Unsupervised Plan Detection with Factor Graphs

George B. Davis
School of Computer Science
Carnegie Mellon University
gbd@cs.cmu.edu

Jamie Olson
School of Computer Science
Carnegie Mellon University
jolson@cs.cmu.edu

Kathleen M. Carley
School of Computer Science
Carnegie Mellon University
carley@cs.cmu.edu

ABSTRACT

Recognizing plans of moving agents is a natural goal for many sensor systems, with applications including robotic pathfinding, traffic control, and detection of anomalous behavior. This paper considers plan recognition complicated by the absence of contextual information such as labeled plans and relevant locations. Instead, we introduce 2 unsupervised methods to simultaneously estimate model parameters and hidden values within a Factor graph representing agent transitions over time. We evaluate our approach by applying it to goal prediction in a GPS dataset tracking 1074 ships over 5 days in the English channel.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

1. INTRODUCTION

Many real world sensor networks are capable of simultaneously tracking many agents as they navigate a space. Examples include network-linked GPS devices in phones and vehicles, passive tracking mechanisms such as radar (when paired with technology to distinguish agent identities), and entry-point systems such as keycard and RFID scanners. (Many additional examples emerge when we consider agents navigating virtual spaces such as the World Wide Web, but this paper will concentrate on focus on a physical system). As the volume and complexity of data produced by these systems has grown, human monitors are increasingly dependent on algorithms that can efficiently extract relevant patterns for their analysis.

One successful approach to pattern mining in this domain has been to presume the existence of hidden variables which mediate the transitions observed by sensors. For example, there may be a hidden activity that explains the time an agent spends at a certain location, or an underlying plan that explains the choice to travel from one place to another. Probabilistic relationships between hidden variables and observed variables can be encoded with graphical models such as Conditional Random Fields (CRFs), which support efficient algorithms for inferring missing values. Previous work have used this general approach to predict future agent actions and detect surprising deviations from typical behavior [8]. Applications have been discussed in contexts ranging from robotic and human planning [2] to assistance of seniors and disabled individuals [7].

Inference from this type of model is generally preceded by a training phase, in which model parameters are optimized against a dataset for which "true" values of hidden variables have been provided. In previous experiments, training data were drawn from journals of experiment participants, hand-coded by human oservers, or extracted from existing databases (*e.g.* maps of known locations or obstacles). In this paper, we examine a case where such data would be useful but is unavailable. Our dataset tracks the movement of 1700 merchant marine vessels servicing ports in the English channel over a 5 day period. Maritime navigation is influenced by "soft" conventions, such as shipping lanes and waypoints, rather than fixed constraints, such as roads or walls. Because some conventions differ between nationalities, companies, and ship types, there is no single collation that could be built directly into our model. The same diversity of backgrounds and conventions would make conducting a survey with reasonable accuracy and breadth cost-prohibitive.

We believe our domain is one of many for which assuming the existence of training data is unrealistic. Others include covert applications where subjects cannot be polled, large scale applications where surveys would be expensive or inaccurate, and evolving environments in which training data quickly becomes outdated. As a solution we propose and unsupervised approach to graphical models, allowing the user to exploit knowledge of the structure of hidden variables in a system without observing them - even during training. Instead, we introduce 2 algorithms that simultaneously assigns variable values and optimizes model parameters in a way that is self-consistent given the model structure. Our model and algorithm are simpler than the most advanced supervised methods, but they nonetheless give compelling results on a goal prediction task, and serve to demonstrate a variety of challenges involved in creating an unsupervised approach.

The rest of this paper is organized as follows. In section 2, we review undirected graphical models of distributions, including the relationship between our chosen representation, the factor graph (FG), and the more commonly discussed conditional random field (CRF). We also review prior applications of CRFs to plan detection and introduce our notation and dataset. In section 3 we discuss new challenges involved in unsupervised inference and introduce our own factor graph and algorithms. In section 4, we empirically evaluate our approach with results regarding the intermediate variables, the goal prediction task, and convergence rates of the algorithm. In section 5 we summarize our contributions and discuss future work in this area.

2. BACKGROUND

2.1 Factor Graphs

What follows is a terse introduction to the rich topic of undirected probabilistic graphical models (PGMs). As our primary goal is to introduce our notation, we refer the reader to [6] for a more thorough discussion. We restrict discussion of supervised learning and inference to a brief comparison in section 3, when we describe our unsupervised approach. For deeper background on that topic, we defer to [12].

Consider a set X of random variables, each of which can take one of at most n discrete states. A joint assignment to X is notated with the integer vector $\vec{x} \in \{\mathbb{Z}^+ \leq n\}^{|X|}$. We can parameterize the joint PMF $P(X = \vec{x})$ with a vector $\vec{w} \in \mathbb{R}^d$ whose entries specify the probability of each combination of value assignments to X. This naive representation would require $d = n^{|X|} - 1$ parameters, a quantity too great to store, fit to a dataset, or conduct inference from unless X is trivially small. The goal of a factor graph (FG) is to allow the required number of parameters to scale better with $\mid X \mid$ by identifying a set of functions with smaller domains which factorize the PMF.

