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 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 (sub)gradient $\nabla f(\boldsymbol{x}_i)$ equals $\boldsymbol{p}_i$ for all $i \in [T] = \{1, \cdots, T\}$? Despite a basic question of convex analysis, `FCF` has surprisingly been overlooked in the past literature. With an efficient constructive proof, we provide 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

A natural application of *first-order convex fitting* (`FCF`) is in the theory of *revealed preferences*, which starts from the seminal work of (Samuelson 1938) and has formed a celebrated subfield of consumer theory (Varian 2006). 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 $v(\boldsymbol{x})$ that "explains" (a.k.a., *rationalizing*) the observed data $\{(\boldsymbol{p}_i, \boldsymbol{x}_i)\}_{i=1}^T$ assuming a rational utility-maximizing buyer. It turns out that, under the standard assumption of concave value function $v(\boldsymbol{x})$ and quasi-linear utility $v(\boldsymbol{x}) - \boldsymbol{x} \cdot \boldsymbol{p}$, the utility-maximizing buyer purchase $\boldsymbol{x}_i = \mathrm{argmax}_{\boldsymbol{x}}[v(\boldsymbol{x}) - \boldsymbol{x}\cdot\boldsymbol{p}]$ precisely satisfies $\boldsymbol{p}_i = \nabla v(\boldsymbol{x}_i)$ for any i (Roth, Ullman, and Wu 2016; Dawkins, Han, and Xu 2021). Therefore, data $\{(\boldsymbol{p}_i, \boldsymbol{x}_i)\}_{i=1}^T$ can be rationalized if and only if there is a concave value function $v(\boldsymbol{x})$ that fits the data in the *first-order* sense. Notably, we will primarily work with *convex* functions later due to the convention in convex analysis; however, all our results apply equally to *concave* functions, simply by negating the function.

Recent computational research has studied the problem of learning from revealed preferences but has focused on learning the revenue-optimal prices from data (Roth, Ullman, and Wu 2016; Roth et al. 2020). In contrast, this work aims at fitting, and learning, the underlying *value function* directly.

Given any data $(\boldsymbol{p}_1, \boldsymbol{x}_1), \cdots, (\boldsymbol{p}_T, \boldsymbol{x}_T)$, when can we efficiently construct a convex function so that its 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?

Notably, the above objective is also more aligned with the original motivation of the revealed preference literature, which is to explain consumer behaviors as opposed to learn optimal prices. Concretely, 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. $\nabla f(\boldsymbol{x}_i) = \boldsymbol{p}_i$. For learning-theoretic questions, suppose the sequence of $\{\boldsymbol{x}_i\}_{i\in[T]}$ is drawn from a fixed distribution $\mathcal{D}$, and the sequence $\{\boldsymbol{p}_i\}_{i\in[T]}$ is their first-order gradients generated according to some unknown convex $f(\boldsymbol{x})$. We seek to understand whether one can with high probability learn an approximately correct hypothesis convex function such that when a new $\boldsymbol{x}$ is drawn from the same distribution $\mathcal{D}$, the hypothesis correctly determines the function's gradient at $\boldsymbol{x}$ with small expected error.

1.1 Our Results and Techniques

We aim to characterize when a dataset of two vector sequences can be "first-order" fitted by an underlying convex function. Our first main result is a *sufficient and necessary* characterization of the question. That is, first-order convex fitting is possible if and only if the dataset is *permutation stable*, or formally, $\sum_{i=1}^T \boldsymbol{p}_i \cdot \boldsymbol{x}_i \geq \sum_{i=1}^T \boldsymbol{p}_i \cdot \boldsymbol{x}_{\sigma(i)}$ for any permutation σ of $[T]$. We provide a proof that can efficiently construct such a convex function to fit the given dataset. As an application of this result, we leverage the technique to accelerate convex optimization and propose an optimization scheme that experimentally matches or outperforms both state-of-the-art and classical solvers for large-scale optimization problems arising from machine learning.

Next, we turn to the learning-theoretic question. Suppose the dataset was generated by an unknown convex function. Our next set of results study the efficient learning of the underlying convex function so as to predict test data with small error and high confidence, i.e., in the PAC learning sense. Here, for both separable and piecewise linear convex functions, we prove *polynomial* sample complexity results. This illustrates that one can use our proposed method to provably rationalize the agents' behaviors and learn the agents' private utility information with high precision. We then illustrate how this can be used in various economic applications including learning from revealed preferences, traffic prediction and contract theory.

1.2 Related Work

Our work is motivated by the literature of revealed preferences started by Samuelson (1938). Classic research on revealed preference asks a similar question as us, i.e., 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). However, their buyer behaviour model differs slightly from us. They assume that the buyer has some budget B and would like to maximize value within budget, i.e. $\boldsymbol{x}_i = \text{argmax}_{\boldsymbol{x}\cdot\boldsymbol{p}_i \le B}\, u(\boldsymbol{x})$. This different buyer model turns out to lead to quite different answers from us. More related to our problem is the learning from revealed preferences under quasi-linear buyer utility model (Roth, Ullman, and Wu 2016; Roth et al. 2020). However, these works study how to learn the revenue-maximizing price for the seller, whereas we are learning the value function directly. Going beyond pricing problems, there has also been researches 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.

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 first consider any separable convex functions, i.e. the buyer's value over different goods are independent. What's more, we also consider a more general piecewise linear value function which doesn't need to be separable. We provide efficient sample complexity for the PAC-learning model in both cases.

2 The Problem of First-order Convex Fitting

We start with some preliminaries. Let $f(\boldsymbol{x}) : X \to \mathbb{R}$ 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 $f(\boldsymbol{x}') \ge f(\boldsymbol{x}) + \boldsymbol{p}\cdot(\boldsymbol{x}' - \boldsymbol{x})$. Function f is called *convex* if for any $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$ and any $\alpha \in [0,1]$ we have $\alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{x}') \ge f(\alpha\boldsymbol{x} + (1-\alpha)\boldsymbol{x}')$. 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*

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

$\{\boldsymbol{x}_i\}_{i\in[T]}$ 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).[1]

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, 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.

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]The full appendix of this paper can be found online.

