Emulating the Expert: Inverse Optimization through Online Learning

Andreas Bärmann [*1] **Sebastian Pokutta** [*2] **Oskar Schneider** [*1]

Abstract

In this paper, we demonstrate how to learn the objective function of a decision maker while only observing the problem input data and the decision maker's corresponding decisions over multiple rounds. Our approach is based on online learning techniques and works for linear objectives over arbitrary sets for which we have a linear optimization oracle and as such generalizes previous work based on KKT-system decomposition and dualization approaches. The applicability of our framework for learning linear constraints is also discussed briefly. Our algorithm converges at a rate of $\mathcal{O}(\frac{1}{\sqrt{T}})$, and we demonstrate its effectiveness and applications in preliminary computational results.

1. Introduction

Human decision makers are very good at making decisions under rather imprecise specification of the decision-making problem both in terms of constraints as well as objective. One might argue that the human decision maker can pretty reliably learn from observed previous decisions – a traditional learning-by-example setup. At the same time, when we try to turn these decision-making problems into actual optimization problems, we often run into all types of issues in terms of specifying the model. In an optimal world, we would be able to *infer or learn* the optimization problem from previously observed decisions taken by an expert.

This problem naturally occurs in many settings where we do not have direct access to the decision maker's preference or objective function but we can observe her behaviour and the learner as well as the decision maker have access to the same information. Natural examples are as diverse as making recommendations based on user history and strategic planning problems, where the agent's preferences are unknown but the system is observable. Other examples include knowledge transfer from a human planner into a decision-support system: often human operators have arrived at finely-tuned 'objective functions' through many years of experience, and in many cases it is desirable to replicate the decision-making process both for scaling up and also for potentially including it in large-scale scenario analysis and simulation to explore responses under varying conditions.

Here we consider the learning of preferences or objectives from an expert by means of observing her actions. More precisely, we observe a set of input parameters and corresponding decisions of the form $\{(p_1, x_1), \ldots, (p_T, x_T)\}$. They are such that $p_t \in P$ with $t = 1, \ldots, T$ is a certain realization of problem parameters from a given set $P \subseteq \mathbb{R}^k$ and x_t is an optimal solution to the optimization problem

$$\begin{aligned} \max \quad & c_{\text{true}}^T x \\ \text{s.t.} \quad & x \in X(p_t), \end{aligned}$$

where $c_{\text{true}} \in \mathbb{R}^n$ is the expert's true but unknown objective and $X(p_t) \subseteq \mathbb{R}^n$ for some (fixed) n. We assume that we have full information on the feasible set $X(p_t)$ and that we can compute $\operatorname{argmax}\{c^T x \mid x \in X(p_t)\}$ for any candidate objective $c \in \mathbb{R}^n$ and $t = 1, \ldots, T$. We present an online learning algorithm based on the multiplicative weights update method that allows us to learn a strategy $(c_1, \ldots, c_T)$ of subsequent objective function choices with the following guarantee: if we optimize according to the surrogate objective function c_t instead of the actual unknown objective function c_{true} in response to parameter realization p_t, we obtain a sequence of optimal decisions (w.r.t. to each c_t) given by

$$\bar{x}_t = \operatorname{argmax}\{c_t^T x \mid x \in X(p_t)\},$$

that are essentially as good as the decisions x_t taken by the expert on average. To this end, we interpret the observations of parameters and expert solutions as revealed over multiple rounds such that in each round t we are shown the parameters p_t first, then take our optimal decision $\bar{x}_t$ according to our objective function c_t, then we are shown the solution x_t chosen by the expert and finally we are allowed

[*]Equal contribution [1]Friedrich-Alexander-Universität Erlangen-Nürnberg, Erlangen, Germany [2]Georgia Institute of Technology, Atlanta, USA. Correspondence to: Andreas Bärmann <Andreas.Baermann@math.uni-erlangen.de>, Sebastian Pokutta <Sebastian.Pokutta@isye.gatech.edu>, Oskar Schneider <Oskar.Schneider@fau.de>.

Proceedings of the 34^{th} International Conference on Machine Learning, Sydney, Australia, 2017. JMLR: W&CP.

to update c_t for the next round. For this setup, we will be able to show that our algorithm attains an error bound of

$$0 \leq \frac{1}{T}\sum_{t=1}^{T}(c_t - c_{\text{true}})^T(\bar{x}_t - x_t) \leq 2K\sqrt{\frac{\ln n}{T}},$$

where $K \geq 0$ is an upper bound on the ℓ_∞-diameter of the feasible regions $X(p_t)$ with $t = 1, \ldots, T$. This implies that both the deviations in true cost $c_{\text{true}}^T(x_t - \bar{x}_t) \geq 0$ as well as the deviations in surrogate cost $c_t^T(\bar{x}_t - x_t) \geq 0$ can be made arbitrarily small on average. In other words, the average regret for having decided optimally according to the surrogate objectives c_t vs. having decided optimally for the true objective c_{true} vanishes at a rate of $\mathcal{O}(\frac{1}{\sqrt{T}})$. This result shows that linear objective functions over general feasible sets can be learned from relatively few observations of historical optimal parameter-solutions pairs. We will also briefly discuss the case where the objective c_{true} is known, but some linear constraints are unknown in this paper.

Literature Overview

The idea of learning or inferring parts of an optimization model from data is a reasonably well-studied problem under many different assumptions and applications and has gained significant attention in the optimization community over the last few years, as discussed for example in (den Hertog & Postek, 2016), (Lodi, 2016), (Simchi-Levi, 2014). These papers argue that there would be significant benefits in combining traditional optimization models with data-derived components. Most approaches in the literature focus on deriving the objective function of an expert decision maker based on past observations of input data and the decisions she took in each instance. In almost all cases, the objective functions are learned by considering the KKT-conditions or the dual of the (parameterized) optimization problem, and as such convexity for both the feasible region and the objective function is inherently assumed. Examples of this approach include (Keshavarz et al., 2011), (Li, 2016) and (Thai & Bayen, 2016), where the latter also consider the derivation of variational inequalities from data. Sometimes also distributional assumptions regarding the observations are made. Applications of such approaches have been heavily studied in the context of energy systems (Ratliff et al., 2014; Konstantakopoulos et al., 2016), robot motion (Papadopoulos et al., 2016), (Yang et al., 2014), medicine (Sayre & Ruan, 2014) and revenue management (Kallus & Udell, 2015; Qiang & Bayati, 2016; Chen et al., 2015; Kallus & Udell, 2016; Bertsimas & Kallus, 2016); also in the situation where the observed decisions were not necessarily optimal (Nielsen & Jensen, 2004). Very closely related to our learning approach in terms of the problem formulation is (Troutt et al., 2005) which was later extended in (Troutt et al., 2006), where an optimization model is defined that searches for a linear optimization problem that minimizes the total difference between the observed solutions and solutions found by optimizing according to that optimization problem. In the latter case, the models are solved using LP duality and cutting planes respectively. In their follow-up work (Troutt et al., 2008), a genetic algorithm is used to solve the problem heuristically under rather general assumptions, but inherently without any quality guarantees, and (Troutt et al., 2011) study experimental setups for learning objectives under various stochastic assumptions, focussing on maximal likelihood estimation, which is generally the case for their line of work; we make no such assumptions.

Closely related to learning optimization models from observed data is the subject of inverse optimization. Here the goal is to find an objective function that renders the observed solutions optimal with respect to the concurrently observed parameter realizations. Here approaches mostly from convex optimization are used for inverse optimal control (Iyengar & Kang, 2005; Panchea & Ramdani, 2015; Molloy et al., 2016), inverse combinatorial optimization (D. Burton, 1997; Burton & Toint, 1994; 1992; Sokkalingam et al., 1999; Ahuja & Orlin, 2000), integer inverse optimization (Schaefer, 2009) and inverse optimization in the presence of noisy data, such as observed decisions that were suboptimal (Aswani et al., 2016; Chan et al., 2015).

All these approaches heavily rely on duality and thus require convexity assumptions both for the feasible region as well as the objectives. As such, they cannot deal with more complex, possibly non-convex decision domains. This in particular includes the important case of integer-valued decisions (such as 'yes/no'-decisions) and also many other non-convex setups (several of which admit efficient linear optimization algorithms). Previously, this was only possible when the structure of the feasible set could be beneficially exploited. In contrast, our approach does not make any such assumptions and only requires access to a linear optimization oracle (in short: LP oracle) for the feasible region.