Formally, a factor graph G on X can be represented by the tuple $G = \langle F, D \rangle$ where each $f \in F$ is a potential function $f : \mathbb{Z}^{|D(f)|} \rightarrow \mathbb{R}$ giving the local likelihood of a vector of values for the variables retrieved by the set-valued domain function $D : f \in F \rightarrow S \subset X$. Notating as $\vec{x^f}$ the integer vector extracting values for variable in $D(f)$ from $\vec{x}$, we can rewrite the PMF via its factorization,

$$P\left(X = \vec{x}\right) = \frac{1}{z} \prod_{f \in F} f\left(\vec{x^f}\right) \tag{1}$$

where z is a normalizing constant. The "graph" in factor graph is the bipartite network between factors and the variables in their domains. There is a corresponding single mode network between variables, the Markov network for G, whose neighborhood function (which also gives the *Markov blanket*) is

$$N(x) = \left(\bigcup_{f \in D^{-1}(x)} D(f) \right) \setminus x \tag{2}$$

The precise relationship between the factor graph and this Markov network is that they induce the same set of conditional independences,

$$\forall_{x, x' \in X}, \; x \perp x' \mid N(x) \tag{3}$$

In fact, any Markov network can be written as a factor graph in which there is one factor per variable, with domain equal to that variable plus its neighborhood. However, by breaking a neighborhood into multiple factors, a factor graph can describe additional structure in potential functions, yielding a tighter bound on the number of parameters necessary. In the worst case, representing a factor f requires a parameter vector $\vec{w^f}$ with dimension $n^{|D(f)|}$ to enumerate potentials for all combinations of variable values in its domain. Definition (2) ensures that there exists a value k satisfying

$$\max_{f \in F} |D(f)| \; = \; k \; \leq \; \max_{x \in X} |N(x)| \;\; , \tag{4}$$

which allows us to bound the dimensionality of the full parameter vector to be exponential only in k:

$$d = \sum_{f \in F} n^{|D(f)|} \;\; \leq \;\; \mid F \mid n^k \tag{5}$$

Some distributions can be further abbreviated using *parameter sharing* between factors. For example, most hidden Markov models (HMMs) apply the same set of transition probabilities to any pair of sequential states. To annotate this we replace the domain function $D(f)$ with an instantiation function $I(f)$ which returns a set of variable sets, the *factor instance domains* on which the factor is instantiated. Repeated instantiation of factors does not affect the number of parameters necessary, but the PMF becomes

$$P\left(X\right) = \frac{1}{z} \prod_{f \in F} \prod_{I \in I(f)} f\left(\vec{d^I}\right) \tag{6}$$

In some applications, there is a clear division between the set X of *hidden* variables, whose values must be inferred, and a second set Y of *observed* variables. Better performance can often be achieved for these cases using *discriminative* models, which represent only the conditional distribution $P(X \mid Y)$ and not relationships between observable variables. Incorporating observed variables into our notation requires no immediate adjustments, but each factor instance domain $D \in I(f)$ may now include observed variables (but must still contain at least one hidden variable). Furthermore, observed variables may be continuous so long as each factor involving them has a potential function with finite parameterization (preferably, one compact enough to maintain the bound in (5)).

The discriminative version of the Markov network given by (2) is known as a Conditional Random Field (CRF), and is the most common model for previous work in plan detection. We use factor graphs in this paper because they generalize many other models and allow simpler notation for our methods.

2.2 Sensor Logs and PGMs

This paper builds on a growing body of work applying PGMs to *sensor logs*. In our context, a sensor log is an observation set O, in which each member $\vec{o_t^a} \in O$ is a vector tagged with timestamp t and subject agent a. The vector itself includes all state information observed by the sensors, which is generally spatial context such as position and speed.

Records of this form obviously record only data visible to sensors, and in so doing break into discrete observations what agents experience continuously. Graphical models of sensor logs attempt to restore hidden state and continuity by learning relationships between co-temporal and sequential variables. The prototypical PGM of this form is the Hidden Markov model (HMM), which can be encoded as two-factor FG. The first factor, $f_S(s_t^a, \vec{o_t^a})$ measures the likelihood of $\vec{o_t^a}$ being observed if s_t^a is the underlying state. The second factor, $f_T(s_t^a, s_{t+1}^a)$, measures the likelihood of transitioning between two discrete hidden states. Work in the past decade has built from this skeleton into more advanced models that discriminate many types of hidden state and exploit a variety of local information.

