# Adaptive Planning for Markov Decision Processes with Uncertain Transition Models via Incremental Feature Dependency Discovery

N. Kemal Ure, Alborz Geramifard, Girish Chowdhary, and Jonathan P. How

Laboratory for Information and Decision Systems, MIT
http://www.lids.mit.edu
Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA, USA
{ure,agf,girishc,jhow}@mit.edu

**Abstract.** Solving large scale sequential decision making problems without prior knowledge of the state transition model is a key problem in the planning literature. One approach to tackle this problem is to learn the state transition model online using limited observed measurements. We present an adaptive function approximator (incremental Feature Dependency Discovery (iFDD)) that grows the set of features online to approximately represent the transition model. The approach leverages existing feature-dependencies to build a sparse representation of the state transition model. Theoretical analysis and numerical simulations in domains with state space sizes varying from thousands to millions are used to illustrate the benefit of using iFDD for incrementally building transition models in a planning framework.

## 1 Introduction

Increasing the level of autonomy for unmanned aerial vehicles (UAVs) through planning algorithms is needed to tackle real-life missions such as persistent surveillance, maintaining wireless network connectivity, and search and rescue [21, 27, 31]. One of the common themes amongst these missions is the presence of stochasticity, such as uncertainty in the outcome of interactions with the environment or the vehicle dynamics. One typical approach for solving these stochastic decision making problems is to cast them as Markov Decision Processes (MDPs) [23] and then use reinforcement learning (RL) [30] methods to generate policies without having to know the transition model. However, a hurdle in applying RL to real-world problems is that RL methods typically require many interactions with the world in order to find good policies. Model-based RL techniques address this sample complexity problem by fitting an approximate model to the data first and then generating simulated samples from the model [29, 33]. However, this approach requires a suitable estimation model with appropriate basis functions. This paper presents a scalable transition model estimation method that can be used in a model-based RL framework for solving MDPs with state-correlated uncertainty.

For motivation, consider a robot navigating from a starting point to a goal location using GPS signals. The GPS is not fully reliable and may fail in each location with a certain probability. A succesful policy guides the robot away from locations with poor signal reception. Yet, as we will see in the numerical results, a planning-modeling scheme that assumes a uniformly distributed failure rate for the GPS can lead to poor performance.

A straightforward approach for modeling the uncertainty in such problems is to estimate the parameters of the model for every state separately (*i.e.,* use a tabular representation). However for large state spaces, this may become intractable. The main aim of this paper is to present a planning/learning technique to form a good approximation of state-dependent uncertainties with low sample complexity. To this effect, we employ an adaptive function approximation technique to estimate the uncertainties that allows flexible representations and alleviates the problem of hand-picking a set of fixed features. The representation starts with a small number of state correlated basis (features) and expands the representation in regions where model parameters are not well captured. This adaptive function approximator is based on the recently developed incremental feature dependency discovery (iFDD) algorithm [13]. By using iFDD for model parameter estimation and bootstrapping techniques for replanning, we demonstrate a substantial sample complexity reduction in learning good policies. In addition, the proposed methodology also possess asymptotic convergence properties. The applicability of the proposed method to integrated model estimation and planning is experimentally demonstrated in a gridworld problem, a block building domain, and a persistent search and track (PST) mission. Simulation results show that the representations learned by iFDD are sufficiently rich and result in a substantial reduction in the sample complexity.

## 2 Related Work

### 2.1 Robust Dynamic Programming

If the uncertain transition model is assumed to be lying on a priori known set, a robust policy can be obtained by considering the worst-case situation within this set. This approach is usually known as robust dynamic programming [15], and different methods have been developed based on how the uncertainty set is modeled and how the model is selected from the uncertainty set [22, 9, 4]. Although policies generated with these methods usually prevent the catastrophic effects of model-environment mismatch, they often lead to conservative solutions, since the assumed uncertainty set is usually a pessimistic estimate of the actual uncertainty representation. For example in the context of the robot navigation, this approach may assume a uniform high GPS failure rate for all states. Hence the planner does not allow the agent to explore the environment.

### 2.2 Adaptive Dynamic Programming (ADP)

Methods in this category start with an initial representation of the model of the system and updates this model as more data is observed from the environment [16]. After updating the model representation, a new policy is obtained by applying a dynamic programming (DP) algorithm suitable to work in real time, such as asynchronous dynamic programming [14]. A successful application of such an algorithm to PST mission in hardware setting can be found in [6], where the authors improve the policy online by estimating fuel dynamics of UAVs. Bertuccelli [5] managed to improve the performance of these algorithms by combining robust and adaptive methods, which resulted in an algorithm that estimates the uncertainty set online using the observed data. Existing ADP methods estimate a fixed set of transition parameters (which can be as large as the state space). Hence such methods are limited to low-dimensional small-scale systems.

### 2.3 Model Based Reinforcement Learning (MBRL)

Basic idea of MBRL [7] is similar to ADP, however development of these methods in different communities led to algorithms with different behaviors and applications. Dyna architecture [33, 29] builds an approximate model of the system based on the observed data and then uses this model to generate samples. These algorithms tackle large state space by applying approximation to system dynamics, however finding a good representation often requires domain expertise.

### 2.4 Bayesian Non-parametric Models

Bayesian non-parametric models (BNPM)[11] are probabilistic models that can grow their complexity in response to measured data. Thus when they are used as prior distributions over models in MBRL setting, they can alleviate the problem of finding a good representation to a certain degree since the expressiveness of the representation grows as more samples are observed. In [2], BNPMs are used for state clustering for an MRBL algorithm and [17] uses BNPMs to model a rich class of motion patterns. Although these methods offer more flexible models than fixed representations, they also tend to have higher sample complexity and they may lack asymptotic convergence guarantees. Gaussian processes and kernel filters are classes of non-parametric models that leverage the theory of reproducing kernel Hilbert spaces (see e.g. [24, 19]) for regression and classification problems. The idea is to associate each measured state with a kernel so that a representation can be built online. Kernel methods need several sparsification tools to ensure that the chosen set of kernels does not become intractable as different parts of the state space are explored. Performing such sparsification online is a difficult problem, because existing methods require optimization over a set of kernels which can become computationally intensive. An incremental model expansion method based on growing hidden Markov models had been proposed by [10], however it has been verified only in learning and prediction of motion patterns.

