Feature Discovery in the Context of Educational Data Mining: An Inductive Approach

Andrew Arnold, Joseph E. Beck, Richard Scheines

Machine Learning Department
Carnegie Mellon University
5000 Forbes Avenue, Pittsburgh, PA 15213, USA
{aarnold@cs, joseph.beck@cs, scheines@andrew}.cmu.edu

Abstract

Automated learning environments collect large amounts of information on the activities of their students. Unfortunately, analyzing and interpreting these data manually can be tedious and requires substantial training and skill. Although automatic techniques do exist for mining data, the results are often hard to interpret or incorporate into existing scientific theories of learning and education. We therefore present a model for performing automatic scientific discovery in the context of human learning and education. We demonstrate, using empirical results relating the frequency of student self-assessments to quiz performance, that our framework and techniques yield results better than those available using human-crafted features.

Introduction and Previous Work

One of the fundamental goals of scientific research is the modeling of phenomena. 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. Researchers usually begin with some general idea of the phenomenon they would like to understand, and then proceed to collect some observations of it. The scientist, for example, might have some prior belief, based on existing scientific theory and intuition, that the amount of time a student spends reading course notes will affect his performance on quizzes, but is not able to specify exactly what he means by "time reading notes." Is it the cumulative number of minutes, split into any number of sessions, conducted under any condition, prior to the evaluation that matters? Or is the intensity of the reading more important? Is it better to read the notes right before the quiz, for higher recall, or perhaps an earlier viewing helps prime the student for learning? Unfortunately, researchers have neither the time nor patience to go through all these logs, by hand, to find ideal instantiations of their features. In this work, we develop a partial solution to this problem of feature discovery that uses a computer to intelligently induce higher-level features from low-level data.

Although computers can produce copious log data, the unstructured, low-level nature of these data unfortunately makes it difficult to design an algorithm that can construct features and models the researcher and his community are interested in and can understand. In fact, the complexity of a full search of the feature space, from a statistical point of view, would depend on the size of the sufficient statistics of the entire data set. Thus, for all real-world problems, brute force search is intractable.

A more insidious problem, however, is that even if the space of features were able to be enumerated and searched efficiently, the number of possible models based on those features would be even larger, and any attempt at learning a true model would suffer from overfitting and the curse of dimensionality. Although techniques do exist for addressing this issue, many do not take into consideration the semantics of the features, instead relying on an estimation of complexity. It turns out that by carefully limiting the types of features that we can represent and search, we reduce our search and overfitting problems without, hopefully, cutting out too many expressive features.

Certain techniques do exist for addressing feature selection. Principle component analysis (PCA) (e.g. Schölkopf, Smola, and Müller 1998), for example, finds a projection of the data from a higher dimensional space to one with fewer dimensions. This projection reduces the number of features needed to represent the data. Unfortunately, these projections distort the original, presumably intuitive, definition of the features of the data into linear combinations of these features. In this process, much of the interpretability of the resulting models is sacrificed. This weakness is present in many of the other methods used for feature selection and dimensionality reduction, such as clustering and kernel methods (Jain, Duin, and Mao 2000). All suffer from a sacrifice of interpretability which has been shown to be essential if computational techniques should ever have a serious impact on the progress of scientific research (Pazzini, Mani, and Shankle 2001). These are the problems this research tries to solve.