Also related to our work is inverse reinforcement learning and apprenticeship learning, where the reward function is the target to be learned. However, in this case the underlying problem is modeled as a Markov decision process (MDP); see e.g. (Syed & Schapire, 2007) and (Ratia et al., 2012). Typically, the guarantees are of a different form though. Similarly, our work is not to be confused with that of (Taskar et al., 2005) and (III et al., 2005), who develop online algorithms for learning aggregation vectors for edge features in graphs that use inverse optimization as a subroutine to define the update rule. In contrast, we do inverse optimization by means of an online learning algorithm; basically the reverse setup.

Our approach is based on online learning, and we use the simple *EXP* algorithm here to attain the stated regret bound. The EXP algorithm is commonly also called *Multiplicative Weights Update (MWU)* algorithm and was developed in the works of (Littlestone & Warmuth, 1994), (Vovk, 1990) and (Freund & Schapire, 1997) (see (Arora et al., 2012; Hazan, 2016) for a comprehensive introduction; see also (Audibert et al., 2013)). A similar algorithm was used by (Plotkin et al., 1995) for solving fractional packing and covering problems. We also note that our feedback is stronger than bandit feedback. This requirement is not unexpected as the costs chosen by the 'adversary' depend on our decision; as such the bandit model (see e.g. (Dani et al., 2008), (Abbasi-Yadkori et al., 2011)) does not readily apply.

Contribution

To the best of the authors' knowledge, this paper makes the first attempt to learn the objective function of an optimization model from data using an online learning approach.

Online learning of optimization problems. Based on samples for the input-output relationship of an optimization problem solved by a decision maker, our aim is to learn an objective function which is consistent with the observed input-output relationship; this is the best one can hope for. In our setup, the expert solves the decision-making problem repeatedly for different input parameter realizations. From these observations, we are able to learn a strategy of objective functions that emulate the expert's unknown objective function such that the difference in solution quality between the solutions converges to zero on average.

While previous methods based on dualization or KKT-system-based approaches can lead to similar or even stronger results in the continuous/convex case, online learning allows us to relax this convexity requirement and to work with arbitrary decision domains as long as we are able to optimize a linear function over them. Thus, we do not explicitly analyze the KKT-system or the dual program (in the case of LPs; see Remark 3.1). In particular, one might consider our algorithm as an algorithmic analogue of the KKT-system (or dual program) in the convex case.

To summarize, we stress that (a) we do not make any assumptions regarding distribution of the observations, (b) the observations can be chosen by a fully-adaptive adversary, and (c) we do not require any convexity assumptions regarding the feasible regions and only rely on access to an LP oracle. We would also like to mention that our approach can be extended to work with slowly changing objectives using appropriate online learning algorithms such as, for example, (Jadbabaie et al., 2015) or (Zinkevich, 2003); the regret bounds will depend on the rate of change.

Preliminary Computational Tests. While a full computational study is beyond the scope of this paper and left for future work, we implemented a first preliminary version of our algorithm, and we report computational results for a few select problems.

2. Problem Setting

We consider the following family of optimization problems $(\text{OPT}(p))_p$, which depend on a parameter $p \in P \subseteq \mathbb{R}^k$ for some $k \in \mathbb{N}$:

$$\begin{aligned} \max \quad & c_{\text{true}}^T x \\ \text{s.t.} \quad & x \in X(p), \end{aligned}$$

where $c_{\text{true}} \in \mathbb{R}^n$ is the objective function and $X(p) \subseteq \mathbb{R}^n$ is the feasible region, where the latter depends on the parameter p. Of particular interest to us will be feasible regions that arise as polyhedra defined by linear constraints and their intersections with integer lattices:

$$X(p) = \{x \in \mathbb{Z}^{n-l} \times \mathbb{R}^l \mid A(p)x \leq b(p)\}$$

with $A(p) \in \mathbb{R}^{m \times n}$ and $b(p) \in \mathbb{R}^m$. However, our approach can also readily by applied in the case of more complex feasible regions, such as mixed-integer sets bounded by convex functions:

$$X(p) = \{x \in \mathbb{Z}^{n-l} \times \mathbb{R}^l \mid G(p, x) \leq 0\}$$

with $G\colon P \times \mathbb{Z}^{n-l} \times \mathbb{R}^l \to \mathbb{R}$ convex – or even more general settings. In fact, for any possible choice of model for the set of feasible decisions, we only require the availability of a linear optimization oracle capable of optimizing linear functions over $X(p)$ for any $p \in P$. We call a decision $x \in \mathbb{R}^n$ *optimal* for p if it is an optimal solution to $\text{OPT}(p)$.

We assume that Problem $\text{OPT}(p)$ models a parameterized optimization problem which has to be solved repeatedly for various input parameters p. Our task is to learn the fixed objective function c_{true} from given observations of the parameter p and a corresponding optimal solution x to $\text{OPT}(p)$. To this end, we further assume that we are given a series of observations $((p_t, x_t))_t$ of parameter realizations $p_t \in P$ together with an optimal solution x_t to $\text{OPT}(p_t)$ computed by the expert for $t = 1, \ldots, T$; these observations are revealed over time in an online fashion: in round t, we obtain a parameter setting p_t and compute an optimal solution $\bar{x}_t \in X(p_t)$ with respect to an objective function c_t based on what we have learned about c_{true} so far. Then we are shown the solution x_t the expert with knowledge of c_{true} would have taken and can use this information to update our inferred objective function for the next round. In the end, we would like to be able to use our inferred objective function to take decisions that are essentially as good

as those chosen by the expert in an appropriate aggregation measure such as, for example, 'on average' or 'with high probability'. The quality of the inferred objective is measured in terms of cost deviation between our solutions $\bar{x}_t$ and the solutions x_t obtained by the expert. Details will be given in the next section, where we will derive an algorithm based on multiplicative weights updates (MWU) to solve the above task.

To fix some useful notations, let $v(i)$ denote the i-th component of a vector v throughout and let $[n] \coloneqq \{1, \dots, n\}$ for any natural number n. Further, let $\mathbb{1}^n \coloneqq (1, \dots, 1)$ denote the all-ones vector in $\mathbb{R}^n$.

As a technical assumption, we further demand $\|c_{\text{true}}\|_1 = 1$ and $c_{\text{true}} \geq 0$. Both requirements are without loss of generality and are motivated by our choice of MWU as the online learning algorithm to simplify the exposition. We can drop these assumptions by employing, for example, the *Online Gradient Descent* algorithm of (Zinkevich, 2003), which requires an explicit projection step.

3. Learning Objectives

Ideally, we would like to find the true objective function c_{true} as a solution to the following optimization problem:

$$\min_{\substack{c \in \mathbb{R}^n_+ : \\ \|c\|_1 = 1}} \sum_{t \in [T]} \left(\left(\max_{x \in X(p_t)} c^T x \right) - c^T x_t \right), \tag{1}$$

where $x_t \in X(p)$ is the *optimal* decision taken by the expert in round t. The (normalized) true objective function $c_{\text{true}} \geq 0$ is an optimal solution to Problem (1) with objective value 0. This is because any solution $\hat{c}$ with $\|\hat{c}\|_1 = 1$ is feasible and produces non-negative summands

$$\left(\max_{x \in X(p_t)} \hat{c}^T x \right) - \hat{c}^T x_t, \quad t \in [T],$$

as we assume $x_t \in X(p_t)$ to be optimal for p_t with respect to c_{true}. Thus, the optimal value of (1) is bounded from below by 0.

When solving Problem (1) we are interested in an objective function vector $c \in \mathbb{R}^n$ that delivers a consistent explanation for why the expert chose x_t as his response to the parameters p_t in round $t = 1, \dots, T$. To be more precise, we want to minimize the deviation between the optimal value obtained when optimizing over $X(p_t)$ with our guess for the objective function c and the value of the expert's decision evaluated according to our guess c, averaged over all observations. Our algorithm presented here will provide even stronger guarantees in some cases, such as the one described in Section 3.1, showing that we can replicate the decision-making behavior of the expert.

Problem (1) contains T instances of the following maximization subproblem:

$$\max \quad c^T x \tag{2a}$$
$$\text{s.t.} \quad x \in X(p_t) \tag{2b}$$

For each $t = 1, \dots, T$, the corresponding Subproblem (2) asks for an optimal solution $\bar{x}_t$ when optimizing over the feasible set $X(p_t)$ with our guess c as the objective function.

