Expert Systems With Applications 128 (2019) 287–300

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Constraint learning based gradient boosting trees

Abraham Israeli*, Lior Rokach, Asaf Shabtai

Department of Software and Information System Engineering, Ben-Gurion University of the Negev Beer-Sheva, Israel

ARTICLE INFO

Article history:
Received 12 August 2018
Revised 5 March 2019
Accepted 5 March 2019
Available online 19 March 2019

Keywords:
Machine learning
Regression
Gradient boosting
Gradient boosting trees
Constraint learning

ABSTRACT

Predictive regression models aim to find the most accurate solution to a given problem, often without any constraints related to the model's predicted values. Such constraints have been used in prior research where they have been applied to a subpopulation within the training dataset which is of greater interest and importance. In this research we introduce a new setting of regression problems, in which each instance can be assigned a different constraint, defined based on the value of the target (predicted) attribute. The new use of constraints is taken into account and incorporated into the learning process, and is also considered when evaluating the induced model. We propose two algorithms which are modifications to the regression boosting method. There are two advantages of the proposed algorithms: they are not dependent on the base learner used during the learning process, and they can be adopted by any boosting technique. We implemented the algorithms by modifying the gradient boosting trees (GBT) model, and we also introduced two measures for evaluating the models that were trained to solve the constraint problems. We compared the proposed algorithms to three baseline algorithms using four real-life datasets. Due to the algorithms' focus on satisfying the constraints, in most cases the results showed significant improvement in the constraint-related measures, with just a minimal effect on the general prediction error. The main impact of the proposed approach is in its ability to derive a model with a higher level of assurance for specific cases of interest (i.e., the constrained cases). This is extremely important and has great significance in various use cases and expert and intelligent systems, particularly critical systems, such as critical healthcare systems (e.g., when predicting blood pressure or blood sugar level), safety systems (e.g., when aiming to estimate the distance of cars or airplanes from other objects), or critical industrial systems (e.g., require to estimate their usability along time). In each of these cases, there is a subpopulation of all instances that is of greater interest to the expert or system, and the sensitivity of the model's error changes according to the real value of the predicted feature. For example, for a subpopulation of patients (e.g., patients under the age of eight, or patients known to be at risk), physicians often require a sensitive model that accurately predicts blood pressure values.

1. Introduction

In this research we introduce a new class of machine learning problems in which each instance can be assigned a different constraint defined based on the value of the target (predicted) attribute. We focus specifically on regression problems. Two examples of instance constraints in such problems include: (i) all predicted values of a specific population in the training set should be above or below a given threshold (e.g., for a model that predicts a person's blood pressure, as in Sideris, Kalantarian, Nemati, and Sarrafzadeh (2016), we might want to constrain the predicted values of a sub population to be higher than a specific value, due to a known medical condition affecting this subpopulation); or, (ii) all prediction errors of a specific population in the training set should be in the range of [-0.1, 0.1] (i.e., the interval doesn't necessarily have to be balanced between the positive and negative errors). A few real-life use cases involving such constraints include:

1. *Semiconductor company prediction process*. In this case, which was provided by a real industrial company, each instance represents a chip in a production line, and the aim of the regression model is to predict the chip's power consumption value in the very early phases of the production process. Since units with a very high power consumption value will not function as required (and hence should not be sold), we wish to constrain the error among this subpopulation. In addition, we are much more sensitive to under prediction rather than over prediction, since

* Corresponding author.
E-mail addresses: isabrah@post.bgu.ac.il (A. Israeli), liorrk@bgu.ac.il, liorrk@post.bgu.ac.il (L. Rokach), shabtaia@bgu.ac.il (A. Shabtai).

https://doi.org/10.1016/j.eswa.2019.03.011
0957-4174/

in the former case the company might end up selling chips that that consume too much power (since the prediction was lower than the true value), and the units may not function properly.

2. *Alcohol level prediction*. For a model that attempts to predict the level of alcohol in a person's blood by analyzing various sensors' data (e.g., accelerometer sensor in a smartwatch), as was explored in Nassi, Rokach, and Elovici (2016), we may want to define tight constraints on the instances with high alcohol levels. The reason for this is to predict the values of such cases more accurately, and thus, prevent a person from driving while intoxicated. We might also want these constraints to be one-sided, since it is better to over predict such cases rather than under predict them (one-sided constraint configuration will be explained in Section 4.1).
3. *Bid price prediction*. For a model that predicts the suggested prices to offer for items in different bids, we might want to more accurately predict specific items (due to business needs). In a standard bid, the highest offer wins everything, which means any underprediction is irrelevant. In such a configuration we would need to constrain specific items which we are more interested in with a one-sided, overprediction constraint.
4. *Blood pressure prediction*. When predicting a patient's blood pressure, as in Sideris et al. (2016), the model may need to be much more accurate for the prediction of extremely high/low blood pressure values, since patients with such blood pressure values are at greater risk. In such a configuration we would need to constrain these extremely high/low blood pressure values to have a relatively low prediction error.
5. *Credit risk prediction*. The credit risk prediction research area Atiya (2001) which identify loans with a high risk of default. Some of the algorithms in this domain are aimed at predicting the total amount of money that will paid back per a given amount of credit. In such cases, enabling the algorithms to accurately predict the highest non refunded credits can be very useful, since such credits are at the center of interest by the financial institutes providing the loans.
6. *Cyber-physical systems*. Prediction algorithms are one of the central components of cyber-physical systems (CPSs) and Internet of Things (IoT) platforms. As described in Li et al. (2011); Sami Sivri and Oztaysi (2018), in many cases, a regression model provides ongoing predictions of environmental variables (e.g., temperature, or distance of a car from the sidewalk) In such cases, the model's assurance is extremely important in specific cases (e.g., when the temperature approaches extreme values or when the distance between the car and the sidewalk is too close), and a prediction model which takes instance level constraints into account can allow the system to meet its most sensitive prediction cases.

The central motivation driving this research comes from problems like those mentioned above, in which two central factors exist: (1) instances are not equally important, and (2) prediction error is not balanced (either between instances or between over and underprediction per instance). In such problems, the prediction error function is very complex and requires an algorithm that takes into account the prediction error per instance constraint. It is our understanding that currently in such settings constraints are usually taken into account more generally (as will be further explained in Section A.7); however, these problems could benefit from a new constraint definition in which constraints per instance are applied. Currently, the general solution for these kinds of problems involves assigning weights to the constrained instances, thus increasing their importance, and focusing the learning algorithm on the constrained instances (Carroll, 2017; Weisberg, 2005). However, there are two main drawbacks to such a solution: (1) it assigns a weight to all of the constrained instances, regardless of whether the constraint can be easily satisfied by the model, and (2) it is not designed to consider cases in which (a) the predicted value of an instance may be very accurate but unable to meet the desired (tight) constraint, or (b) the predicted value of an instance can be relatively inaccurate but still meet the desired constraint. In this manner, the constraints defined can introduce a new dimension in the machine learning domain - specifically, the extent to which a constraint is satisfied should be taken into account during the learning process, as well as when evaluating the induced model. One of the proposed algorithms we present is inspired by the AdaBoost algorithm, in which the importance of the instances in the training set (i.e., weights) are determined in each iteration, not only by the prediction error, but also based on a new component defined as the constraint satisfaction error. It should be noted that the use of the suggested algorithm will likely affect the general prediction error metric (e.g., RMSE - Root Mean Square Error) - consistent with the 'no free lunch theorem' in machine learning problems (Wolpert, 1996). In general, the impact usually depends on the percentage of instances constrained, the tightness of the constraints, and the general hyperparameters used during the learning process.