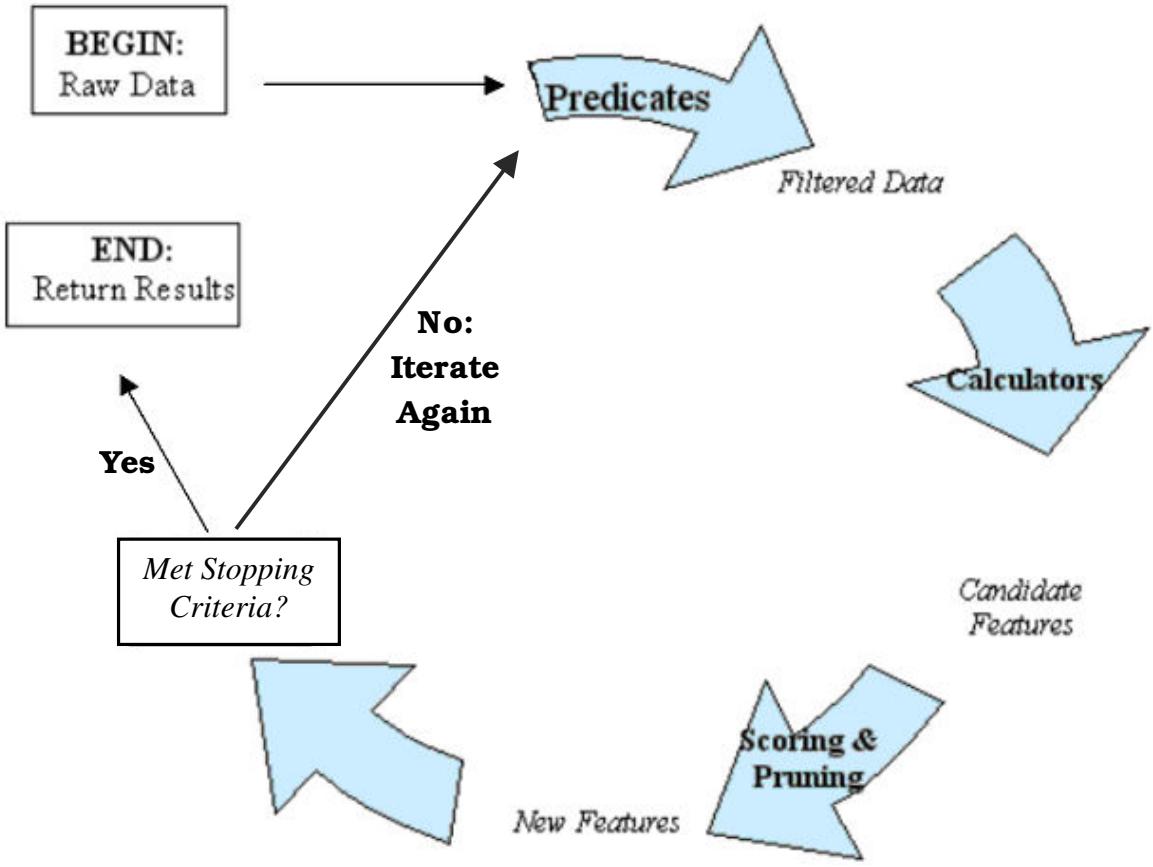

Figure 1. Feature creation flow diagram

It is important to note that we face two distinct problems, each of which is challenging in its own right. The first is defining and searching through the large space of possible features. The second is constraining that feature space and biasing the search to discover new features that improve predictiveness while still preserving the semantics of the original features. To achieve these goals we begin with a relatively small set of core features, defined in terms of the raw data, and grow this set, through an iterative process of feature creation, scoring, and pruning. At each iteration the predictiveness of the model based on the features is increased, while the scientific and semantic interpretability of the features themselves is hopefully preserved. Via this semi-greedy process of growing and pruning we are able to discover novel, non-intuitive features without the burden of a brute-force search.

System architecture (overview)

Figure 1 shows a high level diagram of the cyclical flow of data through the feature creation process and provides the outline for the structure of the paper. The arrows represent processes, and the items in italics are the inputs and results of those processes. We begin with the initial features of the raw data, and iteratively grow candidate features via prediction and calculation, and then prune these candidates down to create a new generation to begin the process again. Each completed cycle represents one iteration of the algorithm. The process continues until a user-defined stopping condition is met, e.g. elapsed computation time, R^2 of discovered model, or number of features discovered.

Models

The basic idea behind our approach is to search for complicated features and put them into a simple model, rather than putting simple features into complicated models. Since our search complexity is taking place in feature space, we use the simple linear and logistic regression frameworks to model the interaction between these features and our outcome variables.

Experiment

To demonstrate our approach, we provide a case study. We looked at whether we could discover complex predictive and interpretable features from raw data, given an appropriate model. Specifically, we tried to learn features that would help predict the student's quiz score.

Table 1. Atomic features

NAME	DESCRIPTION
User_id	(Nominal) Unique user identifier
Module_id	(Nominal) Unique module identifier
Assess_quiz	(Ordinal) Number of self-assessment quizzes taken by this user in this module
Assess_quest	(Ordinal) Number of self-assessment questions taken by this user in this module. Each self-assessment quiz contains multiple self-assement questions.
Quiz_score	(Ordinal) **(Dependent variable)** % of quiz questions answered correctly by this student in this module. In each module, students were given the chance to take the quiz up to three times. The max of these trials was taken to be *quiz_score.*

Raw Data

We began with real data collected from students participating in the Open Learning Initiative [http://www.cmu.edu/oli/]. The raw data from the logging software, which records events on a second by second scale, is then aggregated into a table of atomic module-level features. These features are summarized in Table 1 and a sample is shown in Table 2. For the first iteration, the raw data itself is taken as the initial set of features.

