Learning Market Parameters using Aggregate Demand Queries

Xiaohui Bei
Nanyang Technological University
Singapore

Wei Chen
Microsoft Research
Beijing, China

Jugal Garg
MPI für Informatik
Saarbrücken, Germany

Martin Hoefer
MPI für Informatik
Saarbrücken, Germany

Xiaoming Sun
China Academy of Sciences
Beijing, China

Abstract

We study efficient algorithms for a natural learning problem in markets. There is one seller with m divisible goods and n buyers with unknown individual utility functions and budgets of money. The seller can repeatedly announce prices and observe aggregate demand bundles requested by the buyers. The goal of the seller is to learn the utility functions and budgets of the buyers. Our scenario falls into the classic domain of "revealed preference" analysis. Problems with revealed preference have recently started to attract increased interest in computer science due to their fundamental nature in understanding customer behavior in electronic markets. The goal of revealed preference analysis is to observe rational agent behavior, to explain it using a suitable model for the utility functions, and to predict future agent behavior. Our results are the first polynomial-time algorithms to learn utility and budget parameters via revealed preference queries in classic Fisher markets with multiple buyers. Our analysis concentrates on linear, CES, and Leontief markets, which are the most prominent classes studied in the literature. Some of our results extend to general Arrow-Debreu exchange markets.

1 Introduction

In this paper, we study the following learning problem. There is one seller with m divisible goods, and n buyers with individual utility functions and budgets of money. The seller can repeatedly announce prices, and the buyers request demands – bundles of goods that they can afford and that maximize their utility. The goal of the seller is to learn the utility functions and budgets of the buyers.

This scenario represents a natural learning problem with "revealed preference" in the classic Fisher market model. Revealed preference analysis has a long history in the economics literature (Samuelson 1948; Afriat 1967; Varian 2005). Here a number of observations of agent behavior are either generated by a stochastic process or actively obtained by a learner through queries. The goal is to explain the observed behavior using a suitable model for their utility function and to predict future agent behavior. More recent work has also studied different goals of the learner, such as revenue maximization (see, e.g., (Amin et al. 2015; Roth, Ullman, and Wu 2015) and our discussion of related work below). The algorithmic issues in this domain are starting to attract interest due to numerous applications of problems with revealed preference in online markets. With the advent of big data technology, retailers like Amazon or Tesco collect a large amount of sales data and have a vital interest in gaining insight into the underlying preferences of customers. Such information is even more critical for specialized businesses who rely more heavily on a particular customer base.

Similar to recent work (Balcan et al. 2014), we concentrate on query learning where a learner can repeatedly query a market (without competing sellers) by setting prices for all its goods. Upon a price query, the learner receives as feedback for each good the sum of demands of all buyers. In contrast to previous work, we advance the study of algorithms for revealed preference beyond a single buyer with a single utility function. We assume the more realistic case of multiple buyers and aggregate demand feedback. In fact, our query model is exactly the same as the classic tâtonnement model to find a market equilibrium in economics (Walras 1874; Codenotti, Pemmaraju, and Varadarajan 2005; Cole and Fleischer 2008; Bei, Garg, and Hoefer 2015). However, our goal is not to find an equilibrium but to learn the unknown buyer utilities via revealed preference feedback.

We study query complexity in Fisher markets with divisible goods where buyers have budgets of money that they can spend to purchase a bundle of goods to optimize their utility function. We study linear, constant elasticity of substitution (CES), and Leontief utilities, which are the most central utility functions in the market literature. We assume that all parameters in these functions and the money budgets are given by L-bit integers, and the seller knows the bound L. While the seller knows the number n of buyers, it does not know their utility parameters or budgets. We show how to learn the set of utility functions and the amount of money associated with buyers that have this function. Note that since the feedback is the sum of demands, it is impossible to distinguish buyer identities and to learn, e.g., that a particular utility function belongs to buyer number 5.

More generally, we also extend some of our results to Arrow-Debreu exchange markets, where agents bring endowments of goods instead of money. When the learner sets prices, agents exchange their endowment into money and

then request a demand bundle of goods that they can afford and that maximizes the utility. The goal is to learn endowments and utilities from aggregate demand feedback.

We present the first algorithms to learn utility function parameters and endowments in Fisher and Arrow-Debreu exchange markets. Our algorithms require only a number of queries that is polynomial in n, m and L. Before we describe our contributions in detail, let us formally introduce the model and some basic technical terms.

1.1 Preliminaries

We consider a *Fisher market* with a set N of n buyers and a single seller that has a set M of m divisible goods. The seller has a fixed amount of supply of each good, and w.l.o.g. we assume the supply for each good is normalized to 1. Each buyer i has an initial endowment b_i of *money* or *budget*. Buyer i also has a *utility function* $u_i : \mathbb{R}_+^m \mapsto \mathbb{R}$. The utility functions studied are of the form

$$u_i(\mathbf{x}_i) = \left(\sum_j (a_{ij} x_{ij})^\rho \right)^{1/\rho},$$

where $a_{ij} \geq 0$ and $1 \geq \rho \geq -\infty$ are fixed parameters, and $\mathbf{x}_i = (x_{i1}, \ldots, x_{im})$ is a bundle of goods. In this paper, we consider linear ($\rho = 1$), Leontief ($\rho = -\infty$), and general CES ($1 > \rho > -\infty$) utility functions.

Given a price vector $\mathbf{p} = (p_1, p_2, \ldots, p_m)$, a *demand* of buyer i is a bundle of goods $\mathbf{x}_i$ that the buyer can afford with its endowment and that maximizes its utility. More formally, a demand is $\mathbf{x}_i = \arg\max\{u_i(\mathbf{x}_i) \mid \sum_j x_{ij} p_j \leq b_i, \text{ and } \forall j : x_{ij} \geq 0\}$. The *aggregated demand* of good j is defined as $Z_j = \sum_i x_{ij}$ where $\mathbf{x}_i$ is a demand of buyer i.

