CS 199 Quarter Report: Generative Grammar from Internal Fluents

Kang (Frank) Chen
University of California, Los Angeles

Abstract

Fluents [6] are important formulations in robot learning, as they capture the change in abstract spatial and temporal concepts of an object. In this work, we focus only on the internal fluents, such as a robot's arm with 7 degrees of freedom. We extend the work by Liu et al. [2] on learning semantically labeled Task And-Or graphs. Through supervised motion control (i.e. hand-holding), the robot learns a generative model with semantically significant labels to the internal fluents, whereas previously it was applied only to external fluents.

1 Introduction

A robot arm has multiple degrees of freedom. The combination of these degrees of freedom will correspond to a state the robot arm is in. Our goal is to learn the atomic actions of the robot. To achieve this, we first need to understand what constitutes a state change.

We begin by formulating the state change of our robot, Baxter[1], as a change in its fluents; we then visualize this change through a web user interface. Lastly, we cluster the fluent change by implementing a biclustering algorithm that will allow us to learn the atomic actions of the robot, which we can use as input in the STC-AoG for robot learning [1].

The contribution of this work include:

1. Visualizing high dimensional robot trajectories in real-time on a 3-dimensional web user interface.

2. A framework to learn useful atomic actions of a robot: first, learn the joint trajectories with supervised motion control; second, in an unsupervised way, learn the atomic actions.

3. Creating a generative model of these atomic action sequences with natural language labeled sub-goals (non-terminal nodes in the And-Or graph).

2 Related Works

Tu et al. [5] contribute foundational work to the unsupervised structured learning of stochastic And-Or grammars. In this work, we apply their biclustering algorithm to learning a generative model of robot atomic actions. In addition, Xiong et al. [1] extended Tu's formulations to apply a spatial, temporal, and causal And-Or graph (STC-AoG) to model robot behavior. We adopt the STC-AoG model in this study.

In learning from language and video demonstrations, Liu et al. [2] contribute learning grounded task structures, and Tu et al. [5] made advances in video and text parsing for understanding events. In this work, we also jointly parse language and sensory data, heavily inspired by their approach.

3 Background on Fluent Algebra

Fluents in an object represent the changing properties of that object. We represent the world as a state in time called $s^{(t)}$. Formally, a fluent f is a function on a state $s^{(t)}$. We assume there is a large number of fluents, namely N, and index f_i for $i \in 1,...,N$.

A fluent vector $F = (f_1,...,f_N)$ on a state $s^{(t)}$ is a vector of all fluents f_i stacked together. A fluent change ΔF represents the vector of all fluent changes Δf_i on a given state $s^{(t)}$, or, more concisely, $F(s^{(t)}) \rightarrow F_1(s^{(t+1)})$ [6].

We partition fluents into two categories: internal and external. Internal fluents are defined as fluents that can be directly controlled by the robot, such as the joint po-

sitions of its arms. The other type of fluents are called external, such as the shape of a shirt on a table, which are properties the robot cannot directly control.

4 Experiment Formulation

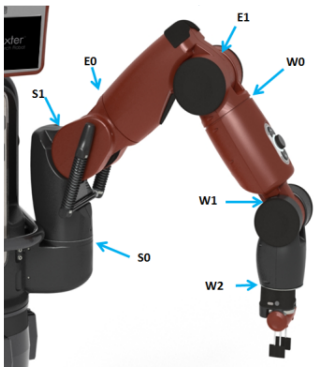


Figure 1: Baxter's 7 degrees of freedom.

We can think of a robot's arm as a vector of the degree of freedom. The unique permutations of each degree of freedom constitutes a state for the robot's arm.

4.1 Defining Raw Data

As shown in Figure 1, Baxter's arm can be formulated as fluent vector of internal fluents $F = [s_0, s_1, e_0, e_1, w_0, w_1, w_2]$. As the arm performs an action, such as raising itself from point A to point B, resulting in an increase in potential energy, its fluent vector F will change.

