Logical Analysis of Data in the TREC-9 Filtering Track

Endre Boros[a], Paul B. Kantor[a,b] and David J. Neu[a]
[a]RUTCOR, Rutgers University
[b]SCILS, Rutgers University
{boros,neu}@rutcor.rutgers.edu, kantor@scils.rutgers.edu

Abstract

In the TREC-9 adaptive filtering and routing sub-tasks of the filtering track we continued to explore utilizing the Logical Analysis of Data (LAD) machine learning methodology to develop Boolean expressions that can be utilized as "rules" for characterizing relevant and irrelevant documents.

1 Logical Analysis of Data

In TREC-9, we continue to view the filtering and adaptive tracks as classification problems which can be approached via machine learning techniques. Specifically, we experiment with a machine learning methodology called Logical Analysis of Data (LAD) which was developed at the Rutgers Center for Operations Research (RUTCOR) at Rutgers University [6], [2], [1], [3], [4], [5]. We begin by providing a brief overview of LAD and then discuss how LAD was adapted for utilization at TREC-9.

LAD accepts as input a training set, each element of which is known to be either a *positive* instance or a *negative* instance. In the filtering and adaptive tracks, the positive instances correspond to *relevant* documents and the negative instances correspond to *irrelevant* documents. We shall refer to the set of positive training instances as T, the set of negative training instances as F, and assume that $T \cap F = \varnothing$. Each element of $T \cup F$ is a Boolean vector $\mathbf{x} = (x_1, \ldots, x_n)$, in which each x_i is referred to as a *feature* or an *attribute* and

$$x_i = \begin{cases} 1 & \text{if the } i^{th} \text{ feature is true,} \\ 0 & \text{if the } i^{th} \text{ feature is false.} \end{cases}$$

It is well known that a *Boolean function*, that is, a mapping $f : \mathcal{B}^n \to \mathcal{B}$, where $\mathcal{B} = \{0, 1\}$, can be represented by a $2^n \times (n + 1)$ table, with the first n columns representing a point in $\mathcal{B}^n$ and the $n + 1$ column representing the value of the function at each point. Similarly, the set $T \cup F$ can be seen to represent a *partial Boolean function* (pBF), that is, a Boolean function in which the value at some points in $\mathcal{B}^n$ is undefined. We shall refer this pBF as f. An *extension*

of a pBF, is a (fully defined) Boolean function which "agrees" with the pBF at every point at which the pBF is defined. An important extension is the Boolean function f^+ which is defined as

$$f^+(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{x} \notin F, \\ 0 & \text{if } \mathbf{x} \in F. \end{cases}$$

In general, machine learning methodologies can be viewed as solving three sub-problems:

> *Feature selection* involves identifying a "small" set of attributes or features which is sufficient for differentiating instances in T from those in F.
>
> *Rule generation* involves using selected features to create elementary Boolean conjunctions which can be used as "rules" to differentiate instances in T from those in F.
>
> *Rule selection* involves identifying the "best" rules and then combining them into a single rule which can be used to differentiate instances in T from those in F.

In LAD, these three sub-problems are called *support set generation*, *pattern generation* and *theory generation*. A set of features is a *support set* if the elements of T and F can be separated (i.e. distinguished) by using only the features in the set [3]. Therefore a support set is a set of essential features. A *minimal* support set is a support set, which does not contain any other support set.

A *positive pattern* [6] is a elementary Boolean conjunction C such that

1. $C(\mathbf{x}) = 0$ for every $\mathbf{x} \in F$
2. $C(\mathbf{x}) = 1$ for at least one vector $\mathbf{x} \in T$

A pattern is called a *positive prime pattern* if for every conjunction $C^{'}$ obtained by dropping a feature from C, there exists a vector $\mathbf{x} \in F$ such that $C^{'}(\mathbf{x}) = 1$. It can be seen that every positive prime pattern is a prime implicant[1] of the extension f^+. The coverage of a positive pattern C is

$$\frac{|\{\mathbf{x} \in T : C(\mathbf{x}) = 1\}| + |F|}{|T \cup F|}$$

That is, it is the proportion of points in the training set which the positive pattern C correctly classifies. The above formula uses the fact that a positive pattern by definition will always correctly classify all points in F. For example, if $|T| = 10$, $|F| = 15$ and if a positive pattern correctly classifies 5 points in T, then its coverage is $20/25$. It should be noted that the concepts for negative patterns are defined similarly by interchanging T and F in the discussion above.