Therefore, we summarize the main contributions of this research are as follows:

- *We introduce a new and generic concept of constraint learning problems - the new concept adds another dimension of interest to the problem definition. In previous cases, we were equally concerned about instance level errors and were indifferent to over/under prediction. The new concept takes these two aspects into consideration when finding the most suitable solution to the problem through instance level constraints defined as part of the new problem settings.*
- *We introduce a new setting of ML regression problems* - this new setting is a subset of the proposed general class of constraint learning problems. It refers specifically to constraints defined for the target attribute of regression problems. Such constraints will allow us to build machine learning algorithms which aim to satisfy the constraints and eventually force the learning process to adapt the induced regression model to comply with the defined instance level constraints, subsequently generating a model with a higher level of assurance, since the model is trained to provide more accurate results for the constrained and more important cases.
- *We present a new regression algorithm* - we propose, implement and evaluate a new variation of the gradient boosting trees (GBT) algorithm, specifically designed to solve the constraint learning problem we introduced above. The new GBT-based algorithm incorporates a novel component to the regular regression error component which is optimized through the learning process. More specifically, we present two variations of the GBT algorithm, each of which uses a different constraint error component (L1 norm-error or the step function error, as described in Section 4.1) and discuss the differences in functionality and performance of the two proposed components. Note that although the constraints are defined for each instance individually, it can eventually be derived from a more general and broad definition of the constraint; for example, we would like to have a specific error limit on the predicted blood pressure for all patients under the age of 35 with a specific disease. Although in this research we present a constraint regression learning algorithm that is based on the GBT, the underlying regression model can also be implemented using other (iterative or gradient-based) regression methods, such as neural networks regression models.
- *We propose new evaluation measures* - as part of the problem definition, two unique measures for evaluating any

implemented instance level constraint regression algorithm are presented. Each measure calculates the satisfaction of the constraints differently and each can be relevant in different use cases. These two measures are taken into account when evaluating a solution along with standard regression measures (e.g., prediction error).
- *We created new code and made it available* - all Python code developed as part of this research can be found in the project's GitHub repository.[1]

The rest of this paper is organized as follows. In Section 2, we briefly review relevant work in the field of constraint learning. Section 3 is dedicated to background about boosting methods. In Section 4, we present the problem definition, relevant hyperparameters, and objective metrics we wish to optimize. In Section 5, we present the proposed algorithms. Sections 6 and 7 provide an in depth description of the experiments we conducted and the results we obtained. In Section 8 we present highlights of our analysis related to the algorithms' results. Section 9 contains short discussion about few algorithm's aspects and in Section 10 we summarize our work and presents future research options.

2. Related work

Related work summary can be found in the Appendix section, Table A.7.

Various aspects of constraint learning have been discussed in the literature. Related work that is relevant to our research can be categorized as follows: (a) constraints that are defined for the learning algorithm's parameters, mainly in order to make the training process more efficient, (b) instance level constraints in fully/semi-supervised classification problems, (c) cost-sensitive learning, and (d) special cases of regression problems.

Category (a) includes previous research such as (Hoerl and Kennard (1970); Tibshirani (1996); Zhou, Tao, & Wu, 2011) which deals with constraints related to the algorithm parameters and explanatory features used. 'Lasso', 'Ridge', and Elastic Net regression models are examples of cases in which the algorithm is limited to the scope and format of the parameters that are used. Such settings allow researchers to address overfitting problems, but they don't in any way, handle instance level domain-specific constraints as suggested in this research.

Category (b) includes research such as DeSarbo and Mahajan (1984); Klastorin and Watts (1981) which focuses on classification problems in which the constraints are associated with the classes themselves (e.g., the size of a specific class must be below 100). In addition, semi-supervised problems are summarized in Nguyen (2010), including solution-based constraints. Most of such semi-supervised problems are classification cases which are largely solved with classification with pairwise constraints. This problem definition was used in Zhang and Yan (2007) and Nguyen and Caruana (2008) where some pairs of instances must (or must not) be classified to the same class. An additional research area which falls in category (b) is the field of prior knowledge - Chang, Ratinov, and Roth (2008) and Yu, Jan, Simoff, and Debenham (2007) solve classification problems through such a prior knowledge approach. In both articles, prior knowledge allows the hypothesis space, H, to be reduced to a size that does not significantly violate prior knowledge. The evaluation measure is composed of a standard prediction model and a measure which takes violations of the prior knowledge into account. Both articles deal with classification problems, and Yu et al. (2007) suggest a very interesting variant of SVM (i.e., VQSVM) which makes an improved choice of a kernel function based on prior knowledge. Note that the most recent problem definitions are classification oriented and in that sense are different from this research direction, since we deal with regression prediction problems.

Category (c) includes research such as Koenker (2005); Zhao, Sinha, and Bansal (2011) also related to our research. As described in Zhao et al. (2011), most research in the cost-sensitive learning field is related to classification problems. However, the article does mention a method for a least squares error model with cost-sensitive aspects. The cost-sensitive aspect refers to the overprediction versus underprediction trade-off. The regression method suggested in Zhao et al. (2011) can apply any given cost function to a regression model. A linear cost function is a special case of cost function which is solved via a quantile regression solution Koenker (2005). Such a solution allows us to better control the over/under prediction sensibility, but it applies such sensibility to all data points in the learning dataset and restricts the cost function so it is convex. In our research, both limitations don't explicitly exist since the cost-sensitive function comes in the form of an instance level constraint matrix.

Category (d) includes research such as the model presented by Coons (1978) which is another form of constraint regression model. The paper proposes a solution for cases in which some data points are considered more reliable than others (i.e., hard-points), and the solution fits these reliable data points exactly. The idea of restricting some of the fitted data points is very similar to the direction of our research, especially since both deal with regression problems, however two main differences exist: (1) the constraints, as defined in the current research, are not necessarily the same as the hard constraints referred to by Coons (1978), and (2) the way in which we solve the problem is based on an arithmetic solution rather than a closed algebraic solution.

Regression constraint problems are also mentioned in Harrell (2015) in the context of survival analysis where the true target value for some of the instances is down/up bounded but not fully known, meaning that the actual true value of some 205 instances in not known, which is different from our problem setting.

The simplest way to address constraints as we define them is to weight the constrained instances higher than other instances so the prediction algorithm will optimize the solution according to the imbalance weighting policy. However, there are two main drawbacks to a solution based on assigning weights to the constrained instances. First, such a solution assigns a weight to all of the constrained instances, regardless of whether the constraint can be easily satisfied by the model. Second, these solutions are not designed to consider cases in which the predicted value of an instance may be very accurate but does not meet the desired (tight) constraint, or alternatively, cases in which the predicted value of an instance is relatively inaccurate but still meets the desired constraint. We use this solution as a benchmark in our experiments, further described in Section 6.3.

Furthermore, none of the aforementioned research deals specifically with constraints related to the continuous target attribute given as input for a regression model. In that sense, the problem definition and solution we suggest is novel and unique.

3. Background

3.1. Gradient Boosting Trees

The Gradient Boosting Trees (GBT) algorithm serve as the baseline in this research. This algorithm is an information-theoretical discriminative predictor. A series of weak learners (decision trees) is constructed, boosting regression accuracy by combining the respective learner Friedman (2002); Schapire (1990). GBT classifiers tend to work well on a broad range of datasets; they also work

[1] https://github.com/abrahami/Constraint-Learning.

Table 1
Notations of variables used in this paper.

	Notation	Explanation
General Notations	y_i	The actual (true) target feature value of the i'th instance.
	$\hat{y}_i$	The predicted target feature value of the i'th instance (predicted by a regression model).
	n	Total number of instances in the dataset.
	M	Number of iterations in the GBT model.
	w_i	Weight assigned to the i'th instance (before/during an algorithm run).
	θ	Learning rate of the GBT algorithm.
Constraints Notations	CM	A matrix used to define the constraints for the instances. The matrix is of size (n, 2), where n is the number of instances, and the first and second column values defines the lower and upper bounds (respectively) for the relevant instance. For unconstrained instances both lower and upper bounds are assigned with $-\infty$ and ∞.
	$\tilde{y}_i$	The constraint error of the i'th instance. The error is measured based on the $L1$ distance from the constraint interval and receives a non-negative value in any situation.
	C_n	Number of instances that are constrained in the learning dataset (i.e., instances in which $LowerBound \neq -\infty$ or $UpperBound \neq \infty$).
	C_η	Dynamic Weight algorithm hyperparameter.
	C_γ	Constraint Loss algorithm hyperparameter. This notation is also used in the general definition of the problem's objective as a general term regarding the importance given to constraints in the learning phase.

effectively with a combination of various unnormalized feature types. Different hyperparameters related to the algorithm have been used; some are related to each tree that is built (e.g., maximum tree depth), whereas others are related to the overall model's configuration (e.g., numbers of trees to build).

