Exceptional Model Mining with Tree-Constrained Gradient Ascent

Thomas E. Krak

Ad Feelders

Technical Report UU-CS-2015-002
January 2015

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Exceptional Model Mining with Tree-Constrained Gradient Ascent

Thomas E. Krak
Universiteit Utrecht
Utrecht, The Netherlands
e-mail: T.E.Krak@uu.nl

Ad Feelders
Universiteit Utrecht
Utrecht, The Netherlands
e-mail: A.J.Feelders@uu.nl

Abstract

Exceptional Model Mining (EMM) generalizes the well-known data mining task of Subgroup Discovery (SD). Given a model class of interest, the goal of EMM is to find subgroups of the data for which a model fitted to the subgroup deviates substantially from the global model. In both SD and EMM, heuristic search is often employed to circumvent the problem that exhaustive search of the subgroup description space is in general not feasible.

We present a novel heuristic search strategy for EMM called tree-constrained gradient ascent (TCGA). It is designed specifically to exploit information about the influence of individual records on the quality of a subgroup, and guarantee at the same time that the subgroups can be described in the pattern language. By generalizing the notion of subgroups to that of *soft* subgroups, numerical optimization can be applied to find subgroups of high quality. We introduce a form of constrained gradient ascent that constructs the constraint in parallel with the numerical optimization so as to guarantee that the subgroups can be described in the pattern language, while restricting the numerical optimization as little as possible.

We show how TCGA can be applied to EMM with linear regression models and to traditional SD. The proposed algorithm is experimentally evaluated on both synthetic and real world data sets. We compare the results to those obtained with beam search, a heuristic search algorithm commonly employed in SD and EMM.

1 Introduction

Exceptional Model Mining (EMM) [16] is a form of exploratory data mining which generalizes Subgroup Discovery (SD) [14]. In EMM we seek to find descriptions of subgroups within a data set, such that a model induced on such a subgroup deviates substantially from the same model induced on the entire data set.

Because it is in general computationally intractable to exhaustively search the space of subgroup descriptions for interesting subgroups, heuristics are often employed instead. For example, a beam search strategy may be used [16, 9].

It has been shown [15] however, that more specialized strategies exploiting information about the influence of individual records on subgroup quality may lead to improved results. The EMDM algorithm proposed by van Leeuwen [15] exploits such information by alternately maximizing quality in the subgroup extension space, and finding descriptions of the resulting subgroup extension. However, generalizing this algorithm to different model classes, such as linear regression, is non-trivial. Furthermore, optimization in the subgroup extension space can produce subgroups that are impossible to describe in the pattern language.

We propose a novel algorithm called tree-constrained gradient ascent (TCGA) that exploits information on the contribution of individual records to subgroup quality and at the same time guarantees that the subgroup extension can be described concisely in the pattern language. Briefly, the algorithm works by generalizing the notion of a subgroup to that of a *soft* subgroup. Hereby the quality measure is made differentiable, and the quality of a subgroup extension is then optimized numerically using

a form of constrained gradient ascent. The constraint is constructed simultaneously with the optimization in order to ensure that subgroups can be described in the pattern language, and at the same time hinder the numerical optimization as little as possible. Subsequently we apply a post-processing step to further simplify and generalize the patterns found by the algorithm.

This paper is organized as follows. In Section 2 we give a brief description of EMM and introduce some notation. In Section 3, the TCGA algorithm is described and motivated; we also discuss the post-processing of patterns found with TCGA. In Section 4 we show how EMM with linear regression models can be performed with the TCGA algorithm. Experimental results are discussed in Section 5, and we finally close with conclusions in Section 6.

2 Exceptional Model Mining

In Exceptional Model Mining (EMM) [16], we consider a data set $\mathcal{D}$ as a bag containing n records r_i, $i = 1, \ldots, n$, of the form

$$r_i \equiv \langle a_1^i, \ldots, a_k^i, x_1^i, \ldots, x_p^i \rangle. \tag{1}$$

Here, we refer to the various a_j^i as the i-th record's *attributes*, and to the x_j^i as that record's *targets*. The attributes are used to define subgroups of the data, and the models are defined on the targets. Let the complete sets of the i-th record's attributes and targets be denoted as a^i and x^i, respectively. We write the domain of attributes as $\mathcal{A}$, and that of the targets as $\mathcal{X}$. The attributes are taken to be some combination of real and categorical attributes, whereas the choice of $\mathcal{X}$ depends more strongly on the model class under consideration.

Let a *pattern* be a function $P : \mathcal{A} \to \{0, 1\}$ that describes a *subgroup* $G_P \subseteq \mathcal{D}$,

$$G_P \equiv \left\{ r_i \in \mathcal{D} \,\middle|\, P\left(a^i\right) = 1 \right\}. \tag{2}$$

Furthermore, the set of all patterns is called the *pattern language*, which is denoted $\mathcal{P}$. Here, we stipulate the pattern language to be the set of all disjunctions of conjunctions of logical conditions on single attributes. In the sequel, we will often drop the subscript P when referring to a subgroup G, in cases where no confusion should arise.

The models fitted to subgroup G and the entire data set $\mathcal{D}$ are denoted by M_G and $M_\mathcal{D}$ respectively. A quality measure $\varphi_\mathcal{D}(\cdot)$ is defined as a function $\varphi_\mathcal{D} : \mathcal{P} \to \mathbb{R}$ that quantifies the exceptionality of M_G with respect to $M_\mathcal{D}$, i.e., $\varphi_\mathcal{D}(\cdot)$ essentially computes some distance function between these models. Examples of model classes and quality measures can be found in [16, 9].

The goal of EMM may now be described as finding a set of patterns that are given a high score by the measure $\varphi_\mathcal{D}(\cdot)$.

3 Tree-Constrained Gradient Ascent for EMM

3.1 Subgroup Optimization

First we consider how to find high quality subgroups in the subgroup extension space, that is, the set of all possible subsets of $\mathcal{D}$. This allows us to learn important information about the influence of individual records on the quality of a subgroup. We rewrite the quality measure $\varphi_\mathcal{D}(\cdot)$ as an objective function $O(\mathbf{w})$, where $\mathbf{w}$ is a length n vector of inclusion indicators $\{0, 1\}$. Thus, we say that the i-th record is included in the subgroup if $w_i = 1$, and excluded from the subgroup if $w_i = 0$. Finding a high quality subgroup extension can then be expressed as finding $\mathbf{w}^*$ such that

$$\mathbf{w}^* = \underset{\mathbf{w} \in \{0,1\}^n}{\arg\max} \{O(\mathbf{w})\} \ .$$

Because exact solution of this problem is in general computationally intractable, we will have to resort to heuristic approximations instead.

Rather than solve this problem using a discrete local-search algorithm, we make use of numerical optimization, in particular, gradient ascent. To allow for this, the quality measure should be differentiable with respect to the inclusion of records in the subgroup. To this end, we generalize the notion of a subgroup to that of a *soft* subgroup by introducing an *inclusion weight* $w_i \in [0, 1]$ for each record $r_i \in \mathcal{D}$. Letting $\mathbf{w}$ now denote the vector of these inclusion weights, $O(\mathbf{w})$ can often be made differentiable by making appropriate changes to the estimation of M_G using a weighted-data scheme. The specifics of such a scheme depend on the model class; in section 4 we show

how this can be done for the linear regression model class. Once we have obtained a differentiable objective function $O(\mathbf{w})$, a locally optimal solution can be found using gradient ascent. The weights at this optimum can then be rounded to obtain a crisp subgroup extension.

By looking at the soft subgroup extension space, we are able to learn an important property of individual records. In particular, consider

$$\text{Sign}\left\{\frac{\partial O(\mathbf{w})}{\partial w_i}\right\}. \tag{3}$$

If this quantity is positive, it indicates that, all other weights being equal, increasing the weight of the i-th record will improve the quality of the subgroup. A similar observation holds for the case where this quantity is negative. We will make use of these observations in the development of our algorithm in the next section.

One way to now obtain a subgroup description from its extension is to induce a rule-based classifier to find a pattern *describing* the subgroup extension found with gradient ascent. That is, the rules induced by such a classifier would constitute the pattern describing the subgroup. This approach is taken by van Leeuwen [15].

The fundamental problem with this approach is that we cannot be sure that the subgroup extension can be described concisely, or even at all, in the pattern language using the records' attributes in the space $\mathcal{A}$. We may obtain an extremely complex subgroup description, or a very poor fit by the classifier. Such a poor fit could then result in a pattern that does not properly capture the subgroup extension, resulting in a much lower quality pattern compared to the quality of the optimized subgroup. Experiments that we performed with this approach using CART [3] as the classifier confirmed that this problem indeed occurs in practice.

This leads to the main contribution of this paper. On the one hand, we would like to exploit information on the contribution of individual records to subgroup quality as is realized by optimization in the subgroup extension space. On the other hand, we would like to guarantee that the subgroup extension can be described concisely in the pattern language. Therefore, we propose to constrain the numerical optimization in such a way that concise description in the pattern language is guaranteed. To this end, the algorithm that we introduce in the next section optimizes the quality of the subgroup, and simultaneously constructs the constraint to guarantee concise description. The latter is done in such a way so as to hinder the 'ideal', viz. unconstrained, optimization as little as possible.

3.2 Tree-Constrained Gradient Ascent

We first consider the requirement that the subgroups can be described concisely in the pattern language. Suppose we have a partition of $\mathcal{D}$ based on the attributes of the records. Suppose furthermore that this partition can be expressed using the pattern language $\mathcal{P}$: each block in the partition is expressible using a rule, which is a conjunction of single-attribute conditions on the attributes, and these rules are mutually exclusive and globally exhaustive.

Now, say that we assign to each block a binary indicator $\{0, 1\}$ which denotes whether the records covered by it are included in the subgroup or not. Clearly, the resulting subgroup can then be expressed by taking the pattern P to be the disjunction of all rules describing blocks with a positive inclusion indicator.

To perform subgroup optimization under this scheme, we have two related problems to solve: how to find a good partition, and which inclusion indicator to assign to each block. Focusing first on how to specify the subgroup inclusion for each block, we can use the same approach here as we did before when optimizing in the subgroup extension space: assign an inclusion *weight* to each block, and perform numerical optimization on these weights. Thus, all records within the same block are assigned the same inclusion weight, and by performing numerical optimization on the set of all these shared weights, a local optimum in the subgroup extension space is found. Since the records within a given block all share the same weight, the resulting subgroups can be described in the pattern language.

To find a partition of the data, we note that the mutually exclusive and globally exhaustive structure that we require of the partition may be obtained by recursively partitioning the space $\mathcal{A}$. We propose to produce this partition by constructing a classification tree [3] on this space. The leafs of such a tree then correspond to the individual blocks.

We aim to construct the tree in such a way so as to hinder the optimization as little as possible. For this reason

we define the positive class to be the points with a positive gradient and the negative class to be the points with a negative gradient. As is usual in classification trees, we choose the split that separates the positive from the negative class as well as possible. Intuitively this makes sense because for the points with positive gradient we would like to increase their weights, and for points with a negative gradient we would like to decrease their weights. Since the weights are constrained to be equal on the points that fall into the same leaf, we have to perform the same weight update on those points, and hence we should separate the points whose weight we would like to increase from those whose weight we would like to decrease.

Note, however, that the sign of the weights' gradient depends on the current position of these weights in the subgroup extension space. Furthermore, because the second-order mixed partial derivative of these weights will in general be non-zero, the signs may actually change as the optimization progresses. We therefore propose to interlace growing of the tree with the gradient ascent updates: we alternately split the tree's leafs and perform gradient ascent updates on the leafs' weights.

