Machine Learning and Software Engineering in Health Informatics

David A. Clifton*, Jeremy Gibbons†, Jim Davies†, Lionel Tarassenko*

*Institute of Biomedical Engineering, Department of Engineering Science, University of Oxford, Oxford, UK
†Department of Computer Science, University of Oxford, Oxford, UK

***Abstract*—Health informatics is a field in which the disciplines of software engineering and machine learning necessarily co-exist. This discussion paper considers the interaction of software engineering and machine learning, set within the context of health informatics, where the scale of clinical practice requires new engineering approaches from both disciplines. We introduce applications implemented in large on-going research programmes undertaken between the Departments of Engineering Science and Computer Science at Oxford University, the Oxford University Hospitals NHS Trust, and the Guy's and St Thomas' NHS Foundation Trust, London.**

I. Introduction

A. Health Informatics

Health informatics is a broad field [1]–[3], involving the acquisition and manipulation of healthcare-related data. Modern healthcare practice involves the acquisition of ever-larger quantities of data from all aspects of a patient's life: "primary care" datasets comprise information from patients' interactions with general practitioners; clinical datasets may exist from patient stays in hospitals or clinics; social-care datasets may record the support for patients with long-term conditions, which typically takes place within their own homes. This increase in the *quantity* of patient data occurs alongside a similar increase in the *complexity* of the data.

Patient data can be derived from many sources, including genomic data, the results of medical tests, notes taken from consultations, and sensor-data acquired automatically from medical instruments and monitors. Patient data vary in their perceived quality: some data are "clinically validated" (i.e., recorded by trained healthcare workers), while others are "patient-owned", having been acquired by the patients themselves. Finally, additional complexity arises from the resolution of patient data, which varies between high-resolution data from images and patient-worn sensors, to the low-resolution data acquired by manual clinical observation of a patient.

This great increase in the scale and complexity of healthcare-related data presents a challenge for clinical practice and medical research, where conventional approaches cannot exploit the information potentially available. The field of health informatics therefore aims to provide large-scale linkage of disparate, potentially incomplete and noisy, healthcare datasets, and then perform large-scale inference using the linked datasets to support clinical practice and medical research. This first task of dataset linkage traditionally falls within the discipline of software engineering; the second task of inference falls within the discipline of machine learning.

B. Machine Learning

Machine learning is sometimes seen as a development of the "artificial intelligence" movement; modern practitioners often see their work as having overlap with statistics, applied mathematics, statistical physics, electrical and information engineering, and computational biology. The goal of machine learning is, as its name suggests, to identify patterns in data, and then to perform useful inference using those patterns that have been learned.

Such inference typically takes the form of *classification*, in which previously-unseen data are determined as belonging to one of a number of classes; or *regression*, in which previously-unseen data are used to predict the behaviour of one or more random variables. An example of classification within healthcare informatics is the determination of whether a hospital patient is "physiologically stable" or "physiologically deteriorating" based on their vital signs [4], [5]; an example of regression is the prediction of a patient's respiration rate based on physiological data acquired from sensors [6].

Inference is often set within a statistical framework, in which statistical models are learned from training sets of example data. The generally tendency in machine learning has been towards adoption of Bayesian methodology, in which reasoning is performed within the principled framework of Bayesian probabilistic inference. This approach offers many advantages over conventional "frequentist" techniques, which were the norm in machine learning in the 1980s and 1990s, at which time the field was more commonly referred to as "artificial intelligence". Since that period, advances in computational power and numerical algorithms have allowed the (typically high-dimensional) integrations that are a necessary part of Bayesian inference to be performed in real-time, and hence Bayesian methods have rapidly become popular. Bayesian methods provide many advantages over conventional frequentist techniques, which we will describe in the coming sections, where they will prove relevant to the interaction between machine learning and software engineering.

C. Case Studies

The case studies discussed within this paper are set within the context of the *Hospital of the Future* programme [7], one of three "Grand Challenge" programmes in information-driven healthcare, funded by the RCUK "Digital Economy" scheme. The aim of the programme is to create a new model of hospital care built on integrated patient monitoring and management. The "digital hospital" includes new software systems that wirelessly collate continuous real-time patient information from a number of sources (including wearable Bluetooth sensors), integrate the data to generate a patient status index, and make the results available to clinicians via hand-held devices such as wireless PDAs. The new system helps to guide and prioritise hospital activity in response to the patient status, and provides early warning of patient deterioration. Clinicians therefore obtain relevant, real-time information at the point of care, ensuring a reduction in adverse events such as unexpected cardiac arrests or unplanned admissions to Intensive Care.

