Statistical Techniques in Robotics 16-831 - The Robotics Institute

Lecture 7: List Prediction

Jiaji Zhou

1 Introduction

List prediction problems concern with predicting a *set* or *list* of options from a candidate pool to maximize some utility function. Perhaps the most common list prediction problem is the *multiple guess* problem, in which we are able to choose more than one options such that at least one option achieves a desired goal. A good example would be ads recommendation where we are interested in displaying three ads on a user's phone that hopefully the user will click at least one of it.

In this lecture, we will look at provable yet simple (easy-to-implement) algorithms to tackle such list prediction problems by exploiting problem structures and analysis tools from online learning.

2 Problem Formulation

Let S denote the set of possible items to choose from, our objective is to construct a list of items $L \subseteq S$ that maximizes a reward function f. Usually, there is a budget k on the size of L, i.e., $|L| \leq k$.

Now let's consider the simplest case where the items' utility/benefit is independent from each other, meaning there is a fixed assigned non-negative benefit value $b(s)$ for item s and the existence of other items in the current list makes no influence. That is, $f(L \oplus s) - f(L) = b(s)$ for arbitrary list L that does not already contain s. In this case, f is modular and we can construct the optimal list L^* by simply picking the top k items sorted by $b(s)$.

2.1 Submodular Utility Function

In many real world application domains when utilities of items overlap, f is not just a modular function, but rather a monotone submodular function that captures the diminishing return property:

1. Monotonicity: $f(L_1) \leq f(L_1 \oplus L_2)$ and $f(L_2) \leq f(L_2 \oplus L_1)$ for any list L_1 and L_2;

2. Submodularity (dimishing return): $b(s|L_1) = f(L_1 \oplus s) - f(L_1) \geq f(L_1 \oplus L_2 \oplus s) - f(L_1 \oplus L_2) = b(s|L_1 \oplus L_2)$ for any list L_1 and L_2 and s.

An example submodular function is the *"multiple guess"* function $f(L;T) = \min(1, |L \cap T|)$, where T is a set of valid options. You can see that if L already contains an valid element in T, then adding more valid guesses are not increasing the function value.

It turns out that exact maximization of a monotone submodular function under a budget constraint is NP-hard. Surprisingly, however, a greedy maximization that sequentially adds an item to the list based on marginal benefit achieves near-optimality.

Theorem 1. *Denote as L^G a size-k list constructed by a sequential greedy rule: $L^G_{i+1} = L^G_i \oplus \arg\max_{s \in S \setminus L^G_i} b(s|L^G_i)$, then $f(L^G) \geq (1-1/e)f(L^*)$ holds for monotone submodular function f, where L^* is the optimum size-k list that maximizes f.*

We will leave the proof as a homework theory question.

3 Online Submodular Maximization

Consider an online setting where we select lists of k advertisements L_t for an incoming stream of users x_t, each with an associated utility function f_t (e.g., multiple guess function where each user has a specific set of interested advertisements). Our goal is to design an online algorithm that is near-optimal (in particular the greedy ratio $1-1/e \approx 0.63$) with respect to the best fixed list of k advertisement L^*, i.e., $\sum_{t=1}^{T} f_t(L_t) \geq (1-1/e)\sum_{t=1}^{T} f_t(L^*)$. The online algorithm consists of k slots of learner π_i and the sequence L_t is the concatenatation of the predictions $L_t = \{\pi_1(x_t), \pi_2(x_t), \ldots, \pi_k(x_t)\}$. For ease of analysis, let's consider the context-free scenarios first, where each π_i consists of experts E that always predict/choose the same item regardless of difference in x_t, e.g., expert e_j always predicts element s_j. The proof can be readily extended to contextual case where e_j chooses elements in S differently when x_t varies. Algorithm 1 describes the pseudocode procedure. The following analysis is mainly adopted from [2] and an improved version of the online learning algorithm where we only need a single online learner that repeatedly makes predictions to form a list can be found in [1].

Lemma 1. *Let S be a set, and f a monotone submodular function defined over S. Let A and B be any size-k list. Denote A_i be the prefix-list of A up to position i. Denote b_i as the ith element in list B. $f(A) \geq (1-1/e)f(B) - \sum_{i=1}^{k} \varepsilon_i$, where $\varepsilon_i = f(A_{i-1} \oplus s_i^*) - f(A_i)$ and $s_i^* = \arg\max_{s \in S} b(s|A_{i-1})$.*

