A Budget-Constrained Inverse Classification Framework for Smooth Classifiers

Conference Paper · November 2017

DOI: 10.1109/ICDMW.2017.174

4 authors:

Michael Lash
University of Kansas
14 PUBLICATIONS **99** CITATIONS
SEE PROFILE

Qihang Lin
University of Iowa
48 PUBLICATIONS **689** CITATIONS
SEE PROFILE

Nick Street
University of Iowa
133 PUBLICATIONS **6,172** CITATIONS
SEE PROFILE

Jennifer G Robinson
University of Iowa
424 PUBLICATIONS **26,360** CITATIONS
SEE PROFILE

A budget-constrained inverse classification framework for smooth classifiers

Michael T. Lash*, Qihang Lin†, W. Nick Street† and Jennifer G. Robinson‡
*Department of Computer Science, †Department of Management Sciences, ‡Department of Epidemiology
The University of Iowa
Iowa City, Iowa 52242
{michael-lash, qihang-lin, nick-street, jennifer-g-robinson}@uiowa.edu

***Abstract*—Inverse classification is the process of manipulating an instance such that it is more likely to conform to a specific class. Past methods that address such a problem have shortcomings. Greedy methods make changes that are overly radical, often relying on data that is strictly discrete. Other methods rely on certain data points, the presence of which cannot be guaranteed. In this paper we propose a general framework and method that overcomes these and other limitations. The formulation of our method uses any differentiable classification function. We demonstrate the method by using Gaussian kernel SVMs. We constrain the inverse classification to occur on features that can actually be changed, each of which incurs an individual cost. We further subject such changes to fall within a certain level of cumulative change (budget). Our framework can also accommodate the estimation of features whose values change as a consequence of actions taken (indirectly changeable features). Furthermore, we propose two methods for specifying feature-value ranges that result in different algorithmic behavior. We apply our method, and a proposed sensitivity analysis-based benchmark method, to two freely available datasets: Student Performance, from the UCI Machine Learning Repository and a real-world cardiovascular disease dataset. The results obtained demonstrate the validity and benefits of our framework and method.**

I. Introduction

In many predictive modeling problems, we are concerned less with the actual prediction, and more with how an individual prediction might be changed. Classification problems such as loan screening and college admission have one output class that is clearly "desired" by the test case. A person turned down for a loan would naturally wonder why the decision was made, and more importantly, what they could do to change the outcome on the next attempt. We use the term *inverse classification* to refer to the process of finding an optimal set of changes to a test point so as to maximize its predicted probability of the desired class label.

Such problems are particularly prevalent in personalized medicine. Consider the example of making lifestyle choices that minimize your long-term risk of cardiovascular disease (CVD). Risk prediction incorporates many factors including demographics (e.g. age, gender), lab measurements (e.g. BMI, triglyceride level), medications, and lifestyle (e.g. diet, exercise). Building an accurate predictor may require a complex nonlinear mapping from an individual's EHR data to CVD risk. Once an individual's risk has been predicted, we can work "backwards" through the classifier to obtain recommendations. We approach the recommendation step by defining an optimization problem: what is the smallest (or, easiest) set of feasible changes that the person can make in order to minimize the predicted probability of developing CVD?

In order to ensure that such recommendations are feasible in the real world, we must segment the predictive features into groups based on whether or not they are changeable. Clearly it is not useful to recommend that a person reduce their age by ten years. We therefore segment features into two categories: unchangeable and changeable features. Changeable features can be further segmented into two groups: directly changeable and indirectly changeable. Directly changeable features are those that are immediately actionable, like diet and exercise. Indirectly changeable features, on the other hand, are affected by the directly changeable features, but are not actionable themselves. Blood glucose is one example – one cannot directly alter the reading on a lab test, but they can change their eating habits, leading to changes in glucose levels.

Our method incorporates these important considerations. We impose individual, attribute-wise costs on the changeable features. Cumulative costs across such features are constrained to be within a budgetary level. We also impose feature-value limitations that restrict the amount that each feature can actually be changed. All of this is performed by minimizing, or maximizing, the score obtained from a trained classifier. In this work, we use SVMs. In practice, our formulation can accommodate any differentiable classifier. Additionally, we estimate the effect of direct actions taken on the indirectly changeable features.

