Imitation Learning for Locomotion and Manipulation

Nathan D. Ratliff
Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
ndr@ri.cmu.edu

J. Andrew Bagnell
Robotics Institute / MLD
Carnegie Mellon University
Pittsburgh, PA 15213
dbagnell@ri.cmu.edu

Siddhartha S. Srinivasa
Intel Research Pittsburgh
Pittsburgh, PA 15213
siddhartha.srinivasa@intel.com

Abstract

Decision making in robotics often involves computing an optimal action for a given state, where the space of actions under consideration can potentially be large and state dependent. Many of these decision making problems can be naturally formalized in the multiclass classification framework, where actions are regarded as labels for states. One powerful approach to multiclass classification relies on learning a function that scores each action; action selection is done by returning the action with maximum score. In this work, we focus on two imitation learning problems in particular that arise in robotics. The first problem is footstep prediction for quadruped locomotion, in which the system predicts next footstep locations greedily given the current four-foot configuration of the robot over a terrain height map. The second problem is grasp prediction, in which the system must predict good grasps of complex free-form objects given an approach direction for a robotic hand. We present experimental results of applying a recently developed functional gradient technique for optimizing a structured margin formulation of the corresponding large non-linear multiclass classification problems.

1 Introduction

Robot manipulation tasks usually involve a large number of actions possible at a given state. Importantly, skilled humans operators are often quite adept at choosing effective actions for a given state of the robot and can demonstrate this correct behavior. It is usually quite difficult for such an expert to articulate their strategy however; the decision is often a nonlinear combination of numerous desiderata such as stability, energy minimization, actuator limits, and future intent. It is much easier for the operator to demonstrate optimal actions than it is to carefully enumerate the complex function being optimized to produce the action. In *imitation learning*, we study algorithms that generalize from such operator demonstration to effectively chose actions for new states. Many of these learning problems can be naturally formalized in the multiclass classification framework, where actions are regarded as labels for states [1, 2]. This multiclass imitation learning approach is especially suited to robot applications because demonstration provides a natural method for an operator to specify optimality as well as to specify actions that the operator considers as "close" or equivalent (due to symmetries, for example).

The multiclass techniques we study in this paper learn a function that operates by scoring each action and returning the maximum scoring action as the optimal choice. The goal of the learning procedure is to find a score function that well captures the demonstrated behavior; in essence, the

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE **DEC 2007**	2. REPORT TYPE	3. DATES COVERED **00-00-2007 to 00-00-2007**
4. TITLE AND SUBTITLE **Imitation Learning for Locomotion and Manipulation**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Carnegie Mellon University ,Robotics Institute,Pittsburgh,PA,15213**		8. PERFORMING ORGANIZATION REPORT NUMBER **CMU-RI-TR-07-45**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Decision making in robotics often involves computing an optimal action for a given state, where the space of actions under consideration can potentially be large and state dependent. Many of these decision making problems can be naturally formalized in the multiclass classification framework, where actions are regarded as labels for states. One powerful approach to multiclass classification relies on learning a function that scores each action; action selection is done by returning the action with maximum score. In this work, we focus on two imitation learning problems in particular that arise in robotics. The first problem is footstep prediction for quadruped locomotion, in which the system predicts next footstep locations greedily given the current four-foot configuration of the robot over a terrain height map. The second problem is grasp prediction, in which the system must predict good grasps of complex free-form objects given an approach direction for a robotic hand. We present experimental results of applying a recently developed functional gradient technique for optimizing a structured margin formulation of the corresponding large non-linear multiclass classification problems.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**11**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

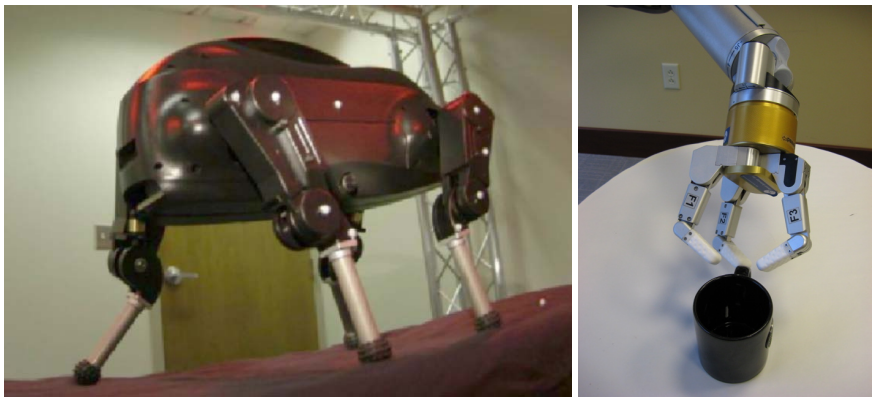

Figure 1: Experimental testbeds: Boston Dynamics' *LittleDog* quadruped robot (left), and the Barrett Technologies three-fingered hand (right).

procedure searches for a score function that makes the human choices appear optimal. Recently, a framework for designing such large multiclass predictors has been developed by using functional gradient techniques to optimize a simple structured-margin criterion. This approach allows us to adapt existing, "off-the-shelf" regression (or binary classification) techniques to learn the potentially complicated score function, making it a modular and simple to implement technique.

