Maximum Margin Planning

Nathan D. Ratliff NDR@RI.CMU.EDU
J. Andrew Bagnell DBAGNELL@RI.CMU.EDU
Robotics Institute, Carnegie Mellon University, Pittsburgh, PA. 15213 USA

Martin A. Zinkevich MAZ@CS.UALBERTA.CA
Department of Computing Science, University of Alberta, Edmonton, AB T6G 2E1, Canada

Abstract

Imitation learning of sequential, goal-directed behavior by standard supervised techniques is often difficult. We frame learning such behaviors as a maximum margin structured prediction problem over a space of policies. In this approach, we learn mappings from features to cost so an optimal policy in an MDP with these cost mimics the expert's behavior. Further, we demonstrate a simple, provably efficient approach to structured maximum margin learning, based on the subgradient method, that leverages existing fast algorithms for inference. Although the technique is general, it is particularly relevant in problems where A* and dynamic programming approaches make learning policies tractable in problems beyond the limitations of a QP formulation. We demonstrate our approach applied to route planning for outdoor mobile robots, where the behavior a designer wishes a planner to execute is often clear, while specifying cost functions that engender this behavior is a much more difficult task.

1. Introduction

In "imitation learning" a learner attempts to mimic an expert's behavior or control strategy. In numerous instances, notably within robotics (Pomerleau, 1989; LeCun et al., 2006), supervised learning approaches have been used to great effect to learn mappings from features to decisions. Long-range and goal-directed behavior, by contrast, has proven more difficult to capture using these techniques.

In mobile robotics, the motivating application of our imitation learning approaches, researchers often encourage long-horizon goal directed behavior by partitioning an autonomy software into a "perception" subsystem and a "planning" subsystem. The perception system computes various models and features of an environment. For instance, perception might determine the supporting surface, obstacles lying above this surface, average color at various locations, density of ladar returns, et cetera. Planning takes as input a cost-map over the vehicle configuration or state space (Hebert et al., 1998) and computes a minimal risk (cost) path through it that serves as a coherent sequence of decisions for the robot over a long horizon. This approach has proven powerful and lies at the heart of many autonomous mobile robotic systems in both indoor and outdoor environments.

Unfortunately, the leap from perception's model to costs for a planner is often a difficult one. In practice, it is often done by hand-designed heuristics that are painstakingly validated by observing the resulting robot behavior. In this work, we propose a novel method whereby we attempt to automate the mapping from perception features to costs. We do so by framing the problem as one of supervised learning to take advantage of examples given by an expert describing desired behavior. In essence, we leverage the fact that the desired behavior is often quite clear to a human designer while specifying costs that engender this behavior is a much more difficult task, particularly when it involves simultaneously tweaking a large set of knobs that map features to costs.

Learning to plan so as to mimic a teacher's behavior may be cast as a structured prediction problem over the space of policies. The learner's goal is to take example input features and example policies or trajectories through the state space (e.g. paths) and learn to predict the same sequence of decisions. Decisions at each state must be coordinated and their

Appearing in *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburgh, PA, 2006. Copyright 2006 by the author(s)/owner(s).

costs balanced to achieve a satisfactory global planning strategy that implements the desired behavior. In this approach, our goal is to learn mappings from features to cost functions so an optimal policy in a Markov Decision Problem with this cost function imitates the expert's behavior. We demonstrate that we can learn such mappings in the structured large margin framework. (Taskar et al., 2005)

The key contributions of this work are three-fold. First, we demonstrate a novel method for learning to plan. Second, we demonstrate an efficient, simple approach to structured maximum-margin classification (in both online and batch settings) that is applicable whenever a fast specialized algorithm is available to compute the minimum of the loss-augmented cost function. This approach demonstrates linear convergence when used in batch settings and is applicable to large problems where other Quadratic Programming techniques are not. We develop an online theory for Structured Maximum Margin and show that our algorithm achieves sublinear regret that scales inversely with the margin and without a dependence on the size of the policy we must learn. Finally, we demonstrate empirically that our method is particularly applicable to problems of relevance in mobile robotics. We outline our current research directions and other natural applications of Max Margin Planning in the conclusions (Section 5). We also discuss related approaches to imitation learning and structured classification in Section 5.

2. Preliminaries

We model the planning problem with discrete Markov Decision Processes. Let x and a index the state and action spaces $\mathcal{X}$ and $\mathcal{A}$, respectively; and let $p(y|x,a)$ and s denote, respectively, the transition probablities and initial state distribution. A discount factor on rewards (if any) is absorbed into the transition probabilities. Our reward functions[1] are learned from supervised examples to produce policies that mimic demonstrated behavior. We repeatedly make use of the linear programming formulation of the MDP problem (Puterman, 1994), denoting by $v \in \mathcal{V}$ the primal variables of the value function, and by $\mu \in \mathcal{G}$ the dual state-action frequency counts. We consider here only stationary policies; the generalization is straightforward.

