DATA-DRIVEN CONSTRUCTIVE INDUCTION: A Methodology and Its Applications

Eric Bloedorn
The MITRE Corporation and George Mason University
1820 Dolley Madison Blvd., W640
McLean, VA 22102
USA
bloedorn@mitre.org

George Mason University
4400 University Drive
Fairfax, VA 22030
USA

Ryszard S. Michalski and
George Mason University
4400 University Drive
Fairfax, VA 22030
USA
michalski@gmu.edu

Institute of Computer Science
Polish Academy of Sciences
Warsaw, ul. Ordona 21
Poland

Abstract

The presented methodology concerns constructive induction, viewed generally as a process combining two intertwined searches: first for the "best" representation space, and second for the "best" hypothesis in that space. The first search employs a range of operators for improving the initial representation space, such as operators for generating new attributes, selecting best attributes among the given ones, and for abstracting attributes. In the methodology presented, these operators are chosen on the basis of the analysis of training data, hence the term *data-driven*. The second search employs an AQ-type rule learning to the examples projected at each iteration to the newly modified representation space. The aim of the search is to determine a generalized description of examples that optimizes a task-oriented multicriterion evaluation function. The two searches are intertwined, as they are executed in a loop in which one feeds into another. Experimental applications of the methodology to text categorization and natural scene interpretation demonstrate a significant practical utility of the proposed methodology.

Keywords: machine learning, inductive learning, constructive induction, attribute generation, attribute selection, attribute abstraction, discretization.

Introduction

Inductive learning algorithms are increasingly being used for data mining and knowledge discovery because they can provide powerful tools for determining useful and comprehensible patterns in large volumes of data. A major limitation of all conventional inductive learning algorithms is that descriptions they build (e.g, decision trees, decision rules, Bayesian nets, etc.) employ only terms (attributes) that are selected from among those explicitly provided in the original data. For that reason, such algorithms have been called *selective*.

Due to the above limitation, the responsibility of determining attributes relevant to the problem at hand falls entirely on the data analyst. This task may be quite difficult in practice. Yet, it is crucial for the success for the learning process. If attributes provided in the training examples are insufficiently adequate for the problem, then descriptions created by a (selective) learning system will likely be excessively complex, and their accuracy will be low regardless of the learning method used. For example, if one measures age in terms of days rather than years and gives the driving test score as a value from 0% to 100% rather than as a boolean pass/fail, it will be difficult for a learning system to generate rules which accurately predict if a person is a legal driver.

In general, attributes provided in the data may be inadequate for the learning task when they are only weakly relevant (weakly statistically correlated), indirectly relevant (relevant only as arguments of some function that combines them with other attributes), conditionally relevant (relevant only for some cases or for selected decision classes, or relevant only under some conditions), or when they are inappropriately measured (with too low or too high precision).

To cope with inadequacy of attributes provided in the original data, the idea of constructive induction has been proposed [1]. The original formulation of the idea was concerned primarily with generating additional, more task relevant attributes from the originally given, in order to improve the learning process. It was subsequently observed that attributes used in the training data define the *problem representation space*, and a learning algorithm searches for boundaries delineating individual concepts or classes in this space. Adding more relevant attributes, removing irrelevant ones, or modifying the measurment precision of attributes are diffrent forms of the improvement of the representation space, which can be applied individually or jointly.

Therefore, a general view of constructive induction has been proposed, in which it is a process of learning concepts desciptions that employs two intertwined searches: one for the “best” representation space, and the second for the “best” hypothesis in the space [3] When searching for the best representation space, the system may make no commitment as to the description language used for creating a hypothesis, or may be dependent on the description language. The second search determines a hypothesis that combines attributes (spanning the representation space) according to the assumed description language. Therefore, what constitutes the “best” representation space is, in principle, dependent on the description language used. The search for the best space and for the best hypothesis in the space are thus interrelated.

In general, the more expressive is the description language used by a learning algorithm, the lesser is the need for improving the original representation space (assuming that the original attributes are sufficiently adequate). On the other hand, the higher is the expressive power (capacity) of the description language, the exponenially higher is the search for the best hypothesis in that language. By separating searches for the best space and for the best hypothesis, constructive induction simplifies the overall learning process in the case of a weak representation space, and makes possible to generate an accurate hypothesis even with a relatively simple learning algorithm..

Methods of constructive induction can be classified on the basis of the source of information that is used for searching for the “best” representation space. In *data-driven* constructive induction (DCI), the search is based on the analysis of the input examples (data); in *hypothesis-driven constructive induction* (HCI) the search is based on the analysis of intermediate hypotheses); and in *knowledge-driven constructive induction* (KCI) the search exploits domain knowledge provided by the expert [3]. There is also a *multistrategy constructive induction* (MCI), that employs two or more previous methods [8].

The above general view of constructive induction and the understanding of different information sources for its implementation has been in the last several years used to guide the development of such constructive induction programs as AQ17-DCI, AQ17-HCI and AQ17-MCI. All these programs use an AQ-type rule learning algorithm for conducting search for hypothesis, hence the “AQ” prefix.

