Paper 148-2011

Conditional Classes and Pattern Recognition in SAS®

Scott Hanson, Bank of America, Calabasas, CA

ABSTRACT

Conditional classes are data objects that do not have tidy rules of membership. Their characteristics make up, more or less, the possibilities of membership rather than probabilities.

How one thinks about conditional classes is critical. Complexities of pattern recognition, an area that has received strong practical implementation in SAS, will serve to illustrate questions of assignment, data ambiguity, and data objects that do not have precise relation and certain membership.

CONDITIONAL CLASSES

Data objects of conditional classes are not binary in any statistical sense. They do not fit a class precisely, or true, or one. They do not fit a class precisely with another class. Conditional classes are characterized by inclusion, convergence, proximity, and intersection, which take precedence over probabilities, truth, and certainty.

Toxicity, adequate tea consumption, and genetically engineered (GE) food are examples of data objects belonging to a conditional class: Two cups of tea consumption a day may be adequate by members of certain British classes but insufficient for more robust tea devotees. Exposure of less than 0.010 mg per liter for arsenic is deemed satisfactorily safe by the EPA but may be objectionable by certain physicians and radiologists. A GE tomato meets the nutritional standards set by the FDA but may not have as much credence in the eyes of the public.

As a mental exercise, consider for a moment the opposite of conditional classes, 'Exact' classes. Exact classes are typified by precise correspondence, truthful properties, and clear membership to a class. Ordered pairs (x,y) set to a probability between 0 and 1 is an expression of an exact class. If the relation (x,y) is clear then the rules of membership must be articulate and vice versa. For conditional classes, the ordered pairs are approximate or subjective. Certainty and equivalence support the concept of an Exact classification as much as uncertainty and approximation belie the concept of conditional classification.

Lastly, a conditional class is a set of characteristics on a hierarchical continuum. The decision to include an element is intricate if the characteristics of the conditional class are abstruse or uncertain or ambiguous. Typically, a hierarchy is present which helps to ascertain the level of uncertainty or ambiguity or imprecision of the relation. An assignment of element A to element B typically is predicated on a hierarchical set of rules whereby imprecision or subjectivity (or other characteristics of a conditional class) can be discerned.

EVALUATING A CONDITIONAL CLASS

Evaluating a conditional class can be problematic. How do we postulate exact or conditional classes in the area of epidemiological research? We may first question if the relationship between exposure and a dichotomous disease variable is unambiguous or causal (Snow (1936) in his famous study on cholera in London, was able to discern a direct relationship). One may argue that the data object is an example of an exact class. On the other hand, the scientific tendency of contemporary epidemiological thought in general is multivariate. In the study of disease on a populace, complexity equates to consideration of extraneous influences. Complexity is not a necessary condition but is often a factor in the absence of an exact class.

How do we postulate exact or conditional classes in certain physiological functions? Sensory interpretation - reading, hearing, touching – follows an inherently complex mechanism of recognizing a pattern of some kind. Faulty reading is a case of imperfect or uncertain recognition, as is memory loss, hearing insensitivity and muted smell. Sight, even if slightly imperfect, involves a reliable, precise correspondence. Would it be appropriate to consider exact class as unequivocal correspondence as a result of a single examination? Perfect vision, albeit a conspicuous attribute of good health, is in fact an exceptionally close approximation.

With what precision can we ascertain data objects whose mechanisms are even less reliable than physiological processes? How well can we recognize oil residue at the lowest depth of a voluminous and turbulent sea? Other examples are genetic sequences mapping and categorizing fresh-water sport fish. Our reliance on machines to correctly identify objects can be paradoxical. We need to distinguish properties of data objects that purport truthful claims when they are often approximations or subjective or imperfect.

PATTERN RECOGNITION AND SAS FUNCTIONS

One area of classification that deals with conditional classes is pattern recognition. SAS provides several algorithms including, in no rank order, the SPEDIS function (short for spelling distance), COMPGED, COMPLEV, SOUNDEX, and CALL COMPCOST. Another algorithm to be discussed, though not available in SAS, is the Daitch-Mokotoff Soundex System. Regardless of the selected algorithm, the underlying problem of pattern recognition is the absence of an exact class.

First, a synopsis of each algorithm:

1. SPEDIS: Determines the likelihood of two words matching, expressed as the asymmetric spelling distance between the two words (slowest performance).

