Preference Elicitation Algorithms

Vagul Mahadevan, Declan Stacy, and Nathan Weatherly

University of Virginia

Problem Statement

In this project, we experiment with some algorithms for preference elicitation in interactive recommender systems. We have some set $I = \{x_1, x_2, \ldots, x_n\} \subset \mathbb{R}^d$ of items and a single user with (unknown) preferences. Their utility is given by $u(x) = \theta_*^T x$, where $\theta_* \in \mathbb{R}^d$. The goal is to figure out $x_* = \text{argmax}_{1 \leq i \leq n} u(x_i)$. We are allowed to ask the user questions like "what is your utility for item x" or "what is your preferred item from the set $\{x_1, \ldots, x_K\}$," and we wish to find x_* in the least number of interactions possible. We will tackle this problem in four different settings, and in each one, we will come up with a solution, implement our algorithm, and run simulations to report the performance of our algorithm. The four different cases are as follows:

1. The user is presented x_t and responds with their exact utility $u(x_t) = \theta_*^T x_t$.
2. The user is presented x_t and responds with $u(x_t) + \epsilon_t$ where ϵ_t are iid with distribution $N(0, \sigma^2)$ (for our simulations, we use $\sigma = 0.1$).
3. The user is presented with $x_{1,t}$ and $x_{2,t}$ and responds with their most preferred item $x_{i,t}$.
4. The user is presented with $K \geq 2$ items $\{x_{1,t}, \ldots, x_{K,t}\}$ and responds with their most preferred item $x_{i,t}$.

Environment Modeling

We will assume that the set of items I is generated by iid samples from a continuous distribution on $\mathbb{R}^d$. In our simulations, we choose them to be uniformly distributed on the sphere $S^{d-1} = \{x \in \mathbb{R}^d \mid ||x|| = 1\}$ (where $||-||$ always denotes the L^2 norm). We may assume that θ_* satisfies $||\theta_*|| = 1$, as scaling the user's utility function does not affect their preferences. We will also assume $n \geq d$ (n is the number of items).

Case 1

By asking the user to provide their utility of d items, we can calculate θ_* exactly. This is because, for each item x_i, we have $\theta_*^\top x_i = y_i$, and we can form a d by $d+1$ augmented matrix to represent this system of equations (d equations in d unknowns). Because the x_i are all drawn independently from a continuous distribution, the probability that they will be linearly dependent is 0, and so we can use row reduction to solve the system of equations and find an exact value for θ_*. Then, we can loop through I and choose the item that maximizes $\theta_*^\top x_i$. This will find the best item in exactly d interactions, regardless of the number of items.

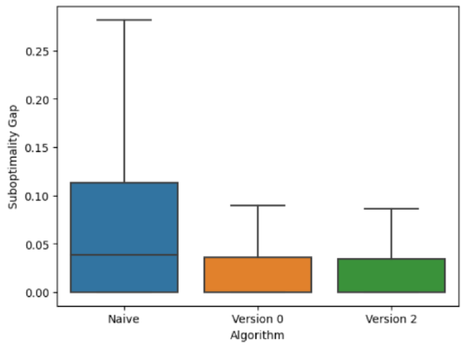
1. Form a d by d matrix A whose rows consist of $x_1^T, \ldots, x_d^T$, the first d items in I
2. For $1 \leq i \leq d$, get the users utility y_i for item i and store them in a vector y
3. Let θ be the solution to $A\theta = y$
4. Return $\text{argmax}_{x \in I} \theta^T x$

Case 2 Algorithm

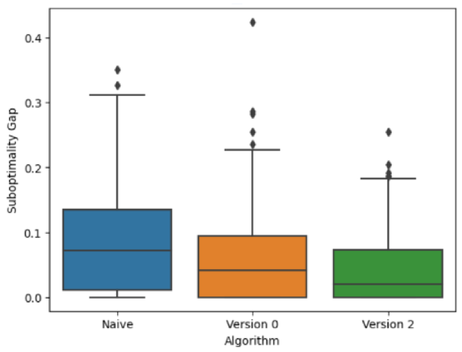
We developed the following algorithm with parameters ϵ (how close we want to be) and β (representing how many standard deviations to use for calculating e). The algorithm is essentially a combination of linear regression and the UCB algorithm discussed in class. We also implemented a naive algorithm that sampled every item and chose the one with the highest noisy utility.