3.2. Other gradient boosting algorithms

The algorithm we suggest can be applied to other boosting methods, and not only GBT algorithm (which is in focus along this paper). There are several modern implementation of Gradient Boosting Machine (GBM) methods that gained popularity among practitioners:

1. **XGBoost** XGBoost (short for eXtreme Gradient Boosting) Chen and Guestrin (2016) added several optimizations and refinements to the original GBM. The most important refinement in XGBoost is that it adds a regularization component to the loss function aimed at creating ensembles that are simpler and more generative.
2. **LightGBM** LightGBM is another gradient boosting tree implementation Ke et al. (2017) that introduces two new improvements. The first is exclusive feature bundling which addresses a training set with a large number of features. The idea is to look for features that never take nonzero values simultaneously and combine them into a new single feature. The second is gradient-based one-side sampling (GOSS) which aims to address a training set with a large number of instances; inspired by the idea behind the AdaBoost algorithm, in each iteration the training process focuses particularly on instances that were misclassified by preceding models. Similarly, when moving from one tree to the next, instead of using all instances, GOSS keeps all of the instances with large gradients but only a random sample of instances with small gradients.
3. **CatBoost** CatBoost is an open-source gradient boosting tree library (Prokhorenkova, Gusev, Vorobev, Dorogush, & Gulin (2018)). As the name implies, this package can effectively handle categorical features. The simplest way to address categorical features is to convert them into one-hot encoding, namely having a dedicated binary variable for each category value. However, when the cardinality of the categorical feature is high, this encoding might run into various difficulties, including: large memory requirements, extensive computational costs, and difficulties that arise due to high-dimensional training data. CatBoost converts each category value to a numerical value that approximates the expected value of the target variable. To avoid data leakage in the estimation of the target attribute, CatBoost uses a random permutation of the training set. Then, given a certain instance, it calculates the corresponding value by averaging the target variable value of the preceding instances in the permutation that share the same category value. In addition to addressing categorical variables, CatBoost also mitigates the bias problem that exists in the pointwise gradient estimates. In a regular gradient boosting tree algorithm, the gradients are estimated using the same data points that were used to train the tree. This means that the trees predictions and the gradient estimations are dependent. Moreover, the residuals estimated on the training set have absolute values that are less than those expected on unseen instances, i.e., the next tree that is trained to predict the gradient, will tend to underestimate the gradient. Thus, CatBoost also uses the abovementioned idea of random permutation to address the bias issue. Specifically, the gradient estimate for a certain instance is based on the prediction learned using the preceding instances.

4. Problem definition

4.1. Problem settings

Table 1 presents the general and constraint related notations used in this paper. The constraint regression problem we consider receives two types of input parameters: (1) a constraint matrix (CM), and (2) constraint hyperparameters. A CM is a table that contains a row per instance with lower and upper constraint boundaries (limits). These limits are referred to as the *Lower Bound* and *UpperBound*. Each instance should have either a two-sided constraint or a one-sided constraint or be unconstrained; an example of these target attribute values with the various types of constraints can be seen in Fig. 1, where seven of the ten instances are constrained, and the constraint type (one or two-sided or unconstrained), as well as the interval size of each constraint can be specified for each instance. We assume that the true target

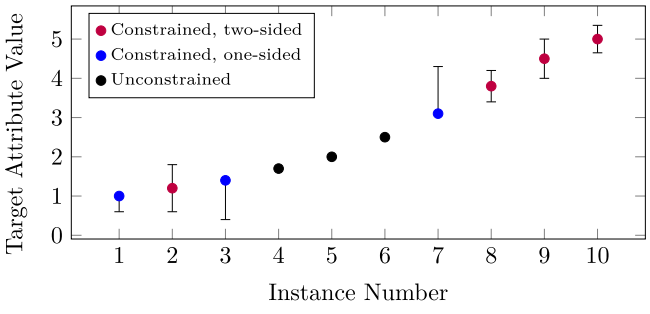

Fig. 1. An example of target attribute values with constraint intervals.

attribute value of each instance is within the range specified in the matrix.

4.2. Problem objective

The problem which is the focus of this paper is defined as a generalization of the standard regression problem, in which an additional problem objective comes into play along with the regular objective of the standard regression problem (the mean squared error in standard cases). The measure of the new problem objective is to minimize Eq. (1); the equation reflects the idea of splitting up the new objective measure into two parts: the regular regression measure and a new constraint measure which is weighted based on the C_γ parameter. In cases in which $C_\gamma = 0$, the problem objective's measure represents a regular regression case. C_n represents the number of instances that are constrained in the learning dataset (i.e., instances in which *Lower Bound* $\neq -\infty$ or *Upper Bound* $\neq \infty$). $\tilde{y}_i$ represents the constraint error of the $i'th$ instance and receives a non-negative value in any situation. In Eq. (1), f can be any function applied on $\tilde{y}_i$.

$$\underbrace{\frac{1}{n}\sum_{i}^{n}(y_i-\hat{y}_i)^2}_{\text{Regression Error}}+\underbrace{C_\gamma\frac{1}{C_n}\sum_{i}^{C_n}f(\tilde{y}_i)}_{\text{Constraint Error}} \tag{1}$$

In our research we focus on two such functions, as can be seen in Eqs. (2) and (3). Both functions are equal to zero in cases in which the constraint of the $i'th$ instance is satisfied. Eq. (2) is equivalent to the $L_1 - norm$ error, and Eq. (3) is a step function which receives the value of one in any case in which the constraint is not satisfied.

$$\textit{Absolute Value: } f(x) \triangleq |x| \tag{2}$$

$$\textit{Indicator: } f(x) = 1_{x>0} \triangleq \begin{cases} 1, & x > 0 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

5. Algorithms

As mentioned earlier, we suggest a solution which uses the GBT algorithm, taking advantage of two of the algorithm's characteristics - (i) iterative progress towards convergence, and (ii) gradient direction steps. We present two algorithms, each of which addresses the constraints differently. Both algorithms are based on the GBT model; the algorithms' pseudo code can be found at the end of this section. The hyperparameter θ refers to the learning rate, M refers to the number of algorithm iterations, CM refers to the constraint matrix, and $F_m(X)$ refers to the model suggested in the $m'th$ iteration. In both of the algorithms r_{im} represents the pseudo-residuals according to the loss (L) function. In the Dynamic Weight algorithm (Algorithm 1), this is the existing MSE loss function, whereas in the Constraint Loss algorithm (Algorithm 2) this is based on a tailor-made loss function which takes into account the constraints' dissatisfaction level. The f and $\tilde{y}_i$ notations used in Algorithm 2 are based on the definitions mentioned in Table 1.

The Dynamic Weight algorithm takes advantage of the iterative working method of GBT and updates instances' weight (i.e., w_i) after each iteration (line 7 in the algorithm below), based on the constraints' status. In cases in which there is no constraint, the instance's weight should not be changed in any step of the algorithm. Such a dynamic weight concept is based on the AdaBoost algorithm (Freund & Schapire, 1997), however this concept is applied within the context of constraint satisfaction. In each iteration a decision is made regarding whether to change each instance's weight according to the constraint satisfaction's binary status. The

Algorithm 1: Dynamic Weight.

Data: Explanatory feature matrix, X; Target attribute vector, y
Parameters: M, θ, CM, C_η
Result: Predictive model, $F_M(X)$
1 $F_0(X) = \underset{\xi}{\text{argmin}} \frac{1}{n}\sum_i^n L(y_i, \xi)$; ▷ `Initialize model with a constant`
2 $w = \vec{1} \iff w_i = 1, \quad \forall i \in n$; ▷ `Constant weight for all instances`
3 **for** $m = 1$ *to* M **do**
4 $r_{im} = -[\frac{\partial L(y_i, F(X_i))}{\partial F(X_i)}]_{F(X)=F_{m-1}(X)} = y_i - F_{m-1}(X_i) \quad \forall i \in \{1 \dots n\}$;
5 fit a tree base learner, $h_m(X, w)$, to the pseudo-residuals (i.e. r_{im});
6 $F_m(X) = F_{m-1}(X) + \theta h_m(X, w)$; ▷ `Update the aggregated model`
7 re-weight w according to constraint's satisfaction status:
$w_i = \begin{cases} w_i & \text{constraints were satisfied in current iteration} \\ w_i(1 + C_\eta) & \text{otherwise} \end{cases}$
8 **end**
9 **return** $F_M(X)$

Algorithm 2: Constraint Loss.

Data: Explanatory features matrix, X; Target attribute vector, y
Parameters: M, θ, CM, C_γ
Result: Predictive model, $F_M(X)$
1 $F_0(X) = \underset{\xi}{\text{argmin}} \frac{1}{n}\sum_i^n L(y_i, \xi)$; ▷ `Initialize model with a constant`
2 **for** $m = 1$ *to* M **do**
3 $r_{im} = -[\frac{\partial L(y_i, F(X_i), CM(i))}{\partial F(X_i)}]_{F(X)=F_{m-1}(X)} \quad \forall i \in \{1 \dots n\}$
where:
$L(y_i, F(X_i), CM(i)) = \overbrace{(y_i - \hat{y}_i)^2}^{RegressionError} + \overbrace{C_\gamma f(\tilde{y}_i)}^{ConstraintError}$;
4 fit a tree base learner, $h_m(X)$, to the pseudo-residuals (i.e. r_{im});
5 $F_m(X) = F_{m-1}(X) + \theta h_m(X)$; ▷ `Update the aggregated model`
6 **end**
7 **return** $F_M(X)$

hyper-parameter that controls the change rate is C_η, which is expected to be a number in the (0, 1) range and is analogous to the learning rate parameter in the GBT algorithm.