More formally, let V_j denote a leaf in the classification tree, and let R_{V_j} denote the set of indices of records that are assigned to V_j. Let $v^j \in \mathbb{R}$ denote the *output value* of V_j, and let $\sigma : \mathbb{R} \to [0, 1]$ denote the logistic sigmoid, $\sigma(v) \equiv (1 + e^{-v})^{-1}$. We use the logistic sigmoid for convenience so that we may perform the numerical optimization in $\mathbb{R}$ rather than $[0, 1]$. Thus, gradient ascent is performed on v^j, and the weights of all records assigned to V_j are given by

$$\forall_{i \in R_{V_j}} : w_i = \sigma(v^j). \tag{4}$$

The (soft) subgroup extension may thus be optimized numerically by performing gradient ascent on the leafs' outputs. Taking partial derivatives of the objective function $O(\mathbf{w})$ with respect to a leaf's output v^j, and making use of the definition of the logistic sigmoid, we have

$$\begin{aligned} \frac{\partial O(\mathbf{w})}{\partial v^j} &= \sum_{i \in R_{V_j}} \frac{\partial O(\mathbf{w})}{\partial w_i} \frac{\partial w_i}{\partial v^j} \\ &= \sigma(v^j)(1 - \sigma(v^j)) \cdot \sum_{i \in R_{V_j}} \frac{\partial O(\mathbf{w})}{\partial w_i}. \end{aligned} \tag{5}$$

Gradient ascent should then be performed on the output values of all leafs simultaneously. We emphasize that the objective function does *not* decompose over the records in the data set, nor over the leafs in the tree, because the second-order mixed partial derivative of the objective function with respect to the output values of two leafs will, in general, be non-zero.

The pseudo-code of the full algorithm, which we refer to as Tree-Constrained Gradient Ascent (TCGA), is given by Algorithm 1. We here first describe a single iteration of the algorithm, which starts at Line 3. First, partial derivatives are computed for all individual records (Line 4), using the tree's current output as the current position in the subgroup extension space $[0, 1]^n$.

The algorithm then tries to split all non-pure leafs in the tree so as to minimize the impurity (Line 8), where impurity is defined using the Gini-index [3], with the signs of the individual records' derivatives constituting the class labels. To prevent over-fitting of the tree, we use the common parameters S_{min} and L_{min}, which specify the minimum number of records that a leaf must contain or assign to each resulting child, respectively, for a split to be considered valid. Whenever a leaf is split in this way, the output values of the resulting children are initialized to the current output of their parent. This ensures differentiability of the objective function by the fact that splitting, by itself, will not change any of the tree's outputs. Interlacing of tree-growing and gradient ascent updating is realized by ensuring that the resulting children are not immediately split as well. That is, leafs are split at most once at each iteration of the algorithm, rather than continuing this process recursively.

After all leafs are considered in this fashion, a single gradient ascent update step is performed on the outputs of all leafs (Line 9), as described above. The gradient that is used in the update step is given by Equation 5. New inclusion weights are then computed for all records (Line 10, *cf.* Equation 4). This process iterates until some stopping criterion is satisfied, which we have implemented by testing for a vanishing gradient (Lines 5-7). We also stop after a maximum number of iterations have elapsed (Line 3), because we cannot in general guarantee convergence for gradient ascent with a fixed step-size.

To find *multiple* high quality subgroups, we make use of random restarts. We set the vector $\mathbf{w}$ at the algorithm's first iteration to a random position in $[0, 1]^n$ (Line 2). Computing derivatives from different such points will

Algorithm 1 TREE-CONSTRAINED GRADIENT ASCENT

Input: A dataset $\mathcal{D}$, step-size η, maximum iterations τ_{max}, minimum gradient ∇_{min}, and minimum split supports S_{min} and L_{min}.
Output: A classification tree T.

$T \leftarrow \{V_{\text{root}}\}$
$\mathbf{w}_{(0)} \leftarrow$ GetRandomStartPos()
for $\tau = 1 \rightarrow \tau_{\text{max}}$ **do**
  $\nabla_{\mathbf{w}_{(\tau-1)}} \leftarrow$ GetGradient($\mathcal{D}, \mathbf{w}_{(\tau-1)}$)
  if $\left(\|\nabla_{\mathbf{w}_{(\tau-1)}}\| \leq \|\nabla_{\text{min}}\|\right)$ **then**
    return T
  end if
  $T \leftarrow$ SplitCurrentLeafs($T, \mathcal{D}, \nabla_{\mathbf{w}_{(\tau-1)}}, S_{\text{min}}, L_{\text{min}}$)
  $T \leftarrow$ UpdateOutputValues($T, \eta, \nabla_{\mathbf{w}_{(\tau-1)}}$)
  $\mathbf{w}_{(\tau)} \leftarrow$ GetOutput($T, \mathcal{D}$)
end for
return T

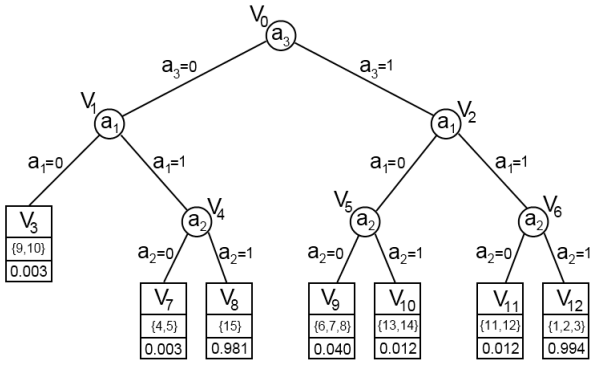


Figure 1: Tree grown on an example data set with 15 records. The leafs are shown as partitioned rectangles, showing from top to bottom the node's identifier V_j, the set of indices R_{V_j} of the records assigned to this leaf, and the inclusion weights $\sigma(v_j)$ of the corresponding records.

then result in different split-decisions and gradient ascent updates on the first iteration of tree construction. Subsequently running the algorithm as described will then enable it to find different local optima for different initial 'output' positions.

After convergence, the sigmoid-transformed outputs of all leafs are rounded to the set $\{0, 1\}$, yielding again a 'crisp' subgroup extension that is meaningful in the EMM context. The rule-based pattern describing this subgroup may then simply be read from the tree.

Observe that this pattern in general consists of a disjunction of conjunctions of logical conditions over single attributes in the space $\mathcal{A}$. Each conjunction is taken over the split-conditions encountered on the path from the tree's root to a leaf with (rounded) output of 1, specifying inclusion in the subgroup. The disjunction of these conjunctions is then taken over all such leafs. In the sequel, we will refer to such conjunctions as *rules*, and to their disjunction as the *rule set*.

To illustrate, Figure 1 shows an example result of the tree-constrained gradient ascent algorithm. This tree is converted into a disjunctive set of crisp inclusion rules. To do this, we consider the leafs V_j for which $\sigma(v^j) \geq \frac{1}{2}$, that is, the leafs with inclusion weights that are not lower than $\frac{1}{2}$. From Figure 1, we see that the leafs satisfying this condition are V_8 and V_{12}. The corresponding pattern is:

$$(a_3 = 0 \wedge a_1 = 1 \wedge a_2 = 1) \vee (a_3 = 1 \wedge a_1 = 1 \wedge a_2 = 1)$$

It is easy to see that this can be simplified to the equivalent description $(a_1 = 1 \ \wedge \ a_2 = 1)$. The post-processing procedure discussed in the next section will, among other things, automatically remove such redundancies from the subgroup description.

3.3 Post-Processing

After the tree-constrained gradient ascent algorithm has converged, and the corresponding pattern extracted as described in Section 3.2, this pattern is subsequently post-processed. Because this post-processing procedure is formulated in terms of the rule-based pattern, rather than in terms of the tree structure, it is in fact quite general and may also be applied to the results of other algorithms such as beam search.

The entire post-processing procedure is carried out on a held-out part of the data. We set aside this data by randomly sub-sampling the dataset into mining and post-processing data at each random restart of the TCGA algorithm. The procedure itself is most conveniently described as a three step process. Briefly, the first step seeks to maximize quality by removing conditions and rules, the

second step serves to remove superfluous conditions without 'significantly' changing the subgroup's model, and the final step attempts to verify whether the resulting pattern is indeed 'interesting'. We detail these steps below.

3.3.1 Quality Maximization

The first step begins by evaluating the quality of all patterns resulting from removing entire rules from the original rule set, one at a time. Whenever the highest quality such pattern has a quality not lower than that of the original, that pattern is selected to replace the original, and the process is repeated until no more rules can be removed in this fashion. Subsequently, the algorithm seeks to maximize the quality by removing single conditions from the remaining rules. This is done by first evaluating the quality of all patterns resulting from removing single conditions, and sorting the conditions in order of decreasing quality-after-removal. Conditions are then considered in order, and removed from the remaining rules whenever this can be done without decreasing the pattern's quality. Whenever a condition is removed in this way, the quality resulting from removing any of the remaining conditions will change. Therefore, the removal-quality is re-evaluated whenever the next condition is considered for removal. This part of the post-processing is quite similar to a post-processing step performed in PRIM [10], but there the removal of conditions is performed by the user in an interactive fashion. The first step of the post-processing is completed when all conditions have been considered in this greedy fashion.

3.3.2 Description Minimization

The second step seeks to remove single conditions from the remaining rules without 'significantly' changing the subgroup's model. Note that, here, we do no try to increase the pattern's quality, *per se*. Indeed, we allow the quality to decrease when we remove conditions, as long as doing so does not 'significantly' change the subgroup's model. The reasoning here may be explained as an application of Occam's razor; if two patterns describe subgroups with roughly identical models, we prefer the least complex of these patterns, that is, the one with the fewest conditions.

Algorithmically, the second step works analogously to the single-condition removal phase of the first step. We begin by sorting all conditions in the remaining rules, this time in order of increasing influence on the subgroup's model. To this end, we use the quality measure $\varphi(\cdot)$, which as noted is essentially a distance measure between two models, computed between the current subgroup and the subgroup resulting from removing one of the conditions. The conditions are then considered for removal in order of the influence of their removal on the subgroup's model.

Let Λ denote the remaining rule set, and let Λ' denote that rule set but with the condition under consideration removed. That condition is then actually removed, whenever the models described by Λ and Λ' are not significantly different at a confidence level to be specified by the user. Because the exact method of testing for the significance of the model difference will depend on the model class, we do not go into detail here; Section 4 will give the specifics for use with the linear regression model class. This procedure is repeated for all conditions in the rule set. The second step of the post-processing is completed when all conditions have been considered in this fashion.

3.3.3 Interestingness Determination

Finally, the third step of the post-processing attempts to verify whether the resulting pattern is indeed 'interesting', and discards the entire result if it is not. This may be seen as a way to guard against returning relatively uninteresting local optima. Procedurally, we do this by testing whether the global model and the subgroup's model are significantly different at some user-specified confidence level. If this difference is found not to be significant, we conclude that the induced models do not differ enough to consider the result interesting. In this case, the result is discarded, and optimization re-run from a different starting position. Otherwise, the result is accepted and added to the set of results to be presented to the user.

4 TCGA with Regression Models

In this section, we instantiate the TCGA algorithm introduced in the previous section for use with linear regression models.

4.1 Model Estimation

For notational convenience, we begin by rewriting the targets x^i of all records in the dataset $\mathcal{D}$ in matrix notation. We note that the domain of these targets is here given by $\mathcal{X} = \mathbb{R}^p$. Let $\mathbf{X}$ be an $n \times p$ matrix, where for all $i = 1, \ldots, n$, the i-th row is given by $[1, x_1^i, \ldots, x_{p-1}^i]$. Let $\mathbf{y}$ denote the $n \times 1$ vector $\mathbf{y} = [x_p^1, \ldots, x_p^n]^\top$.