Proof. Let $\Delta_i = f(B) - f(A_{i-1})$, we have:

$$
\begin{aligned}
\Delta_i &\leq f(A_{i-1} \oplus B) - f(A_{i-1}) = \sum_{j=1}^{k} f(A_{i-1} \oplus B_j) - f(A_{i-1} \oplus B_{j-1}) \\
&\leq \sum_{j=1}^{k} f(A_{i-1} \oplus b_j) - f(A_{i-1}) \leq \sum_{j=1}^{k} \{f(A_{i-1} \oplus s_i^*) - f(A_i)) + (f(A_i) - f(A_{i-1})\} \\
&= \sum_{j=1}^{k} \{\varepsilon_i + (f(A_i) - f(B) + f(B) - f(A_{i-1})\} = \sum_{j=1}^{k} \{\varepsilon_i + (\Delta_i - \Delta_{i+1})\} \\
&= k(\Delta_i - \Delta_{i+1} + \varepsilon_i)
\end{aligned}
$$

Algorithm 1 List prediction algorithm with a sequence of online learners.

Input: Set of items S, length k of list to construct, a sequence of k online learners $\{\pi^1, \pi^2, \ldots, \pi^k\}$ with PREDICT and UPDATE functions.

for $t = 1$ **to** T **do**

  $L_t = \{\}$, Receive observation/feature x_t

  for $i = 1$ **to** k **do**

    Call online learner PREDICT()to append an item to list, i.e., $L_t = L_t \oplus \pi_t^i(x_t)$. (e.g. by sampling from online learner's internal distribution over items)

    For all $s \in S$: define loss $\ell_t(s) = \max_{s' \in S} f_t(L_{t,i-1} \oplus s') - f_t(L_{t,i-1} \oplus s)$

    Call online learner update with loss ℓ_t: UPDATE(ℓ_t)

  end for

end for

Rearrange the terms we get $\Delta_{i+1} \leq (1-1/k)\Delta_i + \varepsilon_i$. Recursively expand the Δ_i terms we get $f(B) - f(A) = \Delta_k \leq (1-1/k)^k \Delta_1 + \sum_{i=1}^k \varepsilon_i \leq f(B)/e + \sum_{i=1}^k \varepsilon_i$. □

Observe that if we let $f(\cdot) = \sum_{t=1}^T f_t(\cdot)$, $A = L_t$[1], and $B = L^*$, we get $\sum_{t=1}^T f_t(L_t) \geq (1-1/e)\sum_{t=1}^T f_t(L^*) - \sum_{i=1}^k \max_{s\in S} \sum_{t=1}^T (f_t(L_{t,i-1} \oplus s) - f_t(L_{t,i}))$.

The term $\hat{R}_i = \max_{s \in S} \sum_{t=1}^T f_t(L_{t,i-1} \oplus s) - \sum_{t=1}^T f_t(L_{t,i})$ should remind you something very similar to regret. In fact, let's define the loss function for the ith learner at round t as $c_{t,i}(s) = \max_{s' \in S} f_t(L_{t,i-1} \oplus s') - f_t(L_{t,i-1} \oplus s)$, then the regret for the ith learner is

$$
\begin{aligned}
R_i &= \sum_{t=1}^T c_{t,i}(\pi(x_t)) - \min_E \sum_{t=1}^T c_{t,i}(E(x_t)) \\
&= \min_{e \in E} \sum_{t=1}^T -f_t(L_{t,i-1} \oplus e(x_t)) - \sum_{t=1}^T f_t(L_{t,i}) \\
&= \max_{e \in E} \sum_{t=1}^T f_t(L_{t,i-1} \oplus e(x_t)) - \sum_{t=1}^T f_t(L_{t,i}) \\
&= \max_{s \in S} \sum_{t=1}^T f_t(L_{t,i-1} \oplus s) - \sum_{t=1}^T f_t(L_{t,i}) \\
&= \hat{R}_i
\end{aligned}
$$

To this point, we can conclude:

$$
\sum_{t=1}^T f_t(L_t) \geq (1-1/e)\sum_{t=1}^T f_t(L^*) - \sum_{i=1}^k R_i
$$

Suppose at each slot i, we are running a randomized weighted majority algorithm, then R_i goes sublinear in T, which implies the additive term of regret will approach zero and our average performance will be near-optimal with $(1-1/e)$ competitive ratio.

[1]Note that A here is not a list, $f(A) = \sum_{t=1}^T f_t(L_t)$, but the proof still holds.

References

[1] Stephane Ross, Jiaji Zhou, Yisong Yue, Debadeepta Dey, and Drew Bagnell. Learning policies for contextual submodular prediction. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1364–1372, 2013.

[2] Matthew Streeter and Daniel Golovin. An online algorithm for maximizing submodular functions. In *Advances in Neural Information Processing Systems*, pages 1577–1584, 2009.