Our approach makes several contributions to the inverse classification problem. First, we propose a new framework by which the inverse classification problem can be solved. The framework introduces a cost function on the recommended changes. Not all recommendations are equally easy; reducing fat intake, for example, might be easier for a person than giving up smoking. By assigning cost values to the feature modifications, we make the recommendations both more realistic and more individualized, since in practice a patient could supply their own costs. Furthermore, we subject cumulative changes and their associated costs to fall within a specified budget, which reflects an overall 'willingness' to change.

Secondly, we specify two bound-setting methods, Elastic

and Hard-line, that iteract with the proposed framework, allowing inverse classification to occur more freely or more rigidly, depending upon the problem.

Thirdly, we incorporate an indirect feature estimator, that adjusts features that change as a consequence of the directly alterable set of features.

Lastly, we propose a method of inverse classification that operates within the proposed framework and can be applied to any differentiable classifier with a Lipschitz-continuous gradient.

The rest of the paper is organized as follows. Section II reviews past works done on inverse classification. Section III describes our proposed framework and new method of inverse classification. Section IV relates two experiments conducted using both our proposed method and a sensitivity analysis-based benchmark method. These experiments are performed on two freely available datasets: a real-world cardiovascular disease dataset, and a benchmark dataset from the UCI Machine Learning Repository. Section V concludes the paper.

II. Related Work

Inverse classification can be seen as a form of sensitivity analysis, the process of examining the input features' effects on the target output. By adjusting the inputs to a model and examining the way the output changes, one can surmise the individual and cumulative effects that the features have. Sensitivity analysis is inherent to inverse classification, as the process involves changing the inputs in such a way that a desired outcome is observed.

Consider a trained logistic regression model. In the absence of multi-collinearity, the beta coefficients and corresponding p-values can be examined. These give an idea of which features are important to the target class. Features having coefficients with very low p-values are important to the predictive problem. The sign of the coefficient tells us whether a particular feature is indicative of the positive class, or the negative class.

There are more complex methods of performing sensitivity analysis. Such methods include variation of parameters, local sensitivity analysis, and domain-wide analysis [1], as well as equation, weight magnitude, and variable perturbation methods [2].

Of these methods, inverse classification is most similar to the variable perturbation method. This method calls for examining the changes in the output as the input values are changed. Inverse classification further extends variable perturbation by incorporating a notion of local sensitivity analysis. That is, the perturbations are centered about a particular test point.

Past works on inverse classification can be looked at from two distinct perspectives: the manner in which the algorithms operate – greedy [3, 4, 5, 6] and non-greedy [7, 8] – and the types of data they can handle – discrete [3, 4, 5] and continuous [6, 7, 8]. Greedy methods work by iteratively selecting attribute values that most increase or decrease some objective value. In [3] the authors select attribute values that maximize class discrimination (measured using gini index) in favor of the target class. In [4, 5] attributes are selected such that the probability of a negative outcome is minimized. The way in which these attributes are greedily selected also differs between the methods. Chi et al. and Yang et al. use a weighted *k*-nearest neighbor [9] approach to provide a set of viable discrete attribute values [4, 5]. Aggarwal et al. leverage their own data structure, called inverted lists, that indexes training instances and each instance's corresponding class by attribute and value. Such a method also relies on discretization [3].

Mannino and Koushik [6] use a genetic algorithm that constructs an artificial data point based on the training data and then attempt to minimize the distance between this point and a given test instance, weighted by the cost of creating such an artificial point. The cost is specified according to the attribute-wise changes that were made to create such a data point. This method works on continuously-valued attributes.

Greedy approaches tend to make the most radical changes possible. When constructing recommendations in applications such as health care, these methods may well make unrealistic, and therefore discouraging, suggestions. This issue is slightly overcome by [6] which incorporates attribute-specific costs. However, it fails to impose a limit or budget on the cumulative amount of cost that can be incurred. The majority of these methods (with the exception of [6]) also rely on discrete data. Such forced discretization leads to course-grained results; for example, reducing fat intake from "medium" to "low" is not a clear recommendation.

