Feature Discovery in the Context of Bayes Nets: An Inductive Approach

Andrew Arnold, Richard Scheines and Joseph E. Beck
Center for Automated Learning & Discovery
Carnegie Mellon University
Pittsburgh, PA 15213
{aarnold, joseph.beck}@cs.cmu.edu, scheines@andrew.cmu.edu

Abstract

Feature induction is used to reduce the complexity of the model search space of a Bayes network. The Bayes net is used to model student behavior in an on-line course. Specifically, the frequency of student self-assessments is used to predict quiz performance. By moving most of the search from the model space to the feature space, prior knowledge and bias can be introduced and the search problem constrained.

1 Introduction

Bayesian networks are frequently used to model the interactions between variables [1]. They provide a useful representation of the independence assumptions of the model. These independencies are crucial in making inference and related applications tractable. One common problem is for a researcher to have some idea of which variables, or features, he thinks are involved in a process, but is unsure exactly how these features interact with one another. Specifically, he does not know where the independencies lay. Some features may not be relevant at all to the process, and some may only have a very small effect. This can be viewed as a model selection problem: deciding which variables to include in the model, and precisely describing the level of their interaction.

One popular method for performing model selection is to fix the variables or nodes of the network and to search over possible configurations of edges. This suffers from the fact that it is not always clear where these nodes come from in the first place. If a search algorithm is forced to consider a possibly spurious node, it might tweak the entire network just to accommodate a bad assumption.

In this work, we instead begin with a relatively small set of core features and grow this set, through a process of inductive feature search, until a satisfactory model is found. This benefits from the fact that unhelpful or deleterious initial node selections can be pruned out by the algorithm and the feature search guided towards more promising areas.

2 Problem

Many applications produce copious amounts of low-level log data. In particular, we are interested in examining the data produced by students using an on-line course. Intuitively, researchers believe many interesting, and potentially useful trends and patterns are contained in these logs. In fact, it has been verified that carefully hand-made features, based on log data, can be used to predict student performance [2]. Unfortunately, researchers have neither the time nor patience to go through all these logs, by hand, to find the useful trends. An obvious solution is to have this search done automatically, by a computer. Unfortunately, the unstructured, low-level nature of the data makes it very difficult to design an algorithm that can tie the data together into models of features and processes the researcher and his community are interested in and can understand. This is the problem this research tries to solve.

3 Model

The basic idea behind our approach is to search for complicated features, and put these into a simple model, rather than putting simple features into complicated models. The basic features, or atoms, that we start with are listed in Table 1. These are raw data fields that are collected natively by the logging software.

Table 1: Atomic features

NAME	DESCRIPTION
User_id	(Nominal) Unique user identifier
Module_id	(Nominal) Unique module identifier
Assesses	(Ordinal) Number of self-assessment quizzes taken by this user in this module
Quiz_score	(Ordinal) **(Dependent variable)** % of quiz questions answered correctly by this student in this module

For our experiment, we tried to learn features that would help predict Quiz_score. Since our search complexity was taking place in feature space, we could assume a simpler graphical model. We chose a two node Bayes network, which we implemented as linear regression. This network is shown in Figure 1.

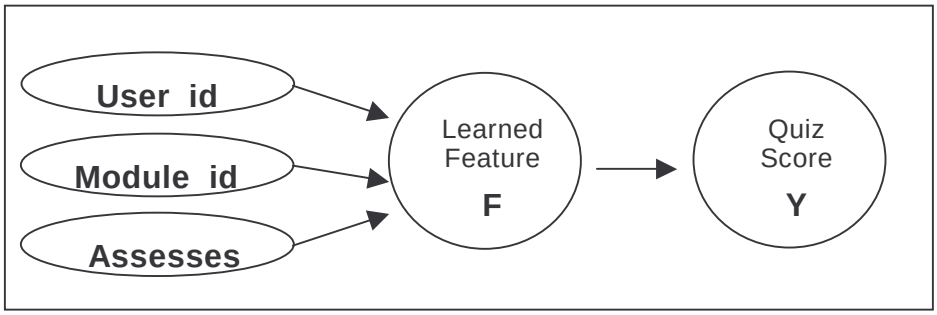

Figure 1: Feature interaction complexity is hidden from outcome variable.

As is shown in the figure, the complexity of the interaction between the four atomic features is hidden by the learned feature F. That is, given F, Y does not need to know about X1-X3. This allows for a simple two-node network between F and Y that models the process we are interested in.