[1] An elementary conjunction C is an *implicant* of a Boolean function f if $C(\mathbf{x}) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathcal{B}^n$, i.e. $C(\mathbf{x}) = 1 \Rightarrow f(\mathbf{x}) = 1$ for all $\mathbf{x} \in \mathcal{B}^n$. An implicant, C is said to be *prime* if dropping any literal causes it not to be an implicant.

A *theory* is a Boolean function which agrees with each $\mathbf{x} \in T \cup F$. For example, the extension f^+ is a theory. It can be seen that a theory can be expressed as a *DNF* in which each term is a positive pattern [6].

Example [2]

Point	x_1	x_2	x_3	f
a	1	1	0	1
b	0	1	0	1
c	1	0	1	1
d	1	0	0	0
e	0	0	1	0
f	0	0	0	0

It can be seen that x_1, x_2 and x_3 form a minimal support set since points c and e differ only in x_1, points a and d differ only in x_2, and points c and d differ only in x_3. It can also be seen that the positive patterns are x_2 and x_1x_3 and that the negative patterns are $\overline{x}_1\overline{x}_2$, $\overline{x}_1x_3$ and $\overline{x}_2\overline{x}_3$. Finally, $x_2 \vee x_1x_3$ is a theory.

LAD seeks to find the optimal (many different criteria for optimality may be employed) support sets, patterns and theories via combinatorial optimization. For example, we may be interested in finding all minimal support sets, all prime patterns and all minimal (i.e. containing no redundant terms) theories. This approach, however, involves solving problems which are extremely computationally expensive. For example, finding all minimal support sets involves solving a Set Covering Problem (SCP) which is known to be NP-hard [7].

2 Implementing LAD

This section discusses an implementation of LAD which solves variants of the support set, pattern and theory generation problems discussed in Section 1. These variants can be solved in polynomial time and are therefore practical for use in solving large machine learning problems. This implementation was developed by one of the authors (Boros), in the Perl programming language and preparations are being made for making this code publicly available.

It should be mentioned that the actual training set which is presented to the LAD algorithm is a set of real vectors where each vector represents a document and each component of a vector represents the relative frequency of a term in that document. This document representation is prepared by running a Perl based indexer which does not do stemming, but does remove the stopwords specified in the Cornell SMART stopword list [1]. As will be seen, these real vectors will be converted to binary vectors during support generation.

We define the value of the Boolean variable x_i for document j as follows

[1] The Cornell SMART stopword list can be found at `ftp://ftp.cs.cornell.edu/pub/smart/english.stop`

$$x_{ij} = \begin{cases} 1 & \text{if } y_{ij} \geq t_i, \\ 0 & \text{if } y_{ij} < t_i. \end{cases}$$

where y_{ij} is the relative frequency, in the j^{th} document, of a term associated with variable x_i, and t_i is a scalar which is calculated by the support set generation algorithm.

The *support set generation* algorithm does not attempt to generate all minimal support sets, but instead employs a greedy heuristic which tries to find a set of binary variables x_i, of minimum cardinality, such that the minimum Hamming distance between any two vectors $a \in T$ and $b \in F$ is at least k, where k is a parameter of the algorithm. Increasing the value of k will result in more variables entering the support set, since it takes more variables to "push the sets farther apart". Sometimes it is not possible to separate T and F by a Hamming distance of at least k. In these cases we utilize a lexicographic rule to decide which variables enter the support set.