Ashbrook and Starner [2] fit HMMs to GPS sensor logs of users in their daily travel routines, and confirmed non-random patterns in the transition probabilities. Nguyen *et al.* [9][10], and later [4], developed successively more complex models using additional hidden layers to represent higher level activities (those lasting multiple observations) in a controlled kitchen environment. Our own work follows most closely on that of Liao *et al.*, who have used CRFs [7] and hybrid graphical models [8] to estimate activities, place types, transportation modes, goal coordinates, and occurrence of behavioral deviations ("novelty") in agents navigating an urban environment.

Since neither our work nor prior models involve relationships or interactions between agents, the corresponding factor graphs can be broken down into separate components for each agent. Previous approaches have generally chosen to train separate models for each agent, save for parameters learned off-line such as sensor error models. [7] experimented with making predictions for one agent based on a model trained on another agent, with some success. Because our dataset has only a 5-day duration, it was necessary for us to smooth results over all agents.

Notably, there are several examples of previous work in which the space of hidden states is drawn from data rather than directly provided. In [2], GPS observations were initially clustered to derive a finite state space (though they were spot-validated by test subjects for accuracy). In [7], training data provided examples of some locations labeled with types, but additional locations were identified by adding them when inference suggested their existence (according to a hard-coded criterion). The distinction between these methods and the unsupervised learning we propose in this paper is that they treat state identification as a separate clustering step specific to their model, distinct from the general purpose algorithms used during supervised training. Although there are many other instances in literature of unsupervised learning algorithms for specific models, this paper is the first work of which we are aware discussing unsupervised algorithms applicable to factor graphs in general.

2.3 AIS Data

For 5 days in June 2005, a sensor network queried Automated Identification System (AIS) transponders on merchant marine vessels navigating the English Channel. The Automated Identification System (AIS) is a communication standard for ocean vessels used by ships and ground stations to coordinate shipping traffic. AIS transponders on compliant vessels are integrated with the ship's radio, GPS, and navigational control systems. When pinged (via broadcast messages from other boats or ground stations), the transponder replies with a radio packet containing ship identity, current GPS coordinates, heading, speed, and various other fields describing navigational state, destination, and more. AIS compliance is required on ships over a certain size by most commercial ports, making it essential for most sizable merchant vessels operating worldwide.

In total, the sensor sweep captured movements of over 1700 vessels were recorded, with activities ranging from simple shipping lane traversals to apparently complex itineraries with stops at multiple ports of call. The reasons for the collection of the data are primarily security related. The global shipping system plays a prominent role in a variety of terrorist attack scenarios, both in the United States and abroad: in any country, the ports are both the most likely means of entry for bombs and other weapons, and themselves a prime economic and symbolic target. In addition to being an attractive target, ports are currently considered unsecure – for example, it has been suggested that only 3% of shipping containers entering the United States are directly inspected by customs officials. The sheer volume of commerce conducted via international shipping makes nave attempts at greater security infeasible, as neither the direct costs associated with detailed surveillance nor the indirect costs incurred by reducing industry efficiency are easily absorbed. If automated techniques such as those designed above can give insight into the behavioral patterns and structural features of the merchant marine population, then limited budgets for surveillance and interdictions can be more precisely targeted to have the greatest impact on overall security. The data under analysis here is especially promising as it represents the result of a relative inexpensive, passive, and consensual surveillance effort.

In many sensor datasets, physical limitations of the sensors are a primary source of error; for example, an error of 10m in a car-installed GPS system can introduce ambiguity as to which street the car is on. In the case of AIS data, the physical error of sensors is so small compared to the scale of navigation (some tankers are themselves 400m long) that a sensor error model is less relevant. Instead, a primary source of error comes from creative utilization of user-input fields such as destination and navigational status. We chose to focus only on numeric fields that would be drawn directly from navigational computers. Even on this set, there were many cases of misconfigurations which, for example, reported 0 latitude and 0 longitude for the study duration. We preprocessed to eliminate all ships with constant values for any numeric field.

AIS responses in the original dataset were intermittent with inconsistent inter-arrival times. Although work exists regarding the use of temporally irregular observations(e.g. [4]), we avoid these issues. Instead, we filter the data to produce streams of observations in which at least 45 minutes and at most 180 minutes passes between observations. We also remove ships that make fewer than 5 consecutive movements, yielding a dataset of 10935 tracks of 576 ships. We also remove 140 erroneous responses sent by malfunctioning or otherwise corrupted responders. Figure 1 shows the final dataset visualized in Google Earth [1].