This paper describes the latest methodology for the data-driven constructive induction that has been implemented in the current version of the AQ17-DCI program, and presents new results from its application to some practical problems. Specifically, the latest methodology combines the AQ-15c learning algorithm with a much wider range of data-driven representation space improvement operators. These operators are classified into *constructors* and *destructors.* Constructors extend the representation space, and destructors reduce the space. Constructors are based on methods of attribute generation (“feature construction”[1]), and destructors are based on methods for attribute selection (“feature selection”) and attribute abstraction. All these operators, which are usually considered in the literature separately, have been integrated in AQ17-DCI in a synergistic fashion. Experimental applications of the methodology to text categorization and natural scene interpretation have confirmed the benefits of improving the initial representation space, and demonstrated the advantages of constructive induction over convential induction in cases where the initial representation space is not directly relevant or not well-tuned for the desription language employed.

An Illustration of Constructive Induction

We distinguish between two types of representation spaces. One is concept (or *hypothesis) representation space*, defined as the space that is searched for a hypothesis generalizing training examples. This space is spanned over descriptors (attributes, predicates, transformations, etc.) that are directly employed in the description of a hypothesis. Second is the *example representation space,* which is spanned over attributes occuring in the training examples. In the conventional machine learning methods, the example representation space and the concept representation space are identical.

In some inductive learning problems, the boundaries of concepts in the example representation space are very complex and thus difficult to learn. Such a problem is illustrated in Figure 1a. This is the so-called second Monk’s problem, which was used in the international competition of machine learning program [2].

[1] The paper makes a distinction between an attribute (a one-argument function that maps objects to attribute values; e.g., color or length of an object) and a feature (that expresses a specific value or property of an object (e.g., red or long).

In the diagram, spanned over six attributes x1, x2,..., x6, individual cells represent unique combination of different attribute values. Positive training examples are marked by “+” and negative training examples are marked by “-”. The shaded area represents the target concept, that is, the concept to be learned. As one can see, the boundaries of the target concept are very complex; therefore, learning the concept in this representation space is difficult. For this reason, conventional symbolic learning methods, such a decision tree learning, did not perform very well on this problem.

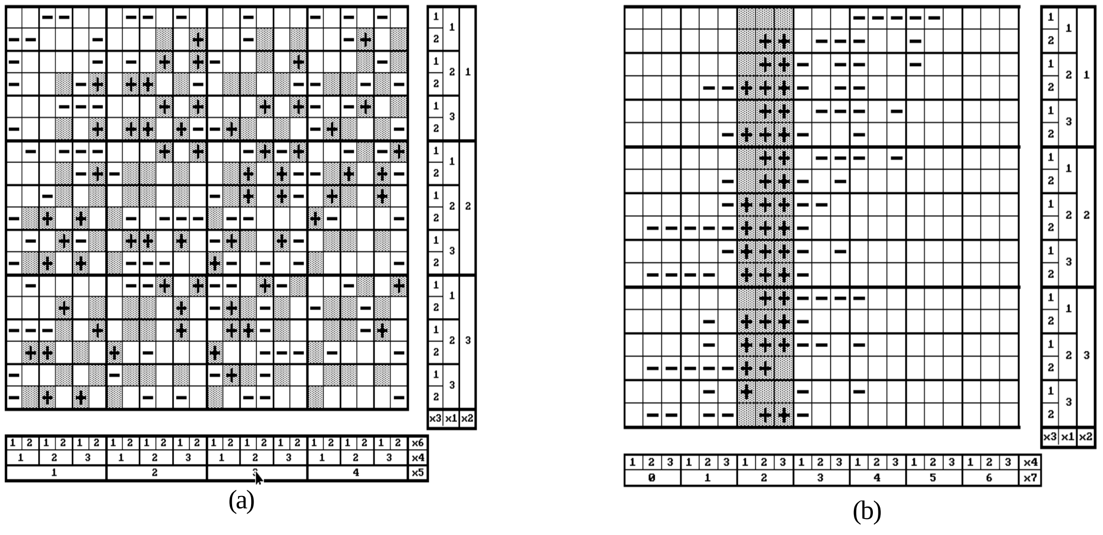

Figure 1. Diagrammatic visualization of the Monk 2 representation spaces: (a) the example representation space; and (b) the concept representaion space derived from the example space by the data-driven constructive induction method described in this paper.

In the example representation space, the learning problem is difficult because the target concept highly distributed. An improved representation space, determined by the AQ17-DCI system, is shown in Fig. 1(b). In this space, training examples are consolidated, and the target concept is highly regular; thus it is easy to learn. It is then desirable to use this improved space as the concept representation space. Details on how this representation space was generated are in the section “A General Schema for Constructive Induction”

What Makes a Learning Problem Difficult?

Learning problems can be difficult for a number of reasons, among which are an inadequate representation space, inadequate description language, or errors in training examples. The constructive induction methodology presented here addresses some problems posed by an inadequacy of the representation space. Specifically, is offers ways to cope with problems caused by indirectly or weakly relevant attributes, and/or an overprecision of attributes.

An overprecision of attributes leads to an unnecessarily large representation space and may make the process of finding the correct hypothesis difficult. It also may cause *overfitting* the data. An attribute overprecision frequently occurs when attributes are continuous. To avoid a potential problem, such attributes can be discretized, that is, their domain is split to ranges of values [4]. Formally, such a discretization is a form of an attribute abstraction operation[2]. Discretization is frequently done without taking into consideration other operators for changing the representation space. The presented methodology combines the effect of quantization with other space modification operators in order to produce a space that is highly amenable for determining a desirable hypothesis.

