Deterministic Inductive Logic: A Multi-Valued Logic for Reasoning About Categories

Allen Brewer (abrewer.phd@verizon.net)

Abstract

Deterministic Inductive Logic (DIL) is a complement to classical (deterministic) deductive logic. It is a multi-valued logic that supports formulating theories (generalizations) by combining "specifics" into categories or classes. It relies primarily on four primitive operators (1) compare(), (2) contrast(), (3) combine(), and (4) factor() to: (a) provide a method for building classifications by generalizing about specifics either by defining a class from its members, or a super-class from its member classes; and (b) facilitate the creation of hierarchical structures compatible with classical deductive logic for evaluating hierarchical subsumption. The approach supports building knowledge structures from user defined concepts and categories into a system with the broader term and narrower term relationships typical of a controlled vocabulary. This paper describes the fundamentals of DIL and illustrates its application in preparing documents under the Federal Acquisition Regulations.

Overview

Deterministic inductive logic (DIL)[1] is the bottom-up complement of deductive logic. B*ottom-up* describes a classification that is constructed inductively as opposed to a classical ontology that is based upon axioms that must be known *a priori*. Induction forms generalizations from specifics. A generalization is a description that can be used to represent a group of objects, concepts, phenomena, etc. that have been associated or aggregated on the basis of some intention to describe a meaningful grouping or order. For consistency, a generalization's description must reflect each and every individual member's characteristics in the aggregate to properly reflect the characteristics of the group or aggregate. The description of a group or aggregate is a first order representation of the features that are common to the group. To represent a first order expression, the logic must express the: (1) characteristics that are present in every member of a group (always *true*); (2) characteristics that are not present in any member (never *true* or always *false*); (3) characteristics that are present in one or more members and absent in one or more members (existentially *true* and existentially *false* or *indeterminate*); and (4) characteristics that are, inaccessible, hidden, or not known, (*unknown*).

Probabilistic inductive values can be mapped to deterministic inductive truth-values. The two probabilistic endpoints 0.00 and 1.00 represent universally quantified characteristics. Values in the probability range $0.00 < x < 1.00$ are indicative of

---

[1] Brewer, Allen E. (2000). *Deterministic Inductive Logic: A multi-valued logic for reasoning about categories.* University of Maryland.

existentially quantified characteristics. The mapping results in three of the four distinct truth-values: (1) universally *false* (0.00), (2) universally *true* (1.00) and (3) existentially *true* or existentially *false* ($0.00 < x < 1.00$). Unknown features are not expressly represented in probabilistic models.

DIL employs four truth-values to insure that its categorical descriptions are quantified. When a truth-value is expressed in relation to a single observation it is universally quantified with respect to that individual and may be any one of: {*true, false, unknown*}, where:

{t*rue*} a characteristic of an individual that is true,

{*false*} a characteristic of an individual that is false,

{*unknown*} a characteristic of an individual that is not known.

When the truth-value representations of two or more individuals are aggregated, the truth-values are represented by one of four truth-values: {T, F, I, U} where:

{T} a characteristic of a category for which that characteristic is universally true,

{F} a characteristic of a category for which that characteristic is universally false,

{I} a characteristic of a category for which that characteristic is both existentially true and existentially false,

(U} a characteristic of a category for which there is insufficient information to assess whether the aggregate is universally or existentially, true or false.

DIL supports reasoning about collections of individual objects and entities by constructing a representation of the characteristics of individual cases, objects, etc., that are associated or aggregated into a group. The reasoning is facilitated by evaluating descriptions of categories or aggregates as expressed in feature vectors.

A feature vector in DIL is a sequence of truth-values such as "TTFFUIIIITFUIITF." To be meaningful the semantics of feature vectors (sequence of truth-values and basis for truth-value assignments) must be the same for any two feature vectors combined, compared or contrasted in DIL.