This paper addresses the aforementioned shortcomings of existing methods by developing a model expansion method that has asymptotic convergence properties, which is sample efficient and is scalable to high dimensional large-scale problems.

## 3 Problem Definition

The problem of sequential decision making under uncertainty is formulated as an MDP, which is defined as the tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}^a_{ss'}, \mathcal{R}^a_{ss'}, \gamma \rangle$, where $\mathcal{S}$ is the discrete state space, $\mathcal{A}$ is the discrete set of actions, $\mathcal{P}^a_{ss'}$ is the state transition model that denotes the probability of transitioning to state $s'$ when action $a$ is applied at state $s$. $\mathcal{R}^a_{ss'}$ is a known reward model representing the reward for executing action $a$ in state $s$ and transitioning to state $s'$. $\gamma \in [0, 1]$ is the discount factor used to balance relative weights of current and future rewards. Only MDPs with discrete and finite state space are considered. A *policy* is a mapping $\pi : \mathcal{S} \to \mathcal{A}$ from state to action space. Together with the initial state $s_0$, a policy forms a trajectory $z = \{s_0, a_0, r_0, s_1, a_1, r_1, \cdots\}$. For each time-step $t$, $a_t = \pi(s_t)$, and both $r_t$ and $s_{t+1}$ are sampled from the reward and transition models correspondingly. The *value* of each state-action pair under policy $\pi$ is defined as:

$$Q^{\pi}(s,a) = E_{\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t \middle| s_0 = s, a_0 = a\right], \tag{1}$$

which is the expected sum of discounted rewards obtained starting from state $s$, taking action $a$ and following the policy $\pi$ thereafter. The optimal policy $\pi^*$ is given by the solution of the following optimization problem:

$$\pi^*(s) = \operatorname*{argmax}_a Q^{\pi^*}(s,a). \tag{2}$$

The optimal solution $Q^*$ satisfies the Bellman equation

$$Q^*(s,a) = \mathop{E}_{s'}\left[\mathcal{R}^a_{ss'} + \max_{a'}\gamma Q^*(s',a')\right]. \tag{3}$$

## 3.1 MDPs with Parametric Uncertainties

An MDP with parametric uncertainty is defined by the tuple, $\mathcal{M}_p = \langle\mathcal{S},\mathcal{A},\mathcal{P}^a_{ss'}(p),\mathcal{R}^a_{ss'},p,\gamma\rangle$, where $p$ is an unknown mapping of the form $\mathcal{S}\to[0,1]$ and $\mathcal{P}^a_{ss'}(p)$ is the transition model as an explicit function of the unknown mapping $p$. The rest of elements are identical to the earlier definition of the MDP. If the mapping $p$ is constant over all states, it corresponds to the *MDP with an unknown parameter* framework, which has been extensively studied in [5]. In a more general setting, $p$ does not necessarily map each state to the same probability value, resulting in the *MDP with state-correlated uncertainty* framework. Solving such MDPs is the central theme of this paper.

From a practical point of view, $p$ usually denotes occurrence probability of some event $E$. Here an event $E$ refers to a set of state transitions $\langle s,a,s'\rangle, s'\sim\mathcal{P}^a_s$, that satisfy a certain condition. For instance, in the context of the robot navigation problem, an event may be defined as all state transitions where GPS signal was not received. Since the probability of the event occurrence depends on the state (*i.e.,* robot's location), then the problem needs to be formulated as an MDP with a state correlated uncertainty. In this setting, $p(s)$ can be written as the following set:

$$p(s) = P(\langle s,a,s'\rangle \in E|s). \tag{4}$$

For brevity, we only consider a single event that is action independent. For example in the robot navigation problem, the event is the GPS signal failure and it is assumed to be independent of the immediate action taken by the robot. Consequently we have removed the dependence of $p$ on $E$ and $a$ in Eq 4. The extension is straight forward.

## 3.2 The Big Picture of Our Approach

Note that if $p$ is known, then MDP with parametric uncertainty reduces to description of a regular MDP. Hence the underlying hypothesis of our approach is that the optimal planning problem for an MDP with parametric uncertainty can be solved by first estimating the mapping $p$ and then solving the corresponding MDP. This is equivalent to estimating the structure of the mapping $p$. However the structure of many state dependencies is usually not known beforehand, as some states may be contributing significantly to the mapping $p(s)$, while others may be completely independent of it. Hence our problem definition is as follows: given an MDP with state-correlated uncertainty, develop an iterative estimation/planning scheme where parameters are estimated from the environment and a good policy is obtained with small number of total interactions with the model and the environment.

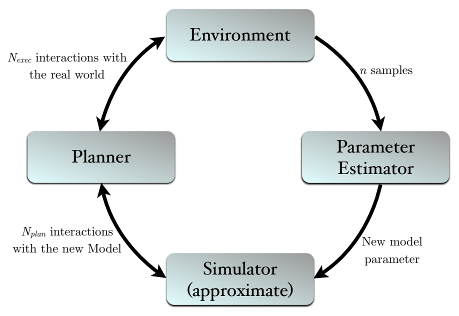


**Fig. 1.** Components of the adaptive planning framework

A general layout for this approach is given in Figure 1, for which one iteration of the algorithm is as follows. At the $k^{th}$ iteration, the simulator model has an estimate $\hat{p}^k$ of the the parameter. The planner interacts with the simulator for $N_{plan}$ steps in order to design a policy $\pi^k$. This policy is executed in the actual environment for $N_{exec}$ steps, where it is expected that $N_{exec} \ll N_{plan}$ because collecting samples is much more expensive from the real world compared to the simulation. The resulting trajectory $z^k$, which is of length $N_{exec}$, is used by the parameter estimator to obtain the new estimate $\hat{p}^{k+1}$ with which the simulator model is updated. In the next iteration, the planner computes an improved policy $\pi^{k+1}$, and the loop continues. Based on this discussion, when the model class and policy class include the true model and the optimal policy then the optimal adaptive planning criteria can be given as: $\lim\limits_{k\to\infty} \hat{p}^k = p$, $\lim\limits_{k\to\infty} \pi^k = \pi^*$, meaning that the estimate converges to its true value, while the policy converges to the optimal policy. However most planning and estimation algorithms have only asymptotic convergence, requiring $N_{plan}, N_{exec} \to \infty$ to satisfy this criteria.