Attributes are indirectly relevant when their relevance to the given classification task is dependent on an interaction with one or more other attributes. The difficulty of describing an interaction depends on the description language used by the learning algorithm. For most symbolic inductive learning algorithms interactions involving logical conjunction or disjunction are easy to describe. However, interactions such as those most simply

[2] The Inferential Theory of Learning [12] identifies abstraction as any transmutation that reduces the amount of detail used to describe a given reference set.

represented as the equality or product of attributes may create significant difficulties for such methods. Even more difficult to capture are interactions that are represented by complex equation involving multiple attributes. An example of a multi-attribute interaction is the even/odd parity classification problem (classifying binary strings on the basis of parity of the number they represent). When there is complex multi-attribute interaction, attribute construction methods can be used that combine attributes in a problem-relevant manner [5], [6].

When the set of training examples contains attributes that are irrelevant for the given task, this can lead to a spurious hypothesis. Conventional selective induction learning methods are usually not affected by a small number of irrelevant attributes. Detecting and removing irrelevant attributes can be done either by a filter approach or a wrapper approach [7]. In the filter approach, attribute selection is performed as a pre-processing step to induction. Because it is separated from the induction algorithm, filters are fast, they can be used with any induction algorithms once filtering is done, and can be used on large datasets. The wrapper approach uses the induction algorithm itself to make estimates of the relevance of the given attributes, and can be viewed as a special case of hypothesis-driven constructive induction.

A General Schema for Construction Induction

Insights gained from reviewing problems that cause difficulties for conventional learning programs clearly indicate that the quality of the representation space is the major factor in learning an accurate hypothesis. When the representation space has high quality, concepts are represented in it simply, and therefore can be learned by almost any method. In real world problems, however, such quality can rarely be assured. The central goal of research on constructive induction is to develop methods capable of generating simple and accurate hypotheses for learning tasks, in which the original representation space is of poor quality. As mentioned earlier, constructive induction splits the process of learning an inductive hypothesis into two intertwined phases. Figure 2 shows a general scheme for constructive induction and illustrates these two searches.

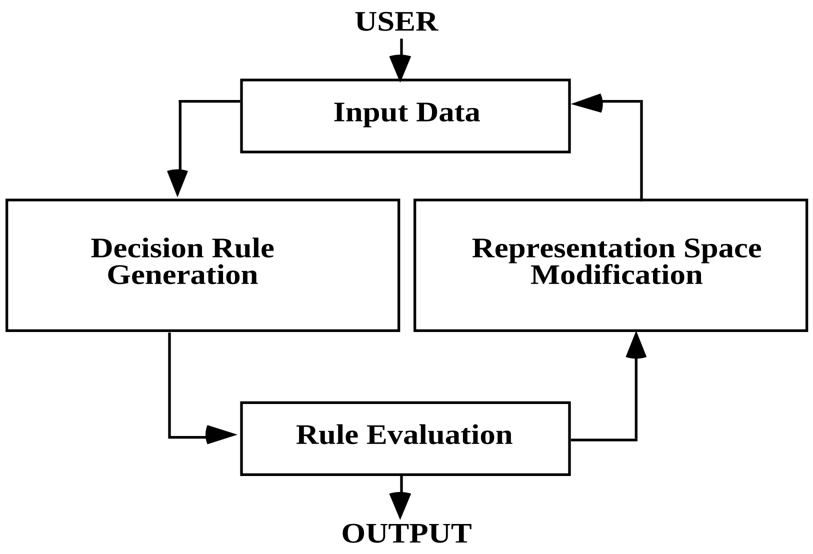

Figure 2. A general schema for constructive induction

Initially, input data consist of a user-provided set of training examples, plus a characterization of the initial representation space. This characterization includes a description of attributes, their types and their domains. The training data set is split into a primary and a secondary data set. The primary training set is supplied to the Decision Rule Generation module which generates initial concept descriptions (in our case, in the form of decision rules). These rules are evaluated in terms of their complexity, and their performance accuracy on the secondary training set. Based on the results of this evaluation, the system decides either to stop the learning process (when the obtained descriptions are satisfactory), or to move to the Representation Space Modification module. This module creates a new representation space, into which input data are then projected. This process repeats in cycles until the learned rules are satisfactory, or all planned modifications to the representation space have been tried. The final rules are evaluated on the testing examples to determine a more precise estimate of their performance accuracy.

The search for hypotheses within a given representation space is performed in AQ17-DCI by the AQ algorithm as implemented in AQ15c [9]. AQ15c performs a *separate and conquer strategy* to determine a set of decision rules which jointly cover all the positive examples and none of the negative examples (in the default case). This search starts by randomly selecting a 'seed' example of a class (concept), and applying the *extension-against* generalization operator to determine a set of general rules (a *star*) that cover the seed and do not cover negative examples. The best rule (according to a multicriterion evaluation function) is selected from the star, and examples covered by this rule are marked. A new seed is selected from among unmarked examples and the process is repeated until all examples in the given class are marked. A similar process is repeated for other classes, until rulesets for each class are obtained. The so produced hypotheses may be additionally improved by a process of *rule truncation*. The end result is a set of decision rules for each class in the data. An example of a rule produced by AQ is shown below.

Class1 <= [color = blue] &[height > 5”] & [shape = square or triangle]

or [height > 10”] & [shape = square]

This rule states "An object is in class1 if it is blue, its height is greater than 5”, and shape is square or triangle, or if its height is greater than 5” and its shape is square".

