Form 101, Part II: Research Proposal

Russell Greiner
Department of Computing Science University of Alberta Edmonton, Alberta T6G 2H1

1 Framework

Computers are renowned for their ability to do what they they are told. Unfortunately, sometimes we humans cannot tell them what to do, perhaps because the relevant information is not known (*e.g.*, the information required to play world-class backgammon, or to identify gene sites), or because we have difficulty expressing the relevant information (*e.g.*, safely driving a car), or because it does not make sense for a person to hand-code a program for the task (*e.g.*, to have programmers build a huge *set* of individual interfaces, each honed to a different specific user, or to continually re-program the control system for a plant each time the dynamics shift).

Fortunately, in many of these situations, it is possible to collect examples of "correct behavior" versus "incorrect behavior" — *e.g.*, examples of effective vs ineffective backgammon play, gene (non)boundaries, or (in)appropriate driving actions. Here, we may be able to replace the hand-coding chore with a *machine learning (ML) algorithm*, which can use these "training examples" to produce a correctly functioning program [11].

I plan to continue designing and building effective learning algorithms for a wide variety of tasks. Section 2 describes my on-going work that addresses the *foundations* of learning — providing new learning techniques that use the available information to produce effective performance systems.

Machine learning is not an end in itself; it is a tool used to improve the performance of some underlying system (*e.g.*, computer game, diagnostic system, web-browser, interface, . . .). As such, the field of ML advances by addressing the new challenges that come from considering new needs and tasks. Section 3 describes some "application pull" activities I plan to pursue.[1]

2 Theoretical Foundation (Technology Push)

Learning and using probabilistic models: Bayesian belief nets (BNs), which are compact encodings of joint probability distributions, are now used by many major companies and organizations (Microsoft, GE, US Army, . . . [1]), for a variety of tasks ranging from classification and diagnosis to control and prediction, including recent work in genomics and proteomics [13]. One challenge is acquiring these probabilistic models. My group (including A. Grove [Netli], D. Schuurmans [UofWaterloo], former postdoc J. Cheng and students, T. Van Allen and W. Zhou) are currently developing a suite of algorithms that learn BNs that give accurate answers, over the range of questions that are likely to be posed **[C13]**, **[C3]**, **[C11]**[2]. As these algorithms exploit this prior knowledge about queries, they can be more (sample) efficient than other approaches, even when they must learn about the query distribution as well. We plan a number of further improvements to these "discriminant learning" algorithms [12], to be yet more efficient, and also to deal with "decision nets", which extend BNs by including the capability of making appropriate decisions, rather than just computing posterior probabilities.

[1] Of course, many of the theoretical results will have fairly direct applications, and most of the applied activities will require theoretical analysis. This dichotomy is more to indicate the original motivation for these work described.

[2] Each **[Jx]**, **[Cx]** and **[Px]** reference refers to an entry in my personal bibliography included in the "Form 100, Part II (Contributions)" portion of this proposal. Those references also list my collaborators on these projects.

We also plan to continue investigating better ways to learn Belief Nets, in general — providing better selection criteria **[C6]** and learning algorithms **[J3]**.

We will also provide effective[3] ways to determine which types of additional instances will be most helpful in producing more effective belief-net based classifiers, using our existing work on computing the posterior distribution of a response from a belief net **[C1]**. (*E.g.*, we will be able to determine which type of instances will best shift the posterior distribution to one side of a decision boundary.) This will be especially useful when dealing with situations where data, while available, is difficult (*e.g.*, expensive) to obtain.

Techniques that help programs run faster: My PALO system **[J10][J9]** was designed to learn the optimal values for a program's parameters, for a wide range of programs, and various measures of "optimality" — *e.g.*, efficiency, accuracy, etc. P van Beek (UWaterloo) and I plan to use this system to improve the efficiency of various generic problems solvers (scheduling, planning, and constraint satisfaction algorithms), by identifying which heuristics work best for a class of specific problems. We will then use the resulting "optimized" system for a variety of real-world tasks.

Techniques that allow learners to scale up: Learning involves collecting a body of data that is sufficient to reliably evaluate competing performance systems. To scale up, learners will have to do well with less data: Here, I would like to extend my results (with D Schuurmans [UofWaterloo], A Grove [Netli], A Kogan [Rutgers]) to help learners work with fewer training instances **[C16]**; to be able to accomodate degraded training instances **[C15]**; and to exploit prior knowledge **[J7]**, **[J5]**, **[J8]**.

3 Practical Learners (Application Pull)

Producing programs that learn to interact better with users: Today there are a wide variety of *interactive* computer applications, ranging from web-browsers and searchers, through spreadsheets and database management systems, to editors, as well as games. While these systems may be very complicated, it is important that their human interface be simple, to accommodate the user. However, every user has his own idiosyncrasies and preferences; this means an interface that is honed for one user may be problematic for another. While most interfaces can be individually customized by hand (*e.g.*, by adjusting parameters, or defining macros), this is a tedious process that requires the user to be aware of his personal preferences. We are therefore building autonomous *adaptive* interfaces, that watch the user interact with the (performance) program, then use these observations to "learn" a better interface.

