ICAPS 2006

The English Lake District, Cumbria, UK

WS2

Workshop on POMDPs, Classification and Regression: Relationships and Joint Utilization

Xuejun Liao
Duke University, USA

Lawrence Carin
Duke University, USA














ICAPS 2006

The English Lake District, Cumbria, UK

WS2

Workshop on POMDPs, Classification and Regression: Relationships and Joint Utilization

Xuejun Liao
Duke University, USA

Lawrence Carin
Duke University, USA

ICAPS 2006
Workshop on POMDPs, Classification and Regression: Relationships and Joint Utilization

Table of contents

ICAPS 2006
Workshop on POMDPs, Classification and Regression: Relationships and Joint Utilization

Preface

The partially observable Markov decision process (POMDP) is a widely used model for planning under uncertainty. Classification and regression are standard statistical tools for reconstructing a source (or its attributes) from noise-corrupted data. Studies of POMDPs and classification/regression have been mostly pursued independently in the past. Recently, however, there has been reported research work showing the possibility of using classification/regression techniques to solve POMDPs or using a POMDP to build cost-sensitive classifiers. The work along the first line includes trajectory-based policy search, value approximation with regression models, while the work along the second line includes POMDP methods for cost-sensitive feature selection and sensor scheduling with the ultimate goal of classification. Despite of the researches reported to date, there remain many unknowns regarding how POMDP and classification/regression techniques can be applied to each other in a mutually beneficial way. The possibilities have not been explored to their full extent. This workshop aims to bring this research topic to the attention of more researchers and stimulate a broader range of contributions to both POMDP and classification/regression by looking at them from new and unified perspectives. The first paper included in this workshop note describes a new way of reducing the POMDP planning problem to a sequence of classification problems. The second paper represents a novel application of POMDPs in multi-modality autonomous sensing in a complicated environment. The third paper presents a POMDP formulation of the random disambiguation path (RDP) problem. While this workshop note does not intend to exhaust all state-of-the-art work on this topic, we hope it will stimulate more researches that could expand the topic in both the depth and the scope.

Xuejun Liao and Lawrence Carin

Organizers

- *Xuejun Liao, Duke University, USA*
- *Lawrence Carin, Duke University, USA*

Programme Committee

- *Alfred Hero , University of Michigan at Ann Arbor, USA*
- *Carey E. Priebe, Johns Hopkins University, USA*
- *Ronald Parr, Duke University, USA*
- *Carey Schwartz, DARPA/DSO, USA*
- *Douglas Cochran, Arizona State University , USA*
- *Vikram Krishnamurthy, University of British Columbia, Canada*
- *David Castanon, Boston University, USA*

Optimal Sensor Scheduling via Classification Reduction of Policy Search (CROPS)

Doron Blatt and Alfred O. Hero*
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, Michigan, USA

Abstract

The problem of sensor scheduling in multi-modal sensing systems is formulated as the sequential choice of experiments problem and solved via reinforcement learning methods. The sequential choice of experiments problem is a partially observed Markov decision problem (POMDP) in which the underlying state of nature is the system's state and the sensors' data are noisy state observations. The goal is to find a policy that sequentially determines the best sensor to deploy based on past data, which maximizes a given utility function while minimizing the deployment cost. Several examples are considered in which the exact model of the measurements given the state of nature is unknown but a generative model (a simulation or an experiment) is available. The problem is formulated as a reinforcement learning problem and solved via a reduction to a sequence of supervised classification subproblems. Finally, a simulation and an experiment with real data demonstrate the promise of our approach.

Introduction

The advent of agile sensing systems that collect data through a variety of sensing modalities has brought about new and exciting challenges to the field of signal processing. Agile, multi-modal, sensing (see e.g. (Krishnamurthy 2002) and (Kastella & Hero 2005)) exploit the capability of controlling the data collection process. Examples of agile sensing systems include a radar that can control its beam direction, a land mine detector that can deploy radar or seismic sensors, or a LANDSAT satellite that can control the frequency band of its radar. The key element that differentiates agile sensing systems from other data collection systems is a resource allocation constraint that precludes using all sensor modalities at all times. We formulate agile sensing as an optimization in which the system must automatically select the best sensing modality based on past observations to maximize a given objective function while minimizing the data collection cost.

When formulated as a sequential choice[1] of experiments problem (DeGroot 1970), the agile sensing problem consists of an episodic task that is divided into a sequence of decision epochs. Each episode begins as the first observation is collected. Then, at each subsequent decision epoch two decisions are made. The first one is to decide if the amount of information collected thus far is sufficient for making inference (detection or estimation) on the data with a desired accuracy or whether more observations are required. This first decision also determines the choices available at the second decision. If more observations are required, the next best sensor modality needs to be determined. If the information is deemed sufficient for inference, the final estimation or detection decision is made. Every sensor modality has an associated deployment cost and a decision rule must balance the expected information gain from a sensor deployment, which results in improved inference capabilities, with the deployment cost. The collection of decision rules, i.e., the sequence of mappings from past observations to the decision space, is called a policy and the goal is to find a policy that optimally trades-offs the overall average sensor deployment costs and the estimation or detection performance, e.g., mean squared estimation error or classification error rate.

The problem of finding optimal policies for sequential choice of experiments suffers from the curse-of-dimensionality (Bellman 1957) and scenarios in which a closed form solution for the optimal policy exists are rare. Past research has focused on the asymptotic regime in which one assumes a large number of data collection iterations (or sensor dwells) and low sensor deployment cost (see (Keener 2005) and references therein). Another focus has been on "experiment sufficiency" – when is one experiment (or sensor modality) always better than another experiment (see (Goel & Ginebra 2003) and references therein).

In this paper, we take a different approach. We assume that the underlying model is unknown and aim at finding *approximate* solutions to the optimal policy. In particular, in the absence of a model, optimal policies are approximated from data using a generative model, where data is generated by a simulator or collected in a field experiment. It is shown that this problem formulation falls into the class of reinforcement learning problems and the Classification Reduction of Policy Search (CROPS) methodology that has been recently

*This research was partially supported by DARPA-MURI grant ARO DAAD 19-02-1-0262.

[1]The key difference from the related sequential design of experiment problem is that instead of adapting a set of continuous experiment parameters, here we choose from a finite set of fixed experiments.

proposed by the authors (Blatt & Hero 2005) is applied. Two case studies are reported as well. The first is the problem of finding sensor scheduling policies for land-mine detection. For this problem a simulator is used to generate data which is then used for policy search. The second problem is perform optimal waveform selection for a multi-band radar on a land classification satellite. In this application competitive policies are found from experimental LANDSAT data.

Problem Formulation

Let $X_1 \in \mathcal{X}_1, X_2 \in \mathcal{X}_2, \ldots, X_K \in \mathcal{X}_K$ be K random variables that correspond to the outputs of K sensors or K sensor modalities. Note that each of these random variables lies, in general, in a different space. We append each random variables with its index so that a value of an observation also indicates which sensor was used to collect it. Let $Y \in \mathcal{Y}$ be a discrete random variable that represent the state of nature whose value we try to predict. The presented results can also be applied when Y is a continuous random variables, whose value we try to estimate, but we focus on the detection problem for concreteness.

A policy π specifies which sensor to deploy first, say sensor k. Then based of the value of X_k the policy determines if an accurate prediction of Y is possible, and if so, what is the best prediction, or, otherwise, which is the next best sensor to deploy to collect additional data. This process continues until either a prediction of Y is made or all available sensors are deployed. We assume that each sensor can be applied at most once and hence, the total number observations is bounded by K. Therefore, a policy π is sequence of $K+1$ decision rules $\pi = [\pi_1, \pi_2, \ldots, \pi_{K+1}]$. This assumption is valid when the randomness in the process, e.g. the observation noise, is governed by clutter that cannot be averaged out by repeated measurements, rather than by thermal noise. Note that π_1 simply indexes the first sensor to deploy (excluding the possibility of predicting Y without taking any observations), and hence, $\pi_1 \in \{1, 2, \ldots, K\}$. Also note that π_{K+1} is used only if at all the decision epochs the decision was to defer the prediction of Y and deploy another sensor. The decision rule π_{K+1} is a map from $\mathcal{X}_1 \times \mathcal{X}_2 \times \ldots \times \mathcal{X}_K$ to $\mathcal{Y}$. If the objective is to try to minimize the detection error, then it is well known that the optimal map is the Bayes classifier (Hastie, Tibshirani, & Friedman 2001)

$$\pi^*_{K+1}(x_1, x_2, \ldots, x_K) = \arg\max_{y \in \mathcal{Y}} \Pr\{Y = y | X_1 = x_1, X_2 = x_2, \ldots, X_K = x_K\}.$$

The domain and range of the decision rules for stages $2, \ldots, K$ depend on the sequence of sensors deployed up to the decision time. For example, if $\pi_1 = k$, then

$$\pi_2 : \mathcal{X}_k \rightarrow (\{1, 2, \ldots, K\} \setminus k) \bigcup \mathcal{Y}.$$

If $\pi_2(x_k) \in (\{1, 2, \ldots, K\} \setminus k)$ then the decision is to take another observation using sensor $\pi_2(x_k)$. Alternatively, if $\pi_2(x_k) \in \mathcal{Y}$, then the decision is that the amount of information is sufficient and $\pi_2(x_k)$ is the predictor of Y. Instead of explicitly defining the policy through a sequence of mappings whose domains and ranges depend of past decisions and observations, we let $Z = [X_1, \ldots, X_K]$ and define the policy π as a two-dimensional function of Z. Given, the value of Z, its first argument $[\pi(Z)]_1$ is the resulting sequence of sensors that were deployed prior to the final decision and its second argument $[\pi(Z)]_2$ is the prediction for Y. Note that in general, only a subset of the elements of Z are observable at the time the final decision is made.

Denote by $P_c(\pi) = \Pr\{[\pi(Z)]_2 = Y\}$ the probability of correctly predicting the value of Y based on the data collected according to the policy π, by $C([\pi(Z)]_1)$ the cost associated with the sequence of sensor deployments $[\pi(Z)]_1$, e.g., the number of sensor dwells, and by $\mathrm{E}\{C([\pi(Z)]_1)\}$ the expected cost. We assume that the cost of the deployment of a sequence of sensors is the sum of the costs of deploying each of the sensors, and hence, does not depend on the order of deployment. The optimal policy π^* is the policy that maximizes

$$P_c(\pi) - \lambda \mathrm{E}\{C([\pi(Z)]_1)\}, \quad (1)$$

where λ is a tuning parameter that trades off the cost of data collection and the cost of prediction error. Under certain regularity conditions, the optimal policy can be defined though backward induction (see e.g. (Puterman 1994)). However, when $\mathcal{X}_1, \ldots, \mathcal{X}_K$ are continuous or discrete and large, the solution becomes intractable. Furthermore, even when $\mathcal{X}_1, \ldots, \mathcal{X}_K$ are finite and relatively small, the backward induction iterations require computing expectations with respect to the joint distribution of Z and Y.

In this paper we allow $\mathcal{X}_1, \ldots, \mathcal{X}_K$ to be continuous or discrete and large, and consider the case in which the joint distribution of Z and Y is unknown. We assume that n realizations of (Z, Y) are available and the goal is find a policy that maximizes (1) based on this data set. Hence, this is a model free instance of the sequential choice of experiments problem as formulated in (DeGroot 1970), which, to the best of our knowledge, has not been considered previously in the literature.