Fisher markets are a special class of more general *(Arrow-Debreu) exchange markets*. In such a market, agents are simultaneously sellers and buyers. Instead of money, each agent has an initial endowment $\mathbf{e}_i = (e_{i1}, \ldots, e_{im})$ with $e_{ij} \geq 0$ for all goods. Without loss of generality, we assume $\sum_i e_{ij} = 1$ for every j, i.e. the total supply of each good remains 1. For price vector $\mathbf{p}$, the value of the endowment of agent i becomes $b_i = \sum_j e_{ij} p_j$, and the agent will request a demand bundle as defined above. Formally, if we consider "money" as a good, budgets can be seen as endowments and Fisher markets as a special case of exchange markets.

In an *unknown market*, the utility function u_i as well as the initial endowment for each agent i are unknown. In each round, one can query the market by proposing a price vector $(p_1, p_2, \ldots, p_m)$. Every agent i then picks a demand bundle, and one observes the aggregated demand Z_j of each good j.

We concentrate on a revealed preference problem and study how to learn the utility functions and endowments of all agents. We assume that all utility parameters $(a_{ij})_{i,j}$ are non-negative L-bit integers. Similarly, all budgets b_i in Fisher markets are non-negative L-bit integers. For exchange markets, the endowments e_{ij} are rational numbers of two non-negative L-bit integers.

When querying the market with any price vector $\mathbf{p}$, an oracle returns us the aggregated demand vector $\mathbf{Z} = (Z_1, \ldots, Z_m)$ with regard to the queried price vector. Since $\mathbf{p}$ is known, we can calculate the money spent on each good $\mathbf{p} \cdot \mathbf{Z} = (p_1 Z_1, \ldots, p_m Z_m)$. We will find it convenient to assume that the oracle directly returns $Q(\mathbf{p}) = \mathbf{p} \cdot \mathbf{Z}$, where we use $Q_j(\mathbf{p}) = p_j Z_j$ to denote the money spent on j.

Initially we only know L, the number of goods m, and the number of agents n. Our goal is to learn all the utility parameters and endowments (or budgets) using as few number of queries to the oracle as possible. For many of the markets discussed here, demand bundles are unique. Only when we discuss linear utility functions, agents might have more than one demand bundle to choose from. We here make no particular assumptions on how the oracle breaks these ties. Our queries will always ensure that there is a unique demand bundle for every agent, even for linear markets.

1.2 Our Contribution

We present efficient algorithms to learn market parameters in Fisher and exchange markets. Aggregate demands are the most natural and most prominent approach to query feedback in markets with multiple agents. However, some inherent ambiguities cannot be resolved with such queries. For an agent with linear utility, we would receive the same feedback if there were several agents with the same utility and the same total budget. Hence, in linear markets it is impossible to learn the exact number of agents, and our algorithms for these markets do not take n as input. The learnable description of the market becomes a set of utility functions and the total budget of agents with each utility function. For linear Fisher markets, we design an algorithm that computes such a description using $O(n^2 m + nmL)$ queries. It does not use the number n of buyers as input, but the running time can be bounded in n since the number of parameters required to describe the market are upper bounded by $n(m+1)$. We use a dynamic programming approach that can be extended to work even for exchange markets, where we also learn the initial endowments of all agents. The algorithm uses $O(n^2 m^2 + nm^2 L)$ queries and returns the set of utility functions present in the market, and for each utility function the total endowment of all agents that have this utility.

For Fisher markets with Leontief utilities, our algorithm uses efficient algorithms for the factorization of polynomials (Kaltofen 1982; Lenstra, Lenstra, and Lovász 1982) and uniquely determines all utility parameters and budgets using $O(n^3 m)$ queries. It can easily be adapted to also work for Fisher markets with general CES utilities using $O(n^3 m)$ queries. It is an interesting open problem to adapt it to exchange markets with Leontief and general CES utilities.

1.3 Related Work

Learning from revealed preferences is a classical approach in economics to learn about consumer preferences from buying patterns. There is a large body of work on this topic, which was first introduced by Samuelson (1948). It has been studied broadly in two frameworks. One is the query learning framework, which the current work belongs to. It allows to choose prices and observe the purchase decisions of the buyer. The goal here is to *exactly* learn the utility function of the agent. The performance of the learner is measured in terms of the number of queries needed. Recently,

Balcan et al. (2014) gave efficient learning algorithms for a general class of utility functions including linear, separable piecewise linear concave, and Leontief. Amin et al. (2015) propose efficient algorithms for a profit maximization problem of a merchant that queries a single buyer with a linear utility function. In another related work, Roth, Ullman, and Wu (2015) study profit maximization of a leader in a Stackelberg game when the follower's utility function is unknown. Note that in this paper, we do not consider profit maximization but instead focus on learning the utility functions. Clearly, once these are determined, a variety of different objectives can be considered for optimization. For example, one can construct online learning scenarios using our algorithms for linear utilities similar as in (Amin et al. 2015).

Note that all these works deal with a single agent only. In our case, there are multiple agents but we can query only aggregate demand of these agents. Clearly, if we assume that individual demand bundles of each agent can be distinguished, there is a trivial solution by applying the algorithms from (Balcan et al. 2014) for each agent sequentially. In contrast, aggregate demand does not rely on such strong assumptions and represents the most prominent approach to query markets, e.g., in natural price update dynamics and general equilibrium theory (Walras 1874; Mas-Colell, Whinston, and Green 1995). Our goal here is to learn the underlying utility function and budget of each agent. With aggregate demands, learning becomes more challenging, and we develop new tools and insights.

Another framework is statistical (PAC) learning, in which we are given what a buyer bought at prices sampled from an unknown distribution. The goal is to learn the underlying utility function in a way to accurately or approximately predict future demands when faced with a price from the same distribution. There are a number of approaches within this framework (Beigman and Vohra 2006; Zadimoghaddam and Roth 2012; Balcan et al. 2014). Our scenario is similar to recent results for a single agent (Balcan et al. 2014). However, due to aggregation of demand over multiple agents, it does not fall into the framework of D-dimensional linear hypotheses, and thus the statistical framework and the multi-class support vector machine of Daniely and Shalev-Shwartz (2014) are not applicable here.