4.2 Collecting Raw Data

The robot communicates its raw joint data using the Robot Operating System (ROS)[2] interface. We use a web server called Flask[3] in order to communicate with ROS to poll data at 1 Hz. The data is fed to the front-end visualization platform.

Below are examples of data points contingent in time when hand-holding the robot to raise its left arm.

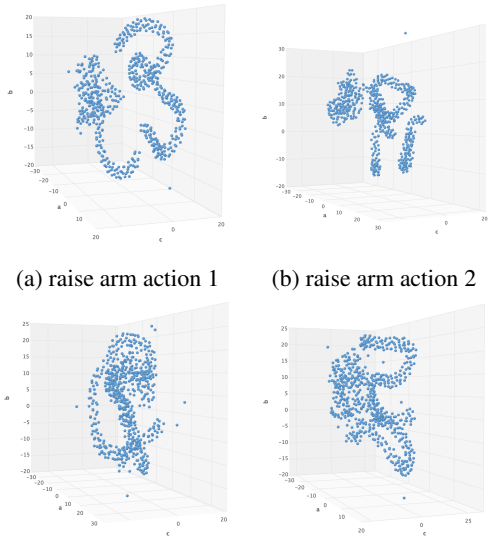
s_0	s_1	e_0	e_1	w_0	w_1	w_2
-0.32	-1.35	0.01	1.47	-0.15	1.33	0.33
-0.32	-1.34	0.01	1.45	-0.15	1.34	0.33
-0.33	-1.34	0.01	1.44	-0.15	1.35	0.33

4.3 Processing and Visualizing Data

We attempt to visualize the internal fluents, but it's a challenge to visualize a 7-dimensional vector, so we employ scikit-learn's t-distributed Stochastic Neighbor Embedding algorithm[4] to reduce the dimensionality of our fluent vector to 3 dimensions.

We then construct an interactive 3D plot using Highcharts[5] to visualize our robot fluents. As the robot moves in the real world, Highcharts shows a 3D point drifting in a vector space. Moreover, the communication is two-way: we can use Highcharts as a controller for the robot.

As shown in Figure 2 below, the robot's states have correlating patterns. We believe these patterns have compositionality and variation between examples; therefore, we would like to cluster these fluent changes in order to better formulate grammars to describe the action.



(a) raise arm action 1 (b) raise arm action 2

(c) raise arm action 3 (d) raise arm action 4

Figure 2: 3-D fluent visualizations of different robot raise arm actions.

4.4 Analyzing Fluent Changes

We would like to cluster the change of fluents, not the fluents themselves. To do so, we must decide how far back in time to look in order to take the difference. We call this difference the skipfactor.

We attempt to cluster our fluent changes in order to

generate a grammar from our raw data. We use the k-Means Clustering algorithm[6] to cluster our robot arm data into k components.

Formally, given a set of observations $(\Delta F_1, \Delta F_1, ..., \Delta F_n)$, we want to partition the n observations into $k\,(\leq n)$ sets $S = \{S_1, S_2, ..., S_n\}$ so as to minimize the within-cluster sum of squares (WCSS). In other words, we want to find:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} \|\Delta F - \mu_i\|^2$$

where μ_i is the mean of points in S_i, and k is the number of clusters.

k-Means Clustering creates k centroids, and each is 7-dimensional. We want to find the centroid that is nearest to each row of data in the ΔF table by looping through the k candidates. We want to create a $n \times 1$ vector containing the cluster classification for each row of the $n \times 7$ ΔF vector. In other words, we want to determine which centroid in S each of the rows in our ΔF falls into.

