Re et al. from Stanford describe DeepDive, which is a system for declaratively specifying (and running) large-scale knowledge-base construction workflows. While no one would describe DeepDive as a database system, it makes extensive use of ideas from database systems, including the use of SQL as a language to perform feature extraction, and the use of a relational engine to perform scalable statistical inference.

In the next article, Daisy Wang and her students at the University of Florida expand on this second idea, and consider how a database can be an excellent platform for running large-scale machine learning algorithms, with a particular emphasis on the learning of graphical models.

Recently, there has been a loud technical argument over the "correct" platform for implementing and running large-scale parallel or distributed machine learning codes, with many proposals from within and without the database community. The next two articles describe specific systems from within the database community for this task. The first, from IBM Almaden, describes SystemML, which is a programming language and platform for writing statistical codes, with an emphasis on supporting distributed linear algebra. The focus in the article here is on SystemML's optimizer, which takes as input a computation described in SystemML's programming language, and generates an efficient execution plan. The second article, from Brown, describes TupleWare, which is a database-like system with a focus on the efficient execution of UDFs. One of the central ideas in the TupeWare system is the compilation of high-level dataflow plans into low-level LLVM code, with the idea being that low-level code that can be executed directly is much more efficient than the classical, Volcano-style iterators.

The issue closes with an article from Duke that describes Cumulon. Deploying large-scale ML and statistical analysis in the cloud is difficult because of the bewildering array of choices facing an analyst: What machines to use? How much work should be given to each parallel task? How many tasks should be defined? Cumulon makes these choices using a database-like optimization process, where, in addition to figuring out the execution steps, the optimizer also figures out how to purchase and utilize the compute resources.

Taken together, these articles represent a broad sampling of some of the work at the intersection of databases, declarative systems, and ML. I hope you have as much fun reading the article as I did putting it together.

Chris Jermaine
Rice University