Remark 3.1. *Note that in the case of polyhedral feasible regions, i.e. $p_t = (A_t, b_t) \in \mathbb{R}^{m \times n} \times \mathbb{R}^m$ and $X(p_t) = \{x \in \mathbb{R}^n_+ \mid A_t x \leq b_t\}$ for $t = 1, \dots, T$, Problem (1) can be reformulated as a linear program by dualizing the T instances of Subproblem (2). This yields*

$$\begin{aligned}
\min \quad & \sum_{t=1}^{T} (b_t^T y_t - c^T x_t) & & \\
\text{s.t.} \quad & A_t^T y_t \geq c & & (\forall t = 1, \dots, T) \\
& y_t \geq 0 & & (\forall t = 1, \dots, T) \\
& \sum_{i=1}^{n} c_i = 1 & & \\
& c \geq 0, & &
\end{aligned}$$

where the y_t are the corresponding dual variables and the x_t are the observed decisions from the expert (i.e. the latter are part of the input data*). This problem asks for a primal objective function vector c that minimizes the total duality gap summed over all primal-dual pairs (x_t, y_t) while all y_t's shall be dual feasible, which makes the x_t's the respective primal optimal solutions. Thus, Problem (1) can be seen as a direct generalization of the linear primal-dual optimization problem. In fact, our approach also covers non-convex cases, e.g. mixed-integer linear programs.*

Problem (1) can be interpreted as a game over T rounds between a player who chooses an objective function c_t in round $t \in [T]$ and a player who knows the true objective function c_{true} and chooses the observations (p_t, x_t) in a potentially adversarial way. The payoff of the latter player in each round t is equal to $c_t^T(\bar{x}_t - x_t) \geq 0$, i.e. the difference in cost between our solution and the expert's solution as given by our guessed objective function c_t.

As Problem (1) is hard to solve in general, we will design an algorithm that, rather than finding an optimal objective c, finds a strategy of objective functions $(c_1, c_2, \dots, c_T)$ to play in each round whose error in solution quality as compared to the true objective function is as small as possible. Our aim will then be to give a quality guarantee for this strategy in terms of the number of observations.

To allow for approximation guarantees, it is necessary that the observed feasible sets have a common upper bound on their ℓ_∞-diameter.

Definition 3.2. *The ℓ_∞-diameter of a set $S \subseteq \mathbb{R}^n$, denoted by $\operatorname{diam}_\infty(S)$ is the largest distance between any two points $x_1, x_2 \in S$, measured in the infinity-norm, i.e. $\operatorname{diam}_\infty(S) \coloneqq \max_{x_1,x_2 \in S} \|x_1 - x_2\|_\infty$.*

In the following, we assume that there exists a $K \geq 0$ with $\operatorname{diam}_\infty(X(p_t)) \leq K$ for all $t = 1, \ldots, T$. With these prerequisites, our application of multiplicative weights updates (MWU) to learn the objective function of an optimization problem proceeds as outlined in Algorithm 1.

Algorithm 1 Online Objective Function Learning

input observations (p_t, x_t) for $t = 1, \ldots, T$
output sequence of objectives $c_1, c_2, \ldots, c_T$

1: $\eta \leftarrow \sqrt{\frac{\ln n}{T}}$ {set learning rate}
2: $w_1 \leftarrow \mathbb{1}^n$ {initialize weights}
3: **for** $t = 1, \ldots, T$ **do**
4: $\quad c_t \leftarrow \frac{w_t}{\|w_t\|_1}$ {normalize weights}
5: $\quad \bar{x}_t \leftarrow \operatorname{argmax}\{c_t^T x \mid X(p_t)\}$ {solve Subproblem (2)}
6: $\quad$ **if** $\bar{x}_t = x_t$ **then**
7: $\quad\quad y_t \leftarrow 0$
8: $\quad$ **else**
9: $\quad\quad y_t \leftarrow \frac{\bar{x}_t - x_t}{\|\bar{x}_t - x_t\|_\infty}$
10: $\quad$ **end if**
11: $\quad w_{t+1}(i) \leftarrow w_t(i)(1 - \eta y_t(i))$ {update weights}
12: **end for**
13: **return** $(c_1, c_2, \ldots, c_T)$.

For the series of objectives functions $(c_t)_t$ that our algorithm produces over rounds $t = 1, \ldots, T$, we can establish the following guarantee:

Theorem 3.3. *Let $K \geq 0$ with $\operatorname{diam}_\infty X(p_t) \leq K$ for all $t = 1, \ldots, T$. Then we have*

$$0 \leq \frac{1}{T} \sum_{t=1}^{T} (c_t - c_{true})^T (\bar{x}_t - x_t) \leq 2K \sqrt{\frac{\ln n}{T}},$$

and in particular it also holds:

1. $0 \leq \frac{1}{T} \sum_{t=1}^{T} c_t^T (\bar{x}_t - x_t) \leq 2K \sqrt{\frac{\ln n}{T}}$,
2. $0 \leq \frac{1}{T} \sum_{t=1}^{T} c_{true}^T (x_t - \bar{x}_t) \leq 2K \sqrt{\frac{\ln n}{T}}$.

Proof. According to the standard performance guarantee of MWU (see, e.g., (Arora et al., 2012), Corollary 2.2), Algorithm 1 attains the following solution quality for the comparison between the objective function c_t chosen in each round t with the unknown true objective function c_{true}:

$$\sum_{t=1}^{T} c_t^T y_t \leq \sum_{t=1}^{T} c_{\text{true}}^T (y_t + \eta |y_t|) + \frac{\ln n}{\eta},$$

where the $|y_t|$ is to be understood component-wise. Using that each each entry of $|y_t|$ is at most 1 and dividing by T, we can conclude

$$\frac{1}{T} \sum_{t=1}^{T} c_t^T y_t - \frac{1}{T} \sum_{t=1}^{T} c_{\text{true}}^T y_t \leq \eta \sum_{i=1}^{n} c_{\text{true}}(i) + \frac{\ln n}{\eta T}$$

and further

$$\frac{1}{T} \sum_{t=1}^{T} c_t^T y_t - \frac{1}{T} \sum_{t=1}^{T} c_{\text{true}}^T y_t = \eta + \frac{\ln n}{\eta T}.$$

The right-hand side attains its minimum for $\eta = \sqrt{\frac{\ln n}{T}}$, which yields the bound

$$\frac{1}{T} \sum_{t=1}^{T} c_t^T y_t - \frac{1}{T} \sum_{t=1}^{T} c_{\text{true}}^T y_t \leq 2\sqrt{\frac{\ln n}{T}}.$$

Substituting back for the y_t's and using

$$\max_{t=1,\ldots,T} \|\bar{x}_t - x_t\|_\infty \leq \max_{t=1,\ldots,T} \operatorname{diam}_\infty(X(p_t)) \leq K,$$

we obtain

$$\frac{1}{T} \sum_{t=1}^{T} c_t^T (\bar{x}_t - x_t) + \frac{1}{T} \sum_{t=1}^{T} c_{\text{true}}^T (x_t - \bar{x}_t) \leq 2K \sqrt{\frac{\ln n}{T}}.$$

Observe that for each summand $t \in [T]$ we have $c_t^T (\bar{x}_t - x_t) \geq 0$ as $\bar{x}_t, x_t \in X(p_t)$ and $\bar{x}_t$ is the maximum over this set with respect to c_t. With a similar argument, we see that $c_{\text{true}}^T (x_t - \bar{x}_t) \geq 0$ for all $t \in [T]$. Thus, we have

$$0 \leq \frac{1}{T} \sum_{t=1}^{T} (c_t - c_{\text{true}})^T (\bar{x}_t - x_t) \leq 2K \sqrt{\frac{\ln n}{T}}, \qquad (3)$$

and similarly for the separate terms with analogue argumentation. This establishes the claim. □

Note that by using exponential updates of the form

$$w_{t+1}(i) \leftarrow w_t(i) e^{-\eta y_t(i)}$$

in line 11 of the algorithm, we could attain the same bound, cf. (Arora et al., 2012, Theorem 2.3). Secondly, we remark that our choice of the learning rate η requires the number of rounds T to be known beforehand.