Apart from these, revealed preference analysis has been applied in several other recent works (Blum, Mansour, and Morgenstern 2015b; 2015a; Jabbari et al. 2015) in auctions, mechanism design, revenue maximization, etc.

We study learning utilities in markets with CES utility functions. They were introduced in (Solow 1956; Dickinson 1954) and prominently used by Arrow et al. (1961) to model production functions and predict economic growth. CES utilities have since become one of the most widely used families of utility functions in the economics literature (de la Grandville 2009) due to their versatility and flexibility in economic modeling. The popular modeling language MPSGE (Rutherford 1999) for equilibrium analysis uses CES functions to model consumption and production.

2 Linear Utilities

In this section, we assume that every agent i has a linear utility function defined by $u_i(\mathbf{x}_i) = \sum_j a_{ij}x_{ij}$. We call vector $\mathbf{a}_i = (a_{i1}, \ldots, a_{im})$ the *preference vector* of agent i. In principle, all numbers a_{ij} are L-bit integers but are otherwise unknown. For convenience, we will assume a normalization to $a_{ik_i} = 1$ for every agent i, where k_i is the smallest value such that a_{ik_i} is non-zero. Then all other a_{ij} with $j \neq k_i$ are ratios of two L-bit integers. Since linear preferences are invariant to scaling with a positive scalar, this does not make a qualitative difference.

With linear utilities, one can completely characterize the demand of agents: under price vector $\mathbf{p}$, an allocation $\mathbf{x}_i$ is a demand for agent i with preference vector $\mathbf{a}_i$ if and only if $\mathbf{x}_i \cdot \mathbf{p} = \mathbf{e}_i \cdot \mathbf{p}$ and for every $x_{ik} > 0$, $\frac{a_{ik}}{p_k} = \max_\ell \frac{a_{i\ell}}{p_\ell}$.

2.1 Fisher Markets

In a Fisher market with linear utility buyers, each buyer i can be represented as a tuple $(\mathbf{a}_i, b_i)$ where $\mathbf{a}_i$ is the preference vector and b_i the budget. Given a Fisher market with n buyers and m goods, denote $\mathcal{B} = \{(\mathbf{a}_1, b_1), \ldots, (\mathbf{a}_n, b_n)\}$ as the set of all buyers. We assume here that each buyer i has a different preference vector $\mathbf{a}_i$, since a linear market allows to treat buyers with the same preference vector as a single one with the accumulated budget.

Learning budgets with known preference vectors. Let us first concentrate on a simpler problem: Assume the preference vector $\mathbf{a}_i$ for each buyer i is known, how can we learn the budgets b_i? For any value $x \in \mathbb{R}^+$, we let $x^+ = x + \epsilon$ and $x^- = x - \epsilon$, where $\epsilon > 0$ is a small enough value such that $\epsilon < |a_{ij} - a_{i'j}|$ for any $1 \leq j \leq m$ and $1 \leq i, i' \leq n$ with $a_{ij} \neq a_{i'j}$. Since we assume that all utilities are ratios of two L-bit integers, it suffices to set $\epsilon = 2^{-2L}$. Next, we consider a simple algorithm for learning the budgets with given preference vectors.

Lemma 1. *Given the preference vectors $\mathcal{A}$ of a set of n buyers,* LIN-LEARN-BUDGET *learns the budgets of these buyers using n queries.*

Proof. Before we show correctness, we briefly outline the main idea of algorithm LIN-LEARN-BUDGET. We first learn the budgets of buyers in $\mathcal{A}_1$ by observing the demand of good 1. Then we recursively invoke the algorithm with a reduced set of buyers $\mathcal{A} \setminus \mathcal{A}_1$ and a reduced set of goods. Note that, in principle, we can only query the complete market with all buyers and goods. However, since we have learned all budgets and preferences of buyers in $\mathcal{A}_1$, we can compute their demand bundles in future queries and can reduce the demands returned by the oracle accordingly. Also, no buyer in $\mathcal{A} \setminus \mathcal{A}_1$ spends any budget on good 1. Hence, we can drop good 1 and buyers $\mathcal{A}_1$ from consideration.

We will now show correctness of the algorithm. By reverse induction on the set of goods, we assume that the budgets for buyer set $\mathcal{A} \setminus \mathcal{A}_1$ are determined when treating goods $\{2, \ldots, m\}$. Note that the base case of our induction is the last good m, where we have $\mathcal{A} \setminus (\mathcal{A}_1 \cup \ldots \cup \mathcal{A}_m) = \emptyset$, since otherwise such a buyer would have $\mathbf{a}_i = \mathbf{0}$. Now, by induction we assume that LIN-LEARN-BUDGET $(\mathcal{A} \setminus \mathcal{A}_1, M \setminus$

Algorithm 1: LIN-LEARN-BUDGET $(\mathcal{A}, M, L)$

Input : Set of buyers with preference vectors $\mathcal{A} = \{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$, set of goods M, precision bound L

Output: Buyer preferences and budgets $\mathcal{B} = \{(\mathbf{a}, b) \mid \mathbf{a} \in \mathcal{A}\}$.

Let $\mathcal{A}_1 = \{\mathbf{a}_i \mid \mathbf{a}_i \in A, a_{i1} > 0\}$ and $n_1 = |\mathcal{A}_1|$.
Sort preference vectors of $\mathcal{A}_1$ in non-decreasing lexicographical order, number buyers $\mathbf{a}_1 \prec \cdots \prec \mathbf{a}_{n_1}$.
$\epsilon \leftarrow 2^{-2L-1}$
for $i \leftarrow 1$ **to** n_1 **do**
  $\mathbf{p} = (p_1 = a_{i1} = 1, p_2 = a^+_{i2}, \ldots, p_m = a^+_{im})$
  $b \leftarrow Q_1(\mathbf{p})$
  for $i' \leftarrow 1$ **to** $i - 1$ **do**
    if $\max_j\{\frac{a_{i'j}}{p_j}\} = \frac{a_{i'1}}{p_1} = 1$ **then**
      $b \leftarrow b - b_{i'}$
  Associate preference vector $\mathbf{a}_i$ with budget $b_i = b$.
Run LIN-LEARN-BUDGET $(\mathcal{A} \setminus \mathcal{A}_1, M \setminus \{1\}, L)$.
return $\{(\mathbf{a}_i, b_i) \mid 1 \le i \le n\}$

$\{1\}, L)$ learns the budgets of $\mathcal{A}\setminus\mathcal{A}_1$ correctly. It suffices to show the correctness of the algorithm for buyers in $\mathcal{A}_1$. Recall that by normalization, $a_{i1} = 1$ for all $\mathbf{a}_i \in \mathcal{A}_1$.