We consider the linear regression model given by

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{6}$$

where $\boldsymbol{\beta}$ is the $p \times 1$ vector of coefficients, and $\boldsymbol{\varepsilon}$ is an $n \times 1$ vector of i.i.d. Gaussian errors with $\mathbb{E}[\boldsymbol{\varepsilon}] = \mathbf{0}$ and $\mathrm{Var}[\boldsymbol{\varepsilon}] = \sigma^2\mathbf{I}$. When fitting this model to the entire dataset $\mathcal{D}$, it is well known that the OLS estimate $\hat{\boldsymbol{\beta}}$ is given by

$$\hat{\boldsymbol{\beta}} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{y}, \tag{7}$$

giving estimates $\hat{\mathbf{y}} = \mathbf{X}\hat{\boldsymbol{\beta}}$ and $\mathbf{e} = \mathbf{y} - \hat{\mathbf{y}}$.

To fit the model to a subgroup G_P described by pattern P, we introduce the $n \times n$ diagonal zero-one matrix $\mathbf{W}_P$, with diagonal elements $w_{ii}^P = P(a^i)$, for $i = 1, \ldots, n$. In the sequel, we will generally drop the subscript and refer to the matrix $\mathbf{W}$ for a subgroup G when no confusion should arise. Using this matrix, we may obtain the matrix $\mathbf{X}_G = \mathbf{W}\mathbf{X}$, which contains the targets of all records in the subgroup, and zero-rows at positions of records that are not included in G. When we instead consider a *soft* subgroup parameterized by $\mathbf{w}$, we let the diagonal of $\mathbf{W}$ be given by $w_{ii} = w_i$. In either case, the estimate $\hat{\boldsymbol{\beta}}_G$ of the coefficients of the model induced on G is then given by the Weighted Least Squares [5] estimate

$$\hat{\boldsymbol{\beta}}_G = \left(\mathbf{X}^\top\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{W}\mathbf{y}. \tag{8}$$

Using these coefficients, we can still obtain estimates for *all* records in the dataset, given by $\hat{\mathbf{y}}_G = \mathbf{X}\hat{\boldsymbol{\beta}}_G$ and $\mathbf{e}_G = \mathbf{y} - \hat{\mathbf{y}}_G$.

4.2 Quality Measure

Previous work by Duivesteijn *et al.* [9] proposed the use of Cook's Distance [5] as a quality measure for EMM with linear regression models. Cook's Distance $D(\cdot)$ is given by

$$D(G) \equiv \frac{\left(\hat{\boldsymbol{\beta}}_G - \hat{\boldsymbol{\beta}}\right)^\top \mathbf{X}^\top\mathbf{X}\left(\hat{\boldsymbol{\beta}}_G - \hat{\boldsymbol{\beta}}\right)}{ps^2}, \tag{9}$$

where $s^2 = \frac{\mathbf{e}^\top\mathbf{e}}{n-p}$ is the unbiased estimator of the error variance σ^2. Use of this measure may be motivated as follows. Cook [5] notes that the $(1-\alpha) \times 100\%$ confidence ellipsoid of $\hat{\boldsymbol{\beta}}$ is given by all vectors $\boldsymbol{\beta}^*$ satisfying

$$\frac{\left(\boldsymbol{\beta}^* - \hat{\boldsymbol{\beta}}\right)^\top \mathbf{X}^\top\mathbf{X}\left(\boldsymbol{\beta}^* - \hat{\boldsymbol{\beta}}\right)}{ps^2} \leq F(p, n-p, 1-\alpha), \tag{10}$$

where $F(p, n-p, 1-\alpha)$ denotes the $1-\alpha$ probability point of the central F-distribution with p and $n-p$ degrees of freedom. Thus, for example, if $D(G) \approx F(p, n-p, 0.5)$, we may say that $\hat{\boldsymbol{\beta}}_G$ is located roughly at the edge of the 50% confidence ellipsoid of $\hat{\boldsymbol{\beta}}$.

Now, although this measure has the advantage of being easy to interpret, it does not take into account the *support* of a subgroup, and hence has no built-in protection against over-fitting. Therefore we define the new quality measure

$$\begin{aligned}\varphi_{\mathcal{D}}(P) &\equiv \frac{|G_P|}{n} \cdot \frac{\left(\hat{\boldsymbol{\beta}}_{G_P} - \hat{\boldsymbol{\beta}}\right)^\top \mathbf{X}^\top\mathbf{X}\left(\hat{\boldsymbol{\beta}}_{G_P} - \hat{\boldsymbol{\beta}}\right)}{ps^2} \\ &= \frac{|G_P|}{n} \cdot \frac{\left(\hat{\mathbf{y}}_{G_P} - \hat{\mathbf{y}}\right)^\top \left(\hat{\mathbf{y}}_{G_P} - \hat{\mathbf{y}}\right)}{ps^2} \\ &= \frac{\mathrm{Tr}(\mathbf{W}_P)}{n} \cdot \frac{\|\hat{\mathbf{y}}_{G_P} - \hat{\mathbf{y}}\|^2}{ps^2}. \end{aligned} \tag{11}$$

Here, $\|\cdot\|^2$ is the squared L_2-norm, and the trace $\mathrm{Tr}(\mathbf{W}) = \sum_{i=1}^n w_{ii} = |G|$ gives the size of the subgroup. Due to the factor p^{-1} in Equation 11, the quality is scale-dependent on the number of regression coefficients used, which makes the choice of the gradient ascent step-size more data set-dependent. Hence we finally adjust the quality measure to:

$$\varphi_{\mathcal{D}}(P) \equiv \frac{\mathrm{Tr}(\mathbf{W}_P)}{n} \cdot \frac{\|\hat{\mathbf{y}}_{G_P} - \hat{\mathbf{y}}\|^2}{s^2}. \tag{12}$$

When we parameterize this quality measure as an objective function $O(\mathbf{w})$ of a vector $\mathbf{w}$ of soft inclusion weights, we may take partial derivatives with respect to the individual weights. For the i-th weight, this derivative is given (cf. Appendix A) by

$$\begin{aligned}\frac{\partial O(\mathbf{w})}{\partial w_i} &= \frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2ns^2} \\ &\quad - \frac{\mathrm{Tr}(\mathbf{W})}{ns^2}\left(\mathrm{diag}(\mathbf{e}_G)\mathbf{X}\left(\mathbf{X}^\top\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{e}_G\right)_i. \end{aligned} \tag{13}$$

Inspection of this equality reveals that, if one is careful with the order and method of multiplication used, the complete set of derivatives can be computed in a time complexity order of $\mathcal{O}(p^2 n)$ (cf. Appendix B).

4.3 Significance of Model Difference

As described in Section 3.3, we require for the post-processing procedure a method to determine whether the difference between two models is significant at a given confidence level. For two models described by, say, the rule sets Λ and Λ', we have implemented this by testing for the intersection of the $(1-\alpha) \times 100\%$ confidence ellipsoids of the coefficients of these models (cf. Appendix C). The general formula for these ellipsoids is given by Equation 10. Note that α is thus a parameter of the post-processing procedure and that its value is to be specified by the user.

4.4 Application to SD

There is an interesting relation between the general problem of Subgroup Discovery (SD) [14] and EMM with linear regression models. In SD we consider a data set with records of the form

$$r \equiv \langle a_1, \ldots, a_k, x \rangle. \tag{14}$$

Using this dataset for EMM with linear regression models is easily seen to reduce the models under consideration to only the intercept term. Furthermore, the OLS estimate of the intercept is $\hat{\beta}_0 = \bar{x}$, where $\bar{x}$ is the sample mean of the single target variable. The quality measure given by Equation 11 thus becomes, after some simplification,

$$\begin{aligned} \varphi_{\mathcal{D}}(P) &= |G_P| \cdot \frac{(\bar{x}_{G_P} - \bar{x})^2}{s^2} \\ &\propto |G_P| \cdot (\bar{x}_{G_P} - \bar{x})^2, \end{aligned} \tag{15}$$

where $\bar{x}_G$ denotes the sample mean of the target variable within the subgroup G. Compare this to the *Mean Test* (MT) quality measure described by Klösgen [13], written for the case where one is interested in subgroups with both exceptionally high or low target values:

$$\text{MT}(P) \equiv \sqrt{|G_P|} \cdot |\bar{x}_{G_P} - \bar{x}|. \tag{16}$$

Comparison of Equation 15 and Equation 16 reveals that maximization of $\varphi_{\mathcal{D}}(\cdot)$ is equivalent to maximization of $\text{MT}(\cdot)$. This relationship shows that we may regard some forms of SD as special cases of EMM with linear regression models. Hence, the tree-constrained gradient ascent algorithm may also be readily applied to these instances of SD.

5 Experiments

Experiments were performed to evaluate the performance of the Tree-Constrained Gradient Ascent (TCGA) algorithm. We compare the algorithm both with a standard Beam Search (BS) strategy, and with a BS strategy to which we also apply the post-processing method described in Section 3.3, which we will refer to as BSPP.

5.1 Synthetic Data

We first discuss experiments performed on synthetic data. This is useful in the sense that, when considering real world datasets, we do not have a golden standard. That is, on real world data we do not know whether a result is indeed an interesting, or objectively 'correct', answer. By constructing synthetic datasets with explicitly defined high quality subgroups, we may evaluate the performance of the algorithms by looking at how many of these 'correct' subgroups are returned.

5.1.1 Initial Experiments

For each data set we defined a global regression model and several subgroup regression models. The latter were constructed to have slopes orthogonal to that of the global model. The values of the target attributes were all drawn from a normal distribution.

We subdivide the experiments into variation of the number of subgroups, number of regression coefficients, and a set of experiments that used numeric descriptive attributes; the first two experiments used binary attributes. In all of these experiments, we furthermore varied the length of the subgroup descriptions and the error variance of the global model. The numeric values of these settings, as well as the size and number of the datasets sampled, are given in Table 2. Note that we set aside a third of each

	SUBGROUPS		COEFFICIENTS		NUM. ATTRIBUTES	
ALGORITHM	F_1	$\varphi_{\mathcal{D}}$ HM	F_1	$\varphi_{\mathcal{D}}$ HM	F_1	$\varphi_{\mathcal{D}}$ HM
(a) TCGA $\eta = 0.1$	$0.74^{\text{c,d,e,f}}$	$2.61^{\text{c,d,e,f}}$	$\mathbf{0.77}^{\text{c,d,e,f}}$	$\mathbf{1.65}^{\text{c,d,e,f}}$	$\mathbf{0.67}^{\text{b,c,d}}$	$\mathbf{1.40}^{\text{b,c,d}}$
(b) TCGA $\eta = 10.0$	$\mathbf{0.79}^{\text{a,c,d,e,f}}$	$\mathbf{2.23}^{\text{a,c,d,e,f}}$	$0.76^{\text{c,d,e,f}}$	$1.95^{\text{c,d,e,f}}$	$0.64^{\text{c,d}}$	$1.85^{\text{c,d}}$
(c) BS	0.06	5.87	0.04	5.80	0.20	3.33
(d) BSPP depth $= 6$	0.56^{c}	3.46^{c}	0.39^{c}	3.90^{c}	0.22	3.43
(e) BSPP depth $= 7$	0.56^{c}	3.43^{c}	0.40^{c}	3.85^{c}	$-$	$-$
(f) BSPP depth $= 8$	0.57^{c}	3.40^{c}	0.40^{c}	3.85^{c}	$-$	$-$
FRIEDMAN TEST p-VAL	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**

Table 1: Synthetic data results, by experiment type. Columns marked 'F_1' show the algorithms' average F_1 scores (higher is better). Columns marked '$\varphi_{\mathcal{D}}$ HM' show average *ranks* based on that measure (lower is better). There is a significant difference between the algorithms' performance on both the 'F_1' and '$\varphi_{\mathcal{D}}$ HM' measures for all experiments, with $p \approx 0$ in all cases. Superscripts behind results denote that the result is significantly better than that of the algorithm marked with the corresponding index, at the $\alpha = 0.01$ level. We did not run entries marked $-$ because they were expected to take a prohibitive amount of time.

dataset for evaluation purposes, and that this part of the data is not used for mining.