Table 2. Sample raw data

User_id	Module_id	Assess quiz	Assess quest	Quiz score
Alice	module_1	12	27	86
Bob	module_1	14	31	74
Alice	module_2	18	35	92
Bob	module_2	13	25	87

Predicates

A predicate is a logical statement that is applied to each row of the raw data and selects the subset of that data which satisfies it. The process of filtering the raw data into subsets in this way is called *predication*. The idea is that these subsets would form the basis of each new feature. We group certain data points together into clusters, based on their attributes, and calculate features over these subsets. 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`. By thus predicating on each unique value of a field (in this example `Alice` and `Bob` are the only unique values of the field `User_id`), we can completely partition the data easily and automatically.

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*. These calculators are a way of distilling our instance space (a subset of training examples) into their constituent summary statistics. This compression of the space allows us to search it more efficiently. A calculator can perform various operations over the fields of the selected data. For instance, we might define a calculator that returned the mean of the *Assess_quiz* field. Or we could define a *count* calculator that simply returned the number of rows in the subset selected by the predicate. A calculator can be defined to be applicable to any field, a certain field, or a class of fields. 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)*. The strength of our technique is that it searches through this space of calculators automatically and suggests those that seem the best. Table 3 presents a summary of the calculators used in our experiments.

Table 3. Calculators used in our study

NAME	DESCRIPTION
Mean	Calculates the mean over any ordinal feature
Sum	Calculates the sum over any ordinal feature
Max	Calculates the max over any ordinal feature
Min	Calculates the min over any ordinal feature

Candidate 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. Given this formulation of the feature space as a series of predications and calculations, we have a large set of features that we can express. For example, if we partitioned the data by predicating on *Module_id* = 1, and *Module_id* = 2, and then applied the *mean* calculator to the *assess_quiz* field, we would create a new feature describing the mean number of assessment quizzes taken within the module. In terms of the sample data given before, this feature, F: *mean_assess_quiz*, would look like Table 4.

Table 4. Feature construction over sample data

X1: User_id	X2: Module_id	X3: Assess quiz	F: Mean Assess Quiz	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 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. It is important to note that, although the space of representable features is large, it is by no means complete. For instance, our system of predicates and calculators could not express the feature "mean assessment quizzes for students whose name starts with an 'A'" or even the simpler feature "mean of assessment quizzes which were less than 15." By avoiding such freer, even arbitrary, feature construction we hope to bias the search space towards useful and interpretable results.

Scoring & Pruning

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, as mentioned before, is that because of the joint nature of the search, its complexity is very large in the number of features. Since we want to create an iterative method that can search deeply into the feature space, this approach will quickly blow-up and become intractable.

Our solution is two-fold: first, the atomic features are segmented into semantically similar groups of features. For example, 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. This segmentation not only avoids the joint search problem but also prevents the creation of less interpretable and more unlikely features like "mean assess quizzes for math modules on Fridays." It may also, of course, exclude other more interpretable and likely features. These buckets reduce the search space by limiting certain types of interactions. It is up to the user to decide how he wants to bias the space to balance the trade-off between a tractable search and an expressive feature space.

After this step the *b*-best features are returned from each bucket, and then they are incorporated as candidate features into the next step 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 mired in intractability. In practice, *b* is a parameter that can be tuned to balance this tradeoff and can have different values for different bucket types.

Additionally, since we are dealing with a multivariate feature selection problem, we also consider correlation between features. We use Fast Correlation-Based Filtering to heuristically find the subset of features that minimize the correlation between features while maximizing the R^2 of the resulting joint multivariate model (Yu and Liu 2003). Decoupling correlated features is particularly important since one of the strengths of our automated approach is the ability to examine many similar, but distinct feature formulations. This similarity creates many highly correlated features that, when combined, can lead to brittle, unstable features and models if not properly addressed. In addition, removing correlated features significantly reduces the size of our search space without discarding too much information.

