Statistical Techniques in Robotics (16-831, F14) Lecture#26 (December 4, 2014)

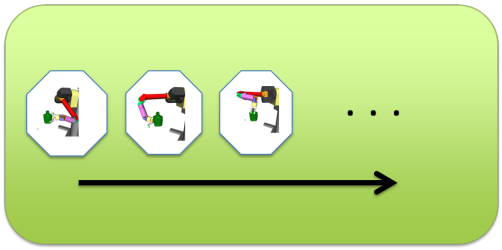
Multiple Prediction in Robotics

Lecturer: Drew Bagnell Scribes: Sasanka Nagavalli, Jane Sun, Xiao Xuesu, Ke Sun, Weikun Zhen

1 Introduction

So far, we have primarily considered scenarios where we make a single prediction from a set of possible options. For example, we might want to predict the most likely label for an object given a set of features. However, there are many situations in which it is necessary to make multiple predictions or choose the best subset from set of possible options. For example, consider the case of robotic grasping, which is often performed by first producing a library of grasps (e.g. by demonstration) and searching the library for a feasible grasp based on sensing information at run time. Another example is trajectory optimization, which is often performed by first producing a trajectory library (e.g. set of dynamically feasible motion primitives), picking a particular seed trajectory from the library, applying local trajectory optimization to ensure a collision-free path and iterating with a new seed trajectory from the library if a collision cannot be avoided using the chosen seed trajectory. In both of these scenarios, consider the following question: what are the particular grasps or trajectories that should be stored in the library and in what order should they be accessed? Other problems that present a similar challenge include feature selection and summarizing documents.

(a) Set of pre-computed grasps



(b) Evaluation of the grasps

2 Content and Order of Items in Library

Objective 1: Content - Predict successful actions.

Objective 2: Order - Predict sequences such that the average search depth for successful action can be minimized. This is a "min-sum submodular set cover" problem.

3 Submodular Set Function

3.1 Definition

Given a set $\mathcal{S}$, the power set (i.e. set of all subsets) of $\mathcal{S}$ is denoted $2^{\mathcal{S}}$. Consider a set function $F : 2^{\mathcal{S}} \to \mathbb{R}^+$ that maps from the power set of $\mathcal{S}$ to a positive real value. This function is called submodular if for every $X \subseteq Y \subseteq \mathcal{S}$ and $z \in \mathcal{S} \setminus Y$, the following condition is satisfied:

$$F(X \cup \{z\}) - F(X) \geq F(Y \cup \{z\}) - F(Y) \tag{1}$$

Submodularity captures the idea of diminishing returns: there is more incremental benefit when adding an item to a smaller subset.

3.2 Monotone

A submodular function F is monotone if $\forall X \subseteq Y \subseteq \mathcal{S} : F(X) \leq F(Y)$. For the remainder of the discussion, assume that when we say submodular, we also mean monotone.

3.3 Maximizing Submodular Functions - "Greed is Good"

Maximizing submodular functions is NP-hard. However, it has been shown that greedy algorithms provide a $\left(1 - \frac{1}{e}\right)$ approximation to the optimal solution. Specifically, for the problem of choosing a subset of size k from $\mathcal{S}$ to maximize the submodular function F, if the optimal solution is $L_k^* \subseteq \mathcal{S}$ with $|L_k^*| = k$, then it can be shown that a greedy algorithm will select $L_k \subseteq \mathcal{S}$ with $|L_k| = k$ such that the following is true.

$$F(L_k) \geq \left(1 - \frac{1}{e}\right) F(L_k^*) \tag{2}$$

If $F(L)$ measures the utility of set L, then applying the greedy algorithm will ensure that the utility of our selected set L_k will be no worse than $\left(1 - \frac{1}{e}\right) \approx 63\%$ of the utility of the optimal set L_k^*. But the formula only applies when we are able to evaluate utility function F.