This example illustrates two important features of the program. One is that conditions of a rule can include “internal disjunction”, e.g., shape is square <u>or</u> triangle. The second is that rules for a given class can logically intersect. These features extend the expressive power of the learning method.

Each time a new set of rules (ruleset) is generated, its predictive accuracy is estimated using the secondary training set (if the number of training examples is small, a cross-validation method can be used). An advantage of the holdout method of evaluation at this stage (splitting training examples to a primary training and a secondary training set) is that rules learned from the primary training set perform well on the secondary set, are less likely to overfit the original data. Predictive accuracy is measured as the percentage of secondary training examples correctly classified. The *complexity* of a ruleset is evaluated by counting the number of rules in the ruleset and the total number of conditions (or selectors).

The quality of a ruleset is evaluated lexicographically. Rule sets are evaluated first according to the accuracy criterion. If the accuracy is within a user defined threshold of the goal accuracy, the rule set is then further evaluated according to the complexity criterion. If the rule set does not meet the minimum standard for accuracy, it is rejected and no further processing is done. The lexicographic evaluation permits the user to set a constraint on the minimum allowable accuracy.

The representation space modification (RSM) module is responsible for determining which modification operator to apply at a given stage, and making the changes to the training and testing examples. In AQ17-DCI the user can select which of the RSM operators should be tried (attribute construction, attribute selection, and/or attribute abstraction), or can accept a default setting that applies all operators. If in the "all operators" mode, they are applied in a pre-defined order: attribute selection, attribute abstraction, and then attribute construction[3].

Operator	**Arguments**	**Notation**	**Interpretation**
Equivalence	Attributes x,y	x = y	If x = y then 1, otherwise 0
Greater than	Attributes x,y	x > y	If x = y then 1, otherwise 0
Greater than or Equal	Attributes x,y	x>=y	If x ≥y then 1, otherwise 0
Addition	Attributes x,y	x + y	Sum of x and y
Subtraction	Attributes x,y	x - y	Difference between x and y
Difference	Attributes x,y	|x - y|	Absolute difference between x and y
Multiplication	Attributes x,y	x * y	Product of x and y
Division	Attributes x,y	x/y	Quotient of x divided by y
Maximum	Attribute set S	Max(S)	Maximum value in set S
Minimum	Attribute set S	Min(S)	Minimum value in set S
Average	Attribute set S	Ave(S)	Average of values in set S
Counting	Attribute set S,C	#Attr(S,C)	No. of attributes in S satisfying C

Table 1. Data-driven representation space expansion operators used in AQ17-DCI.

An exhaustive generate and test approach is used by the data-driven attribute construction process. A number of different operators are available to construct new attributes. New operators can easily be added to this set, but the aim was to provide simple, generally applicable operators that would be easy to generate and easy to interpret by a user. These operators include both binary operators and multi-argument operators (functions). Currently implemented in the binary group are the relational operator (determining whether the first input is less than, greater than, or equal to the second one) and a number of mathematical operators including addition, subtraction, absolute difference, multiplication, and integer division (Table 1).

[3] The AQ17-MCI [8] method extends this model by including additional representation space modification operators and by using meta-rules which relate problem characteristics to appropriate representation space modification operators.

In the attribute construction process each possible combination of attributes and the operators selected by the user from Table 1 is generated and evaluated. New attributes must exceed a user-defined minimum discriminatory power (as calculated by information gain ratio) and must also 'cost' no more than a user-defined threshold. The cost of a new attribute is the sum of the weights given to the original attributes used in the definition of the new attribute, plus the system-defined weight associated with each operator. Binary operators select attributes in pairs and multi-argument operators use unit information to determine set membership. Both attributes must satisfy type (e.g. ordered attribute values for addition), as well as unit constraints specific for the operator (e.g. both attributes must have the same units for the subtraction operator to be applied). These constraints are useful in reducing the number of possible combinations generated, as well as insuring the resulting new attributes are meaningful. In addition, the user may set a limit on the number of newly constructed attributes that are added to the representation space.

As an example of how AQ17-DCI works, recall the second Monk’s problem described earlier. For this problem, the improved representation space was found by the program by generating and evaluating the result of using multiple user-selected operators against the set of available attributes. One of the general operators available in AQ17-DCI (and the one that happened to be most useful for this problem), is an operator that generates *counting attributes*: #Attr(S, C). Such attributes measure the number of attributes in a set S that satisfy some condition C. As currently implemented, determination of membership in S is based either on user-provided attribute units, or by simply using the entire set of available attributes. The latter proved useful here. The program generated a number of new attributes and found that the counting attribute #Attr({x1,...x6}, Firstvalue) is highly relevant for this problem (the condition C=Firstvalue means that an attribute takes the first value of from its domain for a given object). In the hypothesis determination phase, the AQ17-DCI program found the following consistent and complete description of all the examples:

$$\#Attr(\{x1,...x6\}, Firstvalue) = 2,$$

which can be paraphrased: *an example belongs to the concept, if exactly two of six attributes take their first value*. It turned out that this rule exactly represents the target concept, and thus has a predictive accuracy of 100%.

Attribute selection can be done by applying one of the many existing attribute selection criteria. The current AQ17-DCI employs the information gain ratio. It selects for future processing the attributes with gain ratio greater than or equal to some predefined threshold.