Functionally, *deterministic inductive logic* provides a method for defining a full First Order Logic (FOL) description of an aggregate and by representing the aggregate as a FOL description provides an approach for building hierarchical classifications by aggregating descriptions of individuals and aggregates to create successively higher order generalizations. These hierarchical classifications can be used to represent conceptual structures. The classification or conceptual structure can be deductively evaluated by assessing the subsumption of category's descriptive feature vectors.

Deterministic Inductive Logic Operators

Deterministic inductive logic relies on four operators that are used to analyze descriptive feature vectors.

1. The *compare* operator [COMPARE( )] is used to compare two descriptive feature vectors to assess their similarity.
2. The *contrast* operator [CONTRAST( )] is used to compare two descriptions to assess their difference.
3. The *combine* operator [COMBINE( )] is used to formulate an aggregate descriptive feature vector by combining two or more feature vectors used to represent individual cases, combinations of individuals or combinations of individuals and aggregates.
4. The *factor* operator [FACTOR( )] is used to compute a *remainder* description by differentiating one description (factor(1)) from another (factor(2)).

Truth tables are used to define the operators in terms of the four basic truth-values defined in DIL. For illustration a traffic light metaphor can be used where the colors of the lights in a traffic signal can be conceptualized, as they would represent one's permission to transit an intersection. The color green is associated with *true*, red with *false*, yellow with *indeterminate* and white (no-color) with *unknown*.

Compare

The rules for comparison are:

1. Like colors result in green [if similar then *true*]
2. Unlike colors result in red [if dissimilar then *false*]

Compare	Unknown	True	False	Indeterminate
Unknown	True	False	False	False
True	False	True	False	False
False	False	False	True	False
Indeterminate	False	False	False	True

Figure 1. Deterministic inductive logic's compare operator truth table

Contrast

The rules for contrast are:

1. Like colors result in red [if similar then *false*]
2. Unlike colors result in green [if dissimilar then *true*]

Contrast	Unknown	True	False	Indeterminate
Unknown	False	True	True	True
True	True	False	True	True
False	True	True	False	True
Indeterminate	True	True	True	False

Figure 2. Deterministic inductive logic's contrast operator truth table

Combine

The rules for combination are:

1. If all the values to be combined are green then the result is green [always true].
2. If all the values to be combined are red then the result is red [always *false*].
3. If there are green and red values to be combined, or any of the values to be combined are yellow then the result is yellow. Mixed greens and reds or yellows are not a single determinate value and so inferentially the value of that respective feature, in the aggregate, is *indeterminate*.
4. If none of the first three rules has resulted in the assignment of a truth-value, the result is white. Missing or unknown values render the value of the aggregate to be unknown since at least one value is required to determine if the aggregate is *true* or *false*. If the aggregate is *indeterminate* combining an *unknown* with an *indeterminate* does not alter the fact that the aggregate is *indeterminate*.

Combine	Unknown	True	False	Indeterminate
Unknown	Unknown	Unknown	Unknown	Indeterminate
True	Unknown	True	Indeterminate	Indeterminate
False	Unknown	Indeterminate	False	Indeterminate
Indeterminate	Indeterminate	Indeterminate	Indeterminate	Indeterminate

Figure 3. Deterministic inductive logic's combine operator truth table

Multi-faceted Combine

Figure 4 illustrates extending DIL for situations in which the representation of expressed features is not binary. The example is used to demonstrate how the DIL combination operator might be used to model a characteristic that may have multiple non-overlapping values. The example illustrates a four-facet situation, such as might be presented by the chemicals sequenced in DNA, where each base pair of a molecule of DNA is composed of one of four chemical bases represented by the letters A (adenine), T (thymine), G (guanine) and C (cytosine). A DIL multi-valued logic, using DNA as a basis of illustration would represent the four chemicals present in DNA base pairs {A, C, T, G} as four distinct truth-values and would require in addition values for *indeterminate* and *unknown*. The values displayed in Figure 4 as *indeterminate* are analogous to single nucleotide polymorphisms or SNPs.