From the above theorem, we can conclude that the average error over all observations (p_t, x_t) for $t = 1, \ldots, T$ when choosing objective function c_t in iteration t of Algorithm 1 instead of c_{true} converges to 0 with an increasing number of observations T at a rate of roughly $\mathcal{O}(\frac{1}{\sqrt{T}})$:

Corollary 3.4. *Let $K \geq 0$ with $\operatorname{diam}_\infty X(p_t) \leq K$ for all $t = 1, \ldots, T$. Then we have*

1. $\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} c_t^T(\bar{x}_t - x_t) = 0$ *and*
2. $\lim_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T} c_{true}^T(x_t - \bar{x}_t) = 0$.

In other words, both the average error incurred from replacing the actual objective function c_{true} by the estimation c_t as well as the average error in solution quality with respect to c_{true} tend to 0 as T grows.

Moreover, using Markov's inequality we also obtain the following quantitative bound on the deviation by more than $\varepsilon > 0$ from the average cost:

Corollary 3.5. *Let $\varepsilon > 0$. Then the fraction of observations x_t with $c_{true}^T(x_t - \bar{x}_t) \geq 2K\sqrt{\frac{\ln n}{T}} + \varepsilon$ is at most $1 - \frac{\varepsilon}{2K\sqrt{\frac{\ln n}{T}}+\varepsilon}$. In particular, for any $0 < p < 1$ after $T \geq \ln n \left(\frac{(1-p)2K}{p\varepsilon}\right)^2$ observations the fraction of observations x_t with cost $c_{true}^T(x_t - \bar{x}_t) \geq \frac{\varepsilon}{1-p} \geq 2K\sqrt{\frac{\ln n}{T}} + \varepsilon$ is at most p.*

Proof. The first part is an obvious application of Markov's inequality. The second part follows from solving $1 - \frac{\varepsilon}{2K\sqrt{\frac{\ln n}{T}}+\varepsilon} \leq p$ for T and plugging in values. □

3.1. The Stable Case

Note that $\lim_{T\to\infty}(c_t - c_{\text{true}})^T(\bar{x}_t - x_t) = 0$, as derived from Equation (3) does not necessarily imply that we can approximate c_{true} itself. A counterexample is the case where $X(p_t) \subseteq \{x \in \mathbb{R}^n \mid x(1) = 0\}$, which means that any two objective functions $c_1, c_2 \neq 0$ with $c_2(i) = \kappa c_1(i)$ for $i = 2, \ldots, n$ and $0 < \kappa \leq 1$ are equivalent if $c_1(1), c_2(1)$ are chosen such that $\|c_1\|_1 = \|c_2\|_1 = 1$. Using this construction, we can easily find examples for which $\|c_1 - c_2\|_1 = 2(1 - \kappa)$, but where the two objective functions are equivalent in terms of optimal solutions.

While in most applications it is sufficient to be able to produce solutions via the surrogate objectives that are essentially equivalent to those for the true objective, we will show now that under slightly strengthened assumptions we can obtain significantly stronger guarantees for the convergence of the solutions: we will show that in the long run we learn to emulate the true optimal solutions provided that the problems have unique solutions as we will make precise now.

We say that the sequence of feasible regions $(X(p_t))_t$ is Δ-*stable for* c_{true} for some $\Delta > 0$ if for any $t \in [T]$, $c \in \mathbb{R}^n$ with $\|c\|_1 = 1$, $c \neq c_{\text{true}}$ and $\bar{x}_t \coloneqq \operatorname{argmin}\{c^T x \mid x \in X(p_t)\}$ so that for $x_t \neq \bar{x}$ we have

$$c_{\text{true}}^T(x_t - \bar{x}_t) \geq \Delta,$$

i.e. either the two optimal solutions coincide or they differ by at least Δ with respect to c_{true}. In particular, optimizing c_{true} over $X(p_t)$ leads to a unique optimal solution for all p_t with $t \in [T]$. While this condition sounds unnatural at first, for example it is trivially satisfied for the important case where $X(p_t)$ with $t \in [T]$ is a polytope with vertices in $\{0, 1\}$ and c_{true} is a rational vector. In this case, write $c_{\text{true}} = \frac{d}{\|d\|_1}$ with $d \in \mathbb{Z}_+^n$ and observe that the minimum change in objective value between any two vertices x, y of the 0/1-polytope with $c_{\text{true}}^T x \neq c_{\text{true}}^T y$ is bounded by $|c_{\text{true}}^T(x - y)| \geq \frac{1}{\|d\|_1}$, so that Δ-stability with $\Delta \coloneqq \frac{1}{\|d\|_1}$ holds in this case. The same argument works for more general polytopes via bounding the minimum non-zero change in objective function value via the encoding length.

We obtain the following simple corollary of Theorem 3.3.

Corollary 3.6. *Let $K \geq 0$ with $\operatorname{diam}_\infty X(p_t) \leq K$ for all $t = 1, \ldots, T$, let $(X(p_t))_t$ be Δ-stable for some $\Delta > 0$, and let $N_T \coloneqq \{t \in [T] \mid \bar{x}_t \neq x_t\}$. Then*

$$|N_T| \leq 2K\sqrt{\frac{T \ln n}{\Delta}}.$$

Proof. We start with the guarantee from the proof of Theorem 3.3: $0 \leq \frac{1}{T}\sum_{t=1}^{T} c_{\text{true}}^T(x_t - \bar{x}_t) \leq 2K\sqrt{\frac{\ln n}{T}}$. Now let N_T be as above so that $0 \leq \frac{1}{T}\sum_{t\in N_T} c_{\text{true}}^T(x_t - \bar{x}_t) \leq 2K\sqrt{\frac{\ln n}{T}}$. Observe that $\Delta \leq c_{\text{true}}^T(x_t - \bar{x}_t)$ as x_t was optimal for c_{true} together with Δ-stability. We thus obtain $\frac{1}{T}|N_T|\Delta \leq 2K\sqrt{\frac{\ln n}{T}}$, which is equivalent to $|N_T| \leq 2K\sqrt{\frac{T \ln n}{\Delta}}$. □

From the above corollary, we obtain in particular that in the Δ-stable case we have $\frac{1}{T}|N_T| \leq 2K\sqrt{\frac{\ln n}{T\Delta}}$, i.e. the average number of times that $\bar{x}_t$ deviates from x_t tends to 0 in the long run. We hasten to stress, however, that the convergence implied by this bound can potentially be slow as it is exponential in the actual encoding length of c_{true}; this is to be expected given the convergence rates of our algorithm and online learning algorithms in general.

3.2. Learning Constraints

We will only very briefly address the case of learning constraints due to space limitations. We consider the family of optimization problems $(\text{OPT2}(p))_p$, $p \in P \subseteq \mathbb{R}^k$, given

by

$$\begin{array}{lrcl} \max & c(p)^T x & & \\ \text{s.t.} & Ax & \leq & b_{\text{true}} \\ & x & \geq & 0, \end{array}$$

where $c(p) \in \mathbb{R}^n$ is the objective function, $A \in \mathbb{R}^{m \times n}$ is the constraint matrix and $b_{\text{true}} \in \mathbb{R}^m$ is the right-hand side.

We assume that the c_t's are known to both the learner and the expert. The same can be assumed for A without loss of generality by standard arguments. The right-hand side b_{true} is only known to the expert and to be learned from observing p_t as well as an optimal solution x_t for OPT2(p_t) in round t.

The most natural approach for solving this learning problem is to apply Algorithm 1 to the dual of OPT2(p_t):

$$\begin{array}{lrcl} \min & b_{\text{true}}^T y & & \\ \text{s.t.} & A^T y & \geq & c(p_t) \\ & y & \geq & 0, \end{array}$$

where y are the dual variables for the linear constraints. In the dual problem, b_{true} is the unknown objective function ($b_{\text{true}} \geq 0$ without loss of generality), while the constraints to be optimized over in each round are known – the same setting as before. It is important to note though that the learner has to observe the *dual* optimal solutions y_t and the guarantee will be that the *dual regret* is tending to 0. It remains open whether this can be also achieved when receiving only the primal optimal solutions x_t as feedback; we suspect the answer to be in the negative in general.

4. Applications

We will now sketch two select applications of our framework for learning objective functions. These are the learning of customer preferences from observed purchases and the learning of travel times in a road network.

Our preliminary computational experiments have been obtained on a Mac Book Pro (2016) with an Intel Core i5 CPU with two 2.00 GHz cores. We have implemented our framework using *python* and *Gurobi 7.0.1* (Gurobi Optimization, Inc., 2016).