The idea of the algorithm is to learn the budgets of buyers in $\mathcal{A}_1$ one by one by the preference vector in lexicographical order. Now assume by induction that we have already correctly learned the budget $b_{i'}$ of every buyer $i' < i$ in $\mathcal{A}_1$. At iteration i, let $\mathbf{p}$ be the price vector that we queried in this round. Note that the value of $Q_1(\mathbf{p})$ equals to the sum of budgets of all buyers with good 1 in their demand set[1].

- For buyer i, we have $\frac{a_{i1}}{p_1} = 1$ and $\frac{a_{ij}}{p_j} = \frac{a_{ij}}{a^+_{ij}} < 1$ for any $j \neq m$. Hence buyer i's budget is included in $Q_1(\mathbf{p})$.
- For every buyer $i' < i$, good 1 is the (unique) demand good if and only if $\max_j \left\{\frac{a_{i'j}}{p_j}\right\} = \frac{a_{i'1}}{p_1} = 1$. This is exactly the condition that we used in the algorithm, and we remove $b_{i'}$ from $Q_1(\mathbf{p})$ in this case.
- Next, for every buyer $i'' > i$, since $\mathbf{a}_i \prec \mathbf{a}_{i''}$, i.e., there must exist some j such that $a_{ij} < a_{i''j}$. Hence for the same price vector $\mathbf{p}$, we have $\frac{a_{i''j}}{p_j} = \frac{a_{i''j}}{a^+_{ij}} > 1 = \frac{a_{i''1}}{p_1}$. This means good m will not be in the demand bundle of i'', hence its budget will not be included in $Q_1(\mathbf{p})$.
- For every buyer in set $\mathcal{A}\setminus\mathcal{A}_1$, it will never request good 1 under any price vector.

Hence we can conclude that the value b at the end of iteration i is exactly the budget of buyer i. □

Remark. LIN-LEARN-BUDGET works correctly as long as the input $\mathcal{A}$ is a superset of the set of all preference vectors in the market. If $\mathcal{A}$ contains some preference vector that no buyer has, the algorithm will assign a corresponding budget 0 to this vector in the output.

[1]By the definition of ϵ, it is easy to check that if good 1 is in some buyer's demand set, that buyer will demand only good 1.

Learning preference vectors. The next step is to identify all possible preference vectors that could belong to a buyer with non-zero budget. Our approach is to learn all possible values good by good. More specifically, for each good j, we want to learn $S_j = \{a_{ij} \mid 1 \le i \le n\}$, which is the set of possible values that buyers have for good j. The proof of the following lemma is deferred to the full version.

Lemma 2. *For any good j, we can learn all values in S_j using at most $O(nL)$ queries.*

All candidate preference vectors must be contained in $S = S_1 \times S_2 \times \cdots \times S_m$, but the set S may contain n^m vectors in the worst case, where n is the actual number of buyers. In order to avoid computing the budgets for all exponentially many preference vectors in S (with almost all vectors ending up with budget 0), we consider a dynamic programming approach. We study markets with only a subset of fewer than m goods and observe an "aggregated" buyer behavior in these markets.

More precisely, for every $1 \le j \le m$, we construct $\mathcal{B}_j$ as a set of buyers of a market with goods $1, \ldots, j$ as following. Here $\mathbf{a}[1:j] = (a_1, \ldots, a_j)$, the subvector of $\mathbf{a}$ with the first j elements.

1. For any $(\mathbf{a}, b) \in \mathcal{B}$, add $(\mathbf{a}[1:j], b)$ to $\mathcal{B}_j$. The only exception is $\mathbf{a}[1:j] = \mathbf{0}$, which we will treat separately.
2. After step (1), in case there are multiple buyers in $\mathcal{B}_j$ with some same preference $\mathbf{a}'$, merge them into one single buyer $(\mathbf{a}', b')$, in which b' is the sum of all budgets of these buyers.

Intuitively, if we keep the prices of all goods $j+1, \ldots, m$ very high (i.e., $\frac{p_k}{a_k} > \max_{1 \le \ell \le j} \frac{p_\ell}{a_\ell}$ for every $j < k \le m$), then no buyer is interested in buying any of those goods. The only exception are buyers with $a_\ell = 0$ for all $1 \le \ell \le j$, which we will exclude for now and treat them separately. Consequently, the behavior of $\mathcal{B}$ on the first j goods is equivalent to a buyer set $\mathcal{B}_j$ on a market with only the first j goods. If we provide the preference vector set $\mathcal{A}_j = \{\mathbf{a} \mid (\mathbf{a}, b) \in \mathcal{B}_j\}$ (or any superset of $\mathcal{A}_j$) to LIN-LEARN-BUDGET, the algorithm will successfully compute the corresponding budgets and output the buyer set $\mathcal{B}_j$ (and budget 0 for all vectors that are in the input but not in $\mathcal{A}_j$). Note that the algorithm will only specify the prices of the first j goods for each query. We keep the prices of the remaining goods large enough, such that no buyer will be interested in them.