## 4 Estimation and Planning Methods

### 4.1 Updating The Approximate Uncertainty Representation

An approximate representation of the uncertainty can be formed using linear function approximation with binary features [8] as follows

$$p(s) \approx \hat{p}^k(s) = \phi^\top(s)\theta^k, \tag{5}$$

where $\hat{p}^k\left(s\right)$ is the approximate representation at $k^{th}$ step and $\phi\left(s\right)$ is the vector of features [1]. Each component $\phi_j$ is a binary feature characterized by a mapping from state to a binary value; $\phi_j\left(s\right): s \to \{0,1\}, j = 1,...,m$, where $m$ is the total number of features and $\theta^k \in \mathbb{R}^m$ is the weight vector at step $k$. A feature $\phi_j$ is called *active* at state $s$ if $\phi_j(s) = 1$. Set of active features at state $s$ is given by $\mathsf{A}\left(s\right) = \{j | \phi_j\left(s\right) = 1\}$. Hence, Eq. 5 can be written as:

$$\hat{p}^k\left(s\right) = \sum_{j \in \mathsf{A}(s)} \theta_j^k,$$

where $\theta_j^k$ denotes the $j^{th}$ component of $\theta^k$.

New estimates are formed by updating the weight vector at each step. For that purpose, define a Bernoulli random variable $\Psi\left(s\right)$ with parameter $p(s)$. That is, $\Psi(s)$ is equal to $1$ with probability $p(s)$ and zero otherwise. Let $z^k$ be the $k^{th}$ experienced trajectory with length $N_{exec}$, obtained from interacting with the environment. Define $s^{k,l}$ as the state at the $l^{th}$ time-step of the $k^{th}$ trajectory where $l = 1,...,N_{exec}$. Let $\theta^{k,l}$ denote the corresponding weight vector. Define $\zeta(s^{k,l})$ to be the value that random variable $\Psi(s^{k,l})$ takes. This value can be gathered from $z^k$ based on the occurrence of the event that is defined in Eq. 4. The weight vector can be updated applying a gradient descend type update as follows

$$\begin{aligned}\theta^{k,l+1} &= \theta^{k,l} - \frac{1}{2}\alpha^{k,l}\frac{\partial}{\partial\theta^{k,l}}[p(s^{k,l}) - \hat{p}^{k,l}(s^{k,l})]^2 \\ &= \theta^{k,l} + \alpha^{k,l}[p(s^{k,l}) - \hat{p}^{k,l}(s^{k,l})]\phi(s^{k,l}),\end{aligned}$$

where $\alpha^{k,l}$ is a scalar step-size parameter. Since $p$ is unavailable to the algorithm, it is replaced by its estimate $\zeta(s^{k,l})$. Define the sampled estimation error at state $s$ as $\Delta p^{k,l}(s) = \zeta(s) - \hat{p}^k(s) = \zeta(s) - \phi(s)^T\theta^{k,l}$. Then, the final form of the parameter update law is

$$\theta^{k,l+1} = \theta^{k,l} + \alpha^{k,l}\Delta p^{k,l}(s^{k,l})\phi(s^{k,l}). \tag{6}$$

Eq. 6 is a variant of the well studied stochastic gradient descent (SGD) algorithm [18]. Since the structure of $p$ is not known beforehand, quality of the resulting approximation found by SGD depends strongly on the selected set of features.

## 4.2 Adaptive Function Approximation using iFDD

Adaptive function approximators also modify the set of features based on the observed data based on the following general update rule:

$$\begin{aligned}\hat{p}^{k,l}(s) &= \phi^{k,l}(s)^{\top}\theta^{k,l}, \\ \phi^{k+1,l+1}(s) &= h(z^k, \theta^{k,l}, \phi^{k,l}),\end{aligned} \tag{7}$$

where $h$ is the representation expansion function that adds new features to the feature vector based on sampled trajectories, weight vector, and previous set of features. Based

[1] Our methodology can be extended to state-action correlated uncertainties by introducing feature vectors $\phi(s,a)$.

on the successful results in representing high dimensional value functions, we employ iFDD [13, 12] as the adaptive function approximator for our framework to represent the uncertainty. The basic idea of iFDD is to expand the representation by adding conjunctions of the initial features based on an error metric, thus reducing the error in parts of the state space where the feedback error persists. The general outline of the algorithm is as follows: given a set of initial binary features, when performing the update for state $s \in z^k$, a conjunction set $\phi^c(s) = \{\phi_j(s) \wedge \phi_k(s) | j, k \in \mathsf{A}(s)\}$ is formed. These features are referred as candidate features. If the sum of sampled estimation error $\Delta p(s)$ over active candidate features exceeds some pre-determined threshold $\xi$, these conjunctions are added to set of features. The evaluation function learner (ELF) algorithm [32], expands the representation akin to iFDD that we use, yet candidate features are selected based on a limited set of heuristically selected features.

## 4.3 Planning

The goal of the planning algorithm is to find a value function which satisfies the Bellman equation (*i.e.,* Eq. 3) as closely as possible. Since in our approach an approximate model is always present, we focus our attention on DP types of planners. A particular property of interest is the sample efficiency of the algorithm, meaning that a policy with reasonable expected cumulated reward is obtained with small number of interactions with the model (*i.e.,* $N_{plan}$). For this purpose, we compare two DP approaches: 1) classical Value Iteration (VI) [30] and 2) Trajectory Based Value Iteration (TBVI) [30], which can be viewed as an specific instance of Real Time Dynamic Programming (RTDP)[1].

**Value Iteration** VI is a classic DP algorithm that updates state-action values by sweeping through the whole space, applying the *Bellman update*

$$Q(s,a) = \sum_{s' \in \mathcal{S}} \mathcal{P}^a_{ss'} \left[\mathcal{R}^a_{ss'} + \gamma \mathrm{max}_{a'} Q(s',a')\right], \tag{8}$$

until no significant change is observed.[2] In our framework the number of state updates are limited by $N_{plan}$. When $N_{plan} \ll |\mathcal{S}|$, VI may not find reasonable approximation to the value function as Bellman updates may be applied to states with trivial change to the value function. *Prioritized Sweeping* [20] addresses this problem, by applying the Bellman update to the state with the highest predicted value change. While efficient for performing exact DP, when the number of updates is fixed, this method may also focus updates in regions of the state space where the Bellman error is high, yet not frequently visited.

