Coordinated Multi-Agent Imitation Learning

Hoang M. Le[1] **Yisong Yue**[1] **Peter Carr**[2] **Patrick Lucey**[3]

Abstract

We study the problem of imitation learning from demonstrations of multiple coordinating agents. One key challenge in this setting is that learning a good model of coordination can be difficult, since coordination is often implicit in the demonstrations and must be inferred as a latent variable. We propose a joint approach that simultaneously learns a latent coordination model along with the individual policies. In particular, our method integrates unsupervised structure learning with conventional imitation learning. We illustrate the power of our approach on a difficult problem of learning multiple policies for fine-grained behavior modeling in team sports, where different players occupy different roles in the coordinated team strategy. We show that having a coordination model to infer the roles of players yields substantially improved imitation loss compared to conventional baselines.

1. Introduction

The areas of multi-agent planning and control have witnessed a recent wave of strong interest due to the practical desire to deal with complex real-world problems, such as smart-grid control, autonomous vehicles planning, managing teams of robots for emergency response, among others. From the learning perspective, (cooperative) multi-agent learning is not a new area of research (Stone & Veloso, 2000; Panait & Luke, 2005). However, compared to the progress in conventional supervised learning and single-agent reinforcement learning, the successes of multi-agent learning have remained relatively modest. Most notably, multi-agent learning suffers from extremely high dimensionality of both the state and actions spaces, as well as relative lack of data sources and experimental testbeds.

[1]California Institute of Technology, Pasadena, CA [2]Disney Research, Pittsburgh, PA [3]STATS LLC, Chicago, IL. Correspondence to: Hoang M. Le <hmle@caltech.edu>.

Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, PMLR 70, 2017. Copyright 2017 by the author(s).

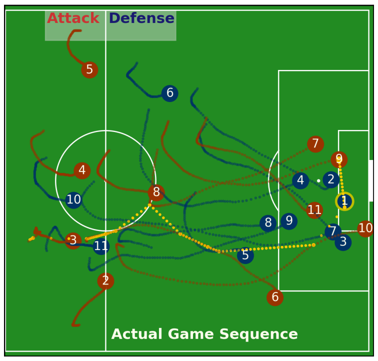

Figure 1. Our motivating example of learning coordinating behavior policies for team sports from tracking data. Red is the attacking team, blue is the defending team, and yellow is the ball.

The growing availability of data sources for coordinated multi-agent behavior, such as sports tracking data (Bialkowski et al., 2014), now enables the possibility of learning multi-agent policies from demonstrations, also known as multi-agent imitation learning. One particularly interesting aspect of domains such as team sports is that the agents must coordinate. For example, in the professional soccer setting depicted in Figure 1, different players must coordinate to assume different roles (e.g., defend left field). However, the roles and role assignment mechanism are unobserved from the demonstrations. Furthermore, the role for a player may change during the same play sequence. In the control community, this issue is known as "index-free" multi-agent control (Kingston & Egerstedt, 2010).

Motivated by these challenges, we study the problem of imitation learning for multiple coordinating agents from demonstrations. Many realistic multi-agent settings require coordination among collaborative agents to achieve some common goal (Guestrin et al., 2002; Kok et al., 2003). Beyond team sports, other examples include learning policies for game AI, controlling teams of multiple robots, or modeling collective animal behavior. As discussed above, we are interested in settings where agents have access to the outcome of actions from other agents, but the coordination mechanism is neither clearly defined nor observed, which makes the full state only partially observable.

We propose a semi-supervised learning framework that integrates and builds upon conventional imitation learning and unsupervised, or latent, structure learning. The latent structure model encodes a coordination mechanism, which approximates the implicit coordination in the demonstration data. In order to make learning tractable, we develop an alternating optimization method that enables integrated and efficient training of both individual policies and the latent structure model. For learning individual policies, we extend reduction-based single-agent imitation learning approaches into multi-agent domain, utilizing powerful black-box supervised techniques such as deep learning as base routines. For latent structure learning, we develop a stochastic variational inference approach.

We demonstrate the effectiveness of our method in two settings. The first is a synthetic experiment based on the popular predator-prey game. The second is a challenging task of learning multiple policies for team defense in professional soccer, using a large training set[1] of play sequences illustrated by Figure 1. We show that learning a good latent structure to encode implicit coordination yields significantly superior imitation performance compared to conventional baselines. To the best of our knowledge, this is the first time an imitation learning approach has been applied to jointly learn cooperative multi-agent policies at large scale.

2. Problem Formulation

In coordinated multi-agent imitation learning, we have K agents acting in coordination to achieve a common goal (or sequence of goals). Training data $\mathcal{D}$ consists of multiple demonstrations of K agents. Importantly, we assume the identity (or indexing) of the K experts may change from one demonstration to another. Each (unstructured) set of demonstrations is denoted by $U = \{U_1, \ldots, U_K\}$, where $U_k = \{u_{t,k}\}_{t=1}^{T}$ is the sequence of actions by agent k at time t. Note that each set of demonstrations can have varying sequence length T. Let $C = \{c_t\}_{t=1}^{T}$ be the context associated with each demonstration sequence.

Policy Learning. Our ultimate goal is to learn a (largely) decentralized policy, but for clarity we first present the problem of learning a fully centralized multi-agent policy. Following the notation of (Ross et al., 2011), let $\vec{\pi}(\vec{s}) := \vec{a}$ denote the joint policy that maps the joint state, $\vec{s} = [s_1, \ldots, s_K]$, of all K agents into K actions $\vec{a} = [a_1, \ldots, a_K]$. The goal is to minimize imitation loss:

$$\mathcal{L}_{imitation} = \mathbb{E}_{\vec{s}\sim d_{\vec{\pi}}}\left[\ell(\vec{\pi}(\vec{s}))\right],$$

where $d_{\vec{\pi}}$ denotes the distribution of states experienced by joint policy $\vec{\pi}$ and ℓ is the imitation loss defined over the demonstrations (e.g., squared loss for deterministic policies, or cross entropy for stochastic policies).

The decentralized setting decomposes the joint policy $\vec{\pi} = [\pi_1, \ldots, \pi_K]$ into K policies, each tailored to a specific agent index or "role".[2] The loss function is then:

$$\mathcal{L}_{imitation} = \sum_{k=1}^{K} \mathbb{E}_{s\sim d_{\pi_k}}\left[\ell(\pi_k(s_k))\right].$$

Black-Box Policy Classes. In order to leverage powerful black-box policy classes such as random forests and deep learning, we take a learning reduction approach to training $\vec{\pi}$. One consequence is that the state space representation $s = [s_1, \ldots, s_K]$ must be consistently indexed, e.g., agent k in one instance must correspond to agent k in another instance. This requirement applies for both centralized and decentralized policy learning, and is often implicitly assumed in prior work on multi-agent learning. A highly related issue arises in distributed control of index-free coordinating robots, e.g., to maintain a defined formation (Kloder & Hutchinson, 2006; Kingston & Egerstedt, 2010).

Motivating example: Soccer Domain. Consider the task of imitating professional soccer players, where training data includes play sequences from different teams and games. Context C corresponds to the behavior of the opposing team and the ball. The data includes multiple sequences of K-set of trajectories $U = \{U_1, U_2, \ldots, U_K\}$, where the actual identity of player generating U_k may change from one demonstration to the next.

One important challenge for black-box policy learning is constructing an indexing mechanism over the agents to yield a consistent state representation. For example, the same index should correspond to the "left defender" in all instances. Otherwise, the inputs to the policy will be inconsistent, making learning difficult if not impossible. Note that barring extensive annotations or some heuristic rule-based definitions, it is unnatural to quantitatively define what makes a player "left defender". In addition, even if we had a way to define who the "left defender" is, he may not stay in the same role during the same sequence.

Role-based Indexing. We address index-free policy learning via role learning and role-based index assignment. To motivate our notion of role, let's first consider the simplest indexing mechanism: one could equate role to agent identity. However, the data often comes from various sequences, with heterogeneous identities and teams of agents. Thus instead of learning identity-specific policies, it is more natural and data-efficient to learn a policy per role. However, a key challenge in learning policies directly is that *the roles are undefined, unobserved, and could change dynamically within the same sequence.* We thus view learning the coordination, via role assignment, as an unsupervised structured prediction problem.

[1]Data at http://www.stats.com/data-science/ and see video result at http://hoangminhle.github.io

[2]It is straightforward to extend our formulation to settings where multiple agents can occupy the same role, and where not all roles are occupied across all execution sequences.

