Ad-hoc Teamwork: Change-Points Detection

###

March 2, 2018

Abstract

In many setting agents are required to communicate with other agents whom their type they are unfamiliar with. By maintaining beliefs over a set of hypothetical behaviors, or types an agent can improve its interaction with other agents in order to achieve a common goal. Many work already investigated ways to correctly and efficiently calculate the relatively likelihood of an agent's type as well as the values of any bounded continuous parameters within types. Those work all assumed stationarity (i.e., agents' types does not change during time). The assumption of stationarity, even-though suitable for number of realistic scenarios, is a very strong assumption which can not be guarantied in many real world settings. In this work however, we are investigating settings in which an agent can change its type during the mission to be accomplished. Allowing the different agents change their types add complexity to the planning problem as now an agent need to recognize that a change have occur in addition to figuring out what is the new type of the agent it is interact with.

From here on we need to decide what is our direction/s:

- **Change-point detection algorithms Comparison:** We will test two existing change-point detection algorithms (Fearnhead and Liu and ???) to track change-points during a performances of a joint task. We will describe both algorithms, will note their strengths and weaknesses and will present a compression between their capabilities.
- **Domains compassion:** We will use only Fearnhead and Liu's algorithm but will perform it both on our current domain and on additional domain, comparing the results achieve it them both and analyze the differences.
- **Finding heuristics for setting change points:** We will present two or more heuristics in order to set change points and will compare the results achieved for each of them.

1 Introduction

2 Preliminaries

This paper's terminology follows the one used in albrecht et al. ([2]).

2.1 Model

We consider a multi-agent model where agents interact each other in order to achieve a common goal. The process starts at time $t = 0$. At time t, each agent i receives a signal s_i^t and independently chooses an action a_i^t from some countable set of actions A_i. We do not put any limitations on s_i^t's structure and dynamic. This process continues indefinitely or until some termination criterion is satisfied (i.e., the goal achieved).

We will use $P(a_i^t|H_i^t, \theta_i, p)$ in order to denote the probability in which the action a_i^t is chosen where $H_i^t = (s_i^0, ..., s_i^t)$ is agent i's history of observations, θ_i is i's *type* and $p = (p_1, ..., p_n)$ is a vector of *continuous parameters* in θ_j. Each type's *continuous parameters* influence this type's abilities and therefore on its action choice (e.g., if one of those internal parameters sets that this agent can only see black objects then the probability for it to choose an action involving a white object is very low). Since this work mainly focus on detecting type changing points and since the work of Albrecht et al has provided a method for reasoning about the values of any bounded continuous parameters within types, we will assume that the reasoning about the likelihood of a type also includes the evaluation of its internal continuous parameters. Therefore we will adjust the above notation to be: $P(a_i^t|H_i^t, \theta_i)$.

To simplify the exposition, we assume that we control a single agent, i, which reasons about the behavior of another agent, j. We also assume that i knows j's action space A_j and that it can observe j's past actions, i.e. $a_j^{t-1} \in s_i^t$ for $t > 0$. The true type of j, denoted θ_j^* is unknown to i. However, i has access to a finite set of *hypothetical types* $\theta_j \in \Theta_j$, with $\theta_j^* \in \Theta_j$. We furthermore assume that i has all information relevant to j's decision making, so that H_j^t is a function of H_i^t and we can write $P(a_j^t|H_i^t, \theta_j)$. Finally, we assume that agent j will change his type during the process in a number of chosen time points $\Lambda = \{\lambda_1, ..., \lambda_k\}$ (the criteria for choosing those time points will be elaborate later on the paper).

Our goal is to devise a method which allows agent i to be able both to identify the specific time point in which the change in agent j's type has occur and its new type, based only on agent j's observed actions.

3 Analysis

We will first describe two change-point detection existing algorithms.

3.1 Maximum A-Posteriori

The first one was presented by Fearnhead and Liu [3]. The model of Fearnhead and Liu assumes a time-series observations $y_{1:n} = (y_1, y_2, ..., y_n)$ and a set of candidate models Q. The goal is to infer the MAP (Maximum A-Posteriori) set of change-points times $c_1, c_2, ..., c_m$, where $c_0 = 0$ and $c_{m+1} = n$ (i.e., there exist $m + 1$ segments). Therefore, the observations $y_{c_i+1:c_{i+1}}$ creating the ith segment associated the model $q_i \in Q$ and parameters θ_i.

The basic assumption in this model is that data after a change-point is independent of data prior to that change-point. Thus, we can model the change-point positions as a Markov chain in which the transition probabilities are defined by the time since the last change-point in the following way:

$$Pr(c_i + 1 = t|c_i = s) = g(t - s) \tag{1}$$

where $g(x)$ is a probability distribution over time and $G(x)$ is its cumulative distribution function.