The *pattern generation* algorithm does not try to find all minimal patterns, but rather exhaustively generates all patterns which have a coverage of at least c and which have degree less than d, where c and d are parameters of the algorithm. Since the expected number of these patterns in randomly generated training sets is polynomial, this algorithm can be implemented to run in expected polynomial-time [5]. In cases where either no positive or no negative patterns were generated with the initial settings for c and d, the value of c was iteratively lowered until at least one positive and at least one negative pattern was generated.

While the Perl LAD package does support *theory generation*, we do not use this feature in our TREC-9 runs. Instead we use all the patterns to calculate a real-valued score σ as follows

$$\sigma = \sum_{i \in P} c(i) - \sum_{j \in N} c(j)$$

where P is a set of positive patterns, N is a set of negative patterns, and $c(i)$ denotes the coverage of the i^{th} pattern.

For the routing runs, we simply calculate σ for every document in the test set and present the first 1000 documents ranked by σ. For the adaptive filtering runs we use σ to create a linear classifier with the rule that a document in the test set is retrieved only if $\sigma > 0$. We did experiment with computing the threshold in a less arbitrary manner. Specifically, for a given topic, we computed σ for every document in the training set and then found a threshold value τ which correctly classifies the maximum number of training set documents using the following rule: documents for which $\sigma \geq \tau$ are classified as relevant and those for which $\sigma < \tau$ are classified as irrelevant. However, we found that simply letting $\tau = 0$ generally resulted in better performance.

3 Results

For both the routing and the adaptive filtering runs we set the parameters discussed in Section 2 as follows:

- desired Hamming distance between T and $F = 5$
- maximum degree of positive and negative patterns = 5
- minimum fraction of T covered by a positive pattern and minimum fraction of F covered by negative pattern = 0.95

This resulted in

- about 5 to 75 terms in the support set for each topic
- a large number of degree 1, 2 and 3 patterns even though the maximum degree is set to 4 or 5
- about 2 to 150 patterns generated for each topic

The ability to generate patterns with such a high coverage is uncommon when LAD is applied to data sets other than information retrieval ones. In addition, anecdotal evidence supports the idea that the support set is a short set of terms which are "relevant" to the topic at hand.

3.1 Routing Sub-Task

In the TREC-9 the routing sub-task we submitted two types of runs for the OHSU topics and the same two types of runs for the MESH-SAMPLE topics. The two types differed in that the antrpohsu00, antrpms00 runs only used the coverage of positive patterns in the computation of σ while the antrpnohsu00, antrpnms00 runs used the coverage of both positive and negative patterns. The following tables present score statistics for each run. We list the mean, standard deviation, maximum, median and minimum of our scores for each run. We also list the number of topics in which our score was greater or equal to the track median score, and the number of times we achieved the maximum score of the track.

Run:	antrpohsu00
Topic Name:	OHSU
Number Topics:	63
Mean	0.099
Standard Deviation	0.137
Max	0.648
Median	0.054
Min	0.000
# times $\geq$ median	6
# times = max	1

Run:	antrpnohsu00
Topic Name:	OHSU
Number Topics:	63
Mean	0.177
Standard Deviation	0.160
Max	0.690
Median	0.132
Min	0.000
# times $\geq$ median	20
# times = max	0

Run:	antrpms00
Topic Name:	MESH-SAMPLE
Number Topics:	500
Mean	0.078
Standard Deviation	0.156
Max	0.856
Median	0.004
Min	0.000
# times $\geq$ median	12
# times = max	1

Run:	antrpnms00
Topic Name:	MESH-SAMPLE
Number Topics:	500
Mean	0.158
Standard Deviation	0.198
Max	0.855
Median	0.065
Min	0.000
# times $\geq$ median	93
# times = max	12

3.2 Adaptive Filtering Sub-Task

In the TREC-9 the adaptive filtering sub-task we submitted two runs for the OHSU topics. Both the antadapt001 and antadapt002 runs used ten copies of the topic as positive training documents in addition to the two initial training documents provided. The antadapt001 and antadapt002 differed in that antadapt002 added some randomly selected documents from the training set which to act as negative training documents.