Our interest in this work is the demonstration of the multiclass learning technique to solving problems in robotic grasping and quadruped locomotion. From the machine learning viewpoint there is a surprising fundamental unity to these tasks. In both cases, the desired policy involves complex score functions, while demonstrations of desired behavior can be provided by expert operators at relatively little expense. Additionally, both problems have a large number of actions and can be straightforwardly optimized for the optimal action. We believe that many related robotics tasks have similar properties and may benefit from the approach taken here.

In what follows, we start by briefly introducing the learning technique before describing the experiments in detail. We finish with some concluding statements and comments on future work.

2 Structured-margin techniques

Many problems in imitation learning can be posed naturally as multiclass classification problems. However, while traditional multiclass classification problems often have a relatively small number of possible class labels (typically from 3-26), multiclass classification formulations of imitation learning problems often have many class possibilities.

For instance, in our grasping experiments below we take the set of class labels to be the set of preshape configurations a robot hand can take on given an approach direction. The corresponding classification problem has 2496 labels, orders of magnitude more than is typically considered in traditional multiclass problems. Because there are so many possible labels, traditional (unstructured) margin-based classification models often fail since no notion of label similarity is built into the learning system. In what follows, we review the structured margin multiclass classification setting of [3] used throughout this paper which explicitly utilizes information of this sort.

Let $\mathcal{X}$ be the input domain (state space) and assume that the set of labels (actions) can potentially be different for each domain element $x \in \mathcal{X}$. We denote the set of labels for a given x by $\mathcal{Y}_x$, and the combined set of all labels by $\mathcal{Y} = \bigcup_{x \in \mathcal{X}} \mathcal{Y}_x$. A multiclass classifier is defined by a score function $s : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ over these two sets. Given $x \in \mathcal{X}$, the classifier predicts the optimal scoring label

$$y^* = \arg\max_{y \in \mathcal{Y}_x} s(x, y) = \arg\max_{y \in \mathcal{Y}_x} s_x(y), \tag{1}$$

where we denote $s_x(y) = s(x, y)$ for notational convenience.

The algorithm [2] we describe here for solving large-scale multiclass classification problems is very general. In [1] we demonstrate the success of a similar technique on problems for which the number

of class labels is exponential is some domain variable; in principle the set of labels can be infinite. For our purposes, the only requirement is that the optimization in Equation 1 required for making predictions can be performed efficiently. In the experiments presented in this paper, the number of labels, while large, is small enough that the optimization can be implemented by a simple brute-force enumeration.

2.1 Structured-margin loss function

Given a data set $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$, our algorithm optimizes a convex upper bound on a loss function $\mathcal{L}(x_i, y_i, y)$. This loss function specifies the basic notion of loss on choosing label y for example x_i when the true label is y_i. For notational convenience, we often denoted the loss function as $\mathcal{L}_i(y)$ and the set of labels $\mathcal{Y}_{x_i}$ as $\mathcal{Y}_i$. This upper bound is given by

$$r[s] = \frac{1}{N} \sum_{i=1}^{N} \left(\max_{y \in \mathcal{Y}_i} \{ s_i(y) + \mathcal{L}_i(y) \} - s_i(y_i) \right). \tag{2}$$

If the loss function is always zero $\mathcal{L}_i = 0$, this function measures the sub-optimality of the example label. Minimizing this zero-loss objective, attempts to find a score function for which the example labels are scored higher than all other labels. Choosing nonzero loss, however, improves generalization by introducing a notion of structured-margin. Instead of requiring only that the example label is scored higher than all other labels, we require it to be better than each label y by an amount proportional to how bad we deem that label to be as measured by our loss function $\mathcal{L}_i(y)$.

When the number of classes is small a commonly used loss function is the binary loss. In this case, $\mathcal{L}_i(y) = 0$ when $y = y_i$ and 1 otherwise, and the margin loss of Equation 2 reduces to the well-known SVM loss [4]. However, for multiclass classification problems that arise in imitation learning there is often a natural notion of loss; the structured margin adapts accordingly for such problem.

Within the structured setting, the relative size of the required margin is allowed to vary from label to label proportionally to the loss of the label in question. If the loss of the label is large, the label is very different from the desired label and the margin term is large. However, if the loss of the label is small, the label is considered similar to the desired label and the margin term is small. Intuitively, we allow the learned score on labels similar to the example label to be similar to the example score, but force the score on vastly different labels to be much smaller than that on the example label. The experimental section (Section 3) details natural loss functions that arise for both the footstep and grasp prediction problems.

