Resource Allocation using Decision Focused Learning

Qazi Zarif Ul Islam

December 12, 2024

1 Abstract

2 Introduction

"Given some input data, output data, and a feasible input space, determine the optimal combination of input data to achieve the desired output"—this is the fundamental problem we seek to address. For instance, consider a scenario where the per-unit profit for Product A and Product B is unknown, but historical data on production quantities and corresponding total profits is available. The task is to determine the optimal production quantities of Product A and Product B to maximize profit, given resource constraints, without explicit knowledge of per-unit profits.

Traditional data-driven techniques, such as Machine Learning (ML) and statistical methods, have achieved remarkable success in prediction and inference. However, they are often inadequate for problems where the goal is not only to predict but to prescribe decisions that optimize outcomes under constraints. For instance, while ML models can predict profits based on past data, they do not inherently account for how the predicted outcomes influence or are influenced by decision variables.

This paper argues that solving such problems requires integrating learning with optimization to form a cohesive decision-making framework. Specifically, we propose a framework rooted in **Predict, then Optimize**, which aims to bridge the gap between predictive models and optimization problems.

The rest of the paper is structured as follows: Section 2.1 discusses the central tenets of machine learning and its limitations in solving decision-making problems. Section 2.2 explores existing techniques for decision-making using data, such as optimization, stochastic programming, and data-driven control. Finally, we propose our framework in Section 3 and outline potential applications and future directions.

3 Background

3.1 The central tenets of Machine Learning

Machine learning (ML) is a field that primarily focuses on predicting some outcome, given some data. The central tenet of using such a framework is that the input variables that are used are somehow correlated with the outcome. For example, in the popular iris dataset, the input variables are the sepal and petal characterisitcs and the outcome is one of three species of the iris flower. The secondary tenet is that no observation made in this world is by itself the true representation of what is being observed and so *all observations* are noisy (contain noisy information). Thus, it follows that a single prediction made from a single noisy observation is also noisy (Noisy input gives noisy output). However, time and time again, such predictions have proven helpful. For example, a $\geq 90\%$ accurate transformer is deemed capable enough to complete incomplete passages or poems, computer vision systems are used to identify objects, understand human sentiments, etc. These wonders are due to the fact that all phenomena are generated by probability distributions and a random sample approaches a limiting probability distribution as the size of the sample increases. Thus, even though the learner is unaware of the underlying limiting distribution, as we increase the sample size, the particular error-correction algorithm of the learner leads it towards the true prediction.

3.2 Techniques that drive decision making

ML based products have been used primarily to make tasks more efficient, not to induce social policy and decision. We trust qualitative arguments when deciding where to build a hospital or where to put more health care funding. One explanation for this is that we can determine biases in an argument instantly and doing the same for a learner is much harder, simply due to the secondary tenet (see 3.1). Another reason for the insuitability of ML techniques in this area is that the predictions do not take into account *its own effects* on downstream decisions. But where we have not trusted predictions, we have trusted optimization and statistical testing. **Optimization techniques** have been heavily relied on when a deterministic answer is deemed to exist. Examples of such problems include The Travelling Salesman problem, The Knapsack problem, etc. Apart from **Stochastic programming**, a particular paradigm of optimization method, these problems differ from Machine Learning problems in that the problem is not solved using data but rather a fixed set of conditions. This fixed set contains a function relating the inputs to the output and equations that give the 'feasible answer region'. But, there are numerous problems where we

don't have a function that gives the input-output relationship but yet we want to optimize the inputs for either maximizing or minimizing the output based on some constraints on the inputs. For example, the problem of socio-economic resource allocation in order to lessen viral infections. **Stochastic programming** however, derives the *expected objective function* from historical data and then minimizes that, which is similar to Machine Learning settings. Another area that contains techniques that lead to decisions from data as opposed to predictions from data is **statistical hypothesis testing**. However, using hypothesis testing requires knowledge of the underlying probability distributions and real world data does not explicitly express the probability distribution it is generated from. **Data driven control** provides a way to *control* the input variables and thereby, optimize them for a desired output however, requires knowledge of the input-output relationships. When this relationship is unknown, **System identification** is used to derive it. Data driven control and system identification are usually applied to dynamic systems where the variables describing the system changes with time. It is however possible to use another variable as a proxy for time to hold information about how the system changes with that proxy variable. First, we need to have a model structure, for example,

$$y(k) + ay(k-1) = bu(k) \tag{1}$$

Where **a** and **b** are adjustable (learnable) parameters. From this model structure, we use historical (input, output) data to derive the learnable parameters. Usually, the input and output data are recorded through time. If we instead have static data, though it is possible to simply index every observation and then use the indices as k, when the data is shuffled, it changes the dynamical system itself. Thus every shuffling order of the data gives a new dynamic system. Besides, there is no inherent temporal information in the indices, they are simply numbers with no physical or temporal meaning.

Genetic algorithms on the other hand, use search algorithms inspired by population behaviours that exist in nature to find the global minimum of a manifold.