The adaptive strategy employed was as follows:

- a document is retrieved only when $\sigma > 0$
- all retrieved documents are added to the training set

- support set and pattern generation are only rerun when we "make a mistake", that is, when an irrelevant document is retrieved

Run:	antadapt001
Topic Name:	OHSU
Number Topics:	63
Measure:	T9P
Mean	0.088
Standard Deviation	0.132
Max	0.580
Median	0.040
Min	0.000
# times $\geq$ median	8
# times = max	1

Run:	antadapt002
Topic Name:	OHSU
Number Topics:	63
Measure:	T9P
Mean	0.102
Standard Deviation	0.150
Max	0.791
Median	0.056
Min	0.000
# times $\geq$ median	9

Run:	antadapt001
Topic Name:	OHSU
Number Topics:	63
Measure:	T9U
Mean	-32.270
Standard Deviation	45.406
Max	50.00
Median	-9.00
Min	-100.00
# times $\geq$ median	26
# times = max	6

Run:	antadapt002
Topic Name:	OHSU
Number Topics:	63
Measure:	T9U
hline Mean	-43.571
Standard Deviation	53.405
Max	118.00
Median	-22.00
Min	-100.00
# times $\geq$ median	24
# times = max	4

4 Conclusion

Our best performance in both the routing and the adaptive filtering sub-tasks was substantially below the hypothetical "median system". In our best routing run on the OHSU topic set, we achieved the median about one-third of the time, while on the MESH-SAMPLE topic set we only achieve the median about one-fifth of the time. Although we did not tune our algorithm for any particular utility function, it appears that that it is more nearly tuned to T9U than T9P. Using T9U, we achieved the median about 38% of the time on the OHSU topic set. It is also interesting to note that in the adaptive filtering run, the addition of randomly selected documents for use as a negative training set did not substantially alter the performance. Our results in TREC-9 imply that more experimentation on how to utilize LAD for information retrieval is required. One conclusion is that using both positive and negative patterns resulted in better performance.

Ideas which might worth considering include

1. the use of support set terms for query expansion
2. incorporating stemming into our indexer
3. utilization of other term weighting schemes
4. development of additional methods for utilizing patterns in a document scoring function
5. development of more intelligent calculations of the threshold τ used in the linear classifier used in the adaptive filtering runs

References

[1] Endre Boros, Peter L. Hammer, Toshihide Ibaraki, and Alexander Kogan. Logical analysis of numerical data. *Mathematical Programming*, 79:163–190, 1997.

[2] Endre Boros, Peter L. Hammer, Toshihide Ibaraki, Alexander Kogan, Eddy Mayoraz, and Ilya Muchnik. An implementation of logical analysis of data. *IEEE Transactions on Knowledge and Data Engineering*, in press.

[3] Endre Boros, Takashi Horiyama, Toshihide Ibaraki, Kazuhisa Makino, and Mutsunori Yagiura. Finding small sets of essential attributes in binary data. *To appear in the Second International Conference on Intelligent Data Engineering and Automated Learning*, Hong Kong, December 13-15, 2000.

[4] Endre Boros, Paul B. Kantor, and Dave J. Neu. Pheromonic representation of user quests by digital structures. *In Proceedings of the 62nd Annual Meeting of the American Society for Information Science*, 36:633–642, 1999.

[5] Endre Boros, Lijie Shi, and Mutsunori Yagiura. Generating all good patterns in polynomial expected time. *To appear in the 6th International Symposium of Artificial Intelligence and Mathematics, Fort Lauderdale, FL*, pages 633–642, January 5-7, 2000.

[6] Yves Crama, Peter L. Hammer, and Toshihide Ibaraki. Cause-effect relationships and partially defined boolean functions. *Annals of Operations Research*, 16:299–326, 1988.

[7] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeeman and Company, 1979.

[8] Tom M. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[9] J. Ross Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufamnn, 1993.