The parameters of the algorithms were chosen from some preliminary testing on related, but non-included, datasets. We ran the TCGA algorithm with 2 choices of step-size ($\eta \in \{0.1, 10.0\}$), performing 50 restarts. TCGA's minimum split supports were chosen to be $S_{\text{min}} = 50$ and $L_{\text{min}} = 10$. The BS algorithm's search depth was chosen to always correspond to the correct length of the subgroup descriptions, with a beam width of 50. The BSPP algorithm also had a beam width of 50, but several choices of search depth (values {6,7,8}), relying on the post-processing to simplify the descriptions to find the 'correct' ones. The minimum support for both BS and BSPP was set to 10. The post-processing's significance-level was chosen to be $\alpha = 0.01$ for both TCGA and BSPP.

Two measures were used for assessing performance. The first is the F_1 score, where we defined recall as the fraction of 'correct' subgroups found, and precision as the fraction of 'correct' results in the entire set of results. Intuitively, there will be a trade-off between finding *all* 'correct' subgroups (perfect recall), and finding *only* 'correct' subgroups (perfect precision), which is captured by the F_1 score.

The second measure is based on the quality $\varphi_{\mathcal{D}}$, and it is intended to capture the same trade-off for scenarios where we do *not* have access to the 'correct' answers, that is, when we apply the algorithm to real data. It is based on the assumption that 'correct' subgroups will have far higher quality than 'incorrect' ones. To emulate the recall measure, we compute the average $\varphi_{\mathcal{D}}$ over the top m *unique* results, where m is the actual number of correct subgroups in the data (which one would still have to guess in actual practice). To capture the properties of the precision measure, we simply take the average $\varphi_{\mathcal{D}}$ over the entire set of results. We then introduce the aggregate measure '$\varphi_{\mathcal{D}}$ HM', which takes the harmonic mean of these measures in an attempt to again reflect the mentioned trade-off.

Performance on $\varphi_{\mathcal{D}}$ is computed on the held-out evaluation datasets. The aggregated results are shown in Table 1. The results are compared using a Friedman test [7]. Inspection of Table 1 reveals that there is a significant difference in performance, on both measures and within all experiments, with $p \approx 0$ in all cases. This was followed by a Nemenyi *post-hoc* test [7], to perform a pairwise evaluation of the algorithms. This test corrects for the number of comparisons performed. We see that TCGA always significantly outperforms both BS and BSPP at the $\alpha = 0.01$ level. BSPP furthermore significantly outperforms BS in the 'subgroups' and 'coefficients' experi-

	EXPERIMENT		
SETTING	subgroups	coefficients	num. attributes
#Subgroups	{2, 4, 6}	4	4
#Coefficients	1	{2, 3, 4}	2
#Attributes	64	64	32
Description Length	{4, 5}	{4, 5}	{2, 4}
Global σ^2	{0.5, 2}	{0.5, 2}	{0.5, 2}
Datasets Sampled	50	50	50
Mining Records	3,000	3,000	3,000
Evaluation Records	1,500	1,500	1,500
Subgroup Support	12%	12%	12%
Total Combinations	12	12	4
Total Datasets	600	600	200

Table 2: Settings used for dataset generation in the three experiments performed on synthetic data.

ments, at the same confidence level. We finally observe a strong agreement between the F_1 score and the '$\varphi_{\mathcal{D}}$ HM' measure, which indicates that the latter here seems to capture the characteristics of the results as intended.

5.1.2 Varying the Slope of the Subgroups

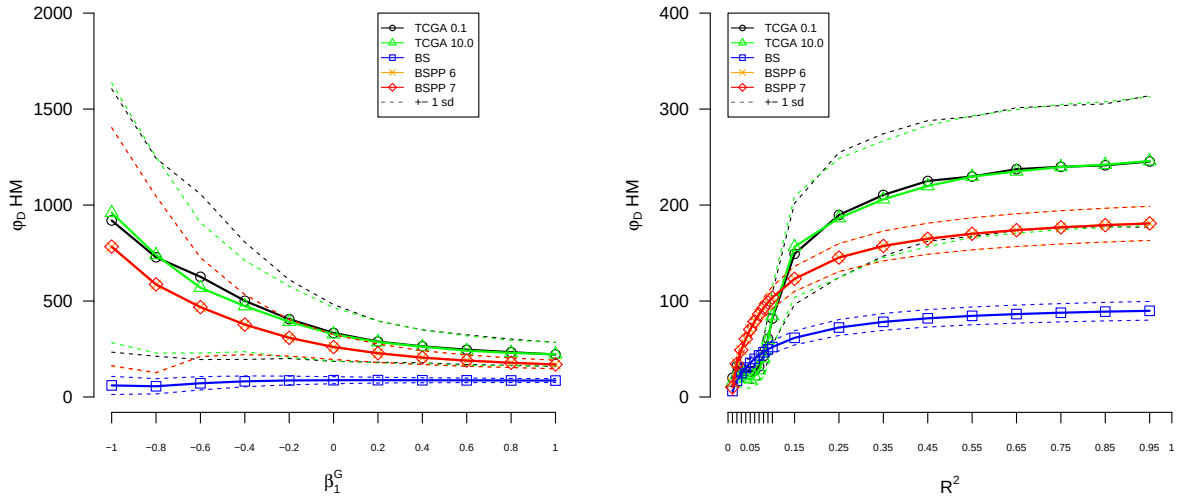
Further experiments were performed to investigate whether there are specific situations in which TCGA's performance becomes notably worse, in particular in comparison to BS. We conjectured that both a smaller difference between global slope and subgroup slope, and a worse degree of fit of the global model could have a negative impact on the performance of TCGA. This was tested by performing experiments in which these characteristics were explicitly controlled. We first consider the impact of slope difference on the performance of the algorithms.

In our previous experiments, all subgroups' models were defined as being orthogonal to the global model. To vary the subgroups' quality, experiments were performed using simple linear regression models, i.e., using models with only a single regression coefficient. Here, the subgroups' regression lines were varied from being completely orthogonal, to being completely parallel, to that of the global model. Thus, in the latter case, only the intercept term of the subgroups' models would be different. We performed these experiments with the same parameter settings for the algorithms as before, only leaving out BSPP with a search depth of 8 because of the previously observed lack of sensitivity to this parameter. We again used subgroup description lengths of 4 and 5 conditions, and two different values for the error variance of the global model. As before, 50 datasets were generated for each combination of these settings. This was repeated at 11 settings for the subgroups' slopes. The results of these experiments are given in Figure 2(a), with performance shown on the '$\varphi_{\mathcal{D}}$ HM' measure averaged over all 200 datasets at each setting of the slope. Here, a value of -1 for the slope corresponds to complete orthogonality, while at the value 1 the subgroup model is completely parallel to the global model.

We observe that the algorithms' average performance on this measure is indeed negatively influenced by the subgroup models' slope. We furthermore see that TCGA on average still performs better than both BSPP and BS, and that BSPP on average has a higher score than BS.

However, we are also interested in the relative performance of TCGA compared to the other algorithms. We therefore look at the average rank of TCGA, and its correlation with the slope of the subgroup models. To do this, we use the non-parametric Spearman's rank correlation coefficient ρ [20], computed between the subgroup models' slope and the average rank of TCGA, where the rankings of the algorithms are aggregated over their parameter settings. Here, we correct for the fact that lower ranks are better, so that a positive correlation implies improved performance.

From our conjecture, we expected to find a negative correlation between TCGA's average rank and the slope of the subgroup model. However, we found no significant negative correlation between the (lack of) orthogonality of the slope of the subgroup models and the rank of the TCGA algorithm based on the '$\varphi_{\mathcal{D}}$ HM' measure, at the $\alpha = 0.05$ level. When we perform the same test for the ranking based on the algorithm's F_1 score, however, we



(a) (b)

Figure 2: (a) Average '$\varphi_{\mathcal{D}}$ HM' vs. slope in the subgroups. (b) Average '$\varphi_{\mathcal{D}}$ HM' vs. R^2 of the global model.

do find a significant negative correlation (Spearman's $\rho \approx -0.84$, with $p \approx 0.001$).

5.1.3 Varying R^2 of the Global Model

To investigate the influence of the global model's fit to the data, we performed experiments in which its R^2 value was varied. Because we are using simple linear regression models, we can solve for the desired R^2 in terms of the model's error variance, when keeping fixed the model's slope and the explanatory variable's variance.

We again used the same parameter settings as before, but here only varied the length of the subgroup descriptions as the error variance was already accounted for. Thus, we generated 100 datasets for each setting of the R^2. We initially tested 10 different R^2 values ranging from 0.05 to 0.95. Upon observing a very steep increase in performance of all algorithms between the 0.05 and 0.15 values, we then sampled an additional 9 settings over the 0.01 to 0.10 range, which smoothed out the results in this region as desired. The results of these experiments are given in Figure 2(b), with performance shown on the '$\varphi_{\mathcal{D}}$ HM' measure averaged over all 100 datasets at each R^2 setting.

We see that the average performance of all the algorithms increases with the global model's R^2. Furthermore, we see that TCGA on average scores higher than both BSPP and BS, except at fairly low values of the R^2. In such cases, the algorithm's average performance drops below that of the other two algorithms. We furthermore observe that BSPP on average outperforms BS, regardless of the R^2 value.

From our conjecture about the R^2's influence on TCGA's relative performance, we here expect to find a positive correlation between TCGA's average rank and the global model's R^2. Performing the same correlation test as before, we indeed find a significant positive correlation between the R^2 of the global model and the rank based on the '$\varphi_{\mathcal{D}}$ HM' score (Spearman's $\rho \approx 0.77$, with $p \approx 8 \times 10^{-5}$). This is further corroborated by the correlation of the R^2 of the global model with TCGA's rank based on the F_1 score (Spearman's $\rho \approx 0.54$, with $p \approx 0.009$).

5.2 Real Data

A second set of experiments was performed on real world datasets, taken from a number of sources. Several of these datasets were used multiple times, with different regression models as targets. A full overview, including the sources, is given in Table 3. Table 4 shows some summary statistics of the datasets. Note that model instances with $p = 1$ correspond to instances of SD as described in Section 4.4. In total, we used 21 different dataset/model pairs, divided into 10 EMM dataset/model pairs and 11 SD datasets.

DATASET	n	k	p
Adult	30,162	5; 5	4; 1
AmesHousing	2,930	8	3
Bioassay	27,189	153	1
Contraceptives	1,473	8; 6; 8	2; 4; 1
Credit	13,390; 10,461	5; 7	3; 1
EAEF	2,485	39; 36; 41	3; 3; 1
Extramarital	6,366	3; 8	6; 1
Microsoft Web	37,711	293	1
Opt. Digits	5,621	64	1
Plants	34,781	69	1
Spambase	4,601	57	1
WindsorHousing	546	7	5
Wine	9,600	6; 7	4; 1

Table 4: Statistics on the datasets. Columns denote number of records (n), attributes (k), and targets (p). Semicolons separate the values where multiple models are used on one dataset.

We use the '$\varphi_{\mathcal{D}}$ HM' measure, described in Section 5.1.1, as the quality measure. For each dataset/model combination and algorithm, the algorithm's parameter values were optimized on this measure, and performance evaluated, using 5-fold cross-validation with a 4-fold inner cross-validation parameter optimization loop. We picked the value of η from $\{0.1, 10\}$, of α from $\{0.01, 0.05, 0.1\}$, and the depth of the beam search from $\{1, 2, 4\}$. As before, all algorithms were to return 50 results, corresponding to 50 restarts or a beam width of 50. The minimum split supports for TCGA were fixed to $S_{\text{min}} = 125$ and $L_{\text{min}} = 50$. The minimum support for both BS and BSPP was set to 50. For the computation of the '$\varphi_{\mathcal{D}}$ HM' measure we chose $m = 5$.