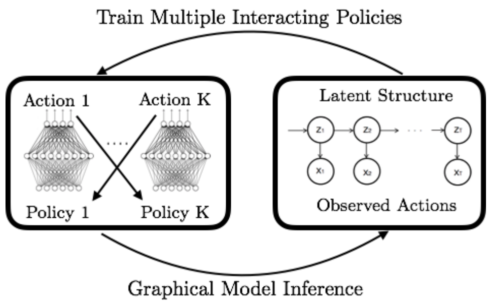

Figure 2. Alternating stochastic optimization training scheme for our semi-supervised structure regularization model.

Coordination via Structured Role Assignment. Instead of handcrafting the definition of roles, we learn the roles in an unsupervised fashion, without attaching any semantic labels to the roles. At the same time, role transition should obey certain structural regularity, due to coordination. This motivates using graphical models to represent the coordination structure.

Coordinated Policy Learning. We formulate the indexing mechanism as an assignment function $\mathcal{A}$ which maps the unstructured set U and some probabilistic structured model q to an indexed set of trajectory A rearranged from U, i.e.,

$$\mathcal{A} : \{U_1, .., U_K\} \times q \mapsto [A_1, .., A_K],$$

where the set $\{A_1, .., A_K\} \equiv \{U_1, .., U_K\}$. We view q as a latent variable model that infers the role assignments for each set of demonstrations. Thus, q drives the indexing mechanism $\mathcal{A}$ so that state vectors can be consistently constructed to facilitate optimizing for the imitation loss.

We employ entropy regularization, augmenting the imitation loss with some low entropy penalty (Grandvalet et al., 2004; Dudik et al., 2004), yielding our overall objective:

$$\min_{\pi_1,..,\pi_K,\mathcal{A}} \sum_{k=1}^{K} \mathbb{E}_{s_k \sim d_{\pi_k}} \left[\ell(\pi_k(s_k)) | \mathcal{A}, \mathcal{D}\right] - \lambda H(\mathcal{A}|\mathcal{D}) \quad (1)$$

where both imitation loss and entropy are measured with respect to the state distribution induced by the policies, and $\mathcal{D}$ is training data. This objective can also be seen as maximizing the mutual information between latent structure and observed trajectories (Krause et al., 2010).

3. Learning Approach

Optimizing (1) is challenging for two reasons. First, beyond the challenges inherited from single-agent settings, multi-agent imitation learning must account for multiple simultaneously learning agents, which is known to cause non-stationarity for multi-agent reinforcement learning (Busoniu et al., 2008). Second, the latent role assignment model, which forms the basis for coordination, depends on the actions of the learning policies, which in turn depend on the structured role assignment.

Algorithm 1 Coordinated Multi-Agent Imitation Learning

```
Input: Multiple unstructured trajectory sets $U = \{U_1, \dots, U_K\}$
    with $U_k = \{u_{t,k}\}_{t=1}^T$ and context $C = \{c_t\}_{t=1}^T$.
Input: Graphical model $q$ with global/local parameters $\theta$ and $z$.
Input: Initialized policies $\pi_k, k = 1, \dots, K$
Input: Step size sequence $\rho_n, n = 1, 2, \dots$
 1: repeat
 2:     $[A_1, \dots, A_K] \leftarrow \texttt{Assign}\{U_1, \dots, U_K | q(\theta, z)\}$
 3:     $[\pi_1, \dots, \pi_K] \leftarrow \texttt{Learn}\,[A_1, \dots, A_K, C]$
 4:     Roll-out $\pi_1, \dots, \pi_K$ to obtain $\widehat{A}_1, \dots, \widehat{A}_K$
 5:     $A_k \leftarrow \widehat{A}_k \ \forall k$
        (Alternatively: $A_k \leftarrow \widehat{A}_k$ with prob $\eta$ for $\eta \to 1$)
 6:     $q(\theta, z) \leftarrow \texttt{LearnStructure}\{A_1, \dots, A_K, C, \theta, \rho_n\}$
 7: until No improvement on validation set
output $K$ policies $\pi_1, \pi_2, \dots, \pi_K$
```

We propose an alternating optimization approach to solving (1), summarized in Figure 2. The main idea is to integrate imitation learning with unsupervised structure learning by taking turns to (i) optimize for imitation policies while fixing a structured model (minimizing imitation loss), and (ii) re-train the latent structure model and reassign roles while fixing the learning policies (maximizing role assignment entropy). The alternating nature allows us to circumvent the circular dependency between policy learning and latent structure learning. Furthermore, for (i) we develop a stable multi-agent learning reduction approach.

3.1. Approach Outline

Algorithm 1 outlines our framework. We assume the latent structure model for computing role assignments is formulated as a graphical model. The multi-agent policy training procedure `Learn` utilizes a reduction approach, and can leverage powerful off-the-shelf supervised learning tools such as deep neural networks (Hochreiter & Schmidhuber, 1997). The structure learning `LearnStructure` and role assignment `Assign` components are based on graphical model training and inference. For efficient training, we employ alternating stochastic optimization (Hoffman et al., 2013; Johnson & Willsky, 2014; Beal, 2003) on the same mini-batches. Note that batch training can be deployed similarly, as illustrated by one of our experiments.

We interleave the three components described above into a complete learning algorithm. Given an initially unstructured set of training data, an initialized set of policies, and prior parameters of the structure model, Algorithm 1 performs alternating structure optimization on each mini-batch (size 1 in Algorithm 1).

- Line 2: Role assignment is performed on trajectories $\{A_1, \dots, A_K\}$ by running inference procedure (Algorithm 4). The result is an ordered set $[A_1, \dots, A_K]$, where trajectory A_k corresponds to policy π_k.
- Line 3-5: Each policy π_k is updated using joint multi-agent training on the ordered set $[A_1, \dots, A_K, C]$

(Algorithm 2). The updated models are executed to yield a rolled-out set of trajectories, which replace the previous set of trajectories $\{A_k\}$.

- Line 6: Parameters of latent structured model are updated from the rolled-out trajectories (Algorithm 3).

The algorithm optionally includes a mixing step on line 5, where the rolled-out trajectories may replace the training trajectories with increasing probability approaching 1, which is similar to scheduled sampling (Bengio et al., 2015), and may help stabilize learning in the early phase of the algorithm. In our main experiment, we do not notice a performance gain using this option.

3.2. Joint Multi-Agent Imitation Learning

In this section we describe the `Learn` procedure for multi-agent imitation learning in Line 3 of Algorithm 1. As background, for single agent imitation learning, reduction-based methods operate by iteratively collecting a new data set $\mathcal{D}_n$ at each round n of training, consisting of state-action pairs (s_t, a_t^*) where a_t^* is some optimal or demonstrated action given state s_t. A new policy can be formed by (i) combining a new policy from this data set $\mathcal{D}_n$ with previously learned policy π (Daumé III et al., 2009) or (ii) learning a new policy π directly from the data set formed by aggregating $\mathcal{D}_1, \ldots, \mathcal{D}_n$ (Ross et al., 2011). Other variants exist although we do not discuss them here.

The intuition behind the iterative reduction approach is to prevent a mismatch in training and prediction distributions due to sequential cascading errors (also called covariate-shift). The main idea is to use the learned policy's own predictions in the construction of subsequent states, thus simulating the test-time performance during training. This mechanism enables the agent to learn a policy that is robust to its own mistakes. Reduction-based methods also accommodate any black-box supervised training subroutine. We focus on using expressive function classes such as Long Short-Term Memory networks (LSTM) (Hochreiter & Schmidhuber, 1997) as the policy class.[3]

Algorithm 2 outlines the `Learn` procedure for stable multi-agent imitation learning. Assume we are given consistently indexed demonstrations $A = [A_1, \ldots, A_K]$, where each $A_k = \{a_{t,k}\}_{t=1}^T$ corresponds action of policy π_k. Let the corresponding expert action be $a_{t,k}^*$. To lighten the notation, we denote the per-agent state vector by $s_{t,k} = \varphi_k([a_{t,1}, \ldots, a_{t,k}, \ldots, a_{t,K}, c_t])$[4]

[3]Note that conventional training of LSTMs does not address the cascading error problem. While LSTMs are very good at sequence-to-sequence prediction tasks, they cannot naturally deal with the drifting of input state distribution drift caused by action output feedback in dynamical systems (Bengio et al., 2015).

