Stochastic matching in simple settings

Someone

Abstract

Motivated by applications like kidney exchange, recommendation system and online dating, we study a matching problem with query-commit process. In this setting, a centralized system allocates agents into pairs. If a pair of agents accept their assignments, they are matched and leave the market. Otherwise, they will stay in the market. Most previous work focus on giving worst case guarantee on different settings. However, surve

1 Introduction

2 The setting

Our problem can be modeled in an undirected graph $G = (V, E)$, $V = \{v_1, v_2, \dots, v_n\}$ denotes the set of agents. $e_{ij} \in E$ denotes an edge between v_i and v_j. The goal of the matching is to maximize cardinality. Each time, the system prob a pair of agents v_i and v_j with an edge e_{ij} connecting them. (1) If v_i and v_j accept to match with each other with p probability, both of the two agents are removed from the market. (2) If v_i and v_j do not accept with $1 - p$ probability, the system knows that they cannot match and removes e_{ij} from E.

3 ESD algorithm

To maximize the number of matched agents, we apply edge serial dictatorship algorithm (ESD). The outline is described In Algorithm 1.

Algorithm 1 is quite straightforward. The system just picks edges one by one in some order (denoted by S) and try to match the current edge. Our goal is to maximize the expected size of the output set A.

We use ESD for the following reasons.

1. The solution of ESP has a size of $O(|E|)$, which equals to the size of input. The executor of the matching only needs to use O(1) computation for each prob. These two properties make the solution perfect to be used as handouts. In other words, the order can be generated by a computer and any people can execute the matching without a computer. This property makes ESP quite practical. For example, when matching roommates among a class of students, the teacher can input a graph showing the compatibilities between students and an estimated acceptance ratio into the program. The program outputs an order of probs. The teacher can just follow the order to query pairs of students.
2. The optimal solution can be solved by Optimization 1 in small graphs and can be approximated in larger graphs.

Algorithm 1 ESD algorithm

```
Require: A sorted array of $E$, denoted by $S = s_1, s_2, \dots, s_m$ and the set of vertices $V$.
Ensure: A set of disjoint edges $A$, such that each two agents connected by an edge in $A$ agree to match.
 1: $A = \emptyset$
 2: for do$i = 1, 2, \dots, m$
 3:     Let $v_x$ and $v_y$ be the two agents connected by $s_i$
 4:     if then$v_x \in V$ and $v_y \in V$
 5:         Prob $v_x$ and $v_y$ whether they agree to match with each other.
 6:         if B thenoth $v_x$ and $v_y$ accept to match
 7:             Put $s_i$ into $A$
 8:             Remove $v_x$ and $v_y$ from $V$
 9:         end if
10:     end if
11: end for
12: Outputs $A$ as the set of matched edges.
```

In Algorithm 1, The generating of array S is not mentioned. Even though we find the S that can maximize the expected size of A, the current matching policy may not be optimal. However, ESD has the following advantages that motivates us to find a good policy based on it.

The most important advantage is that given S the expected size of A can be solved. It is not natural for other policies for optimizing stochastic matching. The expected cardinality of the matching can be solved as follows.

$$E_S(|A|) = 2 \sum_{i=1}^{m} p * \prod_{j=1}^{i-1} (1 - p)^{\delta(s_i, s_j)} \tag{1}$$

$\delta(s_i, s_j)$ is an indicator, which equals to 1 if s_i and s_j are disjoint, otherwise equals to 0.

3.1 Optimizing the order

A natural question is how to decide S to maximize cardinality. The most straightforward way is by enumerating all the permutations and find one permutation with the best payoff (the expected number of matched agents). In this way, the computational complexity is $O(|E|! * |E|^2)$. It can only handle a graph with up to 12 or 13 edges.

To solve the optimal order of edges for lager graphs, we model it as an integer linear program (ILP). With the ILP we proposed, we can easily handle a graph with more than 20 edges. For larger cases, we can approximate the optimal payoff by restricting the maximum iteration of the ILP.

The ILP problem is listed as Optimization 1. Renumber all the elements in E as $e_1, e_2, \ldots, e_m$.

x_{ij} is the indicator for whether s_i is the j-th edge in .

Optimization 1

$$\text{Maximize} p_i \tag{2}$$