4 Method

4.1 Predicates

We think of features as functions of the data. Raw data is a table of atomic features:

Table 2: Sample data

X1:User_id	X2:Module_id	X3:Assesses	Y:Quiz_score
Alice	module_1	12	86
Bob	module_1	14	74
Alice	module_2	18	92
Bob	module_2	13	87

This data can then be filtered into subsets through the process of *predication*. A predicate is a logical statement that is applied to each row of the raw data. It selects the subset of that data which satisfies this statement. For example, if we had the predicate: `User_id=Alice`, rows one and three would be selected. We could likewise filter on `User_id=Bob`. This would completely partition the data. These predicates can be formed exhaustively and automatically based on the definition of the atomic features.

4.2 Calculators

Once a predicate has been applied to the raw data, a function can be applied to the resulting filtered subset. We call these functions *calculators*. A calculator can perform various operations over the fields of the selected data. For instance, we might define a statistical calculator that returned the mean of the *Assesses* field. Or we could define a *count* calculator that simply returned the number of rows in the subset selected by the predicate. The specific definition of these calculators is guided by domain knowledge. For instance, educational research theory may say that timing-effects follow a log-scale decay. Thus, instead of a simple *subtraction* calculator, we may look at *log(difference)*.

Table 3: Calculators

NAME	DESCRIPTION
Mean	(Statistical) Calculates the mean over an ordinal feature
Sum	(Statistical) Calculates the sum over an ordinal feature
Max	(Statistical) Calculates the max over an ordinal feature
Min	(Statistical) Calculates the min over an ordinal feature

4.3 Features

Once we have applied our calculator to the filtered data, the result is a new feature. This feature is precisely, and entirely, defined by the predicate and calculator that produced it. For example, if we predicated on Module_id = 1, and Module_id = 2, and then applied the *mean* calculator, we would create a new feature: mean_assessments_per_module. In terms of the sample data given before, this feature, F: mean_assesses, would look like:

Table 4: Feature construction over sample data

X1:User_id	X2:Module_id	X3:Assesses	F:Mean_assess	Y:Quiz_Score
Alice	module_1	12	13	86
Bob	module_1	14	13	74
Alice	module_2	18	15.5	92
Bob	module_2	13	15.5	87

The feature can then be evaluated for fitness, based on the model being used, and then either be discarded, or incorporated into the data. This process continues iteratively, with high quality new features being appended to the data set after each round, and becoming atomic features in the next round from which to create ever more complicated features. Since all features are per module, the data can be combined.

5 Experiment

For the experiment we looked at whether we could discover complex predictive and interpretable features from raw data, given an appropriate model.

5.1 Data

We began with real data collected from students participating in the Open Learning Initiative [http://www.cmu.edu/oli/]. For each student, for each module, we collected the three atomic features *user_id, module_id, and assessments*. We also collected their *quiz_score* for that module.

5.2 Algorithm

The naïve approach to discovering new features would be to exhaustively split the data based on all instantiations of the predicates, and then apply all calculators to all the subsets created by those predicates. The problem with this solution, of course, is that its complexity is super-exponential in the number of features. Since we want to create an iterative method that can search deeply into the feature space, this will quickly blow-up and become intractable.

Our solution is two-fold: first, the atomic features are segmented into partitions. That is, again using a domain expert, all user-related features (such as the user's id, school, section, age, etc) are put into a logical bucket. Similarly for all course related features, module related features, etc. Then, the algorithm is applied greedily to each bucket, independently. The *k*-best features are returned from each bucket, and then they are incorporated as raw features into another iteration of the

algorithm. In this way, the algorithm is able to explore both broadly and deeply, but with enough bias so as not to become entrenched in intractability.

Second, after each iteration within each sub-step, each feature is evaluated in terms of its predictiveness and interpretability. R^2 is used to score predictiveness, and the depth of nesting of the feature, in addition to the specific predicates and calculators used, contribute to its interpretability score (*mean* is more interpretable than *max*, predicating on user_id is more interpretable than predicating on time_zone). After each iteration, the features are scored, and the *k* features with the best score are graduated to the next iteration, the rest are pruned away.

The definition of the score function is another critical lever in which we can incorporate and apply the theory already developed by the educational research community. This is vital to the mission of this work, as it has been shown that the results of automated learning methods tend not to be incorporated by the communities they serve, despite their statistically demonstrated predictiveness, if the community does not feel that its existing corpus of study has been used as a starting point for the work [3]. In other words, science is an iterative process, with the results of previous experiments informing not just the interpretation, but also the design and execution of subsequent studies. It would not make sense, therefore, for machine learning techniques always to be applied *de novo*. One of the crucial elements of this algorithm is the degree to which is allows for the leveraging and systematic inclusion of existing scientific knowledge.