Given a segment from time s to t and a model q, define the model

The model evidence for a model q and a given segment starting from a time point s and ending at a time point t is defined by:

$$L(s, t, q) = Pr(y_{s+1:t}|q) = \int Pr(y_{s+1:t}|q, \Theta)Pr(\theta)d\theta \tag{2}$$

We will denote the event that a change-point will occur at time t by ψ_j and the event that given a change-point at time j, the MAP choice of change-points has occurred prior to time j by ω_j. We can now use the following notations:

$$Pr_t(j, q) = Pr(FC_t = j, q, \omega_j, y_{1:t}) \tag{3}$$

$$P_t^{MAP} = Pr(\psi_j, \omega_j, y_{1:t}) \tag{4}$$

Where FC_t is the distribution over the position of the first change-point prior to time t which can be efficiently estimate using the standard Bayesian filtering recursions and an on-line Viterbi algorithm [3].

A development of the above equation will result in:

$$Pr_t(j, q) = (1 - G(t - j - 1))L(j, t, q)Pr(q)P_j^{MAP} \tag{5}$$

$$P_t^{MAP} = {}^{max}_{j,q}\Big[\frac{g(t-j)}{1 - G(t - j - 1)}Pr_t(j, q)\Big]$$

Finally, the Viterbi path can be recovered by finding the j and q values that maximize (3.1) at time t. We then can repeat the process again in order to find the values which maximize (3.1) at time j or any time point beforehand until reaching zero.

The algorithm is fully online, but requires O(n) computation at each time step, since Pt(j, q) values must be calculated for all j ¡ t. To reduce computation time to a constant, ideas from particle filtering can be leveraged to keep only a constant number of particles, M, at each time step, each of which represent a support point in the approximate density p(Ct = j, y1:t). At each time step, if the number of particles exceeds M, stratified optimal resampling [5] can be used to choose which particles to keep in a manner that minimizes the KL divergence from the true distribution in expectation.

3.2 CHAMP Algorithm

CHAMP algorithm (**Ch**angepoint detection using **A**pproximate **M**odel **P**arameters) was presented by Niekum et al. [4]. CHAMP is an algorithm for on-line Bayesian change-point detection intended for settings where it is difficult or undesirable to integrate over the parameters of candidate models. For those settings, CHAMP algorithm can be consider as an improvement of the on-line MAP algorithm. In order to allow finding change-point in cases where the parameters of the underlying model can not be marginalized CHAMP allows the use of models of any form, in which parameter estimates are available via means such as maximum likelihood fit, MCMC, or sample consensus methods. CHAMP requires three main changes from the work of Fearnhead and Liu: (i) *Approximate model evidence*- instead of calculating directly model evidence for each segment by integrating over the parameters of candidate models they used the Bayesian Information Criterion (BIC) in order to approximate it; (ii) *Minimum segment length*- since the model evidence is no longer well-defined a minimum segment length, α, is chosen for all models (i.e., change-points can only begin to be considered at time $t = 2\alpha$); (iii) *Particle definition*- a revised definition of a particle is offered such that each particle also include the model relevant to each time step.

4 Experimental Evaluation

We provide a detailed experimental evaluation of the proposed method in the level-based foraging domain [1], which was introduced as a test domain for ad hoc teamwork [5]. ***Manish - please provide a description of the domain we used, the agent's different types, the different experiments that we performed (or intended to perform- write down the experiment even if we still don't have results). As I recall we have 2 experiments without change-points and 4 experiments with change points. In case that we already have results, please describe it. In addition, can you please write down the explanation to why we removed the use in CHAMP from the inner loop.***

- We will compare the following four cases:
  1. **Perfect Information** - whenever agent j changes its type it reveal its new type (i.e., agent i knows that agent j changed its type and also knows what the new type is).
  2. **No Changes Allowed** - agent i knows agent j's initial type but assumes that no changes of type are allowed/possible.
  3. **Stationary Types** - no change of type is being allowed, Agents have the same type during the whole process (Using Stefano's method in order to figure out j's type).
  4. **Dynamic Types** - agent j can change its type without revealing it to agent i which will use a combination of CHAMP and Fearnhead and Liu's algorithm in order to identify change-points.

We note that the Monte Carlo Tree Search (MCTS) planning algorithm was used in each time step in order to compute optimal actions with respect to the agent's beliefs and types.

5 Performance Improvement Using Designated Change-Points

Up until now no special attention has been given to the question of how should an agent choose its change-points in order to improve the performance of the overall task. In this section we will propose possible heuristic to finding the optimal set of change-points during a given task. We note that since no real planning is possible due to the ad-hoc nature of the team, we will only try to find a specific agent's set of change-points assuming that the rest of the agents are not aware to those changing point.

6 Related work

7 Conclusions

References

[1] Stefano V Albrecht and Subramanian Ramamoorthy. A game-theoretic model and best-response learning method for ad hoc coordination in multiagent systems. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1155–1156. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[2] Stefano V Albrecht and Peter Stone. Reasoning about hypothetical agent behaviours and their parameters. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 547–555. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[3] Paul Fearnhead and Zhen Liu. On-line inference for multiple changepoint problems. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(4):589–605, 2007.

[4] Scott Niekum, Sarah Osentoski, Christopher G Atkeson, and Andrew G Barto. Champ: Changepoint detection using approximate model parameters. Technical report, CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 2014.

[5] Peter Stone, Gal A Kaminka, Sarit Kraus, Jeffrey S Rosenschein, et al. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI*, 2010.