2.2 Functional gradient optimization

In [5], a simple but effective subgradient method was developed for optimizing the upper bound given in Eqn. 2 assuming the score function is linear in a set of features $f_i(y)$ extracted from the combined example $x_i \in \mathcal{X}$ and hypothesized label $y \in \mathcal{Y}_i$. The linearity requirement is removed in [1] by generalizing the subgradient method to learning nonlinear score functions using functional gradient techniques like those first formulated in [6] and [7].

Denoting the feature vector extracted for the pair $x_i \in \mathcal{X}$ and $y \in \mathcal{Y}_i$ as $f_i(y)$, the functional gradient[1] of the structured-margin loss (Eqn. 2) is given as

$$\begin{aligned} \nabla_s R[s] &= \nabla_s \frac{1}{N} \sum_{i=1}^{N} \left(\max_{y \in \mathcal{Y}_i} (s(f_i(y)) + \mathcal{L}_i(y)) - s(f_i(y_i)) \right) \\ &= \frac{1}{N} \sum_{i=1}^{N} \left(\delta_{f_i(y^*)} - \delta_{f_i(y_i)} \right), \end{aligned} \tag{3}$$

where we denote the loss-augmented prediction as $y_i^* = \arg\max_{y \in \mathcal{Y}_i} \{ s(f_i(y)) + \mathcal{L}_i(y) \}$. A full derivation of this algorithm can be found in [1]; we provide here only the high level intuition. The functional gradient is the direction in the space of score functions that would most improve

[1] Here we have used the property that the functional gradient of a function evaluated at a point is the delta function centered at that point. In this case, $\nabla_s s(f_i(y)) = \delta_{f_i(y)}$.

Algorithm 1 Imitation learning via functional exponential gradient descent

procedure IMITATIONLEARNING(Training set $\{f_i(\cdot), y_i, \mathcal{L}_i(\cdot)\}_{i=1}^N$, Step size sequence $\{\alpha_t\}$, Max iterations T)
  initialize $s = 0$
  for $t = 1 \dots T$ **do**
    initialize $\mathcal{D} = \emptyset$
    for $i = 1 \dots N$ **do**
      find $y^* = \arg\max_{y \in \mathcal{Y}_i} \{s(f_i(y)) + \mathcal{L}_i(y)\}$
      set $\mathcal{D} \leftarrow \mathcal{D} \cup \{(f_i(y_i), 1)\}$
      set $\mathcal{D} \leftarrow \mathcal{D} \cup \{(f_i(y^*), -1)\}$
    end for
    train binary classifier/regressor on $\mathcal{D}$ to produce h_t
    set $s \leftarrow s + \alpha_t h_t$
  end for
  return $s = \sum_{t=1}^T \alpha_t h_t$.
end procedure

performance on the loss function $r[s]$. Intuitively, to misclassified examples the functional gradient specifies a desire to increase the score on the demonstrated action thereby making the classifier more likely to choose it during the next iteration, and a decrease in score to the action that the classifier incorrectly chooses at the current iteration of learning thereby making the classifier less likely to choose it again during the next iteration.

Because of the delta functions, the functional gradient as defined above is tied particularly to the training examples and therefore does not generalize to new states. To provide generalization to new states we rely upon the generalization ability of standard classification and regression approaches. We "project" the functional gradient onto a simpler functional form that generalizes. For instance, we may try to find a neural network in a hypothesis space $\mathcal{H}$ of such functions that is both simple (low complexity, high prior probability) and represents the functional gradient well. Such a projection can be derived by maximizing the inner product with the negative functional gradient over the hypothesis space.

$$h^* = \arg\max_{h \in \mathcal{H}} \langle h, \ -\nabla_s R[s] \rangle \tag{4}$$

$$= \arg\max_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N \langle h, \ \delta_{f_i(y_i)} - \delta_{f_i(y^*)} \rangle \tag{5}$$

$$= \arg\max_{h \in \mathcal{H}} \frac{1}{N} \sum_{i=1}^N \left(h(f_i(y_i)) - h(f_i(y^*)) \right). \tag{6}$$

This projection step can be implemented as a reduction to binary classification or regression using a data set generated by collecting two examples for each x_i, one corresponding to the correct label $(f_i(y_i), 1)$, and another corresponding to the current loss-augmented prediction $(f_i(y^*), -1)$. Training a binary classifier using this data set returns a function approximating h^*. Intuitively, for each i, the first example attempts to make h^* have a positive value at $f_i(y_i)$ so that adding h^* to the previous hypothesis will increase the function at that point thereby increasing the score function of the correct label. Similarly, the second example attempts to make h^* negative at the current loss-augmented label, so as to reduce the score function at that point.

The technique generalizes gradient descent in the following sense: we identify the negative functional gradient of the loss function, project it onto a tangible space of functions using a binary classification or regression algorithm, and take a step in the direction of the resulting learned approximator.

