First-order Convex Fitting and Its Application to Economics and Optimization

Quinlan Dawkins, Minbiao Han, Haifeng Xu

Department of Computer Science, University of Virginia
{qed4wg, mh2ye, hx4ad}@virginia.edu

Abstract

This paper studies a basic function fitting problem, which we coin *first-order convex fitting* (`FCF`): given any two vector sequences $\{\boldsymbol{x}_i\}_{i\in[T]}$ and $\{\boldsymbol{p}_i\}_{i\in[T]}$ in $\mathbb{R}^d$, when is it possible to *efficiently* construct a convex function $f(\boldsymbol{x})$ that "fits" the two sequences in the first-order sense, i.e, its gradient $\nabla f(\boldsymbol{x}_i)$ equals $\boldsymbol{p}_i$ for all $i \in [T] = \{1, \cdots, T\}$? Despite the close connection between function fitting and machine learning, `FCF` has surprisingly been overlooked in the past literature. With an efficient constructive proof, we discover a clean answer to this question: `FCF` is possible *if and only if* the two sequences are *permutation stable*: $\sum_{i=1}^{T} \boldsymbol{x}_i \cdot \boldsymbol{p}_i \geq \sum_{i=1}^{T} \boldsymbol{x}_i \cdot \boldsymbol{p}_{\sigma(i)}$ for any permutation σ of $[T]$.

We demonstrate the usefulness of `FCF` in two applications. First, we study how it can be used as an empirical risk minimization procedure to learn the original convex function. We provide efficient PAC-learnability bounds for special classes of convex functions learned via `FCF`, and demonstrate its application to multiple economic problems where only function gradients (as opposed to function values) can be observed. Second, we empirically show how it can be used as a surrogate to significantly accelerate the minimization of the original convex function.

1 Introduction

The theory of *revealed preferences* starts from Samuelson's seminal work in the 1940s (Samuelson 1938) and has formed a celebrated subfield of consumer theory. Consider a buyer who looks to buy *fractional* bundles of d goods repeatedly from a seller and generates a sequential purchase history of $(\boldsymbol{p}_1, \boldsymbol{x}_1), \cdots, (\boldsymbol{p}_T, \boldsymbol{x}_T)$ where $\boldsymbol{p}_i \in \mathbb{R}^d$ is the price vector at time $i \in [T]$ and $\boldsymbol{x}_i \in \mathbb{R}^d$ is the customer's purchase bundle. Classic economic research of revealed preferences studies when it is possible to find a buyer value function $u(\boldsymbol{x})$ that explains the observed data $\{(\boldsymbol{p}_i, \boldsymbol{x}_i)\}_{i=1}^{T}$ assuming a rational utility-maximizing buyer with some budget, or more formally, guarantees $\boldsymbol{x}_i \in \mathrm{argmax}_{\boldsymbol{x}\cdot\boldsymbol{p}_i \leq B}\, u(\boldsymbol{x})$. This is sometimes also called *rationalizing the data*. Since (Samuelson 1938), there has been a rich line of economic research; We refer the reader to an excellent survey by Varian (Varian 2006).

The above problem has nice characterizations under some mild assumptions on the buyer's behavior. In a textbook style setup (Mas-Colell et al. 1995), the buyer's value function over the goods is assumed to be *concave* and non-decreasing. In addition, the buyer's utility over purchasing goods $\boldsymbol{x} \in \mathbb{R}^d$ is a quasi-linear function as $v(\boldsymbol{x}) - \boldsymbol{x} \cdot \boldsymbol{p}$. It turns out in this case, the historical price vector $\boldsymbol{p}_i$ in the observed dataset $\{(\boldsymbol{p}_i, \boldsymbol{x}_i)\}_{i=1}^{T}$ will always be the gradient of buyer's value function $v(\boldsymbol{x})$ at $\boldsymbol{x}_i$, i.e. $\boldsymbol{p}_i = \nabla v(\boldsymbol{x}_i)$ (Roth, Ullman, and Wu 2016). However, the aim of previous work is to find the revenue optimizing price without knowing the buyer's value function. Given the above characterization, we aim to answer the underlying mathematical questions:

> *Given any data $(\boldsymbol{p}_1, \boldsymbol{x}_1), \cdots, (\boldsymbol{p}_T, \boldsymbol{x}_T)$, when can we efficiently construct a convex function whose gradients fit the observed dataset? How many data points are needed in order to guarantee the fitted convex function is "close" to the underlying true function?*

We consider a dataset with two vector sequences $\{\boldsymbol{x}_i\}_{i\in[T]}$ and $\{\boldsymbol{p}_i\}_{i\in[T]}$. Our goal is to determine if there exists a *convex* function $f(\boldsymbol{x})$ whose first-order gradients fit the given dataset, i.e. $\nabla f(\boldsymbol{x}_i) = \boldsymbol{p}_i$. Notably, our results hold equally for both convex and concave functions. We chose to pose everything as convex simply because it is the more common setting in computer science. Let the sequence of $\{\boldsymbol{x}_i\}_{i\in[T]}$ be drawn from a fixed distribution $\mathcal{D}$, and the other sequence $\{\boldsymbol{p}_i\}_{i\in[T]}$ was their first-order gradients generated according to some unknown convex $f(\boldsymbol{x})$. Can we learn an approximately correct hypothesis convex function with probability at least $1 - \delta$, such that when a new $\boldsymbol{x}$ is drawn from the same distribution, the hypothesis correctly determines the function's gradient at $\boldsymbol{x}$ with error at most $1 - \varepsilon$.

1.1 Our Results and Techniques

We aim to characterize the dataset of two vector sequences such that it can be "first-order" fitted by an underlying convex function. Our first main result is a clean characterization which we term *permutation stability* of the dataset: $\sum_{i=1}^{T} \boldsymbol{p}_i \cdot \boldsymbol{x}_i \geq \sum_{i=1}^{T} \boldsymbol{p}_i \cdot \boldsymbol{x}_{\sigma(i)}$ for any permutation σ of $[T]$. We provide a constructive proof which can efficiently construct such a convex function to fit the given dataset. Leveraging this result, we apply our technique to convex optimization and propose an optimization scheme that experimentally matches or outperforms state-of-the-art and classical solvers for large-scale optimization problems typically arising in machine learning.