Partially Observable Markov Decision Processes and Reinforcement Learning

The field of reinforcement learning is centered around the challenge of designing agents that learn to act in a stochastic environment by interacting with it (Sutton & Barto 1998). As the agent interacts with the environment it receives rewards, and the goal is to eventually learn through these rewards which actions maximize the future sum of rewards. A common mathematical model for reinforcement learning is the problem of finding the optimal policy for controlling a finite-horizon partially observable Markov decision process (POMDP) (Kearns, Mansour, & Ng 2000). The formulation of our sequential choice of experiments problem as finite-horizon POMDP consists of several elements:

- **The decision epochs** determine the times in which the agent is to take an action. In the discrete model adopted here, decision epochs occur at $t = 0, \ldots, \tau$. At every decision epoch either another observation is collected, or a final prediction of Y is made. In the later case the processes terminates. Therefore, τ is a random variable that depends on the deployed policy and Z.

- The system's state is the realization of Y which is fixed throughout the episode.
- The state at time zero is a random variable with distribution D over $\mathcal{Y}$.
- The state of the system cannot be directly observed but instead after every decision epoch $t = 0, \ldots, \tau$, in which the decision is to collect another observation, a noisy **observation** O_t of the systems' state is collected. The domain and distribution of the observation depends on the underlying systems' state Y and the deployed sensor. Denote by $\overline{O}_t = [O_0, O_1, \ldots, O_t]$ the observations up to and including time $t < \tau$, and note that $\overline{O}_t$ is a subset of Z.
- At every decision epoch $0 \leq t \leq \tau$ the agent chooses an **action** a_t, based on the past observations, from a set of possible actions called the **action space** $\mathcal{A}_t$. Though not explicitly appearing in the notation, the set of available actions $\mathcal{A}_t$ may depend on the past actions. In our application, only actions that correspond to sensors that have not be previously deployed can be taken.
- There exists a termination action which ends the process, such as the action of making the prediction of Y.
- We note that even though in our formulation the state of the system is fixed throughout the episode, the results can be generalized to the case in which upon taking action a at state y, the system makes a transition to state y' according to a **transition probability** $P_{y,a}$. In other wards, it is possible to generalize to the case in which the system's states evolve as a Markov process. This generalization is important for cases in which sensor deployment may be sensed by the target and lead to changes in the target's state as in (Kastella & Hero 2005).
- A **reward** $r(Y, a)$ is received after each time an action is taken. When a sensor is deployed to collect another observation, $r(Y, a)$ is minus the cost of deploying sensor a regardless of the state of the system. When the final prediction is made a reward of one unit is received only if the prediction $a = \widehat{Y}(\overline{O}_{\tau-1})$ equals Y, i.e., $r(Y, a) = I(a = Y)$, where I is the indicator function that equals one when its argument is true and zero otherwise.
- A **policy** π is a sequence of decision rules, or mappings from past observations to the action spaces, which specifies the action to take at each decision epoch. The policy is composed of $K + 1$ decision rules $(\pi_0, \pi_1, \ldots, \pi_K)$, however, if the termination action is taken prior to decision epoch K then not all decision rules are executed.

A typical episode is a sequence

$$a_0 \rightarrow O_0 \rightarrow a_1(O_0) \rightarrow O_1 \rightarrow a_2(\overline{O}_1) \ldots$$
$$O_{\tau-1} \rightarrow a_\tau(\overline{O}_{\tau-1}) = \widehat{Y}(\overline{O}_{\tau-1}),$$

where a_0 is the first decision to deploy a sensor before any observations were collected, $O_0, O_1, \ldots, O_{\tau-1}$ are the observations whose domains and distributions depend on Y and the decisions $a_0, a_1, \ldots, a_{\tau-1}$, respectively, and $a_\tau(\overline{O}_{\tau-1})$ is a decision that the past observations are sufficient for making a prediction on Y, and it specifies the predictor $\widehat{Y}(\overline{O}_{\tau-1})$. The objective is to find a policy π that maximizes the expected sum of rewards:

$$V(\pi) = \mathrm{E}_\pi \left\{ \sum_{t=0}^{\tau} r(Y, \pi_t(\overline{O}_{t-1})) \right\}, \tag{2}$$

where the expectation is taken with respect to the joint distribution of Z and Y, which, through π, induce a distribution on the observations $O_0, O_1, \ldots, O_{\tau-1}$. The expected sum of rewards $V(\pi)$ is called the value of the policy π.

It is well known that when the underlying joint distribution of the system state and the observations is known and the observations can take a finite number of possible values, it is possible to formulate the problems in terms of the information state and solve for the optimal policy (Kaelbling, Littman, & Cassandra 1998). In our setting, however, the joint distribution is unknown and the observations are, in general, continuous random variables. Approximating the optimal policy in this case is a classic problem in reinforcement learning. Here, we adopt the generative model assumption of (Kearns, Mansour, & Ng 2000). Under this assumption, the initial distribution D and the distribution of the observations conditioned on the system state and the deployed sensor are unknown but it is possible to generate realizations of the system state Y according to D and observations conditioned on arbitrary state Y and deployed sensor. In particular, we assume that we have n realizations of the pair (Z, Y) denoted by $\{(Z_1, Y_1), (Z_2, Y_2), \ldots, (Z_n, Y_n)\}$. Note that given a realization (Z_1, Y_1) it is possible to generate the entire decision tree associated with the sequential choice of experiment problem. An example of the decision tree in a problem in which there are two sensors $K = 2$ and $\mathcal{Y} = \{0, 1\}$ is given in Figure 1. Given a realization (Z_1, Y_1) and a policy π, it is possible to follow the path that a system that uses π will follow and compute the sum of rewards for this realization. Prior to the prediction of Y, the rewards are minus the sensor deployment costs, and, at the prediction epoch, a unit reward is received only if $\widehat{Y}(\overline{O}_{\tau-1}) = Y_1$, where $\widehat{Y}(\overline{O}_{\tau-1})$ is chosen by following the path induced by π.

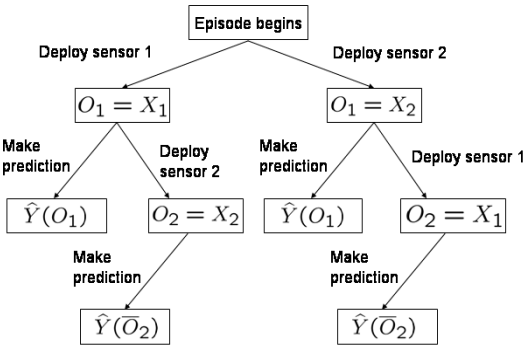


Figure 1: A decision tree for a sequential choice of experiment problem with $K = 2$ and $\mathcal{Y} = \{0, 1\}$.

Now, consider a class of policies Π, i.e., each element $\pi \in \Pi$ is a sequence of decision rules $\pi = (\pi_0, \pi_1, \ldots, \pi_K)$.

It is possible to estimate the value $V(\pi)$ (2) of any policy in the class from the set of trajectory trees by simply averaging the sum of rewards on each tree along the path that agrees with the policy (Kearns, Mansour, & Ng 2000). A policy specifies the action to take at each decision epoch and so there is exactly one path in every tree that agrees with a given policy. Denote by $\widehat{V}^i(\pi)$ the observed sum of rewards on the i'th tree along the path that corresponds to the policy π. Then the value of the policy π is estimated by

$$\widehat{V}_n(\pi) = n^{-1} \sum_{i=1}^{n} \widehat{V}^i(\pi). \quad (3)$$

In (Kearns, Mansour, & Ng 2000), the authors show that with high probability (over the data set) $\widehat{V}_n(\pi)$ converges uniformly (over Π) to $V(\pi)$ with rates that depend on the VC-dimension of the policy class. This result motivates the use of policies π with high $\widehat{V}_n(\pi)$, since with high probability these policies have high values of $V(\pi)$.

In (Blatt & Hero 2005) it is shown that while the task of finding the global optimum within a class of non-stationary policies may be overwhelming, the componentwise search, i.e., optimizing a single decision rule at a time, leads to single step reinforcement learning problems which can be reduced to a sequence of multi-class weighted classification problems. Multi-class weighted classification problems can be solved using re-sampling methods or heuristic extensions of methods for binary weighted classification (see (Abe, Zadrozny, & Langford 2004) for both approaches). Below, it is shown how to convert a multi-action RL problem into a binary RL problem by introducing dummy decision epochs. Then, applying the method in (Blatt & Hero 2005) leads to a sequence of binary weighted classification problems that can be directly solved using off-the-self classification methods.

A Nonlinear Gauss Seidel Approach

Suppose an initial policy is given and one wishes to improve upon it by optimizing one of the decision rules at a time while holding the rest fixed. In (Blatt & Hero 2005) it is shown that this component-wise search is equivalent to simple tree pruning operations. In particular suppose π_k is updated while the decision rules $(\pi_0, \ldots, \pi_{k-1})$ and $(\pi_{k+1}, \ldots, \pi_K)$ are held fixed. Since $(\pi_0, \ldots, \pi_{k-1})$ are held fixed, the path taken by the policy up to and including epoch $k-1$ will not change when we update π_k. Hence, it is possible to prune the tree from the top down to epoch k by removing the branches that do not agree with the actions taken according to $(\pi_0, \ldots, \pi_{k-1})$. Since $(\pi_{k+1}, \ldots, \pi_K)$ are held fixed, the path taken by the policy after taking each of the possible actions at epoch k are known and will not change when we update π_k. Hence, it is possible to prune the tree from $k+1$ to the leaves by removing the branches that do not agrees with the actions taken according to $(\pi_{k+1}, \ldots, \pi_K)$. Furthermore, since by the second pruning the path that will be followed after taking each of the actions at decision epoch k is known, it is possible to obtain realizations of the sum of future rewards that results in taking each of the actions at decision epoch k. In mathematical programming, a component-wise optimization of a nonlinear function is often referred to as nonlinear Gauss-Seidel algorithm (Bertsekas 1999). It is in this sense that we call the above component-wise policy search a nonlinear Gauss-Seidel approach.

This procedure is illustrated for the simple decision tree of Figure 1 in Figure 2. Note that since after taking an action at decision epoch 1 the path of the tree is fixed regardless of the policy (see Figure 1), there is no tree pruning, only reward propagation according to the value of $\widehat{Y}(\overline{O}_2)$. Specifically, after the reward propagation, we can observe that taking action 'make prediction' results in an immediate and final reward $I(\widehat{Y}(\overline{O}_1) = Y)$ and taking action 'deploy sensor 2' lead to an immediate reward of minus the deployment cost associated with sensor 2 plus the subsequent reward $I(\widehat{Y}(\overline{O}_2) = Y)$. Since at epoch 1 the future sum of rewards is determined for every action, the task of updating policy π_1 is a single step RL problem. Below it is shown that this problem is equivalent to a certain supervised learning problem. Before the conversion to supervised learning, we convert the RL problem into a binary RL problem.

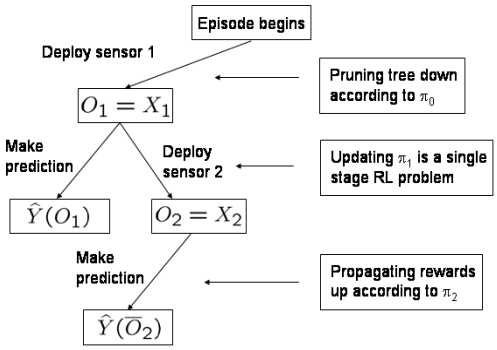


Figure 2: Updating π_1 while holding π_0 and π_2 fixed.

From Multiple-Action Reinforcement Learning to Binary Reinforcement Learning

The nonlinear Gauss-Seidel approach of the previous section breaks the multi-stage search associated with the trajectory tree method into a sequence of single-stage RL subproblems. In (Blatt & Hero 2005) these single-stage RL subproblems were converted to multi-class weighted classification problems, which can then be solved using, e.g., re-sampling methods (Abe, Zadrozny, & Langford 2004). In this section it is shown that it is possible to convert a single-stage RL problem into multi-stage binary RL problem, apply the nonlinear Gauss-Seidel approach, and arrive at a sequence of binary single-stage RL subproblems.