This procedure leads to a simple iterative algorithm. Given a step size sequence $\{\alpha_t\}_{t=1}^\infty$, the algorithm proceeds as shown in Figure 2.2. In our experiments, we use $\alpha_t = \frac{\gamma}{\sqrt{t}}$ where γ is the initial step size.

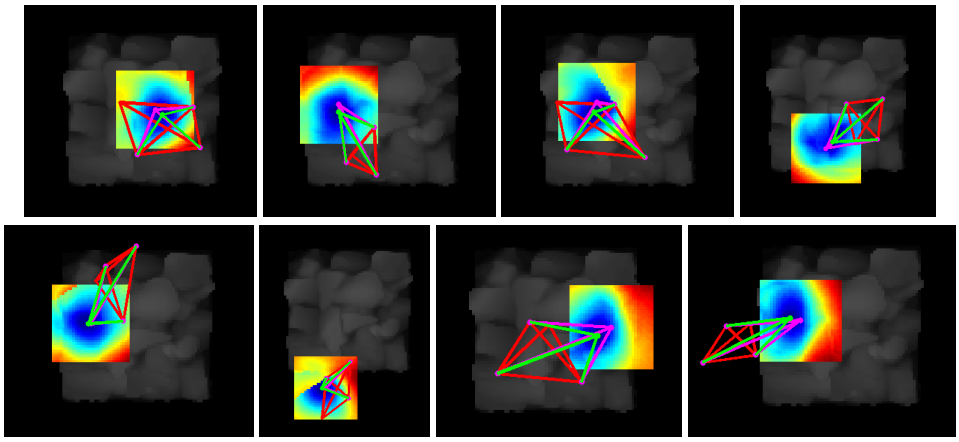

Figure 2: Results on a number of training examples of foot placement prediction demonstrating qualitative accuracy of predictions. The original configuration is depicted as red lines connecting each of the four feet. The example step is shown in magenta, and the predicted footstep (centered at the minimum of the rendered cost function), is given in green. In the rendered cost function, bluish shades are low cost while reddish shades are high cost. In all cases, this is overlaid atop the terrain height map.

2.3 Exponentiated gradient variant

The algorithm above is a generalization of standard gradient descent. In many cases, for instance when we wish the score function to be always positive or when we add together scores over multiple states (i.e. when performing minimum cost planning instead of brute force enumeration), we instead generalize a powerful related method: exponentiated gradient descent [8, 2]. Implementing the functional version of exponentiated gradient descent is a simple modification of the algorithm above. We replace the update of the score function with an exponentiated variant:

$$s \leftarrow \exp\{\log(s) + \alpha_t h_t\}. \tag{7}$$

Exponentiated gradient descent share similar convergence guarantees [9], but implements a different prior over the space of score functions. It places large prior weight on score functions with a great deal of dynamic range. We note that in this paper (unlike in [2]), exponentiating the scores does not change the argmax since we are not adding together scores over multiple states. It does, however, change the effect of the margin term. The exponentiated gradient variant was used in the grasp prediction experiments described below in Section 3.

3 Applications

This section details experiments using the functional gradient imitation learning techniques described above on two problems: quadruped locomotion, and grasp planning. These problems are detailed in Sections 3.1 and 3.2, respectively.

3.1 Quadruped Locomotion

The quadruped (Boston Dynamics' *LittleDog*) used for this experiment is depicted in Figure 1. The input state-space $\mathcal{X}$ consists of a four foot quadruped pose situated at a particular location of a 2.5-dimensional height map (specifying a height for each x-y location), in conjunction with an "active" foot which is to be moved next. For a given x, the prediction range $\mathcal{Y}_x$ is the set of all possible next step locations for the action foot. In these experiments, we take this region to be a square centered at a point computed from the current four foot configuration. This region is discretized into 961 $(= 31 \times 31)$ locations.

Training examples are extracted from the intermediate poses and next step foot locations chosen by a human operator remote controlling the robot across the terrain. While we could apply a full