2. COMPLEV: Returns the Levenshtein edit distance between two strings (symmetric, fastest performance).

3. COMPGED: Returns the generalized edit distance between two strings (asymmetric).

4. CALL COMPCOST: Sets the costs of operations for later use by the COMPGED function (asymmetric to symmetric).

5. SOUNDEX: Encodes a string to facilitate searching. The SOUNDEX function encodes a character string according to an algorithm that was originally developed by Margaret K. Odell and Robert C. Russell (US Patents 1,261,167 (1918) and 1,435,663 (1922)).

6. DAITCH-MOTOKOFF: A modification to the U.S. (i.e., Russell's SOUNDEX) System. Essentially, DAITCH-MOTOKOFF is distinguishable by logic that addresses Germanic and Slavic surnames. The algorithm is often misidentified as the Jewish Soundex System, the Eastern European Soundex System, or the European Soundex System because of its origins.

SPEDIS FUNCTION

SPEDIS determines the distance between a keyword and a query, expressed as the asymmetric spelling distance between two words. Gershteyn (2000) hypothesizes a writer who mistakenly types 'SUGI' instead of 'SAS' and wants to know the spelling distance between 'SAS' and 'SUGI' (a novel idea!).

SPEDIS resolves this problem of pattern divergence by calculating the cost of each operation necessary to convert the keyword to the query. The costs of each operation that is required to convert are listed in the following:

Operation	Cost	Explanation
Match	0	no change
Singlet	25	delete one of a double letter
Doublet	50	double a letter
Swap	50	reverse the order of two consecutive letters
Truncate	50	delete a letter from the end
Append	35	add a letter to the end
Delete	50	delete a letter from the middle
Insert	100	insert a letter in the middle
Replace	100	replace a letter in the middle
Firstdel	100	delete the first letter
Firstins	200	insert a letter at the beginning
Firstrep	200	replace the first letter

In addition to the hierarchical set of Operation and Costs, SPEDIS has these characteristics:

1. Returns non-negative value <= 200
2. The distance is the sum of the costs divided by the length of the query. If this ratio is greater than one, the result is rounded down to the nearest whole number.
3. Asymmetric, SPEDIS(QUERY, KEYWORD) <> SPEDIS(KEYWORD, QUERY)

In the case of the writer's typographical error, the spelling distance is either 58 or 83, depending on the order of evaluation. The writer decides that a low score of less than 50 is a match and rejects that the typo was in fact 'SAS' (a 0 score is an exact match). A low score is directly a measure of the writer's error tolerance.

SOUNDEX FUNCTION

Imagine genealogists poring over an index of a several hundred thousand records over several years only to notice a misspelled word. In 1918, Robert C. Russell developed an algorithm to assist these ardent genealogists who wrestled with changing surnames from census data (Fan, NESUG 16). In his patent application filing, Russell claims a 'certain new and useful Improvements in Indexes' and 'full, clear, and exact description of the invention, such as will enable others skilled in the art to which it appertains to make and use the same'. What does he hope to achieve exactly?

> "…one object of the invention being to provide an index wherein names are entered and grouped phonetically rather than according to the alphabetical construction of the names.
>
> A further object is to provide an index in which names which do not have the same sound do not appear in the same group, thus shortening the search.
>
> A further object is to provide an index in which each name or group of names having the same sound but differently spelled shall receive the same phonetic description and definite location.
>
> With these and other objects in view, the invention consists in certain novel features as hereinafter set forth and pointed out in the claims…"

In Russell's thinking, we see a deep understanding of the language at work, a language variant, approximate, and inconsistent. Unsophisticated knowledge of the language and other circumstances most likely led to inconsistent spellings in the index census. The surnames "Smith" and "Smyth" for example were a common point of divergence in the index census. Although these two names converge phonetically, they fall far apart in an alphabetical index.

"Soundexing" helped to address this inadequate mapping between the English alphabet and its pronunciation.

As a conditional class, the observations of genealogical data are arbitrary lengths and vast in number (besides alphabetic dissimilarity). SOUNDEX maps this conditional data object to a four-byte literal, which is a relation from a vast space to a small space. A very serious question is how to infer two dissimilar names which result in the same Soundex code:

> The SOUNDEX code for "Hilbert" is "H416".
> The SOUNDEX code for "Heilbrom" is "H416".