4 Applying Submodularity in Trajectory Library Generation

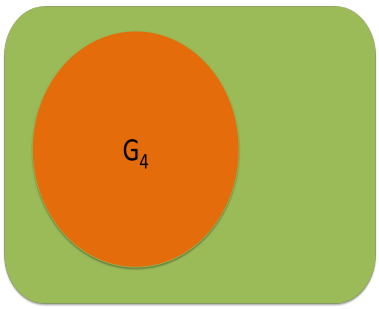
Given a set of trajectories $\mathcal{T} = \{T_1, T_2, \ldots\}$, a set of environments $\mathcal{E} = \{E_1, E_2, \ldots\}$ and some data showing in which environments $\mathcal{E}_{T_i} \subseteq \mathcal{E}$ each trajectory T_i succeeds, we want to form a library $\mathcal{L} \subseteq \mathcal{T}$ of k trajectories (i.e. $|\mathcal{L}| = k$) with the maximum utility. The utility of our library $F(\mathcal{L})$ is the number of environments which can be successfully handled by the trajectories in the library.

$$F(\mathcal{L}) = \left| \bigcup_{T_i \in \mathcal{L}} \mathcal{E}_{T_i} \right| \tag{3}$$

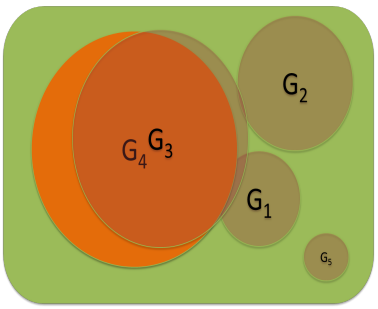
The utility function F is submodular, so we can use a greedy algorithm to add each trajectory to our library. Beginning with an empty library ($\mathcal{L} = \varnothing$), the greedy algorithm proceeds as follows.

	G_1	G_2	G_3	G_4	G_5
ENV_1	0	1	0	1	0
ENV_2	1	0	0	1	0
ENV_3	0	0	0	0	1
ENV_4	0	0	0	1	0
ENV_5	1	0	0	0	1
ENV_6	0	0	0	0	1
ENV_7	0	1	1	0	0
ENV_8	1	0	1	1	0
Avg. Benefit	0.38	0.25	0.25	0.5	0.38

(a) A table showing the performances of five grasps



(b) Our first best choice is S = {G4}



(c) The subsequent choices will be chosen according how well it performs and how little overlapping it has with previous best choice, S = {G4, G2, G1, G5, G3}

Figure 2: Greedy Training Example

1. Find the trajectory $T^* \in \mathcal{T}$ which results in the maximum utility when added to the current library: $T^* \leftarrow \arg\max_{T_i \in \mathcal{T}} F(\mathcal{L} \cup \{T_i\})$
2. Add T^* to the library: $\mathcal{L} \leftarrow \mathcal{L} \cup \{T^*\}$
3. If the library size has not reached the budget (i.e. $|\mathcal{L}| < k$), then repeat from (1). Otherwise, we are done.

Since F is submodular, the generated library $\mathcal{L}$ will succeed in at least $\left(1 - \frac{1}{e}\right) \approx 63\%$ as many environments as the optimal trajectory library $\mathcal{L}^*$, but not necessarily the same subset of environments.

5 Challenges in a Contextual Learning Setting

In a contextual learning setting, the utility function F can only be evaluated on training data. However, we want to learn a predictor that genalizes and generates a list with high utility for new test problems. For example, from a set of known successful seeds in training environments, we want to generate a set of currently unknown successful seeds for test environments.

In traditional machine learning, we provide a single best guess. How do we extend this notion to lists?

1. "Model-based" Approach
   (a) Learn a submodular function $F(\mathcal{S}; x)$ parameterized by features x.
   (b) Approximately optimize it (e.g. using the greedy algorithm).
   (c) Use linear combinations of submodular functions, which are also guaranteed to be submodular.
2. "Model-free" Approach

(a) Learn to predict actions to compete / mimic greedy algorithm.