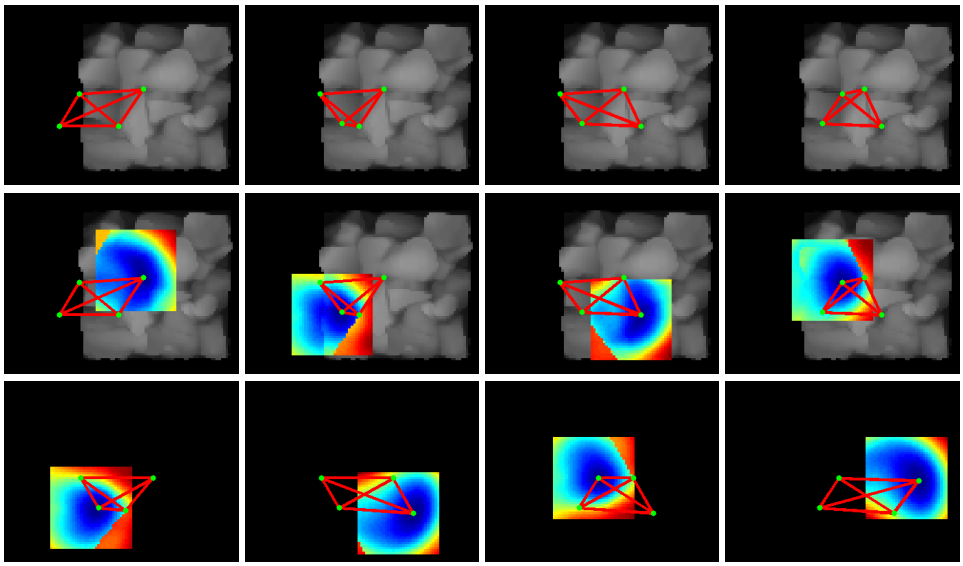

Figure 3: Generalization of quadruped footstep placement. The four foot stance was initialized to a configuration off the left edge of the terrain facing from left to right. The images shown demonstrate a sequence of footsteps predicted by the learned greedy planner using a fixed foot ordering. Each prediction starts from result of the previous. The first row shows the footstep predictions alone; the second row overlays the corresponding cost region (the prediction is the minimizer of this cost region). The final row shows footstep predictions made over flat ground along with the corresponding cost region showing explicitly the kinematic feasibility costs that the robot has learned.

planning based solution to the imitation learning problem as in [1], a one-step look-ahead, greedy approach approach is sufficient for the terrains we considered.

Features for each possible next location fall into two categories: action features and terrain features. Action features account for the kinematic constraints of the robot as they manifest themselves in the four-foot configuration. They include the distance from the hypothesized next-step location ν and each of the original locations of the feet as well as the radius of the inscribed circle of the support triangle resulting from that action. Terrain features, on the other hand, contain information describing local variation in the terrain. For these experiments a very simple set of terrain features was used. Seven smoothings of the height map were generated by performing Gaussian convolutions, and the feature vector was extracted as the vector 8 responses (including the raw height) at the pixel corresponding to the desired foot location.

We apply the the above gradient boosting algorithm to this data using the following loss function

$$\mathcal{L}(\nu, \nu_i) = 1 - \exp\left\{-\frac{(\nu - \nu_i)^2}{2\sigma^2}\right\} \tag{8}$$

where ν and ν_i are the hypothesis next foot location and the example next foot location, respectively. This loss function increases rapidly from 0 at ν_i and saturates to 1 at a distance regulated by the hyperparameter σ. The space $\mathcal{H}$ of regression algorithms we chose was small. It consisted of one hidden-layer neural networks which were trained using back-propagation.

Figure 2 shows select training examples along with their corresponding next step predictions. In each image, the original four-foot pose is shown with red lines connecting each foot to emphasize the relative orientation of the original locations of the feet. The example and predicted next step are depicted in magenta and green, respectively, and the learned cost map over the local search region is colored with blue shades corresponding to low cost graduating to red shades corresponding to high cost. All of this is superimposed over the terrain height map where the example resides.

The learned cost function combines the kinematic constraints of the robot with local terrain variation as can be seen in the images. Without terrain input, the cost function represents the forward stepping bias seen throughout the examples. The variation seen in the cost functions learned for each example

