Inferring Patterns of Neural Response with STL Learning

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Robert Michaels
Spring, 2020

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature _____ Date _____
Robert Michaels

Approved _____ Date _____
Lu Feng, Department of Computer Science

Acknowledgments

To Dr. Lu Feng for giving me the opportunity to work on this project.

To Josephine Lamp, who this project relied on for success. In addition to the obvious such as her existing STL learning tool and extensive knowledge of STL, her direction and guidance kept me moving forward and taught me much about the procedure and format for best expressing my results.

Introduction

Signal temporal logic (STL) is a formal specification language which incorporates symbolic logic and temporal constraints to determine whether a changing system of various metrics or conditions satisfies certain criteria. As opposed to standard symbolic logic which can only determine if a set of variables satisfies logical formulae at a particular state, temporal logic includes operators accounting for time-based criteria such as variables eventually reaching some satisfactory condition, always satisfying some condition within a specified time frame, and so on. Signal temporal logic in particular employs the usage of temporal logic formulae with a stream of variable data over time which can be either analyzed after data has been collected, or run in-line with live data to see if a system fits the criteria in real time. Additionally, instead of a qualitative satisfaction requirement, STL formulae can be implemented in a quantitative manner with a robustness score which indicates to what degree a set of variables satisfy the parameters of the STL system (Bartocci, 2018).

STL has potential to be of particular use within the field of cyber-physical systems. Since this type of system is one which provides several types of biometric data over time, STL can be used to create monitoring systems in order to classify when metrics would indicate some sort of overall state outside of normal parameters. The group I am working with has done previous research on incorporating STL with data monitored from diabetic patients in order to classify anomalous changes in blood glucose levels or detect when a patient may have forgotten to record a meal in relevant monitoring software (Young, 2018).

The STL learning algorithm used by Young et al has the potential for novel applications in analysis of neurological data. For this particular project, I used neurological data provided through UC Berkeley's Collaborative Research in Computational Neuroscience (CRCNS) program. Specifically, I analyzed the pvc-3 data set, which is based upon recording neuronal

activity in the primary visual cortex in response to a variety of stimuli (Blanche, 2009). Since this data tracks individual neuronal activity in a sensory region of the brain, STL learning has the potential to map incoming stimuli to the activity of a particular neuron or combinations of multiple neurons firing in groups.

As a brief overview, the concept of receptive fields has been used for decades in neurophysiology to make this sort of sensory stimulus response mapping on the neuronal level (Ringach, 2004). These receptive fields in the context of visual stimuli include perception of particular shapes, as well as perception of patterns in motion. There have been studies performed confirming that the visual cortex of various animals does, in fact, follow these sort of direct response to geometric visual perceptions of their environment (Donner, 1981) (Camarda, 1976). While these sort of correlations have traditionally been made using statistical inference on the level of individual neurons, STL learning provides the potential for more complex associations and combinations of neural activity from a particular stimulus to be expressed.

Since stimuli response is a complex neuronal activity, it's possible that there are discernable patterns within responses of multiple neurons associated with some stimulus. As opposed to conventional statistical inference, STL has the potential to represent more robust patterns of response, particularly by incorporating more temporal aspects of response patterns, as well as generating more distinct thresholds for neuronal activity.

STL Learning

STL provides a framework for satisfiability conditions given a signal, but deriving STL formulae is not necessarily a straightforward affair. Instead of trying to assess data for patterns, STL learning provides an algorithmic approach for such that a machine would be able to synthesize STL formulae and their parameters.

Nenzi et al provide such an approach for implementing machine learning in STL formula synthesis (2018). The algorithm works by generating formula structures followed by best fitting parameters to the data set at hand, then taking the most robust formulae for classification and mutating/combining these to develop a new generation of formulae. This pattern can be repeated any number of times with the intention of developing more accurate formulae at each generation.

Practically, this involves dividing a group of temporal signals into a labeled set as to whether a given set of variables describes a particular state of data. In the example of the diabetic assessment application previously mentioned, this would be a physiological state of components such as blood glucose at odds with reported data of meal intake. This step is not automated and is manually performed by the data analyst who knows about the actual state of the system (in the Young paper on diabetic data, this would be knowing an individual failed to record a meal in the diabetic monitoring case).

Formula generation is initially performed by randomly assigning conjunctions of one or more variables using STL formulae (ie. glucose levels being above a certain level within some future time given that a certain amount of carbohydrates have been ingested at a time). These formulae then have their robustness values (the extent to which they discriminate between the labeled cases and other states) calculated and the best formulae are chosen for inclusion in the next generation of the algorithm. The STL properties of included formulae are then used with some degree of random change to create formulae for the next generation. These new formulae are then assessed for robustness and mutated/combined for however many generations are specified.