Reconciling "Hilbert" and "Heilbrom" presents a crucial dilemma, that is, the errors associated with an imperfect data object.

Below are a few additional examples of phonemic contrasts and SOUNDEX results:

SOUNDEX codes:

Name1	Name2	SOUNDEX (Name1)	SOUNDEX2 (Name2)
Smith	**Smythe**	**S53**	**S53**
Jean	**Gene**	**J5**	**G5**
Beijing	**Peking**	**B252**	**P252**

DAITCH-MOTOKOFF (D-M) SOUNDEX SYSTEM

In 1985, Gary Motokoff had the challenging task of indexing the names of 28,000 persons who changed their surnames while living in Palestine between 1921 and 1948. The majority of these names were Germanic or Slavic. Because many Jewish names with identical phonetic sounds did not yield identical Russell Soundex values, Motokoff created an improved phonetic system to address this divergence from the English standard by Russell. The aim is a more satisfactory correspondence.

Name1	Name2	SOUNDEX (Name1)	SOUNDEX2 (Name2)
Moskovitz	Moskowitz	M2132	M232

It was obvious to Motokoff there were numerous spelling variants of the same basic surname and thus the list should be soundexed accordingly.

D-M has a wide audience, accepted by the Hebrew Immigrant Aid Society (HIAS) and acknowledged as the standard at the U.S. Holocaust Memorial Museum in Washington, D.C. (see **http://www.avotaynu.com/soundex.htm).**

LEVENSHTEIN ALGORITHM

Russian scientist Vladimir Levenshtein developed a string-matching algorithm, similar to SPEDIS, in 1965. Known also as the "Edit Distance", the algorithm constructs an (m,n) cell matrix, and performs operations of Insert, Delete, Replace. Because its operations are performed in a matrix, the algorithm is symmetric, unlike the SPEDIS function.

A BRIEF EVALUATION

Given this backdrop, of the algorithms that we select, how would they evaluate a simple if innocuous text string '123456789'? Should we assume that the refinements of the algorithms, which are often rooted in a particular data object (Slavic names, e.g.), have a wider utility? Further, knowing the availability of these algorithms in SAS and that the documentation's examples are generic, suggests a receptive audience.

We have seen that imprecision and other characteristics of an archival data base led Russell to subjective but systematic and rigorous undertaking. Let us assume that the string '123456789' is perhaps a Social Security Number, a default value, a special term, or an outlier with no explanation. How should we discriminate Russell's algorithm and other specific, reliable algorithms when evaluating innocuous terms? Should we expect pattern recognition algorithms, text recognition algorithms in this case, to evaluate closely given the text string is not nuanced towards any particular data object?

The SAS code below tests various unique but not exhaustive typical operations on a nine-byte string '123456789'. What is the extent of convergence of the string we are testing and a string that is closely similar? Does choice of the algorithm matter?

The following SAS code creates a data set of 11 unique types of operations, which are simple, non- intricate modifications. However, note that the last three operations are relatively distant cases, highly divergent, from the string '123456789'.

```
data testssn;
        length ssn1 $9 operation $14;
        ssn1 = '123456789'; operation = 'match'; output;
        ssn1 = '23456789'; operation = 'insert first'; output;
        ssn1 = '12345678'; operation = 'insert last'; output;
        ssn1 = '12346789'; operation = 'insert middle'; output;
        ssn1 = '213456789'; operation = 'swap first'; output;
        ssn1 = '123456798'; operation = 'swap last'; output;
        ssn1 = '113456789'; operation = 'double first'; output;
        ssn1 = '123455789'; operation = 'double middle'; output;
        ssn1 = '987654321'; operation = 'reverse'; output;
        ssn1 = '         '; operation = 'null'; output;
        ssn1 = 'abcdefghi'; operation = 'no match'; output;
run;
```

```
data testssn_;
        set testssn;
        ssn = '123456789';
        v_spedis = spedis(ssn, ssn1);
        v_complev = complev(ssn, ssn1);
        v_compged = compged(ssn, ssn1);
run;

proc rank data=testssn_ out=test ties=condense;
        var v_spedis v_complev v_compged;
        ranks r_spedis r_complev r_compged;
run;

proc sort data=test; by r_spedis r_complev; run;
```

The table below reveals the rank order of each algorithm for each operation. In comparing string '123456789' with a null value and a no match (all a-z characters), COMPGED clearly assesses that a null value is much more distant than a string of not a single character matching whereas SPEDIS and COMPLEV indicate comparisons of null and no match as equivalent. That is, the distance from '123456789' and a null value and a non-matching value is the same. Do we agree that the cost of a null value of 1800 compared to a cost of 1000 of no match is plausible or should we believe they are equally divergent?