Combine	Unknown [U]	A	C	T	G	Indeterminate [I]
Unknown [U]	U	U	U	U	U	I
A	U	A	I	I	I	I
C	U	I	C	I	I	I
T	U	I	I	T	I	I
G	U	I	I	I	G	I
Indeterminate [I]	I	I	I	I	I	I

Figure 4. Combine operator using a four-facet logic

Factor

Given two descriptive vectors {*vector*(1), *vector*(2)}; *vector*(3) = FACTOR(*vector*(1), *vector*(2)) where *vector*(2) = COMBINE(*vector*(1), *vector*(3)) AND *vector*(1) = COMBINE(*vector*(2), *vector*(3)). The factor operator, unlike the other operators has multiple "possible" truth-value results in cases of factoring *indeterminate* and *unknown* features.

Factors	Unknown	True	False	Indeterminate
Unknown	Unknown	NONE	NONE	True False Indeterminate
True	NONE	True	NONE	False
False	NONE	NONE	False	True
Indeterminate	True False Indeterminate	False	True	Indeterminate

Figure 5. Deterministic inductive logic's factors function truth table

Analyzing vectors using DIL

Five types of descriptions are defined in DIL and used to facilitate building and evaluating the relationships between and among categories in hierarchical classifications that are constructed inductively. The five types of vectors are:

(1) Individual case description (feature vector) used to describe individual cases, observations, phenomena, etc.;

(2) Generalization descriptions (feature vectors) used to describe aggregates, collections or associations of objects, entities, situations, etc.;

(3) Identity vectors;

(4) Difference vectors; and

(5) Discriminant vectors.

An individual or case description (feature vector) may be used to express the presence or absence of features or characteristics exhibited, expressed or associated with or by an individual. The individual is the most specific level of description. Features that for any reason cannot be assessed are assigned a truth-value of *unknown*.

An unknown might be used, for example, to allow inclusion of features in a quality assurance context where those features are only knowable as the result of a destructive test. Since the truth-value of the feature only becomes known upon completion of the destructive test, the description may be complete by specifying the feature as *unknown* until after the destructive test is conducted. When known, the *unknown* can be replaced with the appropriate value discovered during the destructive test.

A *generalization*-description (feature vector) is defined by combining descriptive feature vectors. A generalization-vector may be used, for example, to test two vectors for hierarchical subsumption, where one of the vectors subsumes the other on the basis of the features expressed. A hierarchical generalization can be formed to subsume any two feature-vectors by combining those two feature vectors (see Figure 6).

Generalization-XX+XY = COMBINE(*feature-vector-XX*, *feature-vector-XY*)

Figure 6. Generalization formula

The role of an *identity-vector* is to facilitate comparing two vectors. An *identity-vector* is derived from the comparison of any feature vector with itself.

Identity-vector = COMPARE(*feature-vector-XX*, *feature-vector-XX*)

Figure 7. Formula for creating identity vectors

The negation of an *identity-vector* is a *difference-vector*. A *difference-vector* is derived from the contrast of any feature vector with itself.

Difference-vector = CONTRAST(*feature-vector-XX*, *feature-vector-XX*)

Figure 8. Formula for creating difference vectors

Discriminants

A *discriminant* is a homogeneous characteristic (universally quantified) in an aggregate description. The term *discriminant* is not applied to individual *case-vectors*, but is reserved for *generalization-vectors*. In a binary logic, discriminants can be either inclusionary (*true)* or exclusionary *(false)*. A *discriminant-vector* is a derived vector type, used to identify the features that might be represented in a query to represent the discriminating characteristics or conditions for formulating inclusionary and exclusionary query components. Figures 9 and 10 specify the formulas for constructing inclusionary and exclusionary discriminant vectors.