The Constraint Loss algorithm takes advantage of GBT's gradient step framework, and tunes the gradient to include constraints in each iteration (line 3 in the algorithm below). The constraint gradient will not be changed for cases in which there are no constraints provided. The hyperparameter that controls the importance proportion given to the constraint gradient and the standard regression gradient is C_γ and it is expected to be a number in the (0, ∞) range. As can be seen in the expansion of line 3, the loss function related to the constraints is generally defined as $f(\tilde{y}_i)$. Two examples of such functions can be seen in Section 4.2. We used the *Absolute Value function* in our study, since it is widely used as a loss function, and its gradient can easily be computed. Such a loss function is analogous to the L_1 loss function in regression problems (which is usually defined as the *LAD* problem and is discussed in Narula & Wellington (1982)).

Note that the loss function used by the Dynamic Weight algorithm is based on the Indicator function (shown in Eq. (3)), which is not a continuous loss function. Since the derivative of the Indicator function cannot be computed (and hence the gradient step cannot be modified), we update the model by increasing the weight of instances that are not satisfying their constraint (see line 7 in Algorithm 1). In other cases (i.e., unconstrained instances, or instances that satisfy their constraints), the model is updated according to the original gradient steps of the GBT algorithm (see line 4 in Algorithm 1). The Constraint Loss algorithm is based on the Absolute Value function (shown in Eq. (2)). In this case, since the Absolute Value function is continuous and its derivative can be obtained, we use the updated loss function along the gradient steps instead of the original gradient steps of the GBT algorithm (see line 3 in Algorithm 2).

Except for these main changes, the algorithms follow the standard flow of a GBT algorithm. In this sense, the suggested solution wraps a given model and can be modified to work with any iterative and/or gradient descent framework. Constraint hyperparameters are used to tune the algorithm with regard to the constraints' status (see Section 7.1 for further details). Since the suggested solution wraps the GBT algorithm, there are additional hyperparameters related to the GBT model (e.g., tree depth, learning rate) that must be defined as well.

6. Experimental process

In order to validate the added value of the new algorithms, we defined the following evaluation metrics. Later in this section we describe the diverse datasets used and the experimental setup. All code developed as part of the research can be found in the project's GitHub open repository.[2]

6.1. Evaluation metrics

Based on the definitions presented in Section 4.1 and Table 1, we used the following metrics in order to measure the performance of the algorithms after adding the constraints: (a) MSE $\triangleq \frac{1}{n}\sum_i^n (y_i - \hat{y}_i)^2$ (b) CMSE $\triangleq \frac{1}{C_n}\sum_i^{C_n} \tilde{y}_i$ (c) CER $\triangleq \frac{1}{C_n}\sum_i^{C_n} \mathbb{1}_{\tilde{y}_i > 0}$.

A standard way of evaluating any regression solution is the MSE (mean squared error) which does not give any significance to constraints. CMSE (constraint mean squared error) and CER (constraint error rate), new evaluation measures developed for this research, take constraints into account, but each one evaluates the constraint satisfaction differently - either based on a more continuous measurement ability approach or a binary measurement ability approach (respectively). As mentioned in Section 5, Algorithm 2 is more suitable for optimizing the CMSE (i.e., metric (b)) because it optimizes the solution using a continuous gradient descent approach. Algorithm 1 is more suitable for optimizing the CER metric (i.e., metric (c)) as it examines, in a binary way, whether each constraint was satisfied or not. A high degree of correlation is expected to be found between the CMSE and the CER metrics, and a clear trade-off between optimizing the MSE metric (i.e., metric (a)) and the other two metrics will logically be found. The magnitude of the trade-off between the measures depends largely on the constraints' tightness and how many instances are actually constrained. We expect to see a decrease in the MSE measure in cases in which there are very tight constraints on a relative large proportion of the population.

[2] https://github.com/abrahami/Constraint-Learning.

6.2. Datasets

6.2.1. Parkinson's dataset

This dataset, which consists of 5875 instances and 25 explanatory features, was provided from UCI.[3] The dataset is composed of a range of biomedical voice measurements from 42 people with early-stage Parkinson's disease recruited to a six-month trial of a telemonitoring device for remote symptom progression monitoring. The target attribute is the UPDRS score, which is used to follow the longitudinal course of Parkinson's disease. We constrained the target attribute so that high values would be better predicted, by setting two-sided constraints on a subpopulation with the highest target values. This is based on the assumption that the output model should be more sensitive to very high UPDRS score values, since a high score reflects a progressive disease status and deserves higher treatment priority.

6.2.2. Industry dataset

This dataset, which consists of 25,150 instances and 813 explanatory features, comes from a real use case of a very large industrial company. As stated in Section 1, the problem definition is to predict, for each instance, the power consumption value in the very early phases of the production process. For business reasons, instances with high target attribute values are much more important than others, and overprediction is preferred for this subset of the population. Hence, we constrained the target attribute so that high values would be over predicted (one-sided constraints).

6.2.3. Flight dataset

This dataset, which consists of 272,520 instances and 35 explanatory features, comes from the OpenSky Network Project[4] and contains technical flight information (e.g., altitude, speed). The data used from this project focuses on predicting the altitude of the airplane, given other features, in order to detect anomalies and cases of unreliable reports. Using this dataset, we constrained the target attribute (i.e., altitude) with a two-sided constraint, so that low values would be better predicted. This is based on the assumption that an unreliable report is much more destructive during landing or takeoff than it is during cruising.

6.2.4. House sales dataset

This dataset, which consists of 21,613 instances and 18 explanatory features, comes from Kaggle[5] - a platform for predictive and analytical model competitions. The dataset was introduced to Kaggle users in 2016, and it consists of the prices of house sold in King County, Washington (USA), which includes the city of Seattle. The dataset includes homes sold between May 2014 and May 2015, and the original purpose of the competition was to predict final sale price of the house. We used this dataset as an analogy to a case involving multiple bids, where for each bid the highest offer wins everything. In such a case, underprediction leads to losing the bid offer, and hence a one-sided constraint has to be applied on specific items which are of greater interest (i.e., in this case we applied the constraint on houses with the highest or lowest target values).

6.2.5. CTSlices Dataset

This dataset, which consists of 53,500 instances and 386 explanatory features, comes from UCI.[6] The dataset is composed of

[3] https://archive.ics.uci.edu/ml/datasets/Parkinsons+Telemonitoring.

[4] https://opensky-network.org/.

[5] https://www.kaggle.com/harlfoxem/housesalesprediction.

[6] https://archive.ics.uci.edu/ml/datasets/Relative+location+of+CT+slices+on+axial+axis.

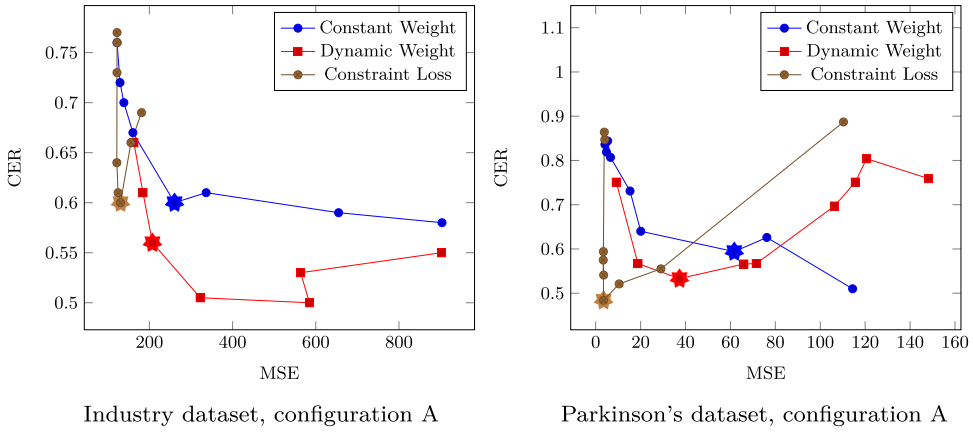

Fig. 2. Constraints' hyperparameter tuning. The optimization process was performed on a different parameter (e.g., C_η in the Dynamic Weight algorithm) for each model. Each point represents an algorithm's result with a different value for the constraint's hyperparameter. A star signifies the best value (the selected value), when taking the trade-off between the standard MSE measure and the constraint's related measure into consideration. In some cases overfitting was observed.

CT images from 74 individuals. The set of explanatory features consist of two types of features: (a) features describing the location of bone structures in the image, and (b) features describing air inclusions inside the patient's body. The target attribute is the relative location of the CT slice on the axial axis of the human body. We constrained the target attribute so that high values would be better predicted, by setting two-sided constraints on a subpopulation that has the highest target value.