**Trajectory Based Value Iteration** Motivated by on-policy RL methods, Trajectory Based Value Iteration (TBVI), focuses the Bellman updates in states that are sampled through Monte-Carlo simulations. The policy used for generating trajectories are $\epsilon$-greedy with respect to the current value function:

$$\pi^\epsilon(s,a) = \begin{cases} 1-\epsilon & a = \mathrm{argmax}_a Q(s,a) \\ \frac{\epsilon}{|\mathcal{A}|} & \text{otherwise} \end{cases}. \tag{9}$$

[2] This formula represents the asynchronous VI [3], as new estimates are used instantaneously for future estimates.

The $\epsilon$ parameter assures that in the limit all state-action pair are updated infinitely, guaranteeing the convergence to the optimal value function. Notice that both TBVI and VI apply the Bellman update (*i.e.,* Eq. 8) to $N_{plan}$ state-action pairs. Their difference lies on their choice for selecting state-action pairs for the update. TBVI focuses the Bellman updates in regions of the state space that are deemed important based on the current policy. We will investigate the effectiveness of TBVI against VI in Section 6.4.

## 5 Theoretical Analysis

This section investigates the asymptotic properties of iFDD combined with the SGD algorithm presented in the previous section (iFDD-SGD). For the analysis, we consider the estimation part, assuming that the iteration number $k$ is fixed and that each state $s^{k,l}$ and its corresponding random variable $\Psi(s^{k,l})$ is sampled by following an exploratory policy that turns the underlying MDP into an ergodic Markov chain.[3] For brevity, superscript $k$ will be dropped from notation since it is fixed. We provide a theoretical proof showing that iFDD-SGD asymptotically converges to a solution with probability one using existing results on stochastic gradient descent theory [28, 18]. Moreover, we show that if $p$ can be captured through the representation class, iFDD-SGD converges to this point. Throughout the section we assume the standard diminishing step-size parameter for Eq. 6.

### 5.1 Preliminaries

Define $\mathcal{V}$ as space of all functions of the form $v : \mathcal{S} \to \mathbb{R}$. Define tabular representation as a set of features of the form $\phi_j(s_i) = \delta_{ij}$, where $\delta_{ij}$ is the Kronecker delta function and $s_i \in \mathcal{S}, i = 1, 2, ..., |\mathcal{S}|$. Note that tabular representation forms a basis for this space, since any $v \in \mathcal{V}$ can be represented as $v = \sum_{j=1,2,\ldots,|\mathcal{S}|} v\,(s_j)\,\phi_j$. It can be easily shown that $\phi_j$ are orthogonal to each other, hence the $\dim(\mathcal{V}) = |\mathcal{S}|$. Let $f = \{\phi_j \in \mathcal{V}, j = 1, \cdots, n\}$ be a set of $n$ linearly independent features. Define $\Phi_f \in \mathbb{R}^{|\mathcal{S}|\times n}$ to be the feature matrix with elements $\Phi_{(i,j)} = \phi_j(s_i), i = 1, 2, ..., |\mathcal{S}|, j = 1, 2, ..., n$. It can be shown that $\mathtt{span}(\Phi_f)$ is a subspace of $\mathcal{V}$ [12], hence the orthogonal projection operator $\Pi^{\mathcal{F}} : \mathcal{V} \to \mathcal{F}$ is well defined, where $\mathcal{F}$ is the subspace $\mathtt{span}(\Phi_f)$. The weight matrix used for the projection operator is a diagonal matrix with the steady state distribution of states as its non-zero elements. More details about the matrix description of the projection operator can be found in [12]. Moreover, define $\Omega(f)$ as the set of all possible conjunctions of the elements of $\phi_j \in f$ and let $\Omega(\mathcal{F})$ be the corresponding subspace. Define $C(s)$ as the total number of non-zero features at state $s$. Finally define $D_{\phi_i}$ as the set of features constituting feature $\phi_i$. For instance if $\phi_i = \phi_j \wedge \phi_k \wedge \phi_l$ then $D_{\phi_i} = \{\phi_j, \phi_k, \phi_l\}$. If $\phi_i$ belongs to set of initial features then, $D_{\phi_i} = \{\phi_i\}$.

### 5.2 Convergence of iFDD-SGD

**Lemma 1.** *Under the ergodicity and diminishing step-size assumptions, using SGD with fixed feature representation $f$, $\hat{p}_l$ defined in Eq. 7 converges to $\Pi^{\mathcal{F}} p$ as $l \to \infty$, with probability one.*

[3] Here we are restricting ourselves to the class of MDPs that can be turned into an ergodic Markov Chain.

*Proof.* Ergodicity assumption simply turns the problem into a least-squares parameter estimation problem for $p$, with infinite amount of data. With the diminishing step-size assumption, it is sufficient to invoke the results of Thm 5.3.1 in [18] showing that stochastic approximation algorithm SGD will converge to least-squares solution asymptotically. That is $\hat{p}_l = \Pi^{\mathcal{F}} p$ as $l \to \infty$. □

The following lemma states that, when a new feature, which is constructed by taking conjunctions of existing features is added to the feature set, it will only be activated at the parts of the state space with more than one active feature. This conclusion will simplify the development of the convergence theorem.