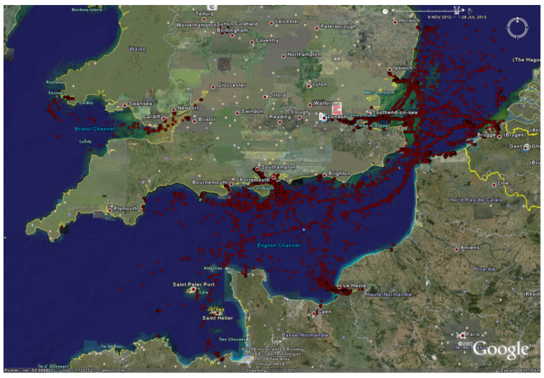

Figure 1: AIS sensor data from the English channel

3. METHODS

3.1 Factor Graphs for AIS

The trend in previous work has been to provide increasingly complex graphical models to incorporate additional sensor data (e.g. use of street maps in [8]) or knowledge regarding relationship structure (e.g. modeling of activity duration by [4]). In order to concentrate on unsupervised learning, we employed the relatively simple, two-layer model shown in figure 2. The variables in our factor graph include:

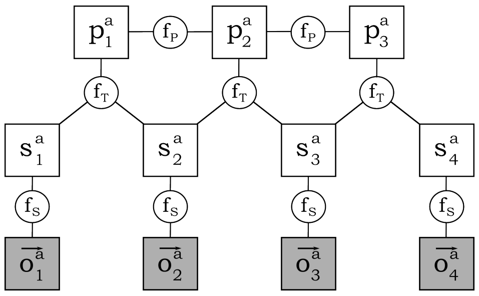

Figure 2: Plan Prediction Factor Graph

- $\vec{o_t^a}$ is an *observed* vector in $\mathbb{R}^3$ containing the latitude, longitude, and speed of agent a at time t,
- s_t^a is a discrete *state* variable representing the instantaneous state of an agent. It takes on integer values $0 \le s_t^a < n_s$.
- p_t^a is a discrete *plan* variable capturing an internal agent state persisting over several time periods. It takes on integer values $0 \le p_t^a < n_p$

The following factors model relationships in our graphs.

- $f_S(s_t^a, \vec{o_t^a})$ is a *state compatibility* factor which measures the likelihood of observation $\vec{o_t^a}$ being generated when within state s_t^a. f_S is implemented by maintaining n_s Gaussians, so that $f_S(c, \vec{o_t^a})$ is equal to the probability density at $\vec{o_t^a}$ of distribution $\mathcal{N}(\mu_c, \Sigma_c)$ where the mean vectors and covariance matrices for each Gaussian comprise the factor parameters. To avoid overfitting the Gaussians corresponding to infrequently observed states, each one is initialized with a a mean prior drawn from a uniform distribution over the range of latitudes, longitudes and speeds. The prior covariance is the covariance matrix for the same uniform distribution.
- $f_T(s_t^a, s_{t+1}^a, p_t^a)$ is a *state transition* factor which measures the likelihood of transitioning from s_t^a to s_{t+1}^a. This likelihood is mediated by the plan state p_t^a, representing (for example) the propensity of an agent to select a different route when targeting a different destination. This factor is parameterized as a likelihood table for all possible transitions, and is initialized with a uniform prior to ensure that a minimal probability remains for unobserved transitions.
- $f_P(p_t^a, p_{t+1}^a)$ is a *plan transition* factor which measures the likelihood of switching from p_t^a to p_{t+1}^a. Whereas the state transition factors capture physical constraints (the need to move between continuous states), the primary purpose of the plan transition factor is to model a time scale on how frequently agents are expected to change objectives. This factor has a single parameter, the change probability, which we initialize to .2 to indicate an expected time scale of plans being maintained for approximately 5 hours (the average time-at-sea we observed in ships that went to port). Although this parameter (and therefore the time-scale of a plan) can be changed during training, this initial setting plays an important role in determining which maximal labeling we will reach. This is discussed further in section 3.2.

3.2 Unsupervised Learning for Factor Graphs

During *supervised* learning, factor parameters are generally found maximizing the expectation of some training set $T = \vec{t}$.

$$\vec{w}^*(\vec{t}) = \underset{\vec{w}}{\text{argmax}}\, P_{\vec{w}}(T = \vec{t}) \tag{7}$$

Maximum likelihood estimation (MLE) can then be performed by finding the assignment to hidden variables X that has maximum likelihood under factor parameters $\vec{w}^*$. [1]

[1]In most applications the training sets X and T may be different sizes or even "shapes" in terms of relations between variables. However, if a generator is provided for instantiating the same factors on both sets, parameter sharing allows us to reuse a single parameter vector.

$$\vec{x}^*(\vec{w}) = \operatorname*{argmax}_{\vec{x}} P_{\vec{w}^*(\vec{t})}\left(X = \vec{x}\right) \tag{8}$$