[4]Generally, state vector $s_{t,k}$ of policy π_k at time t can be constructed as $s_{t,k} = [\phi_k([a_{1:t,1}, c_{1:t}]), \ldots, \phi_k([a_{1:t,K}, c_{1:t}])]$

Algorithm 2 Joint Multi-Agent Imitation Learning $\texttt{Learn}(A_1, A_2, \ldots, A_K, C)$

```
Input: Ordered actions $A_k = \{a_{t,k}\}_{t=1}^T \ \forall k$, context $\{c_t\}_{t=1}^T$
Input: Initialized policies $\pi_1, \ldots, \pi_K$
Input: base routine $\texttt{Train}(S, A)$ mapping state to actions
1: Set increasing prediction horizon $j \in \{1, \ldots, T\}$
2: for $t = 0, j, 2j, \ldots, T$ do
3:   for $i = 0, 1, \ldots, j-1$ do
4:     Roll-out $\hat{a}_{t+i,k} = \pi_k(\hat{s}_{t+i-1,k}) \quad \forall$ agent $k$
5:     Cross-update for each policy $k \in \{1, \ldots, K\}$
       $\hat{s}_{t+i,k} = \varphi_k([\hat{a}_{t+i,1}, \ldots, \hat{a}_{t+i,k}, \ldots, \hat{a}_{t+i,K}, c_{t+i}])$
6:   end for
7:   Policy update for all agent $k$
     $\pi_k \leftarrow \texttt{Train}(\{\hat{s}_{t+i,k}, a^*_{t+i+1,k}\}_{i=0}^j)$
8: end for
output $K$ updated policies $\pi_1, \pi_2, \ldots, \pi_K$
```

Algorithm 2 employs a roll-out horizon j, which divides the entire trajectory into T/j segments. The following happens for every segment:

- Iteratively perform roll-out at each time step i for all K policies (line 4) to obtain actions $\{\widehat{a}_{i,k}\}$.
- Each policy simultaneously updates its state $\widehat{s}_{i,k}$, using the prediction from all other policies (line 5).
- At the end of the current segment, all policies are updated using the error signal from the deviation between predicted $\widehat{a}_{i,k}$ versus expert action $a^*_{i,k}$, for all i along the sub-segment (line 7).

After policy updates, the training moves on to the next j-length sub-segment, using the freshly updated policies for subsequent roll-outs. The iteration proceeds until the end of the sequence is reached. In the outer loop the roll-out horizon j is incremented.

Two key insights behind our approach are:

- In addition to the training-prediction mismatch issue in single-agent learning, each agent's prediction must also be robust to imperfect predictions from other agents. This non-stationarity issue also arises in multi-agent reinforcement learning (Busoniu et al., 2008) when agents learn simultaneously. We perform joint training by cross-updating each agent's state using previous predictions from other agents.
- Many single-agent imitation learning algorithms assume the presence of a dynamic oracle to provide one-step corrections a_t^* along the roll-out trajectories. In practice, dynamic oracle feedback is very expensive to obtain and some recent work have attempted to relax this requirement (Le et al., 2016; Ho & Ermon, 2016). Without dynamic oracles, the rolled-out trajectory can deviate significantly from demonstrated trajectories when the prediction horizon j is large ($\approx T$), leading to training instability. Thus j is gradually increased to allow for slowly learning to make good sequential predictions over longer horizons.

For efficient training, we focus on stochastic optimization, which can invoke base routine `Train` multiple times and thus naturally accommodates varying j. Note that the batch-training alternatives to Algorithm 2 can also employ similar training schemes, with similar theoretical guarantees lifted to the multi-agent case. The Appendix shows how to use DAgger (Ross et al., 2011) for Algorithm 2, which we used for our synthetic experiment.

3.3. Coordination Structure Learning

The coordination mechanism is based on a latent structured model that governs the role assignment. The training and inference procedures seek to address two main issues:

- `LearnStructure`: unsupervised learning a probabilistic role assignment model q.
- `Assign`: how q informs the indexing mechanism so that unstructured trajectories can be mapped to structured trajectories amenable to Algorithm 2.

Given an arbitrarily ordered set of trajectories $U = \{U_1, \dots, U_K, C\}$, let the coordination mechanism underlying each such U be governed by a true unknown model p, with global parameters θ. We suppress the agent/policy subscript and consider a generic featurized trajectory $x_t = [u_t, c_t]\ \forall t$. Let the latent role sequence for the same agent be $z = z_{1:T}$. At any time t, each agent is acting according to a latent role $z_t \sim \texttt{Categorical}\{\bar{1}, \bar{2}, \dots, \bar{K}\}$, which are the local parameters to the structured model.

Ideally, role and index assignment can be obtained by calculating the true posterior $p(z|x, \theta)$, which is often intractable. We instead aim to approximate $p(z|x, \theta)$ by a simpler distribution q via techniques from stochastic variational inference (Hoffman et al., 2013), which allows for efficient stochastic training on mini-batches that can naturally integrate with our imitation learning subroutine.

In variational inference, posterior approximation is often cast as optimizing over a simpler model class $\mathcal{Q}$, via searching for parameters θ and z that maximize the evidence lower bound (ELBO) $\mathcal{L}$:

$$\mathcal{L}\left(q(z, \theta)\right) \triangleq \mathbb{E}_q\left[\ln p(z, \theta, x)\right] - \mathbb{E}_q\left[\ln q(z, \theta)\right] \leqslant \ln p(x)$$

Maximizing $\mathcal{L}$ is equivalent to finding $q \in \mathcal{Q}$ to minimize the KL divergence $\mathrm{KL}\left(q(z, \theta|x) || p(z, \theta|x)\right)$. We focus on the structured mean-field variational family, which factorizes q as $q(z, \theta) = q(z)q(\theta)$. This factorization breaks the dependency between θ and z, but not between single latent states z_t, unlike variational inference for i.i.d data (Kingma & Welling, 2013).

3.3.1. Training to learn model parameters

The procedure to learn the parameter of our structured model is summarized in Algorithm 3. Parameter learning proceeds via alternating updates over the factors $q(\theta)$ and $q(z)$, while keeping other factor fixed. Stochastic variational inference performs such updates efficiently in mini-batches. We slightly abuse notations and overload θ for the natural parameters of global parameter θ in the exponential family. Assuming the usual conjugacy in the exponential family, the stochastic natural gradient takes a convenient form (line 2 of Algo 3, and derivation in Appendix), where $t(z, x)$ is the vector of sufficient statistics, b is a vector of scaling factors adjusting for the relative size of the mini-batches. Here the global update assumes optimal local update $q(z)$ has been computed.

Algorithm 3 Structure Learning
`LearnStructure` $\{U_1, \dots, U_K, C, \theta, \rho\} \mapsto q(\theta, z)$

```
Input: $X_k = \{x_{t,k}\}_{t=1}^T = \{[u_{t,k}, c_t]\}\ \forall t, k. X = \{X_k\}_{k=1}^K$
    Graphical model parameters $\theta$, stepsize $\rho$
1: Local update: compute $q(z)$ via message-passing while fixing $\theta$ (See Appendix for derivations)
2: Global parameter update: via natural gradient ascent
    $\theta \leftarrow \theta(1-\rho) + \rho(\theta_{prior} + b^\top \mathbb{E}_{q(z)}[t(z, x)])$
output Updated model $q(\theta, z) = q(\theta)q(z)$
```

Fixing the global parameters, the local updates are based on message-passing over the underlying graphical model. The exact mathematical derivation depends on the specific graph structure. The simplest scenario is to assume independence among z_t's, which resembles naive Bayes. In our experiments, we instead focus on Hidden Markov Models to capture first-order dependencies in role transitions over play sequences. In that case, line 1 of Algorithm 3 resembles running the forward-backward algorithm to compute the update $q(z)$. The forward-backward algorithm in the local update step takes $O(K^2T)$ time for a chain of length T and K hidden states. For completeness, derivation of parameter learning for HMMs is included in the Appendix.

3.3.2. Inference for role and index assignment

We can compute two types of inference on a learned q:

Role inference. Compute the most likely role sequence $\{z_{t,k}\}_{t=1}^T \in \{\bar{1}, \dots, \bar{K}\}^T$, e.g., using Viterbi (or dynamic programming-based forward message passing for graph structures). This most likely role sequence for agent k, which is the low-dimensional representation of the coordination mechanism, can be used to augment the contextual feature $\{c_t\}_{t=1}^T$for each agent's policy training.