Methodology

The pvc-3 data I analyzed was based on the “drifting bar” stimulus case. For this

experiment, an anesthetized cat watched a screen with a line moving perpendicular to its longitudinal orientation (Figure 1, Blanche 2009). There were eighteen possible permutations of orientations, with each orientation being at a 20 degree differential in movement direction from the other stimuli. Data was recorded from implanted electrodes within 10 neurons within the primary visual cortex of the cat over the course of nine minutes (540 seconds) of sequential 5 second stimuli. The orientations over the course of this recording were randomly distributed with each individual stimulus being presented a total of six times.

Figure 1: Drifting Bar Example

Data was stored in a specific file for each recorded neuron as a series of timestamps referring to times when the neuron fired within the recorded period. Additionally, there was another file representing the timestamps at which each stimulus was presented, along with the stimulus type at the given time point.

I worked with Josephine Lamp's STL Learning tool (Lamp, 2019). Because this tool requires data to be formatted as a series of variables over delineated time points, I needed to modify the initial data from its timestamp based format. In order to accomplish this, I iterated through each neuron's firing times and divided them into chunks of 100 ms. The end result was 54000 rows of time data for each neuron, where each row corresponded to the number of neuronal spikes within the next 100 ms. All of these neuron data were concatenated into a single

spreadsheet with one column for time and another ten columns corresponding to each neuron's activity. Finally, this data needed a series of labels for STL classification purposes. I generated 18 different final tables, with each state satisfaction labeled with times corresponding to one of the stimuli orientations.

For the STL learning tool itself, I kept most of the parameters the same as default. For the slice size (the number of time points in each block used for comparing temporal differences in STL classification), I used 5 points per slice, which translates to 20 slices for each particular stimulus instance. Finally, I capped the variable parameters at the maximum number of spikes per 100 ms for each particular neuron. I proceeded to run this through with two generations of formula permutations to get the final results for each stimulus.

Results/Analysis

Initial results provided some promising formulae, but closer investigation led to some clear limitations in their utility. Though each stimulus case ended up with a number of formulae with good robustness, discrimination scores, and low misclassification rates (Tables 1 and 2), manual inspection led to the revelation that these results aren't doing a particularly good job of describing any sort of signature behavior for any individual stimulus. The proceeding future work section will go more into detail on how these results may be improved by modifying the manner in which data is handled, particularly in labeling differences and filtering certain rules.

To begin, there seem to be many rules which are descriptive of the entirety of the data set as opposed to classifying a particular stimulus. Due to the fact that each stimulus is only accounting for one out of eighteen data points for the set as a whole, rules which classify the entirety of the data will appear to have better classification than they actually provide. For instance, Table 2 shows some of the best positive rules for a particular stimulus based on

misclassification rate. As is apparent, these rules do an excellent job of classifying the set as a whole, with the true negative classifying the majority of the data set accurately 17/18ths of the time. That being said, since we are trying to create descriptive rules for a particular stimulus, the fact that a false negative happens every time for our desired stimulus is an issue to say the least. These sort of rules also show up in the negative classifying rules, with 17/18ths of the set receiving a true positive, and the relevant stimulus being classified as all false positives.

Additionally, as rules begin to successfully differentiate particular stimuli classifications (see Table 3), the false positive rate becomes much higher. There may be several ways to account for this. As mentioned previously, visual receptive fields are frequently tuned to particular shapes (ie. vertical or horizontally aligned shapes). There may be some sort of overlap in neural activity between stimuli on the same axis.

Certain neurons also appear to be over-represented in the results. In particular, the t27 neuron shows up in a majority of rules for all of the different stimuli. The reason for this is likely due to it having a much higher range of activity than any other neuron in the data set, with a maximum firing rate of ~200 times per second, where the average neuron caps out at ~50 times per second. Due to the robustness factor playing a large role in the rule selection part of the genetic STL algorithm, it's likely that the rules involving t27 are inherently more robust. This could potentially be ameliorated by normalizing the rate on a per-neuron basis before feeding the data into the STL learning algorithm. This is far from the only modification that may improve performance of the algorithm on this data set, however.

Future Work

As far as continuing development on the application of STL learning to neurological processes, there are several directions to take. For the data set analyzed, there are several

modifications which could be made to the existing processing to see if more compelling results could be produced. Additionally, this particular data set may not be well suited to STL learning due to a number of factors, and finding a set more amenable to producing valuable results via STL learning.