All our characterizations so far focus on constructing one specific convex function to fit the observed dataset. Suppose the dataset was generated by some specific unknown convex function, the other result we have shows we can efficiently learn the underlying continuous function to predict future data with small error and high confidence, using samples *polynomial* to the size of the discretized set of the function's value space. As in high-dimension, we also give efficient algorithms with polynomial sample complexity for piecewise linear convex functions. This illustrates an interesting message that we can use the proposed method to rationalize the agents' behaviors and learn the agents' private utility information in some economic applications.

1.2 Related Work

Our work is motivated by the literature of revealed preferences started by (Samuelson 1938). Classic research on revealed preference studies how to construct a function to fit a sequence of purchase history from a buyer with unknown value function (Beigman and Vohra 2006; Zadimoghaddam and Roth 2012; Balcan et al. 2014). Their model assumes that the buyer's behavior at each round i is optimal under some budget, i.e. $\boldsymbol{x}_i \in \operatorname{argmax}_{\boldsymbol{x}\cdot\boldsymbol{p}_i \leq B} u(\boldsymbol{x})$. Going beyond pricing problems, there has also been a line of research studying learning from revealed preference in other classes of games such as general Stackelberg games (Letchford, Conitzer, and Munagala 2009; Peng et al. 2019) and Stackelberg security games (Blum, Haghtalab, and Procaccia 2014; Marecki, Tesauro, and Segal 2012; Peng et al. 2019). In these cases, the revealed preferences are simply the follower's best responses to the leader's strategy at each round. More related to our problem is the other buyer model in the pricing problem of this literature, (Roth, Ullman, and Wu 2016; Roth et al. 2020) used a different quasi-linear buyer model $v(\boldsymbol{x}) - \boldsymbol{p}\cdot\boldsymbol{x}$ and found that a rational buyer's purchase at each round i would make $\boldsymbol{p}_i = \nabla v(\boldsymbol{x}_i)$. They proposed a method to learn the revenue-maximizing price for the seller without knowing the buyer's private value function. Our work is totally different since our goal is to learn the buyer's private value function from the purchase history.

Another area of relevant literature is with machine learning problems. The study of revealed preferences is intrinsically a function fitting problem. However, different from standard function fitting for (variable, value) sequences, here we look for a function $f(\boldsymbol{x})$ to fit the (variable, gradient) sequences. (Beigman and Vohra 2006) also studied the prediction aspects of the revealed preferences theory. They considered the PAC-learning model and introduced the sample complexity and learnability of different classes of value functions. (Zadimoghaddam and Roth 2012) then proposed specific efficient learning algorithms for linearly separable concave value functions in the PAC-learning model. Their sample complexity bound was later improved by (Balcan et al. 2014), and (Balcan et al. 2014) also provided efficient algorithms for other specific classes of value functions including linear, separable piecewise-linear concave (SPLC), CES and Leontief (Mas-Colell et al. 1995). Our work considers a different buyer behavior model without budgets and deals with more general classes of value functions. We consider any convex value function in single dimension, i.e. when the buyer only purchases single good from the seller and provide efficient sample complexity for the PAC-learning model. In higher dimension, we consider a more general piecewise linear value function which doesn't need to be separable.

2 The Problem of First-order Convex Fitting

We start with some preliminaries. Let $f(\boldsymbol{x}) : X \rightarrow \mathbb{R}$ be any function where the compact set $X \subseteq \mathbb{R}^d$ is the domain of f. A vector $\boldsymbol{p} \in \mathbb{R}^d$ is called a *sub-gradient* for f at $\boldsymbol{x} \in X$ if for any $\boldsymbol{x}' \in X$ we have $f(\boldsymbol{x}') \geq f(\boldsymbol{x}) + \boldsymbol{p}\cdot(\boldsymbol{x}' - \boldsymbol{x})$. Function f is called *convex* if for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$ and any $\alpha \in [0,1]$ we have $\alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{x}') \geq f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{x}')$. The function is *strictly* convex if the above inequality is always strict, except for $\boldsymbol{x} = \boldsymbol{x}'$. Sub-gradients do not always exist. However, a convex function has at least one sub-gradient at any $\boldsymbol{x} \in X$. For a differentiable convex function f, its gradient $\nabla f(\boldsymbol{x})$ is the only sub-gradient at $\boldsymbol{x}$ for any $\boldsymbol{x} \in X$. If f is convex but not differentiable, it may have multiple sub-gradients at some $\boldsymbol{x}$. In this case, we use $\partial f(\boldsymbol{x})$ to denote the *set* of all sub-gradients of f at $\boldsymbol{x}$. For convenience of stating our results, we will mostly work with *differentiable* convex functions in this paper though most of our results easily generalize to non-differentiable functions.

This paper studies a very basic problem of using a convex function to fit two vector sequences in the first-order sense, formally stated as follows.

Problem. *[First-order Convex Fitting (*`FCF`*)] Given any two vector sequences $\{\boldsymbol{x}_i\}_{i\in[T]}$ and $\{\boldsymbol{p}_i\}_{i\in[T]}$ in $\mathbb{R}^d$, when is it possible to* efficiently *construct a convex function $f(\boldsymbol{x})$ such that $\boldsymbol{p}_i$ is a sub-gradient at $\boldsymbol{x}_i$, i.e., $\boldsymbol{p}_i \in \partial f(\boldsymbol{x}_i)$, for any $i \in [T] = \{1, \cdots, T\}$?*

A "stronger" version of the above function fitting problem is to require the convex function f to be strictly convex. In this case, the problem is referred to as strict first-order convex fitting, or strict `FCF`.

3 A Complete Characterization of `FCF`

In this section, we provide a necessary and sufficient condition on the two sequences $\{\boldsymbol{x}_i\}_{i\in[T]}, \{\boldsymbol{p}_i\}_{i\in[T]}$, under which there exists a convex function $f(\boldsymbol{x})$ to fit the two sequences in the sense of `FCF`. To state our result, we only need the following notion, which we coin *permutation stability*. Let Σ_T denote the set of all permutations over the set $[T]$. Recall that a permutation $\sigma \in \Sigma_T$ is a bijection from $[T]$ to $[T]$.