For the final algorithm we now do the following. Assume that we have correctly determined $\mathcal{B}_j$ for some $j \ge 1$. To determine $\mathcal{B}_{j+1}$, we note that $\mathcal{A}_{j+1} \subseteq \mathcal{A}_j \times S_{j+1}$. Thus, we feed $\mathcal{A}_j \times S_{j+1}$ into LIN-LEARN-BUDGET (with very high prices on goods $j+2, \ldots, m$). This tells us the budget corresponding to each of these preference vectors and thereby determines $\mathcal{B}_{j+1}$ and $\mathcal{A}_{j+1}$ (by removing all entries with zero budget). Note that we have $|\mathcal{A}_j| < n$ and $|S_j| < n$ for every j, thus the procedure runs in polynomial time. By removing preference vectors with zero budget, we can thus iteratively restrict the set of candidate preference vectors.

Algorithm 2: LIN-FISHER-MAIN (m, L)

Input : number of goods m, precision bound L
Output: Set of buyers with preference vectors and corresponding budgets $\mathcal{B} = \{(\mathbf{a}, b)\}$.

Construct S_j, for all $1 \le j \le m$, as in Lemma 2.
$b \leftarrow Q(1, 2^L + 1, 2^L + 1, \ldots, 2^L + 1)$
$\mathcal{A}_1 \leftarrow \{(1)\}$, $\mathcal{B}_1 \leftarrow \{((1), b)\}$
for $j \leftarrow 2$ **to** m **do**
  $\mathcal{A}'_j \leftarrow \{(\mathbf{a}, a_j) \mid \mathbf{a} \in \mathcal{A}_{j-1}, a_j \in S_j\} \cup \{(0, \ldots, 0, 1)\}$
  $\mathcal{B}'_j \leftarrow$ LIN-LEARN-BUDGET $(\mathcal{A}'_j, \{1, \ldots, j\}, L)$
  $\mathcal{A}_j = \{\mathbf{a} \mid (\mathbf{a}, b) \in \mathcal{B}'_j, b > 0\}$
  $\mathcal{B}_j = \{(\mathbf{a}, b) \mid (\mathbf{a}, b) \in \mathcal{B}'_j, b > 0\}$
return $\mathcal{B}_m$

In addition, there might be buyers with $a_{i\ell} = 0$ for all $1 \le \ell \le j$ and $a_{i,j+1} > 0$. These buyers did not appear in the demands for goods $1, \ldots, j$. We know that their preference vector has $a_{i,j+1} = 1$. We thus also add $(\mathbf{0}_j, 1)$ in our candidate set, and due to the large prices on goods $j + 2, \ldots, m$ they will only demand good $j + 1$. Hence, we will correctly capture their total budget in this round and continue to correctly subdivide them in later iterations. For the overall algorithm see LIN-FISHER-MAIN. A straightforward analysis of the running time gives us the main theorem of this section.

Theorem 1. *An unknown Fisher market with linear utility functions can be learned using $O(n^2m + nmL)$ queries in polynomial time.*

2.2 Exchange Markets

We now extend our analysis to exchange markets. To learn the endowments of each agent, we note the following properties in our previous algorithm for Fisher markets:

(1) The price vector queried in the algorithm always has $p_1 = 1$. Hence, $Q_1(\mathbf{p})$ is both the aggregated demand of good 1, as well as the amount of money spend on good 1.

(2) In any query within the algorithm, every agent either spends all its budget on good 1, or spends nothing on good 1. Hence, there is no ambiguity in $Q_1(\mathbf{p})$.

Property (2) is essentially a consequence of our condition that our queries result in a unique demand bundle for every agent. Thus, we can also slightly alter a queried price and get the same output. More precisely, there is a sufficiently small value $\epsilon' > 0$ with the following property. For any price vector $\mathbf{p}$ queried in LIN-FISHER-MAIN and any good j, if we increase the price of good j by ϵ' and keep other prices unchanged, then property (2) does still hold. This follows since all prices and preference values come from a finite set based on ratios of L-bit numbers (or $(3L + 1)$-bits if prices are based on $\epsilon = 2^{-2L}$). Thus, it is obvious that a sufficiently small ϵ' exists that keeps the unique demand bundle the same. For this, it is sufficient to preserve the strict inequalities between the maximum bang-per-buck ratio $\max_j a_{ij}/p_j$ and the one of every other good. Here a_{ij} is itself a ratio of two L-bit numbers and p_j is a $(3L+1)$-bit number. Hence, the denominator of each bang-per-buck ratio is a number of at most $(4L+1)$ bits, and when comparing two of these, the common denominator becomes a number of at most $8L + 2$ bits. Thus, a (very conservative) sufficient value is $\epsilon' = 2^{-8L-2}$.

Algorithm 3: LIN-EXCHANGE (m, L, k)

Run LIN-FISHER-MAIN (m, L), but replace each query $Q(\mathbf{p})$ in the algorithm by $\frac{1}{\epsilon'}[Q(\mathbf{p}^{k+}) - Q(\mathbf{p})]$.
Return the computed set $\mathcal{B}$

Algorithm 4: LIN-EXCHANGE-MAIN (m, L)

Input : number of goods m, precision bound L
Output: Set of agents with preference vectors and initial endowments $\mathcal{B}^j = \{(\mathbf{a}, \mathbf{e})\}$.

for $j \leftarrow 1$ **to** m **do**
  $\mathcal{B}^j \leftarrow$LIN-EXCHANGE (m, L, j)
$\mathcal{A} \leftarrow \{\mathbf{a} \mid (\mathbf{a}, b) \in \bigcup_j \mathcal{B}^j\}$
$\mathcal{B} \leftarrow \emptyset$
foreach $\mathbf{a} \in \mathcal{A}$ **do**
  for $j \leftarrow 1$ **to** m **do**
    $e_j \leftarrow \begin{cases} b_j & \text{if there exists } (\mathbf{a}, b_j) \in \mathcal{B}^j \\ 0 & \text{otherwise.} \end{cases}$
  Add $(\mathbf{a}, \mathbf{e} = \{e_1, \ldots, e_m\})$ to $\mathcal{B}$.
return $\mathcal{B}$