Role-based Index Assignment Transform the unstructured set U into an ordered set of trajectories A to facilitate the imitation learning step. This is the more important task for the overall approach. The intuitive goal of an indexing mechanism is to facilitate consistent agent trajectory to policy mapping. Assume for notational convenience that we want index k assigned to an unique agent who is most likely assuming role $\bar{k}$. Our inference technique rests on the

Algorithm 4 Multi-Agent Role Assignment

```
Assign $\{U_1, \ldots, U_K | q\} \mapsto [A_1, \ldots, A_K]$
Input: Approximate inference model $q$. Unordered trajectories $U = \{U_k\}_{k=1}^K$.
1: Calculate cost matrix $M \in \mathbb{R}^{K \times K}$ per equation 2
2: $\mathcal{A} \leftarrow$ MinCostAssignment$(M)$
output $A_k = U_{\mathcal{A}(k)} \quad \forall k = 1, 2, \ldots, K$
```

well-known Linear Assignment Problem (Papadimitriou & Steiglitz, 1982), which is solved optimally via the Kuhn-Munkres algorithm. Specifically, construct the cost matrix M as:

$$M = M_1 \odot M_2 \tag{2}$$

$$M_1 = \left[q(\{x_{t,k}\} | z_{t,k} = \bar{k}) \right] = \left[\prod_{t=1}^{T} q(x_{t,k} | z_{t,k} = \bar{k}) \right]$$

$$M_2 = \left[\ln q(\{x_{t,k}\} | z_{t,k} = \bar{k}) \right] = \left[\sum_{t=1}^{T} \ln q(x_{t,k} | z_{t,k} = \bar{k}) \right]$$

where $k = 1, \ldots, K, \bar{k} = \bar{1}, \ldots, \bar{K}, \odot$ is the Hadamard product, and matrices M_1, M_2 take advantage of the Markov property of the graphical model. Now solving the linear assignment problem for cost matrix M, we obtain the matching $\mathcal{A}$ from role $\bar{k}$ to index k, such that the total cost per agent is minimized. From here, we rearrange the unordered set $\{U_1, \ldots, U_K\}$ to the ordered sequence $[A_1, \ldots, A_K] \equiv [U_{\mathcal{A}(1)}, \ldots, U_{\mathcal{A}(K)}]$ according to the minimum cost mapping.

To see why this index assignment procedure results in an increased entropy in the original objective (1), notice that:

$$H(\mathcal{A}|\mathcal{D}) \approx -\sum_{\bar{k}=1}^{K} P(\bar{k}) q(\mathcal{A}(A_k) = \bar{k}) \log q(\mathcal{A}(A_k) = \bar{k})$$
$$= -\frac{1}{K} \sum_{\bar{k}=1}^{K} M(\bar{k}, k),$$

where we assume each latent role $\bar{k}$ has equal probability. The RHS increases from the linear assignment and consequent role assignment procedure. Our inference procedure to perform role assignment is summarized in Algorithm 4.

4. Experiments

We present empirical results from two settings. The first is a synthetic setting based on predator-prey, where the goal is to imitate a coordinating team of predators. The second is a large-scale imitation learning setting from player trajectores in professional soccer games, where the goal is to imitate defensive team play.

4.1. Predator-Prey Domain

Setting. The predator-prey problem, also frequently called the Pursuit Domain (Benda, 1985), is a popular setting for multi-agent reinforcement learning. The traditional setup is with four predators and one prey, positioned on a grid board. At each time step, each agent has five moves: N,S,E,W or no move. The world is toroidal: the agents can move off one end of the board and come back on the other end. Agents make move simultaneously, but two agents cannot occupy the same position, and collisions are avoided by assigning a random move priority to the agents at each time step. The predators can capture the prey only if the prey is surrounded by all four predators. The goal of the predators is to capture the prey as fast as possible, which necessarily requires coordination.

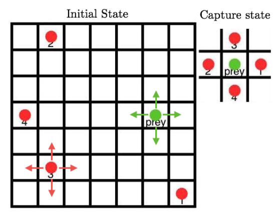

Figure 3.

Data. The demonstration data is collected from 1000 game instances, where four experts, indexed 1 to 4, are prescribed the consistent and coordinated role as illustrated in the capture state of Figure 3. In other words, agent 1 would attempt to capture the prey on the right hand side, which allows for one fixed role for each expert throughout the game. However, the particular role assignment is hidden from the imitation learning task. Each expert is then exhaustively trained using Value Iteration (Sutton & Barto, 1998) in the reinforcement learning setting, with the reward of 1 if the agent is in the position next to the prey according to its defined role, and 0 otherwise. A separate set of 100 games was collected for evaluation. A game is terminated after 50 time steps if the predators fail to capture the prey. In the test set, the experts fail to capture the prey in 2% of the games, and on average take 18.3 steps to capture the prey.

Experiment Setup. For this experiment, we use the batch version of Algorithm 1 (see appendix) to learn to imitate the experts using only demonstrations. Each policy is represented by a random forest of 20 trees, and were trained over 10 iterations. The expert correction for each rolled-out state is collected via Value Iteration. The experts thus act as dynamic oracles, which result in a multi-agent training setting analogous to DAgger (Ross et al., 2011). We compare two versions of multi-agent imitation learning:

- **Coordinated Training.** We use our algorithm, with the latent structure model represented by a discrete Hidden Markov Model with binomial emission. We use Algorithm 4 to maximize the role consistency of the dynamic oracles across different games.
- **Unstructured Training.** An arbitrary role is assigned to each dynamic oracle for each game, i.e., the agent index is meaningless.

In both versions, training was done using the same data aggregation scheme and batch training was conducted using the same random forests configuration.

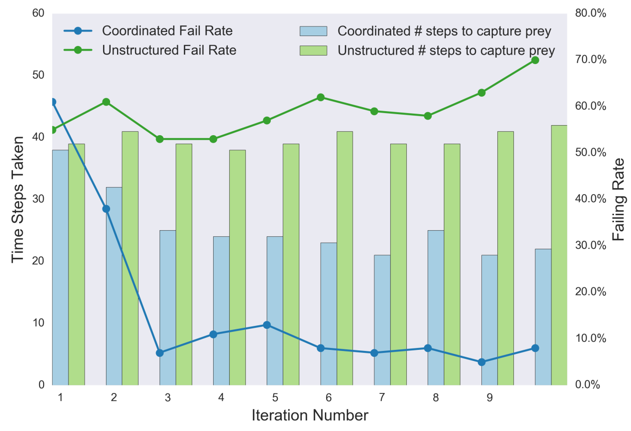

Figure 4. Comparing performance in Predator-Prey between our approach and unstructured multi-agent imitation learning, as a function of the number of training rounds. Our approach demonstrates both significantly lower failure rates as well as lower average time to success (for successful trials).

Results. Figure 4 compares the test performance of our method versus unstructured multi-agent imitation learning. Our method quickly approaches expert performance (average 22 steps with 8% failure rate in the last iteration), whereas unstructured multi-agent imitation learning performance did not improve beyond the first iteration (average 42 steps with 70% failure rate). Note that we even gave the unstructured baseline some advantage over our method, by forcing the prey to select the moves last after all predators make decisions (effectively making the prey slower). Without this advantage, the unstructured policies fail to capture the prey almost 100% of the time. Also, if the same restriction is applied to the policies obtained from our method, performance would be on par with the experts (100% success rate, with similar number of steps taken).

4.2. Multi-agent Imitation Learning for Soccer

Setting. Soccer is a popular domain for multi-agent learning. RoboCup, the robotic and simulation soccer platform, is perhaps the most popular testbed for multi-agent reinforcement learning research to date (Stone, 2016). The success of MARL has been limited, however, due to the extremely high dimensionality of the problem. In this experiment, we aim to learn multi-agent policies for team soccer defense, based on tracking data from real-life professional soccer (Bialkowski et al., 2014).

Data. We use the tracking data from 45 games of real professional soccer from a recent European league. The data was chunked into sequences with one team attacking and the other defending. Our goal is to learn up to 10 policies for team defense (11 players per team, minus the goal keeper). The training data consists of 7500 sets of trajectories $A = \{A_1, \ldots, A_{10}\}$, where $A_k = \{a_{t,k}\}_{t=1}^{T}$ is the sequence of positions of one defensive player, and C is the context consisting of opponents and the ball. Overall, there are about 1.3 million frames at 10 frames per second. The average sequence length is 176 steps, and the maximum is 1480.