In the unsupervised case, no true values $\vec{t}$ are provided, preventing sequential learning and inference. As an alternative goal, we seek to find an assignment satisfying the fixed point of (7) and (8):

$$\vec{x}^* = \operatorname*{argmax}_{\vec{x}} P_{\vec{w}^*(\vec{x}^*)}(X = \vec{x}) \tag{9}$$

To compare the many possible solutions to (9), we introduce the *self-consistency likelihood*, a scoring function favoring assignments which receive high probability under their own optimal parameter values:

$$\bar{L}(\vec{x}) = P_{\vec{w}^*(\vec{x})}(X = \vec{x}) \tag{10}$$

The global maximum of $\bar{L}$ is the fixed point with maximum self-consistency. However, finding it is challenging on several levels. First, the space of possible assignment vectors is far too large (size $n^{|X|}$) to enumerate or sample meaningfully. Second, evaluating $\bar{L}(\vec{x})$ is expensive: one must first compute the parameter values $\vec{w}^*(\vec{x})$, and then the partition constant z for the corresponding distribution.

Algorithms for supervised inference on PGMs face the same challenges above, and most overcome them using local search informed by the graphical structure. For example, the max-residual belief propagation (MRBP) algorithm maintains a set of messages corresponding to graphical ties, and incrementally update message values in a way that it is guaranteed to reduce a free energy quantity. Unfortunately, these methods cannot be directly applied to maximize our target, $\bar{L}$. Whereas changing the value of a variable x in a graph with fixed factor parameters affects only local likelihoods, it can potentially effect *all* factor instances used to calculate $\bar{L}$. This is because the change may affect the optimal parameter settings for all factors for which x participates in an instance. An alternate way to describe this effect is that the distribution $\bar{P}$ achieved by normalizing $\bar{L}$ no longer induces the independences given in (3) – the Markov blanket for x under $\bar{P}$ includes all variables with which it shares a *factor*, not an instance.

However, we can offer a preliminary argument regarding a bound on the impact of these "long range effects". Let $w^{f*}(\vec{x})$ be the optimal parameter assignments for a single factor under assignments $\vec{x}$, and let $\vec{x} \leftarrow (x, c)$ be an operator returning an updated assignment vector with variable x set to state c. Now consider the condition

$$\forall_{x,c} \lim_{|I(f)|\to\infty} w^{f*}(\vec{x}) - w^{f*}(\vec{x} \leftarrow (x, c)) = 0 \tag{11}$$

In other words, as the number of instances of a factor grows, the incremental change to optimal parameters caused by changing the value of a single variable approaches zero. Many common factor parameterizations satisfy this condition, including those we use and list in section 3.1 (modulo the assumption that we observe all Gaussians and state transitions a sufficient number of items). Under this condition, the effect under $\bar{P}$ that changing x has on local likelihoods outside $N(x)$ becomes negligible as our graph becomes larger.

Armed with this intuition, we define a local search with an operator $\delta : \mathbb{Z}^{|X|} \to \mathbb{Z}^{|X|}$, which produces a sequence of assignment vectors following $\vec{x^i} = \delta(\vec{x^{i-1}})$. If δ is such that

$$P_{\vec{w}^*(\vec{x})}(X = \delta(\vec{x})) \geq P_{\vec{w}^*(\vec{x})}(X = \vec{x}) \tag{12}$$

then its fixed point must satisfy (9) as well (assuming that it does not trivially self-cycle). In the following subsections we introduce two operators that satisfy this condition, but have different properties in terms of convergence rate and susceptibility to local maxima while maximizing $\bar{L}$.

Asynchronous EM

One way graphical models support efficient computation is by defining marginal distributions $P(x \mid N(x))$ that can be efficiently computer. This allows quick updates to be designed for Gibbs samplers and belief propagation algorithms [12]. Our first local search method exploits this to improve an assignment vector incrementally by setting one variable at a time to the value with maximum expectation under the current state. The successor is

$$\delta_A(\vec{x}^t) = \vec{x}^t \leftarrow \left(x^t, \operatorname*{argmax}_{c} P_{\vec{w}*(\vec{x}^t)}\left(X = \vec{x}^t \leftarrow (x^t, c)\right)\right), \tag{13}$$

where x^t is drawn from a round robin schedule established in advance. This operator is easy to implement for our graph because our factors support incremental updates: changing the value of x changes only factor instances $I^{-1}(x)$, and each of our factors can be readjusted to give maximum expectation to a new instance assignment in constant time. When describing an iteration of the algorithm we include one update for each variable, in order to standardize the unit of work by graph size. Pseudocode for an implementation of δ_A can be found as Algorithm 1.