4.4.1 Example

This is our original dataset:

```
[ row1,
  row2,
  ...
  rowN ]
```

After running k-means (with k = 10), we will get 10 centroids.

```
centroid1, ..., centroid10
```

We then look back at the original dataset and find the corresponding centroid by comparing (row10-row1) with the 10 possible centroids, and choosing the one with lowest distance.

```
[ row1 -> centroid4,
  ...
  rowN -> centroid1 ]
```

Now, we have our $n \times 1$ vector.

5 Generating Grammar

Tu et al. [5] formulated a learning algorithm, which we can use to generate a stochastic And-Or grammar from a set of raw data samples (e.g. natural language sentences, quantized images, action sequences). The objective function is the posterior probability of the grammar given the unannotated training data:

$$P(G|X) \propto P(G)P(X|G) = \frac{1}{Z} e^{-\alpha\|G\|} \prod_{x_i \in X} P(x_i \mid G)$$

where G is the grammar, $X = x_i$ is the set of training samples, Z is the normalization factor of the prior, α is a constant, and $\|G\|$ is the size of the grammar. We approximate the likelihood $P(x_i|G)$ with the Viterbi likelihood (the probability of the best parse of the data sample x_i). We chose the Viterbi likelihood because it has been shown to lead to better grammar learning results [3, 8].

Using this formulation, we can similarly generate grammar from robot atomic actions, similar to what is shown in Figure 4, which was a sample And-Or graph generated from observing humans demonstrating cloth-folding videos. The energy of the joint parse graph [4] combines the energy terms of each:

$$E_{STC}(pg) = E_S(pg) + E_T(pg) + E_C(pg) + \sum_{r \in R^*_{pg}} E_R(r)$$

5.1 Learning AoG Parameters

Ranking parse graphs and learning values is out of scope for this quarter project, but it is research that is currently in progress. In order to get initial results, we used the following approach:

One way to learn the probability distribution of STC parse graphs $P(pg)$ is using the Minimax Entropy Principle[7]. We assume that the sample mean of statistics $\phi_j(pg)$ should approach the true expectation s_j from observations. The parameters are solved by minimizing the Kullback-Leibler divergence between the observed distribution and the candidate

$$KL(f\|p) = E_f[\log f(pg)] - E_f(\log p(pg))$$

This simplifies to a maximum likelihood estimate, formulated by

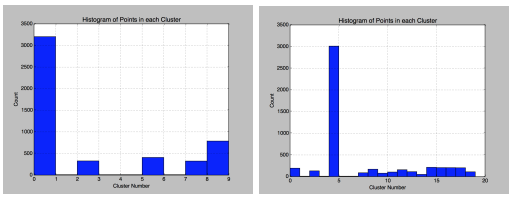
$$p^* = \arg\max_{p \in \Omega} E_f[\log p(pg)] = \arg, \max_{p \in \Omega} \sum_{i=1}^{n} \log p(pg_i) + \varepsilon$$

Iteratively, we choose the statistics $F = \{\phi_1, \phi_2, ...\}$ that minimizes the entropy of the model, and the parameters β that yield maximum entropy.

$$p^* = \arg\min_{F,\beta} \{\max \text{entropy}(p(pg;\theta)\}$$

Effectively, the robot "daydreams" possible probability distributions of parse graphs to converge with observations. During interference, it samples a parse graph to perform the actions[1].

6 Results



(a) skip: 10; cluster: 10 (b) skip: 10; cluster: 20

(c) skip: 20; cluster: 10 (d) skip: 20; cluster: 20

Figure 3: Histogram of varying skipfactors for ΔF and number of clusters

We want to generate histograms of our clustering to observe any patterns with how the robot actions were clustered according to different combinations of skipfactors and cluster number settings.

As shown in Figure 3, the histograms of the clusters show a constant outlier; this can be explained by long periods of no movement. In fact, the fluent visualization indicates that there was no immediate movement in the 3D space initially, most likely due to the experiment set-up process.

The processed data, indicating the action sequence, was surprisingly not noisy for a skipfactor of 10. This shows promising results for feeding the data into the STC-AoG because it means that each time the robot raised its left arm, its raw data is consistently classified into clusters. Using the formulations from Section 5, we can confidently generate a task grammar from the robot atomic action sequences.