II. Software Engineering in Machine Learning

A. The State of Software in Machine Learning

The production of quality machine learning software is currently in a state of infancy, and therefore has much to learn from advances in software engineering, for example in model-driven engineering. A large proportion of published work in machine learning comprises investigations of algorithms and projects that are not deployed at scale in actual practice. Typical articles in the literature will consider innovations in algorithms, with retrospective application to existing datasets; sometimes the latter are real-world datasets collected for the project, but often they are simple "library" datasets used for benchmarking algorithms. This is beginning to change, as machine learning tackles the challenges of inference in large-scale datasets, and the large-scale deployments that these ultimately entail.

Furthermore, it has become increasingly common for machine learning researchers to adopt the strategy that the algorithms they produce should be available to other researchers, so that their work might be disseminated, evaluated, and further improved. This is becoming increasingly common with the novelty of "full disclosure" journals, in which data and algorithms must be published (typically as appendices available on-line) alongside scientific articles. Machine learning algorithms often take the form of modules written in MATLAB or C / C++ / C#. The majority of these modules are written in an ad hoc manner, by small teams of researchers (often individuals), for use by other small teams or individuals.

B. Large-Scale Deployment

Such practices must change with the increase in scope of machine learning, as occurs in health informatics, in which the databases of entire hospitals and the surrounding healthcare infrastructure are linked, and inference performed on the results. If machine learning is to make a significant contribution to practice, its outputs must be robust and reliable. While much progress has been made, particularly in Bayesian inference, in ensuring that the algorithms are *statistically* robust, the quality of the software can be a significant factor, and is not an aspect usually considered by machine learning researchers. Progress has been made, for example, in the area of agent-based reasoning, whereby software failures in individual agents can be compensated by the decentralised decision-making abilities of the greater network of agents [8]–[10].

As projects begin to scale up to the demands of modern practice, it is inevitable that machine learning researchers need to work more closely with software engineers. In the development of systems for use in health informatics, the Food and Drug Administration (FDA) certifies systems for use in medical practice in the USA. This requires fully traceable, auditable procedures for software development of the kind developed by software engineers over the past few decades. Recent work on algorithms for the health monitoring of jet engines, deployed in the engines of the Eurofighter Typhoon, the Airbus A380, and the Boeing 787 "Dreamliner" [11], [12], require equally rigorous approval from bodies such as the Civil Aviation Authority (CAA) before they can be used in practice, and are therefore also dependent on the successful interaction between machine learning and software engineering.

C. Re-use of Machine Learning Algorithms.

As more modules for machine learning algorithms are produced, the prospect of re-using code becomes possible. While the majority of machine learning modules are standalone entities constructed by individuals or small teams, there are some platforms that have become more widely adopted, such as the *WinBUGS* framework for Bayesian inference using Markov chain Monte Carlo techniques [13] and Microsoft's *infer.net* [14]. These platforms offer APIs and suitably abstracted data types for the performance of inference in their own respective frameworks; users are encouraged to develop and share modules for these platforms, which the producers then make available for distribution. Sometimes, toolboxes gain sufficient popularity such that their data types become more widely accepted, allowing code re-use; a popular example is the NETLAB toolbox of algorithms [15].

It is an unfortunate consequence of the infancy of the state of machine learning software that code re-use is minimal, and that the majority of projects are re-implemented from scratch. There is scope, then, for a more formal approach to modular code abstraction, design, and specification, based on the methods developed within software engineering. While adoption of a standard specification of data types and functions for machine learning may be impossible, given the disparate, individualised nature of research, larger projects, such as those in health informatics and equivalent areas, could evolve standardised forms of their own. As these large-scale programmes get closer to maturity, and as they begin to interact with other programmes in the same field, across multiple research institutions, techniques from software engineering will play an important role in determining these forms.

D. Increasing the Robustness of Machine Learning Software

The robustness of machine learning software, and hence its capacity for re-use and implementation at scale, is strongly dependent on the individual machine learning researcher's training in software engineering. Typically, such researchers come from scientific disciplines where some formal software engineering teaching has occurred, but this varies greatly in both quality and quantity by discipline and by institution. Therefore, the quantity of testing and formal code design that will have occurred in the production of machine learning software varies in a similar manner.