(b) Approach is a reduction. Reduce the problem of list prediction to a set of simple classification problems.

(c) ConSeqOpt

5.1 Contextual Sequence Optimization

Transform sequence prediction into sequence of classification or regression problems using reduction, meaning doing well on the classification or regression implies doing well on the prediction. The method accounts for features of the environment and it is within constant factor of the (intractable) problem of predicting from the powerset of sequences of classifiers. Refer to Theorem 3 in Contextual Sequence Prediction via Submodular Function Optimization, Dey, Liu, Hebert, Bagnell, RSS 2012.

5.2 Submodular Contextual Policy

Submodular Contextual Policy (SCP) trains single policy to mimic greedy on training data. It generalizes across positions which leads it to learns more efficiently. It requires learning iteratively. See the lecture slides below for illustration.

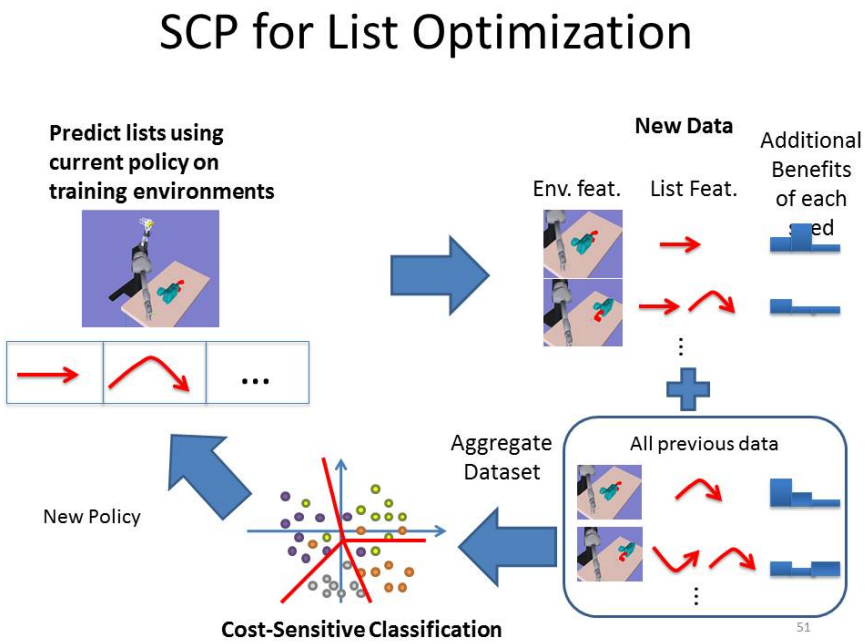


Figure 3: SCP for list optimization.

Algorithm 2 Submodular Contextual Policy (SCP) Algorithm.

Input: Set of items $\mathcal{S}$, policy class $\tilde{\Pi}$, length m of list we construct, length k of best list we compete against.
Pick initial policy π_1 (or distribution over policies)
for $t = 1$ **to** T **do**
Observe features of a sampled state $x_t \sim D$ (e.g. features of user/document)
Construct list L_t of m items using π_t with features of x_t (or by sampling a policy for each position if π_t is a distribution over policies).
Define m new cost-sensitive classification examples $\{(v_{ti}, c_{ti}, w_{ti})\}_{i=1}^{m}$ where:

1. v_{ti} is the feature vector of state x_t and list $L_{t,i-1}$
2. c_{ti} is the cost vector such that $\forall s \in \mathcal{S}$: $c_{ti}(s) = \max_{s' \in \mathcal{S}} b(s'|L_{t,i-1}, x_t) - b(s|L_{t,i-1}, x_t)$
3. $w_{ti} = (1 - 1/k)^{m-i}$ is the weight of this example

$\pi_{t+1} = \text{UPDATE}(\pi_t, \{(v_{ti}, c_{ti}, w_{ti})\}_{i=1}^{m})$
end for
return π_{T+1}

52

Figure 4: The SCP algorithm.