4.1. Learning Customer Preferences

We consider a market, where different goods G can be bought by its customers. The prices for the goods can vary over different days $t \in [T]$. We assume that the goods are chosen by the customer to maximize utility given their budget constraints. Each sample (p_t, x_t) corresponds to a day $t \in [T]$ where $p_t = (p_{t0}, p_{tG})$ with p_{t0} is the budget of the buyer and p_t^G contains the prices p_{tg} for each good g at time t. The customer solves the following optimization problem OPT(p_t) on day t:

$$\begin{array}{lrcl} \max & \sum_{g \in G} u_g x_g & & \\ \text{s.t.} & \sum_{g \in G} p_{tg} x_g & \leq & p_{t0} \\ & x & \in & \{0,1\}^{|G|}, \end{array}$$

where the utility u_g of good g of the customer is unknown (and kept constant over time).

We consider two different setups: in the first setup, we assume that the goods are divisible, which means that the condition $x \in \{0,1\}^{|G|}$ is relaxed to $x \in [0,1]^{|G|}$; this is the *Linear Knapsack Problem*. In the second setup, the goods are not divisible, so that we solve the problem with the original constraint $x \in \{0,1\}^{|G|}$ as an integer program; this is the *Integer Knapsack Problem*.

We generated random instances for our computational results, considering $T = 1000$ observations for a varying number of goods $n \in \{100, 500, 1000\}$. The customer's unknown utility vector is chosen at random as (arbitrary) integer numbers from the interval $[1, 1000]$ from a uniform distribution and then normalized to have ℓ_1-norm 1. The prices on day t are chosen to be $p_{tg} = u_g + 100 + r_{tg}$, where r_{tg} is an integer uniformly chosen at random from the interval $[-10, 10]$. Choosing utilities and weights similar to each other typically leads to harder (integer) knapsack problems, cf. (Pisinger, 2005). The right-hand side p_{t0} is then again an integer drawn uniformly from the interval $[1, \sum_{g \in G} p_{tg} - 1]$.

Table 1 Errors for Integer Knapsack with $n = 1000$ items

Error / T	10	100	1000
Average objective error	0.1511	0.0185	0.0036
Average solution error	0.0492	0.0087	0.0013
Average cumulative error	0.2003	0.0272	0.0049

In Table 1, we show the computational results for the Integer Knapsack Problem with $n = 1000$ items. We report errors for 10, 100 and 1000 iterations of our algorithm. The *objective error* for each round $t \in [T]$ is defined by $c_t^T(\bar{x}_t - x_t)$ and describes the deviation between the solution $\bar{x}_t$ found by the oracle in that round and the solution x_t observed from the expert as evaluated with our guess for the objective function c_t. Accordingly, the *solution error* in each round t is defined as $c_{\text{true}}^T(\bar{x}_t - x_t)$ and evaluates the deviation between the two solutions in the true objective function. Together they yield the *total error*, given by $(c_t - c_{\text{true}})^T(\bar{x}_t - x_t)$, which is the total deviation between choosing c_t and c_{true} in Problem (1) in round t. Each of these error types is shown in our plots over time, depicting both the error in a given round t and the average error

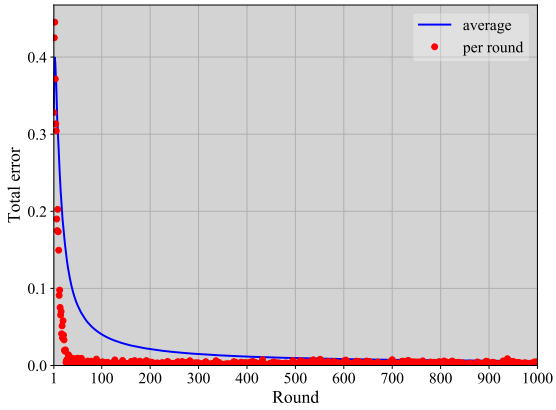


Figure 1. Linear Knapsack problem with $n = 100$ items over $T = 1000$ iterations. We plot $\frac{1}{T}\sum_{t=1}^{T}(c_t - c_{\text{true}})^T(\bar{x}_t - x_t)$ over T on the x-axis in blue. In red we plot the cost $(c_t - c_{\text{true}})^T(\bar{x}_t - x_T)$ of round t. As can be seen, after few iterations most solutions reside on the x-axis and only few deviate beyond the average.

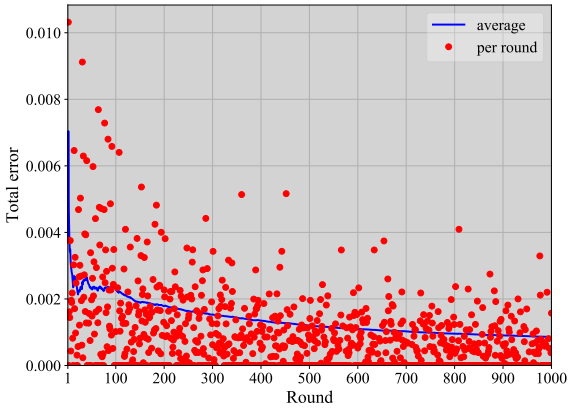


Figure 2. Resource-constrained shortest path problem on a grid graph with $m = 15$ rows and $n = 30$ columns as described above for $T = 1000$ iterations. Total error as in Figure 1. Convergence is slower here (although with an error that is several orders of magnitude smaller) as the problem is much more complex. Still, in most rounds we have an error close to 0.

over rounds $t' \in [t]$. We depict a representative knapsack instance in Figure 1.

4.2. Learning Travel Times

While the first example explored learning over a temporal horizon, in this example the various observations $t \in [T]$ arise from different drivers in a road network. More precisely, we consider a resource-constrained shortest path problem, where we are given a graph $G = (V, E)$, where drivers have to find cost-minimal s-t-paths subject to a resource or budget constraint. For each arc $e \in E$, we denote the arc length with a_e. The observations $t \in [T]$ represent drivers in the network and each observation (p_t, x_t) consists of $p_t = (p_t^1, p_t^2, p_t^3)$, where p_t^1 is the starting point and p_t^2 is the ending point of the journey of driver t. It is assumed that driver t takes the path with the shortest travel time with respect to the unknown travel times c_e with $e \in E$ while at the same time being subject to a limit p_t^3 of total distance that can be traveled. The values of x_t indicate the traversed edges of the graph that driver t takes. The optimization problem $\text{OPT}(p_t)$ solved by driver t is then:

$$
\begin{aligned}
\min \quad & \sum_{e \in E} c_e x_e \\
\text{s.t.} \quad & \sum_{e \in \delta^-(v)} x_e - \sum_{e \in \delta^+(v)} x_e = \begin{cases} -1, & \text{if } v = p_t^1 \\ 1, & \text{if } v = p_t^2 \\ 0, & \text{otherwise} \end{cases} \quad (\forall v \in V) \\
& \sum_{e \in E} a_e x_e \leq p_t^3 \\
& x \in \{0,1\}^{|E|},
\end{aligned}
$$

and we want to learn the values c_e for $e \in E$ corresponding to the travel time to traverse arc e.

We created instances of the problem based on grid graphs with m rows and n columns. The unknown driving times vector of the network and the resource value for each arc (in our case the length of the arc) were chosen at random in the same fashion as for the knapsack problems. For each sample, we chose a random pair of an origin and a destination node. The resource limit for the sample is calculated as the length of the shortest path between the selected nodes multiplied by 1.25. In other words: find the fastest path while driving at most 25% more than length of the shortest path. See Figure 2 for our result over $T = 1000$ samples.

Additional computations are included in the Appendix.

5. Final Remarks

In its current form, we explicitly use the optimality of the observed actions to learn the objective function. While beyond the scope of this paper, it would be interesting to analyse to what extent this optimality requirement can be relaxed to approximate solutions. Clearly, one can simply redefine the underlying feasible regions $X(p_t)$ to correspond to the approximate feasible regions, however this can lead to unwanted effects of the summands in our regret bound not being non-negative anymore if the solutions obtained for the surrogate objective is better than the approximately optimal observed solution. Another important question is to what extent our framework can be extended to the learning of constraints (and objective functions simultaneously).

Acknowledgements

We would like to thank the reviewers for the helpful comments. Research reported in this paper was partially supported by NSF CAREER award CMMI-1452463.

References

Abbasi-Yadkori, Yasin, Pál, Dávid, and Szepevári, Csaba. Improved algorithms for linear stochastic bandits. In *Conference on Neural Information Processing System (NIPS)*, 2011.