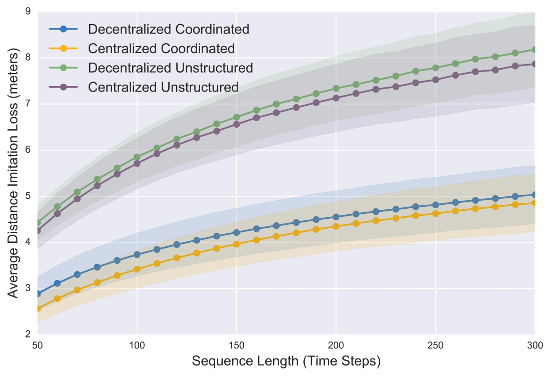

Figure 5. Experimental results on soccer domain. We see that using coordination substantially improves the imitation loss, and that the decentralized policy is comparable to the centralized.

Experiment Setup. Each policy is represented by a recurrent neural network structure (LSTM), with two hidden layers of 512 units each. As LSTMs generally require fixed-length input sequences, we further chunk each trajectory into sub-sequences of length 50, with overlapping window of 25 time steps. The joint multi-agent imitation learning procedure follows Algorithm 2 closely. In this setup, without access to dynamic oracles for imitation learning in the style of SEARN (Daumé III et al., 2009) and DAgger (Ross et al., 2011), we gradually increase the horizon of the rolled-out trajectories from 1 to 10 steps lookahead. Empirically, this has the effect of stabilizing the policy networks early in training, and limits the cascading errors caused by rolling-out to longer horizons.

The structured model component is learned via stochastic variational inference on a continuous HMM, where the per-state emission distribution is a mixture of Gaussians. Training and inference operate on the same mini-batches used for joint policy learning.

We compare against two variations. The first employs centralized policy that aggregates the state vectors of all decentralized learner and produces the actions for all players, i.e., a multi-task policy. The centralized approach generally requires more model parameters, but is potentially much more accurate. The second variation is to not employ joint multi-agent training: we modify Algorithm 2 to not cross-update states between agents, and each role is trained conditioned on the ground truth of the other agents.

Results. Figure 5 shows the results. Our coordinated learning approach substantially outperforms conventional imitation learning without structured coordination. The imitation loss measures average distance of roll-outs and ground truth in meters (note the typical size of soccer field

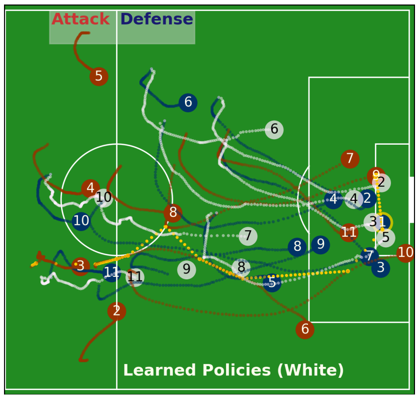

Figure 6. Result of 10 coordinated imitation policies, corresponding with Figure 1. White is the rolled-out imitation policies.

is 110 × 70 meters). As expected, average loss increases with longer sequences, due to cascading errors. However, this error scales sub-linearly with the length of the horizon, even though the policies were trained on sequences of length 50. Note also that the performance difference between decentralized and centralized policies is insignificant compared to the gap between coordinated and unstructured policies, further highlighting the benefits of structured coordination in multi-agent settings. The loss of a single network, non-joint training scheme is very large and thus omitted from Figure 5 (see the appendix).

Visualizations. Imitation loss, of course, is not a full reflection of the quality of the learned policies. Unlike predator-prey, the long-term reward signal is not available, so we rely on visual inspection as part of evaluation. Figure 6 overlays policy prediction on top of the actual game sequence from Figure 1. Additional test examples are included in our supplemental video [5]. We note that learned policies are qualitatively similar to the ground truth demonstrations, and can be useful for applications such as counterfactual replay analysis (Le et al., 2017). Figure 7 displays the Gaussian components of the underlying HMM. The components correspond to the dominant modes of the roles assigned. Unlike the predator-prey domain, roles can be switched during a sequence of play. See the appendix for more details on role swap frequency.

5. Other Related Work

The problem of multi-agent imitation learning has not been widely considered, perhaps with the exception of (Chernova & Veloso, 2007) which focused on very different applications and technical challenges (i.e., learning a model of a joint task by collecting samples from direct interaction with teleoperating human teachers). The actual learning algorithm there requires the learner to collect enough data points from human teachers for confident classification of

[5]Watch video at `http://hoangminhle.github.io`

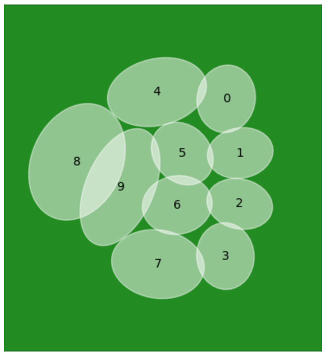

Figure 7. Components of role distributions, corresponding to a popular formation arrangement in professional soccer

task. It is not clear how well the proposed method would translate to other domains.

Index-free policy learning is generally difficult for black-box machine learning techniques. Some recent work has called attention to the importance of order to learning when input or output are sets (Vinyals et al., 2015), motivated by classic algorithmic and geometric problems such as learning to sort a set of numbers, or finding convex hull for a set of points, where no clear indexing mechanism exists. Other permutation invariant approaches include those for standard classification (Shivaswamy & Jebara, 2006).

6. Limitations and Future Work

In principle, the training and inference of the latent structure model can accommodate different types of graphical models. However, the exact procedure varies depending on the graph structure. It would be interesting to find domains that can benefit from more general graphical models. Another possible direction is to develop fully end-to-end differentiable training methods that can accommodate our index-free policy learning formulation, especially deep learning-based method that could provide computational speed-up compared to traditional graphical model inference. One potential issue with the end-to-end approach is the need to depart from a learning-reductions style approach.

Although we addressed learning from demonstrations in this paper, the proposed framework can also be employed for generative modeling, or more efficient structured exploration for reinforcement learning. Along that line, our proposed method could serve as a useful component of general reinforcement learning, especially in multi-agent settings where traditional exploration-based approaches such as Q-learning prove computationally intractable.

Acknowledgement. This work was funded in part by NSF Awards #1564330 & #1637598, JPL PDF IAMS100224, a Bloomberg Data Science Research Grant, and a gift from Northrop Grumman.

References

Beal, Matthew James. *Variational algorithms for approximate Bayesian inference*. University of London United Kingdom, 2003.

Benda, Miroslav. On optimal cooperation of knowledge sources. *Technical Report BCS-G2010-28*, 1985.

Bengio, Samy, Vinyals, Oriol, Jaitly, Navdeep, and Shazeer, Noam. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.

Bialkowski, Alina, Lucey, Patrick, Carr, Peter, Yue, Yisong, and Matthews, Iain. "win at home and draw away": Automatic formation analysis highlighting the differences in home and away team behaviors. In *MIT Sloan Sports Analytics Conference (SSAC)*, 2014.

Blei, David M, Kucukelbir, Alp, and McAuliffe, Jon D. Variational inference: A review for statisticians. *Journal of the American Statistical Association*, (just-accepted), 2017.

Busoniu, Lucian, Babuska, Robert, and De Schutter, Bart. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems Man and Cybernetics Part C Applications and Reviews*, 38(2):156, 2008.

Chernova, Sonia and Veloso, Manuela. Multiagent collaborative task learning through imitation. In *Proceedings of the fourth International Symposium on Imitation in Animals and Artifacts*, pp. 74–79, 2007.

Daumé III, Hal, Langford, John, and Marcu, Daniel. Search-based structured prediction. *Machine learning*, 75(3):297–325, 2009.

Dudik, Miroslav, Phillips, Steven J, and Schapire, Robert E. Performance guarantees for regularized maximum entropy density estimation. In *International Conference on Computational Learning Theory*, pp. 472–486. Springer, 2004.

Grandvalet, Yves, Bengio, Yoshua, et al. Semi-supervised learning by entropy minimization. In *NIPS*, volume 17, pp. 529–536, 2004.

Guestrin, Carlos, Lagoudakis, Michail, and Parr, Ronald. Coordinated reinforcement learning. In *ICML*, volume 2, pp. 227–234, 2002.