Second, features are graded on predictiveness and interpretability. We use R^2 to measure predictiveness. However, there is no standard technique to measure interpretability. Therefore we use heuristics such as measuring the depth of nesting of the feature, in addition to the specific predicates, calculators, and component features used in the new features creation. We also look at the length of a feature's name was, with longer names implying less interpretability. These are all combined to calculate an interpretability score. After each iteration, the features are scored in this manner, and the *k* features with the best score are graduated to the next iteration, while the rest are pruned away. Selecting a value for *k* depends on a number of factors including the desired running time of the algorithm, the memory available and the type of features being searched for. Given a certain allotted running time, smaller *k* will produce deeper searches, while larger *k* will force shallower ones.

We should point out that the specific terms and criteria used in formulating the interpretability score are another critical area in which we can incorporate and apply the theory already developed by the educational research community. For example, one can develop a semantic taxonomy that prescribes the relative interpretability of different combinations of features, based on existing literature or explicitly stated scientific hypotheses. This taxonomy can then be applied to the initial features. Similarly, but perhaps less expensively, active learning could be used to propose a feature to a user who would provide feedback as to the feature's interpretability. Leveraging such existing knowledge 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 (Pazzini, Mani, and Shankle 2001). 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 it allows for the leveraging and systematic inclusion of existing scientific knowledge, as we have seen in terms of defining domain specific predicates and calculators, along with equally tailored measures of predictiveness and interpretability.

Iteration & Stopping Criteria

After the best features have been selected, and the rest pruned away, we assess the overall fitness of the model. If it satisfies user-set criteria (e.g. cross validation performance, or a hard cap on the number of iterations or processor time) the process stops and returns the current set of discovered features. If the stopping conditions are not met, the process continues again, with the current set of features becoming the initial set of features for the next iteration of feature exploration.

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 took into account existing knowledge and bias. For each of 24 students, for each of 15 modules, we collected the three atomic features *user_id*, *assess_quiz*, and *assess_quest*. We also collected the student's *quiz_score* for that module as an outcome variable. Since not every student completed every module, we were left with 203 data points.

Machine Learning

For this part of the experiment, we began by randomly splitting the data into two subsets: 80% of users into training data, and 20% of users into testing data. We decided it was important to split the data by user since observations of the same user are highly correlated and non-independent. We then applied one iteration of our algorithm to the training data, as described in Table 1, which returned a set of $k = 7$ features deemed the most predictive and interpretable. Since we only looked at one type of data (module level) there was no need to set a b parameter. The algorithm took about 30 minutes to finish on a P4 3 GHz machine with one gigabyte of memory. Our algorithm then trained a linear regression model with these seven features to predict *quiz_score*. The results of this training process were two: first, the features' definitions, which include the name of the predicate used to subset the data, and the calculator used to score the data, thus producing a feature. The second product was the parameters of the linear model trained on these features of the training data. The learned feature definitions were then instantiated over the held-out testing data, and plugged into a linear model using the trained parameters. We also considered a baseline model, which we defined as a linear model trained using the initial seed features of the raw data, before any feature creation was performed. This was to give us a comparison of the improvement gained by using our technique. We then recorded and compared the cross-validation R^2 of both models. The results are summarized in Table 5.

The important thing to note here is that these features were discovered automatically. Our algorithm discovered features that, together, explained over 38% more of the variance of unseen data relative to what was possible with just the initial, raw features. This improvement 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 our ability to explain performance, not just semantically, but also quantitatively, in this case by increasing the variance explained.

Table 5. Summary of features and their linear regression coefficients for both the baseline and discovered model. Cross validation R^2 is also included.

Model	Cross Validation R^2	Features in model	Beta
Baseline	7.46	*cumulative_assess_quests*	-0.08
		cumulative_assess_quizzes	0.92
Discovered	10.35	*cumulative_assess_quests*	-2.06
		cumulative_assess_quizzes	3.73
		mean_assess_quests_per_user	-2.82
		mean_assess_quizzes_per_user	4.55
		max_assess_quests_per_user	-1.22
		max_assess_quizzes_per_user	-3.06
		min_assess_quests_per_user	2.59

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. Details matter. Although the high level idea of measuring frequencies of self-assessment questions and quizzes is common across the features examined, the key difficulty is how to extract the signal from the noise.