The input to our algorithm is a set of training instances $\mathcal{D} = \{(\mathcal{X}_i, \mathcal{A}_i, p_i, F_i, y_i, \mathcal{L}_i)\}_{i=1}^n$. A training instance consists of an MDP with transition probabilities p_i, state-action pairs $(x_i, a_i) \in \mathcal{X}_i \times \mathcal{A}_i$ over which d-dimensional feature vectors are place in the form of a $d \times |\mathcal{X}||\mathcal{A}|$ feature matrix F_i. y_i denotes the desired trajectory (or full policy) that exemplifies behavior we hope to match. We often consider the alternate representation for such a trajectory in terms of μ_i, a vector of state-action frequency counts. Note that since we are encoding the policies in terms of the dual state-action frequency counts, this formulation can attempt to learn to match either trajectories or entire policies. This will be described in detail below. We also discuss the role of the loss vector l_i below.

We use subscripts to denote indexing by training instance, and reserve superscripts for indexing into vectors. (E.g. $\mu_i^{x,a}$ is the expected state-action frequency for state x and action a of example i.) Note that we will sometimes write $\mathcal{D} = \{(\mathcal{X}_i, \mathcal{A}_i, p_i, f_i, y_i, \mathcal{L}_i)\}_{i=1}^n \equiv \{(\mathcal{X}_i, \mathcal{A}_i, \mathcal{G}_i, F_i, \mu_i, l_i)\}_{i=1}^n$, using $f_i(y)$ to denote vector of expected feature counts $F_i\mu$ of the ith example. It is useful for some problems, such as robot path planning, to imagine representing the features as a set of maps and example paths through those maps. For instance, one feature map might indicate the elevation at each state, another the slope, and a third the presence of vegetation.

The learner attempts to find a linear mapping[2] of these features to rewards so that for each problem instance the best policy over the resulting reward function $\mu^* = \arg\max_{\mu \in \mathcal{G}_i} w^T F_i \mu$ is "close" to the demonstrated policy μ_i. The notion of closeness is defined by a loss-function $\mathcal{L} : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}_+$ between solutions. In our work this function is of the form $\mathcal{L}(y, y_i) = \mathcal{L}_i(y) = l_i^T \mu$. Intuitively, a loss vector $l_i \in \mathbb{R}_+^{|\mathcal{X}||\mathcal{A}|}$ is placed over state-action pairs that defines for each state-action pair how much the learner pays for failing to match the behavior of an example policy y_i as the cumulative loss of the learned policy through this MDP. (See Figure 4)

The maximum margin principle builds in a degree of robustness where we attempt to make the supplied solution y_i look significantly better than alternative policies. In particular, we adopt the structured max-margin framework and attempt to make y_i better than any other solution $\hat{y}$ by a *margin* that scales with the size of the loss of $\hat{y}$; we wish to ensure that the correct policy looks much more attractive than very bad policies.

[1] In some deciplines is it more common and suitable to describe problems in terms of costs rather than rewards. In what follows, the term "cost" is used interchangeably to mean "negative reward".

[2] Although space doesn't permit full details, both formulations we use for learning later straightfowardly generalize to handle nonlinearity through the use of kernelization.

2.1. Loss-functions

The framework thus described admits any loss function that factors over state-action pairs. A natural loss function in the case of deterministic acyclic path planning is the count of the number of states that the planner visits that the teacher did not. We have found somewhat better performance by smoothing this loss function so that nearby paths are also admissible. In a general MDP, we might penalize choosing different actions from the teacher at any states the teacher reaches or penalize reaching states the teacher chooses not to enter. We also assume that $\mathcal{L}(y, y_i) \geq 0$.

2.2. Quadratic Programming Formulation

Given a training set $\mathcal{D} = \{(\mathcal{X}_i, \mathcal{A}_i, p_i, f_i, y_i, \mathcal{L}_i)\}_{i=1}^n$, the structured large margin criteria (Taskar et al., 2005) implies we are solving the following quadratic program:[3]

$$\min_{w,\zeta_i} \frac{1}{2}\|w\|^2 + \frac{\gamma}{n}\sum_i \beta_i \zeta_i^q \tag{1}$$

$$s.t.\ \ \forall i \quad w^T f_i(y_i) + \zeta_i \geq \max_{y \in \mathcal{Y}_i} w^T f_i(y) + \mathcal{L}_i(y) \tag{2}$$

The intuition behind these constraints is that we allow only weight vectors for which the example policies have higher expected reward than all other policies by a margin that scales with the loss. The slack variables ζ_i permit violations of these constraints for a penalty that scales with the hyperparameter $\gamma \geq 0$. $\beta_i > 0$ are data dependent scalars that can be used for normalization when the examples are of different lengths. $q \in \{1, 2\}$ distinguishes between L_1 and L_2 slack penalties commonly found in the literature (Tsochantaridis et al., 2005).