Ho, Jonathan and Ermon, Stefano. Generative adversarial imitation learning. In *Advances in Neural Information Processing Systems*, pp. 4565–4573, 2016.

Hochreiter, Sepp and Schmidhuber, Jürgen. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Hoffman, Matt and Blei, David. Structured stochastic variational inference. *CoRR abs/1404.4114*, 2014.

Hoffman, Matthew D, Blei, David M, Wang, Chong, and Paisley, John William. Stochastic variational inference. *Journal of Machine Learning Research*, 14(1):1303–1347, 2013.

Johnson, Matthew James and Willsky, Alan S. Stochastic variational inference for bayesian time series models. In *ICML*, pp. 1854–1862, 2014.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingston, Peter and Egerstedt, Magnus. Index-free multi-agent systems: An eulerian approach. *IFAC Proceedings Volumes*, 43(19):215–220, 2010.

Kloder, Stephen and Hutchinson, Seth. Path planning for permutation-invariant multirobot formations. *IEEE Transactions on Robotics*, 22(4):650–665, 2006.

Kok, Jelle R, Spaan, Matthijs TJ, Vlassis, Nikos, et al. Multi-robot decision making using coordination graphs. In *Proceedings of the 11th International Conference on Advanced Robotics, ICAR*, volume 3, pp. 1124–1129, 2003.

Krause, Andreas, Perona, Pietro, and Gomes, Ryan G. Discriminative clustering by regularized information maximization. In *Advances in neural information processing systems*, pp. 775–783, 2010.

Le, Hoang, Kang, Andrew, Yue, Yisong, and Carr, Peter. Smooth imitation learning for online sequence prediction. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 680–688, 2016.

Le, Hoang M., Carr, Peter, Yue, Yisong, and Lucey, Patrick. Data-driven ghosting using deep imitation learning. In *MIT Sloan Sports Analytics Conference (SSAC)*, 2017.

Panait, Liviu and Luke, Sean. Cooperative multi-agent learning: The state of the art. *Autonomous agents and multi-agent systems*, 11(3):387–434, 2005.

Papadimitriou, Christos H and Steiglitz, Kenneth. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1982.

Ross, Stephane, Gordon, Geoff, and Bagnell, J. Andrew. A reduction of imitation learning and structured prediction to no-regret online learning. In *Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

Shivaswamy, Pannagadatta K and Jebara, Tony. Permutation invariant svms. In *Proceedings of the 23rd international conference on Machine learning*, pp. 817–824. ACM, 2006.

Stone, Peter. What's hot at RoboCup. In *The AAAI Conference on Artificial Intelligence (AAAI)*, 2016.

Stone, Peter and Veloso, Manuela. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8 (3):345–383, 2000.

Sutton, Richard S and Barto, Andrew G. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Vinyals, Oriol, Bengio, Samy, and Kudlur, Manjunath. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

A. Variational Inference Derivation for Hidden Markov Models

In this section, we provide the mathematical derivation for the structured variational inference procedure. We focus on the training for Bayesian Hidden Markov Model, in particular the Forward-Backward procedure to complete the description of Algorithm 3. The mathematical details for other types of graphical models depend on the family of such models and should follow similar derivations. Further relevant details on stochastic variational inference can be found in (Hoffman et al., 2013; Johnson & Willsky, 2014; Beal, 2003). d

Settings. Given an arbitrarily ordered set of trajectories $U = \{U_1, \dots, U_K, C\}$, let the coordination mechanism underlying each such U be governed by a true unknown model p, with global parameters θ. We suppress the agent/policy subscript and consider a generic featurized trajectory $x_t = [u_t, c_t]\ \forall t$. Let the latent role sequence for the same agent be $z = z_{1:T}$.

At any time t, each agent is acting according to a latent role $z_t \sim \texttt{Categorical}\{\bar{1}, \bar{2}, \dots, \bar{K}\}$, which are the local parameters to the structured model.

Ideally, role and index assignment can be obtained by calculating the posterior $p(z|x, \theta)$, which is often intractable. One way to infer the role assignment is via approximating the intractable posterior $p(z|x, \theta)$ using Bayesian inference, typically via MCMC or mean-field variational methods. Since sampling-based MCMC methods are often slow, we instead aim to learn to approximate $p(z|x, \theta)$ by a simpler distribution q via Bayesian inference. In particular, we employ techniques from stochastic variational inference (Hoffman et al., 2013), which allows for efficient stochastic training on mini-batches that can naturally integrate with our imitation learning subroutine.

Structured Variational Inference for Unsupervised Role Learning. Consider a full probabilistic model:

$$p(\theta, z, x) = p(\theta) \prod_{t=1}^{T} p(z_t|\theta) p(x_t|z_t, \theta)$$

with global latent variables θ, local latent variables $z = \{z_t\}_{t=1}^{T}$. Posterior approximation is often cast as optimizing over a simpler model class $\mathcal{Q}$, via searching for global parameters θ and local latent variables z that maximize the evidence lower bound (ELBO) $\mathcal{L}$:

$$\begin{aligned}\log p(x) &\geqslant \mathbb{E}_q \left[\log p(z, \theta, x)\right] - \mathbb{E}_q \left[\log q(z, \theta)\right] \\ &\triangleq \mathcal{L}\left(q(z, \theta)\right).\end{aligned}$$

Maximizing $\mathcal{L}$ is equivalent to finding $q \in \mathcal{Q}$ to minimize the KL divergence $\text{KL}\left(q(z, \theta|x) || p(z, \theta|x)\right)$.

For unsupervised structured prediction problem over a family of graphical model, we focus on the structured mean-field variational family, which factorizes q as $q(z, \theta) = q(z)q(\theta)$ (Hoffman & Blei, 2014) and decomposes the ELBO objective:

$$\begin{aligned}\mathcal{L} = &\ \mathbb{E}_q[\log p(\theta] - \mathbb{E}_q[\log q(\theta] \\ &+ \mathbb{E}_q[\log(p(z, x|\theta)] - \mathbb{E}_q[\log(q(z))].\end{aligned} \tag{3}$$

This factorization breaks the dependency between θ and z, but not between single latent states z_t, unlike variational inference for i.i.d data (Kingma & Welling, 2013).

Variational inference optimizes the objective $\mathcal{L}$ typically using natural gradient ascent over global factors $q(\theta)$ and local factors $q(z)$. (Under mean-field assumption, optimization typically proceeds via alternating updates of θ and z.) Stochastic variational inference performs such updates efficiently in mini-batches. For graphical models, structured stochastic variational inference optimizes $\mathcal{L}$ using *natural gradient* ascent over global factors $q(\theta)$ and message-passing scheme over local factors $q(z)$. We assume the prior $p(\theta)$ and complete conditionals $p(z_t, x_t|\theta)$ are conjugate pairs of exponential family, which gives natural gradient of $\mathcal{L}$ with respect to $q(\theta)$ convenient forms (Johnson & Willsky, 2014). Denote the exponential family forms of $p(\theta)$ and $p(z_t, y_t|\theta)$ by:

$$\begin{aligned}\ln p(\theta) &= \langle \eta_\theta, t_\theta(\theta) \rangle - A_\theta(\eta_\theta) \\ \ln p(z_t, x_t|\theta) &= \langle \eta_{zx}(\theta), t_{zx}(z_t, x_t) \rangle - A_{zx}(\eta_{zx}(\theta))\end{aligned}$$

where η_θ and η_{zx} are functions indicating natural parameters, t_θ and t_{zx} are sufficient statistics and $A(\cdot)$ are log-normalizers ((Blei et al., 2017)). Note that in general, different subscripts corresponding to η, t, A indicate different function parameterization (not simply a change in variable value assignment). Conjugacy in the exponential family yields that (Blei et al., 2017):

$$t_\theta(\theta) = [\eta_{zx}(\theta), -A_{zx}(\eta_{zx}(\theta))]$$

and that

$$p(\theta|z_t, x_t) \propto \exp\{\langle \eta_\theta + [t_{zx}(z_t, x_t), 1]\, , t_\theta(\theta) \rangle\} \tag{4}$$

Conjugacy in the exponential family also implies that the optimal $q(\theta)$ is in the same family (Blei et al., 2017), i.e.

$$q(\theta) = \exp\{\langle \widetilde{\eta}_\theta, t_\theta(\theta) \rangle - A_\theta(\widetilde{\eta}_\theta)\}$$

for some natural parameters $\widetilde{\eta}_\theta$ of $q(\theta)$.