Ahuja, Ravindra K. and Orlin, James B. A faster algorithm for the inverse spanning tree problem. *Journal of Algorithms*, pp. 177–193, 2000.

Arora, Sanjeev, Hazan, Elad, and Kale, Satyen. The multiplicative weights update method: A meta-algorithm and applications. *Theory of Computing*, 8:121–164, 2012.

Aswani, Anil, Shen, Zuo-Jun Max, and Siddiq, Auyon. Inverse optimization with noisy data. Technical report, University of California, Berkeley, 2016. available at: https://arxiv.org/abs/1507.03266.

Audibert, Jean-Yves, Bubeck, Sébastien, and Lugosi, Gábor. Regret in online combinatorial optimization. *Mathematics of Operations Research*, 39(1):31–45, 2013.

Bertsimas, Dimitris and Kallus, Nathan. Pricing from observational data. Technical report, Massachusetts Institute of Technology, 2016. available at:https://arxiv.org/pdf/1605.02347.

Burton, D. and Toint, Ph. L. On an instance of the inverse shortest paths problem. *Mathematical Programming*, 53 (1):45–61, 1992.

Burton, D. and Toint, Ph. L. On the use of an inverse shortest paths algorithm for recovering linearly correlated costs. *Mathematical Programming*, 63(1):1–22, 1994.

Chan, Timothy C. Y., Lee, Taewoo, and Terekhov, Daria. Goodness of fit in inverse optimization. Technical report, University of Toronto, Canada, 2015. available at: https://arxiv.org/abs/1511.04650.

Chen, Xi, Owen, Zachary, Pixton, Clark, and Simchi-Levi, David. A statistical learning approach to personalization in revenue management. Technical report, New York University, 2015.

D. Burton, W. R. Pulleyblank, Ph. L. Toint. *Network Optimization*, chapter The Inverse Shortest Paths Problem with Upper Bounds on Shortest Paths Costs, pp. 156–171. Springer, 1997.

Dani, Varsha, Hayes, Thomas P., and Kakade, Sham. M. Stochastic linear optimization under bandit feedback. In *Conference on Learning Theory (COLT)*, 2008.

den Hertog, Dick and Postek, Krzysztof. Bridging the gap between predictive and prescriptive analytics – new optimization methodology needed. Technical report, Tilburg University, Netherlands, 2016. Available at: http://www.optimization-online.org/DB_HTML/2016/12/5779.html.

Freund, Yoav and Schapire, Robert E. Adaptive game playing using multiplicative weights. *Games and Economic Behavior*, 29:79–103, 1997.

Gurobi Optimization, Inc. Gurobi optimizer reference manual. http://www.gurobi.com, 2016. URL http://www.gurobi.com.

Hazan, Elad. Introduction to online convex optimization. *Foundations and Trends in Optimization*, 2(3–4):157–325, 2016. doi: 10.1561/2400000013. URL http://ocobook.cs.princeton.edu/.

III, Hal Daumé, Khuller, Samir, Purohit, Manish, and Sanders, Gregory. On correcting inputs: Inverse optimization for online structured prediction. In *Proceedings of the IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, 2005.

Iyengar, Garud and Kang, Wanmo. Inverse conic programming with applications. *Operations Research Letters*, 33 (3):319–330, 2005.

Jadbabaie, Ali, Rakhlin, Alexander, Shahrampour, Shahin, and Sridharan, Karthik. Online optimization: Competing with dynamic comparators. In *AISTATS*, 2015.

Kallus, Nathan and Udell, Madeleine. Learning preferences from assortment choices in a heterogeneous population. Technical report, Massechusetts Institute of Technology, 2015. available at: https://pdfs.semanticscholar.org/b29d/026b6776e94a00d1ea15f83518ebbbd14d85.pdf.

Kallus, Nathan and Udell, Madeleine. Dynamic assortment personalization in high dimensions. Technical report, Cornell University, 2016. https://arxiv.org/pdf/1610.05604.

Keshavarz, Arezou, Wang, Yang, and Boyd, Stephen. Imputing a convex objective function. In *Proceedings of the 2011 IEEE International Symposium on Intelligent Control (ISIC)*, pp. 613–619, 2011.

Konstantakopoulos, Ionnias C., Ratliff, Lillian J., Jin, Ming, Spanos, Costas, and Sastry, S. Shankar. Smart building energy efficiency via social game: A robust utility learning framework for closing-the-loop. In *2016 1st International Workshop on Science of Smart City Operations and Platforms Engineering (SCOPE) in partnership with Global City Teams Challenge (GCTC) (SCOPE - GCTC)*, pp. 1–6, 2016.

Li, Jonathan Yu-Meng. Inverse optimization of convex risk function. Technical report, University of Ottawa, Canada, 2016. available at: https://arxiv.org/abs/1607.07099.

Littlestone, Nick and Warmuth, Manfred K. The weighted majority algorithm. *Information and Computation*, 108: 212–261, 1994.

Lodi, Andrea. Big data & mixed-integer (nonlinear) programming. Presentation, available at: https://atienergyworkshop.files.wordpress.com/2015/11/andrealodi.pdf, 2016.

Molloy, Timothy L., Tsai, Dorian, Ford, Jason J., and Perez, Tristan. Discrete-time inverse optimal control with partial-state information: A soft-optimality approach with constrained state estimation. In *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 1926–1932, 2016.

Nielsen, Thomas D. and Jensen, Finn V. Learning a decision makers's utility function from (possibly) inconsistent behavior. *Artificial Intelligence*, 160:53–78, 2004.

Panchea, Adina M. and Ramdani, Nacim. Towards solving inverse optimal control in a bounded-error framework. In *2015 American Control Conference (ACC)*, pp. 4910–4915, 2015.

Papadopoulos, Alessandro Vittorio, Bascetta, Luca, and Ferretti, Gianni. Generation of human walking paths. *Autonomous Robots*, 40(1):55–75, 2016.

Pisinger, David. Where are the hard knapsack problems? *Computers & Operations Research*, 32(9):2271–2284, 2005.

Plotkin, Serge A., Shmoys, David B., and Éva Tardos. Fast approximation algorithms for fractional packing and covering problems. *Mathematics of Operations Research*, 20(2):257–301, 1995.

Qiang, Sheng and Bayati, Mohsen. Dynamic pricing with demand covariates. Technical report, Stanford University, 2016. available at: https://papers.ssrn.com/sol3/papers2.cfm?abstract_id=2765257.

Ratia, Héctor, Montesano, Luis, and Martinez-Cantin, Ruben. On the performance of maximum likelihood inverse reinforcement learning. *arXiv preprint arXiv:1202.1558*, 2012.

Ratliff, Lillian J., Dong, Roy, Ohlsson, Henrik, and Sastry, S. Shankar. Incentive design and utility learning via energy disaggregation. In *Proceedings of the 19th World Congress of the International Federation of Automatic Control*, pp. 3158–3163, 2014.

Sayre, G. A. and Ruan, D. Automatic treatment planning with convex imputing. *Journal of Physics: Conference Series*, 489(1), 2014.

Schaefer, Andrew. Inverse integer programming. *Optimization Letters*, 3(4):483–489, 2009.

Simchi-Levi, David. OM research: From problem-driven to data-driven research. *Manufacturing & Service Operations Management*, 16(1):2–10, 2014.

Sokkalingam, P. T., Ahuja, Ravindra K., and Orlin, James B. Solving inverse spanning tree problems through network flow techniques. *Operations Research*, 47(2):291–298, 1999.

Syed, Umar and Schapire, Robert E. A game-theoretic approach to apprenticeship learning. In *Conference on Neural Information Processing System (NIPS)*, 2007.

Taskar, Ben, Chatalbashev, Vassil, Koller, Daphne, and Guestrin, Carlos. Learning structured prediction models: A large margin approach. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2005.

Thai, Jérôme and Bayen, Alexandre M. Imputing a variational inequality function or a convex objective function: A robust approach. *Journal of Mathematical Analysis and Applications*, 2016. to appear, available at: http://www.sciencedirect.com/science/article/pii/S0022247X16305340.

Troutt, Marvin D., Tadisina, Suresh K., Sohn, Changsoo, and Brandyberry, Alan A. Linear programming system identification. *European Journal on Operational Research*, 161:663–672, 2005.

Troutt, Marvin D., Pang, Wan-Kai, and Hung-Huo, Shui. Behavioral estimation of mathematical programming objective function coefficients. *Management Science*, 52 (3):422–434, 2006.