Next, for a price vector $\mathbf{p}$, let $\mathbf{p}^{k+}$ be the price vector where we increase the price of good k by ϵ' and keep other prices unchanged. Now LIN-EXCHANGE (m, L, k) is the following algorithm: Since for every agent i, it either spends all its money on good j in both $Q(\mathbf{p}^{k+})$ and $Q(\mathbf{p})$, or spends nothing on good j in both queries. Furthermore, for an agent i with initial endowment $\mathbf{e}_i$, its money is increased by $\epsilon' e_{ik}$ when we change the price vector from $\mathbf{p}$ to $\mathbf{p}^{k+}$. Thus, by replacing $Q(\mathbf{p})$ by $\frac{1}{\epsilon'}[Q(\mathbf{p}^{k+}) - Q(\mathbf{p})]$, we essentially replace each agent i with initial endowment $\mathbf{e}_i$ by an agent with fixed budget e_{ik}. This reduces the problem to the Fisher market case and allows us to use the previous results to identify the endowment of good k of each agent. Hence, by running LIN-EXCHANGE m times for each good k, we are able to fully learn the initial endowment of each agent as well as their preference vectors. The final algorithm is summarized in LIN-EXCHANGE-MAIN, and the following theorem summarizes our result.

Theorem 2. *An unknown exchange market with linear utility functions can be learned using $O(n^2m^2+nm^2L)$ queries in polynomial time.*

3 CES Utilities

In this section we consider markets where each buyer has utilities with constant elasticity of substitution (CES). The utility function for buyer i is of the form $u_i(\mathbf{x}_i) = \left(\sum_j (a_{ij}x_{ij})^\rho\right)^{1/\rho}$, with parameter $\rho < 1$. We assume a_{ij}

and b_j are rational, for every $1 \le i \le n, 1 \le j \le m$.

Fact 1. *For each buyer i with CES utility function and any given price vector $\mathbf{p} = (p_1, \ldots, p_m)$, there exists a unique demand bundle $\mathbf{x}_i$ given by*

$$x_{ij} = \frac{w_{ij}}{q_i} p_j^{\frac{1}{\rho-1}} b_i \ , \tag{1}$$

where $w_{ij} = a_{ij}^{\frac{1}{1-\rho}}$ and $q_i = \sum_k w_{ik} p_k^{\frac{\rho}{\rho-1}}$.

This follows from solving the optimization problem and combining

$$\frac{\partial u_i}{\partial x_{ij}} \cdot \frac{1}{p_j} = \frac{\partial u_i}{\partial x_{ij'}} \cdot \frac{1}{p_{j'}}$$

for all $j \neq j'$ with constraint $\sum_{j=1}^m x_{ij} p_j = b_i$.

3.1 Fisher Markets with Leontief Utilities

In a Fisher market with CES utilities, each buyer i has a fixed budget b_i of money. We start our analysis in the special case of Leontief utilities – CES utility functions with $\rho = -\infty$. When we apply this condition to Eqn (1), we obtain

$$x_{ij} = \frac{w_{ij} b_i}{\sum_k w_{ik} p_k} \ ,$$

and $Z_j = \sum_i x_{ij} = \sum_i \frac{w_{ij} b_i}{\sum_k w_{ik} p_k}$, where $w_{ij} = 1/a_{ij}$. Note that Leontief utilities are only well-defined if all $a_{ij} > 0$. We here consider a slight generalization that proves useful below. We assume that each buyer i is interested in a subset S_i of goods, where $w_{ij} > 0$ holds for all $j \in S_i$, and $w_{ij} = 0$ and $x_{ij} = 0$ for all $j \notin S_i$. We will learn all w_{ij} correctly, which allows to identify subset S_i and a_{ij} for each $j \in S_i$.

Theorem 3. *An unknown Fisher market with Leontief utilities can be learned using $O(n^3 m)$ queries in polynomial time.*

Proof. For Leontief utilities we again normalize w.l.o.g. to $w_{ik_i} = 1$ for every buyer i, where we define k_i to be the smallest value such that $w_{ik_i} > 0$. Similar to linear utilities, we define $\mathcal{F}_j = \{i \mid k_i = j\}$ and first learn the utility functions of buyers in $\mathcal{F}_1$ by observing the demand of good 1. We then proceed to $\mathcal{F}_2, \ldots, \mathcal{F}_m$.

We set the price vectors according to the following format: Given $x, y > 0$, we set $p_1 = y$ and $p_2 = x, p_3 = x^2, \ldots, p_m = x^{m-1}$. The demand of good Z_1 can then be written as the following bivariate rational polynomial

$$Z_1(x, y) = \sum_{i \in \mathcal{F}_1} \frac{b_i}{y + w_{i2} x + w_{i3} x^2 + \cdots + w_{im} x^{m-1}} \ .$$

Note that only buyers from $\mathcal{F}_1$ will demand good 1. We reformulate $Z_1(x, y) = \frac{P(x,y)}{Q(x,y)}$, where $P(x, y)$ and $Q(x, y)$ are all bivariate polynomials with degree no more than $n(m-1)$. In particular, we have

$$Q(x, y) = \prod_{i \in \mathcal{A}_1} \left(y + \sum_{k=1}^m w_{ik} x^{k-1} \right) \ .$$

All terms in $Q(x, y)$ or $P(x, y)$ with non-zero coefficients are of form $x^\alpha y^\beta$ where $1 \le \alpha \le n(m-1)$ and $1 \le \beta \le n$. This means both $Q(x, y)$ and $P(x, y)$ have no more than $n^2(m-1)$ non-zero coefficients. Consider all these coefficients as variables. A point (x^*, y^*) and the corresponding $Z_1(x^*, y^*)$ can be transformed into a linear equation of these variables, more precisely,

$$P(x^*, y^*) = Z_1(x^*, y^*) Q(x^*, y^*) \ .$$

Hence, by picking $k = 2n^3 m + 1$ pairs of positive numbers $(x_1, y_1), (x_2, y_2), \ldots, (x_k, y_k)$, querying price vector $(y_i, x_i, x_i^2, \ldots, x_i^{m-1})$ in round i and observing the corresponding demand $Z_1(x_i, y_i)$ for $i = 1, \ldots, k$, one can get $2n^3 m + 1$ linear equations. Solving this linear system (which can be done in polynomial time) yields us the values of all coefficients in polynomial $P(x, y)$ and $Q(x, y)$.