Inclusionary-discriminant-vector = COMPARE(COMBINE*(case-list), identity vector*)

Figure 9. Formula for creating an inclusionary discriminant vector

Exclusionary-discriminant-vector = COMPARE(COMBINE*(case-list), difference-vector*)

Figure 10. Formula for creating an exclusionary discriminant vector

Subsumption tests

A classification uses hierarchy to manage descriptions at multiple levels of specificity. For example, the general topic “physics” subsumes many sub-topics such as thermodynamics, motion, etc. An inductive classification defines a class using the combine operator to aggregate a set of individual descriptions. Similarly it may be used to define a super-ordinate class by combining two or more class descriptions. When the combine function is used to define a class or category, each of the constituents at each level of combination that were included, in any of the combined descriptions, are logically members of the super-ordinate class. The created class necessarily subsumes every case that was explicitly or implicitly combined to form that generalized category. Figure 11 illustrates the test for hierarchical subsumption. In this illustration the

aggregate is defined by combining a list of individual cases. The test is structured to determine whether a particular case(X) is one of the type of cases given by the case list by testing the hierarchical relationship between the class description and the case description.

case(X) is subsumed by COMBINE (*case-list*)

IFF COMBINE (*case-list, case(X)*) = COMBINE (*case-list*)

Figure 11. Test for hierarchical subsumption

The subsumption test can be generalized into two tests that can be used for testing the relationship between any two sets of objects in terms of their characteristics as described by feature vectors, as illustrated in Figure 12.

IF COMBINE (*case-list(1), case-list(2)*) = COMBINE (*case-list(1)*) **THEN**

COMBINE (*case-list(2)*) **is subsumed by** COMBINE (*case-list(1)*)

IF COMBINE (*case-list(2), case-list(1)*) = COMBINE (*case-list(2)*) **THEN**

COMBINE (*case-list(1)*) **is subsumed by** COMBINE (*case-list(2)*)

Figure 12. Set relationship subsumption tests

Equivalence tests

Assessing the equivalence of two descriptions (feature vectors) requires a measure of *similarity*. Alternatively assessing the relative difference between two vectors requires a measure of *difference*. Either of these measures can be used as the basis for assessing whether two descriptive vectors are equivalent. For two vectors to be equivalent, every feature in the two vectors must be identical.

Two measures (1) *degree of similarity* (DOS), and (2) *degree of difference* (DOD) are defined to assess vector equivalence. If any two vectors are 100% similar or 0% different, they are identical.

- The degree of similarity (DOS) is calculated by dividing the number of *true* truth-values in a vector created by a compare function by the total number of truth-values in that vector.

- The degree of difference (DOD) is calculated by dividing the number of *false* truth-values in a vector created by a contrast function by the total number of truth-values in that vector.

  For two vectors (represented by *a* and *b*) being compared:

Equality:	DOD(COMPARE(a, b)) = 0.00
	DOS(COMPARE(a, b)) = 1.00
Inequality:	DOD(COMPARE(a, b)) > 0.00
	DOS(COMPARE(a, b)) < 1.00

Document Creation Illustration

To illustrate DIL, an application in which DIL provides a system for encoding decisions will be demonstrated. DIL was defined for use in constructing decision logics for complex contexts where relationships are discovered over time or where conditions tend to change with usage and experience. The particular illustrative application deals with controlling the inclusion and exclusion of standard texts for constructing documents.

This particular application of DIL is intended to facilitate the generation of solicitation and contract instruments under the Federal Acquisition Regulations (FAR). The federal acquisition environment is both a highly complex environment and it is subject to legislative, regulatory and experiential changes, developments and improvements. The specific illustration addresses a set of standard text components selected from 48 CFR 14.201-6 to illustrate how DIL might be used in deciding inclusion and exclusion of provisions and clauses.

The steps required for constructing a system to control inclusion and exclusion of standard text components requires:

- Determining the characteristics of the *environment* and mapping those characteristics to features in a feature vector;
- Encoding the applicability and use of individual standard texts for use in acquisition instruments in terms of:
  - Mandatory inclusion,
  - Mandatory exclusion,
  - Required when applicable – applicability and use provisions,
  - Optional – scope of permissive applicability,
  - Authority – delegations of authority to a user that affect their permission to autonomously include/exclude optional texts;

- Encoding the characteristics of a specific *acquisition action* to construct an acquisition instrument containing all relevant standard texts given the characteristics of a specific situation.