Consider a single-stage RL problem with K possible actions. It is possible to describe any action as the answer to at most $\lceil \log_2(K) \rceil$ 'yes or no' questions, where $\lceil x \rceil$ is the smaller integer larger than or equal to x. Then, the single-stage RL problem is described by the decision tree associated with these binary decision epochs. Once an intermediate decision is made, it corresponds to a transition

to the same state, i.e., the state does not evolve, but with a reduced (halved) action space. Only when the decision is between two actions, does the chosen action is executed and a state transition occurs. Figure 3 demonstrate converting a 4-action single-stage RL problem into a two-stage binary RL problem.

Finally, reapplying the nonlinear Gauss-Seidel algorithm of the previous section leads to a sequence of single-stage binary RL subproblems.

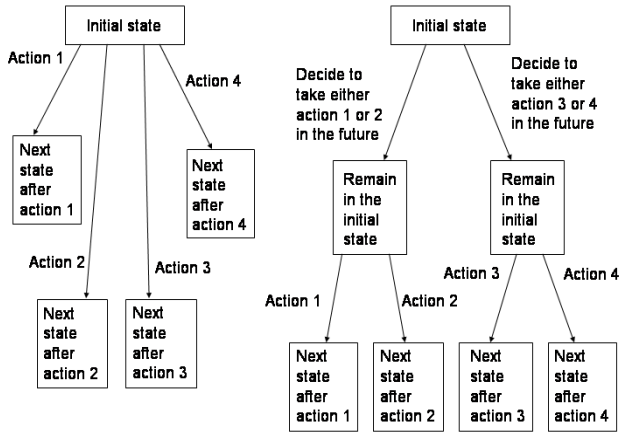


Figure 3: Converting a 4 action single-stage RL problem into a two-stage binary RL problem.

A Reduction From a Single Step Reinforcement Learning Problem to Weighted Classification

In this section we present the conversion of a single-step binary RL problem into a supervised learning problem, which is a special case of the classification reduction in (Blatt & Hero 2005). The goal is to leverage techniques and theoretical results from supervised learning for solving the more complex problem of reinforcement learning (Barto & Dietterich 2004). To simplify to presentation we do not carry the heavy notation of the previous section but rather introduce a simple generic notation to explain the conversion. Consider a single-step binary RL problem. An initial state $S_0 \in \mathcal{S}$ generated according to the distribution D is followed by one of 2 possible actions $A \in \{0, 1\}$, which leads to a transition to state S_1 whose conditional distribution given that the initial state is s and the action is a is given by $P_{s,a}$. Given a class of policies Π, where a policy in Π is a map from $\mathcal{S}$ to $\mathcal{A}$, the goal is to find

$$\widehat{\pi} \in \arg\max_{\pi \in \Pi} \widehat{V}_n(\pi). \tag{4}$$

In this single stage problem the data are n realizations of the random element $\{S_0, S_1|0, S_1|1\}$, where $S_1|0$ (respectively $S_1|0$) is a realization of S_1 after taking action 0 (respectively 1) at state S_0. Denote the i'th realization by $\{s_0^i, s_1^i|0, s_1^i|1\}$. In this case, $\widehat{V}_n(\pi)$ can be written explicitly by

$$\widehat{V}_n(\pi) = \mathrm{E}_n \left\{ \sum_{l=0}^{1} r(S_0, l, S_1|l) I(\pi(S_0) = l) \right\}, \tag{5}$$

where $r(S_0, l, S_1|l)$ is the reward gained when taking action l at state S_0 and making a transition to state $S_1|l$, for a function f, $\mathrm{E}_n \{f(S_0, S_1|0, S_1|1)\}$ is its empirical expectation $n^{-1} \sum_{i=1}^{n} f(s_0^i, s_1^i|0, s_1^i|1)$, and $I(\cdot)$ is the indicator function taking a value of one when its argument is true and zero otherwise.

The following theorem shows that the problem of maximizing the empirical reward (5) is equivalent to a binary weighted classification problem.

Proposition 1 *Given a class of policies* Π *and a set of* n *trajectory trees,*

$$\arg\max_{\pi \in \Pi} \mathrm{E}_n \left\{ \sum_{l=0}^{1} r(S_1|l) I(\pi(S_0) = l) \right\} = \arg\min_{\pi \in \Pi}$$
$$E_n \left\{ |r(S_1|0) - r(S_1|1)| I(\pi(S_0) \neq \arg\max_k r(S_1|k)) \right\}$$

Proof 1 *Take* $L = 2$ *in Proposition* 1 *in* (Blatt & Hero 2005).

The theorem implies that the maximizer of the empirical reward over a class of policies is the output of an optimal weights-dependent classifier for the data set:

$$\left\{ \left(s_0^i, \arg\max_{k \in \{0,1\}} r(s_1^i|k), |r(s_1^i|0) - r(s_1^i|1)| \right) \right\}_{i=1}^{n},$$

where for each sample, the first argument is the example, the second is the label, and the third is a realization of the cost incurred when misclassifying the example. The implication is that a variety of supervised learning methods, such as k-nearest neighbors (Devroye, Györfi, & Lugosi 1996), neural networks (Bishop 1995), Boosting (Freund & Schapire 1997), and support vector machines (Schölkopf & Smola 2002), can be applied to solve the single-stage binary RL problem.

Sensor Scheduling for Land-Mine Detection

This section reviews a sequential choice of experiment problem that arises in the design of unmanned land-mine detection vehicle. The vehicle carries three sensors for performing the detection: an EMI sensor, a ground penetrating radar (GPR), and an acoustic sensor. As can be seen in Figure 4, the sensors have different responses under different types of land-mines and clutter. In addition, deploying a sensor takes time and energy and hence not all sensors are deployed at every potential land-mine location. Upon reaching a new location, in which a land-mine is potentially present, a policy that trades of the cost of a sensor deployment and detection probability determines the first sensor to deploy. Based on the collected measurement, either a prediction regarding the presence of the land-mine is made or a second sensor is

deployed. Finally, based on the output of the first two deployed sensors, either a prediction regarding the presence of the land-mine is made or a third sensor is deployed followed by the final prediction based on all three measurements. The goal is to maximize the probability of correct detection minus a constant $c > 0$ (1) times the number of sensor dwells.

Since there are a total of three sensors $Z = [X_1, X_2, X_3]$. The state space is binary $\mathcal{Y} = \{0, 1\}$, where $Y = 0$ means no land-mine is present and $Y = 1$ indicates the presence of a land-mine. The decision tree associated with this problem is presented in Figure 5.

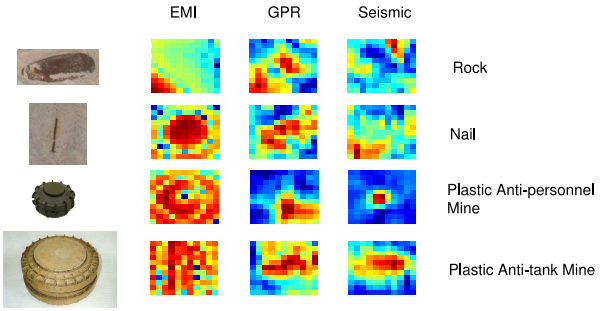


Figure 4: Sensors signatures for several land-mine and clutter types.

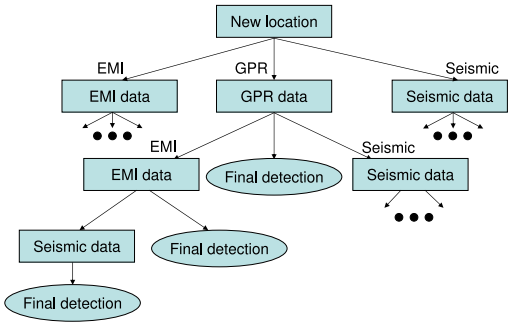


Figure 5: The decision tree associated with the land-mine detection problem.

Figure 7 summarizes the features extracted from each sensor and their expected signatures under different scenarios. In the simulation, one of the possible eight scenarios was first chosen randomly. Then, a realization of each of the features, which together compose Z, is generated as a Gaussian random variable with means 0, 0.5, or 1, corresponding to low, medium, or high, respectively. The covariances of sensors 1, 2, and 3, were $0.5I$, $0.45I$ and 0.1, respectively, where I is the 2-dimensional identity matrix. These values of means and covariances were chosen in correspondence with experiments that were conducted in a sand box (Marble, Blatt, & Hero 2006). Hence the marginal distribution of the vector of sensor outputs is a five-dimensional eight-component Gaussian mixture.

Before searching for the optimal sensor scheduling policy, the classifiers $\widehat{Y}(O_1), \widehat{Y}(\overline{O}_2), \widehat{Y}(\overline{O}_3)$ for all possible combinations of sensor selections

$$X_1, X_2, X_3,$$
$$(X_1, X_2), (X_1, X_3), (X_2, X_3),$$
$$(X_1, X_2, X_3)$$

were found by training two-layer feed-forward neural networks, each with ten input and two output nodes, on 1000 samples of (Z, Y). By testing the performance of these classifiers on a separate test set of 1000 samples, we found that the best single sensor to use for detecting a land-mine is the EMI sensor, that the two best fixed sensors are GPR plus the Seismic, and that in this scenario the classifier which is based on the output of all three sensors has a probability of correct detection of 0.887. The search for the optimal sensor scheduling policy was conducted while these classifiers remained fixed. In other words, only decisions regarding whether or not to deploy a sensor, and which sensor to deploy next were considered. Since the classifiers remained fixed during the policy search, once a decision to make prediction is made, the reward is gained according to the classifier output, without trying to further optimize its performance.

As explained above, the optimal policy was approximated by introducing dummy decision epochs, so that all the decisions are binary. We then performed the nonlinear Gauss-Seidel decomposition into a sequence of single-stage binary reinforcement learning problems. Each subproblem was then converted to a weighted classification problem that was solved by a weights-sensitive two-layer feed-forward neural network with seven input and two output nodes.

Figure 6 summarizes the results. The horizonal axis is the average number of sensor dwells and the vertical is the probability of correct detection. The three solid circles correspond to the performance of the best single sensor, best two sensors, and the performance when all three sensors are deployed, respectively. These points are connected by a solid line that corresponds to performance that can be achieved by randomly selecting one of these fixed sensor configurations. The crosses corresponds to the performance (estimated from a 1000 trail test set) obtained by the approximated optimal sensor scheduling policies. Each cross correspond to a different choice of c (1), ranging from $c = 0.2$ at the left lower corner and $c = 0$ at the outmost upper right cross. When $c = 0.2$ the price of taking more than a single measurement is too dear compared to the improvement in the probability of correct detection and the policy dictates making decision using only a single sensor. As c decreases, more and more observations are allowed. It is interesting to see that when c is zero, i.e, the sensor deployment cost is zero, the algorithm does not always deploy all three sensors, but achieves better performance than when all three sensors are always deployed. This happens since the classifiers used at the prediction stages are not the Bayes classifiers (in which more information can never worsen performance) but rather suboptimal classifiers that were found by training neural networks.