The folds were constructed by uniform random subsampling from the dataset, without replacement, up to a maximum of 1,000 records per fold. This process was repeated 10 times, with replacement, for each dataset/model and algorithm pair, after which algorithms were paired and ranked for each sub-sampling. Average ranks were then computed, and the algorithms re-ranked, to obtain the relative performance of each algorithm on that dataset/model combination. Finally, average ranks were computed for both the EMM and SD results, to obtain the final relative performance on both of the problems.

We first discuss the EMM results, which are shown in the left column of Table 5. Here, the different datasets are shown in order of their global model's R^2, along with the algorithms' average ranks per dataset. We find that TCGA on average ranks highest on 5 of these 10 dataset/model pairs, with BS performing best on the other 5. BSPP here never ranks highest.

Looking at the algorithms' average ranks over all these datasets, we see that BS performs slightly better than TCGA, and that both of these algorithms outperform BSPP. Again performing a Friedman test to determine the significance of these differences, we find $p \approx 0.016$. Following with a Nemenyi *post-hoc* test for the pairwise differences, we find no significant difference, at the $\alpha = 0.05$ level, between TCGA and BS, nor between TCGA and BSPP. However, we find that BS outperforms BSPP with $p \approx 0.04$.

Returning to the algorithms' performance on the individual datasets, we see that TCGA tends to rank better when the global models' R^2 is higher, whereas BS tends to work better when the models' fit isn't as good. This is consistent with our findings on the synthetic data (see Section 5.1.3). Indeed, quantifying the relationship between the average ranks of TCGA and the global models' R^2, we here also find a significant positive correlation (Spearman's $\rho \approx 0.59$, with $p \approx 0.036$).

Looking next at the results on the SD datasets, shown in the right column of Table 5, we find that BS here tends to rank highest. On average, the difference between the algorithms' ranks is found significant by a Friedman test, with $p \approx 0.006$. A *post-hoc* test reveals that BS significantly outperforms TCGA, with $p \approx 0.01$, but that no other significant pairwise differences can be established.

EMM Dataset ID	Regression Model
Adult [2]	$(\geq 50K) \sim \beta_0 + \beta_1 \times$ age $+ \beta_2 \times$ fnlwgt $+ \beta_3 \times$ hours_per_week
AmesHousing [12]	sale_price $\sim \beta_0 + \beta_1 \times$ lot_area $+ \beta_2 \times$ overall_quality
Contraceptives 1 [2]	num_children $\sim \beta_0 + \beta_1 \times$ age
Contraceptives 2 [2]	num_children $\sim \beta_0 + \beta_1 \times$ age $+ \beta_2 \times$ education $+ \beta_3 \times$ uses_contraceptive
Credit [11]	is_cardholder $\sim \beta_0 + \beta_1 \times$ minordrg $+ \beta_2 \times$ majordrg
EAEF 1 [8]	earnings $\sim \beta_0 + \beta_1 \times$ years_school $+ \beta_2 \times$ years_experience
EAEF 2 [8]	weight02 $\sim \beta_0 + \beta_1 \times$ weight85 $+ \beta_2 \times$ height
ExtraMarital [11]	num_children $\sim \beta_0 + \beta_1 \times$ marriage_rating $+ \beta_2 \times$ years_married $+\beta_3 \times$ religiosity $+ \beta_4 \times$ education $+ \beta_5 \times$ occupation
WindsorHousing [1]	sale_price $\sim \beta_0 + \beta_1 \times$ lot_size $+ \beta_2 \times$ nbed $+ \beta_3 \times$ nbath $+ \beta_4 \times$ nstoreys
Wine [6]	price $\sim \beta_0 + \beta_1 \times$ cases $+ \beta_2 \times$ score $+ \beta_3 \times$ age
SD Dataset ID	**Target**
Adult [2]	$(\geq 50K)$
BioAssay 688 [19]	outcome
Contraceptives [2]	uses_contraceptive
Credit [11]	default
EAEF [8]	earnings
ExtraMarital [11]	YRB
MicrosoftWeb [2]	visited_MS_Office_Support
OptDigits [2]	is_five
Plants [2]	occurs_in_Minnesota
Spambase [2]	is_spam
Wine [6]	price

Table 3: Names of the dataset/model pairs that we used, including sources of the datasets. The corresponding EMM regression models and SD targets are shown in the second column.

EMM RESULTS

DATASET/MODEL	MODEL'S R^2	ALGORITHMS' $\varphi_{\mathcal{D}}$ HM TCGA	BS	BSPP
Adult	0.10	3.00	**1.10**	1.90
Credit	0.18	2.05	**1.60**	2.35
EAEF 1	0.21	**1.15**	2.00	2.85
Contraceptives 1	0.29	1.90	**1.10**	3.00
Wine	0.31	2.85	**1.35**	1.80
Contraceptives 2	0.37	**1.20**	1.80	3.00
WindsorHousing	0.54	**1.00**	2.00	3.00
ExtraMarital	0.60	1.90	**1.35**	2.75
EAEF 2	0.67	**1.45**	2.65	1.90
AmesHousing	0.68	**1.05**	2.05	2.90
MEAN		1.76	**1.70**	2.55
FRIEDMAN TEST		$p \approx 0.016$		

PAIRWISE p-VAL	TCGA	BS	BSPP
TCGA	–	0.97	0.07
BS	0.97	–	**0.04**
BSPP	0.07	**0.04**	–

SD RESULTS

DATASET	ALGORITHMS' $\varphi_{\mathcal{D}}$ HM TCGA	BS	BSPP
Adult	2.90	**1.10**	2.00
BioAssay 688	2.60	**1.00**	2.40
Contraceptives	2.75	**1.20**	2.05
Credit	3.00	**1.00**	2.00
EAEF	2.00	**1.35**	2.65
ExtraMarital	**1.80**	2.10	2.10
MicrosoftWeb	**1.05**	2.00	2.95
OptDigits	3.00	**1.00**	2.00
Plants	3.00	**1.00**	2.00
Spambase	3.00	**1.00**	2.00
Wine	3.00	1.65	**1.35**
MEAN	2.55	**1.31**	2.14
FRIEDMAN TEST	$p \approx 0.006$		

PAIRWISE p-VAL	TCGA	BS	BSPP
TCGA	–	**0.01**	0.60
BS	**0.01**	–	0.13
BSPP	0.60	0.13	–

Table 5: Experimental results on the real data. The table on the left shows the results of EMM with linear regression models. The table on the right shows the results of the experiments with conventional subgroup discovery.

6 Conclusions and Future Work

We have introduced tree-constrained gradient ascent (TCGA), a novel search strategy for EMM. On the one hand, TCGA exploits information on the contribution of individual records to subgroup quality, and on the other hand it guarantees that the subgroup extension can be described concisely in the pattern language.

Furthermore, we have shown how TCGA can be applied to EMM with the linear regression model class, and performed extensive experiments with this instantiation of TCGA.

Our experiments on synthetic data have shown that TCGA can significantly outperform beam search. These experiments also showed that our post-processing technique can improve the results of beam search, and at the same time circumvent explicit optimization of that algorithm's depth parameter.

In further experiments we varied the slope difference between the global model and the subgroup model, as well as the degree of fit of the global model. These experiments showed that the relative performance of TCGA compared to BS decreased as the slope difference and the fit of the global model became smaller.

The latter association was also observed in our experiments with real data, where we found a significant correlation between the degree of fit of the global model and the relative performance of TCGA. However, overall we did not find a significant difference between the performance of TCGA and BS in finding exceptional regression models. On the task of classical subgroup discovery, BS clearly outperformed TCGA.

Overall we conclude that the task of EMM with linear regression models can be effectively solved by TCGA when the degree of fit of the global regression model is not too small. We do not consider this a large constraint as it seems to fit the use case in an applied setting.

Our findings also raise a number of questions for further research concerning this instantiation of TCGA. For

instance, we would like to find an explanation for the observed correlation between the R^2 of the global model and the relative performance of TCGA. Also, it is not clear why TCGA performs relatively badly on the classical subgroup discovery task. Finally, while the post-processing procedure improved the performance of BS on the synthetic data, the opposite effect was observed on the real data.

The performance of the generic TCGA algorithm might be improved with more sophisticated numerical optimization methods, e.g., by incorporating a line search to automatically determine the step-size. Also, currently only a single gradient ascent update step is performed in between two consecutive splitting steps. It would be interesting to investigate the effect of performing more update steps.

In closing, we note that the TCGA algorithm is quite general, requiring only that the quality measure can be made differentiable by the use of soft subgroups. Thus, extending this work to other EMM model classes may prove to be both useful and relatively straightforward.

Appendix

This appendix contains some supplementary derivations and proofs related to our instantiation of TCGA with linear regression models.

Appendix A contains the derivation of the partial derivative of the objective function $O(\cdot)$ with respect to a given inclusion weight w_i. We show that this set of n partial derivatives can be computed in a time complexity order of $\mathcal{O}(p^2 n)$ in Appendix B. Finally, a method for testing for the intersection of two confidence ellipsoids is given in Appendix C.

A Derivative of Objective Function.

In this section, we give the derivation of the derivative of the objective function $O : [0,1]^n \to \mathbb{R}$ given by

$$O(\mathbf{w}) \equiv \frac{\mathrm{Tr}(\mathbf{W})}{n} \cdot \frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2s^2},$$

where for all $i = 1, \ldots, n$, the i-th diagonal element of the diagonal matrix $\mathbf{W}$ is given by w_i.

We show the derivation of the partial derivative of $O(\cdot)$ with respect to w_i. We first note that by application of the product rule for partial derivatives, and the equality $\frac{\partial \mathrm{Tr}(\mathbf{W})}{\partial w_i} = 1$, we have

(17)
$$\begin{aligned}
\frac{\partial O(\cdot)}{\partial w_i} &= \frac{\partial}{\partial w_i}\left[\frac{\mathrm{Tr}(\mathbf{W})}{n} \cdot \frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2s^2}\right] \\
&= \frac{\partial \mathrm{Tr}(\mathbf{W})}{\partial w_i} \cdot \frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2ns^2} \\
&\quad + \frac{\mathrm{Tr}(\mathbf{W})}{ns^2} \cdot \frac{\partial}{\partial w_i}\left[\frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2}\right] \\
&= \frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2ns^2} + \frac{\mathrm{Tr}(\mathbf{W})}{ns^2} \cdot \frac{\partial}{\partial w_i}\left[\frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2}\right].
\end{aligned}$$

We expand the remaining partial derivative in the final equality of Equation 17 separately. Noting first that this derivative depends only on w_i through $\hat{\boldsymbol{\beta}}_G$, we obtain

(18)
$$\begin{aligned}
\frac{\partial}{\partial w_i}\left[\frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2}\right] &= \frac{\partial}{\partial w_i}\left[\frac{(\hat{\mathbf{y}}_G - \hat{\mathbf{y}})^\top (\hat{\mathbf{y}}_G - \hat{\mathbf{y}})}{2}\right] \\
&= \frac{\partial}{\partial w_i}\left[\frac{\left(\hat{\boldsymbol{\beta}}_G - \hat{\boldsymbol{\beta}}\right)^\top \mathbf{X}^\top \mathbf{X}\left(\hat{\boldsymbol{\beta}}_G - \hat{\boldsymbol{\beta}}\right)}{2}\right] \\
&= \left(\hat{\boldsymbol{\beta}}_G - \hat{\boldsymbol{\beta}}\right)^\top \mathbf{X}^\top \mathbf{X} \frac{\partial \hat{\boldsymbol{\beta}}_G}{\partial w_i}.
\end{aligned}$$

Again expanding the term $\frac{\partial \hat{\boldsymbol{\beta}}_G}{\partial w_i}$ separately, we find