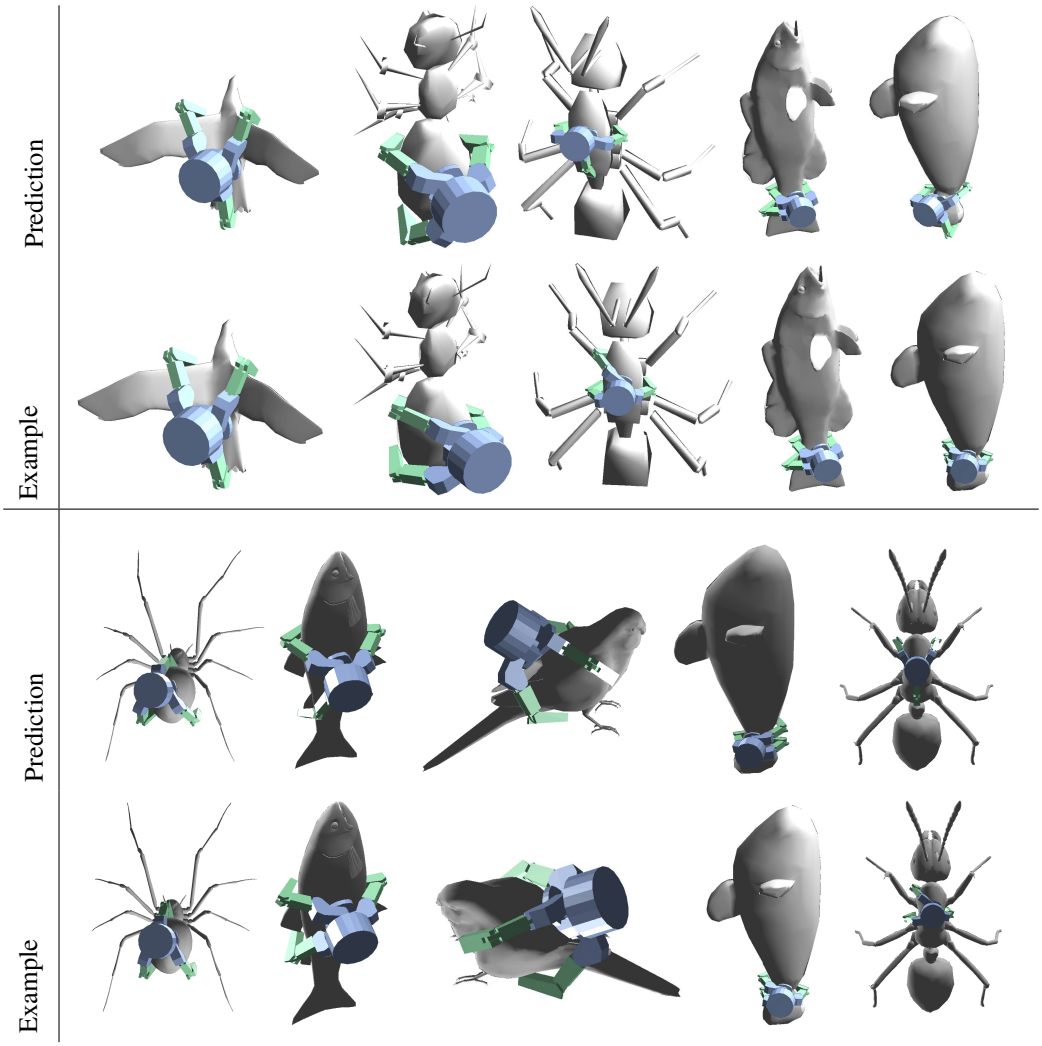

Figure 4: Grasp prediction results on ten hold-out examples. The training set consists of 23 training examples; each test result was generated by holding the example in question out and training on the rest.

comes from terrain components. The system learns to trade off reachability, a forward stepping bias, and local terrain considerations in a way that mimics the behavior exemplified in the data. For instance, the human operator had a tendency to step in local convexities (i.e. cracks) to improve walking stability and robustness. In a number of these examples, the system tends to place lower cost to such regions.

Generalization of the learned footstep predictor is demonstrated in Figure 3. The top row of this figure shows four steps of a footstep sequence over rough terrain generated by initializing the system to a nominal four foot configuration off the left hand edge of the terrain, and recursively predicting each next foot location given the current configuration using a fixed foot ordering. The middle row shows the same generated sequence segment along with the associated learned cost function used to predict each step. The final row shows a footstep sequence generated in the absence of terrain demonstrating explicitly the kinematic feasibility profile that the robot has learned from the data.

3.2 Grasp planning

Grasp planning is often framed as an optimization over a grasp metric which evaluates the quality of a grasp configuration relative to the object being grasped. Variation in planners can be categorized into the method used to discretize the continuous space of grasp configurations, and the grasp met-

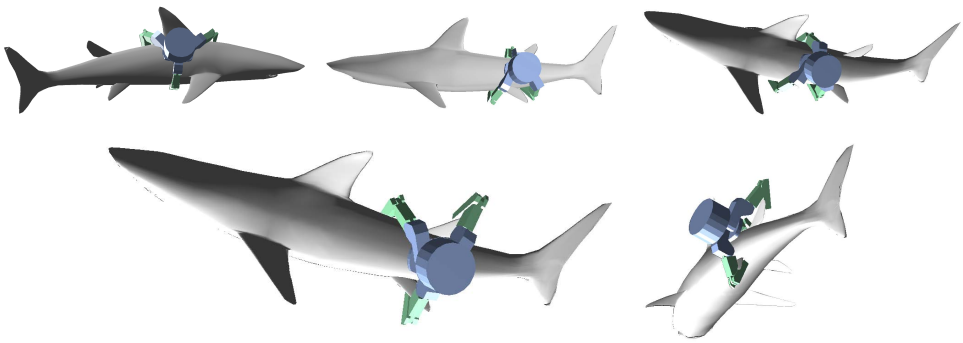

Figure 5: The top row shows three grasps of the same object from varying approach direction. The bottom rows shows from two perspectives a unique grasp that arises because the current feature set does not include information about fragility or flexibility of various parts of the object. Were this object perfectly rigid, this would be a reasonable grasp. The framework allows for easy addition of such extra features.

ric begin optimized. The discretization of the grasp space can be further segmented into a general approach direction for the hand and the configuration of the hand given the approach direction. In this work we assume the approach direction has been given to us by external means for two reasons. First, [10] demonstrated that a good approach direction/point can be predicted from binocular imagery, and in that work generalization of pinch grasping using a simple parallel jaw gripper was demonstrated for a number of previously unseen objects. Second, the true approach direction chosen for a given object is very strongly influenced by task parameters as well as environmental considerations, like workspace obstacles and the capabilities of the arm to which the hand is attached.