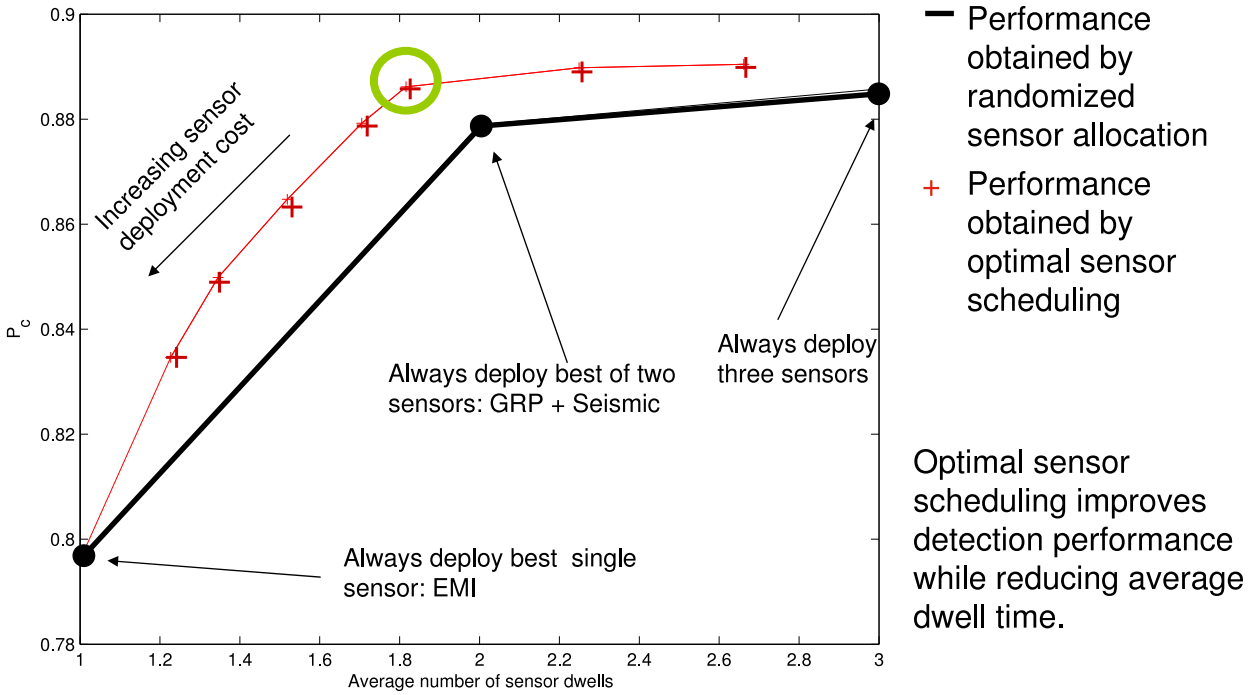


Figure 6: Performance of sensor-scheduling-based detection compared to detection under optimal fixed sensor allocations.

		Object Type								
		1	2	3	4	5	6	7	8	Feature
		M-AT	M-AP	P-AT	P-AP	Cltr-1	Cltr-2	Cltr-3	Bkg	Description
Sensor	EMI (1)	High	High	Medium	High	High	Low	Low	Low	Conductivity
		High	High	High	Medium	Medium	Low	Low	Low	Size
	GPR (2)	High	Medium	High	Medium	Low	Low	Low	Low	Depth
		High	Medium	High	Medum	High	High	High	Low	RCS
	Seismic (3)	High	Medium	High	Medium	Medium	Medium	Low	Low	Resonance
Optimal sequence for mean state		2 3 D	2 1 D	2 3 D	2 1 3 D	2 3 D	2 3 D	2 3 D	2 3 D	

Figure 7: Sensor mean responses under various scenarios. M-Metal, P-Plastic, AP-Anti personal, AT-Anti tank, Cltr-1-Hallow metal clutter, Cltr-2-Hallow non-metal clutter, Cltr-3-Non-metal non-hallow clutter, Bkg-Background.

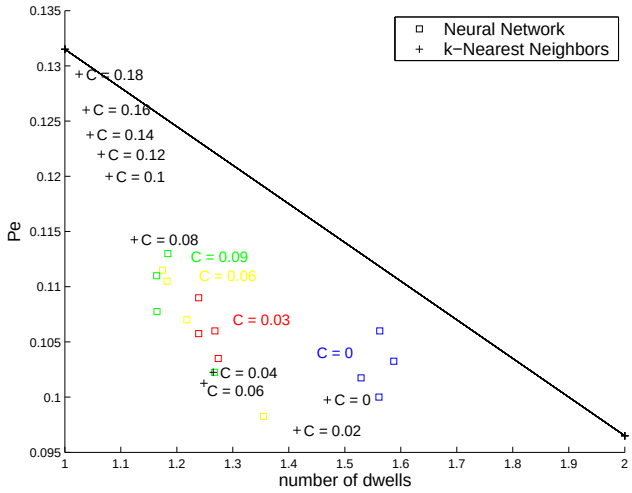


Figure 8: Performance of sensor scheduling algorithm for the land monitoring satellite problem.

It is encouraging that by training the neural networks we found a policy that accounts for generalization errors at the predictor level and do not collect the third observation when that observation might lead to a worse prediction. In summary, it can be seen that through sensor scheduling it is possible to achieve better classification performance with fewer average number of sensor dwells. The actual sensor sequences taken under the possible eight scenarios when the policy whose performance cross is circled is presented in Figure 7. It is seen that the optimal policy dictates that the first deployed sensor is the GPR sensor even though the optimal single sensor is the EMI sensor. This is not surprising since an optimal sensor scheduling optimizes the future sum of rewards rather than choosing the sensor whose stand alone performance are the best. Furthermore, only when the underlying system state is a plastic anti-personal land-mine, which has the weakest signature, does the policy dictate using all three sensors. In other cases, two sensors are sufficient for the land-mine detection.

Waveform Selection for Land Monitoring Satellite

In this section, the optimal sensor scheduling algorithm is applied to real data for the problem of waveform selection for a LANDSAT land monitoring satellite. The satellite collects a radar backscatter on a patch of land and the goal is to classify the land type based on the returned signal. Given a new probing location, the satellite can transmit one of four possible waveforms. The different waveforms correspond to different frequency bands. Therefore, $Z = [X_1, X_2, X_3, X_4]$. Each of the observations $X_1, \ldots, X_4$ is a 9-dimensional vector taking values in $[0, 255]^9$, and hence, Z is a 36-dimensional vector. There are six land types, and hence $\mathcal{Y} = \{1, 2, \ldots, 6\}$. In the public data set (Srinivasan 1994), there are 4435 points in the training set and 2000 in the test set. For a more detailed explanation of the problem see (Hastie, Tibshirani, & Friedman 2001) chapter 13. In this section we explore using sensor scheduling for reducing the number of waveform (frequency band) transmissions. In particular, we find policies that select the first best two frequency bands and based on the outcome determine if the remaining frequency bands are required, or whether the first two bands provide sufficient information for classifying the land type. Hence, at the first decision epoch there are six possible actions leading to siz possible measured pairs of frequency bands:

$$\{[X_1, X_2], [X_1, X_3], [X_1, X_4], ...$$
$$[X_2, X_3], [X_2, X_4], [X_3, X_4]\}.$$

The land type classifiers are the k-nearest neighbors algorithm with k set to 5, as recommended in (Hastie, Tibshirani, & Friedman 2001) for the non-sequential problem. Two classifiers for the policy search were considered. The first is a $[7, 5, 2]$ feed-forward weights-sensitive neural network. The second is a weights-sensitive k-nearest neighbor, where $k = 30$. The performance are summarized in Figure 8. The crosses correspond to the performance of policies that were found by weights-sensitive k-nearest neighbor classifiers as c ranges from 0 to 0.18. The squares correspond to the performance of policies that were found by weights-sensitive $[7, 5, 2]$ feed-forward neural networks for four values of c. To study the effect of the initial network weights distribution, for each value of c, the neural networks training was initiated at four random weights selections, leading to four resulting policies. As can be seen, under both learning configurations it is possible to obtain a range of tradeoffs between sensor deployment cost and classification performance. Particularly, the policy learned by the k-nearest neighbor classifier with $c = 0.02$ almost achieves the same performance as when all sensor modalities are used, but with a significant reduction in deployment cost. From comparing the performance of the k-nearest neighbor classifier based policy with the one based on the neural networks it is seen that the performance achieved by the two architectures are comparable.

Conclusions

Sensor scheduling for controlling agile sensing systems was formulated as a sequential choice of experiments problem and solved via a reduction of the associated RL problem to a sequence of supervised learning problems. The method was applied to both real and synthetic data – land mine detection and LANDSAT terrain classification. Finally, the authors would like to thank Jay Marble and Raviv Raich for helpful discussions.

References

Abe, N.; Zadrozny, B.; and Langford, J. 2004. An iterative method for multi-class cost-sensitive learning. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 3–11.

Barto, A. G., and Dietterich, T. G. 2004. Reinforcement learning and its relationship to supervised learning. In Si, J.; Barto, A.; Powell, W.; and Wunsch, D., eds., *Handbook*

of learning and approximate dynamic programming. John Wiley and Sons, Inc.

Bellman, R. 1957. *Dynamic Programming*. Princeton, NJ: Princeton University Press.

Bertsekas, D. P. 1999. *Nonlinear programming: second edition*. Belmont, MA: Athena Scientific.

Bishop, C. M. 1995. *Neural Networks for Pattern Recognition*. Oxford, Great Britain: Oxford University Press.

Blatt, D., and Hero, A. O. 2005. From weighted classification to policy search. In *18th Annual Conference on Neural Information Processing Systems (NIPS)*.

DeGroot, M. H. 1970. *Optimal Statistical Decisions*. New York: McGraw-Hill.

Devroye, L.; Györfi, L.; and Lugosi, G. 1996. *A Probabilistic Theory of Pattern Recognition*. Springer.

Freund, Y., and Schapire, R. E. 1997. A decision-theoretic generalization of on-line learning and an application to Boosting. *Journal of Computer and System Sciences* 55(1):119–139.

Goel, P. K., and Ginebra, J. 2003. When is one experiment always better than another. *The statistician* 52(4):515–537.

Hastie, T.; Tibshirani, R.; and Friedman, J. 2001. *The elements of statistical learning*. New York: Springer-Verlag.

Kaelbling, L. P.; Littman, M.; and Cassandra, A. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101.

Kastella, C. K. K., and Hero, A. 2005. Sensor management using an active sensing approach. *Signal Processing* 85(3):607–624.

Kearns, M.; Mansour, Y.; and Ng, A. 2000. Approximate planning in large POMDPs via reusable trajectories. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Keener, R. W. 2005. Local information and the design of sequential hypothesis tests. *Journal of Statistical Planning and Inference* 130(1-2):111–125.

Krishnamurthy, V. 2002. Algorithms for optimal scheduling and management of hidden markov model sensors. *IEEE Trans. Signal Process.* 50(6):1382–1397.

Marble, J.; Blatt, D.; and Hero, A. 2006. Confirmation sensor scheduling using a reinforcement learning approach. In *SPIE Defense and Security Symposium*. to appear.

Puterman, M. L. 1994. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, Inc.

Schölkopf, B., and Smola, A. J. 2002. *Learning with Kernels*. MIT, Press.

Srinivasan, A. 1994. The landsat data set. `http://www.niaad.liacc.up.pt/old/ statlog/ datasets/ satimage/ satimage.doc.html`.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning*. MIT Press.

Application of Partially Observable Markov Decision Processes to Robot Navigation in a Minefield

Lihan He, Shihao Ji, and Lawrence Carin
Department of Electrical and Computer Engineering
Duke University
Durham, NC 27708-0291, USA
{lihan,shji,lcarin}@ee.duke.edu

Abstract

We consider the problem of a robotic sensing system navigating in a minefield, with the goal of detecting potential mines at low false alarm rates. Two types of sensors are used, namely, electromagnetic induction (EMI) and ground-penetrating radar (GPR). A partially observable Markov decision process (POMDP) is used as the decision framework for the minefield problem. The POMDP model is trained with physics-based features of various mines and clutters of interest. The training data are assumed sufficient to produce a reasonably good model. We give a detailed description of the POMDP formulation for the minefield problem and provide example results based on measured EMI and GPR data.

Introduction

In many sensing problems, a robotic platform is preferred to a humanly-operated platform, an important example being that of ground-based sensing of landmines (MacDonald 2003). The robotic platform navigates in a minefield in an autonomous fashion, with optimal decisions dynamically made for its position, orientation, and the deployment of multiple sensors. The decision optimization is based on minimizing two fundamental types of costs in landmine detection: the detection cost and the sensing cost.