Definition 1. *[Permutation Stability] Any two vector sequences $\{\boldsymbol{x}_i\}_{i\in[T]}$ and $\{\boldsymbol{p}_i\}_{i\in[T]}$ are* permutation stable *if for any permutation $\sigma \in \Sigma_T$, the following holds*

$$\sum_{i\in[T]} \boldsymbol{x}_i \cdot \boldsymbol{p}_i \geq \sum_{i\in[T]} \boldsymbol{x}_{\sigma(i)} \cdot \boldsymbol{p}_i \qquad (1)$$

$\{\boldsymbol{x}_i\}_{i\in[T]}$ and $\{\boldsymbol{p}_i\}_{i\in[T]}$ are strictly *permutation stable if the above inequality is strict for any σ that is not the identical mapping.*

Intriguingly, it turns out that `FCF` is fully characterized by the permutation stability of any two data sequences $\{\boldsymbol{x}_i\}_{i\in[T]}$ and $\{\boldsymbol{p}_i\}_{i\in[T]}$ in $\mathbb{R}^d$.

Theorem 1. *For any $T \geq 1$ and any two vector sequences $\{\boldsymbol{x}_i\}_{i\in[T]}$ and $\{\boldsymbol{p}_i\}_{i\in[T]}$ in $\mathbb{R}^d$, there exists a convex function $f(\boldsymbol{x})$ that first-order fits these two sequences — i.e., $\boldsymbol{p}_i \in \partial f(\boldsymbol{x}_i)$ for any $i \in [T]$ —* if and only if *the two sequences are permutation stable.*

Before proceeding to proving Theorem 1, we make a few remarks. First, Theorem 1 generalizes to strictly `FCF` in a straightforward way: strictly `FCF` is possible if and only if the two sequences are strictly permutation stable (see Appendix A for a formal proof).

Second, our proof of Theorem 1 is constructive. That is, whenever $f(\boldsymbol{x})$ exists, we can construct such a $f(\boldsymbol{x})$ efficiently in polynomial time. More concretely, the construction only needs to solve a linear inequality system with T variables and $O(T^2)$ constraints. The main technical challenge, however, is to prove the equivalence between the feasibility of this linear system that we set up and permutation stability. Our proof employs interesting techniques such as Farkas' lemma and network flow decomposition, which does not appear to be apparently relevant at the first glance.

Third, though `FCF` appears a quite basic question even for its own sake, we are not aware of any previous study. To the best our knowledge, the characterization question of `FCF` was studied only in the mathematical literature by Rockafellar (1966; 1970), but from a completely different perspective. Rockafellar considers an abstract *relation* between two (typically continuum) Banach spaces, and identifies a property of this relation termed *cyclical monotonicity* that captures the existence of a convex function that fits the two Banach spaces in the sense of `FCF`. The major difference between our Theorem 1 and Rockafellar's result is that our result is constructive — i.e., we can *efficiently* construct such a convex function whenever it exists. However, the proof technique used by Rockafellar, when adapted to our problem, will require $\Omega(T!)$ time to construct a feasible convex function. The efficiency of our approach is due to the aforementioned novel techniques in our proof, which is clearly not applicable in the abstract Banach space studied by Rockafellar. Moreover, our characterization of permutation stability is much simpler and intuitive than the cyclically monotonicity condition of Rockafellar.

Finally, we note that a close relative of the `FCF` Problem is the zeroth-order fitting question, which is perhaps more commonly seen in machine learning. That is, given variable sequence $\{\boldsymbol{x}_i\}_{i\in[T]}$ and *function value* sequence $\{z_i\}_{i\in[T]}$, when is it possible to find a convex function f such that $f(\boldsymbol{x}_i) = z_i$? We are not aware of previous studies on this question neither. For the reader's curiosity, in Appendix B we derive conditions on $\{\boldsymbol{x}_i\}_{i\in[T]}$ and $\{z_i\}_{i\in[T]}$ to decide whether the two sequence can be fitted by a convex function.

3.1 Proof of Theorem 1

The proof of necessity direction, given any convex function $f(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}$, we show that any sequence $\{\boldsymbol{x}_1, \cdots, \boldsymbol{x}_T\}$ and $\{\boldsymbol{p}_1, \cdots, \boldsymbol{p}_T\}$ where $\boldsymbol{p}_i \in \partial f(\boldsymbol{x}_i), \forall i \in [T]$ are permutation stable, is relatively easy, we refer the reader to Appendix C for a complete proof.

Proof of Sufficiency

Much more involved is the proof of the other direction of Theorem 1, i.e., to prove that permutation stability implies the existence of a convex function f that fits the given sequences. Our proof is constructive as shown in Algorithm 1. At a high-level, we construct such a convex function as follows. For any i, we consider a linear function $l_i(\boldsymbol{x}) = \boldsymbol{p}_i \cdot (\boldsymbol{x} - \boldsymbol{x}_i) + c_i$ where $\boldsymbol{p}_i, \boldsymbol{x}_i$ are from the given sequence and c_i is the only parameter to be determined. The convex function we will construct is precisely $f(\boldsymbol{x}) = \max_{i\in[T]} l_i(\boldsymbol{x})$, i.e., the maximum of T linear functions. The maximum of linear functions is known to be convex. What remains is that with carefully chosen parameters c_i's, the constructed f indeed satisfies $\boldsymbol{p}_i \in \partial f(\boldsymbol{x}_i)$. Specifically, key to this argument is to prove that under permutation stability there always exists c_i's such that $l_i(\boldsymbol{x}_i) = f(\boldsymbol{x}_i) = \max_{i\in[T]} l_i(\boldsymbol{x})$, i.e., $l_i(\boldsymbol{x}_i) \geq l_j(\boldsymbol{x}_i)$ for any $i \neq j$. That is, when $\boldsymbol{x} = \boldsymbol{x}_i$, $\max_{i\in[T]} l_i(\boldsymbol{x})$ achieves the maximum at $l_i(\boldsymbol{x})$. Consequently, the gradient of $l_i(\boldsymbol{x})$ at $\boldsymbol{x} = \boldsymbol{x}_i$ (i.e., $\boldsymbol{p}_i$) will be a subgradient to $f(\boldsymbol{x})$ at $\boldsymbol{x} = \boldsymbol{x}_i$, completing the proof.

The remainder of this proof is thus devoted to prove that permutation stability implies the existence of c_i's such that $l_i(\boldsymbol{x}_i) \geq l_j(\boldsymbol{x}_i)$ for any $i \neq j$. This can be formulated as a linear feasibility problem. The main challenge of the proof is to prove that permutation stability of the given sequences implies feasibility of the linear system. Our argument features an elegant connection to *network flow decomposition* and *permutation*.