The hand used for this experiment (Barrett technology's *Barrett Hand*) is depicted in Figure 1. This hand has three fingers, each of which has two joints driven by a single actuator. When moving freely, the distal joint moves at a fixed rate with respect to the proximal joint, much like the motion of a human finger. One of the fingers is stationary relative to the palm, while the other two can move radially around the palm in unison. This degree of freedom is term the *fingerspread*. The hand has ten degrees of freedom in total coming from the six global translation and rotation degrees, in conjunction with the three individual finger joints, and the fingerspread. While the two joints on the fingers are constrained, each finger has a torque redirection mechanism to transfer all force to the distal joint once the proximal link has made contact. This mechanism, called *breakaway*, induces stronger grasps by allowing each finger to continue curling around an object even after the proximal link has made contact.

The score function s for this problem is log grasp metric, and the grasp chosen for a given space of grasps $\mathcal{Y}_x$ for grasping object x is given by a grasp planner which implements $y^* = \arg\max_{y \in \mathcal{Y}_x} \exp\{s(x, y)\}$.

3.3 Grasp demonstration

We discretize the space of grasp candidates in a way similar to that described in [11]. We define a preshape to be a configuration of the hand at a distance from the surface of the object. Given a preshape we run a simple grasp controller which moves the hand toward the object along an approach direction until it is a particular standoff distance away from the first collision with the object. At that point we close each of the fingers around the object implementing breakaway as described above.[2]

We are provided the approach direction and we orient the palm normal to this direction. The resulting free degrees of freedom consist of a standoff parameter and preshape parameters, namely the roll of the hand around the axis of approach, and the fingerspread. This gives us a three-dimensional space of grasp parameters (roll, fingerspread, standoff) which we discretize. We chose steps of size $\pi/24$

[2]Since there may be occlusions (e.g. appendages in our case) that we want to avoid during the approach, in our implementation we actually curl in the fingers to their stopping point, move the hand forward until it is inside the object, open the fingers entirely, and then back the hand out of the object until it is at a particular standoff distance from the last collision point before closing the fingers around the object.

to discretize the roll and fingerspread in the ranges $(-\pi, \pi]$ and $[-\pi/24, \pi/2]$, respectively, giving 48 roll points and 13 fingerspread points. That combined with four standoff values in increments of 0.01 in the range $[0, 0.04]$ gives a total of $48 \times 13 \times 4 = 2,496$ distinct grasp parameters.

Grasp examples were demonstrated in simulation by manually moving through the space of grasp parameters and selecting a setting which produces a good grasp. Force closure was explicitly not evaluated for these demonstrated grasps since a number of the grasps chosen by the trainer are form closure grasps that cage the object. A total of 27 training examples were generated in this way from a set of animal-like object models from the Princeton Shape Database[3] by varying approach direction and object scale. The various protrusions of the models (legs, antennae, fins, beaks) make them particularly challenging for grasping.

3.4 Features and loss function

To demonstrate the versatility of the learning algorithm we chose a relatively simple set of features that describe locally the shape of the object beneath each of the three fingertips and beneath the palm.

Let p and v be the point and direction of interest. We shoot a set of n rays $\mathcal{R} = \{r_i\}_{i=1}^n$ from the point p in a distribution of directions around v and extract from the collision points $\{c_i\}_{i=1}^n$ and normals at those points $\{u_i\}_{i=1}^n$ a set of features. The first elements of the feature vector are inversely correlated to the distance to collision: $\exp\{-\lambda\|c_i - p\|\}$. In this way the feature elements are bounded between 0 and 1, and rays that do not collide receive a value of 0. The second set of feature elements are computed as a projection of the distribution of the vectors formed by combining the contact point with the contact normal $w_i = [c_i; u_i]$ onto the space of Gaussians with diagonal covariance. This is simply computed by finding the vectors of means and standard deviations of the set $\{w_i\}_{i=1}^n$. These values are appended to the feature vector.

These ray features are computed for each finger and the palm and are then combined into a single vector representing the local relation between the hand and the object. A preprocessing step standardizes the features and then whitens them. The whitening step is implemented by performing PCA, keeping top 10 component projections, and normalizing their values by dividing by the standard deviation (latent value) along the component.

These features were used primarily to show that the algorithm produces reasonable results even when using a very simple set of features. There is a large amount of information which may be important in grasp prediction that these features do not account for. Two of the most obvious of these are torque produced by the object at the grasp point (dependent on both the mass of the object and the center of mass relative to the grasp point), as well as local properties of the object such as structural integrity and surface friction. An example of how the lack of structural integrity information can affect generalization is shown in the bottom row of Figure 5. Humans have an bias toward avoiding the fins of a shark or fish when grasping because of their flexibility. However, without representing that bit of information in the feature set, the learned system utilizes the flat surfaces of the fins as though the shark were a rigid statue. Additionally, we note that a better candidate for representing the distribution of normals described above is to use wrench coordinates, which are used in computations of force volumes and force closure measures.