With this rudimentary table, we recognize the hierarchical and subjective properties of each algorithm.

Rank			Value			Operation
r_spedis	r_complev	r_compged	v_spedis	v_complev	v_compged	
1	1	1	0	0	0	match
2	2	3	3	1	50	insert last
3	3	2	5	2	20	swap first
3	3	2	5	2	20	swap last
4	2	4	11	1	100	insert middle
4	2	4	11	1	100	double first
4	2	4	11	1	100	double middle
5	2	5	22	1	200	insert first
6	4	6	100	8	900	reverse
7	5	8	111	9	1800	null
7	5	7	111	9	1000	no match

SUMMARY

For conditional classes, model complexity implies inexact class separability. In the area of text recognition, it is axiomatic that the number of names must be greater than the number of variations of the algorithm. For example, how much information is lost If one attempts to account for all features of French names (or Portuguese)? How well does the algorithm of French names work against novel or unforeseen cases? Hypothesize for a moment that an algorithm is strong: for a given subset of the data, how well does the model converge on the unprecedented and the unexplained?

Hierarchical rules are descriptive of a conditional class object as well. For example, Russell's phonetic algorithm consists of a letter and three numbers. We may question the logic if the data object has materially changed over time. How well has the U.S. phonetic system developed by Russell over 90 years ago endured? Thus, an algorithm may be subject to improvement, validation, and critical argument, despite enduring features and claims of accuracy and clarity. The Daitch-Motokoff algorithm, possessing similar properties as Russell's invention, faces similar scrutiny.

In their seminal work on pattern recognition, Duda and Hart emphasize the breadth of this topic. They posit that the concept of pattern recognition transcends text recognition to genetic sequencing, biological sensory mechanisms, machine functions, and other complex systems of classification. We can step back and appreciate the immensity, subjectivity and approximation of these classification systems and our dependence on these systems. Although the algorithms discussed, SAS and non-SAS related, inform just a small segment of pattern classification systems, they illustrate a way of thinking of these idiosyncratic data objects. Their common thread is the absence of an exact class and clear membership.

ACKNOWLEDGMENTS

I wish to thank colleagues Russ Phanvong and Yongsuk Park for their comments and suggestions. I am also grateful to Suresh Baral, my manager, for his support and confidence.

BIOGRAPHY

Scott currently is SVP in Credit Risk (Modeling) at Bank of America in Calabasas, California. Previously, he held similar roles at Washington Mutual and HSBC, completed post-graduate studies at Reed College, and taught philosophy at the University of British Columbia. Scott is SAS certified in Advanced Programming and Predictive Modeling.

REFERENCES

Duda, R. O. and Hart, P. E (2001). Pattern Classification (2nd Edition). John Wiley & Sons.

Fan, Zizhong. “Matching Character Variables by Sound: A closer look at SOUNDEX function and Sounds-Like Operator (=*)”, Westat, Rockville, MD. NESUG 16.

Gershteyn, Yefim. “Use of SPEDIS Function in Finding Specific Values”, SCIREX Corporation, Chicago, IL. SUGI 25, Paper 86-25.

Knuth, Donald (1998). The Art of Computer Programming, Volume 4. Addison-Wesley.

Motokoff, Gary (2010). “Soundexing and Genealogy”. http://www.avotaynu.com/soundex.htm

SAS(R) 9.2 Language Reference: Dictionary, Third Edition.
http://support.sas.com/documentation/cdl/en/lrdict/63026/HTML/default/viewer.htm#a000245949.htm

Snow, John (1936). “On the Mode of Communication of Cholera”. Commonwealth Fund: New York.

CONTACT INFORMATION

Scott Hanson, PhD
Bank of America
Mail Code: CA7-910-02-27
4500 Park Granada
Calabasas CA 92032

Phone: 1.213.345.8805

Email: scott.p.hanson@bankofamerica.com