Attribute abstraction in AQ17-DCI uses the ChiMerge algorithm to create ranges of attribute values as described in [10]. This is a bottom-up algorithm in which initially all values are stored in separate intervals, and then merged into ranges until a termination condition is met. The interval merging process consists of continuously repeating two steps: 1) compute χ^2 values (correlations between the value of the class attribute and the value of an attribute), and 2) merge the pair of adjacent intervals with the lowest χ^2 value. Intervals are merged until all pairs of intervals have χ^2 values exceeding the user-defined chi-threshold. The chi-threshold can be determined from a table and is a function of the desired significance level and the number of degrees of freedom (1 fewer than the number of classes). The χ^2 value measures the probability that the attribute interval and class value are independent. If the interval has a χ^2 value greater than threshold then class and interval are correlated and should be retained. High χ^2 threshold settings cause more intervals to be merged resulting in fewer total intervals, or attribute values. Empirically we have found that a χ^2 threshold of 0.9 (values range from 0.1 to 1.0) is a good default.

Experimental Applications

The presented methodology recognizes that attribute construction, attribute selection and abstraction all serve the same purpose, that is, to improve the initial representation space. A question arises how well this integration works in practice. To answer this question, this section presents results for applying the methodology to two real-world problems: text categorization and natural scene interpretation.

A. Text Categorization

Text categorization is the problem of classifying segments of text (usually documents) into a the best single class from a set of classes. In this problem, the specific task is to classify incoming newswire text as either of interest, or not of interest to a given user. The goal of constructive induction is to learn a description of the user' s interest (profile) from feedback from the user. This feedback consists of labels for a (usually small) set of documents giving the user's (binary) interest.

Attributes	**Description**
x1..x5	Top 5 subject categories as computed by the SFC text classifier.
x6..x59	POL people tags as computed by the IDD POL tagger. For each person identified, the vector contains the following string attributes: [name, gender, honorific, title, occupation, age]. 9 people (each with these subfields) are identified for each article.
x60..x104	POL organization tags as computed by the IDD POL tagger. For each organization identified, the vector contains the following string attributes: [name, type, acronym, country, business]. 9 organizations (each with these subfields) are identified for each article.
x105..x140	POL location tags as computed by the IDD POL tagger. For each location identified, the vector contains the following string attributes: [name, type, country, state] 9 locations (each with these subfields) are identified for each article.
x141..x141+n	The top n ranked tf.idf terms t1...tn are selected over all articles. For each article, position k in t1...tn has the tf.idf weight of term tk in that article.

Table 2. Attributes used for text description.

One of the most difficult aspects of this problem is in finding a good representation for the text. The use of constructive induction method builds on work reported in [11], which found that a hybrid representation for text, consisting of extracted ‘subjects’, person, organization and location (POL) attributes and keywords, coupled with a generalization hierarchy performs well for modeling newswire text (Table 2). In this previous work, combinations and subsets of attributes were generated and evaluated by hand. The goal of this work was to determine the effectiveness of a constructive induction approach to the problem of finding a good representation automatically.

In this problem, a user has an interest in “Medicine in the United States,” and has provided feedback on the relevance of 38 (18 positive and 20 negative) articles from a collection of 442 taken from the Colorado Springs Gazette Telegraph (Oct. through November 1994). The goal of learning is to find a description of the user’s interest profile so that news articles of interest may be automatically directed to the user. In this domain, articles are represented by the set of attributes including subject categories, POL entities and keywords as shown in Table 2.

In the previous work, reported in [11], the experiments were performed on representations consisting of a) only keywords (KEYWORDS), b) only POL’s (POL), c) only subjects (SUBJECTS) and d) all attributes (COMPLETE). The COMPLETE and SUBJECTS attribute sets were found to be the highest performing in the experiments previously performed. However, it was felt that constructive induction could improve both of these representations: the COMPLETE set by attribute selection and the SUBJECTS set by attribute construction.

The COMPLETE attribute set consists of 145 attributes. The DCI-Generate (AQ+DCI-Generate) method was run in addition to the DCI-Select (AQ+DCI-Select), which were then both compared to the results obtained for the COMPLETE (AQ-only), set. The expectation is that the selection of attributes will have the greatest impact on predictive accuracy. A 10-fold cross-validation methodology with a 70/30 split of the data set was used. The averaged results with the 90% confidence interval are shown in Table 3.

Attribute Set	**Method**	**Average Predictive Accuracy**
COMPLETE	AQ-only	54.2 +- 6.9
	AQ+DCI_Select	70.1 +- 5.8
	AQ+DCI_Generate	67.4 +- 7.6

Table 3. Comparison of Predictive Accuracy on COMPLETE attribute set

These results show that a significant performance improvement was obtained both by attribute selection and attribute construction. The greatest improvement was made by DCI_Select. This was expected because the COMPLETE set contains redundant ways of describing the article by providing a list of keywords, proper names and assigned subjects to each article. It seemed clear for this small sample that only some of these attributes would be needed for discrimination.. Eighty five attributes were removed including gender_person1, gender_person5 and type_org, which aren't strongly correlated with the USMED interest

Some improvement was also made by DCI-Generate in constructing new attributes using the counting attribute constructor #Attr(S,C), where S={Subject_1...Subject_i...Subject_n} and C is true when Subject_i is present in the given example. This transformation overcomes an awkwardness of the original representation. In the original representation, an article is described by an ordered vector of subjects, for example the description of a document as: [subject1=medicine] or [subject2= sports] or [subject3= finance]. The problem with this representation is that it does not capture the non-ordered nature of a person's interest in a subject. For example, if I am interested in sports, I don't really care if sports is in the subject1 or subject2 place. My interest is best stated as ‘Does any of the subjects include the value sports’. In this dataset there are 100 different possible subjects so extending the attribute set with all possible binary attributes would make the total vector very long and reduce the capability of the system to produce useful generalizations. These concepts are difficult to represent in the attribute-value representation, unless there is an additional attribute such as #Attr(Subjects, sports)>=1. AQ17-DCI did generate just such attributes.