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
        $f(\boldsymbol{x}) = \max_{i \in [T]} l_i(x)$
```

$(1, 2, 3, 4, 5)$, 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 $\boldsymbol{p}_i \in \nabla f(\boldsymbol{x}_i)$. 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.[2] 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 Separable Convex Functions

Separable convex 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*

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

[2]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 separable convex 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 Piecewise Linear Convex Functions

When the value function is not separable, 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*

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

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*

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

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 $\boldsymbol{p}_j \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j) + c_j = l_j(\boldsymbol{x}_i)$):

$$\boxed{\begin{array}{ll} \max & \eta \\ \text{s.t.} & c_i - \left[\boldsymbol{p}_j \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j) + c_j\right] \geq \eta, \text{ for } i \neq j \end{array}} \tag{8}$$

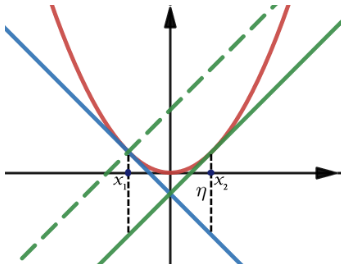

Figure 1: An example when $T = 2$. We want to avoid finding the surrogate formed by the dashed green line and blue line which makes $\eta = 0$. If so, we can't find new optimal point by optimizing the FCF surrogate function. On the other hand, the FCF surrogate formed by the solid green line and blue 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 $g(\boldsymbol{x})$ [3] 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.[4] Nevertheless, zeroth order information are not essential for convex minimization[5], 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

[3]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.

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

[5]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 $\psi(\boldsymbol{x})$ 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).[6] 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*[7] from UCI which has $N = 581,012$ data points and $d = 54$ features dimensions; (2) *ijcnn1*[8] 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.

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.[9] 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. 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.

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.

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

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

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

[9]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.

Acknowledgments

This work is supported by a Google Faculty Research Award and an NSF grant CCF-2132506.

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.

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.

Dawkins, Q.; Han, M.; and Xu, H. 2021. The Limits of Optimal Pricing in the Dark. *Advances in Neural Information Processing Systems*, 34.

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.

A Strict First-Order convex Fitting

A "stronger" version of Theorem 1 concerns the fitting with a *strictly convex* function $u(\mathbf{x})$. It turns out that the characterization in Theorem 1 naturally generalizes to this version. In particular, using a similar proof of Theorem 1, we can prove that there exists a strictly convex function $u(\mathbf{x})$ that first-order fits these two sequences, i.e., $\mathbf{p}_t = \nabla u(\mathbf{x}_t)$ for all $t \in [T]$, *if and only if* the two sequences are strictly permutation stable: $\sum_{t=1}^{T} \mathbf{p}_t \cdot \mathbf{x}_t > \sum_{t=1}^{T} \mathbf{p}_t \cdot \mathbf{x}_{\sigma(t)}$ for any non-identical permutation $\sigma \in \Sigma(T)$. The proof is mostly similar to the proof of Theorem 1, modulo some modifications to the weak and strong inequality labels. The only major difference is the use of a different version of the Farkas' lemma for strict inequalities.

The necessity proof is the same except that we need the strict convexity condition to obtain strict permutation stability, as follows:

$$u(\mathbf{x}_t) + \mathbf{p}_t \cdot (\mathbf{x}_{t'} - \mathbf{x}_t) < u(\mathbf{x}_{t'}), \forall t' \in [T] \tag{10}$$

The proof of sufficiency also follows a similar procedure, except that we now require that there always exists c_t's such that $l_t(\mathbf{x}_t) = u(\mathbf{x}_t) = \max_{t\in[T]} l_t(\mathbf{x})$ in a stronger sense: $l_t(\mathbf{x}_t) > l_{t'}(\mathbf{x}_t)$ for any $t \neq t'$. That is, when $\mathbf{x} = \mathbf{x}_t$, $l_t(\mathbf{x})$ is the only hyperplane that achieves $\max_{t\in[T]} l_t(\mathbf{x})$ and thus the gradient at $\mathbf{x}_t$ is unique, i.e., $\mathbf{p}_t$. Note that $u(\mathbf{x})$ is not a strictly convex function yet. However, one can easily "smooth" it by maintaining the gradients at all $\mathbf{x}_t$ whereas make other parts strictly convex.

What remains is to prove that strict permutation stability implies the existence of c_t's such that $l_t(\mathbf{x}_t) > l_{t'}(\mathbf{x}_t)$ for any $t \neq t'$. The proof also uses Farkas' lemma and flow decomposition. However, here we need a slightly different version of Farkas' lemma to count for the strict inequalities. We were not able to find this version of Farkas' lemma, so we state formally the lemma as follows and also give a proof for completeness.

Lemma A.1. *Let $A \in \mathbb{R}^{m\times n}$ be a real matrix and $\mathbf{b} \in \mathbb{R}^m$ be a vector. Then exactly one of the following linear system is feasible:*

1. *There exists $\mathbf{x} \in \mathbb{R}^n$ such that $A\mathbf{x} < \mathbf{b}$;*
2. *There exists $\mathbf{y} \in \mathbb{R}^m$ such that $A^T\mathbf{y} = \mathbf{0}, \mathbf{b}^T\mathbf{y} \leq 0, \sum_{i=1}^{m} y_i = 1,$ and $\mathbf{y} \geq 0$.*

Proof. Note that system (1) is feasible if and only if there exists large enough $t \geq 1$ such that $A\mathbf{x} + \frac{1}{t}\mathbf{1} \leq \mathbf{b}$, or equivalently

$$tA\mathbf{x} + \mathbf{1} \leq t\mathbf{b}. \tag{11}$$

Note that t here is viewed as a large constant and the only variable in Inequality (11) is $\mathbf{x}$. Now utilizing the standard Farkas' lemma (a slight variant of (C.1)), we know that among Linear System (11) and the following linear system, exactly one of them is feasible: $tA^T\mathbf{y} = \mathbf{0}, (t\mathbf{b} - \mathbf{1})^T\mathbf{y} < 0, \mathbf{y} \geq 0$. We slightly re-arrange this linear system as follows:

$$A^T\mathbf{y} = \mathbf{0}, \tag{12}$$

$$\mathbf{b}^T\mathbf{y} < \frac{\sum_i y_i}{t} \tag{13}$$

$$\mathbf{y} \geq 0 \tag{14}$$

If linear system (11) is feasible for some large t, then linear system (12)-(14) is infeasible, which implies that the second linear system in the stated lemma is infeasible as well since it is more constrained (i.e., any of its feasible solutions is also feasible to linear system (12)-(14)).