If we consider the case for which both $f_i(\cdot)$ and $\mathcal{L}_i(\cdot)$ are linear in the state-action frequencies μ as described above (i.e. they factor over state-action pairs), the maximum margin problem becomes

$$\min_{w,\zeta_i} \frac{1}{2}\|w\|^2 + \frac{\gamma}{n}\sum_i \beta_i \zeta_i^q \tag{3}$$

$$s.t.\ \ \forall i \quad w^T F_i \mu_i + \zeta_i \geq \max_{\mu \in \mathcal{G}_i} w^T F_i \mu + l_i^T \mu \tag{4}$$

where $\mu \in \mathcal{G}_i$ expresses the Bellman-flow constraints for each MDP, namely that $\mu \geq 0$ satisfies:

$$\sum_{x,a} \mu^{x,a} p_i(x'|x,a) + s_i^{x'} = \sum_a \mu^{x',a}$$

The nonlinear, convex constraints in Equation 4 can be transformed into a compact set of linear constraints (Taskar et al., 2005; Taskar et al., 2003) by computing the dual of the right hand side of each yielding:

$$\forall i \quad w^T F_i \mu_i + \zeta_i \geq \min_{v \in V_i} s_i^T v \tag{5}$$

where $v \in V_i$ are the value-functions that satisfy the Bellman primal constraints:

$$\forall x, a \quad v^x \geq (w^T F_i + l_i)^{x,a} + \sum_{x'} p_i(x'|x,a) v^{x'} \tag{6}$$

By combining the constraints together we can write one compact quadratic program:

$$\min_{w,\zeta_i,v_i} \frac{1}{2}\|w\|^2 + \frac{\gamma}{n}\sum_i \beta_i \zeta_i^q \tag{7}$$

$$s.t.\ \ \forall i \quad w^T F_i \mu_i + \zeta_i \geq s_i^T v_i \tag{8}$$

$$\forall i, x, a \quad v_i^x \geq (w^T F_i + l_i)^{x,a} + \sum_{x'} p_i(x'|x,a) v_i^{x'} \tag{9}$$

This results provides a represenation of what we call the Maximum Margin Planning (MMP) problem as a compact quadratic program. Note that the number of constraints scales linearly with state-action pairs and training examples. While off-the-shelf quadratic programming software can be applied at this point to directly optimize this program, we provide an alternative formulation in Section 3 that allows us improve significantly over this via the utilization of subgradient methods.

We consider additional useful loss functions in Section 4, where we detail examples relevant to path planning problems.

3. Efficient Optimization

In practice, solving the quadratic program in Equation 9 is at least as hard as solving the linear programming formulation of a single MDP. While this can be an appropriate strategy for a class of MDPs, it is generally appreciated that for many problems there exist specially designed algorithms such as policy iteration and A* that can solve particular classes of MDPs both theoretically and empirically more rapidly. Developing efficient specialized algorithms, and especially leveraging existing inference algorithms, is an open problem for general structured maximum margin problems. (Taskar et al., 2005). We present a simple approach based on the subgradient method (Shor, 1985) that allows the use of fast maximization algorithms (e.g. fast planners) for a provably efficient learning strategy.

The first step is to transform the optimization program into a "hinge-loss" form. The hinge-loss view is a common way to understand the maximum margin problem and relate it to other methods like logistic regression and AdaBoost. This view comes from

[3] Unless stated otherwise, $\|.\|$ denotes the L_2 norm.

noting that the slack variables ζ_i are tight and thus equal $\max_{\mu \in \mathcal{G}_i}(w^T F_i + l_i^T)\mu - w^T F_i \mu_i$. We can therefore move these constraints into the objective function, simplifying the problem into a single cost function:

$$c_q(w) = \frac{1}{n}\sum_{i=1}^{n} \beta_i \left(\max_{\mu \in \mathcal{G}_i}(w^T F_i + l_i^T)\mu - w^T F_i \mu_i \right)^q + \frac{\lambda}{2}\|w\|^2 \qquad (10)$$

where we have multiplied through by $\frac{\lambda}{\gamma}$ to make a regularized risk functional interpretation more clear (Rifkin & Poggio, 2003). Again, $q \in \{1, 2\}$ defines the slack penalization.

This objective function is convex, but nondifferentiable. We can optimize it by utilizing a generalization of gradient descent called the subgradient method (Shor, 1985). A *subgradient* of a convex function $c : \mathcal{W} \to \mathbb{R}$ at w is defined as a vector g such that

$$\forall w' \in \mathcal{W},\ g^T(w' - w) \le c(w') - c(w) \qquad (11)$$

Note that subgradients need not be unique, though at points of differentiability, they necessarily agree with the gradient. We denote the set of all subgradients of $c(\cdot)$ at point w by $\partial c(w)$.