Non-greedy methods overcome these issues by focusing on moderate objectives instead of radical ones. In [8] Pendharkar uses data envelopment analysis (DEA) to find the efficient frontier of those training instances that take on the negative class (e.g., high risk). Test points that fall within such a frontier are then manipulated such that they are moved across it. Barbella et al. use a maximally separating surface, given by support vectors, their corresponding labels and alpha values, learned from the training data [7]. They attempt to minimize the squared Euclidean distance between a test point and a training point, requiring that such a training point fall exactly on the separating surface. The algorithm is referred to as border classification. While these methods are an improvement over those that are greedy, they have shortcomings. In the former case, a sufficient number of negative instances must exist in order to ensure that a suitable frontier is constructed. In the latter, there is no guarantee that any points fall on the separating surface itself, which would mean that no feasible points exist.

III. An Inverse Classification Framework and Method

In this section we propose a new inverse classification framework, and a method that can be used within the framework to solve the problem. We begin by generally discussing the problem and introducing some notation.

Suppose $\{(\mathbf{x}^i, y^i)\}_{i=1,2,\ldots,n}$ is a dataset of n instances where $\mathbf{x}^i \in \mathbb{R}^p$ is a column feature vector of length p and $y^i \in \{-1, 1\}$ is the binary label associated with $\mathbf{x}^i$ for $i = 1, 2, \ldots, n$. Let $X = [\mathbf{x}^1, ..., \mathbf{x}^n]^T \in \mathbb{R}^{n \times p}$ denote the matrix

of training instances with $(\mathbf{x}^i)^T$'s being its rows. Many classification models, including logistic regression, naive Bayes, support vector machines (SVMs), decision trees and neural networks, can be trained with this dataset and used to predict the class of new instances. Different from most of the existing literature, given a new instance $\mathbf{x} \in \mathbb{R}^p$, our goal is not only to classify it to the positive or the negative class but also to recommend an update on $\mathbf{x}$ that minimizes its probability of being classified as positive. We assume one unit change in each feature of $\mathbf{x}$ will incur a cost and that only a limited amount of budget B is available. We propose a numerical framework and algorithm that recommends an optimal change on $\mathbf{x}$ based on a classification model that incorporates this budgetary constraint.

A. Framework

Suppose we are allowed to change some of the features of instance $\mathbf{x}$ to obtain a new version $\mathbf{x}'$. Also suppose we want this change to minimize the probability of $\mathbf{x}'$ being classified as positive. With a classifier $f(\mathbf{x})$, such an $\mathbf{x}'$ can be obtained by minimizing $f(\mathbf{x})$ over the features of the new version $\mathbf{x}'$.

However, due to some physical or economical reasons, we cannot search for the optimal $\mathbf{x}$ over the whole feature space $\mathbb{R}^p$. In particular, we assume the features $\{1, 2, \ldots, p\}$ can be partitioned into two subsets C and U. Given a feature vector $\mathbf{x}$, let $\mathbf{x}_C$ and $\mathbf{x}_U$ represent the sub-vectors of $\mathbf{x}$ that contain only changeable and only unchangeable features, respectively. Since $\mathbf{x}_U$ cannot be changed, we will minimize $f(\mathbf{x})$ by optimizing $\mathbf{x}_C$. Hence, we represent $f(\mathbf{x})$ as $f(\mathbf{x}_U, \mathbf{x}_C)$ to distinguish these two sub-vectors. In addition, we assume the reasonable value of each changeable feature in C must be within an interval, denoted by $[l_i, u_i]$ for $i \in C$. Moreover, the costs for increasing and decreasing any feature x_i by one unit are denoted by c_i^+ and c_i^-, respectively. Give a limited budget B, the optimal feature design problem for a given instance $\mathbf{x}$ can be formulated as follows:

$$\begin{aligned} \min_{\mathbf{x}'_C \in \mathbb{R}^{|C|}} \quad & f(\mathbf{x}_U, \mathbf{x}'_C) \qquad (1) \\ \text{s.t.} \quad & \sum_{i \in C} c_i^+ (x'_i - x_i)_+ + c_i^- (x'_i - x_i)_- \leq B \\ & l_i \leq x'_i \leq u_i \text{ for } i \in C, \end{aligned}$$

where $(x)_+ = \max\{0, x\}$ and $(x)_- = \max\{0, -x\}$.