To optimize over global parameters $q(\theta)$, conjugacy in the exponential family allows obtaining convenient expression for the gradient of $\mathcal{L}$ with respect to natural parameters $\widetilde{\eta}_\theta$. The derivation is shown similarly to (Johnson

& Willsky, 2014) and (Blei et al., 2017) - we use simplified notations $\widetilde{\eta} \triangleq \widetilde{\eta}_\theta, \eta \triangleq \eta_\theta, A \triangleq A_\theta$, and $t(z,x) \triangleq \sum_{t=1}^{T}\left[t_{zx}(z_t,x_t),1\right]$. Taking advantage of the exponential family identity $\mathbb{E}_{q(\theta)}[t_\theta(\theta)] = \nabla A(\widetilde{\eta})$, the objective $\mathcal{L}$ can be re-written as:

$$\begin{aligned}\mathcal{L} &= \mathbb{E}_{q(\theta)q(z)}\left[\ln p(\theta|z,x) - \ln q(\theta)\right]\\ &= \langle \eta + \mathbb{E}_{q(z)}[t(z,x)], \nabla A(\widetilde{\eta})\rangle - (\langle \widetilde{\eta}, \nabla A(\widetilde{\eta})\rangle - A(\widetilde{\eta}))\end{aligned}$$

Differentiating with respect to $\widetilde{\eta}$, we have that

$$\nabla_{\widetilde{\eta}}\mathcal{L} = \left(\nabla^2 A(\widetilde{\eta})\right)\left(\eta + \mathbb{E}_{q(z)}[t(z,x)] - \widetilde{\eta}\right)$$

The *natural gradient* of $\mathcal{L}$, denoted $\widetilde{\nabla}_{\widetilde{\eta}}$, is defined as $\widetilde{\nabla}_{\widetilde{\eta}} \triangleq \left(\nabla^2 A(\widetilde{\eta})\right)^{-1}\nabla_{\widetilde{\eta}}$. And so the natural gradient of $\mathcal{L}$ can be compactly described as:

$$\widetilde{\nabla}_{\widetilde{\eta}}\mathcal{L} = \eta + \sum_{t=1}^{T}\mathbb{E}_{q(z_t)}\{[t_{zx}(z_t,x_t),1]\} - \widetilde{\eta} \tag{5}$$

Thus a stochastic natural descent update on the global parameters $\widetilde{\eta}_\theta$ proceeds at step n by sampling a mini-batch x_t and taking the global update with step size ρ_n:

$$\widetilde{\eta}_\theta \leftarrow (1-\rho_n)\widetilde{\eta}_\theta + \rho_n(\eta_\theta + b^\top \mathbb{E}_{q*(z_t)}[t(z_t,x_t)]) \tag{6}$$

where b is a vector of scaling factors adjusting for the relative size of the mini-batches. Here the global update assumes optimal local update $q^*(z)$ has been computed. In each step however, the local factors $q^*(z_t)$ are computed with mean field updates and the current value of $q(\theta)$ (analogous to coordinate ascent). In what follows, we provide the derivation for the update rules for Hidden Markov Models, which are the particular instantiation of the graphical model we use to represent the role transition for our multi-agent settings.

Variational factor updates via message passing for Hidden Markov Models. For HMMs, we can view global parameters θ as the parameters of the underlying HMMs such as transition matrix and emission probabilities, while local parameters z govern hidden state assignment at each time step.

Fixing the global parameters, the local updates are based on message passing over the graphical model. The exact mathematical derivation depends on the specific graph structure. The simplest scenario is to assume independence among z_t's, which resembles naive Bayes. We instead focus on Hidden Markov Models to capture first-order dependencies in role transitions over play sequences. In this case, global parameters $\theta = (p_0, P, \phi)$ where $P = [P_{ij}]_{i,j=1}^{K}$ is the transition matrix with $P_{ij} = p(z_t = j|z_{t-1} = i)$, $\phi = \{\phi_i\}_{i=1}^{K}$ are the emission parameters, and p_0 is the initial distribution.

Consider a Bayesian HMM on K latent states. Priors on the model parameters include the initial state distribution p_0, transition matrix P with rows denoted $p_1, \ldots, p_K$, and the emission parameters $\phi = \{\phi_i\}_{i=1}^{K}$. In this case we have the global parameters $\theta = (p_0, P, \phi)$. For Hidden Markov Model with observation $x_{1:T}$ and latent sequence $z_{1:T}$, the generative model over the parameters is given by $\phi_i \sim p(\phi)$ (i.i.d from prior), $p_i \sim \mathrm{Dir}(\alpha_i)$, $z_1 \sim p_0, z_{t+1} \sim p_{z_t}$, and $x_t \sim p(x_t|\phi_{z_t})$ (conditional distribution given parameters ϕ). We can also write the transition matrix:

$$P = \begin{bmatrix} p_1 \\ \vdots \\ p_K \end{bmatrix}$$

The Bayesian hierarchical model over the parameters, hidden state sequence $z_{1:T}$, and observation sequence $y_{1:T}$ is

$$\begin{aligned}&\phi_i \overset{\text{iid}}{\sim} p(\phi), p_i \sim \mathrm{Dir}(\alpha_i)\\ &z_1 \sim p_0, z_{t+1} \sim p_{z_t}, x_t \sim p(x_t|\phi_{z_t})\end{aligned}$$

For HMMs, we have a full probabilistic model: $p(z,x|\theta) = p_0(z_1)\prod_{t=1}^{T} p(z_t|z_{t-1},P)p(x_t|z_t,\phi)$. Define the likelihood potential $L_{t,i} = p(x_t|\phi_i)$, the likelihood of the latent sequence, given observation and model parameters, is as follows:

$$p(z_{1:T}|x_{1:T},P,\phi) = \exp\left(\log p_0(z_1) + \sum_{t=2}^{T}\log P_{z_{t-1},z_t} + \sum_{t=1}^{T}\log L_{t,z_t} - Z\right) \tag{7}$$

where Z is the normalizing constant. Following the notation and derivation from (Johnson & Willsky, 2014), we denote $p(z_{1:T|x_{1:T},P,\phi}) = \mathrm{HMM}(p_0,P,L)$. Under mean field assumption, we approximate the true posterior $p(P,\phi,z_{1:T}|x_{1:T})$ with a mean field variational family $q(P)q(\phi)q(z_{1:T})$ and update each variational factor in turn while fixing the others.

Fixing the global parameters θ, taking expectation of log of (7), we derive the update rule for $q(z)$ as $q(z_{1:T}) = \mathrm{HMM}(\widetilde{P}, \widetilde{p}_0, \widetilde{L})$ where:

$$\begin{aligned}\widetilde{P}_{j,k} &= \exp\{\mathbb{E}_{q(P)}\ln(P_{j,k})\}\\ \widetilde{p}_{0,k} &= \exp\{\ln \mathbb{E}_{q(p_0)}p_{0,k}\}\\ \widetilde{L}_{t,k} &= \exp\{\mathbb{E}_{q(\phi_k)}\ln(p(x_t|z_t=k))\}\end{aligned}$$

To calculate the expectation with respect to $q(z_{1:T})$, which is necessary for updating other factors, the `Forward-Backward` recursion of HMMs is defined by forward messages F and backward messages B:

$$F_{t,i} = \sum_{j=1}^{K} F_{t-1,j}\widetilde{P}_{j,i}\widetilde{L}_{t,i} \tag{8}$$

$$B_{t,i} = \sum_{j=1}^{K} \widetilde{P}_{i,j}\widetilde{L}_{t+1,j}B_{t+1,j} \tag{9}$$

$$F_{1,i} = p_0(i)$$

$$B_{T,i} = 1$$

As a summary, calculating the gradient w.r.t z yields the following optimal variational distribution over the latent sequence:

$$q^*(z) \propto \exp\Big(\mathbb{E}_{q(P)}[\ln p_0(z_1)] + \sum_{t=2}^{T}\mathbb{E}_{q(P)}[\log P_{z_{t-1},z_t}] + \sum_{t=1}^{T}\mathbb{E}_{q(\phi)}\ln[p(x_t|z_t)]\Big), \tag{10}$$

which gives the local updates for $q^*(z)$, given current estimates of P and ϕ:

$$\widetilde{P}_{j,k} = \exp\left[\mathbb{E}_{q(P)}\ln(P_{j,k})\right] \tag{11}$$

$$\widetilde{p}(x_t|z_t = k) = \exp\left[E_{q(\phi)}\ln p(x_t|x_t = k)\right], \tag{12}$$

for $k = 1,\dots,K$, $t = 1,\dots,T$, and then use $p_0, \widetilde{P}, \widetilde{p}$ to run the forward-backward algorithm to compute the update $q^*(z_t = k)$ and $q^*(z_{t-1} = j, z_t = k)$. The forward-backward algorithm in the local update step takes $O(K^2T)$ time for a chain of length T and K hidden states.