Our starting point is to formalize the existence of c_i's as a linear feasibility problem. Recall that $l_i(\boldsymbol{x}_i) = \boldsymbol{p}_i \cdot (\boldsymbol{x}_i - \boldsymbol{x}_i) + c_i = c_i$ and $l_j(\boldsymbol{x}_i) = \boldsymbol{p}_j \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j) + c_j$. Therefore, the constraints $l_i(\boldsymbol{x}_i) \geq l_j(\boldsymbol{x}_i)$ becomes $c_i - c_j \geq \boldsymbol{p}_j \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j)$. The desirable c_i's exist if the following linear system is feasible.

$$c_i - c_j \geq \boldsymbol{p}_j \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j), \quad \text{for } i \neq j. \tag{2}$$

Consequently, the desired convex function can be constructed by Algorithm 1.

```
Algorithm 1: Construction of the FCF Convex Function
Input: $X = [\boldsymbol{x}_1 \cdots \boldsymbol{x}_T]$, $P = [\boldsymbol{p}_1, \cdots, \boldsymbol{p}_T]$
Function main():
    Solve the following linear system to find any feasible $C = [c_1 \cdots c_T]$:
        $c_i - c_j \geq \boldsymbol{p}_j \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j) \quad \forall i, j \in [T]; i \neq j$
    Construce $T$ linear function $l_1 \cdots l_T$ where
        $l_i(\boldsymbol{x}) = \boldsymbol{p}_i \cdot (\boldsymbol{x} - \boldsymbol{x}_i) + c_i, \forall i \in [T]$
    Return function
        $f(\boldsymbol{x}) = \max_{i\in[T]} l_i(x)$
```

So far we have not seen any connection to permutation sequences yet. The key step of our proof is to instead investigate the dual program of the above linear system (2) — with dual variable $y_{i,j}$ for the primal constraint with respect to i, j — via the *Farkas' lemma* (Farkas 1902). Our major insight is to realize that the dual program can be interpreted as *network flows* where: (1) each $y_{j,i}$ can be interpreted as directed flow from node j to node i; (2) dual constraints are precisely the flow conservation constraints, plus an additional constraint with coefficients depending on $\{\boldsymbol{x}_i\}_{i\in[T]}$ and $\{\boldsymbol{p}_i\}_{i\in[T]}$.

Here then comes the crux of the proof. The well-known *flow decomposition theorem* (Williamson 2019) says that any feasible flow $\{y_{i,j}\}_{i\neq j}$ can be decomposed into *cycle flows*. Notably, any permutation can also be decomposed into cycles (e.g., $(2, 1, 5, 3, 4)$, as a permutation of $(1, 2, 3, 4, 5)$, can be viewed as two cycles: $1 \to 2 \to 1$ and $3 \to 5 \to 4 \to 3$). Leveraging this connection, we are able to prove via a somewhat sophisticated argument that any feasible flow must violate the additional linear constraint when the permutation stability is satisfied. This shows that the dual program is infeasible, which implies the feasibility of the primal program (2) by Farkas' lemma. We refer the reader to the detailed proof of this theorem in Appendix C.

3.2 Efficient Verification of Permutation Stability

The permutation stability condition fully characterizes FCF, a key challenge in verifying this condition is that it requires the comparison between all the $T!$ permutations. Fortunately, it turns out that carefully designed algorithm can efficiently verify permutation stability in polynomial time, as shown in the following proposition.

Proposition 1. *Checking whether any two vector sequences $\{\boldsymbol{p}_i\}_{i\in[T]}$ and $\{\boldsymbol{x}_i\}_{i\in[T]}$ satisfy the* permutation stability *condition* (1) *or not can be computed in polynomial time.*

The key idea is to reduce the verification of permutation stability to compute the maximum weighted matching of a carefully constructed bipartite graph. It is well-known that bipartite matching can be solved efficiently in polynomial time, e.g., by solving LPs. This proves our proposition. Detailed proof can be seen in Appendix D

4 Learning Convex Functions via FCF

Abstractly, most machine learning problems look to generate a hypothesis function that fits the input data. Naturally, FCF can also be employed as a procedure to generate a hypothesis function, i.e., the output convex function, to fit any given data sequence. In this section, we study how FCF and its efficient computation can be employed to efficiently learn convex functions.

PAC-Learning of Gradients. To formally study the learnability problem, we adopt the well-known Probably Approximately Correct (PAC) learning framework. Specifically, suppose vector sequence $\{\boldsymbol{x}_i\}_{i\in[T]}$ in $\mathbb{R}^d$ are independent and identically distributed where each $\boldsymbol{x}_i$ is drawn from distribution $\mathcal{D}$ and the corresponding sequence $\{\boldsymbol{p}_i\}_{i\in[T]}$ is generated by a ground-truth convex function f such that $\boldsymbol{p}_i \in \partial f(\boldsymbol{x}_i)$. We examine the natural learning question of generating a hypothesis function $h \in \mathcal{H}$ that is "close" to f. We first restrict both the hypothesis class $\mathcal{H}$ and concept class $\mathcal{C}$ to be the set of all convex functions.

To describe the learning objective, we remark that since our input data does not contain the zeroth order information, i.e., function values, it is generally impossible to learn from $\{\boldsymbol{x}_i\}_{i\in[T]}$ and $\{\boldsymbol{p}_i\}_{i\in[T]}$ to fit the function values.[1] Therefore, in out setting, the more natural objective will also be learning to gradients of the original function f. We thus aim at finding a hypothesis $h \in \mathcal{H}$ that minimize the expected discrepancy with high probability, or formally

$$\text{Objective:} \quad \Pr_{\boldsymbol{x}\sim\mathcal{D}}[\|\nabla f(\boldsymbol{x}) - \nabla h(\boldsymbol{x})\|_2 \geq \varepsilon] \leq \delta \qquad (3)$$

where $\|\cdot\|_2$ is the l_2 norm of a vector. Similar to the standard PAC-learning framework, we are interested in identifying the number of samples needed in order to guarantee Objective (3) with high probability. Notably, we will always focus on continuous distribution $\mathcal{D}$. In this case, the above objective is well-defined even for non-differentiable convex functions, since all convex functions is differentiable almost everywhere and the measure of non-differentiable points is zero and thus has no effect in Objective (3).