The initial assignments $\vec{x}^0$ are selected in the following way. First, a vector of random assignments $\vec{x}'$ is established. Then, each variable is set to its maximum likelihood value with neighbor variables assigned according to $\vec{x}'$ using the prior factor parameters. This "stacking" of the initial state assures that initial factor parameters fully explore the range of possible values they can take on. In testing, we found that making sure that initial parameters were distributed was essential to avoiding bad local maxima. For example, maximizing initial factor parameters against a random allocation vector tended to intialize all Gaussians in state factors to have means near the actual mean coordinates for the data. This initial clustering resulted in poor exploration of the state space, with most clusters remaining near the map center even after many iterations.

Algorithm 1 ASYNCHRONOUS

```
$\vec{w}^0 \leftarrow$ Draws from prior
$\vec{x}^0 \leftarrow$ Random allocation
$t \leftarrow 1$
loop
    $\vec{w}^t \leftarrow \vec{w}^{t-1}$
    $\vec{x}^t \leftarrow \vec{x}^{t-1}$
    for all $x \in X$ do
        $\vec{x}^t \leftarrow (\vec{x}^t \leftarrow x, \text{argmax}_c\, P_{\vec{w}^t}\left(X = \vec{x}^t \leftarrow (x^t, c)\right))$
        $\vec{w}^t \leftarrow \text{argmax}_{\vec{w}}\, P_{\vec{w}}(X = \vec{x}^t)$ (local updates)
    end for
    $t \leftarrow t + 1$
end loop
```

Algorithm 2 SYNCHRONOUS

```
$\vec{w}^0 \leftarrow$ Draws from prior
$t \leftarrow 0$
loop
    $\vec{x}^t \leftarrow \text{MLE}_{\vec{w}}(\vec{x})$ (calls MLBP)
    $\vec{w}^{t+1} \leftarrow \text{argmax}_{\vec{w}}\, P_{\vec{w}}(X = \vec{x}^t)$
    $t \leftarrow t + 1$
end loop
```

Synchronous EM

Our second operator synchronously updates all variable values to a maximum likelihood estimate under the current factor parameters:

$$\delta_S(\vec{x}^t) = \text{MLE}_{\vec{w}^*(\vec{x}^t)}(X) \tag{14}$$

This is analogous to a standard EM algorithm, in which cluster assignments and cluster parameters are updated in an alternating fashion. We hypothesized that taking larger steps in the space of assignment vectors might make us less susceptible to local minima. However, by changing assignments to many variables at once, we may be less protected by the guarantee in (11).

Pseudocode for this method is listed as Algorithm 2. We initialize cluster parameters with priors as we did for the asynchronous method, but it is unnecessary to initiate the first state as we will be using maximum likelihood belief propagation, which depends only on observed variables and factor parameters. Then, at each step, we conduct inference with the current factor parameters using MLBP. Finally, we re-optimize factor parameters to the new assignment.

3.3 Plan Projection Experiment

We designed an experiment to simulate our system's performance at a fundamental task: using the estimated plan of an agent at sea to predict where it will next make port. Our experiment proceeds in two phases. First, we perform unsupervised learning on a test set representing sequences that would have occurred prior to some test sequences, as well as on the first potion of the test sequences themselves. Then, using the labels and factor parameters assigned during the first phase, we sample a distribution of future states for the test set, in order to estimate its next stop.

To create a dataset of sequences appropriate to this task, we developed the following test. First, we included only observations from ships with five consecutive observations "in motion" (reported velocity over 1 km / h) to eliminate a large percentage of ships that did not move often enough to assist training of transition probabilities. Since our model does not explicitly address the duration between observations, we standardized this quantity by eliminating sequences whose inter-observational interval was outlying (over 3 hours). A total of 13715 individual observations fell into sequences in this category. Then, for the test set, we isolated the 457 subsequences within the test set that consisted of routes beginning in motion and ending stopped, with at least 5 segments in between. The criterion on test sequence length is the only one of these filter that could not be applied without full knowledge of the test set, but was necessary to ensure that each test sequence A) was long enough for us to instantiate a factor graph with all factors on, and B) had a buffer beyond this so that we would be forced to predict multiple state transitions.

To calculate a maximum likelihood estimate for the next portfall of a particular test ship, we appended 100 additional hidden plans and states (along with associated factors) to the section of the ship's factor graph which was optimized during training. We then ran Gibbs a sampler on these hidden states using the factor parameters learned during training. Every 1000 iterations we would extract a prediction from the sampler by recording the mean position of the first state whose expected velocity was under 1 km / h.

4. RESULTS AND ANALYSIS

Visual inspection of the locations and transition probabilities learned by our algorithm confirms that it produces a coarse but credible summary of traffic flow in the channel. Figure 3 shows one model trained with our asynchronous algorithm and visualized using Google Earth. Vertexes are placed at the Gaussian mean for each state, with edges placed between transitions with high probability under any plan.