If the first linear system in the stated lemma is infeasible, then linear system (11) is infeasible for any $t \geq 1$ which implies that linear system (12)-(14) is feasible for any $t \geq 1$. Therefore, the second linear system in the stated lemma must be feasible. This follows an argument via contradiction. Suppose the second linear system in lemma is infeasible, that is, for any $\mathbf{y} \in \mathbb{R}^m$ such that $A^T\mathbf{y} = \mathbf{0}, \sum_{i=1}^{m} y_i = 1,$ and $\mathbf{y} \geq 0$, we always have $\mathbf{b}^T\mathbf{y} > 0$. Then for any *non-zero* $\mathbf{y} \in \mathbb{R}^m$ such that $A^T\mathbf{y} = \mathbf{0}$, and $\mathbf{y} \geq 0$, we always have $\mathbf{b}^T\mathbf{y} > 0$ and thus there always exists a large t to make $\mathbf{b}^T\mathbf{y} < \frac{\sum_i y_i}{t}$ infeasible, which contradicts the fact that linear system (12)-(14) is feasible for any $t \geq 1$. Clearly, $\mathbf{y} = \mathbf{0}$ does not make linear system (12)-(14) feasible neither. This contradiction shows that the second linear system in lemma must be feasible. □

With Lemma A.1, the remainder just mimics the proof of Theorem 1.

B Feasibility of Zero'th-Order convex Fitting

In this section, we consider a close relative to the convex fitting problem as considered in Theorem 1. However, instead of considering the first-order gradients, we are given the function values (i.e., zero'th order information). Specifically, given any two finite sequences $\{\mathbf{x}_t\}_{t\in[T]}$ where $\mathbf{x}_t \in \mathbb{R}^d$ and $\{z_t\}_{t\in[T]}$ where $z_t \in \mathbb{R}$, when is it possible to construct a convex function u such that $u(\mathbf{x}_t) = z_t$ for all $t = 1, \cdots, T$? We term this problem the *zero'th order convex fitting* problem. We also provide a necessary and sufficient characterization for this question, though we are not aware of any nice interpretation for this characterization any more and the proof is also much simpler.

Theorem B.1. *[Feasibility of Zero'th Order convex Fitting] For any two finite sequences $\{\mathbf{x}_t\}_{t\in[T]}$ where $\mathbf{x}_t \in \mathbb{R}^d$ and $\{z_t\}_{t\in[T]}$ where $z_t \in \mathbb{R}$, there exists a convex function $u(\mathbf{x})$ that "fits" these two sequences — i.e., $u(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t} = z_t$ for all $t \in [T]$ —* if and only if *the following linear system with variables $\{\mathbf{w}_t\}_{t\in[T]}$ is feasible:*

$$(\mathbf{x}_t - \mathbf{x}_{t'}) \cdot \mathbf{w}_t \geq z_t - z_{t'}, \quad \textit{for } t \neq t'. \tag{15}$$

Proof. First of all, we start the proof from the direction that if the linear system (15) is feasible, then there exists a desirable convex function $u(\mathbf{x})$. We provide a constructive proof for the above theorem, and the way to construct the convex function is as follows. First of all, we define $l_t(\mathbf{x}) = \mathbf{w}_t \cdot (\mathbf{x} - \mathbf{x}_t) + z_t$ for any $t = 1, \cdots, T$ where

$\mathbf{x}_t$ and z_t are from the given sequences and $\mathbf{w}_t$ is the only parameter to be determined. Then we construct the convex function as $u(\mathbf{x}) = \max_{t\in[T]} l_t(\mathbf{x})$. It's known that the maximum of linear functions is convex, so what remains is carefully choosing $\mathbf{w}_t$ such that the constructed $u(\mathbf{x})|_{\mathbf{x}=\mathbf{x}_t} = z_t$ which is equivalent to $\max_{t\in[T]} l_t(\mathbf{x_t}) = l_t(\mathbf{x_t}) = z_t$, i.e. $l_t(\mathbf{x_t}) \geq l_{t'}(\mathbf{x_t})$ for any $t' \neq t$.

Next, we formalize the existence of $\mathbf{w}_t$'s as a linear feasibility problem. Recall that $l_t(\mathbf{x}_t) = \mathbf{w}_t \cdot (\mathbf{x}_t - \mathbf{x}_t) + z_t = z_t$ and $l_{t'}(\mathbf{x}_t) = \mathbf{w}_{t'} \cdot (\mathbf{x}_t - \mathbf{x}_{t'}) + z_{t'}$. Therefore, the constraint $l_t(\mathbf{x}_t) \geq l_{t'}(\mathbf{x}_t)$ becomes $z_t \geq \mathbf{w}_{t'} \cdot (\mathbf{x}_t - \mathbf{x}_{t'}) + z_{t'}$ which is equivalent to $(\mathbf{x}_{t'} - \mathbf{x}_t) \cdot \mathbf{w}_{t'} \geq z_{t'} - z_t$. The desirable $\mathbf{w}_t$'s exists if the following Linear System is feasible:

$$(\mathbf{x}_t - \mathbf{x}_{t'}) \cdot \mathbf{w}_t \geq z_t - z_{t'}, \quad \text{for } t \neq t'. \tag{16}$$

proving our claim that if the linear system (15) is feasible, there exists a desirable convex function $u(\mathbf{x}) = \max_{t\in[T]} l_t(\mathbf{x})$.

On the other hand, what remains to prove in Theorem B.1 is the necessity condition, i.e. if there exists a convex function $u(\mathbf{x})$ that "fits" the given two sequences $\{\mathbf{x}_t\}_{t\in[T]}$ and $\{z_t\}_{t\in[T]}$, then we must have the linear system (15) is feasible. To prove this, define the convex function $u(\mathbf{x})$ such that $u(\mathbf{x}_t) = z_t, \forall t \in [T]$, then let $\mathbf{w}_t = \nabla u(\mathbf{x}_t)$ be any element of the supergradient of $u(\mathbf{x})$ at $\mathbf{x}_t$. Thus, by the definition of the convexity, we have

$$u(\mathbf{x}_t) + \mathbf{w}_t \cdot (\mathbf{x}_{t'} - \mathbf{x}_t) \leq u(\mathbf{x}_{t'}), \quad \text{for any } t \neq t' \tag{17}$$

Substituting $u(\mathbf{x}_t)$ with z_t from (17), we can get

$$(\mathbf{x}_t - \mathbf{x}_{t'}) \cdot \mathbf{w}_t \geq z_t - z_{t'}, \quad \text{for any } t \neq t' \tag{18}$$

which implies that system (15) is feasible, proving the statement. □

C Proof for the Characterization of FCF

Theorem C.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.*

C.1 Proof of Theorem C.1

Proof of Necessity

We start from the easier direction, i.e., the necessity of permutation stability. 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. This follows from the definition of subgradients. In particular, we have for any i

$$f(\boldsymbol{x}_i) + \boldsymbol{p}_i \cdot (\boldsymbol{x} - \boldsymbol{x}_i) \leq f(\boldsymbol{x}), \forall \boldsymbol{x} \in X \tag{19}$$

Let $\sigma \in \Sigma_T$ be any non-trivial permutation over $[T]$. Instantiating Inequality (19) with $\boldsymbol{x} = \boldsymbol{x}_{\sigma(i)}$ and summing over the inequalities over i we have

$$\begin{aligned}\Sigma_{i\in[T]}\, \boldsymbol{p}_i \cdot \left(\boldsymbol{x}_{\sigma(i)} - \boldsymbol{x}_i\right) &\leq \Sigma_{i\in[T]} \left(f(\boldsymbol{x}_{\sigma(i)}) - f(\boldsymbol{x}_i)\right) \\ &= \Sigma_{i\in[T]} f(\boldsymbol{x}_{\sigma(i)}) - \Sigma_{i\in[T]} f(\boldsymbol{x}_i) \\ &= 0.\end{aligned}$$

This implies, $\Sigma_{i\in[T]}\, \boldsymbol{p}_i \cdot \boldsymbol{x}_{\sigma(i)} \leq \Sigma_{i\in[T]} \boldsymbol{p}_i \cdot \boldsymbol{x}_i$ for any permutation σ, proving the necessity of permutation stability.

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.

$$c_i - c_j \geq \boldsymbol{p}_j \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j), \quad \text{for } i \neq j. \tag{20}$$

Consequently, the desired convex function can be constructed by Algorithm 3.

Algorithm 3: 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 $C = [c_1 \cdots c_T]$:
        $c_i - c_j \geq \boldsymbol{p}_j \cdot (\boldsymbol{x}_i - \boldsymbol{x}_j) \quad \forall i,j \in [T]; i \neq j$
    Construe $T$ 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
        $f(\boldsymbol{x}) = \max_{i\in[T]} l_i(x)$
```