FCF as Empirical Risk Minimization. A natural approach for learning is to employ FCF as a empirical risk minimization procedure that, given data sequence $\{\boldsymbol{x}_i\}_{i\in[T]}$ and $\{\boldsymbol{p}_i\}_{i\in[T]}$ as input, outputs a function h that first-order fits these data, i.e., achieving the minimum *empirical* risk. Note that, by our assumption, the ground truth is convex and thus FCF will be able to find a convex h that perfectly fits the given data with 0 empirical risk (in classification, this is also known as the *separable* case).

Obviously, it is generally difficult to learn the gradients of an *arbitrary* convex function since without any additional structure, it will essentially need the learner to query the gradient at every point in the entire feasible region. In the next two subsections, we show that efficient sample complexity can be derived for two special class of convex functions. We then demonstrate the application of PAC learning of gradients in the last subsection.

4.1 Learning Linear Separable Convex Functions

Linear separable functions are defined as $f(\boldsymbol{x}) = \sum_{i=1}^{d} f_i(x_i)$. In this section, we show an algorithm that outputs any gradient that is consistent with the observations, i.e. an empirical risk minimization algorithm, learns the ground truth efficiently. Moreover the sample complexity of any empirical risk minimization algorithm is optimal up to a factor of poly(ε, δ). Let $m_{\mathcal{H}}(\varepsilon, \delta)$ be the sample complexity of learning $h \in \mathcal{H}$ with error ε and confidence $1 - \delta$, we show:

Theorem 2. *The sample complexity of learning $h \in \mathcal{H}$ with error ε and confidence $1 - \delta$ is*

$$m_{\mathcal{H}}(\varepsilon, \delta) = O\left(d\frac{2ln(\frac{1}{\varepsilon}) + ln(\frac{1}{\varepsilon} + 1) + ln(\frac{1}{\delta})}{\varepsilon^2}\right)$$

[1] For example, for any learned hypothesis h, adding any constant to h will shift the function value but will not change its gradient at any point.

Throughout this subsection, we assume that the gradient space is in $\mathcal{P} = [0,1]^d$ without loss of generality (we can always normalize the space). A hypothesis takes as input the function's variable value $\boldsymbol{x} \in \mathbb{R}^d$, which it uses to choose a gradient outcome. We denote the gradient as $\boldsymbol{p} \in \mathcal{P}$, which is an element of an infinite set. We learn the linear separable function $f(\boldsymbol{x})$ by learning each dimension $f_i(x_i)$ separately. In each dimension, we propose covering the gradient space using a $\varepsilon-$discretized set $\mathcal{P}_\varepsilon$ over $[0,1]$, by which we mean a finite set cover of points in $[0,1]$ such that for all $p \in [0,1]$, there exists a point $p' \in \mathcal{P}_\varepsilon$ such that $\|p - p'\|_2 \leq \varepsilon$. More concretely, we let $\mathcal{P}_\varepsilon = \left\{0, \frac{1}{\lceil 1/\varepsilon \rceil}, \frac{2}{\lceil 1/\varepsilon \rceil}, \cdots, 1\right\}$. We prove this theorem by posing a multi-class PAC learning problem to fit the function at a discrete set of values. The key to our proof is to argue that any input data sequences satisfying the permutation stability must have Natarajan dimension (Natarajan 1989) at most $1/\varepsilon$. We defer the formal proof to Appendix E.

4.2 Learning k-piecewise linear convex functions

When d is greater than 1, the learning becomes more complicated. it is generally difficult to learn the gradients of an *arbitrary* convex function since without any additional structure. However, when the function is k-piecewise linear convex, we show that with just gradient information, polynomial samples are enough to learn the function gradient given arbitrary confidence and error. The hypothesis class $\mathcal{H}$ and concept class $\mathcal{C}$ are the set of k-piecewise linear convex functions. We now present two learning results based on the kind of data provided to the learner. We first consider the case where the learner is only provided with gradient information.

Theorem 3. *Let samples $\boldsymbol{x}, \boldsymbol{p}$ be selected from distribution $\mathcal{D}$ where $\boldsymbol{p} = \nabla f(\boldsymbol{x})$. Given ε and δ, an inferred hypothesis $h \in \mathcal{H}$ can be constructed with*

$$m_{\mathcal{H}}(\varepsilon, \delta) = O\left(\frac{k^3}{\varepsilon^2}\left(\log k + \log \frac{1}{\delta}\right)\right) \tag{4}$$

samples such that

$$P[P_{\mathcal{D}}(\nabla h(\boldsymbol{x}) \neq \nabla f(\boldsymbol{x})) \geq \varepsilon] \leq \delta \tag{5}$$

One may wonder since the ground-truth function is the max of k piecewise linear function, would the learning simply only need to sample the point and corresponding gradient (i.e., coefficients of the linear function) until at least one point is sampled from each region. We remark that the learning is more intricate than this since we also need to learn the boundaries where these k regions intersect. This explains why our sample complexity in the above theorem is larger than k/ε^2.

It turns out that when we have access to the zeroth order information during the learning process, we can indeed improve the sample complexity since the function value can help us very quickly identify where the k regions intersect. We attach the proofs for both of these results in Appendix F for completeness of the result.

Theorem 4. *Let samples $\boldsymbol{x}, (y, \boldsymbol{p})$ be selected from distribution $\mathcal{D}$ where $y = f(\boldsymbol{x})$ and $\boldsymbol{p} = \nabla f(\boldsymbol{x})$ is given by the true function f. Given ε and δ, an inferred hypothesis h can be constructed with*

$$m_{\mathcal{H}}(\varepsilon, \delta) = O\left(\frac{k}{\varepsilon}\left(\log k + \log \frac{1}{\delta}\right)\right) \tag{6}$$

samples such that

$$P[P_{\mathcal{D}}(h(\boldsymbol{x}) \neq f(\boldsymbol{x}) \vee \nabla h(\boldsymbol{x}) \neq \nabla f(\boldsymbol{x})) \geq \varepsilon] \leq \delta \tag{7}$$

4.3 Economic Applications of Gradient Learning

As we have seen in the revealed preferences literature, the seller wants to learn and predict the buyer's behavior from their purchase history. It turns out this kind of scenario also exists in other economic problems besides the pricing problem. In this section, we show that our PAC learning via FCF technique can be applied to learn and predict the agent's behavior in some typical economic scenarios.