(19)
$$\begin{aligned}
\frac{\partial \hat{\boldsymbol{\beta}}_G}{\partial w_i} &= \frac{\partial}{\partial w_i}\left[(\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} \mathbf{y}\right] \\
&= \frac{\partial}{\partial w_i}\left[(\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1}\right] \mathbf{X}^\top \mathbf{W} \mathbf{y} \\
&\quad + (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \frac{\partial}{\partial w_i}\left[\mathbf{X}^\top \mathbf{W} \mathbf{y}\right],
\end{aligned}$$

by the definition of $\hat{\boldsymbol{\beta}}_G$ and application of the product rule for partial derivatives. For notational convenience, we define an auxiliary $p \times n$ matrix $\mathbf{A}$ such that

$$\mathbf{A} \equiv (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top.$$

Making use of the definition of $\mathbf{A}$, we can simplify the second summand in Equation 19 to find

$$\begin{aligned}\frac{\partial \hat{\boldsymbol{\beta}}_G}{\partial w_i} &= \frac{\partial}{\partial w_i}\left[(\mathbf{X}^\top \mathbf{W}\mathbf{X})^{-1}\right]\mathbf{X}^\top\mathbf{W}\mathbf{y} \\ &\quad + (\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}\frac{\partial}{\partial w_i}\left[\mathbf{X}^\top\mathbf{W}\mathbf{y}\right] \\ &= \frac{\partial}{\partial w_i}\left[(\mathbf{X}^\top \mathbf{W}\mathbf{X})^{-1}\right]\mathbf{X}^\top\mathbf{W}\mathbf{y} \\ &\quad + (\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^\top\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{y} \\ &= \frac{\partial}{\partial w_i}\left[(\mathbf{X}^\top \mathbf{W}\mathbf{X})^{-1}\right]\mathbf{X}^\top\mathbf{W}\mathbf{y} + \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{y}.\end{aligned}$$

Also simplifying the first summand, using the equality [17] $\frac{\partial \mathbf{M}^{-1}}{\partial x} = -\mathbf{M}^{-1}\frac{\partial \mathbf{M}}{\partial x}\mathbf{M}^{-1}$ for any matrix $\mathbf{M}$, we obtain

$$\begin{aligned}\frac{\partial \hat{\boldsymbol{\beta}}_G}{\partial w_i} &= -(\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}\frac{\partial}{\partial w_i}\left[\mathbf{X}^\top\mathbf{W}\mathbf{X}\right] \\ &\qquad \cdot (\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{W}\mathbf{y} + \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{y} \\ &= -(\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}\frac{\partial}{\partial w_i}\left[\mathbf{X}^\top\mathbf{W}\mathbf{X}\right]\mathbf{A}\mathbf{W}\mathbf{y} + \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{y} \\ &= -(\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^\top\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{X}\mathbf{A}\mathbf{W}\mathbf{y} + \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{y} \\ &= -\mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{X}\mathbf{A}\mathbf{W}\mathbf{y} + \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{y} \\ &= \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{y} - \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{X}\mathbf{A}\mathbf{W}\mathbf{y}.\end{aligned}$$

We note that $\hat{\boldsymbol{\beta}}_G$ is given by

$$\begin{aligned}\hat{\boldsymbol{\beta}}_G &= (\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{W}\mathbf{y} \\ &= \mathbf{A}\mathbf{W}\mathbf{y}.\end{aligned}$$

We thus find

$$\begin{aligned}\frac{\partial \hat{\boldsymbol{\beta}}_G}{\partial w_i} &= \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{y} - \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{X}\hat{\boldsymbol{\beta}}_G \\ &= \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{y} - \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\hat{\mathbf{y}}_G \\ &= \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\left(\mathbf{y} - \hat{\mathbf{y}}_G\right) \\ &= \mathbf{A}\frac{\partial \mathbf{W}}{\partial w_i}\mathbf{e}_G.\end{aligned}$$

Note that $\frac{\partial \mathbf{W}}{\partial w_i}$ is an $n \times n$ matrix with all elements zero, except for the i-th diagonal element, which equals one. Denote this matrix as $\mathbf{W}'$, and its diagonal elements as w'_j for all $j = 1, \ldots, n$. We have

$$\frac{\partial \hat{\boldsymbol{\beta}}_G}{\partial w_i} = \mathbf{A}\mathbf{W}'\mathbf{e}_G \tag{20}$$

$$= \sum_{1\le j\le n} \mathbf{A}_{\cdot j} w'_j (\mathbf{e}_G)_j \tag{21}$$

$$= \mathbf{A}_{\cdot i} w'_i (\mathbf{e}_G)_i + \sum_{\substack{1\le j\le n \\ j\neq i}} \mathbf{A}_{\cdot j} w'_j (\mathbf{e}_G)_j \tag{22}$$

$$= \mathbf{A}_{\cdot i}(\mathbf{e}_G)_i, \tag{23}$$

where $\mathbf{A}_{\cdot i}$ is the i-th column of $\mathbf{A}$. Substituting Equation 23 into Equation 18 and transposing yields

$$\begin{aligned}\frac{\partial}{\partial w_i}\left[\frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2}\right] &= \left(\hat{\boldsymbol{\beta}}_G - \hat{\boldsymbol{\beta}}\right)^\top \mathbf{X}^\top\mathbf{X}\frac{\partial \hat{\boldsymbol{\beta}}_G}{\partial w_i} \\ &= \left(\hat{\boldsymbol{\beta}}_G - \hat{\boldsymbol{\beta}}\right)^\top \mathbf{X}^\top\mathbf{X}\mathbf{A}_{\cdot i}(\mathbf{e}_G)_i \\ &= (\mathbf{e}_G)_i\left((\mathbf{A}_{\cdot i})^\top\mathbf{X}^\top\mathbf{X}\left(\hat{\boldsymbol{\beta}}_G - \hat{\boldsymbol{\beta}}\right)\right) \\ &= (\mathbf{e}_G)_i\left((\mathbf{A}_{\cdot i})^\top\mathbf{X}^\top\left(\mathbf{X}\hat{\boldsymbol{\beta}}_G - \mathbf{X}\hat{\boldsymbol{\beta}}\right)\right).\end{aligned}$$

Noting that we may take the dependency on i outside, and using the definition of $\hat{\mathbf{y}}_G$, we can simplify this as

$$\begin{aligned}\frac{\partial}{\partial w_i}\left[\frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2}\right] &= (\mathbf{e}_G)_i\left((\mathbf{A}_{\cdot i})^\top\mathbf{X}^\top\left(\hat{\mathbf{y}}_G - \mathbf{X}\hat{\boldsymbol{\beta}}\right)\right) \\ &= (\mathbf{e}_G)_i\left(\mathbf{A}^\top\mathbf{X}^\top\left(\hat{\mathbf{y}}_G - \mathbf{X}\hat{\boldsymbol{\beta}}\right)\right)_i.\end{aligned}$$

Temporarily multiplying through the $\mathbf{X}^\top$, expanding the term $\hat{\boldsymbol{\beta}}$, and multiplying by -1 twice, we find

$$\begin{aligned}\frac{\partial}{\partial w_i}\left[\frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2}\right] &= (\mathbf{e}_G)_i\left(\mathbf{A}^\top\left(\mathbf{X}^\top\hat{\mathbf{y}}_G - \mathbf{X}^\top\mathbf{X}\hat{\boldsymbol{\beta}}\right)\right)_i \\ &= (\mathbf{e}_G)_i\big(\mathbf{A}^\top\left(\mathbf{X}^\top\hat{\mathbf{y}}_G - \mathbf{X}^\top\mathbf{X}(\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}\right)\big)_i \\ &= (\mathbf{e}_G)_i\left(\mathbf{A}^\top\left(\mathbf{X}^\top\hat{\mathbf{y}}_G - \mathbf{X}^\top\mathbf{y}\right)\right)_i \\ &= (\mathbf{e}_G)_i\left(\mathbf{A}^\top\mathbf{X}^\top\left(\hat{\mathbf{y}}_G - \mathbf{y}\right)\right)_i \\ &= -(\mathbf{e}_G)_i\left(\mathbf{A}^\top\mathbf{X}^\top\left(\mathbf{y} - \hat{\mathbf{y}}_G\right)\right)_i \\ &= -(\mathbf{e}_G)_i\left(\mathbf{A}^\top\mathbf{X}^\top\mathbf{e}_G\right)_i.\end{aligned}$$

We observe that we may take the element $(\mathbf{e}_G)_i$ inside, using the diagonal matrix $\text{diag}(\mathbf{e}_G)$. We obtain

$$\frac{\partial}{\partial w_i}\left[\frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2}\right] = -\left(\text{diag}(\mathbf{e}_G)\mathbf{A}^\top \mathbf{X}^\top \mathbf{e}_G\right)_i.$$

Expanding the term $\mathbf{A}^\top$ yields

$$\frac{\partial}{\partial w_i}\left[\frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2}\right] = -\left(\text{diag}(\mathbf{e}_G)\mathbf{X}\left(\mathbf{X}^\top\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{e}_G\right)_i. \tag{24}$$

Substituting Equation 24 into Equation 17, we finally obtain

$$\begin{aligned}\frac{\partial O(\cdot)}{\partial w_i} &= \frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2ns^2} + \frac{\text{Tr}(\mathbf{W})}{ns^2}\cdot\frac{\partial}{\partial w_i}\left[\frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2}\right] \\ &= \frac{\|\hat{\mathbf{y}}_G - \hat{\mathbf{y}}\|^2}{2ns^2} \\ &\quad - \frac{\text{Tr}(\mathbf{W})}{ns^2}\left(\text{diag}(\mathbf{e}_G)\mathbf{X}\left(\mathbf{X}^\top\mathbf{W}\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{e}_G\right)_i.\end{aligned} \tag{25}$$

■

B Computational Complexity of Derivatives.

In this section, we show that the complete set of n derivatives, the i-th of which is given by Equations 13 and 25, can be computed in a time complexity order of $\mathcal{O}(p^2n)$.

In particular, care should be taken that multiplication with the diagonal matrices $\mathbf{W}$ and $\text{diag}(\mathbf{e}_G)$ is not performed naively. Then, the derivatives can be computed in the following way. We begin by showing how to compute $\hat{\boldsymbol{\beta}}_G$ in time $\mathcal{O}(p^2n)$.

Start with the computation of $\mathbf{X}_G = \mathbf{W}\mathbf{X}$, which if not done naively can be performed in $\mathcal{O}(pn)$. Form the $p \times p$ matrix $\mathbf{M} = (\mathbf{X}^\top\mathbf{W}\mathbf{X})$ by computing $\mathbf{X}^\top\mathbf{X}_G$, which takes $\mathcal{O}(p^2n)$. Likewise, form the $p \times 1$ vector $\mathbf{z} = \mathbf{X}^\top\mathbf{W}\mathbf{y}$ from $\mathbf{X}_G^\top\mathbf{y}$, taking $\mathcal{O}(pn)$. Find $\mathbf{M}^{-1}$, the complexity of which depends on the specific algorithm used, but a straightforward implementation takes $\mathcal{O}(p^3)$. Finally compute $\hat{\boldsymbol{\beta}}_G = \mathbf{M}^{-1}\mathbf{z}$ in $\mathcal{O}(p^2)$. The total time complexity is thus $\mathcal{O}(pn+p^2n+pn+p^3+p^2) = \mathcal{O}(p^3+p^2n)$. By the linear regression assumption that $\mathbf{X}$ has full column rank, we have $n \geq p$. Hence, $p^2n \geq p^3$, and $\mathcal{O}(p^3+p^2n) = \mathcal{O}(p^2n)$.

Having obtained $\hat{\boldsymbol{\beta}}_G$ for the current $\mathbf{W}$, we compute the estimates of the regression target values and the vector of residuals. We compute $\hat{\mathbf{y}}_G = \mathbf{X}\hat{\boldsymbol{\beta}}_G$, taking $\mathcal{O}(pn)$. We then find $\mathbf{e}_G = \mathbf{y} - \hat{\mathbf{y}}_G$ in $\mathcal{O}(n)$.