It turns out that directly arguing the feasibility of Linear System (2) is difficult. Therefore, the second step of the proof is to transform the feasibility of (2) to its dual space via the *Farkas' lemma* (Farkas 1902).

Lemma C.1. *[Farkas' Lemma] Let $A \in \mathbb{R}^{m\times n}$ be a real matrix and $\boldsymbol{b} \in \mathbb{R}^m$ be a vector. Then exactly one of the following linear system is feasible:*

1. *There exists $\boldsymbol{x} \in \mathbb{R}^n$ such that $A\boldsymbol{x} \le \boldsymbol{b}$;*
2. *There exists $\boldsymbol{y} \in \mathbb{R}^m$ such that $A^T\boldsymbol{y} = \boldsymbol{0}$, $\boldsymbol{b}^T\boldsymbol{y} < 0$, and $\boldsymbol{y} \ge 0$.*

Utilizing Lemma C.1, we reduce the feasibility of linear system (2) to the infeasibility of the following linear system:

$$\sum_{j\neq i} y_{i,j} - \sum_{j\neq i} y_{j,i} = 0 \qquad \text{for any } i \in [T] \qquad (21)$$

$$y_{i,j} \ge 0 \qquad \text{for any } i \neq j \qquad (22)$$

$$\sum_{j\neq i} \boldsymbol{p}_j(\boldsymbol{x}_i - \boldsymbol{x}_j)\cdot y_{i,j} > 0 \qquad \text{for any } i \in [T] \qquad (23)$$

Interestingly, we observe that Constraint (21) and (22) can be interpreted as the feasibility constraints of flow on the directed complete graph. In particular, consider the *directed complete graph $K_T = (V, E)$* with T nodes. For any $i, j \in [T]$, there is a directed edge (i, j) pointing from node i to j. We can interpret $y_{i,j}$ as the flow going through edge (i, j). Constraint (21) is precisely the flow conservation condition at node i — the flow amount $\sum_{j\neq i} y_{j,i}$ entering i equals the flow amount $\sum_{j\neq i} y_{i,j}$ existing i. Constraint (22) simply means that the flow must be non-negative.

Here comes the crux of our proof. We now leverage the classic *flow decomposition theorem* (Williamson 2019) to show that any $\{y_{i,j}\}_{i\neq j}$ satisfying Constraints (21) and (22) must violate Constraint (23), implying the infeasibility of linear system (21)-(23). The flow decomposition theorem says that any feasible flow satisfying conservation constraints at all nodes can be decomposed into at most $T(T-1)$ *cycles*. That is, for any $\{y_{i,j}\}_{i\neq j}$ satisfying Constraints (21) and (22) can be decomposed into $m(\le T(T-1))$ cycles: cycle $\mathcal{C}^k$, represented as $i_1^k \to i_2^k \to \cdots \to i_{l^k}^k$ where $i_l^k \in [T]$ is a node and $i_{l^k}^k = i_1^k$, has flow amount $y^k > 0$. That is, any feasible flow $\{y_{i,j}\}_{i\neq j}$ can be decomposed such as

$$y_{i,j} = \sum_{k:(i,j)\in\mathcal{C}^k} y^k \qquad (24)$$

for any directed edge (k, k'). Here, with slight abuse of notation, we use $(i, j) \in \mathcal{C}^k$ and $i \in \mathcal{C}^k$ to denote edge (i, j) and node i is in the cycle.

Crucially, each cycle $\mathcal{C}^k : i_1^k \to i_2^k \to \cdots \to i_{l^k}^k$ naturally corresponds to a permutation σ^k by mapping each node to its *parent*. Formally, σ^k is defined as follows: $\sigma^k(i_{l+1}^k) = i_l^k$ for $l < l^k$ and $\sigma^k(i) = i$ for any $i \notin \mathcal{C}^k$. By permutation stability with respect to σ^k, we thus have $\sum_{i=1}^T \boldsymbol{p}_i \cdot (\boldsymbol{x}_i - \boldsymbol{x}_{\sigma^k(i)}) \ge 0$, which implies

$$\sum_{l=1}^{l^k-1} \boldsymbol{p}_{i_{l+1}^k} \cdot (\boldsymbol{x}_{i_{l+1}^k} - \boldsymbol{x}_{i_l^k}) \ge 0 \qquad (25)$$

for all cycles $\mathcal{C}^k$ where $k = 1, \cdots, m$.

It turns out that Inequality (25) leads to contradiction to Constraint (23), derived as follows:

$$\begin{aligned}
0 &< \sum_{i=1}^T \sum_{i\neq j} \boldsymbol{p}_j(\boldsymbol{x}_i - \boldsymbol{x}_j)\cdot y_{i,j} \\
&= \sum_{i,j\in[T]:j\neq i} \boldsymbol{p}_j(\boldsymbol{x}_i - \boldsymbol{x}_j)\cdot \sum_{k:(i,j)\in\mathcal{C}^k} y^k \\
&= \sum_{k=1}^m \sum_{(i,j)\in\mathcal{C}^k} \boldsymbol{p}_j(\boldsymbol{x}_i - \boldsymbol{x}_j)\cdot y^k \\
&= \sum_{k=1}^m \left[\sum_{l=1}^{l^k-1} \boldsymbol{p}_{i_{l+1}^k}(\boldsymbol{x}_{i_l^k} - \boldsymbol{x}_{i_{l+1}^k}) \right]\cdot y^k \\
&\le \sum_{k=1}^m 0\cdot y^k \\
&= 0
\end{aligned}$$

which is a contradiction. This proves that any $\{y_{t,j}\}_{i\neq j}$ satisfying Constraints (21) and (22) must violate Constraint (23). This shows that Linear System (21) (23) is infeasible and thus Linear System (2) is feasible.

D Proof for Efficient Verification of Permutation Stability

Proposition D.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.*

Proof. First, observe that the permutation stability can be reduced to computing the σ^* that maximizes the term $\sum_{i=1}^T \boldsymbol{p}_i \cdot \boldsymbol{x}_{\sigma(i)}$ over all permutations, i.e., $\sigma^* = \text{argmax}_{\sigma\in\Sigma_T} \sum_{i=1}^T \boldsymbol{p}_i \cdot \boldsymbol{x}_{\sigma(i)}$. After computing σ^*, we know that the condition is satisfied if and only if $\sum_{i=1}^T \boldsymbol{p}_i \cdot \boldsymbol{x}_{\sigma^*(i)} = \sum_{i=1}^T \boldsymbol{p}_i \cdot \boldsymbol{x}_i$.