My students (B Korvemaker, C Thompson) and I designed and implemented a system **[C5]** that first predicts the user's next Unix command, then fills the buffer with that command, allowing the user to run that command by simply entering "return". We plan to extend this system to be more general and more accurate, and also to learn "macros" — *i.e.*, sequences of commands that frequently occur together. We hope to apply the resulting general ideas to other programs and operating systems.

One important class of programs are commercial electronic games. I plan to work with J Schaeffer and other members of his "Games Group" [7], as well as Electronic Arts ([4], the world's largest producer of computer games) on a tool to hone game interfaces: For example, if a player repeatedly uses the some sequence of commands (*e.g.*, "forward, turn, shoot, turn, back"), the adaptive system will first figure out why the player did this (here, to sneak around the corner and

[3]Throughout, we will use term "effective" in the decision theoretic sense (with respect to a cost function that may involve accuracy and efficiency), as opposed to the "computable" sense.

fire where his opponent is likely to be), then map this general "plan" to a single key. To do this effectively, our system will need to first recognize the human agent's plan, to know how to repeat those actions [8].

Other Game-Related Learning: We also plan to investigate other ways learning can enhance the play of an electronic game. One approach is "Trainable Agents". Here, the human player will first show his "computer agents" what s/he would do, over a range of situations and instructions (*e.g.*, "attack this specific opponent", or "fortify this specific position"). That agent will then generalize from these observations, to be able to act as the player would, in new situations. After the player has built up an army of "clone" agents, s/he can then deploy them during the game.

Another idea is building a "worth adversary": People do not want to lose to a program, nor do we enjoy winning by large scores. An ideal computer adversary is one that we can just *barely* beat, in a tight competition. (Think of scoring the winning run in last inning of the seventh game of the world series.) To do this, of course, our computer player must first understand the abilities of its human opponent, and then find ways to be just slightly inferior. Last summer we (B. Stafford and I) began to address this challenge, identifying reinforcement learning [15] as an appropriate tool here. We plan to continue this exploration.

We also anticipate being able to use this technology to address another problem plaguing the computer game industry: sweet spots — *i.e.*, situations where the human player can win too easily, by using some simple trick that was overlooked by the game designers. We will consider ways to use reinforcement learning techniques to first identify these weaknesses, and then provide reasonable patches.

Adaptive Web (Re)Configuration: We (several professors in both computing science and the business school, together with several of our students) have begun working on a system that observes where clients tend to go within a web-site and extracts the common usage patterns. We are now considering various ways to use this information to enable future users to reach their target webpages more efficiently, and thereby increase the effectiveness of the web-site. One approach is a global re-structuring of the web-site (by adding or removing links between pages, and perhaps copying some information to new pages), with the goal of minimizing the expected number of "hops" each user will require to reach his/her destination. Another approach is to leave the static structure unchanged, but to add a "suggestion" button to the various pages, which (when clicked) first classifies the user into some "community" (based on IPaddress, previous click history, time of day, etc.) then suggests one or more pages — the pages that other users in this community have wanted to go. The first step in any of these activities is determining accurately where users really want to go. (Notice this is different from the pages that similar users have gone before as many of those pages are simply intermediate points en route to some useful page. Moreover, the actual target pages need not be "hub pages", using the Clever notation [2].) As this will require additional feedback from users, we are currently beginning a study to collect this data from subjects, using a carefully engineered browser and a set of well-defined user tasks.

Automatic construction of effective interpretation systems: Many imaging systems seek a good interpretation of the scene presented — *i.e.*, a plausible (perhaps optimal) mapping from aspects of the scene to real-world objects. These systems are typically built by hand, by an imaging expert who assembles various "imaging operators" (which can range from low-level edge-detectors and region-growers through high-level token-combination–rules and expectation-driven object-detectors) into an interpretation system. We can view this system as a *policy*, which specifies when to apply which operator, with which parameters. We are currently investigating ways to automate the process of producing good policies: That is, given the costs of these operators and the distribution of possible images, we can determine both the expected cost and expected accuracy of any such policy. Our task is to find a maximally effective policy — typically one with sufficient accuracy, whose cost is minimal. Our earlier papers provided simple myopic systems that worked effectively in various real-world tasks, including a classical approach to recognizing car types **[C4]** and an eigenface-based approach to recognizing faces **[C2]**.

I plan to continue working with both R. Isukapalli and V. Bulitko to further scale up these ideas, to produce a system that can deal with larger tasks. First, our system will need to deal with more complex interactions between the operators — *e.g.*, because one operator requires, as input, the output of another operator (*e.g.*, a line-segmenter produces a set of tokens, which are then used by a line-grower) or because the actual data obtained from one operator may be critical in deciding which next operator+parameters to consider next: *e.g.*, finding that the fuselage is at some position may help determine where to look for the airplane's wings.