Polynomial Factorization: Polynomial rings over a field are unique factorization domains, and polynomials of form $y + f(x)$ (where $f(x)$ is an arbitrary univariate polynomial) are all irreducible. This means $Q(x, y)$ can be uniquely factorized into $\prod_i (y + \sum_k w_{ik} x^{k-1})$. Further, Kaltofen (1982) and Lenstra, Lenstra, and Lovász (1982) show that such a factorization can be computed in polynomial time. Thus, by learning this factorization, one also learns all parameters w_{ij} in every buyer's utility function.

Discovering Budgets: The final step is to learn the budget b_i for each buyer $i \in \mathcal{F}_1$. Let polynomial $R_i(x, y) = Q(x, y)/(y + \sum_k w_{ik} x^{k-1})$. We have

$$P(x, y) = \sum_{i \in \mathcal{A}_1} b_i \cdot R_i(x, y) \ . \tag{2}$$

It is also easy to check that $R_i(x, y)$ are all linearly independent. Hence Eqn (2) is the unique linear combination of $R_i(x, y)$ for $P(x, y)$. The coefficients of such combination, which are exactly the budgets b_i for each buyer $i \in \mathcal{F}_1$, can again be learned by solving a set of linear equations. We omit the details here.

Proceed with $\mathcal{F}_2, \ldots, \mathcal{F}_m$: After learning the utility functions of buyers in $\mathcal{F}_1$, we proceed with buyers in $\mathcal{F}_2$ using the same approach with the demand of good 2, and then to $\mathcal{F}_3, \ldots, \mathcal{F}_m$. The algorithm and analysis remain the same except for one change: when learning utilities of buyers in $\mathcal{F}_j$, buyers in $\mathcal{F}_1, \ldots, \mathcal{F}_{j-1}$ may also be demanding good j at all queried price vectors. Since we already know utility functions and budgets of these buyers, we can compute exactly how much of good j they demand for any price vector. We can then subtract these demands from Z_j, and the remaining value are demands from $\mathcal{F}_j$. This allows us to learn the utilities and budgets of all buyers.

Note that when applying the algorithm for $\mathcal{F}_j$, we do not need to query a new set of $2n^3 m + 1$ prices. Instead, we revisit the same set of demand vectors that we collected for $\mathcal{F}_1$, and use Z_j, the demands of good j, as our data for learning utilities of buyers in $\mathcal{F}_j$. Hence, the overall number of queries for the algorithm remains in $O(n^3 m)$. □

3.2 Fisher Markets with CES Utilities

For general CES utility functions, the excess demand is

$$Z_j = \sum_{i=1}^n x_{ij} = p_j^{\frac{1}{\rho-1}} \sum_{i=1}^n \frac{w_{ij} b_i}{q_i} \ .$$

Let us rewrite it as

$$\frac{Z_j}{p_j^{\frac{1}{\rho-1}}} = \sum_{i=1}^{n} \frac{w_{ij} b_i}{\sum_j w_{ij} p_j^{\frac{\rho}{\rho-1}}} \quad .$$

Note that we can compute the value of all w_{ij} by querying appropriate price vectors $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_k$ for Leontief utilities. For general CES utility functions, we use $p'_j = p_j^{\frac{\rho}{\rho-1}}$ and $Z'_j = \frac{Z_j}{p_j^{\frac{1}{\rho-1}}}$ to turn the problem into a Leontief case. Observe that by (1) for CES functions, $a_{ij} = 0$ implies $w_{ij} = 0$ and $x_{ij} = 0$. Nevertheless, the algorithm for the Leontief case above can be applied, since we explicitly treated the generalization that allows $w_{ij} = 0$. This implies the following theorem.

Theorem 4. *An unknown Fisher market with general CES utilities can be learned using $O(n^3 m)$ queries in polynomial time.*

Acknowledgment

This work was supported by DFG Cluster of Excellence MMCI, in part by the National Natural Science Foundation of China Grant 61170062, 61222202, 61433014 and the China National Program for support of Top-notch Young Professionals.

References

Afriat, S. N. 1967. The construction of utility functions from expenditure data. *Intl. Econom. Rev.* 8(1):67–77.

Amin, K.; Cummings, R.; Dworkin, L.; Kearns, M.; and Roth, A. 2015. Online learning and profit maximization from revealed preferences. In *Proc. 29th Conf. Artificial Intelligence (AAAI)*, 770–776.

Arrow, K.; Chenery, H.; Minhas, B. S.; and Solow, R. 1961. Capital-labor substitution and economic efficiency. *Rev. Econom. Stat.* 43(3):225–250.

Balcan, M.; Daniely, A.; Mehta, R.; Urner, R.; and Vazirani, V. V. 2014. Learning economic parameters from revealed preferences. In *Proc. 10th Intl. Conf. Web and Internet Economics (WINE)*, 338–353.

Bei, X.; Garg, J.; and Hoefer, M. 2015. Tatonnement for linear and gross substitutes markets. *CoRR* abs/1507.04925.

Beigman, E., and Vohra, R. 2006. Learning from revealed preference. In *Proc. 7th Conf. Electronic Commerce (EC)*, 36–42.

Blum, A.; Mansour, Y.; and Morgenstern, J. 2015a. Learning valuation distributions from partial observation. In *Proc. 29th Conf. Artificial Intelligence (AAAI)*, 798–804.

Blum, A.; Mansour, Y.; and Morgenstern, J. 2015b. Learning what's going on: Reconstructing preferences and priorities from opaque transactions. In *Proc. 16th Conf. Economics and Computation (EC)*, 601–618.

Codenotti, B.; Pemmaraju, S.; and Varadarajan, K. 2005. On the polynomial time computation of equilibria for certain exchange economies. In *Proc. 16th Symp. Discrete Algorithms (SODA)*, 72–81.

Cole, R., and Fleischer, L. 2008. Fast-converging tatonnement algorithms for one-time and ongoing market problems. In *Proc. 40th Symp. Theory of Computing (STOC)*, 315–324.