6.3. Experimental setup

Each dataset was used twice, with different constraint configurations. In each of the experiments conducted, we used one of the four datasets and applied constraints to a specific subpopulation within this dataset based on the target attribute value, for example, constraining the top 10% of the target value instances. The constraint type (one-sided, two-sided or unconstrained) was set according to the use case. The constraint interval size, which also reflects the constraint's difficulty, was tested on a few values for each dataset. Regular cross-validation techniques were used during the experiments, and all of the results presented below are based on unseen test datasets. Each measure reported is the average value over five random iterations. For each dataset, hyperparameters related to the GBT model (e.g., maximum tree depth, number of iterations) were fixed, so that the results of the experiments would be comparable. The results of each experiment were compared to the following three baselines:

1. *Standard GBT algorithm* – a regular GBT algorithm, without taking constraints into consideration.
2. *Constant Weight baseline* – a GBT algorithm in which constrained instances are given a constant weight which, of course, is much higher than the unconstrained instances.
3. *Smart Weight baseline* – a GBT algorithm in which constrained instances are given a weight according to the tightness of the constraint which is always higher than the unconstrained instances.

7. Experimental results

7.1. Constraint hyperparameter tuning process

Although GBT hyperparameters were fixed for each dataset (based on a preliminary process we performed for each dataset in order to find the most suitable GBT hyperparameters), the *constraints' hyperparameters* were still optimized using a grid search method process. In the Dynamic Weight and the Constraint Loss algorithms we optimized the C_η and the C_γ, respectively, whereas we optimized the weight parameter in the Constant Weight baseline. During this process we used data that was held out from the original dataset, used for evaluation purposes. Fig. 2 shows two examples of this process, where the results in the figure are based on unseen data taken from the original dataset. The best hyperparameter for each algorithm (marked by a star in Fig. 2) was selected based on the criteria of minimizing the CER, and having the least effect possible on the MSE.

7.2. Parkinson's dataset results

The two constraint configurations examined are: constraining the top 20% of the target value instances, with a constraint interval size equal to 1% of the target attribute value (i.e., *configuration A*), and constraining the top 30% of the target value instances, with a constraint interval size equal to 2% of the target attribute value (i.e., *configuration B*). As can be seen in Table 2, the Constraint Loss algorithm outperforms all of the other algorithms in both configurations, maintaining a very low MSE value and relatively low values of the other constraint measures. Although the CER and CMSE values in configuration A are not the lowest achieved, the balance between all of the measures of interest suggested by this algorithm is the most worthwhile compered to other methods.

7.3. Industry dataset results

As explained in Section 6.2, only high target value instances needed to be constrained (one-sided). The two constraint configurations examined are: constraining the top 5% of the target value instances with a constraint interval size equal to 10% of the target attribute value (i.e., *configuration A*), and constraining the top 5% of the target value instances, with a constraint interval size equal to 5% of the target attribute value (i.e., *configuration B*). As can be seen in Table 3, compared to all of the other algorithms, the Dynamic Weight algorithm achieves the minimal values for both constraint measures. Although the MSE measure is affected in this case, the balance between all of the measures of interest obtained by this algorithm is the most impressive one compered to other methods.

Table 2
Parkinson's dataset - average and standard deviation results.

Algorithm	Configuration A			Configuration B		
	MSE	CER	CMSE	MSE	CER	CMSE
Standard GBT	4.65 ± 0.22	0.88 ± 0.023	7.67 ± 0.79	4.65 ± 0.22	0.74 ± 0.021	5.20 ± 0.48
Constant Weight	58.61 ± 3.54	0.58 ± 0.038	0.47 ± 0.37	57.11 ± 3.46	0.44 ± 0.023	0.43 ± 0.064
Smart Weight	6.77 ± 0.22	0.80 ± 0.036	1.38 ± 0.28	5.10 ± 0.21	0.65 ± 0.026	2.21 ± 0.29
Dynamic Weight	42.87 ± 3.82	0.55 ± 0.05	0.30 ± 0.16	60.25 ± 2.64	0.46 ± 0.016	0.33 ± 0.033
Constraint Loss	**3.76 ± 0.29**	**0.59 ± 0.062**	**1.88 ± 0.25**	**3.01 ± 0.26**	**0.40 ± 0.023**	**0.32 ± 0.006**

Table 3
Industry dataset results - average and standard deviation results.

Algorithm	Configuration A			Configuration B		
	MSE	CER	CMSE	MSE	CER	CMSE
Standard GBT	122.6 ± 4.98	0.75 ± 0.016	416.28 ± 69.38	125.66 ± 4.94	0.66 ± 0.012	371.27 ± 45.73
Constant Weight	256.03 ± 6.65	0.59 ± 0.02	221.04 ± 68.11	313.17 ± 10.47	0.55 ± 0.008	235.09 ± 43.6
Smart Weight	127.22 ± 4.18	0.68 ± 0.032	329.06 ± 60.83	125.92 ± 5.54	0.66 ± 0.008	370.82 ± 43.96
Dynamic Weight	128.92 ± 4.5	0.62 ± 0.014	264.55 ± 49.67	140.58 ± 6.85	0.5 ± 0.011	221.49 ± 43.68
Constraint Loss	**126.56 ± 4.68**	**0.58 ± 0.018**	**269.23 ± 44.63**	**139.76 ± 7.57**	**0.45 ± 0.016**	**204.15 ± 34.79**

Table 4
Flight dataset - average and standard deviation results.

Algorithm	Configuration A			Configuration B		
	MSE	CER	CMSE	MSE	CER	CMSE
Standard GBT	64.01 ± 2.95 e3	0.77 ± 0.008	114.75 ± 19.13 e3	63.92 ± 2.96 e3	0.64 ± 0.008	83.66 ± 8.12 e3
Constant Weight	**107.77 ± 4.84 e3**	**0.51 ± 0.014**	**69.89 ± 16.32 e3**	**141.11 ± 8.15 e3**	**0.40 ± 0.001**	**44.77 ± 11.88 e3**
Smart Weight	64.62 ± 3.9 e3	0.77 ± 0.007	119.80 ± 30.05 e3	64.66 ± 3.8 e3	0.64 ± 0.007	86.84 ± 14.51 e3
Dynamic Weight	**100.43 ± 2.68 e3**	**0.44 ± 0.004**	**88.54 ± 46.36 e3**	**145.02 ± 6.72 e3**	**0.33 ± 0.005**	**52.07 ± 14.03 e3**
Constraint Loss	61.87 ± 1.48e3	0.68 ± 0.005	102.16 ± 30.62 e3	85.97 ± 4.84 e3	0.45 ± 0.011	65.70 ± 11.41 e3

Table 5
House Sales dataset - average and standard deviation results.

Algorithm	Configuration A			Configuration B		
	MSE	CER	CMSE	MSE	CER	CMSE
Standard GBT	13.79 ± 0.68 e9	0.64 ± 0.019	35.54 ± 3.96 e9	13.79 ± 0.72 e9	0.51 ± 0.007	0.928 ± 0.17 e9
Constant Weight	14.61 ± 1.15 e9	0.59 ± 0.013	34.66 ± 6.46 e9	14.64 ± 0.15 e9	0.55 ± 0.008	0.8 ± 0.15 e9
Smart Weight	13.86 ± 0.76 e9	0.64 ± 0.02	35.8 ± 0.45 e9	13.76 ± 0.62 e9	0.51 ± 0.008	0.92 ± 0.15 e9
Dynamic Weight	**22.03 ± 1.37 e9**	**0.42 ± 0.01**	**27.09 ± 5.56 e9**	**17.84 ± 0.85 e9**	**0.43 ± 0.008**	**0.7 ± 0.16 e9**
Constraint Loss	13.76 ± 0.73 e9	0.64 ± 0.02	35.42 ± 3.84 e9	13.79 ± 0.78 e9	0.51 ± 0.011	0.92 ± 0.17 e9

7.4. Flight dataset results

The two constraint configurations examined are constraining the bottom 5% of the target value instances, with a constraint interval size equal to 5% of the target attribute value (i.e., *configuration A*) and constraining the bottom 10% of the target value instances, with a constraint interval size equal to 5% of the target attribute value (i.e., *configuration B*). The results presented in Table 4 show that both the Dynamic Weight algorithm and the Constant Weight baseline provide impressive results. The Dynamic Weight algorithm shows the best CER results, while the Constant Weight baseline is better when focusing on the CMSE. The CMSE measure suffers from very high variance values for all of the algorithms, a fact which leads us to focus and rely on the MSE and CER measures.