Next, we show that σ^* can indeed be computed efficiently by computing the maximum weighted matching of a carefully constructed bipartite graph. Consider a complete bipartite graph $G = ([T] \cup [T], E)$ where both sides have nodes indexed by $i \in [T]$. Each edge $e = (i, j) \in E$ has weight $(\boldsymbol{p}_i \cdot \boldsymbol{x}_j)$ for any $i, j \in [T]$. It is not difficult to see that each bipartite matching corresponds to a permutation σ ($\sigma_i = j$ if i on the left is matched to j on the right) and the weight it has is $\sum_{i=1}^T \boldsymbol{p}_i \cdot \boldsymbol{x}_{\sigma(i)}$. Therefore, the matching that maximizes total weight is precisely the permutation that maximizes $\sum_{i=1}^T \boldsymbol{p}_i \cdot \boldsymbol{x}_{\sigma(i)}$, as desired. It is well-known that bipartite matching can be solved efficiently in polynomial time, e.g., by solving LPs. This concludes the proof of this proposition. □

E Proofs for Learning One-Dimensional Convex Functions

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

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

Proof. We prove this theorem by posing a multi-class PAC learning problem to fit the function at a discrete set of values. First of all, we investigate the *Natarajan dimension* of a hypothesis class $\mathcal{H}$, as defined below:

Definition E.1 (Natarajan dimension (Natarajan 1989))**.** *The Natarajan dimension of $\mathcal{H}$, denoted as $d_N(\mathcal{H})$, is the maximal size of a shattered set $C \subset X$. A set C is shattered by $\mathcal{H}$ if there exists two functions $f_0, f_1 : C \to \mathcal{P}_\varepsilon$ such that*

- *For every $x \in C$, $f_0(x) \neq f_1(x)$*
- *For every $B \subset C$, there exists a function $h \in \mathcal{H}$ such that*

$$\forall x \in B, h(x) = f_0(x) \text{ and } \forall x \in C/B, h(x) = f_1(x)$$

The Natarajan dimension is a generalization of the VC dimension to multiclass predictors. In additon, let $m_{\mathcal{H}}(\varepsilon,\delta)$ denote the sample complexity of learning $\mathcal{H}$ with error ε and confidence $1-\delta$. With these definitions, an important theorem in multiclass PAC learning is shown by Ben-David et al. (Bendavid et al. 1995) and Daniely et al. (Daniely et al. 2011) is the following:

Theorem E.2 ((Bendavid et al. 1995; Daniely et al. 2011), rephrased)**.** *For every concept class $\mathcal{H} \subseteq [\mathcal{P}_\varepsilon]^{\mathcal{X}}$*

$$m_{\mathcal{H}}(\varepsilon,\delta) = O\Big(\frac{d_N(\mathcal{H})\big(ln(\frac{1}{\varepsilon}) + ln(|\mathcal{P}_\varepsilon|) + ln(d_N(\mathcal{H})) + ln(\frac{1}{\delta})\big)}{\varepsilon}\Big)$$

where $d_N(\mathcal{H})$ is the Natarajan dimension of $\mathcal{H}$ and $k = |Y_\varepsilon|$. This upper bound is attained by any ERM algorithm.

Using this result, we can derive an upper bound on the sample complexity for our hypothesis class $\mathcal{H}$. From Theorem E.2, we know the sample complexity is determined by the Natarajan dimension of the hypothesis class $\mathcal{H}$. Next, we show the Natarajan dimension of our hypothesis class:

Lemma E.1. *Let $d = 1$. Then the Natarajan dimension of $\mathcal{H}$ as defined above has an upper bound $d_N(\mathcal{H}) \leq \frac{1}{\varepsilon}$.*

Proof. Let $S = \{x_1, \cdots x_n\}$ be the set of unique inputs and let $T \subseteq S$ be any subset of S. Let $f_0, f_1 : \mathbb{R}^d \to \mathcal{P}_\varepsilon$ be functions such that $f_0(x_i) \neq f_1(x_i)$ for all $\{x_i\}_{i\in[n]}$, and let $h(x_i) = f_0(x_i)$ if $x_i \in T$ and $h(x_i) = f_1(x_i)$ otherwise. Now, because this is a special case with $d = 1$, we note the following relationship implied by the permutation stability constraint.

$$x_i > x_j \implies h(x_i) \geq h(x_j)$$

Let S be sorted such that if $i < j$ then $x_i < x_j$. Suppose $f_0(x_1) = 0$ and $f_1(x_1) = \varepsilon$. In other words, the two smallest values in $\mathcal{P}_\varepsilon$. Note that we have the following constraints on $f_0(x_2)$ and $f_1(x_2)$ by permutation stability over h

$$\begin{aligned} f_0(x_2) &\geq f_0(x_1) \\ f_0(x_2) &\geq f_1(x_1) \\ f_1(x_2) &\geq f_1(x_1) \\ f_1(x_2) &\geq f_0(x_1) \\ f_1(x_2) &\neq f_0(x_2) \end{aligned}$$

Now, note that if $f_0(x_2) = 0$ or $f_1(x_2) = 0$, these constraints are violated. Because $f_0(x_2) \geq \varepsilon$ and $f_1(x_2) \geq \varepsilon$, at least one of $f_0(x_2)$ or $f_1(x_2)$ must be a value $y_2 \in \mathcal{P}_\varepsilon$ such that $y_2 \neq 0$, $y_2 \neq \varepsilon$ and $f_0(x_2) \neq f_1(x_2)$. In other words, it must be a new value from the hypothesis class such that $y_2 > \varepsilon$. If we extend this argument, we get that for each $x_i \in S$, $f_0(x_i)$ or $f_1(x_i)$ must take on a value of $y_i \in \mathcal{P}_\varepsilon$ such that for all $x_j < x_i$, $y_i > f_0(x_j)$ and $y_i > f_1(x_j)$. Thus, if $|S| >= |\mathcal{P}_\varepsilon|$, f_0 and f_1 cannot exist. This gives

$$d_N(\mathcal{H}) \leq \frac{1}{\varepsilon}$$

□

Thus, Theorem E.2 and Lemma E.1 let us prove our main Theorem 2 on the sample complexity. □

F Proofs for learning k-piecewise convex functions

When d is greater than 1, the learning becomes more complicated. We hypothesize that we need infinite number of samples to learn the function. However, when the function is k-piecewise linear convex, we show that polynomial samples are enough to probably learn the approximately correct function. Additionally, the construction methods for the learned hypotheses match the construction methods presented in the our convex optimization algorithms.