The loss function we used for this experiment measured the physical discrepancy between the final configurations produced by the simple controller. This is implemented as the minimum distance matching between points in the fingertips of the example configuration and corresponding points in the predicted configuration. Specifically, let p_1, p_2, and p_3 be points in the three fingertips of the example configuration y and p'_1, p'_2, and p'_3 be corresponding points in the fingertips of the predicted configuration y'. Let Π be the set of all permutations of the set of indices $\mathcal{S} = \{1, \ldots, 3\}$, and denote a particular permutation as a mapping $\pi : \mathcal{S} \rightarrow \mathcal{S}$. We define the loss function as

$$\mathcal{L}(y, y') = \min_{\pi \in \Pi} \sum_{i=1}^{3} |p_i - p_{\pi(i)}|. \tag{9}$$

[3] http://shape.cs.princeton.edu/benchmark/

This gives low loss to configurations that are similar despite having vastly differing grasp parameters due to symmetries in the hand, while still giving high loss to configurations that are physically different.

3.5 Generalization

For each training example, we trained on the other 26 examples and used the final grasp metric to predict a grasp for the held out example. A single hidden layer neural network with 3 sigmoidal hidden units and a linear output was used as the weak learner $\mathcal{H}$. Because of the variability in neural network training with random initialization, for each boosting iteration, we trained an ensemble of 10 of these base learners by simply averaging the functions resulting from 10 separate trainings of the neural network on the same data set. We ran 10 iterations of scaled conjugate gradient to train each neural network.

Figure 4 displays renderings of the resulting grasp prediction (top row) along side the example grasp the trainer would have chosen for the corresponding approach direction (bottom row). We emphasize that these are generalization results and that the system was trained without knowledge of the grasp chosen by the trainer for these holdout examples. In particular, some of the grasp predictions are effectively the same but rotated when object symmetries make the grasps non-unique. Occasionally, the system predicts a grasp that is not stable. This is because of the limited number of examples and a lack of task-oriented reward function. The primary goal of imitation learning in this setting, however, is to produce a grasp prediction policy that is in the neighborhood of a good policy so that a reinforcement learning algorithm can be applied effectively to directly optimize this task-oriented reward function.

The top row of figure 5 demonstrates predicted grasps for the same object generalized to various approach directions. The prediction is fast and can be easily used bootstrap a high level planner that chooses an approach direction based on obstacles in the workspace and the kinematics of the arm.

4 Conclusions and future work

Imitation learning in many robotic applications can be naturally posed as a large-scale multiclass classification problem. In this paper, we demonstrated the effectiveness of recently developed functional gradient techniques for optimizing structured-margin multiclass classification machines by applying them to two complex imitation learning problems. We are working on making the currently brute-force search over possible actions more efficient: as the dimensionality of action spaces rises, we expect it will be necessary to consider more refined optimization procedures.

5 Acknowledgments

We would like to thank Rosen Diankov, Dmitry Berenson, Dave Ferguson, Matt Mason, and Mike Vande Weghe for their assistance and for many fruitful discussions. Nathan Ratliff was partially supported as an Intel Summer Fellow and by the DARPA Learning for Locomotion project.

References

[1] N. Ratliff, D. Bradley, J. A. Bagnell, and J. Chestnutt, "Boosting structured prediction for imitation learning," in *NIPS*, Vancouver, B.C., December 2006.

[2] J. A. Bagnell, J. Langford, N. Ratliff, and D. Silver, "The exponentiated functional gradient algorithm for structured prediction problems," in *The Learning Workshop*, San Juan, PR, 2007. [Online]. Available: http://snowbird.djvuzone.org/abstracts/153.pdf

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

[4] V. Vapnik, *The Nature of Statistical Learning Theory*. Springer-Verlag, NY, USA, 1995.

[5] N. Ratliff, J. A. Bagnell, and M. Zinkevich, "(online) subgradient methods for structured prediction," in *Artificial Intelligence and Statistics*, San Juan, Puerto Rico, 2007.

[6] L. Mason, J.Baxter, P. Bartlett, and M. Frean, "Functional gradient techniques for combining hypotheses," in *Advances in Large Margin Classifiers*. MIT Press, 1999.

[7] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," in *Annals of Statistics*, vol. 29(5), 1999a.

[8] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. New York, NY, USA: Cambridge University Press, 2006.

[9] S. I. Hill and R. C. Williamson, "Convergence of exponentiated gradient algorithms," *IEEE Trans. on Signal Processing*, vol. 49, no. 6, pp. 1208–1215, 2001.

[10] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng, "Robotic grasping of novel objects," in *NIPS*, 2007.

[11] A. T. Miller, S. Knoop, P. K. Allen, and H. I. Christensen, "Automatic grasp planning using shape primitives," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.