The counting operator generates a new attribute for each of the 100 possible subject values. Each new attribute represents the number of times (value-cardinality) that value is present in the vector. Each new attribute is filtered for quality, so as not to overwhelm the learning algorithm. This operator constructed new attributes in six of the 10 runs. With the counting operator an article’s subject can be more succinctly stated. For example to say “the article is about finance” is represented as: *#Attr(subjects, finance)=1)*, and *#Attr(subjects, computers)=0* is used to state the “the article is not about computers”. Here is a rule describing articles of interest to the user (good) that used the counting generated attributes. The total coverage of the rule is shown by the t-weight. The u-weight counts the number of uniquely covered examples[4]::

```
Class_good<::
[subject1=sci_&_tech]& [subject2=institutions]& [subject3=economy or medicine]
[#Attr(subjects, finance)=0] &[#Attr(subjects,computer)=0]            (t:3, u:3)
```

An article is of interest, if subject1 is about science and technology, subject2 is about institutions, subject3 is about the economy or medicine, and if none of the subjects include finance or computers. Table 4 contains the results of this DCI-Generate

[4] The condition [Gender=m v f] is present in the rules because gender is assumed here to have three possible values: m, f and unknown.

transformation to the SUBJECTS attribute set, as well as the DCI-Select results compared against the baseline of no modification (AQ-only).

Attribute Set	**Method**	**Average Predictive Accuracy**
SUBJECTS	AQ	78.0 +- 6.6
	AQ + DCI-Select	78.0 +- 6.6 - No change
	AQ + DCI-Generate	84.7 +- 6.9

Table 4. Comparison of Predictive Accuracy on SUBJECTS attribute set

Data-driven constructive induction was able to improve upon both the COMPLETE and SUBJECTS attribute sets for the task of predicting user interest in newswire text. The transformations were quite comprehensible, in addition to bringing about up to a 29% improvement in predictive accuracy. Additionally, the construction of only useful forms of #Attr(Subjects, x) by the counting operator of AQ17-DCI allowed the learning algorithm to more efficiently overcome an awkwardness in the original representation of newswire articles that would have been otherwise greatly expanded the representation space, if generated by hand.

B. Natural Scene Interpretation

In the previous application it was found the both attribute selection and attribute construction improved the representation space, and the predictive accuracy and simplicity of the learned rules. This section details an application of how methods can be used together to solve a difficult problem in computer vision of classifying regions in images of natural scenes. In this problem the goal is develop a method that can accurately distinguish objects in outdoor scenes under varying perceptual conditions. The approach used is to learn characterizations of classes of natural objects (sky, trees, road) from images that have been labeled. These characterizations, based on attributes extracted for pixel windows, can then be applied to new scenes in order to predict the presence of natural objects

In the experiment the input to the learner was a training image which includes selected examples of the visual concepts to be learned: sky, trees and road. A windowing operator, of size 5x5 scanned over the training area, was used to extract a number of attributes. These attributes include color (color intensity of red, green and blue) and texture (in this case, we used Law masks for detecting horizontal and vertical lines, high frequency spot, horizontal and vertical v-shapes, and Laplacian operators). The quality of the generated rules was evaluated using a 10-fold cross-validation method. This data set has 450 examples equally distributed between the three classes.

	Average Accuracy	Average # of Rules	Average # of Selectors	Average Learning Time
AQ alone	72.5 %	27.7	94.8	231.7 sec

Table 5. Results after learning in the original representation space

The standard approach to solving this problem is to apply a selective induction learning algorithm to the raw data directly. The results obtained using this approach are shown in Table 5. This is the baseline performance. The results obtained after attribute construction, attribute selection and attribute abstraction are shown in table 6.

	Average Accuracy	Average # Rules	Average # Selectors	Average L. Time	Average CI Time
DCI-Sel	75.3 %	34.7	74.6[1]	215.1 sec.	3.1 sec.
DCI-Quant	85.9%[1]	34.6	114.7	10.8 sec.[1]	5.7 sec.
DCI-Gen	87.1%[1]	18.5[1]	63.5[1]	171.1 sec.[1]	8.1 sec.

([1]: significance $\alpha = 0.01$ [2]: significance $\alpha = 0.05$ [3]: significance $\alpha = 0.1$)

Table 6. Results after applying a single RSM operator to the representation space

Table 6 shows a dramatic improvement resulting from the data quantization. This abstraction operator reduced average attribute domain size from 256 to 14. The rules learned after the attribute values had been quantized, had a significantly higher prediction accuracy, and the learning time was significantly shorter than for rules learned in the original space. However, rule complexity increased from an average of 27.7 rules to 34.6 rules, and there was a significant increase in number of selectors (conditions) used in the rules. This increase in complexity is surprising and points to the need for further work on rule truncation.