The landmines and mine-like clutter vary considerably in their contents (metal, plastic, etc) and size (small, large, etc), therefore it is vital to build a unified model to represent the mines and clutter so as to make the decision making possible. There are several typical sensors used in landmine detection, including ground-penetrating radar (GPR) and electromagnetic induction (EMI) sensor, which we consider in the present paper.

The minefield problem may be cast in the form of an adaptive sensor-management problem (Kastella 1997; Abdel-Samad & Tewfik 1999) (here with two sensors, the GPR and EMI sensors), though the problem is complicated significantly by the variety of the landmine and clutter signatures. We here consider a partially observable Markov decision process (POMDP) formalism (Kaelbling, Littman, & Cassandra 1998). In the POMDP formulation the environment under test is assumed to reside within a particular state S_E, and this state is not observable directly; the state of the environment, defined by the presence/absence of a mine in the region being sensed, is unchanged by the sensing itself. The state S_E is partially observable, in the form of the measured sensor data. The agent has particular actions at its disposal, including "moving to a new location", "deploying one of the sensors", "declaring the presence or absence of landmines'. Each of these actions has an expected immediate cost, as well as an impact on the long-term sensing cost. The POMDP constitutes a framework that balances the (discounted) infinite-horizon performance of this multi-sensor problem, i.e., it accounts for the immediate expected cost, as well as discounted future costs, over an infinite horizon (Kaelbling, Littman, & Cassandra 1998).

The POMDP is employed to constitute a sensing policy, defining the optimal next action to take based upon the agent's current belief about the environment under test (Kaelbling, Littman, & Cassandra 1998). The belief is defined in terms of a belief state, a probability mass function (pmf) of the environmental states S_E, conditional on all previous actions and observations (Kaelbling, Littman, & Cassandra 1998). To compute the belief state one requires an underlying model of the environment under test (Kaelbling, Littman, & Cassandra 1998), characterized by a statistical representation of observations given a sequence of controlling actions. We assume that we have access to a sufficient ensemble of measured data collected by the GPR and EMI sensors of the mines and mine-like clutter, so that we can design the POMDP model and find the corresponding optimal policy. The target states S_T of the POMDP are defined by sensor positions relative to the target, and the sequence of target states visited is modeled as a Markov process, conditioned on the sensor-platform motion; since the target position is unknown (hidden), the state is partially observable. In this setting we must distinguish the overarching state of the environment under test S_E, which is to be inferred by the POMDP policy (via the belief state), *vis-a-vis* the states of the underlying target model S_T, which are visited when performing the adaptive sensing. Given a set of GPR and EMI data, measured at a sequence of spatial positions relative to the target, we must now develop the POMDP model.

In this paper we develop a POMDP formulation based on the assumption that *a priori* and adequate training data are available for model development. We here employ measured

GPR and EMI data, for real mines and realistic clutter. The measured data considered in this study are available upon request, and therefore it is hoped that it will evolve to a standard data set researchers may use to test different adaptive sensor-management algorithms.

Partially Observable Markov Decision Processes

A POMDP model is represented by a six-element tuple $\langle S, A, T, \Omega, O, R\rangle$, where S is a finite set of discrete states, A is a finite set of discrete actions, and Ω is a finite set of discrete observations. The state-transition probability

$$T(s, a, s') = \Pr(S_{t+1} = s' | S_t = s, A_t = a) \quad (1)$$

describes the probability of transitioning from state s to state s' when taking action a. The observation function

$$O(a, s', o) = \Pr(O_{t+1} = o | A_t = a, S_{t+1} = s') \quad (2)$$

describes the probability of sensing observation o after taking action a and transiting to state s'. Finally, the reward function $R(s, a)$ represents the immediate expected reward the agent receives by taking action a in state s.

Since the state is not observed directly, a belief state b is introduced. The belief state is a probability distribution over all states, representing the agent's probability of being in each of the states based on past actions and observations, assuming access to the correct underlying model. The belief state is updated by Bayes rule after each action and observation, based on the previous belief state:

$$b_t(s') = \frac{1}{c} O(a, s', o) \sum_{s \in S} T(s, a, s') b_{t-1}(s) \quad (3)$$

with the normalizing constant

$$c = \sum_{s' \in S} O(a, s', o) \sum_{s \in S} T(s, a, s') b_{t-1}(s) = \Pr(o | a, b) \quad (4)$$

A POMDP policy is a mapping from belief states to actions, telling the agent which action to take based on the current belief state. The goal of the POMDP is to find an optimal policy by maximizing the expected discounted reward

$$V = E[\sum_{t=0}^{k-1} \gamma^t R(s_t, a_t)] \quad (5)$$

which is accrued over a horizon of length k. The discount factor $\gamma \in (0, 1]$ describes the degree to which future rewards are discounted relative to immediate rewards. If k is finite the optimal action depends on the distance from the horizon, and therefore the policy is termed non-stationary. However, often an appropriate k is not known, so we may consider an infinite-horizon policy, i.e., k goes to infinity, for which we require $\gamma < 1$. An infinite horizon also implies a stationary policy, independent of the agent's temporal position.

When in belief state b, the maximum expected reward k steps from the horizon $V(k)$ is

$$V^{(k)}(b)$$

$$= \max_{a \in A} \left[\sum_s R(s, a) b(s) + \gamma \sum_o p(o | a, b) V^{(k-1)}(b_a^o) \right] \quad (6)$$

where b_a^o the belief state after the agent takes action a and observes o, as updated in (5). The $V^{(k)}(b)$ represents the maximum expected discounted reward the agent will receive if it is in belief state b and takes actions according to the optimal policy for future steps. In this paper policy design is performed using the PBVI algorithm, with details provided in (Pineau, Gordon, & Thrun 2003).

The POMDP Model for Landmine Detection

We consider a minefield as an area of land where mines of several known types and other mine-like objects (clutter) are buried underground. The positions of the mines and clutter are unknown. The task is to detect the mines at a low false alarm rate, with an economic use of sensors. This is a highly dangerous task and therefore a robot platform is designed to perform it. Below we specify the POMDP model for this problem.

Feature extraction

The EMI measurement in any position is the complex response of the magnetic field as a function of frequency. A typical EMI response when the sensor is above a metal mine is shown in Figure 1. The magnetic field induced by a target is represented by the formula (Gadar, Mystkowski, & Zhao 2001)

$$H(\omega) \propto a + \frac{b_1 \omega}{\omega - j\omega_1} + \frac{b_2 \omega}{\omega - j\omega_2} \quad (7)$$

where a, b_1, b_2 are related to the magnetic dipole moments of the target, and ω_1 and ω_2 represent the associated EMI resonant frequencies. Features can be extracted from an EMI observation by fitting the measured data to the model in (7), assuming additive noise n in the observation, i.e.,$Y(\omega) = H(\omega) + n$. The nonlinear fitting parameters $\{a, b_1, b_2, \omega_1, \omega_2\}$ are our EMI features.

The GPR observation for a given position is recorded as the radar signature as a function of time. The time dimension is associated with the depth of the soil: the signals reflected from deeper positions have larger time delays. Figure 2(a) shows a typical GPR observation when the sensor is above a plastic mine, and Figure 2(b) is a 2-dimensional scan of the landmine signature. Features extracted from a GPR observation include the raw moments (corresponding to energy features) and central moments (corresponding to variance features) of the time series.

Specification of S, A, Ω, and $R(s, a)$

The landmine detection problem can be viewed as a generalization of the tiger problem (Kaelbling, Littman, & Cassandra 1998). Each mine type represents a type of tiger, and each clutter type represents a type of non-tiger (reward). The robot can observe sensor readings (listening in the tiger problem) to gain information or make a declaration with regard to the presence or absence of a mine (opening the

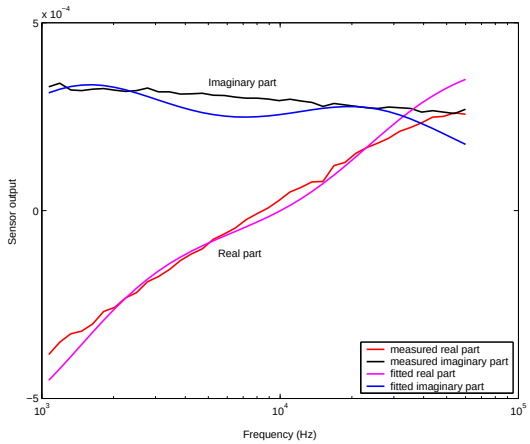
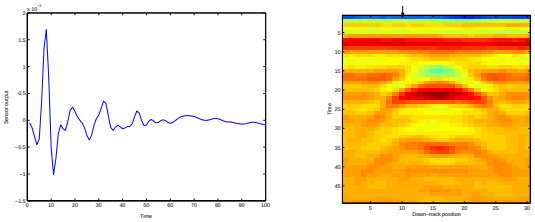


Figure 1: EMI response and model fit when the sensor is above a metal mine.



(a) (b)

Figure 2: The GPR response when the sensor is above a plastic mine. (a) Amplitude vs. time signal in one position. Units in time are 0.05 ns. The first peak corresponds to the reflection from the ground surface. (b) 2-dimensional scan of a plastic mine signature. Units in down-track position are 2 cm. The arrow indicates the position where a sensor measures the signal in (a).

door in the tiger problem) to complete the present detection phase. When learning the policy, the problem resets immediately after a declaration is made, and a mine or clutter is randomly presented to the robot. This corresponds to the robot randomly encountering a mine or clutter in the next detection phase.

Across all five types of mines and clutter considered, we define a total of 29 states, i.e. $S = \{1, 2, \cdots, 29\}$. The 29 states are divided into 5 disjoint subsets: $S = S_m \cup S_p \cup S_{c_1} \cup S_{c_2} \cup S_{c_0}$, denoting metal mine, plastic mine, Type-1 clutter (large-sized), Type-2 clutter (small-sized), and "clean", respectively. The number of states in each of the five subsets are 9, 9, 9, 1, and 1, respectively. The multiple states of metal mine, plastic mine, Type-1 clutter represent their respective 9 annulus sectors. Definition of the states is illustrated in Figure 3(a).

In most cases, a mine and clutter is cylindrically symmetric and is buried with its axis perpendicular to the ground surface. This implies that the robot will not distinguish states $1, 2, 3, 4$ (which are approximately equidistant to the metal mine) by observing a single sensor reading in each respective state. However, by remembering its past observations and actions, the robot will be able to tell apart these ambiguous states.

The robot has 15 possible actions, i.e., $A = \{1, 2, \cdots, 15\}$, of which the first 10 are sensing actions and the rest are declaration actions. Each sensing action has the format of "$move$ and then $sense$", where $move \in \{$stay, walk south, walk north, walk east, walk west$\}$ and $sense \in \{$sense with EMI, sense with GPR$\}$, with EMI representing an electromagnetic induction sensor and GPR a ground penetrating radar. Of the 5 declaration actions, one declares the present sub-area (where the robot currently is) to be "clean", and four respectively declare that there is a "metal mine", "plastic mine", "Type-1 clutter", or "Type-2 clutter" buried beneath the present sub-area.

The set of possible observation Ω is obtained as the codebook resulting from vector quantization (Gersho & Gray 1992) of the continuous sensor signatures. Each of the two sensors, EMI and GPR, generates its own codebook independently, resulting in two disjoint codebooks, which are taken a union over to produce Ω.

The reward function $R(s, a)$ is specified as follows. Denote by m any of the 9 states for a metal mine, by p any of the 9 states for a plastic mine, and by c_1 any of the 9 states for a Type-1 clutter. Denote by c_2 the Type-2 clutter and by c_0 the "clean" state. See Figure 3(a) for definition of the states. Denote by A_t the action of declaring the present sub-area to be the state of t. Then $R(s = t,\ a = A_t) = 10$, for $t = m$, p, c_1, c_2, or c_0; $R(s = m$ or $p,\ a = A_{c_1}$ or A_{c_2} or $A_{c_0}) = -100$; $R(s = c_1$ or c_2 or $c_0,\ a = A_m$ or $A_p) = -50$; $R(s = m,\ a = A_p) = 5$; $R(s = p,\ a = A_m) = 5$. All the remaining entries of $R(s, a)$ are zero.