7.5. House sales dataset results

As explained in Section 6.2, constraints in this use case are one-sided (as illustrated in Fig. 1 for the seventh instance). The two constraint configurations examined are: constraining the top 20% target value instances, with a constraint interval size equal to 30% of the target attribute value (i.e., *configuration A*), and constraining the bottom 20% of the target value instances, with a constraint interval size equal to 30% of the target attribute value (i.e., *configuration B*). As can be seen in Table 5, compared to all of the other algorithms, the Dynamic Weight algorithm achieves the minimal values for both constraint measures. Although the MSE measure is highly affected when using the Dynamic Weight algorithm in this case, no other algorithm succeeded in achieving as impressive constraint measures as this algorithm.

7.6. CTSlices Dataset results

The two constraint configurations examined are constraining the top 10% of the target value instances, with a constraint interval size equal to 5% of the target attribute value (i.e., *configuration A*), and constraining the top 20% of the target value instances, with a constraint interval size equal to 5% of the target attribute value (i.e., *configuration B*). As can be seen in Table 6, compared to all of other algorithms, the Constraint Loss algorithm achieves the minimal CER values in both configurations. The CMSE gets its minimal value with the Dynamic Weight or the Constraint Loss algorithms (depends on the configuration). Over all, the Constraint Loss

Table 6
CTSlices dataset - average and standard deviation results.

Algorithm	Configuration A			Configuration B		
	MSE	CER	CMSE	MSE	CER	CMSE
Standard GBT	76.76 ± 0.35	0.91 ± 0.01	132.13 ± 7.01	76.76 ± 0.35	0.77 ± 0.007	92.1 ± 3.5
Constant Weight	81.23 ± 1.45	0.34 ± 0.02	6.43 ± 0.72	84.1 ± 1.3	0.5 ± 0.005	15.16 ± 0.98
Smart Weight	71.52 ± 1.11	0.7 ± 0.001	64.78 ± 2.81	76.2 ± 0.38	0.766 ± 0.005	88.9 ± 3.27
Dynamic Weight	111.21 ± 3.25	0.32 ± 0.01	3.63 ± 0.21	75.4 ± 0.47	0.74 ± 0.47	79.38 ± 3.24
Constraint Loss	84.52 ± 2.39	0.21 ± 0.03	3.97 ± 1.5	81.9 ± 1.4	0.23 ± 0.02	3.39 ± 0.68

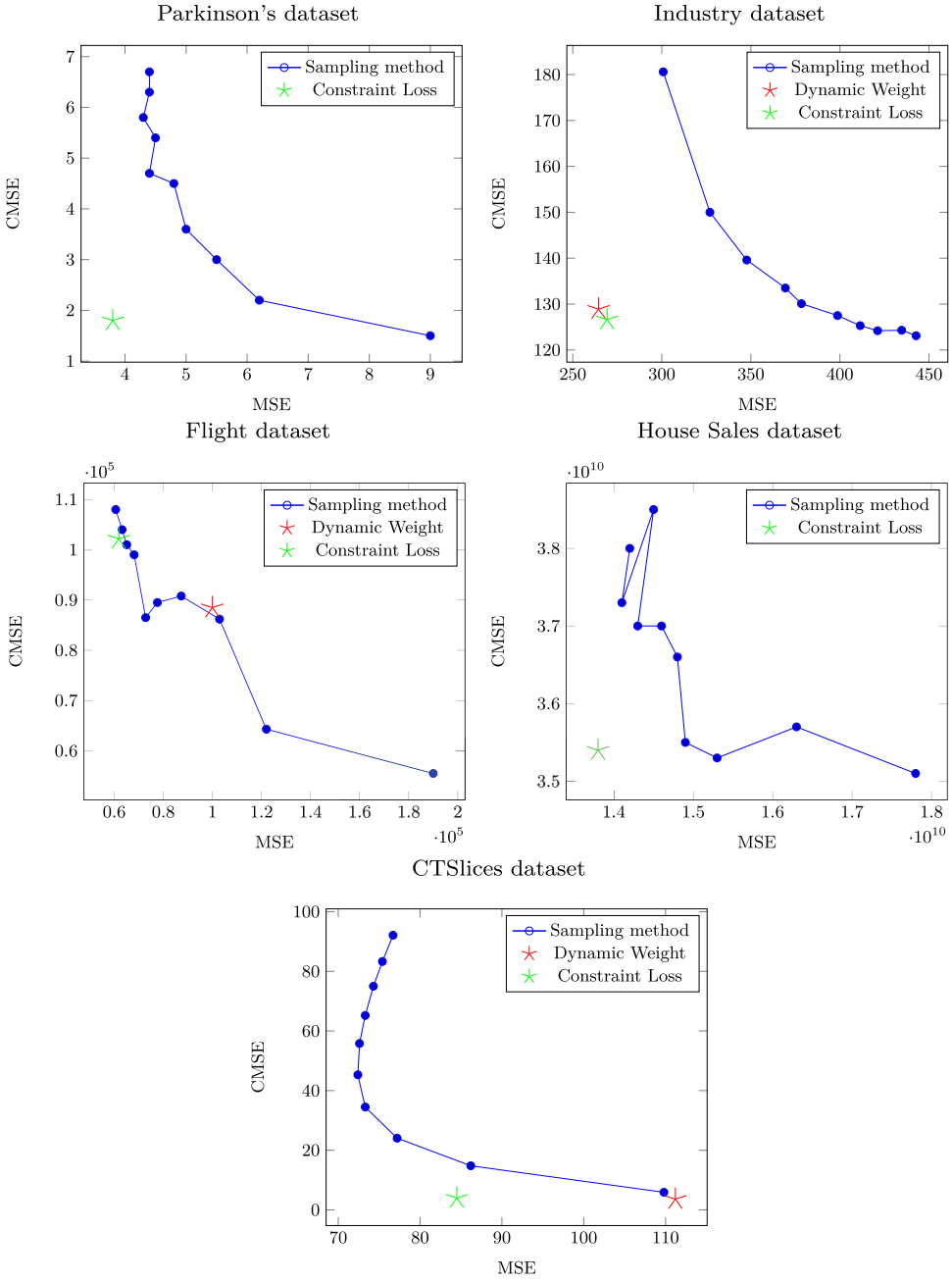

Fig. 3. Sampling method compared with the Dynamic Weight and the Constraint Loss algorithms. Blue points represent different sampling percentage of unconstrained instances. In all graphs, left most blue point represents the smallest sampling percentage (i.e., 10% of the unconstrained instances).

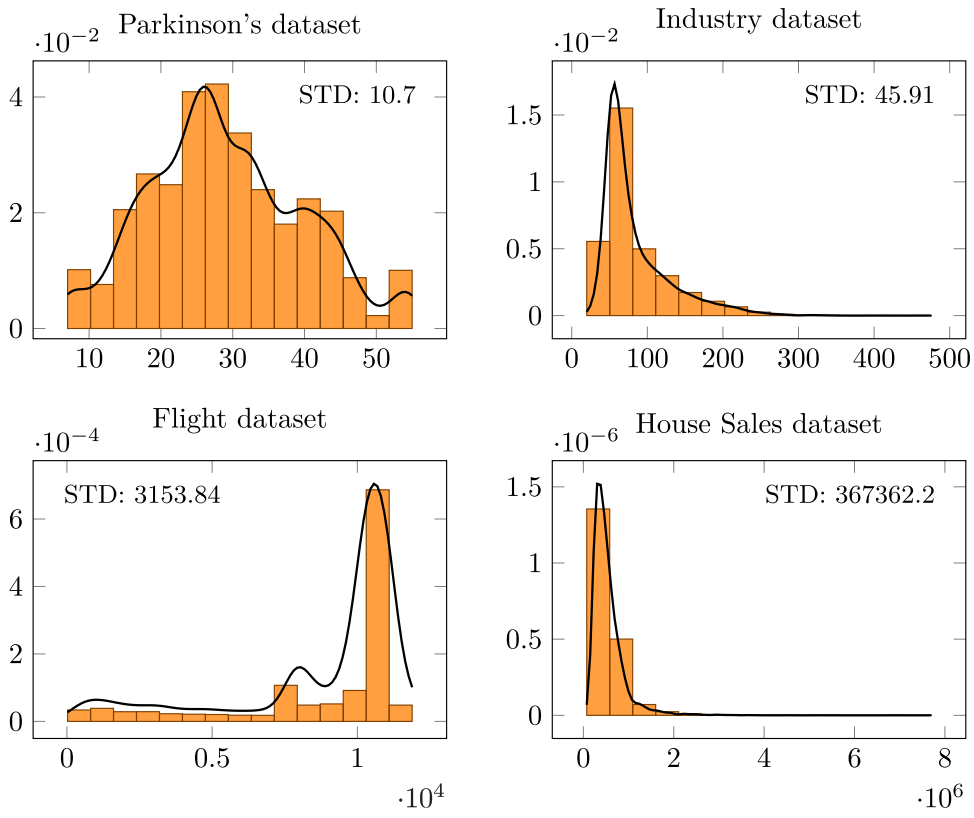

Fig. 4. Target attribute distribution of all datasets. The *y*-axis represents density, and the standard deviation of each attribute is also provided. The Constraint Loss algorithm performs better and manages to converge in cases when the target attribute distribution is not too long tailed and the standard deviation is relatively low.

algorithm yields the most impressive trade off between constraint measures and the MSE measure, with an MSE measure minimally affected in both configurations.