In a more general setting, some of the features in C can be changed directly by the designer. We call these features the directly changeable features. However, there are features that cannot be changed directly. Instead, they are changed as a consequence of the changes in the directly changeable features. We call these indirectly changeable features. In Chi et al. [4] the effects of the directly changeable on the indirectly changeable features are measured upon completion of the inverse classification process. Our method incorporates them as part of the optimization.

To model this phenomenon, we further partition the features in C into two subsets, D and I, which represent the sets of directly and indirectly changeable features, respectively. When we optimize the features, we can only determine the value for $\mathbf{x}_D$ and the values of $\mathbf{x}_I$ will depend on $\mathbf{x}_D$ and $\mathbf{x}_U$. Therefore, we model the dependency of $\mathbf{x}_I$ on $\mathbf{x}_D$ and $\mathbf{x}_U$ as $\mathbf{x}_I = H(\mathbf{x}_D, \mathbf{x}_U)$ where the mapping $H : \mathbb{R}^{|D|+|U|} \rightarrow \mathbb{R}^{|I|}$ is assumed to be smooth and differentiable. Note that the mapping H can be trained using the same training instances for $f(\mathbf{x})$. Therefore, we represent $f(\mathbf{x})$ as $f(\mathbf{x}_U, \mathbf{x}_I, \mathbf{x}_D)$ to distinguish these three blocks so that the feature optimization problem (1) can be generalized to

$$\begin{aligned} \min_{\mathbf{x}'_D \in \mathbb{R}^{|D|}} \quad & f(\mathbf{x}_U, H(\mathbf{x}'_D, \mathbf{x}_U), \mathbf{x}'_D) \qquad (2) \\ \text{s.t.} \quad & \sum_{i \in D} c_i^+ (x'_i - x_i)_+ + c_i^- (x'_i - x_i)_- \leq B \\ & l_i \leq x'_i \leq u_i \text{ for } i \in D. \end{aligned}$$

We outline two specific methods for solving $H(\mathbf{x}'_D, \mathbf{x}_U)$ in Section IV.A.3.

1) Hard-line and elastic bound-setting methods: The constraints in (1) and (2) are flexible enough to model different feature perturbation requirements. Specifically, there are two ways that the lower and upper bounds can be parameterized, each resulting in different algorithmic behavior.

The first is rigid with respect to test point $\mathbf{x}$'s original directly changeable values: if $c_i^- = 0$ then $l_i = x_i$, and if $c_i^+ = 0$ then $u_i = x_i$ where $i \in D$. Such box constraint parameterization prevents feature i from being increased without cost if $c_i^+ = 0$, or from being decreased without cost if $c_i^- = 0$, even if doing so would be beneficial according to the local function space, determined by $f(\mathbf{x})$. This allows for more control over the recommendations being made to individuals. Say, for example, that we have specific domain knowledge that indicates that it is always beneficial to increase feature $i \in D$ and that this increase comes at a cost $c_i^+ > 0$ while the decrease is cost free ($c_i^- = 0$). However, due to noisy data, the learned model believes that decreasing this feature is beneficial. As a result the algorithm will recommend decreasing x_i to an extreme value, e.g., $-\infty$. Setting $l_i = x_i$ prevents our algorithm from making such a change. We refer to this as the Hard-line bound-setting method.

The second is less rigid, allowing feature i to increase even if $c_i^+ = 0$, or to decrease even if $c_i^- = 0$. To obtain such behavior, if $c_i^+ = 0$ then $u_i = \max\{1, x_i\}$ and if $c_i^- = 0$ then $l_i = \min\{0, x_i\}$. We refer to this as the Elastic bound-setting method.