Estimation of $T(s, a, s')$ and $O(a, s', o)$

The two sensing actions involving "stay" do not cause state transitions, hence $T(s, a, s')$ is an identity matrix when a is "stay and sense with GPR" or "stay and sense with EMI". All remaining sensing actions can result in transitions from one state to another. Assuming that the robot travels the same distance in each step and that the robot's position is uniformly distributed in any given state, the probabilities of these transitions are easily determined by using an elementary geometric probability computation. Figure 3(b) illustrates how the transition probabilities for the two sensing actions involving "walk south" are computed.

Computing $T(s, a, s')$ and $O(a, s', o)$ requires prior knowledge of the possible mines and clutters. This poses no problem here, as we have the templates of the possible mines and clutter, which can be employed to compute $T(s, a, s')$ as well as collecting the training signatures for estimating $O(a, s', o)$.

Experimental Results

We consider a robot navigating in three simulated mine fields. The EMI and GPR data are pre-collected over a $1.6 \times 1.6\ m^2$ per simulated mine field, with sensor data collected at a 2 cm sample rate in two coordinate dimensions.

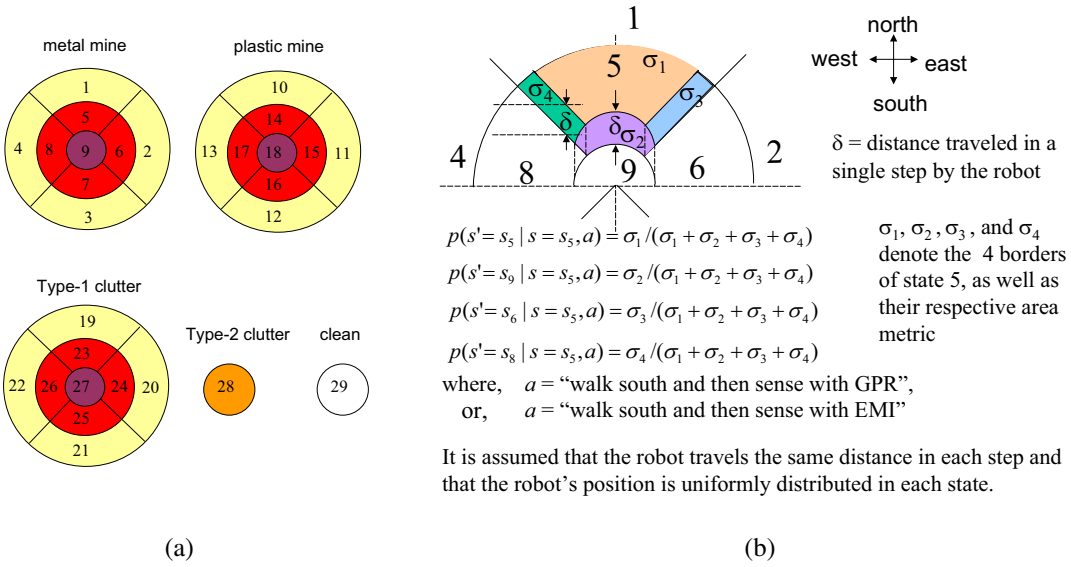


Figure 3: (a) Definition of the states for the minefiel navigation problem. Metal mine, plastic mine, and Type-1 clutter (large-sized) are each modeled by 9 states, indexed 1 to 9, 10 to 18, and 19 to 27, respectively; Type-2 clutter (small-sized) is modelled by a single state (state 28); state 29 is used to indicate "clean" (i.e., there are no mine or mine-like objects buried underground). (b) Illustration of the geometric method in computing the state transition probabilities $T(s, a, s')$ when a is one of the two sensing actions involving "walk south". It is assumed that the robot travels the same distance in each step and that the robot's position is uniformly distributed in any given state.

The pre-collected data are used to simulate the data collected by an autonomous two-sensor agent, as it senses within the mine field. The first simulated mine field is shown in Figure 6.

Clearly, to avoid missing landmines the robot should search almost everywhere in a given mine field. However, we hope that the robot can actively decide where to sense as well as which sensor to use, to minimize the detection cost. Considering these two requirements together, we assign a basic path as shown in Figure 4 (dark blue curve with arrows). The basic path defines the lanes as indicated by light blue in the figure, and the robot is restricted to move along the lanes by taking actions within the lanes. The basic path restrains the robot from moving across the lanes, and the robot defines sectors along each lane as being characterized by one of the mines/clutter, including clean, while moving in an overall direction consistent with the arrows in Figure 4. The distance between two neighboring basic paths should be less than the diameter of a landmine signature.

It is possible that after many measurements in one local area, the agent still cannot make a declaration. For example, this can occur if the model we build does not fit the data in this area, possibly because our model does not include the current underground target. More measurements do not help to make a better decision. If this happens, it is better to say "I do not know" rather than continue sensing or make a reluctant declaration. We let the robot declare unknown in this situation, while in the lifelong learning algorithm the oracle is employed.

In the offline-learning approach the training data are given

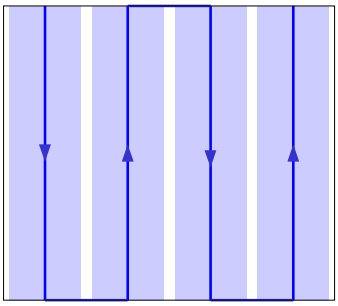
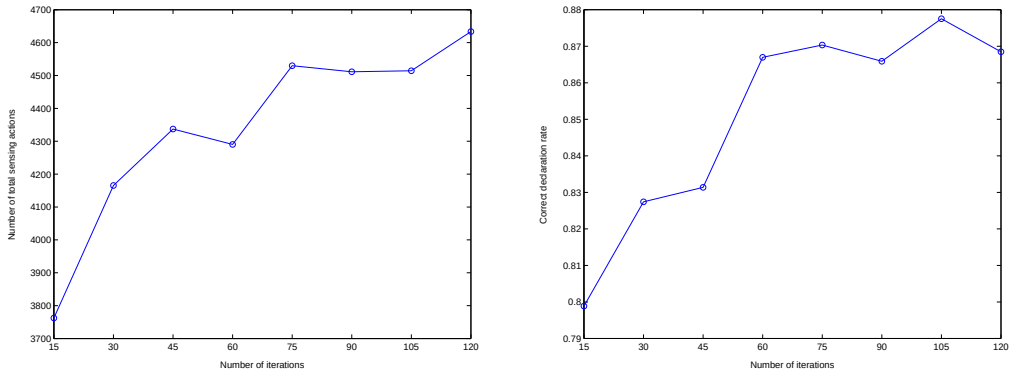


Figure 4: Robot navigation path in a mine field. The dark blue curve is the basic path, which defines the lanes as indicated by light blue. The robot is restricted to move along the lanes by taking actions within the lanes. The basic path restrains the robot from moving across the lanes.

in advance, and the training phase and test phase are separate. We use Mine Field 1 (Figure 6) as the training data to learn the model and the policy, and then test our method on all three mine fields. The training data and test data match well in that the three mine fields contain almost the same types of metal mines, plastic mines and clutter. The clutter includes metal clutter (soda can, shell, nail, coin, screw, lead, rod, and ball bearing) and nonmetal clutter (rock, bag of wet sand, bag of dry sand, and a CD).



(a) (b)

Figure 5: Detection performance as a function of number of iterations when learning the policy. (a) Number of total sensing actions. (b) Correct declaration rate.

Table 1: Detection results on three mine fields

		Mine Field 1	Mine Field 2	Mine Field 3
Ground truth	Number of mines (metal+plastic)	5 (3+2)	7 (4+3)	7 (4+3)
	Number of clutter (metal+nonmetal)	21 (18+3)	57 (34+23)	29 (23+6)
Detection result	Number of mines missed	1	1	2
	Number of mines missed	2	2	2

Model training and policy design

Using Mine Field 1 as the training data set the POMDP model is built and the policy is learned by PBVI. The number of sensing actions and the correct declaration rate as a function of iteration number when determining the policy are plotted in Figure 5. The correct declaration rate is defined as the ratio of the number of correct declarations relative to the number of all declarations. Note that the correct rate is not equivalent to probability of detection since one landmine could be declared multiple times, and the correct declaration of clutter or clean is also counted in the correct rate. However, it does reflect the detection performance by comparing declaration position and ground truth. From Figure 5, after 75 iterations and five belief expansion phases, the PBVI-learned policy becomes stable.

Landmine detection results

The stationary policy from the last subsection is then used to navigate the robot in three simulated mine fields. The ground truth and detection results are summarized in Table 1. As an example, the layout of Mine Field 1, the declaration result and a zoom-in of sensor choices are shown in Figure 6. Note that one target may be declared several times.

Missed landmines are usually caused by one of the following two reasons: the mine has very weak signal in both EMI and GPP responses, such as a small anti-personnel mine, which is a low-metal content mine; or the mine is very close to some large metal clutter, so that the clutters strong response hides the weak signal of the mine. From Figure 6(c), we see that the policy selects GPR sensors to interrogate plastic mines, while it prefers EMI sensors when metal mines are present. This verifies the policy to some degree since the EMI sensor is almost useless for detecting plastic mines, but is good for detecting metal mines. We also see that on the clean area or at the center of a landmine, a declaration is made only based on very few sensing actions, usually two or three, since it is relatively easy for the robot to estimate its current states. However, at the edge of a landmine, where there is an interface between two objects (the landmine and the clean), the robot usually requires many Number of total sensing actions sensing actions to make a declaration. The robot requires, on average, approximately 4500 sensing actions in one mine field; the correct declaration rate is about 0.87 (see Figure 5). As a comparison, if a myopic policy is applied, where the agent considers only one step ahead to select actions, a total of around 8000 sensing actions are needed, and a correct declaration rate of 0.82 is achieved. Note that if one senses on every grid point using both sensors, the total number of measurements is $2 \times 800^2 = 12800$.

Conclusions

We have addressed the problem of employing ground-penetrating radar (GPR) and electromagnetic induction (EMI) sensors placed on a single platform, with the objective of performing adaptive and autonomous sensing of landmines. The problem has been formulated in a partially observable Markov decision process (POMDP) setting, under the assumption that adequate and appropriate data are available for learning the underlying POMDP models, with which policy design can be effected. The algorithm has been tested, with encouraging performance, on measured EMI and GPR data from simulated mine fields.