**Lemma 2.** *Let $g \in \Omega(f)$ be added to the set of existing features by iFDD. Then $\forall s \in \mathcal{S}$ where $C(s) \leq 1$ before adding g, then $\phi_g(s) = 0$.*

*Proof.* If $C(s) = 0$, proof is trivial since no feature is active at state $s$ including $\phi_g$. Let $C(s) = 1$, and let $\phi_j$ be the corresponding active feature. if $D_{\phi_g} \subset D_{\phi_j}$, then $\phi_g(s) = 0$ due to sparse activation property of iFDD explained in subsection 4.2. Assume that $D_{\phi_g} \not\subset D_{\phi_j}$, then there exists a $\phi_i \in f$ such that $\phi_i \in D_{\phi_g}$ and $\phi_i \notin D_{\phi_j}$. Since only $\phi_j$ is active at $s$, $\phi_i(s) = 0$, which in turn implies that $\phi_g(s) = 0$. □

**Theorem 1.** *Under the ergodicity and diminishing step-size assumptions, $\hat{p}^l$, using SGD (Eq. 6) with iFDD representation with an initial set of features $f$ and discovery threshold $\xi > 0$, converges to $\Pi^{\Omega(f)} p$ as $l \to \infty$ with probability one.*

*Proof.* Since the number of conjunctions are finite, there exists a point at time, after which the set of features is unchanged. Let us call this terminal fixed set of features $f^\dagger$ and the corresponding subspace $\mathcal{F}^\dagger$. We show that the claim holds for all possible $\mathcal{F}^\dagger$:

- $(\Pi^{\mathcal{F}^\dagger} p = p)$: This means the representation is rich enough to capture the exact $p$ vector. Lemma 1 shows that in this case $\hat{p}^l$ converges to $\Pi^{\mathcal{F}} p$ as $l \to \infty$, with probability one.
- $(\Pi^{\mathcal{F}^\dagger} p \neq p)$: This means $p \notin \mathcal{F}^\dagger$. Define $\mathcal{S}^- \subseteq \mathcal{S}$ as the set of states for which $\Delta p(s) = p(s) - \hat{p}(s) \neq 0$. We argue that $\forall s \in \mathcal{S}^-, C(s) \leq 1$. Assume this is not true and let $e(s)$ denote the cumulative sampled estimation error at state $s$. Due to ergodicity of the underlying Markov chain, state $s$ will be visited infinitely many times, hence after some time $e(s)$ will exceed any pre-defined threshold $\xi$, and since $C(s) > 1$ iFDD algorithm would expand $f^\dagger$ with the active features at state $s$. This contradicts that $f^\dagger$ is the terminal fixed representation.
  Now, it is sufficient to show that $\Pi^{\mathcal{F}^\dagger} p = \Pi^{\Omega(\mathcal{F})} p$. We prove this part by showing that the projection of the asymptotic residual (*i.e.,* $\delta p = p - \Pi^{\mathcal{F}^\dagger} p$), on any unexpanded feature vector (*i.e.,* $\phi_q \in \Omega(f) \setminus f^\dagger$) is null. To do so, it is sufficient to show that $\delta p^\top \phi_q = \sum_{s \in \mathcal{S}} \delta p(s) \phi_q(s) = 0$. Since $\forall s \notin \mathcal{S}^- \Rightarrow \delta p = 0$, we can write the summation as: $\sum_{s \in \mathcal{S}^-} \delta p(s) \phi_q(s)$. On the other hand, Lemma 2 showed that $\forall \phi_q \in \Omega(\mathcal{F}) \setminus \mathcal{F}^\dagger, \forall s \in \mathcal{S}^-$, if feature $q$ is added to the representation, then $\phi_q(s) = 0$. Hence $\forall \phi_q \in \Omega(f) \setminus f^\dagger, \delta p^\top \phi_q = 0$. Therefore adding any of the remaining features to $f^\dagger$ does not help the representation to reduce the residual error further down. So as $l \to \infty, \hat{p} \to \Pi^{\mathcal{F}^\dagger} p = \Pi^{\Omega(\mathcal{F})} p$. □

Theorem 1 proves that given an initial set of features $f$, iFDD-SGD asymptotically converges to the best possible approximation with probability one. The analysis also provides guidance on how to select the set of initial features. If $f$ is chosen such that $\dim(\Omega(\mathcal{F})) \geq |\mathcal{S}|$, iFDD-SGD's approximation will be exact asymptotically with probability one. For instance, consider the following process for selecting an initial set of features for an MDP with finite state space. Let $s$ be represented by a $d$ dimensional vector, where $s_i$ corresponds to the $i^{th}$ component. Hence $s = (s_1, s_2, \cdots, s_d)$. Let $\{v_i^1, v_i^2, \cdots, v_i^{n_i}\}$ denote the distinct values that $s_i$ can take. Then initial features can be selected selected as follows:

$$\begin{aligned} & f = \left[\phi_{11} \dots \phi_{1n_1}\ \phi_{21} \dots \phi_{2n_2} \dots \phi_{dn_d}\right]^\top, \\ & \phi_{ij}(s) = \begin{cases} 1 & s_i = v_i^j \\ 0 & \text{otherwise} \end{cases}, i = 1, \dots, d, j = 1, \dots, n_i, \end{aligned} \tag{10}$$

amounting to a total of $m = \sum_{i=1}^{d} n_i$ features. Geramifard et al. [12] demonstrated that, for such an initialization, $\Omega(f)$ satisfies $\dim(\Omega(\mathcal{F})) \geq |\mathcal{S}|$.

### 5.3 Time Complexity

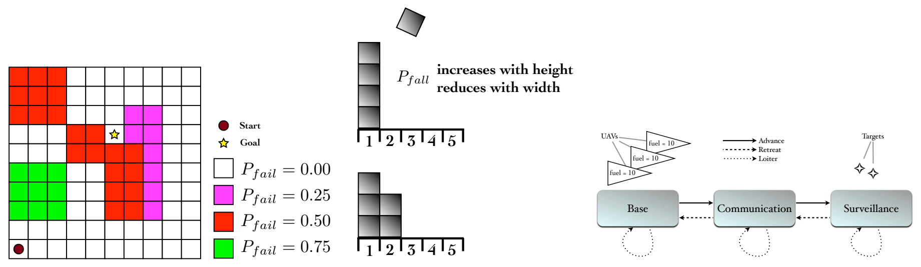
The per-time-step complexity of the iFDD algorithm has been previously investigated in [12]. The study showed that, at each step given $n$ features with maximum $\kappa$ number of non-zero features, the total complexity of the feature expansion and evaluation is $\mathcal{O}(\kappa 2^\kappa)$ which is independent of the total number of features $n$. This property is highly desirable since $n$ may grow to large numbers due to feature expansion, but in turn $\kappa$ gets smaller due to the sparsity property of iFDD [12].