In practice, we acknowledge any combination of these bound-setting methods can be used in a feature-specific manner. Bounds and costs can also be imposed such that individuals costs are incurred differently, depending on whether a specific feature is increased or decreased.

B. Method

To solve the inverse classification problem, according to (1) and (2), we assume the objective function f is differentiable and its gradient is Lipschitz continuous. Under this assumption, if f is linear, the problem can be solved optimally and

efficiently. If, however, the objective function is highly non-linear and non-convex, finding the globally optimal solution is NP-hard, in general. Because we do no wish to make further assumptions about the linearity of f, we focus on methods that can solve both these and the harder non-linear, non-convex class of function.

The available techniques that can be applied to non-convex, constrained optimization problems (see [10, 11, 12] and extensive references therein) include: (a) deterministic approaches such as branch and bound [12, 13], function approximation [14], cutting plane methods [15], difference of convex functions methods [16]; and (b) stochastic approaches such as simulated annealing [17] and genetic algorithms [18]. However, these methods are typically slow and do not scale to large problems[1].

Therefore, our list of potential methods is left to include the projected/proximal gradient method [19, 20, 21, 22] and the zero-order method [21]. If $f(x)$ is second-order differentiable, the list of potential methods can be extended to include regularized Newton's method [23, 24], sequential quadratic programming [25], and BFGS [26]. Among these methods, the projected gradient method and the zero-order method can guarantee that the iterative solution converge to a stationary point at a rate of $O(\frac{1}{t})$. The rest of the methods only guarantee asymptotic convergence, with no specified convergence rate. Since the zero-order method is appropriate only when evaluating the gradient of f is difficult, which is not our case, the appropriate method to apply with good theoretical guarantees is the projected gradient method.

1) The Projected Gradient Method: Before we present the projected gradient method, we need to reformulate (1) or (2) using the difference of the original features and updated features as our decision variables. Because space is limited, we will only conduct the reformulation and present the algorithm for (2), but the same technique can be applied to (1). In (2), we define $\mathbf{z} = \mathbf{x}'_D - \mathbf{x}_D$ and, by changing variables, (2) can be equivalently written as

$$\min_{\mathbf{z}\in\Delta_D} \quad g(\mathbf{z}) \tag{3}$$

where $g(\mathbf{z}) \equiv f(\mathbf{x}_U, H(\mathbf{x}_D + \mathbf{z}, \mathbf{x}_U), \mathbf{x}_D + \mathbf{z})$,

$$\Delta_D \equiv \left\{ \mathbf{z} \in \mathbb{R}^{|D|} \;\middle|\; \begin{array}{c} \sum_{i\in D} c_i^+(z_i)_+ + c_i^-(z_i)_- \le B, \\ l'_i \le z_i \le u'_i \text{ for } i \in D. \end{array} \right\}, \tag{4}$$

$l'_i = l_i - x_i$ and $u'_i = u_i - x_i$ for $i \in D$. The projection mapping onto the set Δ_D is defined as

$$\mathbf{Proj}_{\Delta_D}(\mathbf{w}) \equiv \arg\min_{\mathbf{z}\in\Delta_D} \frac{1}{2}\|\mathbf{z} - \mathbf{w}\|^2. \tag{5}$$

When $g(\mathbf{z})$ is differentiable and its gradient $\nabla g(\mathbf{z})$ is L-Lipschitz continuous,[2] which is true for our class of function, the projected gradient method for solving (4) is then given as Algorithm 1.

[1]This fact is observed first-hand in conducting our own experiments; such an experience will be further elaborated on in Section IV.

[2] $\nabla g(\mathbf{z})$ is L-Lipschitz continuous if $\|\nabla g(\mathbf{z}) - \nabla g(\mathbf{z}')\| \le L\|\mathbf{z} - \mathbf{z}'\|$ for any $\mathbf{z}, \mathbf{z}' \in \mathbb{R}^{|D|}$.