7.7. Comparing the proposed method with a sampling approach

Another method that can be considered as a possible solution for the constraint regression problem is based on ideas coming from the *GOSS* method implemented in the lightGMB algorithm (see Section 3). Such sampling approach implies that instead of taking the whole training data as input, we select a data subset. Since the constrained instances are the most important, all of them are selected. The other unconstrained instances are randomly under-sampled. The percentage of unconstrained instances which are sampled is configurable. Fig. 3 shows a comparison of the sampling method versus the two algorithms suggested in the paper. Ten different under-sampling percentages were tested, with an equal 10% sampling change in the [0%, 100%] interval.

The trade-off comparison is based on the MSE measure versus the CMSE measure. The blue points in the graph represent the sampling approach, where each point is a result of a different sampling percentage value. These results are compared with the Constraint Loss algorithm as well as with the Dynamic Weight algorithm. As before, the results shown are based on a 5-fold cross validation process. As can be seen, over all datasets (besides the Flight dataset) at least one of the algorithms suggested in this paper outperforms the sampling approach and suggest a better trade-off between the MSE and the CMSE measures. Note that in some cases the Dynamic Weight or the Constraint Loss algorithm result is not shown in the graph since it reached a high MSE value beyond the graph scale.

8. Analysis and summary of results

In most of the datasets and configurations examined, one of the two suggested algorithms always outperforms the three baselines. One exception to this statement is when focusing on the CMSE measure in the Flight dataset (note that the standard deviation of this measure is very high with this dataset). The Dynamic Weight algorithm performs well in most situations and demonstrates improved performance compared to the three baselines, in terms of constraint measures. As we have seen in our experiments, this algorithm can lead to very good constraint measures, but this always involves a clear trade-off with other standard regression measures (e.g., MSE). This is in contrast to the Constraint Loss algorithm, which clearly improves constraint measures, but only up to a certain point. Even when trying to increase the C_γ parameter (i.e., during the constraint hyperparameter tuning process, as explained in Section 7.1), constraint measures were optimized up to a certain point which usually was higher compared to the Dynamic Weight algorithm. The Dynamic Weight algorithm might also suffer from a high standard deviation in all of the measures. It still outperforms the Constant Weight baseline and demonstrates a better trade-off between the MSE and constraint related measures in all cases. We analyzed the Constraint Loss algorithm further, in order to identify cases in which the algorithm is expected to perform very well. We found that the Constraint Loss algorithm achieves impressive results in situations in which the basic algorithm performs well (i.e., cases when the Standard GBT algorithm performs well with regard to the MSE measure). Another phenomenon we observed regarding the Constraint Loss algorithm is the fact that it achieves impressive results in cases in which the target attribute (i.e., predicted *y*) doesn't have a very high standard deviation and achieves even

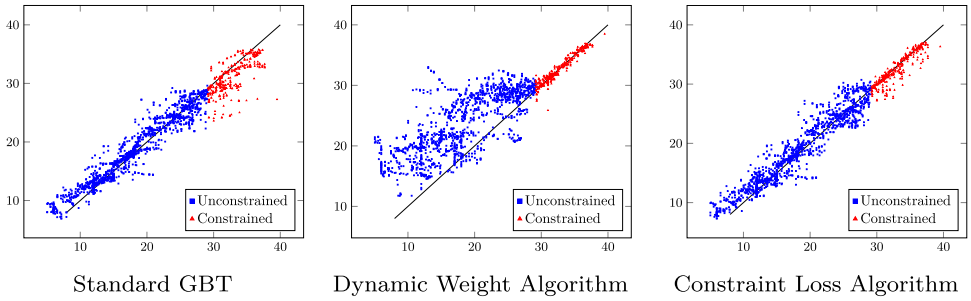

Fig. 5. True values (x-axis) and predicted values (y-axis) for some of the algorithms. Ideally, data points would have been laid on the 45° imaginary line. The data points are based on the Parkinson's Dataset results, configuration A. As can be seen, the Constraint Loss algorithm achieves the best performance in this setting.

better results when the attribute is well distributed (i.e., close to uniform or normal distribution), as can be concluded from the graphs presented in Fig. 4 along with the results discussed in Section 7.

Another interesting analysis can be seen in Fig. 5 which emphasizes the differences between some of the algorithms and the trade-off between constraint measures and regular measures. As shown, the two suggested algorithms predict the constrained instances well, but the Dynamic Weight algorithm significantly over-predicts the other instances while the Constraint Loss algorithm does not.

9. Discussion

Based on our experiment results we can conclude that the proposed algorithms is able to model the constraints and ensure that they are satisfied. This indicates a very important property of the new algorithms – the ability to induce a model that can provide a high level of assurance in the prediction of important/critical cases. The new GBT-based algorithms presented provide the basis for the implementation of additional new regression algorithms (e.g., based on deep learning) that can be further adapted to specific problems and use-cases. While according to the proposed method, the constraints are defined for each instance individually, such constraints can eventually be derived from a more general and broad definition of the constraint on a subpopulation of the instances. Finally, the proposed method allows adjusting the trade-off between the general regression error and constraint satisfaction error through the use of a single hyperparameter. The main strength and the implications of this research is in introducing a new class of regression problems in the form of instance level constraints, which also allows evaluating prediction models according to a new measure (i.e., constraint satisfaction). The induced model is able to provide an accurate prediction for the constrained cases with a high level of assurance. In our evaluation we also identified the following limitations of the approach. First, in some cases, a trade-off between general regression error and the constraint satisfaction error may exist. Furthermore, we expect that in cases where the constraints are defined for a large sub-population it will be difficult for the algorithm to induce a model that satisfies both contradicting requirements (a low regression error while complying with the constraints). In such a case, the resulting model will mainly depend on the new hyper parameter (i.e., C_η or C_γ). Second, although the adjustment of the trade-off between the general regression error and constraint satisfaction error is made through a single new hyperparameter, it still needs to be determined and optimized for each use case. Finally, the constraints need to be defined for the relevant instances. In some use cases, those constraints can be automatically derived or computed for a large set of instances through a more general rule or constraint (e.g., assign constraint c to all blood pressure samples that are approximately value x).

10. Conclusions and future work

In this paper, we presented a constraint learning framework and applied it to regression problems. We presented two algorithms, both of which wrap the GBT algorithm and leverage the GBT's unique characteristics in order to satisfy the given constraints. Based on the experiments conducted, it is clear that the suggested algorithms provide added value in terms of satisfying the constraints. Directing the algorithm so that it satisfies constraints might negatively affect other measures (e.g., MSE), as was clearly seen in the Dynamic Weight algorithm with all of datasets. The trade-off between the general regression measures (such as the MSE) and newly proposed constraint measures is different for the various datasets and constraint settings. Therefore, balancing these two contradicting measures and finding the optimal setup according to the requirements of the given use case can be controlled by adjusting the constraints' hyperparameters.

Future research possibilities include the following:

- *Hybrid approach* – The two new algorithms were tested independently. In future work, a hybrid approach based on combining the two algorithms might lead to better results, as has been observed in many other machine learning algorithms.
- *Semi-supervised learning* - We only applied the new methods to fully supervised problems. In future work, the given constraints can be converted and used as our new target. In such problems the new target attribute will become an interval of values, and the algorithm's aim would be to predict the continuous target attribute as close to each instance interval as possible.
- *Different constraint settings* – Currently, the constraints used relate to the target attribute and are instance dependent. In future work, another useful constraint setting can be studied which involves constraining pairs of instances (e.g., the predicted value of instance x must be higher than instance y's predicted value). This is a much different setting, than the setting investigated in the current study since: (a) the problem becomes instance dependent, and (b) the constraint's satisfaction goal differs from the typical regression goal.
- *Different algorithm usage* – In the scope of the current research, we only used GBT as a prediction algorithm, and we wrapped it in such a way that takes constraints into account. In future work, we can use and wrap neural network algorithms which would work well for the type of modifications we performed, since neural network algorithms are iterative and gradients calculated each iteration.

Appendix A. Related Work Summary Table

Table A.7

Table A.7
Related Work Summary.