As an aside, we assume that $\hat{\mathbf{y}}$ and s^2 are already known at this point, but from the above it is easily seen that these can also be found in $\mathcal{O}(p^2n+pn+n+n) = \mathcal{O}(p^2n)$ by setting $\mathbf{W} = \mathbf{I}$. Here, we have assumed that the estimate of the error variance s^2 is computed in $\mathcal{O}(n)$ from $s^2 = \frac{\mathbf{e}^\top\mathbf{e}}{n-p}$, having obtained $\mathbf{e}$ as outlined above.

We now have all the terms required to compute the partial derivatives in time $\mathcal{O}(pn)$. The first summand in Equation 13 can clearly be computed in $\mathcal{O}(n)$ using the pre-computed vectors $\hat{\mathbf{y}}_G$ and $\hat{\mathbf{y}}$. Further, this term does not depend on i, so we do not need to re-compute it for every partial derivative. Similarly, the scalar of the second summand, $\frac{\text{Tr}(\mathbf{W})}{ns^2}$, only needs to be computed once, also taking $\mathcal{O}(n)$.

We now show that the entire vector in the derivative's second summand can be computed in $\mathcal{O}(pn)$. Start by computing the $p \times 1$ vector $\mathbf{a} = \mathbf{X}^\top\mathbf{e}_G$ in time $\mathcal{O}(pn)$. Having previously obtained $\mathbf{M}^{-1} = (\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}$, compute the $p \times 1$ vector $\mathbf{b} = (\mathbf{X}^\top\mathbf{W}\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{e}_G = \mathbf{M}^{-1}\mathbf{a}$ in time $\mathcal{O}(p^2)$. Form the $n \times 1$ vector $\mathbf{c} = \mathbf{X}\mathbf{b}$ in $\mathcal{O}(pn)$. Taking care not to multiply by the diagonal matrix naively, we finally find the entire vector $\mathbf{d}$ in the derivative's second summand by computing $\mathbf{d} = \text{diag}(\mathbf{e}_G)\mathbf{c}$ in time $\mathcal{O}(n)$. Thus, the total time complexity to find $\mathbf{d}$ is $\mathcal{O}(pn+p^2+pn+n) = \mathcal{O}(pn)$.

The i-th partial derivative can now be computed in $\mathcal{O}(1)$ from the pre-computed first summand, the pre-computed scalar in the second summand, and the i-th element in $\mathbf{d}$.

Thus, the total time complexity is $\mathcal{O}(p^2n)$ to find $\hat{\boldsymbol{\beta}}_G$, $\mathcal{O}(pn)$ for the estimates $\hat{\mathbf{y}}_G$ and $\mathbf{e}_G$, $\mathcal{O}(pn)$ to find the vector $\mathbf{d}$, and n times $\mathcal{O}(1)$ to compute the individual derivatives, for a total of $\mathcal{O}(p^2n+pn+pn+n) = \mathcal{O}(p^2n)$.

■

C Ellipsoid Intersection Testing.

In this section, we describe how to test for the intersection of two ellipsoids, as this is a non-trivial problem in general. The procedure works by reducing the intersection test to a single variable convex optimization problem with a single linear inequality constraint, which may then be solved numerically. In our paper, this procedure is applied to testing for the intersection of two confidence ellipsoids. Here, we give a somewhat more general treatment, noting that applying the procedure to confidence ellipsoids is straightforward.

We consider two ellipsoids E_1 and E_2, the j-th of which is given by all vectors $\mathbf{x}$ satisfying

$$E_j(\mathbf{x}) \equiv (\mathbf{x} - \mathbf{c}_j)^\top \mathbf{M}_j (\mathbf{x} - \mathbf{c}_j) \leq 1. \tag{26}$$

Here, $\mathbf{c}_j$ is the center of the j-th ellipsoid, and $\mathbf{M}_j$ is a symmetric positive definite matrix defining the ellipsoid's norm.

Our procedure for testing this intersection is a two-step process. First, we check whether the centers of either of the ellipsoids lie within the other ellipsoid. If this is the case, the ellipsoids clearly intersect, and the test is completed. If this is not the case, we need to solve a constrained convex optimization problem to test for intersection. This optimization problem would also work without performing the first test, but clearly this would be computationally less efficient.

For the preliminary intersection test, we may simply check whether $E_1(\mathbf{c}_2) \leq 1$ is satisfied, and *mutatis mutandis*, for $\mathbf{c}_1$ and $E_2(\cdot)$. If either of these inequalities holds, the ellipsoids intersect and the test is completed.

Alternatively, if neither of these inequalities is satisfied, we need to perform a somewhat more involved intersection test. This procedure [18] works as follows. Without loss of generality, ellipsoid E_1 is transformed to the unit ball centered on the origin, and ellipsoid E_2 is transformed accordingly into an ellipsoid E_2'. We then find the point on E_2' that is closest to the origin, i.e., to the center of the transformed ellipsoid E_1'. If the distance of this closest point to the origin is less than one, it clearly lies within the unit ball, and hence, the ellipsoids must intersect. To find this closest point to the origin, E_2' is transformed into the unit ball centered on the origin, and the origin of the first transformed space is transformed accordingly. We then find the point on the unit ball that is closest to the transformed origin of the first transformed space. This last step involves solving a specific case of a Quadratically Constrained Quadratic Program (QCQP) that reduces to a convex optimization problem with a single linear inequality constraint [4].

We now discuss these steps in detail. By the assumption that $\mathbf{M}_j$ is symmetric positive definite, we may use the Cholesky decomposition $\mathbf{M}_j = \mathbf{L}_j \mathbf{L}_j^\top$ to rewrite Equation 26 as

$$\begin{aligned} E_j(\mathbf{x}) &= (\mathbf{x} - \mathbf{c}_j)^\top \mathbf{L}_j \mathbf{L}_j^\top (\mathbf{x} - \mathbf{c}_j) \\ &= \|\mathbf{L}_j^\top (\mathbf{x} - \mathbf{c}_j)\|^2 \leq 1. \end{aligned} \tag{27}$$

We are now in a position to transform one of the ellipsoids into the unit ball centered on the origin. Without loss of generality, we will transform E_1 in this way, and transform E_2 accordingly. We write E_1' and E_2' for the transformed ellipsoids. For a vector $\mathbf{x}$ in the original space, we write this vector in the transformed space as $\mathbf{x}'$, and let the transformation be given by

$$\mathbf{x}' \equiv \mathbf{L}_1^\top (\mathbf{x} - \mathbf{c}_1). \tag{28}$$

The transformed ellipsoid E_1' is defined by substitution of Equation 28 into Equation 27 so that we have

$$E_1'(\mathbf{x}') \equiv \mathbf{x}'^\top \mathbf{x}' \leq 1. \tag{29}$$

This transformation may be reversed through

$$\mathbf{x} = \mathbf{c}_1 + \mathbf{L}_1^{-\top} \mathbf{x}'. \tag{30}$$

We may find the transformed ellipsoid E_2' by noting that it is given by all transformed vectors $\mathbf{x}'$ that satisfy Equation 27. Moving the parameterization to the transformed space by substituting Equation 30 into Equation 27, we have

$$\begin{aligned} E_2'(\mathbf{x}') &\equiv \|\mathbf{L}_2^\top \left(\mathbf{c}_1 + \mathbf{L}_1^{-\top} \mathbf{x}' - \mathbf{c}_2\right)\|^2 \\ &= \|\mathbf{L}_2^\top \left(\mathbf{L}_1^{-\top} \mathbf{x}' - (\mathbf{c}_2 - \mathbf{c}_1)\right)\|^2 \\ &= \|\mathbf{L}_2^\top \left(\mathbf{L}_1^{-\top} \mathbf{x}' - \mathbf{L}_1^{-\top} \mathbf{L}_1^\top (\mathbf{c}_2 - \mathbf{c}_1)\right)\|^2 \\ &= \|\mathbf{L}_2^\top \mathbf{L}_1^{-\top} \left(\mathbf{x}' - \mathbf{L}_1^\top (\mathbf{c}_2 - \mathbf{c}_1)\right)\|^2 \\ &= \|\mathbf{L}'^\top_2 \left(\mathbf{x}' - \mathbf{c}_2'\right)\|^2 \leq 1. \end{aligned} \tag{31}$$

Here, we write the center $\mathbf{c}_2$ of E_2 in the transformed space as $\mathbf{c}_2'$. Furthermore, the transformed ellipsoid's matrix $\mathbf{M}_2'$ is obtained through the Cholesky decomposition $\mathbf{M}_2' = \mathbf{L}_2' \mathbf{L}'^\top_2$ where $\mathbf{L}_2' \equiv \mathbf{L}_1^{-1} \mathbf{L}_2$.

We can now formulate the intersection test as the question of whether there exists a point $\mathbf{x'}^*$ that satisfies both Equation 29 and Equation 31, that is, whether there exists a point $\mathbf{x'}^*$ that is inside both E_1' and E_2'. Because E_1' is the origin-centered unit ball in $\mathbf{x'}$-space, we can answer this question by finding the point $\mathbf{x'}^*$ inside E_2' that is closest to the origin. Clearly, we may write the origin of $\mathbf{x'}$-space as $\mathbf{c}_1'$. If this closest point satisfies $\mathbf{x'}^{*\top}\mathbf{x'}^* \leq 1$, it must be within E_1', and hence the ellipsoids intersect.

We can find the closest point $\mathbf{x'}^*$, inside E_2', to the origin by solving

$$\begin{aligned} &\text{Minimize} : (\mathbf{x'} - \mathbf{c}_1')^\top(\mathbf{x'} - \mathbf{c}_1') \\ &\text{Subject to} : E_2'(\mathbf{x'}) = \|\mathbf{L'}_2^\top(\mathbf{x'} - \mathbf{c}_2')\|^2 \leq 1. \end{aligned} \tag{32}$$

The first step in solving this minimization problem is to *again* transform the space, this time so that E_2' becomes the unit ball centered on the origin. We write E_2'' for this transformed ellipsoid, and refer to the transformed space as $\mathbf{x''}$-space. The transformation is given by

$$\mathbf{x''} \equiv \mathbf{L'}_2^\top(\mathbf{x'} - \mathbf{c}_2'), \tag{33}$$

and reversed through

$$\mathbf{x'} = \mathbf{c}_2' + \mathbf{L'}_2^{-\top}\mathbf{x''}. \tag{34}$$

Substituting Equation 34 into the minimization problem, we have

$$\begin{aligned} \text{Minimize} : &\left(\mathbf{c}_2' + \mathbf{L'}_2^{-\top}\mathbf{x''} - \mathbf{c}_1'\right)^\top \left(\mathbf{c}_2' + \mathbf{L'}_2^{-\top}\mathbf{x''} - \mathbf{c}_1'\right) \\ &= \mathbf{x''}^\top\mathbf{L'}_2^{-1}\mathbf{L'}_2^{-\top}\mathbf{x''} + 2(\mathbf{c}_2' - \mathbf{c}_1')^\top\mathbf{x''} \\ &\quad + (\mathbf{c}_2' - \mathbf{c}_1')^\top(\mathbf{c}_2' - \mathbf{c}_1') \\ &\propto \mathbf{x''}^\top\mathbf{L'}_2^{-1}\mathbf{L'}_2^{-\top}\mathbf{x''} + 2(\mathbf{c}_2' - \mathbf{c}_1')^\top\mathbf{x''}. \end{aligned}$$

Substituting Equation 33 into the minimization problem's constraint, the final problem becomes

$$\begin{aligned} &\text{Minimize} : \mathbf{x''}^\top\mathbf{L'}_2^{-1}\mathbf{L'}_2^{-\top}\mathbf{x''} + 2(\mathbf{c}_2' - \mathbf{c}_1')^\top\mathbf{x''} \\ &\text{Subject to} : \mathbf{x''}^\top\mathbf{x''} \leq 1. \end{aligned}$$

After we solve this problem, we can find $\mathbf{x'}^*$ by applying Equation 34 to the solution $\mathbf{x''}^*$, and check whether $\mathbf{x'}^{*\top}\mathbf{x'}^* \leq 1$ to test for intersection. It remains to show how to solve this minimization problem.