The rules learned from the space expanded by DCI-Generate were also significantly more accurate than the rules learned in the original space. DCI-Generate added on average 10 new attributes, the strongest of which described absolute and relative differences in the amount of red, green and blue color intensities. Given the green trees, the dark road and the blue sky present in the training images this is not surprising. The tree class included new attributes that stated: [green > red] and [green > blue]. The introduction of these new attributes to the representation space resulted in a significant improvement to all aspects of the resulting rules. In the new DCI-Gen expanded representation space fewer rules which were significantly more predictively accurate, and less complex were learned in shorter time than in the original representation space.

Importantly, this improvement was achieved after only 8.7 seconds spent on searching for new attributes. The total time spent constructing new attributes, and then learning in the new space was 22% less than learning alone in the original representation space. The transformations the representation space made by the other operators were not as useful as those made by DCI-Quant and DCI-Gen. DCI-Sel was the next most useful transformation. DCI-Sel consistently removed attributes x4..x8, which are attributes like: horizontal and vertical line, high frequency spot, horizontal and vertical v-shape that describe the patterns within the 5x5 extraction window. This resulted in a slight increase in predictive accuracy, a significant reduction in the number of selectors present in the rules and in learning time, but an increase in the number of rules generated. These single-strategy results are encouraging and lead to the investigation of combinations of operators, especially attribute generation through DCI-Gen combined with abstraction of the space through DCI-Quant. The next section describes these experiments.

	Average Accuracy	Average # Rules	Average #Selectors	Average L. Time	Average CI Time
DCI-Quant ->DCI-Gen	85.4%^{1}	25.6	147.8	283.0 s	5.7+1.4 = 7.1 s
DCI-Gen ->DCI-Quant	93.4%$^{1(3)}$	20.5^{1}	63.7^{1}	104.5 s. $^{1(1)}$	8.1+19.3=27.4 s
DCI-Gen->DCI-Sel	90.3%^{1}	25.7	53.6$^{1(3)}$	142.2 s. $^{1(2)}$	8.1+1.0 = 9.1 s

^{1}: significance $\alpha = 0.01$ ^{2}: significance $\alpha = 0.05$ ^{3}: significance $\alpha = 0.1$

Table 7. Results after applying multiple RSM operators. Significance over the baseline is shown with the first superscript value. Significance to the first transformation alone is shown with the superscript in paremtheses.

Table 7 shows the results from combinations of the three RSM operators: DCI-Gen, DCI-Quant, and DCI-Sel that made significant improvements to the representation space. The previous section showed how initial abstraction of the space was useful, but may have been sub-optimal given the increase in rule complexity. This finding was reinforced when DCI-Gen was run on the abstracted space. Although the new attributes resulted in fewer rules, they were less accurate than those learned in the DCI-Quant only space, and were more complex. The contraction of the space may be removing some important information. If the operators are reversed and DCI-Quant is applied to the space already expanded by DCI-Gen, the space is significantly improved in almost all respects: Both learning time and predictive accuracy are now significantly better than in both the original space *and* the DCI-Gen only space. While rule complexity and number slightly increased from the DCI-Gen only space, but is still significantly smaller than in the original representation.

DCI-Sel used after DCI-Gen also results some improvements. DCI-Sel after DCI-Gen results in a small increase in predictive accuracy over DCI-Gen alone, a significant decrease in learning time and the number of selectors used, but an increase in the number of rules. In this space as in the original representation, DCI-Sel removed attributes x4..x8. DCI-Gen followed by DCI-Quant and then DCI-Sel was tried, but resulted in no improvement in predictive accuracy of the learned rules.

The multiple transformations made to the representation space: attribute generation followed by attribute quantization, resulted in rules which are significantly more accurate, can be learned much faster, and are much less complex than rules learned in the original representation.

The DCI-Gen combined with DCI-Quant result suggests that there is both an interaction between attributes and an excess of detail in the original representation. By performing DCI-Gen first, this interaction appears to be at least partially captured, and the abstraction operator of DCI-Quant can now safely perform its operation without looking at the context provided by the other attributes. Because DCI-Quant (using the Chi-merge algorithm) views each attribute independently it may remove information that is important to classification. Doing DCI-Gen first, this danger is reduced. It is premature, however, to draw any general conclusions about the best ordering of RSM operators from this single experiment. The conclusion that can be drawn from this is simply that some patterns are best described in the original formulation of the problem, and some only become apparent after abstraction. If abstraction is sensitive to interactions between attributes it may be possible to eliminate such ordering effects, but such a method would have to search an enormous space of both combinations and abstraction levels. An approach which more tightly couples the search for combinations and abstraction level is an interesting and important area for future research.

The attributes constructed by DCI-Gen were not only useful for classification, but also have an easily interpretable meaning, as differences in color intensities. The difference in color intensity between red and green, and between green and blue, were consistently in the top three most informative attributes as measured by information gain (Table 8). The difference between red and blue was also generated, but was not found to be of high discriminatory power.