6 Results

6.1 Results

Our experiment had two main goals: a machine learning goal of finding features which were predictive of student performance, and a scientific discovery goal of finding interpretable features which, taking into account existing scientific theory, also suggested new possibly scientifically justifiable features.

6.2 Machine Learning

For this part of the experiment, we began by randomly splitting the data into two subsets: 80% into training data, and 20% into testing data. We then applied our algorithm to the training data. This returned the three best complex features.

Table 5: Best discovered features

Mean assessments per user, over all modules
Total assessments per user, per module
Max assessments per user, over all modules

Each of these features was then used as the independent variable in a univariate generalized linear regression model, against the dependent variable *quiz*_score. Their fitness was evaluated using the R^2 metric (measuring the amount of deviance explained by each feature individually, relative to the null model). These features (defined as the predicate and calculator used to create them, not the actual numerical values as calculated over the training data) were then calculated over the held-out

testing data, and plugged into the linear models trained using the training data. Their cross-validation error was then recorded, and compared the error of the null model. The percentage decrease in cross validation root mean squared error was then reported.

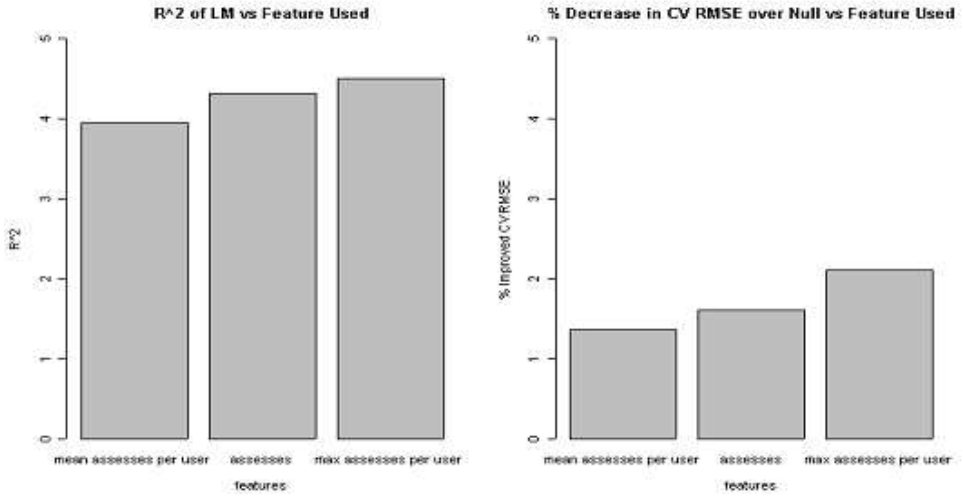

Figure 2. Evaluation of learned features.

The important thing to note here is that our algorithm actually did automatically recover features with some predictiveness. This is important because an algorithm that found scientifically interpretable, but predictively meaningless features would be of little value. In order for our technique to be useful, it must increase, at least minimally, our ability to explain performance, not just semantically, but also in a statistically significant way.

6.3 Scientific Discovery

For this part of the evaluation, we took a closer look at the semantic interpretation of the features discovered. For our algorithm to be valid, it should both reinforce our existing beliefs about which factors predict student performance, and also suggest new features that we have either had intuition about, but not been able to formulate precisely in terms of the raw data, or which we have never considered before. In this experiment, *mean_assessments_per_user_over_all_modules* is one of these kinds of features. Semantically, this feature could represent the average "introspectiveness" of a given user. That is, the more self-assessments a student takes, on average, compared to her classmates, could give an index into that student's propensity for evaluating herself. She might feel insecure in her mastery of the material. This would suggest a negative correlation with quiz_score, that is, the less mastery a student has, the more assessments she take, and the poorer her final quiz score. In fact, this is the opposite of what we find: a positive correlation.

Thus, presented with this evidence, an educational researcher might be forced to rethink his theory: perhaps those students who are most motivated to study, are also most motivated to evaluate their mastery. They keep reviewing the material until they perform well on the self-assessments, and only then proceed to the final quiz, on which they also do well. Another plausible hypothesis is that taking self-assessments actually helps students master the material, which leads to better quiz performance. It is important to note that these features and their correlations to

performance only suggest possible predictive relationships, not causal links. They quantify the definition of semantic features in terms of the raw data, which is a critical prerequisite for the design and implementation of further experiments by the researcher to fully investigate these proposed models, including distinguishing causation and correlation.