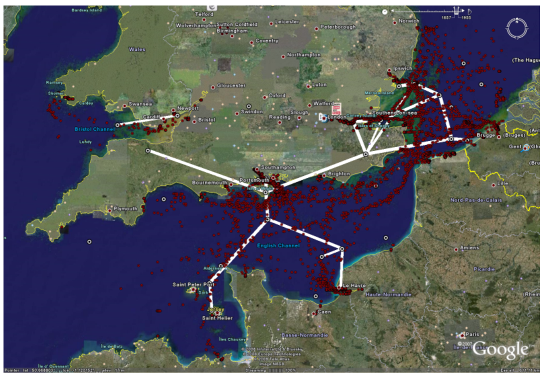

Figure 3: Learned plans and hidden states overlaid on AIS observations

To measure accuracy on our portfall prediction task, we computed the surface distance between the predicted destination and the actual portfall associated with each prediction and plotted the inverse cumulative density for this figure as Figure 4. The curve summarizes a set of probably approximately correct (PAC) bounds for the estimator. For example, models trained with the asynchronous algorithm achieved accuracy under 100km 71% of the time. Synchronous models had only a 53% chance of acheiving this accuracy.

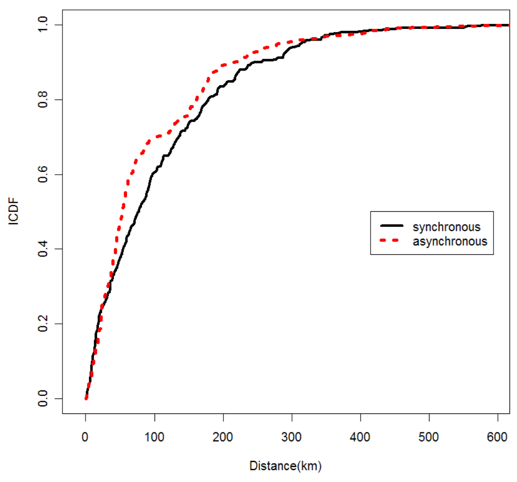

Figure 4: Inverse cumulative density function for error

Another important factor in algorithm choice for probabilistic graphical models is time to convergence. We measured this by counting the number of variables updated in each iteration of the algorithm. To minimize the impact of random starting configuration, we ran 5 trials to 20 iterations with each algorithm, producing the mean updates and error bars shown in figure 5.

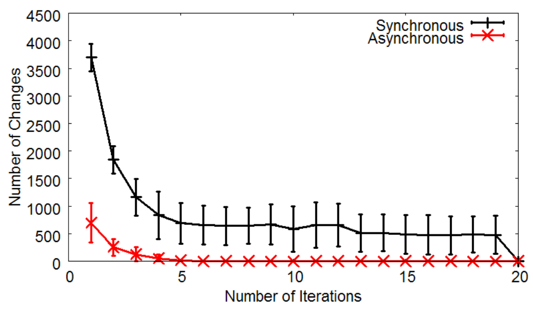

Figure 5: Convergence rates for the two learning algorithms

Overall, the predictions made by the model were well short of the accuracy needed for most real world applications of this system. For example, if the goal was to meet the ship upon portfall, then in many parts of the English channel there would be several potential ports within the 100km radius mentioned above. However, the results do show that asynchronous updates dominate synchronous updates in terms of both probable approximate correctness and convergence rate. We were surprised to find that after only 4 cycles of updates the asynchronous algorithm reached a fixed point in most cases. In contrast, the synchronous algorithm seemed prone to cycles in which each iteration toggled significant number of predictions even after 20 iterations.

5. CONCLUSION AND FUTURE WORK

We presented synchronous and asynchronous expectation maximization algorithms for unsupervised learning in factor graphs. We used these algorithms with a factor graph interpreting AIS data in order to simultaneously detect a map, hidden plans, and transition frequencies between plans. To our knowledge, this was the first report applying general purpose unsupervised algorithms for graphical models to conduct learning with real data. We used the learned models to make projections of portfalls for ships in motion. Although these preliminary results were not accurate enough for real-world application, both prediction accuracy and direct inspection of learned locations and transition probabilities suggested that a reasonable model was being inferred. Our asynchronous method significantly outperformed our synchronous method in terms of both convergence rate and probability of achieving high accuracy in portfall prediction.

In section 3.2 we laid out some principle objectives for unsupervised learning in factor graphs, which we hope can assist future research on this topic. In particular, we would like to make more rigorous the argument associated with (11), so that in future work we can explore properties of the self-consistency probability $\bar{P}$ rather than focusing only on local search. Our appendix contains some initial work in this direction, as we will be exploring $\bar{P}$ as a distribution over partitionings rather than assignment vectors.