To compute the subgradient of $c(w)$, we make use of the following four well known properties: (1) subgradient operators are linear; (2) the gradient is the unique subgradient of a differentiable function; (3) denoting $y* = \text{argmax}_y[f(x, y)]$ for differentiable $f(., y)$, $\nabla_x f(x, y*)$ is a subgradient of the piecewise differentiable convex function $\max_y[f(x, y)]$; (4) an analogous chain rule holds as expected. We are now equipped to compute a subgradient $g_w^q \in \partial c(w)$ of our objective function (10):

$$g_w^q = \frac{1}{n}\sum_{i=1}^{n} q\beta_i \left((w^T F_i + l_i^T)\mu^* - w^T F_i \mu_i\right)^{q-1} \cdot F_i \Delta^w \mu_i + \lambda w \qquad (12)$$

where $\mu* = \arg\max_{\mu \in \mathcal{G}_i}(w^T F_i + l_i^T)\mu$ and $\Delta^w \mu_i = \mu^* - \mu_i$. This latter expression points out that, intuitively, the subgradient compares the state-action visitation frequency counts between the example policy and the optimal policy with respect to current reward function $w^T F_i$.

Note that computing the subgradient requires solving the problem $\mu^* = \arg\max_{\mu \in \mathcal{G}_i}(w^T F_i + l_i^T)\mu$ for each MDP. This is precisely the problem of solving the particular MDP with the reward function $w^T F_i + l_i^T$, and can be efficient implemented via a myriad of specialized algorithms. Algorithm 1 details the application of the subgradient method to the Maximum Margin Planning problem.

Algorithm 1 Max Margin Planning

```
procedure MMP(Training set $\{F_i, \mu_i, l_i\}_{i=1}^N$, Regularization parameter $\lambda > 0$, Stepsize sequence $\{\alpha_t\}$ (learning rate), Iterations $T$)
    $t \leftarrow 1$
    $w \leftarrow 0$
    while $t \le T$ do
        Compute optimal policy and state action visitation frequencies $\mu^{*,i}$ for each input map for loss augmented cost map $(w^T F_i + l_i^T)$.
        Compute $g \in \partial c(w)$ as in Equation 12.
        $w \leftarrow w - \alpha_t g$
        (Optional): Project $w$ on to any additional constraints.
        $t \leftarrow t + 1$
    end while
    return $w$
end procedure
```

The basic iterative update given $g_t \in \partial c(w_t)$ and α_t is

$$w_{t+1} = \mathcal{P}_{\mathcal{W}}\left[w_t - \alpha_t g_t\right] \qquad (13)$$

where $\mathcal{P}$ projects w onto any problem specific (convex) constraints we may impose on w.[4]

3.1. Guarantees in the Batch Setting

In the batch setting, this algorithm is one of a well studied class of algorithms forming the subgradient method (Shor, 1985).[5] Crucial to this method is the choice of stepsize sequence $\{\alpha_t\}$; and convergence guarantees vary accordingly. Our results are developed from (Nedic & Bertsekas, 2000) which analyzes *incremental* subgradient algorithms, of which the subgradient method is a special case.

Our results require a strong convexity assumption to hold for the objective function. Given $\mathcal{W} \subseteq \mathbb{R}^d$, a function $f : \mathcal{W} \to \mathbb{R}$ is η-strongly convex if there exists $g : \mathcal{W} \to \mathbb{R}^d$ such that for all $w, w' \in \mathcal{W}$:

$$f(w') \ge f(w) + (g(w))^T(w' - w) + \eta\|w' - w\|^2 \qquad (14)$$

Theorem 1. Linear convergence of constant stepsize sequence. *Let the stepsize sequence* $\{\alpha_t\}$

[4] It is actually sufficient that P be an approximate projection operator that need only satisfy the inequality $\forall w' \in \mathcal{W},\ \|\mathcal{P}_{\mathcal{W}}[w] - w'\| \le \|w - w'\|$.

[5] The term "subgradent method" is used in lieu of "subgradient descent" because the method is not technically a descent method. Since the stepsize sequence is chosen in advance, the objective value per iterate can, and often does, increase.

of Algorithm (1) be chosen as $\alpha_t = \alpha \leq \frac{1}{\lambda}$. Furthermore, assume for a particular region of radius R around the minimum, $\forall w, g \in \partial c(w)$, $\|g\| \leq C$. Then the algorithm converges at a linear rate to a region of some minimum point x^* of c bounded by $\|x_{\min} - x^*\| \leq \sqrt{\frac{\alpha C^2}{\lambda}} \leq \frac{C}{\lambda}$.