## 6 Simulation Study

The planning and estimation algorithms described in Section 4 are investigated in three distinct domains. The first two domains are classical RL problems: 1) gridworld and 2) block building problems motivated by [26]. Both domain descriptions are extended to include state correlated uncertainty. The third domain is a PST mission with state-correlated uncertainty in fuel dynamics. Structure of this domain is more complex and is included to validate our approach on a large-scale stochastic UAV mission planning problem. In all domains, $\gamma$ was set to $0.9$ and the threshold for iFDD ($\xi$) was set to $1$.

### 6.1 Gridworld Navigation with Sensor Management

This domain consists of a robot navigating in a $10 \times 10$ gridworld shown in Figure 2(a). The task for the robot is to reach the goal ($\star$) from the starting position ($\bullet$). Possible actions are $\{\leftarrow, \rightarrow, \uparrow, \downarrow\}$. There is $20\%$ chance that the robot moves to one of the adjacent grids that was not intended. Reward is $+10$ for reaching the goal and zero for all movement actions. In addition, the robot carries two sensors for navigation: a camera and a GPS. The GPS is the preferred tool for navigation with no additional cost, although the probability of receiving the GPS signal is location dependent shown in Figure 2(a) as



(a) Gridworld (b) Block Building (c) Persistent Search and Track

**Fig. 2.** Three domains used to generate empirical results. Refer to the text for the description of each domain.

$p_{\text{fail}}$ highlighted with four colors. If the GPS signal is not received, the robot uses the camera, incurring an additional $-1$ reward, while receiving the exact position. The goal of the adaptive planner is to estimate the structure of the $p_{\text{fail}}$ through interactions and modify the plan accordingly. The size of the state-action space of the domain is about $800$.

## 6.2 Block Building

In this domain, the objective is to distribute $5$ blocks to $5$ empty slots on a table shown in Figure 2(b). Initially all blocks are located under the table. The set of possible actions is defined by picking any of the blocks on or under the table and put it in any of the $5$ slots. The episode is finished when there is no blocks under the table. The reward at the end of an episode is equal to the hight of the tallest tower minus one. However, placing a block on the top of the others involves $p_{\text{fall}}$ probability of dropping the block under the table, which is increased by the size of the tower (*i.e.,* the longer the tower is, the harder is to place another block on top of it) and decreased by the number of blocks in adjacent slots. Let $n_{slot}$ be the number of blocks in the destination slot, and $n_{adj}$ be the maximum number of blocks in adjacent towers. Define $\bar{p}_{\text{fall}} = 0.1 \times (n_{slot} - n_{adj})^2 + 0.1$. Then $p_{\text{fall}}$ is calculated as:

$$p_{\text{fall}} = \begin{cases} 0 & \bar{p}_{\text{fall}} \leqslant 0 \text{ or } n_{slot} = 0 \\ \bar{p}_{\text{fall}} & 0 < \bar{p}_{\text{fall}} < 1 \\ 1 & \bar{p}_{\text{fall}} \geqslant 1 \end{cases} .$$

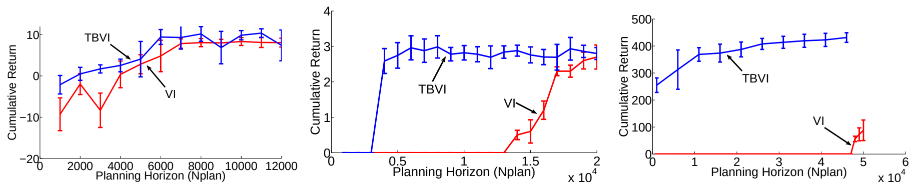
The optimal solution is shown in the bottom of Figure 2(b). The state-action size for this domain is approximately $15 \times 10^3$.

## 6.3 Persistent Search and Track Mission

The persistent search and track is a multi-agent UAV mission planning problem, where a group of UAVs perform surveillance on a group of targets, while maintaining communication and health constraints [6]. Outline of the mission is displayed in Figure 2(c). It should be emphasized that, although this is a multi-agent domain, decision making

**Table 1.** Probability of nominal fuel burn for UAVs across different locations and health status.

| | Health Status | | |
|---|---|---|---|
| Location | No Failure | Sensor Failed | Actuator Failed |
| Base | 1.0 | 1.0 | 1.0 |
| Communication | 0.95 | 0.92 | 0.86 |
| Surveillance | 0.88 | 0.80 | 0.75 |



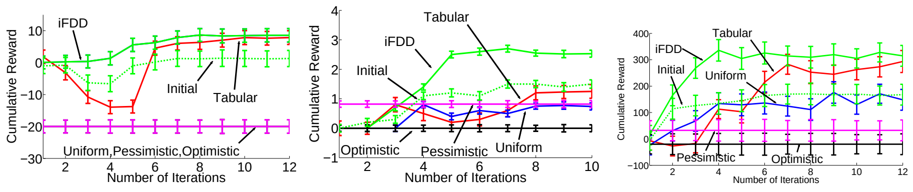
(a) Gridworld (b) Block Building (c) Persistent Search and Track

**Fig. 3.** Comparison of VI and TBVI across experiment domains using their exact model of the domain. The X-axis is the number of Bellman update, while the Y-axis represents the performance of the resulting policy.

process is completely centralized and the state-action space consists of combination of all possible states-action pairs of each UAV. Each UAV's individual state is given by its location, fuel, and health. The health is defined by two bits indicating the functionality of the sensor and the actruator. There are three available actions for each UAV: $\{$Advance, Retreat, Loiter$\}$. The objective of the mission is to travel to the surveillance node and put surveillance on two targets, while one UAV stays at communication to provide the data link with the base. Each UAV starts with 10 units of fuel and burns one unit for all actions with probability $p_{\text{nom}}$ and 2 units with probability $1 - p_{\text{nom}}$. Since UAVs are subject to more aggressive maneuvers in the surveillance area, $p_{\text{nom}}$ should decrease in that region. Similarly when a UAVs health is degraded, maneuverability and fuel consumption degrade accordingly. Therefore $p_{\text{nom}}$ is a state correlated uncertainty shown in the Table 1. If a UAV runs out of fuel, it crashes and can no longer continue the mission. UAVs are also subject to sensor and actuator failures at each transition step with probability $p_{\text{fail}} \in [0, 1]$. A UAV with failed sensor cannot perform surveillance whereas a UAV with failed actuator cannot perform neither surveillance nor communication. When a UAV returns to the base, it is refueled and its failures are repaired. Hence the objective of the planner is to maximize the number of time steps that two UAVs with working sensors are located in the surveillance region and having one UAV with a working actuator in the communication region. The complete MDP description of the mission can be found in [25]. This domain has approximately $19 \times 10^6$ state-action pairs.