Paper	Description	Goal	ML task	Constraints Defined	Method Used	Comparison to our work
Tibshirani (1996), Hoerl and Kennard (1970), Zhou et al. (2011)	Constraint on the learning algorithm parameters	Reducing hypothesis search space (and thus learning time) and avoid overfitting	Constraints Regression	Constraining the learning algorithm parameters	Ridge, lasso elastic net regression	*Different goal:* efficient training process *Different constraint type:* constraints are not related to the instances' target feature but rather to the hyperparameters of the algorithm
Chang et al. (2008), Yu et al. (2007)	Learning with prior knowledge	Reducing hypothesis search space (and thus learning time) by adding information to the problems setting in the form of prior knowledge	Semi/fully supervised classification	No constraints are defined	A framework for various algorithms is suggested	*Different goal:* efficient training process *Different constraint type:* prior knowledge about the domain is used but not in the form of constraint *Different ML task:* focuses on classification (and not on regression problems)
Klastorin and Watts (1981), DeSarbo and Mahajan (1984)	Associated with the classes themselves (e.g., the size of a specific class must be below 100)	Providing a classification solution which takes constraints related to the classes suggested by the solution into account via clustering techniques	Classification	Size of classes and relations between two classes sizes are mainly the constraints being used	CONCLUS (Constrained Clustering)	*Different constraint type:* constraints are not instance level, but rather related to the solution suggested by the algorithm (e.g., size of each class and not necessarily the prediction for each instance) *Different ML task:* focuses on classification (and not on regression problems)
Nguyen (2010), Zhang and Yan (2007), Nguyen and Caruana (2008)	Input dataset contains partly supervised information that indicates whether (some) pairs of instances are associated with the same class or not	Solving a classification problem is in some cases an indication whether pairs of instances are associated with the same class or not is provided; this can be taken into account during the learning process	Semi supervised classification	Pairs of instances must/must not be classified to the same class	PCSVM (Pairwise Constraints SVM)	*Different constraint type:* constraints are related to pairs of instances (and not to individual instances) *Different ML task:* focuses on classification (and not on regression problems)
Zhao (2008), Ting (2002), Zadrozny, Langford, and Abe (2003)	Cost sensitive learning	Inducing a cost sensitive model (prior to modeling or during model learning) by assigning appropriate weights to different training instances	Classification	No constraints are defined; cost function is being used	Assigning weights to instances (or duplicating instances) and then using different types of classification models	*Different goal:* cost sensitive learning *Different constraint type:* a cost function in the form of instance level weights is used (and not a specific constraint on the target attribute) *Different ML task:* focuses on classification (and not on regression problems)

(*continued on next page*)

Table A.7 (*continued*)

Paper	Description	Goal	ML task	Constraints Defined	Method Used	Comparison to our work
Zhao et al. (2011), Koenker (2005)	Cost sensitive learning	Inducing a cost sensitive model by defining a cost function which takes into account asymmetric cost in cases of over/under prediction	Regression	No constraints are defined; cost function is being used	Apply cost function to regression model; least squares error model with cost-sensitive aspects is introduced	*Different goal:* cost sensitive learning *Different constraint type:* an asymmetric cost function which takes into account the over/under prediction; the cost function is generic and applied to all samples in the learning dataset. The method restrict the cost function to be convex (as opposed to our constraint type which can be defined to each sample individually)
Coons (1978)	Focusing on reliable instances (hard-points)	Creating a more reliable and accurate model	Regression	Constraints are applied on specific reliable instances, so the regression will perfectly predict those	New implementation of a regression problem based on closed algebraic solution	*Different goal:* creating a more reliable model using hard-points) *Different constraint type:* constraints, as we define, are not necessarily hard-constraints *Different method:* implemented based on a closed algebraic solution,while we use arithmetic-based solution
Harrell (2015)	Survival analysis related to regression problems	Inducing a regression model where the values of the target attribute of some of the instances indicate the upper/lower bound of the real value (due to the fact that the real value is unknown/determined at a future time)	Regression	Upper or lower bound on the true value of the target attribute is defined	Survival analysis techniques	*Different goal:* learning when partial information about the true values of the target attribute is available; in our case, the true value is fully known *Different constraint type:* only an upper or lower bound is available and reflects the actual true value of the target attribute(and not a desired value)

Credit authorship contribution statement

Abraham Israeli: Conceptualization, Funding acquisition, Formal analysis, Writing - original draft. **Lior Rokach:** Conceptualization, Formal analysis, Writing - original draft. **Asaf Shabtai:** Conceptualization, Formal analysis, Writing - original draft.

References

Atiya, A. F. (2001). Bankruptcy prediction for credit risk using neural networks: A survey and new results. *IEEE Transactions on Neural Networks, 12*(4), 929–935.

Carroll, R. J. (2017). *Transformation and weighting in regression*. Routledge.

Chang, M., Ratinov, L., & Roth, D. (2008). Constraints as prior knowledge. In *Icml workshop on prior knowledge for text and language processing* (pp. 32–39).

Chen, T., & Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (pp. 785–794). ACM.

Coons, S. A. (1978). Constrained least-squares. *Computers & Graphics, 3*(1), 43–47.

DeSarbo, W. S., & Mahajan, V. (1984). Constrained classification: the use of a priori information in cluster analysis. *Psychometrika, 49*(2), 187–215.

Freund, Y., & Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences, 55*(1), 119–139.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis, 38*(4), 367–378.

Harrell, F. (2015). *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*. Springer.

Hoerl, A. E., & Kennard, R. W. (1970). Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics, 12*(1), 55–67.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., et al. (2017). Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems* (pp. 3146–3154).

Klastorin, T., & Watts, C. A. (1981). The determination of alternative hospital classifications. *Health services research, 16*(2), 205.

Koenker, R. (2005). *Quantile regression*. Cambridge university press. 38

Li, L., Liang, C.-J. M., Liu, J., Nath, S., Terzis, A., & Faloutsos, C. (2011). Thermocast: a cyber-physical forecasting model for datacenters. In *Proceedings of the 17th acm sigkdd international conference on knowledge discovery and data mining* (pp. 1370–1378). ACM.

Narula, S. C., & Wellington, J. F. (1982). The minimum sum of absolute errors regression: A state of the art survey. *International Statistical Review/Revue Internationale de Statistique*, 317–326.

Nassi, B., Rokach, L., & Elovici, Y. Virtual breathalyzer. (2016). arXiv:1612.05083.

Nguyen, N., & Caruana, R. (2008). Improving classification with pairwise constraints: a margin-based approach. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 113–124). Springer.

Nguyen, N. H. (2010). *Semi-supervised learning with partially labeled examples* Ph.D. thesis. Cornell University.

Prokhorenkova, L., Gusev, G., Vorobev, A., Dorogush, A. V., & Gulin, A. (2018). Catboost: unbiased boosting with categorical features. In *Advances in neural information processing systems* (pp. 6639–6649).

Sami Sivri, M., & Oztaysi, B. (2018). Data analytics in manufacturing. In *Industry 4.0: Managing The Digital Transformation* (pp. 155–172)). Cham: Springer International Publishing. doi:10.1007/978-3-319-57870-5_9.

Schapire, R. E. (1990). The strength of weak learnability. *Machine learning, 5*(2), 197–227.

Sideris, C., Kalantarian, H., Nemati, E., & Sarrafzadeh, M. (2016). Building continuous arterial blood pressure prediction models using recurrent networks. In *Smart computing (smartcomp), 2016 ieee international conference on* (pp. 1–5). IEEE.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 267–288.

Ting, K. M. (2002). An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering, 14*(3), 659–665.

Weisberg, S. (2005). *Applied linear regression*: 528. John Wiley & Sons.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural computation, 8*(7), 1341–1390.

Yu, T., Jan, T., Simoff, S., & Debenham, J. (2007). Incorporating prior domain knowledge into inductive machine learning. *Unpublished doctoral dissertation Computer Sciences*.

Zadrozny, B., Langford, J., & Abe, N. (2003). Cost-sensitive learning by cost-proportionate example weighting. In *Data mining, 2003. icdm 2003. third ieee international conference on* (pp. 435–442). IEEE.

Zhang, J., & Yan, R. (2007). On the value of pairwise constraints in classification and consistency. In *Proceedings of the 24th international conference on machine learning* (pp. 1111–1118). ACM.

Zhao, H. (2008). Instance weighting versus threshold adjusting for cost-sensitive classification. *Knowledge and Information Systems, 15*(3), 321–334.

Zhao, H., Sinha, A. P., & Bansal, G. (2011). An extended tuning method for cost-sensitive regression and forecasting. *Decision Support Systems, 51*(3), 372–383.

Zhou, T., Tao, D., & Wu, X. (2011). Manifold elastic net: a unified framework for sparse dimension reduction. *Data Mining and Knowledge Discovery, 22*(3), 340–371.