The goal of the particular application of DIL is to construct legally sufficient solicitation or contract instruments by including the provisions and clauses that are appropriate in the particular acquisition context. Table 1, columns 1 and 2 list a set of characteristics found in the FAR acquisition environment derived from evaluating the applicability and use of a set of provisions and clauses in a specific section of the FAR. Table 1, column 3 lists a set of characteristics picked to represent a hypothetical acquisition action.

Table 1. Hypothetical acquisition conditions

Feature	Condition	Acquisition Condition
F(1)	Invitations for Bids	T
F(2)	Construction	T
F(3)	Telegraphic bids are authorized	T
F(4)	Place of performance specified by the government	F
F(5)	Multiple award anticipated	F
F(6)	Facsimile bids authorized	T
F(7)	Uniform contract format applicable	T
F(8)	Step one of two step sealed bidding	F
F(9)	Step two of two step sealed bidding	T
F(10)	Multiple technical proposals authorized	T

A standard text is ***included*** when that text's descriptive feature vector is subsumed by the action's descriptive feature vector. A standard text is ***excluded*** when a text's descriptive feature vector is not subsumed by the action's descriptive feature vector. The test is a hierarchical subsumption test. A standard text should be included when it's applicability and use is consistent with the characteristics of the acquisition action. Restated, a text is relevant in a solicitation or contract when it's applicability is within the scope of the characteristics of that specific acquisition action.

Table 2 illustrates the treatment of selected provisions from 48 CFR 52.214-xx. The applicability and use of the selected provisions is defined at 48 CFR 14.201-6.

Table 2. Included and excluded provisions given the hypothetical acquisition conditions in Table 1

	F(1)	F(2)	F(3)	F(4)	F(5)	F(6)	F(7)	F(8)	F(9)	F(10)
Instrument	**T**	**T**	**T**	**F**	**F**	**T**	**T**	**F**	**T**	**T**
Include										
52.214-1	T	I	I	I	I	I	I	I	I	I
52.214-3	T	I	I	I	I	I	I	I	I	I
52.214-4	T	I	I	I	I	I	I	I	I	I
52.214-5	T	I	I	I	I	I	I	I	I	I
52.214-6	T	I	I	I	I	I	I	I	I	I
52.214-7	T	I	I	I	I	I	I	I	I	I
52.214-13	T	I	T	I	I	I	I	I	I	I
52.214-14	T	I	I	F	I	I	I	I	I	I
52.214-31	T	I	I	I	I	T	I	I	I	I
52.214-12	T	I	I	I	I	I	T	I	I	I
52.214-25	T	I	I	I	I	I	I	I	T	I
Exclude										
52.214-9	T	**F**	I	I	I	I	I	I	I	I
52.214-10	T	**F**	I	I	I	I	I	I	I	I
52.214-22	T	I	I	I	**T**	I	I	I	I	I
52.214-23	T	I	I	I	I	I	I	**T**	I	I
52.214-24	T	I	I	I	I	I	I	**T**	**F**	T

The test conditions that result in the subsumption test failure in Table 2 and thereby cause a provision or clause to be excluded are illustrated in bold type.

Conclusions

DIL facilitates the representation of complex decisions whose applicability can be tested by assessing the relevance of a set of criteria to a context using a logical subsumption test. The example illustrated demonstrates how DIL might be used for deciding whether a provision or clause is included in an acquisition instrument by testing

if the text is applicable within the scope of the instant acquisition. The logic facilitates evaluating contextually appropriate behaviors or results based upon assessing descriptive vectors that represent the characteristics of a case within an environment.

In the demonstration example, DIL facilitates constructing a document generator that can retrieve the relevant components from a database of possible components by selecting the ones that are appropriate in a specific context. The demonstration example illustrates how DIL might be applied in the context of information filtering or retrieval to control the outcome or behavior of a system by selecting and executing only appropriate behaviors in context.