First, we define a few preliminaries. Let examples be drawn from input space $x \in X = \mathbb{R}^d$ according to distribution $\mathcal{D}$. As stated in the main paper, to guarantee the existence of $\nabla f(\boldsymbol{x})$ for some $\boldsymbol{x}$ drawn from $\mathcal{D}$, we only consider continuous distributions on $\mathcal{D}$. The proof methods shown here, however, are extensible to additional distributions with some modification. Once again, we let the hypothesis space $\mathcal{H}$ and concept class $\mathcal{C}$ be the set of all k-piecewise linear convex functions. This class of function can be uniquely defined by the following property.

$$\begin{aligned} &\forall f \in \mathcal{C}, \exists L \in \mathcal{P}(\mathbb{R}^{d+1}) \\ &\qquad \text{s.t. } f(\boldsymbol{x}) = \max_{(y,\boldsymbol{p})\in L} \boldsymbol{p}\cdot\boldsymbol{x} + y \end{aligned}$$

In other words, every element of $\mathcal{C}$ can be written as the maximum over a set of k hyperplanes in $\mathbb{R}^{d+1}$. We denote this set of planes as L_f for $f \in C$. Additionally, for each $i \in [k]$, let $r_i \subseteq X$ denote the subset of inputs such that $l_i(\boldsymbol{x}) = f(\boldsymbol{x})$ if and only if $\boldsymbol{x} \in r_i$. We note that each region r_i must be a convex polytope in $\mathbb{R}^d$ due to the convexity of

f. Define the full set of regions as R_f such that $\bigcup_{r_i \in R_f} r_i = X$. Finally, we note that the set of n-piecewise linear convex functions is a subset of $\mathcal{C}$ for any $n \leq k$.

Theorem F.1. *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*

$$m_{\mathcal{H}}(\varepsilon, \delta) = O\left(\frac{k^3}{\varepsilon^2}\left(\log k + \log \frac{1}{\delta}\right)\right) \tag{26}$$

samples such that

$$P[P_{\mathcal{D}}(\nabla h(\boldsymbol{x}) \neq \nabla f(\boldsymbol{x})) \geq \varepsilon] \leq \delta \tag{27}$$

Proof. We first show the construction of h' in terms of a given set of samples m. Define $C = [c_1 \cdots c_m]$ as a sequence of arbitrary constants and solve for any C such that the following holds.

$$c_i - c_j \geq \boldsymbol{p_j} \cdot (\boldsymbol{x_i} - \boldsymbol{x_j}), \forall i \neq j$$

After solving for C, construct m linear function $l_1 \cdots l_m \in L$ where

$$l_i(\boldsymbol{x}) = \boldsymbol{p_i} \cdot (\boldsymbol{x} - \boldsymbol{x_i}) + c_i$$

And define $h'(\boldsymbol{x}) = \max_{i \in [m]} l_i(\boldsymbol{x})$. Now, uppose two samples $\boldsymbol{x_i}, \boldsymbol{x_j}$ fall in the same region $r \in R_f$. The constraints on c_i and c_j give the following.

$$\boldsymbol{p_j} \cdot (\boldsymbol{x_i} - \boldsymbol{x_j}) \leq c_i - c_j \leq \boldsymbol{p_i} \cdot (\boldsymbol{x_i} - \boldsymbol{x_j})$$

Because $\boldsymbol{x_i}, \boldsymbol{x_j} \in r_z$, we get $\boldsymbol{p_i} = \boldsymbol{p_j}$. This gives $c_i - c_j = \boldsymbol{p_i} \cdot (\boldsymbol{x_i} - \boldsymbol{x_j})$, which can be rewritten as

$$c_i - \boldsymbol{p_i} \cdot \boldsymbol{x_i} = c_j - \boldsymbol{p_j} \cdot \boldsymbol{x_j}$$

Thus showing that $l_i = l_j$ in the construction of h' if both samples are from the same region. Then, because there are at most k hyperplanes that can be constructed from a sample, $h' \in \mathcal{H}$ must hold. The remainder of the proof involves first finding the number of samples we need per region (denoted as m'), followed by the number of samples needed to guarantee m' samples per region, m.

Let $S_{\epsilon/k} \subseteq R_f$ denote the subset of regions such that $\forall s_i \in S_{\epsilon/k}, P_{\mathcal{D}}(x \in s_i) \geq \epsilon/k$. Consider two different regions $s_i, s_j \in S_{\epsilon/k}$, and let there be two samples $\boldsymbol{x_i} \in s_i$ and $\boldsymbol{x_j} \in s_j$. Furthermore, denote the two hyperplanes constructed from these samples as l_i and l_j respectively and let $\boldsymbol{p_i}$ and $\boldsymbol{p_j}$ denote the gradients for regions s_i and s_j. If we assume $l_i(\boldsymbol{x}) \leq l_j(\boldsymbol{x})$, then we get the following.

$$\begin{aligned}\boldsymbol{p_i} \cdot (\boldsymbol{x} - \boldsymbol{x_i}) + c_i &\leq \boldsymbol{p_j}(\boldsymbol{x} - \boldsymbol{x_j}) + c_j \\ (\boldsymbol{p_j} - \boldsymbol{p_i}) \cdot \boldsymbol{x} &\geq (c_i - c_j) + \boldsymbol{p_j}\boldsymbol{x_j} - \boldsymbol{p_i}\boldsymbol{x_i} \\ (\boldsymbol{p_j} - \boldsymbol{p_i}) \cdot \boldsymbol{x} &\geq (\boldsymbol{p_j} - \boldsymbol{p_i}) \cdot \boldsymbol{x_i}\end{aligned}$$

Now, let X_{s_i} be the set of all samples in $[m]$ such that $\forall x \in X_{s_i}, x \in s_i$. Then, for any admissible choice of C and any choice of $\boldsymbol{x}$, $l_i(\boldsymbol{x}) \leq l_j(\boldsymbol{x})$ holds only if $(\boldsymbol{p_j} - \boldsymbol{p_i}) \cdot \boldsymbol{x} \geq \max_{\boldsymbol{x'} \in X_{s_i}} (\boldsymbol{p_j} - \boldsymbol{p_i}) \cdot \boldsymbol{x'}$. Conversely, if

$$(\boldsymbol{p_j} - \boldsymbol{p_i}) \cdot \boldsymbol{x} < \max_{\boldsymbol{x'} \in X_{s_i}} (\boldsymbol{p_j} - \boldsymbol{p_i}) \cdot \boldsymbol{x'}$$

holds, then $l_i(\boldsymbol{x}) > l_j(\boldsymbol{x})$ must be true. If for every region $r_t \in R$, $l_i(\boldsymbol{x}) > l_t(\boldsymbol{x})$ holds for $\boldsymbol{x}$, then $\nabla h'(\boldsymbol{x}) = \nabla f(\boldsymbol{x})$. Now, for each region $r_t \in R$, define d_t^i such that the following conditions hold.