However, the nature of machine learning code makes it particularly difficult to test using traditional software engineering methods. Many machine learning algorithms perform "on-line" inference; i.e., they construct and adapt models in real-time, often while the process of data acquisition is taking place. Such systems change their state, and sometimes even modify their own structure, in a complex manner that makes it difficult to test them *a priori*. Many of the input data are uncertain, in both quantity and quality, which imposes further difficulties for testing modules before release. We note in passing that this facility to change structure during execution, and its consequences for code testing, can make certification (by, for example, the FDA or CAA) a complex and challenging process. Techniques for code testing are therefore required that take this adaptive, uncertain nature into consideration.

E. Intelligent Storage of Metadata

Due to the ad hoc nature of machine learning software design, and the consequent reinvention of algorithms for new projects, it can be difficult to repeat experiments and verify results previously obtained by other researchers, or even those results gained from past projects within the same research group. The ability to repeat experiments is at the heart of the scientific process; as projects become larger in scale and complexity, existing methods employed by machine learning researchers will become ever more unsuitable in allowing the verification of past results.

The scientific approach is further hindered by the fact that many real-world datasets are hard to acquire, and often impossible to share. For example, the datasets acquired in health informatics are typically subject to strict patient confidentiality legislation, whereby access to the data is tightly controlled (even when the data are in anonymised form, where patient-confidential details have been replaced by anonymising identifiers). Patients in the UK must typically individually give consent for use of their data, and those researchers that can access it must be listed, with their access approved by medical ethics committees. The problem is similar when the data are commercially or militarily confidential, as in the case of datasets acquired for jet engine monitoring.

However, statistical models constructed from confidential data may often be shared (as in health informatics), as may be the results of analyses. There is therefore scope for intelligent storage of metadata and models derived from confidential datasets, alongside the algorithms that produced the data. Software engineering has much to offer machine learning in this large-scale linking of metadata to algorithms, which would enable previous experiments to be quickly reproduced, because the data and the original state of the algorithms are stored together. The sharing of these metadatasets would then be possible, allowing collaboration across multiple centres with researchers who do not have the ethical permissions required to access the underlying confidential data.

III. Machine Learning in Software Engineering

A. Coping with Incomplete Real-World Data

The large-scale datasets that are linked in health informatics are often acquired from a variety of sources, with a corresponding variety of robustness. This uncertainty in the data can make linkage difficult. One of the advantages of the Bayesian approach to machine learning is the capacity to cope with incomplete and noisy datasets in a principled manner. While fuzzy logical approaches to data manipulation have been used in many software engineering systems [16], [17], such methods have largely been superceded by probabilistic approaches [18]–[20], which remain an active topic of research. In the latter, the interface between machine learning and software engineering is explicit, allowing statistical techniques to be employed in linking, sorting, and ranking uncertain data within real-world databases.

If the fields within records are treated as random variables (being either continuous or discrete), then probability distributions can be defined over these fields within records. Missing fields may be "integrated out" of incomplete records, using the *marginalisation* property that is one of the hallmarks of Bayesian inference. This allows missing random variables to be treated in a principled manner, using the probabilistic identity of the form

$$p(x_1, x_2, x_4, \ldots, x_N) = \int p(x_1, x_2, x_3, x_4, \ldots, x_N) dx_3 \tag{1}$$

in which the joint probability distribution over the complete set of N random variables $x_1 \dots x_N$ is $p(x_1, \dots, x_N)$, and where we have removed x_3 by integrating over all of its possible values. The result is the joint probability distribution over all variables except x_3, where the removal of x_3 has taken into account all possible values of x_3. This *marginalised* distribution may then be used to compare the incomplete record with other records.

This offers an alternative to the standard approach used in, for example, conventional medical data manipulation, in which a large cohort of patients are initially identified, but a (typically much smaller) subset, consisting of those patients with complete data, is identified for actual use. The Bayesian approach allows us to use even those patients for whom we have limited information, in a principled approach based on the axioms of probability.