Troutt, Marvin D., Brandybarry, Alan A., Sohn, Changsoo, and Tadisina, Suresh K. Linear programming system identification: The general nonnegative parameters case. *European Journal on Operational Research*, 185: 63–75, 2008.

Troutt, Marvin D, Gwebu, Kholekile L, Wang, Jing, and Brandyberry, Alan A. Some experiments on subjective optimisation. *International Journal of Operational Research*, 12(1):79–103, 2011.

Vovk, Volodimir G. Aggregating strategies. In *Conference on Learning Theory (COLT)*, 1990.

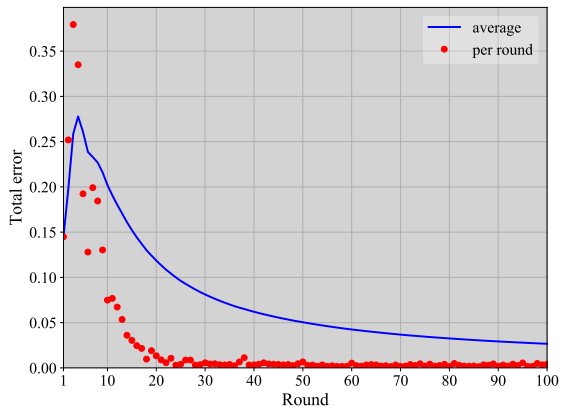
Yang, Insoon, Zeilinger, Melanie N., and Tomlin, Claire J. Utility learning model predictive control for personal electric loads. In *53rd IEEE Conference on Decision and Control*, pp. 4868–4874, 2014.

Zinkevich, Martin. Online convex programming and generalized infinitesimal gradient ascent. Technical report, School of Computer Science, Carnegie Mellon University, 2003. available at: http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/Web/People/maz/publications/techconvex.pdf.

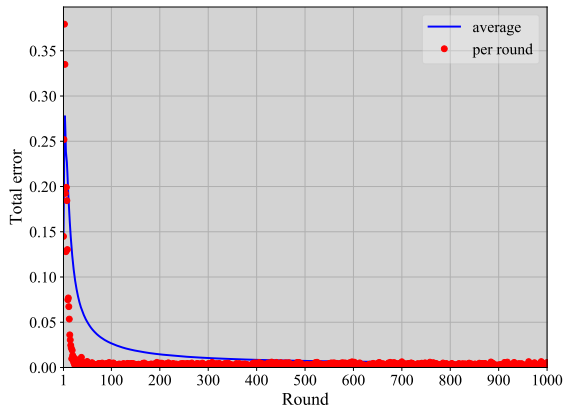
6. Supplementary Material

In this Appendix, we present a somewhat broader range of visualizations of our results on the Linear and Integer Knapsack as well as the RCSP instances.

First, we show the different error measures for the Linear Knapsack over 100 and 1000 rounds each – the total error in Figures 3, the objective error in Figure 4 and the solution error in Figure 5. Confirming the theoretical results in Section 3, we see that the average errors decrease roughly at a rate of $\mathcal{O}(\frac{1}{\sqrt{T}})$ in the number of observations T.



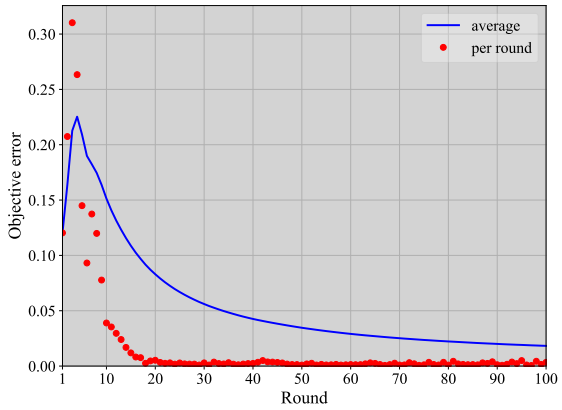
(a) Over $T = 100$ rounds



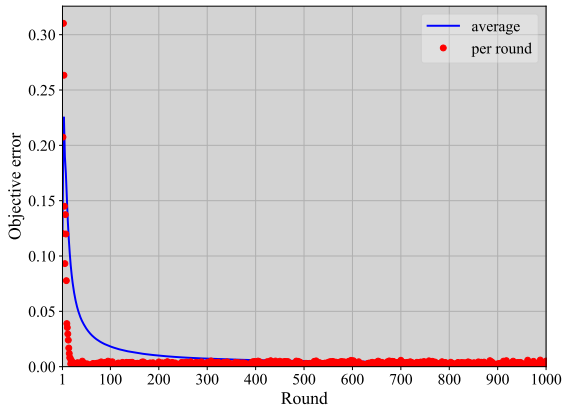
(b) Over $T = 1000$ rounds

Figure 3. Total error for the Linear Knapsack instance with $n = 1000$ items

Figure 6 shows the corresponding development of the distance between the objective function c_t played in round t and the true objective funtion c_{true}, as well as the difference of two successively played objective functions c_t and c_{t-1}, as measured in the 1-norm. We see that the played objective functions get closer to the true objective functions very

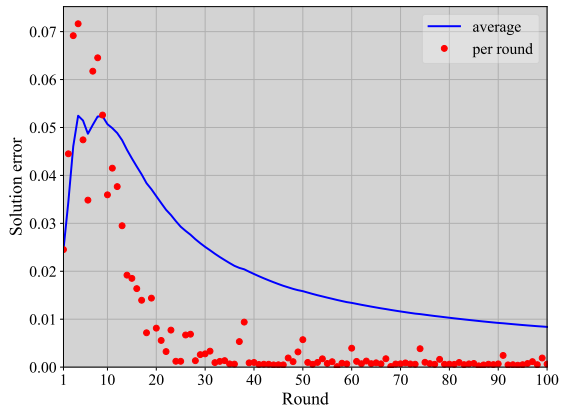


(a) Over $T = 100$ rounds

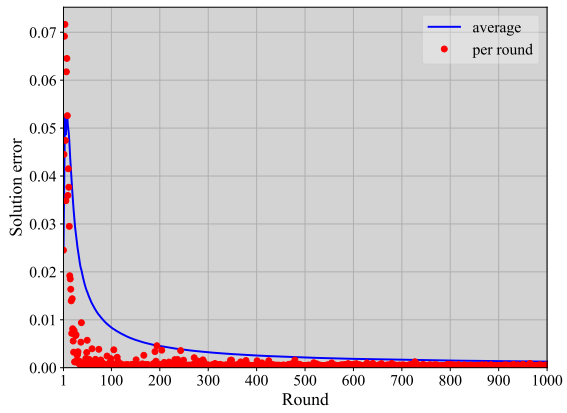


(b) Over $T = 1000$ rounds

Figure 4. Objective error for the Linear Knapsack instance with $n = 1000$ items



(a) Over $T = 100$ rounds



(b) Over $T = 1000$ rounds

Figure 5. Solution error for the Linear Knapsack instance with $n = 1000$ items

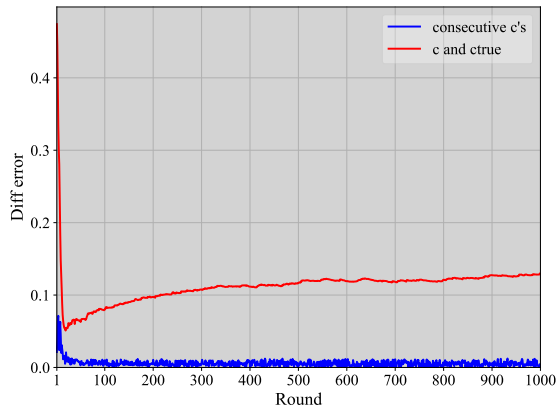
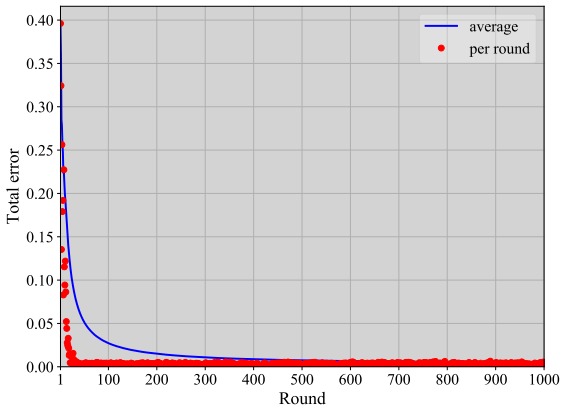