In this experiment, *mean_assess_quizzes_per_user* is such a specific definition of an intuitive feature. Semantically, this feature could represent the average "introspectiveness" of a given user. That is, the number of self-assessment quizzes a student takes, on average, compared to his classmates, could give an index into that student's propensity for evaluating himself. He might take more assessments if he feels insecure in his mastery of the material. This model would suggest a negative correlation with *quiz_score*, that is, the less mastery a student has, the more assessments he takes, and the poorer his final quiz score. We might further reason that *assess_quests* would have the same type of effect, in this case, negative. In fact, this is the opposite of what we find: in our learned model, the beta of *mean_assess_quizzes_per_user* has the opposite sign of the beta for *mean_assess_quests_per_user*. Although we cannot be sure of the interpretation of these signs since it is a multivariate model, the fact that they are opposite is significant. Could it be that this is where our intuition failed us, leading us to conflate the two, distinct phenomena: a student choosing to take an assessment quiz, and then deciding how many of those quiz questions to answer?

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-assessment quizzes actually helps students master the material, which leads to better final 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.

Generality

These results are not limited to the specific data or problem presented in this paper. We have applied our same framework and algorithm to an entirely separate data source and problem, and produced equally significant results (discovering the degree to which tutor interventions affect the reading comprehension performance of students, and how this effect varies over time). The fact that we did not have to substantially change the mechanics of the algorithm, or our methods for creating and evaluating features, shows not only the generality of our approach, but also its utility, as it was able to produce useful results in a domain for which it was not intended.

Future Work

As we move towards data with possibly more complicated underlying structure, we may need to incorporate more atomic features and more iterations of feature formation. Since the running time of our algorithm is very large in the number of features, this expansion raises the very immediate specter of intractability. To this end, we will need to further develop methods for guiding and limiting the exploration of the predicate and calculator space.

Limitations

In this work we have only looked at a relatively small (four) number of initial features and calculators. This investigation was adequate to demonstrate our algorithm's theoretical utility. In practice, however, people are interested in data sets with many more initial features. The techniques we have used to limit the feature space, such as semantic feature segmentation, will need to be expanded to deal with the increase in search space associated with more features and calculators. Similarly, as the number of data points increases, so too does the number of predicates that can be formed, further increasing the number of candidate features to be constructed and evaluated. As the algorithm's search space increases, new search strategies will need to be developed. A greedy search based on a relatively simple score, such as the one presented here, may not be sufficiently powerful to distinguish the few choice features from the increasingly numerous distractors. Although we have achieved promising results in the investigation presented here, as well as in the preliminary examination of a separate data set, more studies will be needed to establish the generality of our approach.

Better & Faster Search

Now that we have established that predictive features requiring relatively few iterations of generation and pruning can be discovered, we need to demonstrate that more complicated features can be discovered from real data. Specifically, we are interested in searching both more widely within each partition of features (that is, increasing the breadth of candidate features considered by pruning fewer features in each iteration) and more deeply (that is, running more iterations). 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 simplification would allow us to incorporate more features into the search, while only incurring an incremental increase in running time. Another idea is to develop a stronger theoretical foundation for the definition of our feature space. A smoother parameterization, along with more theoretical insight, could allow for the application of more efficient search techniques such as gradient ascent. This would also allow the incorporation of data with more initial attributes, more data points, and the incorporation of more calculators.

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.

Conclusions

The main goal of this work was to automatically discover useful, complex features in the context of educational data mining. 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 model so that the features could be fed into even simple, robust models without introducing intractable complexity. This goal was achieved.

In addition, by finding more complicated features that are still based on intuitive raw features, 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 (Schwabacher and Langley 2001).

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 conductive to semantic or scientific parseability. 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. Specifically, interpretability was enforced by carefully partitioning the feature space in terms of the semantics of the initial features before the search began, and predictiveness was preserved by incorporating R^2 into the score used in pruning the candidate features.

References

Jain, A., Duin, R., and Mao, J. (2000). Statistical Pattern Recognition: A Review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):4--37.

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

Schölkopf, B., Smola, A.J., Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299-1319.

Schwabacher, M., and Langley, P. (2001). Discovering Communicable Scientific Knowledge from Spatio-Temporal Data. *In Proceedings of The Eighteenth International Conference on Machine Leaning*, 489--496.

Sirovitch, L. and Kirby, M. (1987). Low-Dimensional Procedure for the Characterization of Human Faces. *Journal of the Optical Society of America*, 2:519--524.

Yu, L. and Liu, H. (2003). Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution. *In Proceedings of The Twentieth International Conference on Machine Leaning*, 856--863.