Daniely, A., and Shalev-Shwartz, S. 2014. Optimal learners for multiclass problems. In *Proc. 27th Conf. Learning Theory, (COLT)*, 287–316.

de la Grandville, O. 2009. *Economic Growth. A Unified Approach*. Cambridge Univ. Press.

Dickinson, H. 1954. A note on dynamic economics. *Rev. Econom. Stud.* 22:169–179.

Jabbari, S.; Rogers, R. M.; Roth, A.; and Wu, Z. S. 2015. Learning from rational behavior: Predicting solutions to unknown linear programs. *CoRR* abs/1506.02162.

Kaltofen, E. 1982. A polynomial-time reduction from bivariate to univariate integral polynomial factorization. In *23rd Symp. Foundations of Computer Science (FOCS)*, 57–64.

Lenstra, A.; Lenstra, H.; and Lovász, L. 1982. Factoring polynomials with rational coefficients. *Math. Ann.* 261:515–534.

Mas-Colell, A.; Whinston, M. D.; and Green, J. R. 1995. *Microeconomic Theory*. Oxford University Press.

Roth, A.; Ullman, J.; and Wu, Z. S. 2015. Watch and learn: Optimizing from revealed preferences feedback. *CoRR* abs/1504.01033.

Rutherford, T. 1999. Applied general equilibrium modeling with MPSGE as a GAMS subsystem: An overview of the modeling framework and syntax. *Comput. Econom.* 14(1–2):1–46.

Samuelson, P. 1948. Consumption theory in terms of revealed preference. *Econometrica* 15.

Solow, R. 1956. A contribution to the theory of economic growth. *Quarterly J. Econom.* 70(1):65–94.

Varian, H. 2005. Revealed preference. In Szenberg, M.; Ramrattan, L.; and Gottesman, A., eds., *Samuelsonian Economics and the 21st Century*. Oxford Univ. Press. 99–115.

Walras, L. 1874. *Éléments d'économie politique pure ou théorie de la richesse sociale (Elements of Pure Economics, or the theory of social wealth)*. (1899, 4th ed.; 1926, rev ed., 1954, Engl. transl.).

Zadimoghaddam, M., and Roth, A. 2012. Efficiently learning from revealed preference. In *Proc. 8th Intl. Workshop Internet and Network Economics (WINE)*, 114–127.

A Proof of Lemma 2

Lemma 2. *For any good j, we can learn all values in S_j using at most $O(nL)$ queries.*

Proof. We denote by $\mathcal{F}_j = \{i \mid j = \min\{j' \mid a_{ij'} > 0\}\}$ the (unknown) set of buyers for which j is the good of smallest index and $a_{ij} > 0$. We define $S_j^{j'} = \{a_{ij} \mid i \in \mathcal{F}_{j'}\}$ and observe $S_j^1 \neq \emptyset$. Initially, we make a single query with price vector $\mathbf{p} = (p_1, \ldots, p_m)$ with $p_j = (2^L + 1)^j$ for every good j. Due to the exponential price growth in $\mathbf{p}$, every $i \in \mathcal{F}_j$ spends its entire budget on j, which reveals whether $\mathcal{F}_j = \emptyset$ or not. Let $j_{\max} = \max\{j \mid \mathcal{F}_j \neq \emptyset\}$ and note that $S_j = \bigcup_{j'} S_j^{j'} \cup \{0\}$ for $j < j_{\max}$ and $S_j = \bigcup_{j'} S_j^{j'}$ otherwise.

We continue by learning S_j^1. Let $D_j^1(x)$ be the aggregate demand of good 1 when we query the market with price vector $\mathbf{p} = (p_1, \ldots, p_m)$, where $p_1 = 1$, $p_j = x - \epsilon$ and $p_{j'} = 2^L + 1$ for all $j' \neq j$ and $j' \neq 1$, where $\epsilon = 2^{-2L}$. For every such query, no buyer in $\mathcal{F}_1$ would be interested in any good other than goods 1 and j. Further, every $i \notin \mathcal{F}_1$ always spends its entire budget on goods other than 1. Now observe

$$D_j^1(y) - D_j^1(x) = \sum_{\substack{i \in \mathcal{F}_1 \\ x \leq a_{ij} < y}} b_i \ .$$

Hence, these kinds of queries allow us to learn all values in S_j^1 through a standard binary search. Note that all possible values in S_j^1 are ratios of two integers of no more than L bits each, hence the binary search tree has depth $O(L)$ and $|S_j^1|$ leaves. Therefore the size of the tree, which is also the number of queries needed to identify these values, is upper bounded by $O(|S_j^1|L)$.

For S_j^2, note that $S_j^2 = \emptyset$ if $\mathcal{F}_2 = \emptyset$ (which is revealed in the very first query). So assume otherwise, and let $D_j^2(x)$ be the aggregate demand of good 2 when we query the market with price vector $\mathbf{p} = (p_1, \ldots, p_m)$, where $p_1 = 1$, $p_2 = (2^L + 1)$, $p_j = (2^L + 1)(x - \epsilon)$, $p_{j'} = (2^L + 1)^2$ for all $j' \notin \{1, 2, j\}$, and $\epsilon = 2^{-2L}$. For every such query, the money spent on good 1 is exactly the budget of all buyers in $\mathcal{F}_1$. From the remaining buyers, no buyer in $\mathcal{F}_2$ would be interested in any good other than goods 2 and j. Further, every $i \notin (\mathcal{F}_2 \cup \mathcal{F}_1)$ always spends the entire budget on goods other than 1 or 2. Observe that $D_j^2(y) - D_j^2(x) = \sum_{\substack{i \in \mathcal{F}_2 \\ x \leq a_{ij} < y}} b_i$. Hence, this allows to learn S_j^2 by the same approach as above. In a similar fashion, we can learn $S_j^3, \ldots, S_j^m$ using exponentially increasing prices. The number of queries to learn S_j is upper bounded by $O\left(\sum_{j'} |S_j^{j'}|L\right) = O(nL)$. □