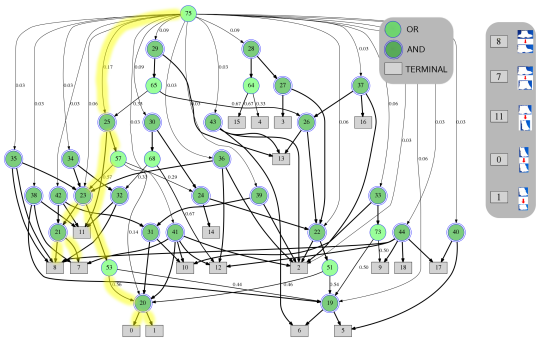


Figure 4: This Temporal-Causal And-Or graph (GTC) represents a generative model of a task-plan after watching 30 video demonstrations of cloth folding. The terminal nodes are atomic actions defined by fluent changes (e.g. terminal node 8 represents the action of folding the right sleeve of a shirt to the middle). The Viterbi parse is highlighted to show the most likely sequence of actions

7 Future Works

This quarter, a goal for robot learning was to generate grammar from raw robot fluent data. As we move forward, we would like to sample an action sequence to solve arbitrary fluent changes. In other words, given a goal represented as a fluent change, our model should be able to generate a sequence of pose and orientation parameters to be run on a robot platform.

Additionally, we plan to conduct experiments to test the robustness of our generated STC-AoG, and create an interpreter that can map the STC-AoG to a probablistic programming language, written in a functional language such as Haskell or OCaml. Lastly, we plan to use this STC-AoG to teach the robot how to fold clothes from raw atomic actions. This will simulate the robot's ability to learn from a low-level reasoning to a high level reasoning (ex. atomic actions such as raise arm, pinch etc. to folding clothes).

8 Acknowledgements

I would like to thank PhD student Nishant Shukla and Prof. Song-Chun Zhu for their mentorship and guidance.

References

[1] CAIMING XIONG, NISHANT SHUKLA, W. X. S.-C. Z. Robot Learning with a Spatial, Temporal, and Causal And-Or Graph. *Int'l Conference on Robotics and Automation (ICRA)* (2016).

[2] CHANGSONG LIU, SHAOHUA YANG, S. S.-S.-N. S. Y. H.-S.-C. Z. J. Y. C. Jointly Learning Grounded Task Structures from Language Instruction and Visual Demonstration. *Conference on*

Empirical Methods in Natural Language Processing (EMNLP) (2016).

[3] HOIFUNG POON, P. D. Sum-Product Networks: A New Deep Architecture. *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence (UAI)* (2011).

[4] KEWEI TU, MENG MENG, M. W. L.-T. E. C. S.-C. Z. Joint Video and Text Parsing for Understanding Events and Answering Queries. *MultiMedia, IEEE* (2014).

[5] KEWEI TU, MARIA PAVLOVSKAIA, S.-C. Z. Unsupervised Structure Learning of Stochastic And-Or Grammars. *Advances in Neural Information Processing Systems 26 (NIPS 2013)* (2013).

[6] SHUKLA, N. Notes on fluent values.

[7] SONG-CHUN ZHU, YING NIAN WU, D. M. Minimax entropy principle and its application to texture modeling. *Neural Computation* (1997).

[8] VALENTIN I. SPITKOVSKY, HIYAN ALSHAWI, D. J. C. D. M. Viterbi Training Improves Unsupervised Dependency Parsing. *Proceedings of the Fourteenth Conference on Computational Natural Language Learning* (2010).

Notes

[1] http://www.rethinkrobotics.com/baxter/
[2] http://www.ros.org/
[3] http://flask.pocoo.org/
[4] http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html
[5] http://www.highcharts.com/
[6] http://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html