As far as improving results on the current data set, filtering for results which are descriptive of the data set as a whole is an intuitive first step. These sort of rules can be filtered out by seeing if a significant portion of the results falls into this paradigm for their respective classes. These filtered rules could also potentially be fed back into the genetic learning portion of the algorithm to create another subsequent generation of rules.

As mentioned previously, there may be some sort of improvement from labeling similar stimuli in groups as opposed to having individual stimuli for each label. Since receptive fields are often in particular elongated shapes on some orientation, it's worth investigating whether there are better results with a positive label applied to stimuli moving in opposite directions. In a similar vein, there is only a 20 degree difference in orientation of each stimulus. There may be some similarity in nearby angles as well. Potentially labeling an arc of stimuli as opposed to a singular angle (ie. 0, 20, and 40 degrees all as a positive classifier) may provide more compelling rules.

Finally, this data set may just not be well suited for STL learning to determine neural activity patterns. First of all, there are only 10 neurons being recorded in this instance. In reality, there are millions of neurons in the primary visual cortex, and statistically speaking it is incredibly unlikely that this particular data set contains enough data points to create correlative mappings of neural activity for a certain stimulus. While data from the primary visual cortex is likely well suited for this type of application given the more direct correlation between sensory

input and this region of the brain, there likely needs to be a greater extent of recorded neural activity for useful results. Future work would prioritize finding a data set with a large amount of recorded neurons.

Another point of difficulty may be from the nature of the stimuli in this particular data. This experiment had a series of sequential stimuli with no sort of intermediate rest period. As such, there may just be too much of the response associated with any sort of stimulus as opposed to individual stimuli with minute differences. For future work, it may be potentially useful to find sensory data which is of a more binary nature.

Overall, STL learning may have some value for expressing neuronal systems, but the applications may be limited to certain types of response behavior. For future work, I will be concentrating on finding more extensive data sets (more neurons recorded) which map some sort of stimulus-response pattern in a sensory region of the brain. This will likely produce more useful rules for expressing neuronal responses.

G[0,3](t04 <= 3.936 & t27 <= 3.704) [Discrimination Score: 0.431; Robustness + : 3.242; Robustness - : 1.646; Percent Class + : 0.646; Percent Class - : 0.354]

((t27 <= 5.360) U[1,2] (t26 <= 0.901)) & t08 <= 3.899 [Discrimination Score: 0.426; Robustness + : 3.810; Robustness - : 3.138; Percent Class + : 0.000; Percent Class - : 1.000]

((G[2,4](t25 <= 5.048)) U[0,4] (t27 <= 11.597)) [Discrimination Score: 0.419; Robustness + : 11.488; Robustness - : 10.375; Percent Class + : 0.798; Percent Class - : 0.202]

((t02 <= 12.570) U[1,4] (t27 <= 1.953)) [Discrimination Score: 0.419; Robustness + : 1.843; Robustness - : 0.731; Percent Class + : 0.675; Percent Class - : 0.325]

G[0,4](t08 <= 9.000 & F[0,0](t27 <= 9.038)) [Discrimination Score: 0.417; Robustness + : 8.017; Robustness - : 6.132; Percent Class + : 0.000; Percent Class - : 1.000]

G[0,4](t04 <= 13.442 & t27 <= 10.427) [Discrimination Score: 0.415; Robustness + : 9.771; Robustness - : 7.993; Percent Class + : 0.771; Percent Class - : 0.229]

G[0,4](t08 <= 2.465 & t27 <= 0.000) [Discrimination Score: 0.412; Robustness + : -0.953; Robustness - : -2.836; Percent Class + : 0.212; Percent Class - : 0.788]

((t27 <= 3.289) U[1,4] (t02 <= 4.769)) [Discrimination Score: 0.412; Robustness + : 2.866; Robustness - : 1.379; Percent Class + : 0.663; Percent Class - : 0.337]

((t08 <= 5.608) U[0,4] (t27 <= 5.517)) [Discrimination Score: 0.411; Robustness + : 5.307; Robustness - : 4.164; Percent Class + : 0.757; Percent Class - : 0.243]

G[0,4](t27 <= 13.884) [Discrimination Score: 0.411; Robustness + : 12.931; Robustness - : 11.052; Percent Class + : 0.791; Percent Class - : 0.209]

G[0,4](t25 < 7.000 & t27 <= 0.000) [Discrimination Score: 0.411; Robustness + : -0.953; Robustness - : -2.832; Percent Class + : 0.216; Percent Class - : 0.784]