In a pricing game, there is a single buyer who repeatedly buys a bundle of d types goods $\boldsymbol{x} = \{x_t\}_{t \in [d]} \in \mathbb{R}^d$, with x_t denotes the specific amount of good t purchased by the buyer from the seller. Let $X \subseteq \mathbb{R}^d$ be the set of feasible goods. The buyer has a *private* concave value function $v(\boldsymbol{x})$ over the goods. On each round i, the seller posts a price $\boldsymbol{p}_i \in \mathbb{R}^d$. Then the buyer purchase a bundle $\boldsymbol{x}_i$ of goods. Then the buyer receives an instantaneous utility of $v(\boldsymbol{x}_i) - \boldsymbol{p}_i \cdot \boldsymbol{x}_i$. A rational buyer would buy the bundle $\boldsymbol{x}_i$ that maximizes their utility again the posted price $\boldsymbol{p}_i$ at round i:

$$\boldsymbol{x}_i = \operatorname*{argmax}_{\boldsymbol{x}} v(\boldsymbol{x}) - \boldsymbol{x} \cdot \boldsymbol{p}_i$$

which indicates $\nabla v(\boldsymbol{x}_i) = \boldsymbol{p}_i$. Note as (Beigman and Vohra 2006) have showed that without any other assumptions on the utility function besides concavity, the sample complexity of learning the utility function is infinite. Though there is no hope for efficiently learning the general utility function and predict the buyer's behavior, our previous results still shows that we can learn and predict the buyer's behavior well when there is only a single good for sell (Amin, Rostamizadeh, and Syed 2013, 2014; Devanur, Peres, and Sivan 2014; Immorlica et al. 2017; Vanunts and Drutsa 2019) or when the buyer's utility function is the Leontief-type piecewise linear function (R. G. D. 1967), both settings have been welled adopted in the literature.

Similar analysis can be applied to the routing game to learn and predict the traffic flow, and to the contract theory problem (principal-agent game) to learn and predict the agent's effort level. See Appendix G for more description about the other economic applications of our techniques.

5 Application of FCF in Convex Optimization

In this section, we demonstrate another potential application of FCF, i.e., accelerating convex optimization, by presenting a set of thorough *empirical studies*. Our promising empirical results give rise to an intriguing future research direction of rigorously understanding how FCF can accelerate convex optimization. We note that this is outside of the scope of the present paper which aims at studying the FCF problem itself and demonstrating its potential usefulness.

The study of efficiently minimizing a convex function has a long history and is also of significant importance especially given today's large machine learning models (see a recent survey by Bubeck (2015)). Among various approaches for accelerating convex optimization, one widely used technique is to use a "surrogate" function to approximate the original convex function and then minimize the (hopefully much simpler) surrogate function (Lange, Hunter, and Yang 2000; Mairal et al. 2010; Lee and Seung 2000; Mairal 2013). Intuitively, a good surrogate should: (1) be easy to optimize; (2) approximate the gradients of the original function well. The former requirement makes the computation efficient whereas the later requirement makes sure that the surrogate can roughly preserve the optimal solution since the optimal solution has the smallest gradient.

To use FCF for convex minimization, we observe that the convex function identified by FCF is a natural candidate for the surrogate of the original convex function, defined as follows.

Definition 2. (FCF Surrogate) *For any convex function $f(\boldsymbol{x})$ and any sequence of data $\{\boldsymbol{x}_i\}_{i\in[T]}$ and $\{\boldsymbol{p}_i = \nabla f(\boldsymbol{x}_i)\}_{i\in[T]}$, the function output by Algorithm 1 is called an* FCF *surrogate.*

Recall that the proof of Theorem 1 guarantees that feasible $c_1, \cdots, c_T$ always exists for Equation (2) and thus FCF *surrogate* always exists for convex $f(\boldsymbol{x})$. When there are many feasible values for $\{c_i\}_{i\in[T]}$, one particular choice is the optimal solution to the following linear program, with variables $\{c_i\}_{i\in[T]}$ and η, which maximizes the minimum gap between the highest and the second highest hyperplanes at different data points (note $c_i = l_i(\boldsymbol{x}_i)$ and $\boldsymbol{p}_j \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j) + c_j = l_j(\boldsymbol{x}_i)$):

$$\boxed{\begin{aligned} \max \quad & \eta \\ \text{s.t.} \quad & c_i - \big[\boldsymbol{p}_j \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j) + c_j\big] \geq \eta, \text{ for } i \neq j \end{aligned}} \qquad (8)$$

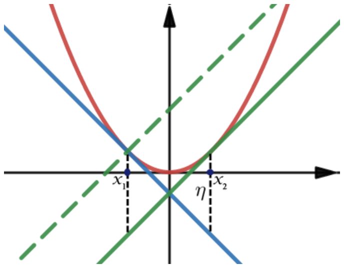

Figure 1: An example when $T = 2$. We want to avoid finding the surrogate formed by the dashed green line and blue line which makes $\eta = 0$. If so, we can't find new optimal point by optimizing the FCF surrogate function. On the other hand, the FCF surrogate formed by the solid green line and blue line is a good surrogate function to optimize.

Since FCF surrogate $\max_{i\in[T]} l_i(\boldsymbol{x})$ is the maximum of T linear functions, minimizing this convex function can be reduced to the a simple linear program with variable $z, \boldsymbol{x}$ — minimizing z, subject to $z \geq l_i(\boldsymbol{x})$ for all $i \in [T]$ — which can be solved efficiently by LP solvers. Moreover, though our learnability results in the previous section are not applicable here due to the violation of i.i.d. data sample assumption, their conceptual messages that the FCF surrogate can approximate the gradients of the original function provide good reasons for the usefulness of the FCF surrogate. This insight is also demonstrated in our extensive experiments. Specifically, we consider Algorithm 2, denoted as **S**urrogate **M**inimization using **C**onvex **F**itting (SMCF). The algorithm simply uses the current data to compute the the FCF surrogate $g(\boldsymbol{x})$ [2] and then minimize $g(\boldsymbol{x})$ by solving a linear program until the gradient of the underlying function $f(\boldsymbol{x})$ has small enough norm (i.e. smaller than parameter ε).