We first reparameterize the problem to clean up our notation. We write $\mathbf{z} \equiv \mathbf{x''}$, $\mathbf{A} \equiv \mathbf{L'}_2^{-1}\mathbf{L'}_2^{-\top}$ and $\mathbf{b} \equiv (\mathbf{c}_2' - \mathbf{c}_1')$. We then formulate the problem as an equivalent QCQP that is given by

$$\begin{aligned} &\text{Minimize} : \frac{1}{2}\mathbf{z}^\top\mathbf{A}\mathbf{z} + \mathbf{b}^\top\mathbf{z} \\ &\text{Subject to} : \frac{1}{2}\mathbf{z}^\top\mathbf{I}\mathbf{z} - \frac{1}{2} \leq 0. \end{aligned} \tag{35}$$

Because $\mathbf{A}$ is symmetric positive definite by the fact that its Cholesky decomposition $\mathbf{A} = \mathbf{L'}_2^{-1}\mathbf{L'}_2^{-\top}$ exists, we can solve this problem in its Lagrangian dual form [4]. We define the Lagrangian $L(\mathbf{z}, \lambda)$ with Lagrange multiplier λ as

$$L(\mathbf{z}, \lambda) \equiv \frac{1}{2}\mathbf{z}^\top\bar{\mathbf{A}}(\lambda)\mathbf{z} + \mathbf{b}^\top\mathbf{z} - \frac{\lambda}{2},$$

where

$$\bar{\mathbf{A}}(\lambda) \equiv \mathbf{A} + \lambda\mathbf{I}.$$

The Lagrangian dual $g(\lambda)$ is given by

$$g(\lambda) \equiv \inf_{\mathbf{z}} L(\mathbf{z}, \lambda), \tag{36}$$

which can be found by taking partial derivatives of $L(\mathbf{z}, \lambda)$ with respect to $\mathbf{z}$ and solving for $\mathbf{z}$. We have

$$\begin{aligned} \frac{\partial L(\mathbf{z}, \lambda)}{\partial \mathbf{z}} &= \frac{\partial}{\partial \mathbf{z}}\left[\frac{1}{2}\mathbf{z}^\top\bar{\mathbf{A}}(\lambda)\mathbf{z} + \mathbf{b}^\top\mathbf{z} - \frac{\lambda}{2}\right] \\ &= \frac{\partial}{\partial \mathbf{z}}\left[\frac{1}{2}\mathbf{z}^\top\bar{\mathbf{A}}(\lambda)\mathbf{z}\right] + \frac{\partial}{\partial \mathbf{z}}\left[\mathbf{b}^\top\mathbf{z}\right] \\ &= \bar{\mathbf{A}}(\lambda)\mathbf{z} + \mathbf{b}. \end{aligned}$$

Setting this derivative to zero, we obtain

$$\begin{aligned} \bar{\mathbf{A}}(\lambda)\mathbf{z} + \mathbf{b} &= \mathbf{0} \\ \mathbf{z} &= -\bar{\mathbf{A}}(\lambda)^{-1}\mathbf{b}. \end{aligned}$$

Substituting into Equation 36, we have

$$\begin{aligned} g(\lambda) = \inf_{\mathbf{z}} L(\mathbf{z}, \lambda) &= \inf_{\mathbf{z}}\left\{\frac{1}{2}\mathbf{z}^\top\bar{\mathbf{A}}(\lambda)\mathbf{z} + \mathbf{b}^\top\mathbf{z} - \frac{\lambda}{2}\right\} \\ &= \frac{1}{2}\left(-\bar{\mathbf{A}}(\lambda)^{-1}\mathbf{b}\right)^\top\bar{\mathbf{A}}(\lambda)\left(-\bar{\mathbf{A}}(\lambda)^{-1}\mathbf{b}\right) \\ &\quad + \mathbf{b}^\top\left(-\bar{\mathbf{A}}(\lambda)^{-1}\mathbf{b}\right) - \frac{\lambda}{2} \\ &= \frac{1}{2}\mathbf{b}^\top\bar{\mathbf{A}}(\lambda)^{-\top}\mathbf{b} - \mathbf{b}^\top\bar{\mathbf{A}}(\lambda)^{-1}\mathbf{b} - \frac{\lambda}{2} \\ &= -\frac{1}{2}\mathbf{b}^\top\bar{\mathbf{A}}(\lambda)^{-1}\mathbf{b} - \frac{\lambda}{2}. \end{aligned}$$

We can now find the optimal value of the Lagrange multiplier λ^* by solving the dual problem

$$\begin{aligned} &\text{Maximize} : -\frac{1}{2}\mathbf{b}^\top \bar{\mathbf{A}}(\lambda)^{-1}\mathbf{b} - \frac{\lambda}{2} \\ &\text{Subject to} : \lambda \geq 0. \end{aligned} \tag{37}$$

This is a single-variable convex optimization problem with a single linear constraint, and is thus readily solved with any number of numerical optimization routines. In case such numerical optimization routines require the specification of the gradient of $g(\lambda)$, we show

$$\begin{aligned} \frac{\partial g(\lambda)}{\partial \lambda} &= \frac{\partial}{\partial \lambda}\left[-\frac{1}{2}\mathbf{b}^\top \bar{\mathbf{A}}(\lambda)^{-1}\mathbf{b} - \frac{\lambda}{2}\right] \\ &= -\frac{1}{2}\mathbf{b}^\top \frac{\partial}{\partial \lambda}\left[\bar{\mathbf{A}}(\lambda)^{-1}\right]\mathbf{b} - \frac{\partial}{\partial \lambda}\left[\frac{\lambda}{2}\right] \\ &= -\frac{1}{2}\mathbf{b}^\top \frac{\partial}{\partial \lambda}\left[\bar{\mathbf{A}}(\lambda)^{-1}\right]\mathbf{b} - \frac{1}{2} \\ &= \frac{1}{2}\mathbf{b}^\top \bar{\mathbf{A}}(\lambda)^{-1}\frac{\partial}{\partial \lambda}\left[\bar{\mathbf{A}}(\lambda)\right]\bar{\mathbf{A}}(\lambda)^{-1}\mathbf{b} - \frac{1}{2} \\ &= \frac{1}{2}\mathbf{b}^\top \bar{\mathbf{A}}(\lambda)^{-1}\frac{\partial}{\partial \lambda}\left[(\mathbf{A} + \lambda\mathbf{I})\right]\bar{\mathbf{A}}(\lambda)^{-1}\mathbf{b} - \frac{1}{2} \\ &= \frac{1}{2}\mathbf{b}^\top \bar{\mathbf{A}}(\lambda)^{-1}\mathbf{I}\bar{\mathbf{A}}(\lambda)^{-1}\mathbf{b} - \frac{1}{2} \\ &= \frac{1}{2}\mathbf{b}^\top \bar{\mathbf{A}}(\lambda)^{-2}\mathbf{b} - \frac{1}{2}. \end{aligned}$$

Here, we have made use of the identity [17] $\frac{\partial \mathbf{Y}^{-1}}{\partial x} = -\mathbf{Y}^{-1}\frac{\partial \mathbf{Y}}{\partial x}\mathbf{Y}^{-1}$ for any matrix $\mathbf{Y}$. Using the same equality, we see that the second-order derivative of $g(\lambda)$ is given by

$$\begin{aligned} \frac{\partial^2 g(\lambda)}{\partial \lambda^2} &= -\mathbf{b}^\top \bar{\mathbf{A}}(\lambda)^{-1}\bar{\mathbf{A}}(\lambda)^{-1}\bar{\mathbf{A}}(\lambda)^{-1}\mathbf{b} \\ &= -\mathbf{b}^\top \bar{\mathbf{A}}(\lambda)^{-3}\mathbf{b}. \end{aligned}$$

Because $\bar{\mathbf{A}}(\lambda)$ is symmetric positive definite for $\lambda \geq 0$, we have that $\bar{\mathbf{A}}(\lambda)^{-3}$ is also symmetric positive definite for non-negative λ. It follows that for $\lambda \geq 0$, $\frac{\partial^2 g(\lambda)}{\partial \lambda^2}$ is negative everywhere, and $g(\lambda)$ is indeed concave.

In closing, by Slater's condition, strong duality holds [4] between Equation 37 and Equation 35. Thus, we can find the closest point $\mathbf{x}''^*$ on E_2'' to $\mathbf{c}_1''$ through

$$\mathbf{x}''^* = \mathbf{z}^* = -\bar{\mathbf{A}}(\lambda^*)^{-1}\mathbf{b}.$$

■

References

[1] P.M. Anglin, R. Gençay: *Semiparametric Estimation of a Hedonic Price Function* Journal of Applied Econometrics 11(6), pp. 633–648, 1996.

[2] K. Bache, M. Lichman: *UCI Machine Learning Repository* [http://archive.ics.uci.edu/ml], University of California, Irvine, School of Information and Computer Science, 2013.

[3] L. Breiman, J. Friedman, C. J. Stone, R. A. Olshen: *Classification and Regression Trees*, Chapman & Hall/CRC, 1984.

[4] S. Boyd, L. Vandenberghe: *Convex Optimization*, Cambridge University Press, 2004.

[5] R. D. Cook, S. Weisberg: *Residuals and Influence in Regression*, Chapman & Hall, London, 1982.

[6] M. Costanigro, R. C. Mittelhammer, J. J. McCluskey: *Estimating Class-Specific Parametric Models under Class Uncertainty: Local Polynomial Regression Clustering in an Hedonic Analysis of Wine Markets*, Journal of Applied Econometrics 24, 2009.

[7] J. Demšar: *Statistical Comparison of Classifiers over Multiple Data Sets*, Journal of Machine Learning Research, Volume 7, 2006.

[8] C. Dougherty: *Introduction to Econometrics (4th edition)*, Oxford University Press, 2011.

[9] W. Duivesteijn, A. Feelders, A. Knobbe: *Different Slopes for Different Folks: Mining for Exceptional Regression Models with Cook's Distance*, Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, 2012.

[10] J. H. Friedman, N. I. Fisher: *Bump hunting in high-dimensional data*, Statistics and Computing, Volume 9, Issue 2, 1999.

[11] W. H. Greene: *Econometric Analysis (7th Edition)*, Prentice Hall, 2011.

[12] Journal of Statistics Education Data Archive, [http://www.amstat.org/publications/jse/jse_data_archive.htm]

[13] W. Klösgen: *Explora: a multipattern and multistrategy discovery assistant*, Advances in Knowledge Discovery and Data Mining, 1996.

[14] W. Klösgen: *Subgroup Discovery*, Handbook of Data Mining and Knowledge Discovery, Ch. 16.3, Oxford University Press, 2002.

[15] M. van Leeuwen: *Maximal exceptions with minimal descriptions*, Data Mining and Knowledge Discovery, Volume 21, Issue 2, 2010.

[16] D. Leman, A. Feelders, A. Knobbe: *Exceptional Model Mining*, Proceedings of the ECML/PKDD'08, Volume 5212, 2008.

[17] K. B. Petersen, M. S. Pedersen: *The Matrix Cookbook*, Technical University of Denmark, 2008.

[18] S. B. Pope: *Algorithms for Ellipsoids*, Sibley School of Mechanical & Aerospace Engineering, Cornell University, Report: FDA-08-01, 2008.

[19] A. C. Schierz: *Virtual Screening of Bioassay Data*, Journal of Cheminformatics, 2009.

[20] C. Spearman: *The proof and measurement of association between two things.*, American journal of Psychology, Volume 15, Issue 1, 1904.