Table 1: Example rules with Robustness and Discrimination Scores, Stimulus #13 (240 Degrees)

Rule	TP	TN	FP	FN	Accuracy	MCR
G[0,4](t25 >= 3.750 & t27 <= 0.000)	0	1363	1	80	0.944559944559944	0.055440055440056
G[3,4](((t04 <= 5.991) U[0,1] (t08 >= 7.313))) & t27 <= 3.553	0	1363	1	80	0.944559944559944	0.055440055440056
((t27 >= 19.698) U[1,2] (t27 <= 10.983))	0	1363	1	80	0.944559944559944	0.055440055440056
((!(t08 <= 3.771)) U[2,2] (t27 <= 2.586))	0	1362	2	80	0.943866943866944	0.056133056133056
((t27 <= 2.609) U[1,1] (t02 >= 8.433))	1	1360	4	79	0.943173943173943	0.056826056826057
((t18 > 8.766) U[1,2] (t27 <= 20.062))	8	1349	15	72	0.94040194040194	0.05959805959806
((t10 <= 4.120 & t25 >= 3.125) U[1,2] (t27 <= 16.840))	0	1354	10	80	0.938322938322938	0.061677061677062
((((t10 <= 1.484) U[1,1] (t27 >= 12.395))) U[1,2] (t27 <= 12.383))	0	1353	11	80	0.937629937629938	0.062370062370062

Table 2: Highest Accuracy Rules for Stimulus #13

Rule	TP	TN	FP	FN	Accuracy	MCR
((t18 >= 6.050) U[1,2] (t27 <= 14.984))	8	1337	27	72	0.932085932085932	0.067914067914068
((F[2,4](t18 >= 4.952)) U[1,2] (t27 <= 18.958))	18	1281	83	62	0.9002079002079	0.0997920997921
G[2,2](t18 >= 0.224 & t27 <= 1.322)	25	1238	126	55	0.875259875259875	0.124740124740125
G[0,4](t25 < 7.000 & t27 <= 0.000)	26	1156	208	54	0.819126819126819	0.180873180873181
G[0,4](t18 <= 26.000 & t27 <= 0.000)	26	1155	209	54	0.818433818433818	0.181566181566182
((t18 > 0.000) U[1,1] (t27 <= 20.513))	19	1136	228	61	0.8004158004158	0.1995841995842
G[0,3](t27 <= 0.125)	34	1116	248	46	0.796950796950797	0.203049203049203
G[0,3](G[1,3](t02 <= 9.816 & t27 <= 0.324))	43	1030	334	37	0.743589743589744	0.256410256410256

Table 3: Intermediate Accuracy Rules for Stimulus #13

Works Cited

Bartocci, E., & Falcone, Y. (2018). Specification-Based Monitoring of Cyber-Physical Systems: A Survey on Theory, Tools and Applications. In *Lectures on Runtime Verification* (pp. 135–175). New York, United States: Springer Publishing.

Blanche, Tim (2009). Multi-neuron recordings in primary visual cortex. CRCNS.org. http://dx.doi.org/10.6080/K0MW2F2J

Camarda, Rosolino, and Giacomo Rizzolatti. "Visual Receptive Fields in the Lateral Suprasylvian Area (Clare-Bishop Area) of the Cat." *Brain Research*, vol. 101, no. 3, 1976, pp. 427–443., doi:10.1016/0006-8993(76)90469-8.

Donner, K. "Receptive Fields of Frog Retinal Ganglion Cells: Response Formation and Light-Dark-Adaptation." *The Journal of Physiology*, vol. 319, no. 1, Jan. 1981, pp. 131–142., doi:10.1113/jphysiol.1981.sp013896.

Lamp, J. (2019, October 16). STL Learning. Retrieved April 25, 2020, from https://github.com/jozieLamp/STLlearning

Nenzi, L., Silvetti, S., Bartocci, E., & Bortolussi, L. (2018). A Robust Genetic Algorithm for Learning Temporal Specifications from Data. *Quantitative Evaluation of Systems Lecture Notes in Computer Science*, 323–338. doi: 10.1007/978-3-319-99154-2_20

Ringach, Dario L. "Mapping Receptive Fields in Primary Visual Cortex." *The Journal of Physiology*, vol. 558, no. 3, 2004, pp. 717–728., doi:10.1113/jphysiol.2004.065771.

Young, W., Corbett, J., Gerber, M. S., Patek, S., & Feng, L. (2018). DAMON: A Data Authenticity Monitoring System for Diabetes Management. *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation*. doi: 10.1109/iotdi.2018.00013