Collective Recommendations

Haukur Jónsson
haukurpalljonsson@gmail.com and Silvan Hungerbühler
silvan.hungerbuehler@bluewin.ch and Grzegorz Lisowski
grzegorz.adam.lisowski@gmail.com and Max Rapp
maxgrapp@gmail.com

Abstract

This is a placeholder for the glorious abstract yet to come.

1 Introduction

Some questions we would like to ask Ulle:

- What's his opinion of Schultze? What is his opinion, in general, about the rules we picked, especially wrt fact that we abandoned multiwinner rules which are solely constrained by cardinality?
- Proofs vs. Simulations
- What's his opinion on the desirable properties we have come up with?

2 Desirables & Dimensions of Performance

The following are a bunch of things we would like a recommendation mechanism to do. We will describe these desiderata formally and describe their relation and relevance to our goal stated in the introduction.

2.1 Desirables

Regret Mbinimization
Minority Consistency
Cost Distribution
There are two polar extremes here: One one end one could conceive a rule that picks chunky, expensive items (long, in-depth articles); on the other end rules that pick snacky, cheap items (Vice). In principle, any distribution over article prices is conceivable.

A bunch of more rules we yet have to get from the article.

3 Recommendation Rules

Here come a bunch of voting rules that we think will perform well with respect to above desiderata.

3.1 K-plurality rule

The *k-plurality rule* with $k < |A|$ is a *positional scoring rule* with the same scoring vector as the normal *plurality rule*, $(1, 0, \ldots, 0)$, but instead of electing the alternative(s) with the highest score it elects the alternative(s) with the highest score, if the number of winners is strictly less than k then the alternative(s) with the second highest score is elected. This is done until $|W| \geq k$. If $|W| > k$ then a tie-breaker should be applied on the last iteration.

Computationally feasible.

3.2 Schulze rule

Has many nice properties and is closely related to the 20% minimal requirement.

Computationally feasible.

3.3 θ-rule

3.4 Extended Θ-Smith

The Θ-Smith set is the smallest non-empty set of candidates s.t. each member of the set defeats every other member outside the set in $\Theta\%$ of cases.

The Θ-Smith set is a Condorcet extension.

A suggestion of an algorithm which selects a "good" Θ-Smith set. Compute the Smith set for $\Theta = 50$ (the normal Smith set), check if it satisfies the cost restricitons. Second, check the upper-bound by finding the Smith set of $\Theta = 100$, if it satisfies the cost restriction, select it. Depending on the case the cost restriction was not satsfied we need to iterate over the interval in which the cost restriction break, start iterating from $\Theta = 50$ and increase/decrease Θ by $(100 - 50)/2 = 25$ or $(50 - 0)/2 = 25$ until a set is found which satisfies the cost restrictions (when found, there might be multiple.).

Computationally feasible.

3.5 Utility Optimization

We could also cast the problem of coming up with the best recommendation, given the items' costs as well as the budget, as a maximization problem. What we try to maximize is the sum of all the consumers' values by choosing W, where a consumer's value of seeing some item in W is represented by a vector $\mathbf{s}$. For each consumer we only count the value from the news items that are actually in the recommended set:

$$\max_W \sum_{j=1}^{n} \sum_{i=1}^{m} [a_i \in W] V(\mathbf{R}, n_j, a_i)$$

Of course, Equation 1 is trivially solved by $W = A$. But the interest in solving it comes from adding the budget constraint.

$$\max_{W} \sum_{j=1}^{n} \sum_{i=1}^{m} [a_i \in W] V(\mathbf{R}, n_j, a_i) subject to \sum_{a_i \in W} C(a_i) \leq B$$

This problem, however, is the well-studied *0-1 Knapsack* problem. Given a set of items, each with a weight and a value, what is the most valuable knapsack that you can put together without exceeding its maximal load. In our case the weight corresponds to an article's cost, the value to the sum of all consumers' value if the item is included in W, and the maximal load corresponds to our budget B.

Although the optimization problem is NP-hard, there exists pseudo-polynomial algorithms using dynamic programming as well as polynomial-time approximation schemes.

4 Simulations

4.1 Method

4.2 Results

4.3 Discussion

4.4 Proofs

5 Conclusion