Proof. (Sketch) By the strong convexity of $c_q(w)$ and Proposition 2.4 of (Nedic & Bertsekas, 2000) we have

$$\|w_{t+1} - w^*\|^2 \leq (1-\alpha\lambda)^{t+1}\|w_0 - w^*\|^2 + \frac{\alpha C^2}{\lambda}$$
$$\underset{t\to\infty}{\longrightarrow} \frac{\alpha C^2}{\lambda} \leq \frac{C^2}{\lambda^2}$$

□

This theorem shows that we attain a linear convergence rate under a sufficiently small constant stepsize, but that convergence is only to a region around the minimum. Alternatively, we can choose a diminishing stepsize rule of the form $\alpha_t = \frac{r}{t}$ for $t \geq 1$, where r is some positive constant that can be thought of as the learning rate. Under this rule, Algorithm 1 is guaranteed to converge to the minimum, but only at a sublinear rate under the above strong convexity assumption (see (Nedic & Bertsekas, 2000), Proposition 2.8).

3.2. Optimization in an Online Setting

In contrast with many optimization techniques, the subgradient method naturally extends from the batch setting (as presented) to an online one. In the online setting one imagines seeing several planning problems on closely related domains: in particular, one may observe a domain, be required to plan a path for it, and then observe the "correct" path (or observe the corrections of the suggested path). At each time step i: (1) We observe $\mathcal{G}_i$ and F_i, (2) Select a weight vector w_i and using this compute a resulting path (3) Finally we observe the true policy y_i. Thus, we can define $c_i(w) = \frac{\lambda}{2}\|w\|^2 + \max_{\mu\in\mathcal{G}_i}(w^T F_i + l_i)\mu - w^T F_i \mu_i$ to be the strongly convex cost function (see Equation 10) at time i, which we can compute given y_i, $\mathcal{G}_i$, and F_i. This is now an **online convex programming problem** (Zinkevich, 2003), to which we will apply the extension of Greedy Projection by (Hazan et al., 2006) where the learning rate is $1/(i\lambda)$.

The loss we truly care about on round i is the planning loss, $\mathcal{L}(\mu_i, \sigma_i)$, where σ_i is the policy we choose. Space doesn't permit a proof,[6] but the following may be derived using tools from (Hazan et al., 2006):

Theorem 2. Sublinear regret for subgradient MMP. Assume that the features in each state are bounded in norm by 1. Further, assume that there is a w^* that with hindsight achieves for all i, $\max_{\mu\in\mathcal{G}_i} w^T F_i(\mu - \mu_i) + l_i^T \mu = 0$, then:

$$\sum_i \mathcal{L}(\mu_i, \sigma_i) \leq \frac{1}{\lambda}(1 + \ln n) + n\lambda\|w^*\|^2 \qquad (15)$$

Choosing $\lambda = \frac{\sqrt{1+\ln n}}{\|w^*\|\sqrt{n}}$, then:

$$\sum_i \mathcal{L}(\mu_i, \sigma_i) \leq \|w^*\|\sqrt{n(1+\ln n)} \qquad (16)$$

Thus, if we know n and the achievable margin, our loss grows only sublinearly in the number of time steps.
□

Further, in the case when there is no perfect w^*, we can instead achieve a competitive ratio that scales inversely with the margin. Observe that Theorem 2 is a result about the additive loss function $\mathcal{L}$ specifically. If we attempted instead to consider, for instance, a setting where we did not require more margin from higher loss paths (e.g. 0/1 loss on paths) our bound would be much weaker and scale with the size of the domain.

3.3. Modifications for Acyclic Positive Costs

For infinite horizon problems in acyclic domains A* and its variants are generally the most efficient route to finding a good plan. Such domains require rewards to be strictly negative (equivalently, costs must be strictly positive), otherwise infinite reward paths may result. The strictness of this negativity is to ensure the existence of an admissable heuristic.

Assuming $F_i \geq 0$ (element-wise), this can be implemented via component-wise negativity constraints on w or a set of constraints enforcing the negativity of the reward for each state-action pair individually. Exact projection onto the former can be implemented simply by setting the violated components of w to 0, and an approximate projection onto the latter can be implemented efficiently by iteratively projecting onto the most violated constraint.

3.4. Incorporating Prior Knowledge

It is often important to be able to build in prior knowledge about cost-functions that may improve learning performance. One useful technique is to regularize the solution about a prior belief on w instead of the 0 vector. Another is to have our loss function mark certain state-action pairs as being poor choices: this forces our algorithm to have large margin with respect to them. Finally, we may incorporate domain knowledge in the form of constaints on w: e.g., we may require that a certain area state have at least double the cost of another state. All of these are powerful methods to

[6] See appendix of extended version of paper at http:// (removed for review).

transfer expert knowledge to the learner *in addition* to the use of training examples.