Training to learn model parameters for HMMs. Combining *natural gradient* step with message-passing scheme for HMMs yield specific update rules for learning the model parameters. Again for HMMs, the global parameters are $\theta = (p_0, P, \phi)$ and local variables $z = z_{1:T}$. Assuming the priors on observation parameter $p(\phi_i)$ and likelihoods $p(x_t|\phi_i)$ are conjugate pairs of exponential family distribution for all i, the conditionals $p(\phi_i|x)$ have the form as seen from equation 4:

$$p(\phi_i|x) \propto \exp\{\langle \eta_{\phi_i} + [t_{x,i}(x), 1], t_{\phi_i}(\phi_i)\rangle\}$$

For structured mean field inference, the approximation $q(\theta)$ factorizes as $q(P)q(p_0)q(\phi)$. At each iteration, stochastic variational inference sample a sequence $x_{1:T}$ from the data set (e.g. trajectory from any randomly sampled player) and perform stochastic gradient step on $q(P)q(p_0)q(\phi)$. In order to compute the gradient, we need to calculate expected sufficient statistics w.r.t the optimal factor for $q(z_{1:T})$, which in turns depends on current value of $q(P)q(p_0)q(\phi)$.

Following the notation from (Johnson & Willsky, 2014), we write the prior and mean field factors as

$$p(p_i) = \text{Dir}(\alpha_i), p(\phi_i) \propto \exp\{\langle \eta_{\phi_i}, t_{\phi_i}(\phi_i)\rangle\}$$
$$q(p_i) = \text{Dir}(\widetilde{\alpha}_i), q(\phi_i) \propto \exp\{\langle \widetilde{\eta}_{\phi_i}, t_{\phi_i}(\phi_i)\rangle\}$$

Algorithm 5 Coordinated Structure Learning
`LearnStructure` $\{U_1,\dots,U_K,C,\theta,\rho\} \mapsto q(\theta,z)$

```
Input: Set of trajectories $U = \{U_k\}_{k=1}^K$. Context $C$
    Previous parameters $\theta = (p_0, \theta^P, \theta^\phi)$, stepsize $\rho$
1: $X_k = \{x_{t,k}\}_{t=1}^T = \{[u_{t,k}, c_t]\}\ \forall t,k.X = \{X_k\}_{k=1}^K$
2: Local update: Compute $\widetilde{P}$ and $\widetilde{p}$ per equation 11 and 12
    and compute $q(z) = \texttt{Forward-Backward}(X, \widetilde{P}, \widetilde{p})$
3: Global update of $\theta$, per equations 16, 17, and 18.
output Updated model $q(\theta, z) = q(\theta)q(z)$
```

Using message passing scheme as per equations (8) and (9), we define the intermediate quantities:

$$\hat{t}_{x,i} \triangleq \mathbb{E}_{q(z_{1:T})}\sum_{t=1}^{T}\mathbb{I}[z_t = i]t_{x,i}(x_t) = \sum_{t=1}^{T}F_{t,i}B_{t,i}[t_{x,i}(x_t), 1]/Z \tag{13}$$

$$(\hat{t}_{trans,i})_j \triangleq \mathbb{E}_{q(z_{1:T})}\sum_{t=1}^{T-1}\mathbb{I}[z_t = i, z_{t+1} = j] = \sum_{t=1}^{T-1}F_{t,i}\widetilde{P}_{i,j}\widetilde{L}_{t+1,j}B_{t+1,j}/Z \tag{14}$$

$$(\hat{t}_{init})_i \triangleq \mathbb{E}_{q(z_{1:T})}\mathbb{I}[z_1 = i] = \widetilde{p}_0 B_{1,i}/Z \tag{15}$$

where $Z \triangleq \sum_{i=1}^{K}F_{T,i}$ is the normalizing constant, and $\mathbb{I}$ is the indicator function.

Given these expected sufficient statistics, the specific update rules corresponding to the natural gradient step in the natural parameters of $q(P)$, $q(p_0)$, and $q(\phi)$ become:

$$\widetilde{\eta}_{\phi,i} \leftarrow (1-\rho)\widetilde{\eta}_{\phi,i} + \rho(\eta_{\phi,i} + b^\top \hat{t}_{x,i}) \tag{16}$$

$$\widetilde{\alpha}_i \leftarrow (1-\rho)\widetilde{\alpha}_i + \rho(\alpha_i + b^\top \hat{t}_{trans,i}) \tag{17}$$

$$\widetilde{\alpha}_0 \leftarrow (1-\rho)\widetilde{\alpha}_0 + \rho(\alpha_0 + b^\top \hat{t}_{init,i}) \tag{18}$$

B. Experimental Evaluation

B.1. Batch-Version of Algorithm 2 for Predator-Prey

B.2. Visualizing Role Assignment for Soccer

The Gaussian components of latent structure in figure 7 give interesting insight about the latent structure of the demonstration data, which correspond to a popular formation arrangement in professional soccer. Unlike the predator-prey domain, however, the players are sometimes expected to switch and swap roles. Figure 8 displays the tendency that each learning policy k would takes on other roles outside of its dominant mode. Policies indexed $0-3$ tend to stay most consistent with the prescribed latent roles. We observe that these also correspond to players with the least variance in their action trajectories. Imitation loss is

Algorithm 6 Multi-Agent Data Aggregation Imitation Learning
`Learn`$(A_1, A_2, \dots, A_K, C|D)$

```
Input: Ordered actions $A_k = \{a_{t,k}\}_{t=1}^T \; \forall k$, context $\{c_t\}_{t=1}^T$
Input: Aggregating data set $D_1, .., D_K$ for each policy
Input: base routine Train$(S, A)$ mapping state to actions
1: for $t = 0, 1, 2, \dots, T$ do
2:     Roll-out $\hat{a}_{t+1,k} = \pi_k(\hat{s}_{t,k}) \quad \forall$ agent $k$
3:     Cross-update for each policy $k \in \{1, \dots, K\}$
       $\hat{s}_{t+1,k} = \varphi_k\left([\hat{a}_{t+1,1}, \dots, \hat{a}_{t+1,k}, \dots, \hat{a}_{t+1,K}, c_{t+1}]\right)$
4:     Collect expert action $a^*_{t+1,k}$ given state $\hat{s}_{t+1,k} \quad \forall k$
5:     Aggregate data set $D_k = D_k \cup \{\hat{s}_{t+1,k}, a^*_{t+1,k}\}_{t=0}^{T-1}$
6: end for
7: $\pi_k \leftarrow$ Train$(D_k)$
output $K$ new policies $\pi_1, \pi_2, \dots, \pi_K$
```

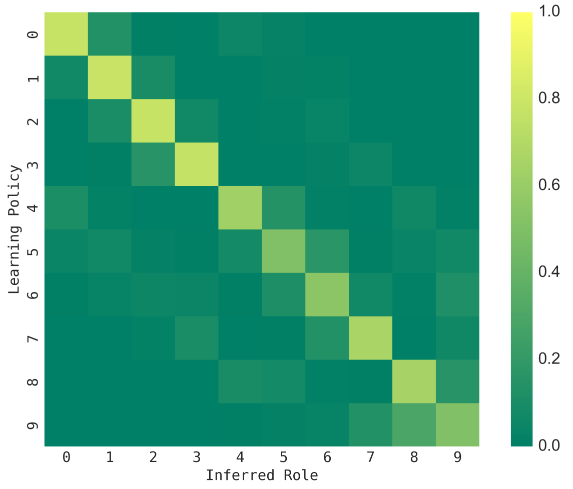

Figure 8. Role frequency assigned to policy, according to the maximum likelihood estimate of the latent structured model

generally higher for less consistent roles (e.g. policies indexed $8-9$). Intuitively, entropy regularization encourages a decomposition of roles that result in learning policies as decoupled as possible, in order to minimize the imitation loss.