The earlier work also assumes we know initially the *distribution* of scenes we will have to interpret; a realistic system will also need to deal with the challenges of *learning* this information as well — while we can word this within our "active classifier" framework **[J2]**, we anticipate a real system will require significant extensions.

BioInformatics: I plan to continue working on a number of projects in the exciting area of computational biology — which involve using computational tools to address molecular biology challenges. One projects is the *Whole Proteome Analysis* (with D Szafron), which is attempting to determine the function for each protein within an organism's entire proteome. Our current system uses a simple naive-bayes model to learn the functionality of individual proteins, based on properties of their homologues. We plan to extend this system in several ways: learning a more general belief net structure (such as TAN [6], or even more general **[C3]**,**[C11]**), selecting the relevant subset of features [10], and using discriminant learning techniques **[C13]**. Our next sub-project — also for identifying the function of a protein — will also exploit properties of the *set* of proteins within a proteome; *e.g.*, using that fact that proteins produced by consecutive regions within a genome typically have similar functions, etc.

Another project (with B Zanke, Alberta Cancer Board) is trying to determine which therapy is most likely to be effective for each specific patient. There are many treatments for a specific disease (here breast cancer); and the one that is most effective for one patient may be unsuccessful, or even fatal, to another. This project is attempting to determine which patient features best predict whether some treatment will be effective, and then find some cost-effective tests for these features. We are now in the process of building a vast database that extensively characterizes the conditions for hundreds of patients — including urinanlysis, SNP data, and gene expression data from DNA microarray analysis — together with their subsequent response to various treatments. We will then mine this data to identify which type of test(s) to run on the patient, whose outcome will help us identify which treatment will work best for each specific patient.

Here we plan to use the "Probabilistic Relational Model" (PRM [14, 9], an extension to belief net models that allows relational information) to represent this heterogenous knowledge base; we anticipate needing to extend the emerging body of tools for *learning* these models.

A third project is CyberCell [3] (with M Ellison, Biochemistry) which is attempting to map all the structures of the approximately 4,000 *E. coli* proteins and then simulate them on a computer. In addition to helping us understand how this organism is self-managed, this will also enable us to predict the effects of new stimuli (*e.g.*, an antibiotic, or a temperature change, etc.) within the cell's environment.

In addition to the biological and biochemical issues, there are also huge computational challenges here. I plan to work (again) in the datamining area: *e.g.*, learning models describing the reactions and interactions, based on observations of the cell's behaviours. I anticipate needing a language as rich as PRMs here as well.

Finally, as evidence of our emerging stature in this area of bioinformatics, please note that some colleages and I are chairing the most-prestigious conference on Computational Biology, "Intelligent Systems for Molecular Biology" (ISMB) in 2002. (This is one of 8 major conferences in Edmonton this summer; see [5].)

4 Conclusions

While the list of activies appearing above may seem diverse, there are a number of commonalities: First, each of these tasks will lead (often immediately) to a clearly useful application. Second, each of these tasks inherently involves some interesting type of learning — *i.e.*, using some experience to improve performance. Finally, we have already produced results (technical publications and/or patents, typically accompanied by an implementation) for most of these tasks; and in every case have (at least) identified the type of learning tools that will be required. For these reasons, we are extremely confident that we will be able to obtain, in the next few years, results that are innovative, publishable, and practical.

References

[1] `http://www.cs.ualberta.ca/~greiner/bn.html`.

[2] Chakrabarti, Dom, Kumar, Raghavan, Rajagopalan, Tomkins, Kleinberg, and Gibson. Hypersearching the web. *Scientific American*, 280, 1999.

[3] `http://www.synthesystems.net/cybercell/html/.html`.

[4] `http://www.EA.com`.

[5] `http://www.cs.ualberta.ca/Edmonton2002`.

[6] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. *Machine Leaning*, 29:131–163, 1997.

[7] `http://www.cs.ualberta.ca/~games`.

[8] Henry Kautz. A formal theory of plan recognition and its implementation. In *Reasoning About Plans*, pages 69–126. Morgan Kaufmann, 1991.

[9] Daphne Koller. Probabilistic relational models. In *Inductive Logic Programming, 9th International Workshop (ILP-99)*, pages 3–13. Springer Verlag, 1999.

[10] Pat Langley and Stephanie Sage. Induction of selective bayesian classifiers. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 399–406, 1994.

[11] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[12] B. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, 1996.

[13] S. Salzberg, D. Searls, and S. Kasif. *Computational Methods in Molecular Biology*. Elsevier, 1998.

[14] E. Segal, B. Taskar, A. Gasch, N. Friedman, and D. Koller. Rich probabilistic models for gene expression. In *ISMB-01*, pages 243–252, 2001.

[15] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.