$$P_{\mathcal{D}}((\boldsymbol{p_j} - \boldsymbol{p_i}) \cdot \boldsymbol{x} < d_t^i \wedge \boldsymbol{x} \in s_i) = \epsilon/k^2$$

Note that d_t^i must exist because $s_i \in S_{\epsilon/k}$ and because the distribution $\mathcal{D}$ is continuous. Now, let F_t^i denote the event that none of m' samples in region s_i satisfy $(\boldsymbol{p_j} - \boldsymbol{p_i}) \cdot \boldsymbol{x} < d_t^i$. If F_t^i does not happen for any $r_t \in R$, then we are guaranteed that $P_{\mathcal{D}}(\nabla h'(\boldsymbol{x}) \neq \nabla f(\boldsymbol{x}) \wedge \boldsymbol{x} \in s_i) \leq \epsilon/k$. Note that the boundaries between regions are the only points where the gradient is undefined and because the boundaries are $\mathbb{R}^{d-1}$ dimensional, they have zero measure under $\mathcal{D}$. Now, we solve for m' such that this happens with probability $1 - \delta/2k$.

$$\begin{aligned}P_{\mathcal{D}}(\bigcup_{t \in [k]} F_t^i) &\leq \sum_{t \in [k]} P_{\mathcal{D}}(F_t^i) \\ &\leq \sum_{t \in [k]} (1 - \epsilon/k^2)^{m'} \\ &\leq k e^{-m'\epsilon/k^2}\end{aligned}$$

and by setting $k e^{-m'\epsilon/k^2} \leq \delta/2k$, we get $m' = \frac{k^2}{\epsilon}(2 \log k + \log \frac{2}{\delta})$. Thus, for any region $s_i \in S_{\epsilon/k}$, if at least m' samples fall within s_i, then $P_{\mathcal{D}}(h'(\boldsymbol{x}) \neq f(\boldsymbol{x}) \wedge \boldsymbol{x} \in s_i) \leq \epsilon/k$ with probability $1 - \delta/2k$. Now, assume for all $s_i \in S_{\epsilon/k}$, at least m' samples out of m fall within region s_i. We then get the following.

$$\begin{aligned}P_{\mathcal{D}}(\nabla h'(\boldsymbol{x}) \neq& \nabla f(\boldsymbol{x})) = \\ &\sum_{r_t \in R} P_{\mathcal{D}}(\nabla h'(\boldsymbol{x}) \neq \nabla f(\boldsymbol{x}) \wedge \boldsymbol{x} \in r_t) \\ &\leq \sum_{s_i \in S_{\epsilon/k}} P_{\mathcal{D}}(\nabla h'(\boldsymbol{x}) \neq \nabla f(\boldsymbol{x}) \wedge \boldsymbol{x} \in s_i) \\ &\quad + \sum_{r_t \in R \setminus S_{\epsilon/k}} \frac{\epsilon}{k} \\ &\leq \sum_{s_i \in S_{\epsilon/k}} \frac{\epsilon}{k} + \sum_{r_t \in R \setminus S_{\epsilon/k}} \frac{\epsilon}{k} = \epsilon\end{aligned}$$

Then, by applying the union bound on the confidence, we get that $P[P_{\mathcal{D}}(\nabla h'(\boldsymbol{x}) \neq \nabla f(\boldsymbol{x})) > \epsilon] \leq \sum_{s_i \in S_{\epsilon/k}} \frac{\delta}{2k} \leq \frac{\delta}{2}$. Put together, this gives

$$P[P_{\mathcal{D}}(\nabla h'(\boldsymbol{x}) \neq \nabla f(\boldsymbol{x})) \leq \epsilon] \geq 1 - \delta/2$$

Finally, to achieve the desired error with probability at least $1 - \delta$, we need at least m' samples per region $s_i \in S_{\epsilon/k}$ with probability at least $1 - \delta/2$. Define $\gamma = 1 - \frac{k}{\epsilon}\frac{m'}{m}$ and let $|X_{S_{\epsilon/k}}| = \min_{s_i \in S_{\epsilon/k}} |X_{s_i}|$. We then get the following.

$$\begin{aligned}
P(|X_{S_{\epsilon/k}}| \leq m') &\leq \sum_{s_i \in S_{\epsilon/k}} P(|X_{s_i}| \leq m') \\
&\leq \sum_{s_i \in S_{\epsilon/k}} P(|X_{s_i}| \leq (1-\gamma)\frac{\epsilon}{k} m) \\
&\leq k e^{-\frac{\epsilon}{k} m \gamma^2 /2} \\
&\leq k e^{-\frac{1}{2}\frac{\epsilon}{k} m} e^{m'} \\
1 - k e^{-\frac{1}{2}\frac{\epsilon}{k} m} e^{m'} &> 1 - \delta/2 \\
k e^{-\frac{1}{2}\frac{\epsilon}{k} m} e^{m'} &< \delta/2 \\
m &> \frac{2k}{\epsilon}(\log \frac{2k}{\delta} + m') \\
m &> \frac{2k}{\epsilon}(\log \frac{2k}{\delta} + \frac{k^2}{\epsilon}(2\log k + \log \frac{2}{\delta}))
\end{aligned}$$

The third inequality is an application of the Chernoff bound on a bernoulli random variable with $p = 1 - \epsilon/k$. □

As stated in the main body, it turns out that when we have access to the zero'th order information during the learning process, we can improve the sample complexity up to $O(\frac{k}{\epsilon}(\log k + \log \frac{1}{\delta}))$. The proof for this is shown here.

Theorem F.2. *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*

$$m_{\mathcal{H}}(\varepsilon, \delta) = O\left(\frac{k}{\varepsilon}\left(\log k + \log \frac{1}{\delta}\right)\right) \quad (28)$$

samples such that

$$P[P_{\mathcal{D}}(h(\boldsymbol{x}) \neq f(\boldsymbol{x}) \vee \nabla h(\boldsymbol{x}) \neq \nabla f(\boldsymbol{x})) \geq \varepsilon] \leq \delta \quad (29)$$

Proof. First, we describe the construction of h' in terms of a set of m samples. Let $\boldsymbol{x_i}, (y_i, \boldsymbol{p_i})$ be a sample and output pair for $i \in [m]$. This pair forms the hyperplane $l_i(x) = (\boldsymbol{x} - \boldsymbol{x_i}) \cdot \boldsymbol{p_i} + y_i$. Now, construct h' as follows.

$$h'(x) = \max_{i \in [m]} l_i(\boldsymbol{x}) \quad (30)$$

Recall that for each $i \in [k]$, we define region $r_i \in R_f$ such that $r_i \subseteq \mathbb{R}^d$ defines a convex polytope with unique and constant gradient. As a result, h' can have at most k unique gradient values and must be k-piecewise linear convex, or in other words, we get $h' \in \mathcal{H}$.