Figure 6. Distance of c_t to c_{true} and c_t to c_{t-1} for the Linear Knapsack instance with n = 1000 items over T = 1000 rounds

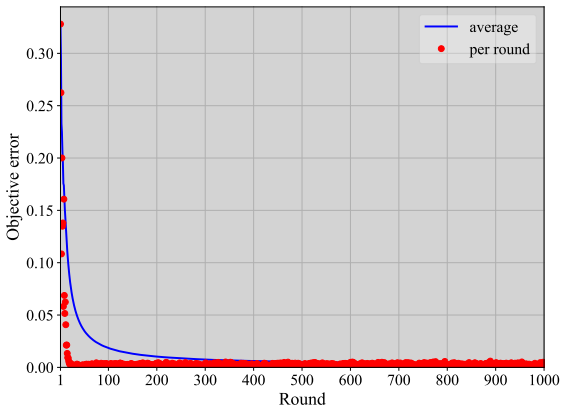
quickly, but that eventually our algorithm converges to an alternative objective function. This confirms that convergence towards the true objective function cannot not be expected in general, see our discussion in Section 3. Nevertheless, even without finding the true objective function, the solutions obtained from the learned sequence of objective functions are undistinguishable from the solutions chosen be the expert on average.

Figures 7 and 8 show similar findings for the Integer Knapsack with 1000 items, where we would like to point out that this is the much harder optimization problem, theoretically.

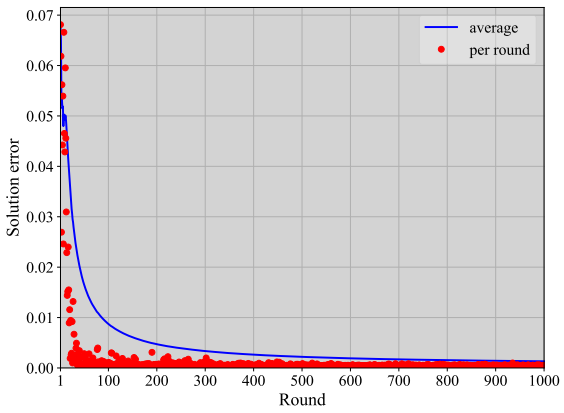
We also did some experiments where the learner is only using the observations for a small number of rounds to update the objective function and from there on only uses this approximation. We depict these results in Figure 9. Finally, we show the same graphics for the case of the RCSP on a grid graph with $m = 15$ rows and $n = 30$ columns and 1000 observations in Figures 10 and 11, again confirming our theoretical results.



(a) Total error



(b) Objective error



(c) Solution error

Figure 7. The different error measures for the Integer Knapsack instance with $n = 1000$ items over $T = 1000$ rounds

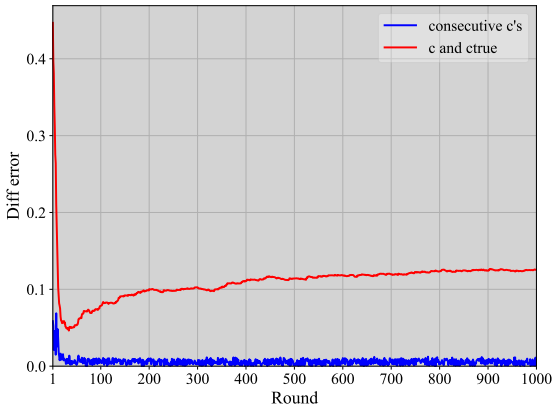


Figure 8. Distance of c_t to c_{true} and c_t to c_{t-1} for the Integer Knapsack instance with $n = 1000$ items over $T = 1000$ rounds

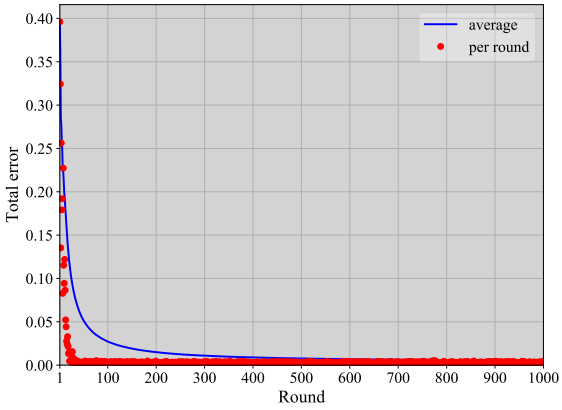
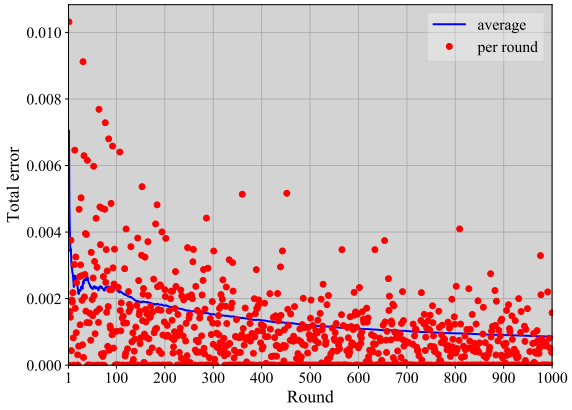
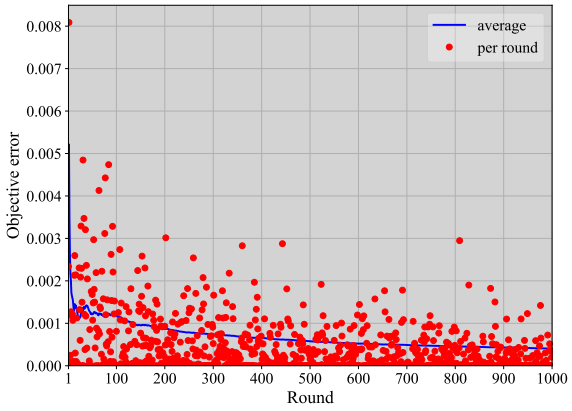


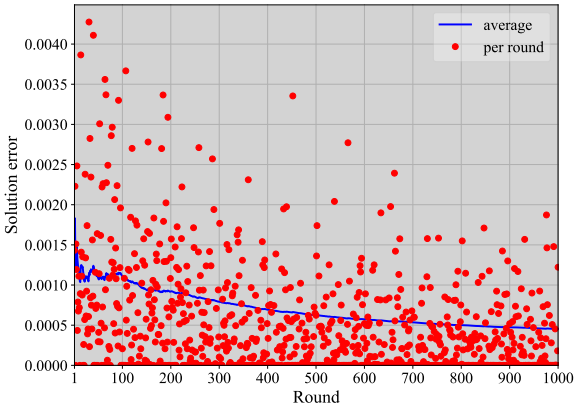
Figure 9. Total error for Integer Knapsack Instance with $n = 1000$ items. Here we only performed updates of the objective for the first 100 rounds and kept c_t constant after that. As can be seen, the learned function c_{100} performs very well. Even after 100 rounds enough information about c_{true} has been learned to be competitive for the next 900 rounds. Note though, that if the parameters are chosen adversarial and not at random as done here, this would not be possible in general.



(a) Total error



(b) Objective error



(c) Solution error

Figure 10. The different error measures for the RCSP instance on a grid graph with $m = 15$ rows and $n = 30$ columns over $T = 1000$ iterations

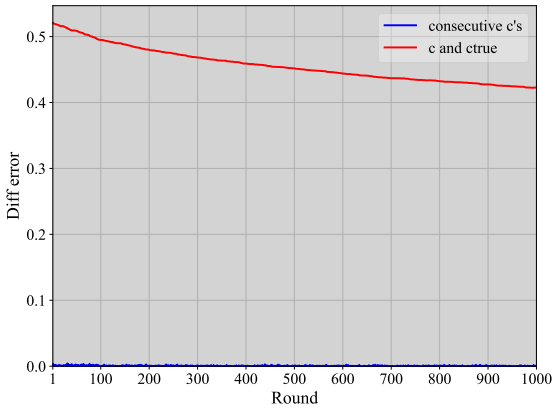


Figure 11. Distance of c_t to c_{true} and c_t to c_{t-1} for the RCSP instance on a grid graph with $m = 15$ rows and $n = 30$ columns over $T = 1000$ iterations