4. Experimental Results

To validating these concepts we focused on the practical problem of path planning using the batch learning algorithm presented in section 3. In this setting, the MDP can be viewed as a two-dimensional map discretized uniformely into an array of cells. Each cell represents a particular location in the world and typical actions include moving from a given cell to one of the eight neighboring cells. In all experiments, we used A^* as our specialized planning algorithm, set $\beta_i = 1/\|\mu_i\|_1^q$, chose $q = 2$, and used reasonable values for regularization.

We first exhibit the versatility of our algorthm in learning distinct concepts within a single domain. Differing example trajectories, demonstrated in one region of a map, lead to a significantly different behavior in a separate holdout region after learning. Figure 1 shows qualitatively the results of this experiment. The behavior presented in the top row suggests a desire to stay on the road, while that portrayed in the bottom row embodies more clandestine needs. By column, from left to right, the images depict the training example presented to the algorithm, the learned cost map on a holdout region after training, and the resulting behavior produced by A^* over this region.[7]

For our second experiment, the data derived entirely from laser range readings (ladar) over the region of interest collected during an overhead helicopter sweep.[8] A visualization of the raw data is depicted in Figure 3. Figure 2 shows typical results from a holdout region. The learned behavior (green) often matches well the desired behavior (red). Even when the learner failed to match the desired trajectory exactly, the learned behavior adheres to the primary rules set forth implicitly by the examples. Namely, the learner finds an efficient path that avoids buildings (white) and grassy areas (gray) in lieu of roads.

Notice that the loss-augmented path (blue) in this figure performs generally worse than the final learned trajectory. This is because loss-augmentation makes areas of high loss more desirable than they would be in the final learned map. Intuitively, if the learner is able to perform well with respect to the loss-augmented cost map, then it should perform even better without the loss-augmentation; that is, the concept is learned with *margin.*

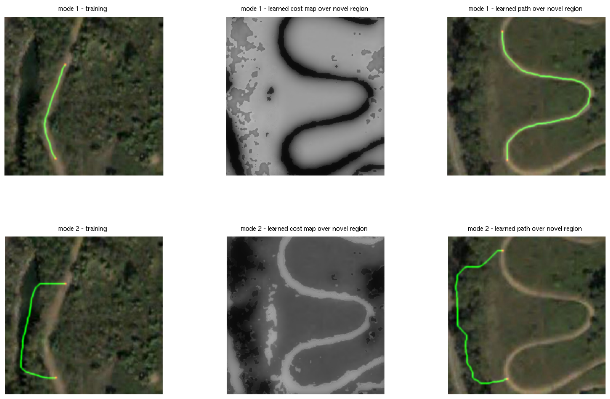
mode 1 - training
mode 1 - learned cost map over novel region
mode 1 - learned path over novel region
mode 2 - training
mode 2 - learned cost map over novel region
mode 2 - learned path over novel region

Figure 1. Demonstration of learning to plan based on satellite color imagery. For a particular training/holdout region pair, the top row trains the learner to follow the road while the bottom row trains the learner to "hide" in the trees. From left to right, the columns depict the single training example presented, the learned cost map over the holdout region, and the corresponding learned behavior over that region. Cost scales with intensity.

For comparison, we attempted to learn similar behavior using two alternative approaches to MMP. First, we tried the reactive approach of directly learning from examples a mapping that takes state features to next actions as in (LeCun et al., 2006).[9] Unfortunately, the resulting paths were rather poor matches to the training data. See Figure 3 for a typical example of a path learned by the classifier.

A somewhat more successful attempt was to try to learn costs directly by a hand labeling of regions. This provides dramatically more explicit information to the learner than MMP requires: a trainer provided examples regions of low, medium, and high costs, based upon (1) expert knowledge of the planner, (2) iterated training and observation, and (3) the trainer had prior knowledge of the cost maps found under MMP batch learning on this data set.[10] Although cost maps

[7] The features used in this experiment were derived entirely from a single overhead satellite image. We discretized the image into five distinct color classes and added smoothed versions of the resulting features to propegate proximity information.

[8] Raw features were computed from mean and standard deviations from each of elevation, signal reflectance, hue, saturation, and local ladar shape information (Vandapel et al., 2004). Again, we added smoothed versions of the raw features to utilize proximity information.

[9] We used the same training data, training Regularized Least Squares classifiers (Rifkin & Poggio, 2003) to predict which nearby state to transition to. It proved difficult to engineer good features here; our best results come from using the same local state features as MMP augmented with distance and orientation to the goal. The learner typically achieved between 0.7-0.85 prediction accuracy.