The assumption that adequate training data are available

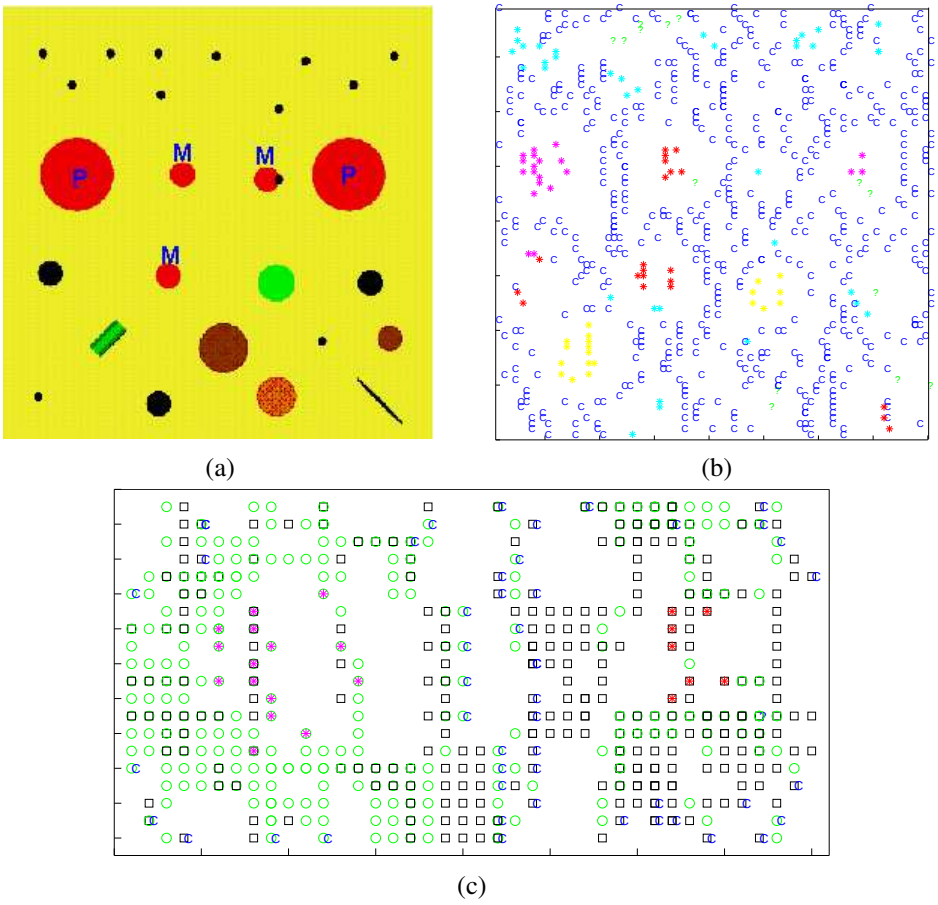


Figure 6: Ground truth and detection details in Mine Field 1. (a) Ground truth. The red circles are landmines, with "M" and "P" indicating metal mine and plastic mine, respectively; the other symbols represent clutter. Black dots are small metal segments and the rest are large-sized metal or nonmetal clutter. (b) Declaration result. The blue "C" means a declaration of "clean", the green "?" means "unknown", and the stars with various colors represent declarations of mines or clutter. Red star: metal mine; pink star: plastic mine; yellow star: Type-1 clutter; cyan star: Type-2 clutter. (c) Sensor choice in the broken-lined rectangular area shown in (b). The black square means sensing with EMI sensor and the green circle means GPR sensor. It can be seen that the policy prefers the GPR sensor for plastic mine (left half in (c)) and the EMI sensor for metal mine (right half in (c)).

is often inappropriate, and therefore in the next phase of this work we will consider a lifelong-learning algorithm in which little if any *a priori* information is assumed with regard to the mines, clutter and soil conditions. The formulation considered for this latter case will be based on the recently developed MEDUSA algorithm (Jaulmes, Pineau, & Precup 2005).

References

Abdel-Samad, A. A., and Tewfik, A. H. 1999. Search strategies for radar target localization. *Proc. International Conf. Image Proc.* 3:862–866.

Gadar, P. D.; Mystkowski, M.; and Zhao, Y. 2001. Landmine detection with ground penetrating radar using hidden markov models. *Geoscience and Remote Sensing* 39:1231–1244.

Gersho, A., and Gray, R. M. 1992. *Vector Quantization and Signal Compression*. Kluwer Academic Press/Springer.

Jaulmes, R.; Pineau, J.; and Precup, D. 2005. Active learning in partially observable markov decision processes. In *Proceedings of ECML*, 601–608.

Kaelbling, L.; Littman, M.; and Cassandra, A. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101:99–134.

Kastella, K. 1997. Discrimination gain to optimize detection and classification. *IEEE Trans. Syst., Man, Cybernetics Part A: System and Humans* 27:112–116.

MacDonald, J. 2003. *Alternatives for Landmine Detection*. Rand Corporation.

Pineau, J.; Gordon, G.; and Thrun, S. 2003. Point-based value iteration: An anytime algorithm for POMDPs. In *IJCAI*, 1025 – 1032.

Adaptation of the Simulated Risk Disambiguation Protocol to a Discrete Setting

Al Aksakalli, Donniell E. Fishkind, and Carey E. Priebe
Department of Applied Mathematics and Statistics
Whiting School of Engineering, Johns Hopkins University
Baltimore, Maryland 21218–2682
{ala,fishkind,cep}@jhu.edu

Abstract

Suppose a spatial arrangement of possibly hazardous regions needs to be speedily and safely traversed, and there is a dynamic capability of discovering the true nature of each hazard when in close proximity of it; the traversal may enter the associated region only if it is revealed to be nonhazardous. The problem of identifying an optimal policy for where and when to execute disambiguations so as to minimize the expected length of the traversal can be cast both as a completely observed Markov decision process (MDP) and a partially observed Markov decision process (POMDP) and has been proven intractable in many broad settings. In this manuscript, we adapt the basic strategy of a policy called the *simulated risk disambiguation protocol* of Fishkind *et al.* (2006) to a different, discretized setting (a Canadian Traveller Problem with dependent edge probabilities), and we compare the performance of this adapted policy against the performance of the optimal policy—on a class of instances that are small enough for the optimal policy to be computed. On random such instances, the adapted simulated risk disambiguation protocol performed nearly as well as the optimal protocol, and used significantly less computational resources.

Overview

Suppose there is a set of (possibly overlapping) *hazard regions* $H_i \subseteq \mathbf{R}^2$, for $i = 1, 2, \ldots, N$, each region marked with the probability ρ_i that H_i is a *true hazard* (as opposed to a *false hazard*), and assume that hazard regions are true hazards independently of each other. Now, suppose a *starting point* s and *destination point* d are given in $\mathbf{R}^2$, and the decision maker's objective is to traverse a shortest continuous curve from s to d avoiding all true hazards, i.e. the curve is constrained to $(\bigcup_{i\in\mathcal{I}} H_i)^C$, where $\mathcal{I} \subseteq \{1, 2, \ldots, N\}$ is the set of indices of true hazards. While $\mathcal{I}$ is unknown to the decision maker at the outset (in particular, the probability distribution of $\mathcal{I}$ is specified by the marks ρ_i), when the curve is on the boundary ∂H_i of any hazard region the decision maker has the dynamic ability to *disambiguate* the region to discover if H_i is a false or true hazard and, accordingly, the curve may or may not proceed through H_i. Each execution of a disambiguation adds a fixed cost $c \geq 0$ to the length of the s-d curve traversed, and it may be useful to sometimes assume that there are a fixed limit of $K \geq 0$ disambiguations that may be performed. How to optimally perform the disambiguations so as to minimize the expected length of the traversal is the *random disambiguation path* (RDP) problem in (Priebe *et al.* 2005).

The RDP problem is a minor modification of the Stochastic Obstacle Scene Problem (SOSP) of (Papadimitriou & Yannakakis 1991), who also describe a discrete version of SOSP that they call the Canadian Traveller Problem (CTP). In CTP, the goal is to minimize the expected traversal length from a starting vertex to a destination vertex in a finite graph whose edges are marked with their respective probabilities of being traversable, and every edge's actual status can be dynamically discovered only when encountered. Papadimitriou and Yannakakis prove the intractability of several variants of SOSP and CTP. (For more on CTP see (Bar-Noy & Schieber 1991)).

CTP is also a special case of the Stochastic Shortest Paths with Recourse (SPR) problem of (Andreatta & Romeo 1988), who present a stochastic dynamic programming formulation for SPR and note its intractability. (Polychronopoulos & Tsitsiklis 1996) also present a stochastic dynamic programming formulation for SPR and they prove the intractability of several variants. (Provan 2003) proves that SPR is intractable even if the underlying graph is directed and acyclic.

In (Fishkind *et al.* 2006), we proposed a class of policies, called *simulated risk disambiguation protocols*, for RDP. In this manuscript, we adapt that basic approach for use on a discretized version of RDP which, in effect, is a Canadian Traveller Problem with dependent arc probabilities. We compare the performance of our adapted policy to the performance of the optimal policy, which we can obtain exactly for relatively small—but nontrivial—instances.

The rest of this manuscript is organized as follows. We next describe a discretization of RDP and mention how it can be cast as a partially or completely observable Markov decision process. Then we adapt the simulated risk disambiguation protocol for use in the discrete setting. Finally, we compare the performance of the adapted simulated risk disambiguation protocol against the optimal policy obtained by a standard implementation of value iteration algorithm on

the Markov decision process formulation.

A Discrete Version of RDP

Because of its continuous setting, the simulated risk disambiguation protocol in (Fishkind *et al.* 2006) is not comparable to the existing heuristics for CTP and SPR, nor are optimal policies readily computable for all but the most trivial instances. With this in mind, we adapt the simulated risk random disambiguation protocol in (Fishkind *et al.* 2006) to a related, discrete setting in which the protocol adaptation's performance may be compared to the optimal policy for relatively small but nontrivial instances.

The discretization of RDP we will consider here is, for simplicity and convenience, a subgraph of the integer lattice $\mathbf{Z}^2$. Specifically, it is the graph G whose vertices are all of the pairs of integers x, y such that $1 \leq x \leq x_{\max}$ and $1 \leq y \leq y_{\max}$, where $x_{\max}$ and $y_{\max}$ are given integers. There are edges between all pairs of vertices of the form x, y and $x+1, y$, and there are edges between all pairs of vertices of the form x, y and $x, y+1$. The hazard regions H_i, for $i = 1, 2, \ldots, N$, are open discs in $\mathbf{R}^2$. (See Figure 1 and Figure 2.) One vertex in G is designated as the starting point s, another vertex in G is designated as the destination point d, and the decision maker is to walk from s to d in G, only traversing edges that do not intersect any true hazards. If an edge intersects any ambiguous hazard region, then a disambiguation of the hazard region may be performed from either of the edge's endpoints that is outside of the hazard region. As before, the goal is to develop a policy for traversing and performing disambiguations that minimizes the expected length of the traversal. We call our problem *discrete RDP* (DRDP).

Markov Decision Process Formulation of DRDP

We next describe how the DRDP problem can be modelled as a partially or completely observable Markov decision process, in a manner similar to that which is done in (Blei & Kaelbling 1999).

An *information vector* $I \in \{\text{“a”},\text{“t”},\text{“f”}\}^N$ keeps track of the decision maker's current knowledge of the hazard regions' status; specifically, for all $i = 1, 2, \ldots, N$, the ith entry of I is “a”, “t”, or “f” according as H_i is currently ambiguous, true, or a false hazard. Let V denote the set of endpoints of edges in G that intersect any boundary of any hazard region; in particular, these are the vertices of G at which the decision maker may execute disambiguations. Now, the set of states is $(V \times \{\text{“a”},\text{“t”},\text{“f”}\}^N) \bigcup \{s, d\}$, which represent possible locations on the lattice at which the decision maker may be at a particular moment in time, coupled with the possible information vectors that may describe the decision maker's knowledge at that moment. The set of actions is the set of ordered pairs (v, i) such that vertex $v \in V$ is the endpoint of an edge which intersects ∂H_i; this pair represents where the next disambiguation is to occur, and which hazard region will be disambiguated.

The reward for any appropriate action at any particular state is the negative of the shortest path distance (avoiding all ambiguous and true hazards indicated by the information vector of the state) between the vertex identified in the state and the vertex identified in the action; also subtract the disambiguation cost c if the vertex identified in the action is not d. The destination d is an absorbative state for which there is a one-time and very large reward for entering. Given a state and action, the state transitioned into is the vertex identified in the action and the information vector of the previous state, updated to indicate that the hazard H_i identified in the action is true or false with respective probabilities ρ_i and $1 - \rho_i$.

The above set of states, actions, rewards, and transition distributions comprise a Markov decision process with K stages (or N stages if there is no limit K on the number of allowed disambiguations). In the rest of this paper, we will compute optimal policies in this formulation using the standard stochastic dynamic programming technique of value iteration for relatively small but nontrivial instances. Let p^* denote the s, d-curve traversed by the optimal policy; since the trajectory under the optimal policy is still random, p^* is an (s, d)-walk-valued random variable, and we denote its expected length $\mathrm{E}(p^*)$.