Algorithm 2: **S**urrogate **M**inimization using **C**onvex **F**itting

```
parameter: resolution $\epsilon = e^{-4}$, integer $k$
Function SMCF():
    Initialize: pick $k$ random samples $\boldsymbol{x}_1, \cdots, \boldsymbol{x}_k$
    Let $X = \{\boldsymbol{x}_1, \cdots, \boldsymbol{x_k}\}$, $P = \{\nabla f(\boldsymbol{x}_1), \cdots, \nabla f(\boldsymbol{x}_k)\}$
    Let $\widehat{\boldsymbol{x}} = \boldsymbol{x}_k$
    while $\|\nabla f(\widehat{\boldsymbol{x}})\|_2^2 \leq \varepsilon$ do
        Compute FCF surrogate $g(\boldsymbol{x})$ using available data from $X, P$ with $\{c_i\}$ solved by LP (8)
        Minimize the FCF surrogate to compute the
            $\widehat{\boldsymbol{x}} = \operatorname{argmin}_{\boldsymbol{x}\in X} g(\boldsymbol{x})$
        Add $\widehat{\boldsymbol{x}}, \nabla f(\widehat{\boldsymbol{x}})$ to $X, P$ accordingly
    end
    Return $\widehat{\boldsymbol{x}}$
```

FCF **surrogate with zeroth order information.** Notably, the FCF surrogate generally cannot closely approximate the function value (i.e., zeroth order information) since it only relies on first-order information of gradients.[3] Nevertheless, zeroth order information are not essential for convex minimization[4], though it could be helpful if available (e.g., the celebrated gradient with backtracking line search method due to Armijo (1966)). Specifically, if additionally we happen to also have the zeroth order information, i.e., $\{f(\boldsymbol{x}_i)\}_{i=1}^T$, it is easy to show that $c_i = f(\boldsymbol{x}_i), \forall i$ is a feasible solution to Equation (2) as well. In this case, we can similarly plug in this FCF surrogate into Algorithm 2, leading to SMCF with *0*'th order information, or FMCF0.

5.1 Experiment Setup

Optimization task. We consider large scale experiments for optimizing the loss of logistic regression over data points $\{(\boldsymbol{w}_s, y_s)\}_{s\in[N]}$ where $\boldsymbol{w}_s$ is the data feature and y_s is the

[2] A simple trick we used in our implementation to accelerate computation is to use only the last 100 data points to construct the surrogate since data points that are far before become redundant as the sequence converges to the minimum.

[3] For instance, any constant shift of the function shall not change the FCF surrogate at all.

[4] For instance, the vanilla gradient descent works without zeroth order information

Name	SMCF0	SMCF	QS	MISO1	MISO2	GD	SGD	GD-B
Zeroth order information	Yes	No	Yes	Yes	Yes	No	No	Yes

Table 1: Different algorithms' requirements of zeroth order information. "Yes" means zeroth order information is required

corresponding label. Given a set of N data points, logistic regression looks for the parameters $\boldsymbol{x} \in \mathbb{R}^d$ which minimizes the following convex loss function:

$$f(\boldsymbol{x}) = \min_{\boldsymbol{x} \in \mathbb{R}^d} \left[\frac{1}{N} \sum_{s=1}^{N} \log(1 + e^{-y_s \cdot \boldsymbol{w}_s \cdot \boldsymbol{x}}) + \lambda \psi(\boldsymbol{x}) \right] \quad (9)$$

where $\psi(\boldsymbol{w})$ is a convex regularization function. Our experiment uses the standard l_2 norm as regularization.

Benchmarks. We compare with various classic optimization algorithms including: (**1**) gradient descent (GD); (**2**) stochastic gradient descent (SGD); (**5**) GD with *backtracking line search* (GD-B) that dynamically sets the step size according to the zeroth order information. Additionally, we also compare our algorithms with similar surrogate-based algorithm. Specifically, (Mairal 2013) proved convergence guarantees (to the optimal solution) for surrogate-based convex minimization algorithms when the surrogate functions are carefully chosen. Most prominent versions of their algorithms are: (**4**) quadratic surrogates (QS); (**5**) incremental scheme using first-order surrogate with two variants, MISO1 and MISO2, designed particularly for functions with additive form like that in (9).[5] Some of these algorithms require zeroth order information, while some do not. See Table 1 for a summary.

Data sets and computing system. We use two classical datasets: (1) *covtype*[6] from UCI which has $N = 581,012$ data points and $d = 54$ features dimensions; (2) *ijcnn1*[7] which has $N = 49,990$ data points and $d = 22$ features dimensions. All the algorithms are coded in Python and the experiments were run on a single core of a 2.20GHz Intel Xeon Silver 4210 CPU using 256 GB of RAM.

5.2 Experimental Results

In this section, we present the experiment results for the larger dateset *covtype* in Table 2. All entries are averaged over 50 trials, unless performance is significantly worse than the best algorithm or cannot be solved within 24 hours (indicated as "N/A" in tables). Each set of experiments has three regularization regimes, high ($\lambda = 0.1$), medium ($\lambda = 0.01$), and low ($\lambda = 0.001$). Though Table 2 only presents two regularization regimes, the entire experimental results for *covtype* and *ijcnn1* with three regularization regimes are presented in Appendix H.

Name	Time (s)	Opt. Value
SMCF0	**5247** ± 1967	**0.00133** ± 0.00018
SMCF	**4634** ± 2455	**0.00136** ± 0.00015
QS	67052	0.00003
MISO1	0.819 ± 0.103	0.3380 ± 0.0
MISO2	0.807 ± 0.011	0.3380 ± 0.0
GD	N/A	N/A
SGD	N/A	N/A
GD-B	N/A	N/A

(a) $\lambda = 0.1$

Name	Time (s)	Opt. Value
SMCF0	**7962** ± 5801	**0.00112** ± 0.00040
SMCF	**4023** ± 783	**0.00400** ±0.00105
QS	83090	0.00003
MISO1	0.868 ± 0.025	0.3380 ± 0.0
MISO2	0.757 ± 0.137	0.3380 ± 0.0
GD	N/A	N/A
SGD	N/A	N/A
GD-B	N/A	N/A

(b) $\lambda = 0.01$

Table 2: *covtype* Dataset: Experimental results with different regularization regimes.

The advantage of our approach is quite evident in both sets of experiments. Specifically, for experiments on *covtype* dateset in Table 2, both SMCF and SMCF0 can achieve close to optimal solution and run efficiently. The MISO1 and MISO2, which is the primary algorithm used for experiments in (Mairal 2013), converges very fast but the solution quality of their objective values are not satisfactory, i.e., a bit far from optimality.[8] The surrogate-based QS algorithm can achieve solution quality but takes at least 10 times more time than our algorithm. Finally, GD/SGD/GD-B cannot finish within 24 hours. Finally, one interesting observation is that SMCF0 with zeroth order information is not necessarily always faster than SMCF. This is because the careful choice of $\{c_i\}_{i \in [T]}$ solved by LP (8) in SMCF may lead to faster convergence (and thus less number of iterations) than directly using the function values for $\{c_i\}_{i \in [T]}$.