### 6.4 Results

First, we evaluated VI and TBVI across all the three domains using their exact models. The exploration schedule ($\epsilon$) for the TBVI followed the form: $\epsilon^k = \frac{0.9}{(\text{Episode number})^{\epsilon_{\text{decay}}}} + 0.1$, where $k$ is the planning step and $\epsilon_{\text{decay}}$ was empirically selected out of $\{0.01, 0.1, 0.5, 1.0\}$. Figure 3 depicts the results of running TBVI and VI in all three domains. The X-axis represents the number of Bellman updates, and the Y-axis corresponds to



(a) Gridworld (b) Block Building (c) Persistent Search and Track

**Fig. 4.** Comparison of five estimation methods combined with TBVI across across experiment domains. The X-axis is the number of iterations of the process shown in Figure 1, while the Y-axis represents the performance of the resulting policies.

the performance of the resulting greedy policy with respect to the estimated value function. Each point shows the mean and the standard error representing 95% confidence interval based on 30 runs. Overall, our empirical results coincide with the earlier experiments [30], showing the sample efficiency of TBVI against VI. Notice as the size of the state space grew, the sample complexity reduction of TBVI over VI became more evident. In particular, in the largest domain, VI with 5 times more data compared to TBVI, achieved less than half of the TBVI's performance. This led us to pick TBVI as our planner.

Secondly, we tested the performance of model estimation methods with fixed $N_{plan}$ and $N_{exec}$ combined with TBVI. $N_{plan}$ was selected based on the performance of the planners in Figure 3, while $N_{exec}$ was chosen based on the domain properties and to be a substantially smaller number than $N_{plan}$. Figure 4 plots the results. The X-axis is the number of iterations. Each iteration is a complete cycle of the process displayed in Figure 1. Each algorithm was executed 30 times and for each execution, after completion of each iteration the resulting policy was evaluated on the actual domain over 30 runs amounting to 900 samples per data point. The mean of the cumulated reward and standard deviation is plotted accordingly on the Y-axis.

Five different representation was used to compare various model estimation methods. In order to emphasize the value of adaptation, fixed model estimators were also included in the evaluation. *Fixed Model Optimistic* and *Fixed Model Pessimistic* approaches assumed that the unknown parameter is fixed to 0 and 0.6 correspondingly across all states and did not update this value. *Uniform* representation ignored the state-correlation in the problem by utilizing a single fixed feature which was active at all times and was adapted based on observations. *Tabular* representation stored a distinct feature for each state, resulting in a number of of parameters equivalent to the size of the state space. Initial features for iFDD were selected using Eq. 10. To emphasize the value of discovering new features, results with the fixed initial features were also included in the plots.

**Gridworld** For adaptive planning experiment, planning and execution horizons were set to $N_{plan} = 8000$ and $N_{exec} = 100$. Performance of various estimation schemes using TBVI planner are displayed in Figure 4(a). In this case uniform estimation converged to the same policy obtained by any fixed model planners. This is due to fact that with a uniform uncertainty the optimal policy is to move directly towards the goal. This explains the poor performance of optimistic, pessimistic, and uniform estimators. Both tabular and iFDD estimators had the capability to capture the $p_{\text{fail}}$ accurately. We conjecture that the dip on the performance of the tabular estimator is due to policy switching. Initially the agent followed the shortest route to the goal. As more samples were obtained, the agent explored the safe route from the right side, yet it required many samples to master the new route. The iFDD estimator performed substantially better early on compared to tabular estimator due to generalization of the features mentioned in section 4.2. In this experiment iFDD started with 22 features and expanded on average a total of 150 features. iFDD representation also performed better than representation that uses only initial features, which emphasizes the importance of expanding the representation.

**Block Building** In this domain $N_{plan} = 6000$ and $N_{exec} = 30$. Trajectories were capped at 200 steps. Results are given in Figure 4(b). The optimistic model achieved $0$ performance, because it assumed that $p_{\text{fall}} = 0$. Hence it kept trying to build a single tower which resulted in frequent collapses. The pessimistic approach placed $4$ blocks into $4$ slots and placed the final block on one of the placed blocks, achieving performance of $1$. Uniform estimation eventually converged to the same policy as the pessimistic model. Both Tabular and iFDD estimators had the capability to capture the model exactly. While tabular estimator outperforms previous methods, iFDD estimator learned the task substantially faster and reached very close to the optimal policy shown in bottom part of Figure 2(b), using on average $\sim 10^3$ features.

**Persistent Search and Track** In this domain $N_{plan} = 10^5$ and $N_{exec} = 1000$. Results are shown in Figure 4(c). Both fixed model approaches perform poorly. The optimistic model assumed no uncertainty in the fuel dynamics, which resulted in an overly aggressive policy with frequent crashes. The pessimistic model assumed high uncertainty in the fuel dynamics, which resulted in a conservative policy that called UAVs back to the base early, resulting in a poor performance. Even though uniform estimation outperformed both fixed estimators, its performance did not improve after a number of trajectories due to its inability to capture the state-correleated uncertainty. Representation with initial features outperformed uniform, optimistic and pessimistic approaches, yet was not competitive. A similar trend to results presented before was observed between Tabular and iFDD estimators – the iFDD estimator settled around the best found accumulated reward among all methods much faster then the tabular estimator due to its capability to represent the uncertainty with fewer parameters. In particular, iFDD was $\sim 2.5$ times more sample efficient than tabular estimation according to Figure 4(c). In this experiment, iFDD expanded a total of $\sim 10^4$ features on average. The size of the parameter for the tabular representation was equivalent to the size of the state space which was larger by about 70 times.