Also, DRDP may be cast as a partially observed Markov decision process by trimming the set of information vectors to be just $\{\text{“t”},\text{“f”}\}^N$, and by folding the ambiguity of hazards into ambiguity of the information vector, hence the partial observability of the state.

Adapting the Simulated Risk Disambiguation Protocol

We next introduce the adaptation of the simulated risk disambiguation protocol. In our framework, the traversal never enters hazard regions while they are still ambiguous or are known to be true hazards. The paradigm of the simulated risk disambiguation protocol is—for the sole purpose of deciding where to disambiguate next—to temporarily pretend (*simulate*) that the ambiguous hazards are riskily traversable.

Under this simulation of risk, for any s, d walk W, the Euclidean length of W, $\ell^{\mathcal{E}}(W)$, is the number of edges in W (since each edge in our lattice clearly has Euclidean length 1). The *risk length* of W is defined as

$$\ell^{\mathcal{R}}(W) := -\log \prod_{H_i : H_i \cap W \neq \emptyset} (1 - \rho_i).$$

This negative logarithm of the probability that W is permissibly traversable is a measure of the risk in traversing W—if you were willing to take on risk. An *undesirability function* is any function $g : \mathbf{R}_{\geq 0} \times \mathbf{R}_{\geq 0} \to \mathbf{R}$ which is monotonically nondecreasing in its arguments; that is to say, for all $r_1, r_2, t_1, t_2 \in \mathbf{R}_{\geq 0}$ such that $r_1 \leq r_2$ and $t_1 \leq t_2$, it holds that $g(r_1, t_1) \leq g(r_2, t_2)$. The number $g(\ell^{\mathcal{E}}(W), \ell^{\mathcal{R}}(W))$ is thought of as a measure of the undesirability of W in the sense that, if you were required to traverse from s to d in G under the simulation of risk and without a disambiguation capability, you would select the walk

$$\phi_g := \arg \min_{s,d \text{ walks } W} g(\ell^{\mathcal{E}}(W), \ell^{\mathcal{R}}(W)).$$

The *adapted simulated risk disambiguation protocol* associated with g would have the decision maker traverse ϕ_g

from s until its first ambiguous edge, say, e is encountered at, say, vertex v, and say e intersects region H_i. At this point (since we may not take on risk in actuality) disambiguate the hazard region H_i and repeat this whole procedure using v as the new s, and then either removing H_i or setting $\rho_i := 1$ according as H_i was just discovered to be a false or true hazard. (If at some point the limit K on the number of disambiguations has been reached, then the shortest unambiguously permissible path is then taken to d.)

The simplest undesirability functions are the linear ones, where $g(r,t) := r + \alpha \cdot t$ for some given constant $\alpha > 0$. To find ϕ_g in this particular case, we just need to find a deterministic shortest s, d path in G (via Dijkstra's algorithm, say) where each edge in G is weighted[1] with

$$w(e) := 1 - \alpha \cdot \frac{1}{2} \sum_{i=1}^{N} \#\text{comp}(e \backslash H_i) \cdot \mathbf{1}_{e \cap H_i \neq \emptyset} \cdot \log(1 - \rho_i),$$

where $\mathbf{1}$ is the indicator function (taking value 1 or 0 according as its subscripted expression is true or false) and comp$(\cdot)$ is the number of connected components of its argument. See illustration in Figure 1 with $N = 4$. Corresponding edge weights are shown in Table 1.

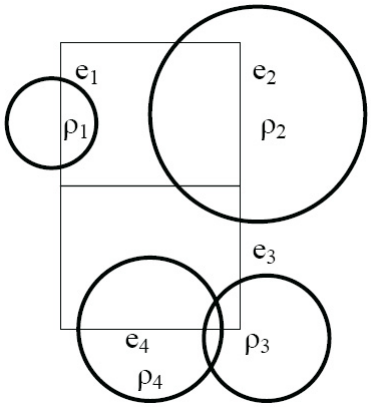


Figure 1: Illustration with $N = 4$

Edge	Edge weight
e_1	$1 - \alpha \log(1 - \rho_1)$
e_2	1
e_3	$1 - \alpha(1/2)(\log(1 - \rho_2) + \log(1 - \rho_3))$
e_4	$1 - \alpha(1/2)(\log(1 - \rho_3) + 2\log(1 - \rho_4))$

Table 1: Weights of edges in Figure 1

For a fixed $\alpha > 0$, denote by p_α the s, d walk traversed under the adapted simulated risk disambiguation protocol; p_α is an s, d-walk-valued random variable, since its realization depends on the outcomes of the dictated disambiguations. We will denote by Ep_α the expected length of this walk.

[1] We are assuming s and d are not inside any hazard region, and that ϕ_g never revisits a hazard region.

Computational Experiments

In this section, we evaluate the performance of the the adapted simulated risk disambiguation protocols against the optimal policy, the latter obtained by a standard implementation of value iteration algorithm on the Markov decision process model described previously.

For all of the experiments in this Section, the lattice used is $x_{\max} = 40$ by $y_{\max} = 20$, with $s = (20, 20)$ and $d = (20, 1)$. Each hazard region is a disc with radius 5.5 units, and the disc's centers are sampled from a uniform distribution on the pairs of integers in $[7, 34] \times [7, 14]$; in particular, this ensures that there is always a permissible path from s to d. Probabilities ρ_i of the hazard regions being true are sampled from a uniform distribution on $[0, 1]$. The cost of disambiguation is taken here as $c = 1.5$. The environment is illustrated in Figure 2 with $N = 7$, $K = 1$.

For the instance shown in Figure 2, the adapted simulated risk disambiguation protocol using $\alpha = 1$ calls for traversing to $(20, 15)$, at which point the hazard region H_4 centered at $(19, 9)$ is disambiguated. In case the hazard turns out to be false, the decision maker traverses directly to the destination at a total cost of 20.5. Otherwise, the decision maker traverses to the destination avoiding all the hazard regions, at a total cost of 50.5. (Both walks are illustrated in black on Figure 2.) Here, we had $\rho_4 = .2230$ thus, in particular, the expected length of the s, d-traversal is $(1 - .2230)(20.5) + (.2230)(50.5) = 27.19$. This protocol turns out to be the overall optimal policy.

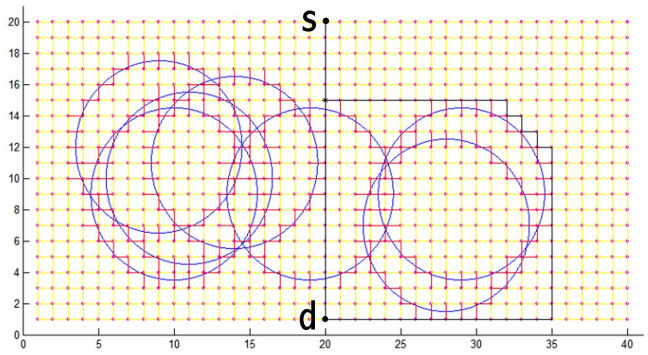


Figure 2: An experimental realization with $N = 7, K = 1$

Similar to what was done in (Fishkind *et al.* 2006), values of α minimizing Ep_α are computed numerically by evaluating Ep_α for a mesh of values of α—starting at $\alpha = 1$, incrementing α successively by 5 units until α is large enough that no disambiguations are performed. We will now denote $\hat{p}^* := p_{\alpha^*}$, where α^* is the value of α minimizing Ep_α.

Table 2 compares E$\hat{p}^*$ (the expected length of the best adapted simulated risk disambiguation protocol) to Ep^* (the expected length of the overall optimal policy) for 50 experiment realizations under each N, K combination listed. The second column shows the percentage of simulations where E$\hat{p}^*$ = Ep^*, in which case the family of adapted simulated risk disambiguation protocols contains the overall optimal

policy. The next column shows the percentage of the simulations where the optimal policy was to perform no disambiguations. The next column shows the mean of the relative errors $\frac{\mathrm{E}\hat{p}^* - \mathrm{E}p^*}{\mathrm{E}p^*}$ for the 50 experiment realizations in each N, K.

	% where $\mathrm{E}\hat{p}^* = \mathrm{E}p^*$	% where p^* didn't disambig.	mean relative errors $\frac{\mathrm{E}\hat{p}^* - \mathrm{E}p^*}{\mathrm{E}p^*}$	VI exec. time
$K = 1$:				
$N = 4$	86	28	.0097	12 sec
$N = 5$	86	30	.0076	15 sec
$N = 6$	72	22	.0103	20 sec
$N = 7$	66	22	.0114	33 sec
$N = 8$	66	36	.0159	48 sec
$K = 2$:				
$N = 4$	78	30	.0058	4 min
$N = 5$	66	22	.0065	6 min

Table 2: Comparison of $\hat{p}^*$ to p^*; for $K = 1$, $N = 4$ the overall optimal policy was indeed an adapted simulated risk disambiguation protocol in 86% of the experiments.

As Table 2 indicates, solutions found by the simulated risk protocol are quite comparable to the optimal solutions. Among all the simulations we performed, the simulated risk protocol found the optimal solution 74.3% of the time and the mean relative error of these simulations was .0096.

The last column in Table 2 shows the execution time for a single instance of the value iteration algorithm used to identify an overall optimal policy and to compute Ep^*, on a 3.5 gigahertz personal computer with 1 gigabyte memory. We observed that value iteration required significant computational resources even for small instances. In fact, value iteration execution time was over an hour for $N = 7, K = 3$ and for $N = 10, K = 1$, value iteration did not run at all due to insufficient memory. This was in sharp contrast to the identification of $\hat{p}^*$ and the computation of E$\hat{p}^*$, which took a negligible amount of time in all of these experiments. Computing E$\hat{p}^*$ continued to take a negligible amount of time for much, much larger values of N and K.

Summary and Conclusions

In this manuscript, we adapted the simulated risk disambiguation protocol from the RDP setting to the discrete RDP setting, which is essentially a Canadian Traveller Problem with dependent edge probabilities. We cast the problem as a Markov decision process and, via value iteration, obtained optimal policies for relatively small but nontrivial instances. Against these optimal policies, we compared the performance of adapted simulated risk disambiguation protocols and discovered that much of the time the adapted risk simulation disambiguation protocols were indeed optimal and, in general, compared very favorably. Furthermore, negligible computing time was needed—compared to that expended on the computation of the optimal policy.

References

Andreatta, G., and Romeo, L. 1988. Stochastic shortest paths with recourse. *Networks* 18(3):193–204.

Bar-Noy, A., and Schieber, B. 1991. The canadian traveller problem. *SODA '91: Proceedings of the Second Annual ACM-SIAM Aymposium on Discrete algorithms*.

Blei, D. M., and Kaelbling, L. 1999. Shortest paths in a dynamic uncertain domain. *IJCAI Workshop on Adaptive Spatial Representations of Dynamic Environments*.

Fishkind, D.; Priebe, C.; Giles, K.; Smith, L.; and Aksakalli, V. 2006. Disambiguation protocols based on risk simulation. *IEEE Transactions on Systems, Man, and Cybernetics* Part A:to appear.

Papadimitriou, C., and Yannakakis, M. 1991. Shortest paths without a map. *Theoretical Computer Science* 84:127–150.

Polychronopoulos, G. H., and Tsitsiklis, J. N. 1996. Stochastic shortest path problems with recourse. *Networks* 27(2):133–143.

Priebe, C.; Fishkind, D.; Abrams, L.; and Piatko, C. 2005. Random disambiguation paths for traversing a mapped hazard field. *Naval Research Logistics* 52:285–292.

Provan, J. S. 2003. A polynomial-time algorithm to find shortest paths with recourse. *Networks* 41(2):115–125.