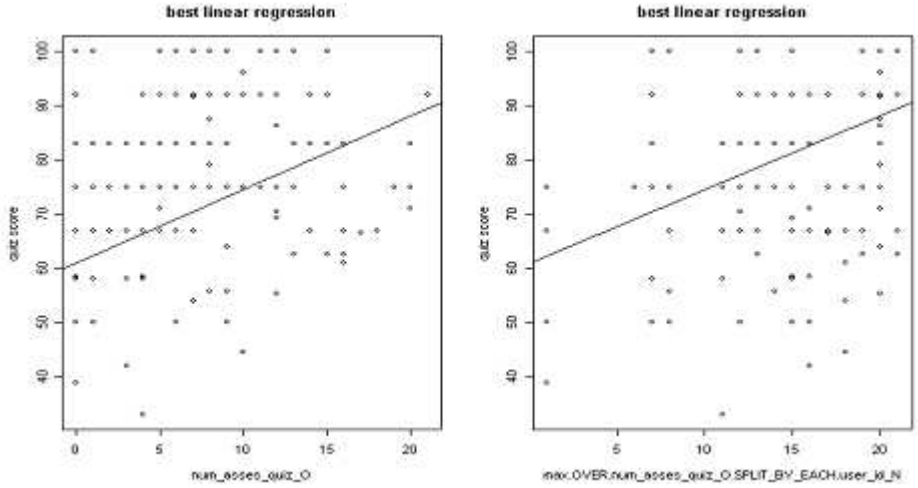

Figure 3. Positive correlation between self-assessments and quiz performance.

7 Conclusions

The main goal of this project was to automatically discover useful, complex features in the context of a Bayes net model. These features would at once elucidate the underlying structure of the raw data to the researcher, while at the same time hiding the complexity of this atomic structure from the network so that the features could be fed into even more complicated models without introducing intractable complexity. This goal was achieved.

In addition, by finding more complicated features that are still based on intuitive atoms, we produce models that are descriptive, but still interpretable and understandable. Thus we work not only towards models with better performance, but also towards the perhaps more important goal of furthering scientists' understanding of the features and relationships underlying the processes they are investigating [4].

We also found that finding novel, useful features is a difficult task. We used the competing biases of predictiveness and interpretability to guide our search through the feature space, while staying keenly aware of the trade-off between these two goals. Namely, predictiveness is useful, but often times not readily semantically or scientifically paresable. And interpretability, while more likely to be incorporated by scientists into theory, if not predictive, may actually move the state of the art backwards. A key result of this work was finding a way to incorporate and balance these competing goals.

8 Future Work

As we move towards data with possibly more complicated underlying model structure, we may need to incorporate more atomic features and more iterations of feature formation. Since the running time of the algorithm is super-exponential in the number of features, this raises the very immediate specter of intractability. To this end, we will need to develop methods for guiding and limiting the exploration of the predicate and calculator space intelligently, rather than relying on an exhaustive enumeration approach.

8.1 Better & Faster Search

Now that we have established that relatively shallow features can be discovered, we need to demonstrate that more complicated features can be discovered from real data. Specifically, we are interested in both searching more widely within each partition of features (that is, increasing the breadth of the partition) and more deeply (that is, running more iterations of nesting features within each other). Obviously, each small increase in either of these dimensions greatly increases the number of calculations needing to be performed. One mitigation of this increase would be the construction of decomposable feature scores, that is, scores that do not have to be computed *de novo* for each feature, but instead could be composed of other, already calculated feature scores. This would allow us to incorporate more features into the search, while only incurring an incremental increase in running time.

8.2 More Interpretable Features

Along these same lines, more intelligent partitioning of the feature space could reduce complexity while also increasing the quality of the features returned. Since this partitioning is one of the key biases we have into the search space, it is important to explore different ways of dividing features so as to minimize search complexity while maximizing the predictive and descriptive power of our features.

Finally, the interpretability metric used in evaluating features could be further refined to better reflect prior beliefs. That is, some features may gain or lose interpretability when combined with others (e.g. day of week and time of day: doing homework at midnight on Monday is very different from midnight on Friday). This is yet another lever that could be used to guide our search.

References

[1] Friedman N., Geiger, D., and Goldszmidt, M. (1997) *Bayesian network classifiers*. Machine Learning, 29:131--163.

[2] Arnold, A., Scheines, R., Beck, J., and Jerome, B. (2005). *Time and Attention: Students and Tasks*. AAAI-05: Educational Data Mining, Technical Report WS-05-02. AAAI Press.

[3] M. J. Pazzani, S. Mani, W. R. Shankle (2001). *Acceptance of Rules Generated by Machine Learning among Medical Experts*. Methods of Information in Medicine; 40: 380-385.

[4] Schwabacher, M., and Langley, P. (2001) *Discovering communicable scientific knowledge from spatio-temporal data*. ICML.