Name	**Operator used**	**Description**
red - green	subtraction	intensity difference between red and green
green - blue	subtraction	intensity difference between green and blue

Table 8 - Examples of new relevant attributes constructed by DCI-Generate

The application of multiple data-driven operators helped not only to increase the prediction accuracy, but also generated meaningful new attributes. These transformations explicitly found one combination of color intensities based on difference that was useful. This ability to clearly see and understand the transformations made by the operators allows researchers to better understand the results. It may also inspire the use of other representations, for example the use of hue, intensity and saturation to represent color information in the image. Such an interaction cannot currently be found by the DCI method as it only combines pairs of numeric attributes. Searching for more complex functions of original attributes is an open area of future research.

The results from applying data-driven constructive induction to the above problem of natural scene interpretation problem shows that data-driven representation space transformations can significantly improve the results of rule induction.

Conclusion

This paper presented a modern view of constructive induction, specifically, as a double intertwined search—one for an improved representation space, and another for a set of hypotheses. Such constructive induction integrates within a uniform framework ideas and methods previously considered separately, such as methods of attribute selection, attribute construction and attribute abstraction. The usefulness of combining these methods has been shown by applications of the implemented program AQ17-DCI to two different practical problems: text categorization and natural scene interpretation.

The experiments with the presented methodology raise a number of interesting topics for future work. One such topic is the question of how tightly to integrate each of the available representation space modification operators. It was found in the natural scene interpretation problem that an ordering effect exists between attribute construction and abstraction. Developing a method for more tightly integrating these operators so that the correct level of granularity is present for the entire set of attributes available is difficult because of the massive size of the space being searched. Another topic for future work is the problem of learning "metarules" that would automatically guide the application of the representation space modification operators for any given problem. Initial work in this directions has been described in [8].

Trial experiments on the application of the AQ17-DCI methodology to two problems—text categorization and natural scene interpretation—indicate a great promise of the presented ideas for real-world inductive learning problems.

Acknowledgements

This research was conducted in the Machine Learning and Inference Laboratory at George Mason University. Thge experimental part concerning text classification was done by the first author at MITRE under internal research support. The GMU MLI Laboratory's research is supported in part by the National Science Foundation under Grants No. CDA-9309725, IRI-9020266, and DMI-9496192, in part by the Office of Naval Research under Grant No. N00014-91-J-1351, and in part by the Advanced Research Projects Agency under Grant No. N00014-91-J-1854, administered by the Office of Naval Research, and the Grant No. F49620-92-J-0549, administered by the Air Force Office of Scientific Research.

Bibliography

[1] Michalski, R.S., "Pattern Recognition as Knowledge-Guided Computer Induction," Technical Report No. 927, Department of Computer Science, University of Illinois, Urbana-Champaign, IL, 1978.

[2] Thrun. S.B., Bala, J., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K.A., Dzeroski, S., Fahlman, S.E., Hamann, R., Kaufman, K., Keller, S., Kononenko, I., Kreuziger, J., Michalski, R.S., Mitchell, T., Pachowicz, P., Vafaie, H., Van de Velde, W., Wenzel, W., Wnek, J., and Zhang, J., “The MONK’s Problems: A Performance Comparison of Different Learning Algorithms,” (revised version), Carnegie Mellon University, Pittsburgh, PA, CMU-CS-91-197, 1991.

[3] Wnek, J., and Michalski, R.S., "Hypothesis-driven Constructive Induction in AQ17-HCI: A Method and Experiments," *Machine Learning,* 14, p. 139-168, Vol. p. 1994.

[4] Dougherty, J., Kohavi, R., and Sahami, M., "Supervised and Unsupervised Discretization of Continuous Features", *Proceedings of the Twelfth International Conference on Machine Learning*, p. 194-202. San Francisco, 1995.

[5] Bloedorn, E., and Michalski, R.S., "The AQ17-DCI System and it Application to World Economics," *Proceedings of the Ninth International Symposium on Methodologies for Intelligent Systems*, p. Zakopane, Poland, 1996.

[6] Hu, Y., and Kibler, D., "Generation of Attributes for Learning Algorithms", *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI96)*, p. 806-811, Portland, OR, 1996.

[7] John, G., Kohavi, R., and Pfleger, K., "Irrelevant Features and the Subset Selection Problem", *Proceedings of the Eleventh International Conference on Machine Learning (ML94)*, Morgan Kaufmann, p. 121-129. 1994.

[8] Bloedorn, E., Wnek J., and Michalski, R.S., “Multistrategy Constructive Induction”, *Proceedings of the Second International Workshop on Machine Learning (MSL93)*, Harper’s Ferry, WV, Morgan Kaufmann, , May 27-29, 1993.

[9] Wnek, J., Kaufman, K., Bloedorn, E., and Michalski, R.S., "Selective Inductive Learning Method AQ15c: The Method and User's Guide", *Reports of the Machine Learning and Inference Laboratory*, MLI95-4, George Mason University, Fairfax, VA, 1995.

[10] Kerber, R., "ChiMerge: Discretization of Numeric Attributes," *Proceedings of the Tenth National Conference on Artificial Intelligence,* p. 123-128, San Jose, CA, 1992.

[11] Bloedorn, E., Mani, I., and MacMillan T.R., “Machine Learning of User Profiles: Representational Issues”, *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Portland, OR, August, 1996.

[12] Michalski, R.S., "Inferential Theory of Learning: Developing Foundations for Multistrategy Learning," *Machine Learning: A Multistrategy Approach*, R.S. Michalski and G. Tecuci (Eds.), San Mateo, CA: Morgan Kaufmann, 1993.