## 7 CONCLUSION

The problem of planning for MDPs with unknown state-correlated uncertainties was considered. Incremental feature dependency discovery (iFDD) was employed as a compact estimator of the state correlated uncertainty together with trajectory based value iteration (TBVI) as the planner. We proved that with a fixed policy and any set of initial features our iFDD-SGD approach will converge to the best approximated model with probability one. In particular, when the true model lies in the space of the fully expanded features, the approximation becomes exact asymptotically. The performance of the resulting algorithm was evaluated over three domains and compared against five uncertainty representations. Numerical experiment results highlighted a statistically significant improvement in terms of sample complexity and performance.

## References

[1] Satinder P. Singh Andrew G. Barto, Steven J. Bradtke. Learning to act using realtimedynamicprogramming. *Artificial Intelligience*, 72(1-2):81–138, 1995.

[2] John Asmuth, Lihong Li, Michael Littman, Ali Nouri, and David Wingate. A Bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Proceedings of the Twenty-Fifth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-09)*, pages 19–26, Corvallis, Oregon, 2009. AUAI Press.

[3] D. P. Bertsekas. *Dynamic Programming and Optimal Control, Vol. I-II, 3rd Ed.* Athena Scientific, Belmont, MA, 2007.

[4] L. F. Bertuccelli and J. P. How. Robust Markov decision processes using sigma point sampling. In *American Control Conference (ACC)*, pages 5003–5008, 11-13 June 2008.

[5] Luca F. Bertuccelli. *Robust Decision-Making with Model Uncertainty in Aerospace Systems*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge MA, September 2008.

[6] B. Bethke, J. P. How, and J. Vian. Multi-UAV Persistent Surveillance With Communication Constraints and Health Management. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, August 2009. (AIAA-2009-5654).

[7] Ronen I. Brafman and Moshe Tennenholtz. R-MAX - a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 3:213–231, 2001.

[8] Lucian Busoniu, Robert Babuska, Bart De Schutter, and Damien Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, 2010.

[9] E. Delage and S. Mannor. Percentile Optimization for Markov Decision Processes with Parameter Uncertainty. *subm to Operations Research*, 2007.

[10] Thierry Fraichard Dizan Vasquez and Christian Laugier. Incremental Learning of Statistical Motion Patterns With Growing Hidden Markov Models. *IEEE Transcations On Intelligent Transportation Systems*, 10(3), 2009.

[11] Emily B. Fox. *Bayesian Nonparametric Learning of Complex Dynamical Phenomena*. PhD thesis, Massachusetts Institute of Technology, Cambridge MA, December 2009.

[12] Alborz Geramifard. *Practical Reinforcement Learning Using Representation Learning and Safe Exploration for Large Scale Markov Decision Processes*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, February 2012.

[13] Alborz Geramifard, Finale Doshi, Joshua Redding, Nicholas Roy, and Jonathan How. Online discovery of feature dependencies. In Lise Getoor and Tobias Scheffer, editors, *International Conference on Machine Learning (ICML)*, pages 881–888, New York, NY, USA, June 2011. ACM.

[14] V. Gullapalli and A. Barto. Convergence of Indirect Adaptive Asynchronous Value Iteration Algorithms. *Neural Information Processing Systems (NIPS)*, 1994.

[15] G. Iyengar. Robust Dynamic Programming. *Math. Oper. Res.*, 30(2):257–280, 2005.

[16] V. Jilkov and X. Li. Online Bayesian Estimation of Transition Probabilities for Markovian Jump Systems. *IEEE Trans. on Signal Processing*, 52(6), 2004.

[17] J. Joseph, F. Doshi-Velez, A. S. Huang, and N. Roy. A Bayesian nonparametric approach to modeling motion patterns. *Autonomous Robots*, 31(4):383–400, 2011.

[18] Harold J. Kushner and G. George Yin. *Convergence of indirect adaptive asynchronous value iteration algorithms*. Springer, 2003.

[19] W. Liu, P.P Pokharel, and J.C. Principe. The kernel least-mean-square algorithm. *IEEE Transactions on Signal Processing*, 56(2):543–554, Feb 2008.

[20] A. W. Moore and C. G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.

[21] N. Nigam and I. Kroo. Persistent surveillance using multiple unmanned air vehicles. In *Aerospace Conference, 2008 IEEE*, pages 1 –14, march 2008.

[22] A. Nilim and L. El Ghaoui. Robust Solutions to Markov Decision Problems with Uncertain Transition Matrices. *Operations Research*, 53(5), 2005.

[23] M. L. Puterman. *Markov Decision Processes: Stochastic Dynamic Programming*. John Wiley and Sons, 1994.

[24] C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[25] J. D. Redding, T. Toksoz, N. Kemal Ure, A. Geramifard, J. P. How, M. Vavrina, and J. Vian. Persistent distributed multi-agent missions with automated battery management. In *AIAA Guidance, Navigation, and Control Conference (GNC)*, August 2011. (AIAA-2011-6480).

[26] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach, 2nd Edition*. Prentice-Hall, Englewood Cliffs, NJ, 2003.

[27] A. Ryan and J.K. Hedrick. A mode-switching path planner for uav-assisted search and rescue. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, pages 1471 – 1476, dec. 2005.

[28] A. Shapiro and Y. Wardi. Convergence Analysis of Gradient Descent Stochastic Algorithms. *Journal of Optimization Theory and Applications*, 91(2):439–454, 1996.

[29] R. S. Sutton, Cs Szepesvari, A. Geramifard, and M. Bowling. Dyna-style planning with linear function approximation and prioritized sweeping. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 528–536, 2008.

[30] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[31] T.C. Tozer and D. Grace. High-altitude platforms for wireless communications. *Electronics Communication Engineering Journal*, 13(3):127 –137, jun 2001.

[32] Paul E. Utgoff and Doina Precup. *Feature extraction, construction, and selection: A data-mining perspective*, chapter Constructive function approximation. 1998.

[33] Hengshuai Yao, Richard S. Sutton, Shalabh Bhatnagar, Diao Dongcui, and Csaba Szepesvári. Multi-step dyna planning for policy evaluation and control. In Yoshua Bengio, Dale Schuurmans, John D. Lafferty, Christopher K. I. Williams, and Aron Culotta, editors, *NIPS*, pages 2187–2195. Curran Associates, Inc., 2009.