There are many areas in which we believe our work can be extended to take advantage of recent developments in supervised learning on PGMs. We are particularly interested in creating unsupervised variants of belief propagation algorithms, where the theory regarding convergence has advanced significantly. The residual belief propagation of [5], in which messages are updated on a schedule that prioritizes "poorly fit" variables, seems especially relevant to our clustering application. In our experiments we saw consistently that some regions of the map stabilized very quickly in terms of cluster locations and transition probabilities, while others were left to slowly improve over many iterations. The result was that our algorithms spent significant computation considering updates for already stable assignments.

The primary focus of our current research is adaptation of algorithms for hierarchical Bayes models [11] to general factor graphs. These models support unsupervised learning with even fewer assumptions: the number of states for a variable class is derived from data and a Dirichlet prior rather than specified in advance. The infinite hidden Markov model of Beal *et. al.* [3] is especially similar in structure to our temporal factor graph, and supports a Gibbs sampler to estimate marginal probabilities in addition to a maximum likelihood assignment.

To improve our AIS application, we are working on algorithms which detect significant locations with higher granularity. This involves both computational challenges and issues of insufficient data, as some regions of the map are far better represented than others in our dataset. Our current experiments in this direction involve hierarchical Gaussian mixtures to allow finer grained notions of location only in areas where there is supporting data. Another important direction is to expand our model to include additional information, such as ship class, heading, or even ownership. Doing so will give us an opportunity to examine how our algorithms scale with more and different types of factors.

6. ACKNOWLEDGMENTS

This work was supported in part by the National Science Foundation under the IGERT program (DGE-9972762) for training and research in CASOS, and the Office of Naval Research under Dynamic Network Analysis program (N00014-02-1-0973, ONR N00014-06-1-0921, ONR N00014-06-1-0104). Additional support was provided by CASOS - the Center for Computational Analysis of Social and Organizational Systems at Carnegie Mellon University.

7. REFERENCES

[1] Google earth. http://earth.google.com/.

[2] D. Ashbrook and T. Starner. Using gps to learn significant locations and predict movement across multiple users. *Personal Ubiquitous Comput.*, 7(5):275–286, 2003.

[3] M. J. Beal, Z. Ghahramani, and C. E. Rasmussen. The infinite hidden Markov model. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2002.

[4] T. V. Duong, H. H. Bui, D. Q. Phung, and S. Venkatesh. Activity recognition and abnormality detection with the switching hidden semi-markov model. In *CVPR '05: Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1*, pages 838–845, Washington, DC, USA, 2005. IEEE Computer Society.

[5] G. Elidan, I. McGraw, and D. Koller. Residual belief propagation: Informed scheduling for asynchronous message passing. In *Proceedings of the Twenty-second Conference on Uncertainty in AI (UAI)*, Boston, Massachussetts, July 2006.

[6] D. Koller, N. Friedman, L. Getoor, and B. Taskar. Graphical models in a nutshell. In L. Getoor and B. Taskar, editors, *Introduction to Statistical Relational Learning*. MIT Press, 2007.

[7] L. Liao, D. Fox, and H. Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *Int. J. Rob. Res.*, 26(1):119–134, 2007.

[8] L. Liao, D. J. Patterson, D. Fox, and H. Kautz. Learning and inferring transportation routines. *Artif. Intell.*, 171(5-6):311–331, 2007.

[9] N. Nguyen, H. Bui, S. Venkatesh, and G. West. Recognizing and monitoring high level behaviours in complex spatial environments. In *CVPR Conference Proceedings*. IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2003.

[10] N. Nguyen, D. Phung, S. Venkatesh, and H. Bui. Learning and detecting activities from movement trajectories using the hierarchical hidden markov model. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2005.

[11] Y. Teh, M. Jordan, M. Beal, and D. Blei. Hierarchical dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581, 2006.

[12] J. Yedida, W. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7):2282–2312, 2005.

8. APPENDIX

Unsupervised learning as maximal partitioning

For any assignment vector $\vec{x}$, there is a corresponding partitioning $\beta(\vec{x})$ which divides X into n unordered clusters, each corresponding to a hidden value. If we assume that factors permit exchangeability between state indexes (which is reasonable, as there is no training data to justify a bias), then the equivalence sets induced by β are also equivalent with respect to optimal parameters:

$$\beta(\vec{x}) = \beta(\vec{x'}) \Rightarrow \vec{w}^*(\vec{x}) = \vec{w}^*(\vec{x'}) \tag{15}$$

β-equivalence extends to whether assignment vectors satisfy (9) and the likelihood the vectors achieve under optimized parameters. Our search for $\vec{x}^*$ can therefore be reduced to a search for $\beta^* = \beta(\vec{x}^*) \in B$, where B is the set of all possible partitionings. Unfortunately, $|B|$ (given by $S(|X|, s)$ where S gives Stirling's number of the second kind) is still far too large to enumerate or sample meaningfully. However, a sensible search for assignment vectors should avoid evaluating multiple vectors in the same β-equivalence class.