B. Predicting Missing Values

Bayesian inference allows us to determine the likely values of missing data. Following the above example, the distribution of the missing random variable x_3 may be estimated by finding the conditional distribution $p(x_3 \mid x_1, x_2, x_4, \dots, x_N)$; i.e., the distribution over our missing variable x_3 given the variables for which we have values, $x_1, x_2, x_4, \dots, x_N$. This conditional distribution may be obtained directly from the joint distribution over all variables that we have formed from our knowledge of the whole dataset:

$$p(x_3 \mid x_1, x_2, x_4, \dots, x_N) = \frac{p(x_1, \dots, x_N)}{p(x_1, x_2, x_4, \dots, x_N)} \qquad (2)$$

The result is a distribution over x_3, which defines the likely values of this missing variable. We could therefore select a point estimate of the value of x_3 as being the mode of the distribution. Exploiting the fact that we have the full distribution over x_3, we could then determine "error bars" around that value, indicating our belief in where we expect the value of x_3 to lie (to some degree of belief, such as $P = 0.99$).

This can be particularly useful in health informatics, in which clinicians can view incomplete data records, and where the missing data may have their values estimated (with appropriate error bars) conditioned on those data that are present in the record.

C. Visualisation

If each record has N fields, then a record can be considered to correspond to a point in an N-dimensional space. In health informatics, for example, N can be very large, such as $N = 500$. Conventional methods would find it difficult to plot data for a large set records of such high dimensionality, unless each field is plotted independently. This obscures any correlation that may exist between fields.

Machine learning offers principled techniques [21], [22] for exploring the structure of complex datasets. The aim is to map the original N-dimensional records down into a 2-dimensional form, so that they may be visualised. If every record corresponds to a point in the original N-dimensional space, then each record subsequently corresponds to a single projected point in the 2-dimensional visualisation map. Records that are similar (and are therefore close together in the original N-dimensional space) will appear close to one another in the 2-dimensional visualisation; conversely, records that are dissimilar (and are therefore far from one another in the N-dimensional space) will appear significantly separated in the 2-dimensional visualisation. This typically causes clusters of similar records to form in the visualisation, with "outliers" corresponding to highly dissimilar records.

This approach can be useful for exploring the structure of records in very high-dimensional datasets, comprising many hundreds of thousands of patient records, where each comprises thousands of fields.

D. Security

Software engineers often create layered levels of access (user, superuser, admin, etc.) for complex datasets, particularly when the access to those datasets is sensitive, as in health informatics. Access patterns must be tracked so that patients and clinicians can accurately determine who has viewed the data, and what, if any, actions have been taken after the access.

The number of such accesses will typically scale up with the system, so that large numbers of users may be interacting with a health informatics system within a single healthcare region. Machine learning offers the ability to model "normal" access patterns for each of the user groups, and automatically track and detect "abnormal" accesses that could be a result of patient or clinician misuse.

This approach in machine learning is termed "novelty detection" [23], and involves construction of statistical models of "normal" behaviour, such that deviations from the model may be identified as being "abnormal". Novelty detection is particularly suitable when the quantity of "normal" data is very large, and where there is insufficient "abnormal" data to be able to model the "abnormal" class. It is therefore often used in the analysis of data from critical systems, which function "normally" for the majority of their operation, and where examples of "abnormality" are rare.

Novelty detection has been used for determining abnormal spending patterns for the detection of fraud [24], or in accessing computer systems [25], which typically involves on-line learning of access patterns from a complex system, updating its behaviour in real-time, and communicating the results of "suspicious" activity to human experts for further investigation. This allows software and security engineers to focus their attention on cases of potentially unsafe access, helping them to make sense out of the large number of complex access patterns that take place.

IV. Conclusions

Machine learning and software engineering are becoming complementary disciplines as applications scale up to become (i) too large for the existing software practices of machine learning and (ii) too complex, noisy, and potentially inconsistent for the existing deterministic approaches of software engineering. These problems are particularly evident in applications involving the manipulation and analysis of very large datasets of different types and qualities, as occurs in health informatics and related fields. We have introduced ways in which the complementary disciplines of machine learning and software engineering have been contributing to improvements in "best practice", each of which is an active area for current research, with scope for considerable future contribution in each discipline.

Acknowledgements

This project was supported by the Centre of Excellence in Personalised Healthcare funded by the Wellcome Trust and EPSRC under grant number WT 088877/Z/09/Z, and by the NIHR Biomedical Research Centre Programme, Oxford.

References

[1] NHS Health Informatics Career Framework, http://www.hicf.org.uk, 2012.

[2] L. Tarassenko and D. Clifton, "Semiconductor wireless technology for chronic disease management," *Electronics Letters*, vol. S30, pp. 30–32, 2011.