[10] The low cost examples came from the example paths and the medium/high cost examples were supplied separately. Low cost and high cost examples were chosen as

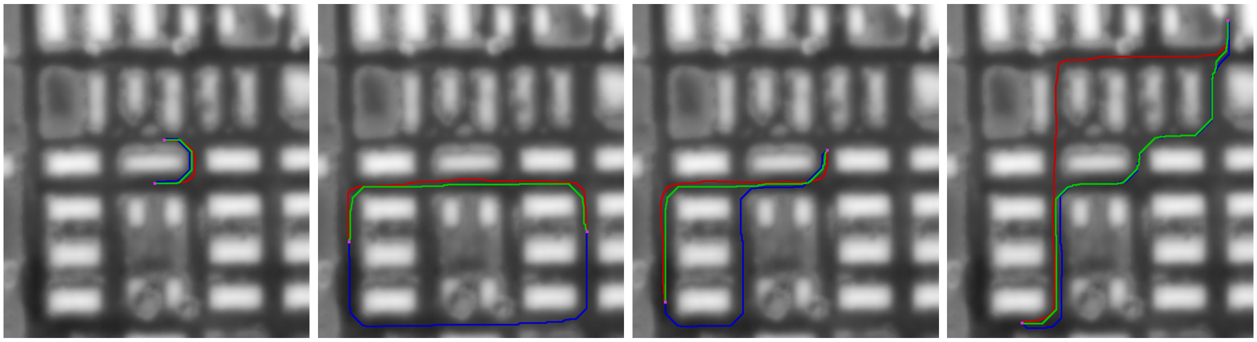

Figure 2. See Figure 3 for data-set. Data shown are MMP learned cost maps (dark low cost) with a teacher supplied path (red), loss-augmented path (blue), and final learned path (green). These are learned results on a holdout set.

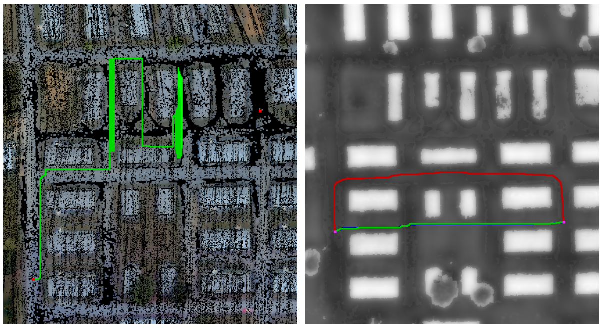

Figure 3. Left: the result of a next-action classifier applied super-imposed on a visualization of the second data-set. Right: a cost map learned by manual training of a regression. The learned paths (green) in both cases are poor approximations of the training examples (not shown on left, red on right).

given this extra information looked qualitatively correct, Figure 3 and Figure 4 demonstrate that the planning performance was significantly inferior.

5. Related and Future Work

Two strands of work are most directly related to Maximum Margin Planning. One is work on Inverse Reinforcement Learning (IRL). The goal in IRL is to observe an agent acting in an MDP (with unknown reward function) and extract from the agent's behavior the reinforcement function it is attempting to optimize. Unfortunately, it is well known that standard IRL is ill-posed: the zero reward function, for instance, admits all policies as optimal ones. Nevertheless, there have been some heuristic attempts to use IRL ideas to similar effect as MMP. One useful heuristic is to have a learning algorithm try to match expected feature counts (in our language $\sum_i F\mu_i$) (Abbeel & Ng, 2004) between the learner's policy and the demonstrated examples. This variant of IRL differs from MMP in that MMP is designed to allow the demonstration of policies from more than a single MDP: we typically demonstrate examples over multiple feature maps and with different start and goal states, with the aim of the learner to extrapolate to entirely new maps and goals. This leads to significantly different algorithmic approaches. The relation between IRL and MMP is reminiscent of the distinction between generative and discriminative learning: IRL feature matching is designed for learning when we believe an agent is acting (near-optimally) in an MDP and further that we can (nearly) match feature expectations. These assumptions, like those of generative models, are strong ones: the ability to match feature expectations, for instance, means that the algorithm's behavior will be near-optimal for *every* cost function linear in the features simultaneously (i.e. for any reward function $w^T F$). MMP makes the weaker assumption that our goal is to directly mimic output behavior and is agnostic about a real underlying MDP or reward function. MDPs here structure the output decisions and provide a potentially natural class of experts that we attempt to compete with.

The other strand of closely related work is that of Structured Max-Margin optimization techniques. The subgradient optimization presented here makes practical learning in problems where straightforward QP techniques are intractable. It is often the case, that path planning problems are quickly solved with MMP that would be too large to even represent for a generic QP solver. Recently, a number of other techniques have been proposed to solve problems of these kinds, including cutting plane (Tsochantaridis et al., 2005) and extra-gradient techniques. (Taskar et al., 2006) The latter is applicable where inference may be written as a linear program and is also able to achieve linear convergence rates. The subgradient method

minimum and maximum values for A*, respectively. Multiple medium cost levels were tried.