Finally, the advantages of our methods in the *ijcnn1* dataset is similar, except that GD/SGD/GD-B now can solve the optimization problem due to the smaller size of this dataset. It is also worth to mention that GD-B is indeed much faster than GD and SGD because of the usage of zeroth order information. Nevertheless, SMCF, which doesn't require zeroth order information of the objective function, performs much better than GD-B.

[5] Code available at http://spams-devel.gforge.inria.fr/

[6] https://archive.ics.uci.edu/ml/datasets/covertype

[7] http://www.geocities.ws/ijcnn/nnc_ijcnn01.pdf

[8] The value of 0.338 is the converged optimal value of their algorithms and can't be improved even if we continue running the algorithm. Moreover, in our experiments, we found MISOs are quite unstable with respect to the values of label y_i. When $y_i \in \{-1, 1\}$, the algorithms perform even worse. We used label $y_i \in \{0, 1\}$ in the experiments, which is a relatively better choice for them.

References

Amin, K.; Rostamizadeh, A.; and Syed, U. 2013. Learning prices for repeated auctions with strategic buyers. In *Advances in Neural Information Processing Systems*, 1169–1177.

Amin, K.; Rostamizadeh, A.; and Syed, U. 2014. Repeated contextual auctions with strategic buyers. In *Advances in Neural Information Processing Systems*, 622–630.

Armijo, L. 1966. Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1): 1–3.

Balcan, M.-F.; Daniely, A.; Mehta, R.; Urner, R.; and Vazirani, V. V. 2014. Learning economic parameters from revealed preferences. In *International Conference on Web and Internet Economics*, 338–353. Springer.

Beigman, E.; and Vohra, R. 2006. Learning from revealed preference. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, 36–42.

Bendavid, S.; Cesabianchi, N.; Haussler, D.; and Long, P. M. 1995. Characterizations of learnability for classes of $\{0,...,n\}$-valued functions. *Journal of Computer and System Sciences*, 50(1): 74–86.

Blum, A.; Haghtalab, N.; and Procaccia, A. D. 2014. Learning optimal commitment to overcome insecurity. In *Advances in Neural Information Processing Systems*, 1826–1834.

Bubeck, S.; et al. 2015. Convex Optimization: Algorithms and Complexity. *Foundations and Trends® in Machine Learning*, 8(3-4): 231–357.

Daniely, A.; Sabato, S.; Ben-David, S.; and Shalev-Shwartz, S. 2011. Multiclass learnability and the erm principle. In *Proceedings of the 24th Annual Conference on Learning Theory*, 207–232. JMLR Workshop and Conference Proceedings.

Devanur, N. R.; Peres, Y.; and Sivan, B. 2014. Perfect bayesian equilibria in repeated sales. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, 983–1002. SIAM.

Farkas, J. 1902. Ober die Theorie der einfachen Ungleichungen. *J. Reine Angew. Math*, 124: 1–24.

Immorlica, N.; Lucier, B.; Pountourakis, E.; and Taggart, S. 2017. Repeated sales with multiple strategic buyers. In *Proceedings of the 2017 ACM Conference on Economics and Computation*, 167–168.

Lange, K.; Hunter, D. R.; and Yang, I. 2000. Optimization transfer using surrogate objective functions. *Journal of computational and graphical statistics*, 9(1): 1–20.

Lee, D. D.; and Seung, H. S. 2000. Algorithms for Non-Negative Matrix Factorization. In *Proceedings of the 13th International Conference on Neural Information Processing Systems*, NIPS'00, 535–541.

Letchford, J.; Conitzer, V.; and Munagala, K. 2009. Learning and approximating the optimal strategy to commit to. In *International Symposium on Algorithmic Game Theory*, 250–262. Springer.

Mairal, J. 2013. Optimization with first-order surrogate functions. In *International Conference on Machine Learning*, 783–791. PMLR.

Mairal, J.; Bach, F.; Ponce, J.; and Sapiro, G. 2010. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(1).

Marecki, J.; Tesauro, G.; and Segal, R. 2012. Playing repeated stackelberg games with unknown opponents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 821–828.

Mas-Colell, A.; Whinston, M. D.; Green, J. R.; et al. 1995. *Microeconomic theory*, volume 1. Oxford university press New York.

Natarajan, B. K. 1989. On learning sets and functions. *Machine Learning*, 4(1): 67–97.

Peng, B.; Shen, W.; Tang, P.; and Zuo, S. 2019. Learning optimal strategies to commit to. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 2149–2156.

R. G. D., A. 1967. *Macro-economic theory: a mathematical treatment*. Springer.

Rockafellar, R. 1966. Characterization of the subdifferentials of convex functions. *Pacific Journal of Mathematics*, 17(3): 497–510.

Rockafellar, R. 1970. On the maximal monotonicity of subdifferential mappings. *Pacific Journal of Mathematics*, 33(1): 209–216.

Roth, A.; Slivkins, A.; Ullman, J.; and Wu, Z. S. 2020. Multidimensional dynamic pricing for welfare maximization. *ACM Transactions on Economics and Computation (TEAC)*, 8(1): 1–35.

Roth, A.; Ullman, J.; and Wu, Z. S. 2016. Watch and learn: Optimizing from revealed preferences feedback. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, 949–962.

Samuelson, P. A. 1938. A note on the pure theory of consumer's behaviour. *Economica*, 5(17): 61–71.

Vanunts, A.; and Drutsa, A. 2019. Optimal Pricing in Repeated Posted-Price Auctions with Different Patience of the Seller and the Buyer. In *Advances in Neural Information Processing Systems*, 939–951.

Varian, H. R. 2006. Revealed preference. *Samuelsonian economics and the twenty-first century*, 99–115.

Williamson, D. 2019. *Network Flow Algorithms*. Cambridge University Press. ISBN 9781107185890.

Zadimoghaddam, M.; and Roth, A. 2012. Efficiently learning from revealed preference. In *International Workshop on Internet and Network Economics*, 114–127. Springer.