[3] G. Clifford and D. Clifton, "Annual review: Wireless technology in disease state management and medicine," *Annual Review of Medicine*, vol. 63, pp. 479–492, 2012.

[4] D. Clifton, S. Hugueny, and L. Tarassenko, "Novelty detection with multivariate extreme value statistics," *Journal of Signal Processing Systems*, vol. 65, pp. 371–389, 2011.

[5] S. Hugueny, D. Clifton, and L. Tarassenko, "Probabilistic patient monitoring with multivariate, multimodal extreme value theory," *Communications in Computer Science*, vol. 127, pp. 199–211, 2011.

[6] D. Meredith, D. Clifton, P. Charlton, J. Brooks, C. Pugh, and L. Tarassenko, "Photoplethysmographic derivation of respiratory rate: A review of relevant respiratory and circulatory physiology," *Journal of Medical Engineering and Technology*, vol. 36, no. 1, pp. 60–66, 2012.

[7] Engineering and Physical Sciences Research Council, UK, http://gow.epsrc.ac.uk/NGBOViewGrant.aspx?GrantRef=EP/H019944/1, 2012.

[8] L. Kaelbling, M. Littman, and A. Cassandra, "Planning and acting in partially observable stochasitc domains," *Artificial Intelligence*, vol. 101, no. 1–2, pp. 99–134, 1998.

[9] S. Seuken and S. Zilberstein, "Formal models and algorithms for decentralized decision making under uncertainty," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 2, pp. 190–250, 2008.

[10] M. Kaufman and S. Roberts, "Coordination vs. information in multi-agent decision processes," in *Proceedings of Autonomous Agents and Multi-Agent Systems*, 2010.

[11] S. King, P. Bannister, D. Clifton, and L. Tarassenko, "Probabilistic approaches to condition monitoring of aerospace engines," *IMechE Part G: Journal of Aerospace Engineering*, vol. 223, no. G5, pp. 533–541, 2009.

[12] L. Tarassenko, D. Clifton, P. Bannister, S. King, and D. King, "Novelty detection," *Encyclopaedia of Structural Health Monitoring*, pp. 653–675, 2009.

[13] D. Lunn, A. Thomas, N. Best, and D. Spiegelhalter, "WinBUGS - a Bayesian modelling framework: Concepts, structure, and extensibility," *Statistics and Computing*, vol. 10, pp. 325–337, 2000.

[14] T. Minka, J. Winn, J. Guiver, and D. Knowles, "Infer.NET 2.4," 2010, Microsoft Research, Cambridge. http://research.microsoft.com/infernet.

[15] I. Nabney, *Netlab: Algorithms for Pattern Recognition*, 1st ed. London: Springer, 2002.

[16] G. Chen, *Fuzzy Logic in Data Modeling: Semantics, Constraints, and Database Design*. Berlin: Springer-Verlag, 1998.

[17] L. Zadeh, "Is there a need for fuzzy logic?" *Information Sciences*, vol. 178, no. 13, pp. 2751 – 2779, 2008.

[18] S. Prithviraj and A. Deshpande, "Representing and querying correlated tuples in probabilistic databases," in *Proceedings of the 23rd IEEE International Conference on Data Engineering*, 2007, pp. 596–605.

[19] C. Koch and D. Olteanu, "Conditioning probabilistic databases," *Proceedings of the Very Large Databases Endowment*, vol. 1, pp. 313–325, 2008.

[20] N. Dalvi, C. Ré, and D. Suciu, "Probabilistic databases: Diamonds in the dirt," *Communications of the Association for Computing Machinery*, vol. 52, pp. 86–94, 2009.

[21] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin: Springer-Verlag, 2006.

[22] N. Lawrence, "Gaussian process latent variable models for visualisation of high dimensional data," in *Advances in Neural Information Processing Systems 16*, 2003, pp. 329–336.

[23] D. Clifton, L. Clifton, P. Bannister, and L. Tarassenko, "Automated novelty detection in industrial systems," *Studies in Computational Intelligence*, vol. 116, pp. 269–296, 2008.

[24] R. J. Bolton and D. J. Hand, "Statistical fraud detection: A review," *Statistical Science*, vol. 17, no. 3, pp. 235–249, 2002.

[25] D. Yeung and Y. Ding, "Host-based intrusion detection using dynamic and static behavioral models," *Pattern Recognition*, vol. 36, no. 1, pp. 229–243, 2003.