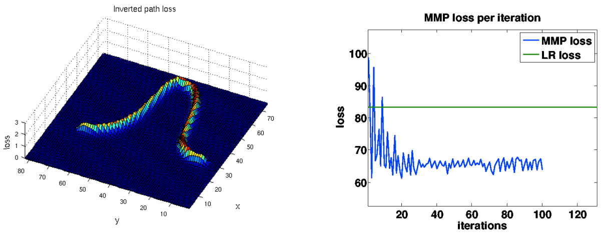

Figure 4. (Left) Visualization of inverted loss function $(1 - l(x))$ for a training example path. (Right) Comparison of holdout loss of MMP (by number of iterations) to a regression method where a teacher hand-labeled costs. The next-action classifier approach had a much larger loss than either method.

has the advantage of being applicable to any problems where loss augmented inference may be quickly solved including by combinatorial methods. Further, our algorithm extends naturally to the online case, where sublinear regret bounds are available. It will be interesting to compare these methods on problems where they are both applicable. In recent work, (Duame et al., 2006) has considered reinforcement learning based approaches to structured classification. Subgradient methods for (unstructured) margin linear classsification were considered in (Zhang, 2004). (LeCun et al., 1998) considers the use of gradient methods for learning using decoding methods such as Viterbi; our approach (if applied to sequence labeling) extends such methods to use notions of structured maximum margin.

Our current research effort is to take advantage of the online behavior and apply the algorithm in a replanning scenario where new features are being generated continuously from on-board perception. Our algorithm has natural applications in a variety of non-robotic domains where planning is essential. We are currently applying improved versions of the subgradient method, also able to leverage existing specialized algorithms inference algorithms, to a variety of maximum margin learning problems.

Acknowledgements

We thank Omead Amidi, Boris Sofman, Tony Stentz, and Nicolas Vandapel for their generous help with the experimental work as well as valuable conversations with Geoff Gordon. The first two authors gratefully acknowledge the partial support of this research by the DARPA Learning for Locomotion contract.

References

Abbeel, P., & Ng, A. Y. (2004). Apprenticeship learning via inverse reinforcement learning. *ICML '04: Proceedings of the twenty-first international conference on Machine learning.*

Duame, H., Langford, J., & Marcu, D. (2006). Search-based structured prediction. In Preparation.

Hazan, E., Kalai, A., Kale, S., & Agarwal, A. (2006). Logarithmic regret algorithms for online convex optimization. To appear in COLT 2006.

Hebert, M., Stentz, A. T., & Thorpe, C. (1998). Mobility planning for autonomous navigation of multiple robots in unstructured environments. *Proceedings of ISIC/CIRA/ISAS Joint Conference.*

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE* (pp. 2278–2324).

LeCun, Y., Muller, U., Ben, J., Cosatto, E., & Flepp, B. (2006). Off-road obstacle avoidance through end-to-end learning. In *Advances in neural information processing systems 18.*

Nedic, A., & Bertsekas, D. (2000). Convergence rate of incremental subgradient algorithms. *Stochastic Optimization: Algorithms and Applications.*

Pomerleau, D. (1989). Alvinn: An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems 1.*

Puterman, M. (1994). *Markov decision processes: Discrete stochastic dynamic programming.* Wiley.

Rifkin, Y., & Poggio (2003). Regularized least squares classification. *Advances in Learning Theory: Methods, Models and Applications.* IOS Press.

Shor, N. Z. (1985). *Minimization methods for non-differentiable functions.* Springer-Verlag.

Taskar, B., Chatalbashev, V., Guestrin, C., & Koller, D. (2005). Learning structured prediction models: A large margin approach. *Twenty Second International Conference on Machine Learning (ICML05).*

Taskar, B., Guestrin, C., & Koller, D. (2003). Max margin markov networks. *Advances in Neural Information Processing Systems (NIPS-14).*

Taskar, B., Lacoste-Julien, S., & Jordan, M. (2006). Structured prediction via the extragradient method. In *Advances in neural information processing systems 18.*

Tsochantaridis, I., Joachims, T., Hofmann, T., & Altun, Y. (2005). Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research*, 1453–1484.

Vandapel, N., Huber, D., Kapuria, A., & Hebert, M. (2004). Natural terrain classification using 3-d ladar data. *IEEE International Conference on Robotics and Automation.*

Zhang, T. (2004). Solving large scale linear prediction problems using stochastic gradient descent algorithms. *Proceedings of ICML.*

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *Proceedings of the Twentieth International Conference on Machine Learning.*