Let region r_i have total probability mass p_i according to distribution $\mathcal{D}$. If for any $j \in [m]$, x_j falls within region r_i, then hyperplane l_i is in the set of planes for h'. Furthermore, for every $x \in r_i$, $h'(x) = f(x)$ because the set of unique hyperplanes used to construct h' is a subset of those used to define f. Because we are assuming continuous $\mathcal{D}$ and the boundaries between regions are in $\mathbb{R}^{d-1}$, for any $\boldsymbol{x} \in X$ such that $\nabla f(\boldsymbol{x})$ or $\nabla h'^{(}\boldsymbol{x})$ has zero probability of being sampled. As a result, all of the probability mass p_i is captured by h' assuming $l_i \in L_{h'}$. Finally, let $S_{\epsilon/k}$ denote the set of regions that have probability mass $p_i \geq \epsilon/k$. We then get the following.

$$\begin{aligned}
P(\exists i \in S_{\epsilon/k}, i \notin [m]) &\leq \sum_{j \in S_{\epsilon/k}} (1 - p_j)^m \\
&\leq k(1 - \frac{\epsilon}{k})^m \\
P(\forall i \in S_{\epsilon/k}, i \in [m]) &\geq 1 - k(1 - \frac{\epsilon}{k})^m \\
&\geq 1 - k e^{-\frac{\epsilon}{k} m} > 1 - \delta \\
e^{-\frac{\epsilon}{k} m} &< \frac{\delta}{k} \\
\frac{\epsilon}{k} m &> \log k + \log \frac{1}{\delta} \\
m &> \frac{k}{\epsilon}(\log k + \log \frac{1}{\delta})
\end{aligned}$$

□

G Additional Economic Applications of Gradient Learning

Application I: Estimating Routing Traffic. A routing game involves a single agent who wants to route some goods from their sources to their destinations. Specifically, a routing game $\mathcal{G}(G, \phi)$ is defined by a graph $G(V, E)$ and a *private* convex latency function ϕ over the traffic. Let $d = |E|$ be the number of edges in the graph. The agent's routing decision induces a d-dimension traffic flow $\boldsymbol{f} = \{f_e\}_{e \in E} \in \mathbb{R}^d$, with f_e denotes the specific amount of traffic routed on every edge $e \in E$. Let $F \subseteq \mathbb{R}^d$ be the set of feasible flows. In addition, there is an authority who has the power to impose constant tolls $\boldsymbol{\tau} = \{\tau_e\}_{e \in E} \in \mathbb{R}^d$ on every edge. On each round t, the authority impose tolls $\boldsymbol{\tau}_t$ over the edges of the network. Then the agent chooses the aggregated traffic flow $\boldsymbol{f}_t \in F$ to route the goods. The agent suffers from a total loss of $\phi(\boldsymbol{f}_t) + \sum_e \tau_{t,e} \cdot f_{t,e}$. A rational agent would minimize their loss against the posted tolls $\boldsymbol{\tau}_t$ at each round:

$$\boldsymbol{f}_t = \operatorname*{argmax}_{\boldsymbol{f} \in F} -\phi(\boldsymbol{f}) - \boldsymbol{\tau}_t \cdot \boldsymbol{f}$$

which indicates $\nabla \phi(\boldsymbol{f}_t) = \boldsymbol{\tau}_t$.

Application II: Predicting Agent Efforts in Contract Design. In a contract theory problem, the principal (the leader) defines a contract by which the agent (the follower) will be paid, as a function of work produced by the agent. Suppose the principal has m products $M = \{1, \cdots, m\}$ for the agent to produce and the agent's outcome space will have size exponential in m. Each outcome S is a subset of M representing the items in M produced by the agent. On each round t, the agent can make some effort e_t to work and produce probabilities $q_{t,1}, \cdots, q_{t,m} \in [0, 1]$ for all the m items. Since the agent knows how effort is stochastically mapped to realizations, we abstract away the agent's choice of an "effort" level, and instead view the agent as choosing a "target contribution" – the expected probability of the agent's ultimate contribution. Thus, we consider a 2^M-dimensional principal-agent problem, in which the agent produces work $\boldsymbol{w} \in \mathbb{R}^{2^M}$ where each w_i represents the probability of all the items in subset $S_i \subseteq M$ are produced. The agent experiences some *private* convex costs $c(\boldsymbol{w})$ for producing a target contribution of $\boldsymbol{w}$. On each round t, the principal posts the contract $\boldsymbol{y}_t$ and a rational agent would produce work $\boldsymbol{w}_t$ that maximizes their utility:

$$\boldsymbol{w}_t = \underset{\boldsymbol{w}}{\operatorname{argmax}}\, \boldsymbol{y}_t \cdot \boldsymbol{w} - c(\boldsymbol{w})$$

which indicates $\nabla c(\boldsymbol{w}_t) = \boldsymbol{y}_t$

H Experimental Results

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) $\lambda = 0.01$

Name	Time (s)	Opt. Value
SMCF0	**6230** ± 3842	**0.03449** ± 0.03366
SMCF	**4503** ± 1659	**0.00058** ± 0.00066
QS	82555	0.00003
MISO1	0.887 ± 0.033	0.3380 ± 0.0
MISO2	0.933 ± 0.037	0.3380 ± 0.0
GD	N/A	N/A
SGD	N/A	N/A
GD-B	N/A	N/A

(c) $\lambda = 0.001$

Table 3: *covtype* Dataset: Experimental results with different regularization regimes.[9]

Name	Time (s)	Opt. Value
SMCF0	**249** ± 51	**0.00127** ± 0.00056
SMCF	**240** ± 136	**0.00183** ± 0.00060
QS	763 ± 101	0.00036 ± $5e^{-20}$
MISO1	12.4 ± 0.101	0.1839 ± 0.0
MISO2	0.926 ± 0.009	0.1839 ± 0.0
GD	24211	0.00044
SGD	22895	0.00049
GD-B	12970	0.00044

(a) $\lambda = 0.1$

Name	Time (s)	Opt. Value
SMCF0	**197** ± 59	**0.00118** ± 0.00064
SMCF	**217** ± 97	**0.00136** ± 0.00066
QS	764 ± 93	0.00028 ± $5e^{-20}$
MISO1	11 ± 0.244	0.1832 ± 0.0
MISO2	0.939 ± 0.009	0.1832 ± 0.0
GD	26333	0.00036
SGD	23634	0.00036
GD-B	13177	0.00036

(b) $\lambda = 0.01$

Name	Time (s)	Opt. Value
SMCF0	**178** ± 61	**0.00228** ± 0.00262
SMCF	**217** ± 117	**0.00209** ± 0.00340
QS	770 ± 97	0.00027 ± 0.0
MISO1	12 ± 0.131	0.1831 ± 0.0
MISO2	0.928 ± 0.0046	0.1831 ± 0.0
GD	27910	0.00035
SGD	23214	0.00035
GD-B	13321	0.00035

(c) $\lambda = 0.001$

Table 4: *ijcnn1* Dataset: Experimental results with different regularization regimes.