The above limitations called for new techniques that can optimize inputs simply using historical data, by predicting the relationship between the inputs and ouput and then using the predicted relationship to formulate an optimization problem. [4] proposed the name "**Predict, then optimize**" for this framework and used the term '**Decision Focused Learning**' (DFL) to describe the techniques. To our knowledge, [1] was the first to propose such a method, wherein a trained predictive model was used to predict a cost vector, given an input vector, and then the cost vector and input vector were used to form an objective function for an optimizer to solve. The simplest representation of such an objective function is,

$$f(\mathbf{x}, \mathbf{c}) = \mathbf{x}.\mathbf{c} \tag{2}$$

The predictive model needs to be trained in a supervised manner and so, we must bear knowledge of how each variable in the input vector $\mathbf{x}$ individually affects the function f, which is not available in many real-world scenarios. To our knowledge, [5], was the first to apply DFL in a real-world scenario to optimize the scheduling of live service calls in a maternal and child health awareness program. They used socio-demographic features (e.g., age, income, education) and historical engagement states (e.g., whether beneficiaries listened to calls) as inputs, while the engagement outcomes (e.g., listening behavior) served as outputs. The model predicted transition probabilities between engagement states (engaging/non-engaging) under different actions (call/no call). These predicted probabilities were used to compute Whittle Indices [6], which prioritized beneficiaries for live service calls based on their potential to benefit. We can use a more simplified approach wherein we are given input-output pairs, $(\mathbf{x}, \mathbf{y})$ and we have to produce a symbolic function, expressing the mapping between the input and output. Thus, in the end, the problem is simply one of **symbolic regression** (SR). After the symbolic expression of the function is obtained via symbolic regression, we use it as the objective function. Symbolic regression has been used to add a layer of 'interpretability' to black box machine learning methods that learn functions. We start with a set of possible arithmetic operators known as the 'library' and define the dimensionality of the function space. For example, a library could be $L = \{id(.), add(.,.), sub(.,.), mul(.,.), +1, -1\}$. This particular library can compose the set of all polynomials in one variable with integer coefficients [3]. [3] conducted a comprehensive survey on SR methods and divided the techniques into 4 fundamental types; (1) Regression based methods (2) Tree based methods (3) Physics inspired methods and (4) Mathematics inspired methods. Genetic algorithms can actually be used as a tree based method. Recently, **Kolmogorov Arnold Networks** were shown to perform better at Symbolic Regression than any other method [2].
In the following section. We discuss two approaches that we think could be adopted to solve resource allocation problems based on historical data.

4 Formalization of the problem

Suppose the goal is to determine a combination of inputs $\{x\}_n$, constrained by some conditions , that maximizes or minimizes an output, which is a function of $\{x\}_n$. This problem can be described as,

$$\text{Given,} \tag{3}$$
$$f(\mathbf{x}) = \mathbf{c} \cdot \mathbf{x}, \tag{4}$$
$$g(\mathbf{x}) \geq 0, \tag{5}$$
$$h(\mathbf{x}) = 0 \tag{6}$$
$$\text{Determine,} \tag{7}$$
$$\mathbf{x}^* = \arg\min_{\mathbf{x}} f(\mathbf{x}), \tag{8}$$

Where $\mathbf{c}$ is the coefficient vector that is unknown. Fortunately, ML techniques provide a way to estimate this vector from data. If $\hat{\mathbf{c}}$ is the predicted coefficient vector, the problem becomes,

$$\text{Given,} \tag{9}$$
$$f(\mathbf{x}, \hat{\mathbf{c}}) = \hat{\mathbf{c}} \cdot \mathbf{x}, \tag{10}$$
$$g(\mathbf{x}) \geq 0, \tag{11}$$
$$h(\mathbf{x}) = 0 \tag{12}$$
$$\text{Determine,} \tag{13}$$
$$\mathbf{x}^*(\hat{\mathbf{c}}) = \arg\min_{\mathbf{x}} f(\mathbf{x}, \hat{\mathbf{c}}), \tag{14}$$

Where $\mathbf{x}^*$ is a function of $\hat{\mathbf{c}}$ because it is dependent on the prediction made by the learner. There are two ways of approaching this problem: (1) Modifying the ML learner so that it incorporates the optimization problem into its learning algorithm or, (2) Predicting and then, optimizing, which follows the method in [1].

4.1 Decision Focused learning (DFL)

In this approach, we don't care about the accuracy of the predictions, $\hat{\mathbf{c}}$, but rather how accurate the decision *based on* the prediction is $(\mathbf{x}^*(\hat{\mathbf{c}}))$, which we call *Regret*. The Regret is formally expressed below.

$$Regret(\mathbf{x}^*(\hat{\mathbf{c}})) = f(\mathbf{x}^*(\hat{\mathbf{c}}), \mathbf{c}) - f(\mathbf{x}^*(\mathbf{c}), \mathbf{c}) \tag{15}$$

Where $f(\mathbf{x}^*(\mathbf{c}), \mathbf{c})$ is the decision that would have been obtained had the optimizer exact knowledge of $\mathbf{c}$. $f(\mathbf{x}^*(\mathbf{c}), \mathbf{c})$ is known as 'full information decision'. Here, as $\hat{\mathbf{c}} \to \mathbf{c}$, $Regret \to$

0. So, the better the prediction, the lower the regret. Instead of making a point estimation, it is also possible to estimate the distribution of $\hat{\mathbf{c}}$ so that it is possible to guard against the worst case using the distributional knowledge, however more challenging.

4.2 Express Symbolically, then optimize (ESO)

5 Conclusions

We need to find a way to automatically discover relationships between the variables themselves to be able to describe the dynamics of the system.