1. Choose items $x_1, \ldots, x_d$ iteratively, where at each step i, x_i is chosen to maximize $||x_i||^2 - \sum_{j<i} (x_i^T \hat{x}_j)^2$ and $\hat{x}_i := \frac{x_i - \sum_{j<i} x_i^T \hat{x}_j}{||x_i - \sum_{j<i} x_i^T \hat{x}_j||}$
2. Initialize a $d \times d$ matrix A with rows $x_1^T, \ldots, x_d^T$ and let $C = A^T A$ and $B = C^{-1}$
3. Get the users utility $y_1, \ldots, y_d$ for the d items chosen. Initialize a column vector y consisting of the y_is.
4. Initialize $e = \infty$.
5. While $e > \epsilon$, do the following:
6. 1. Update $\theta = BA^T y$
   2. Compute $x = \text{argmax}_{x \in I} \theta^T x$
   3. For $x' \in I, x' \neq x$, compute $a_{x'} = B(x' - x)$ and $e_{x'} = \theta^T(x' - x) + \beta\sigma\sqrt{(x' - x)^T a_{x'}}$
   4. Update $e = \max_{x' \in I, x' \neq x} e_{x'}$ and let $x' = \text{argmax}_{x' \in I, x' \neq x} e_{x'}$
   5. Find $x'' = \text{argmax}_{x'' \in I} x''^T B(x' - x)$
   6. Get the user's utility y' for item x''.
   7. Update y by adding y' as an entry.
   8. Update A by adding x''^T as a row.
   9. Update C by adding $x'' x''^T$ to it.
   10. Update B by subtracting $\frac{Bx'' x''^T B}{1 + x''^T B x''}$
7. Return x

Case 2 Testing With $n = 500$, $d = 20$



Case 2 Testing With $n = 1000$, $d = 50$

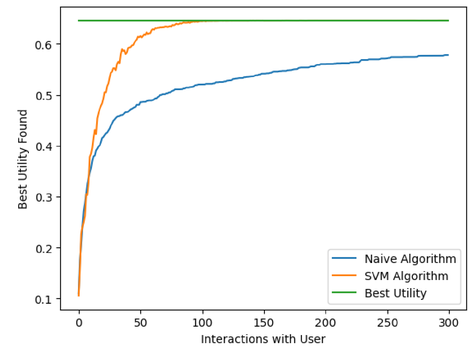


Case 3/4 Algorithm

We developed an algorithm that utilizes an SVM to approximate θ_* at each step and chooses the next K items based on their orthogonality to the approximation. We also implemented a naive approach that kept track of the "best" item observed and compared it with $K - 1$ new items at each step.

1. Initialize a list of items V to be empty
2. Choose the first K items $x_1, \ldots, x_K$ and ask the user for their preferred item x_i.
3. For $1 \leq j < K$, let $x = x_i - x_{i+j}$ and add $\frac{x}{||x||}$ to V
4. Run a separable case of SVM with the points in L labeled as positive and the points in $-L$ labeled as negative to get a linear classifier $\theta^T x \geq 0$
5. For each item x, compute $\theta^T x$ and store in a list L
6. Sort L and find the index i which minimizes $L[i + K - 1] - L[i]$
7. Go back to step 2 using items corresponding to the K elements $L[i], \ldots, L[i + K - 1]$
8. After θ has not changed more than ϵ since the last iteration, return $\text{argmax}_{x \in I} \theta^T x$

Case 3 Testing With $n = 1000$, $d = 20$



Case 4 Testing With $n = 1000$, $d = 20$, $K = 5$