Algorithm 1 Projected Gradient Method

Input: $\mathbf{z}^{(0)} \in \Delta_D$, $t = 0$ and $\eta > 0$
1: **while** Stopping criterion is not satisfied **do**
2: $\quad \mathbf{z}^{(t+1)} = \mathbf{Proj}_{\Delta_D}(\mathbf{z}^{(t)}) - \eta\nabla g(\mathbf{z}^{(t)})$
3: $\quad t \leftarrow t + 1$
4: **end while**
Output: $\mathbf{z}^{(t)}$

According to Theorem 3 of [20], when $\eta \le \frac{1}{L}$, Algorithm 1 guarantees that $\mathbf{z}^{(t)}$ converges to a stationary point (or so-called KKT point) of (3) in a speed of $O(\frac{1}{t})$, which is the best convergence for non-convex smooth optimization.

Algorithm 1 requires solving the projection $\mathbf{Proj}_{\Delta_D}(\mathbf{w})$ in each iteration, which is itself an optimization problem. An efficient scheme for solving this subproblem will be critical to achieve a good time efficiency of Algorithm 1. Fortunately, if the domain $\Delta_D \neq \emptyset$, it has a specific structure which allow us to solve $\mathbf{Proj}_{\Delta_D}(\mathbf{w})$ for any $\mathbf{w}$ with an efficient subroutine. To see this, we define

$$h_i(w,\lambda) = \begin{cases} w - \lambda c_i^+ & \text{if } \lambda \le \frac{w}{c_i^+} \text{ and } w > 0 \\ w + \lambda c_i^- & \text{if } \lambda \le -\frac{w}{c_i^-} \text{ and } w < 0 \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

for each $i \in D$. The subroutine is given in Algorithm 2.

Algorithm 2 Projection Mapping $\mathbf{Proj}_{\Delta_D}(\mathbf{w})$

Input: $\mathbf{w} \in \mathbb{R}^{|D|}$, $\{c_i^+\}_{i\in D}$, $\{c_i^-\}_{i\in D}$, $\{l'_i\}_{i\in D}$ and $\{u'_i\}_{i\in D}$
1: $\mathcal{A}_- \leftarrow \{i | u'_i \le \min(0, w_i)\}$
2: $\mathcal{A}_+ \leftarrow \{i | \max(0, w_i) \le l'_i\}$
3: $z_i \leftarrow u'_i$ for $i \in \mathcal{A}_-$ and $z_i \leftarrow l'_i$ for $i \in \mathcal{A}_+$
4: **if** $\sum_{i\in D\setminus(\mathcal{A}_+\cup\mathcal{A}_-)} \max\{\min\{h_i(w_i, 0), u'_i\}, l'_i\} \le B - \sum_{i\in\mathcal{A}_-} u'_i c_i^- - \sum_{i\in\mathcal{A}_+} l'_i c_i^+$ **then**
5: $\quad \lambda \leftarrow 0$
6: **else**
7: $\quad$ Apply bisection search to find $\lambda \in (0, +\infty)$ such that

$$\sum_{i\in D\setminus(\mathcal{A}_+\cup\mathcal{A}_-)} \max\{\min\{h_i(w_i, \lambda), u'_i\}, l'_i\} = B - \sum_{i\in\mathcal{A}_-} u'_i c_i^- - \sum_{i\in\mathcal{A}_+} l'_i c_i^+$$

8: **end if**
9: $z_i \leftarrow \max\{\min\{h_i(w_i, \lambda), u'_i\}, l'_i\}$ for $i \in D\setminus(\mathcal{A}_+ \cup \mathcal{A}_-)$
Output: $\mathbf{z}$

The correctness of Algorithm 2 is ensured by the following proposition whose proof is given in Appendix.

Proposition 1: If $\Delta_D \neq \emptyset$, the solution $\mathbf{z}$ returned by Algorithm 2 satisfies $\mathbf{z} = \mathbf{Proj}_{\Delta_D}(\mathbf{w})$.

IV. Experiments

In this section we outline our experimental methods and then apply such methods to two datasets. The first dataset is a benchmark dataset from the UCI Machine Learning Repository [27] called Student Performance [28] and the second is

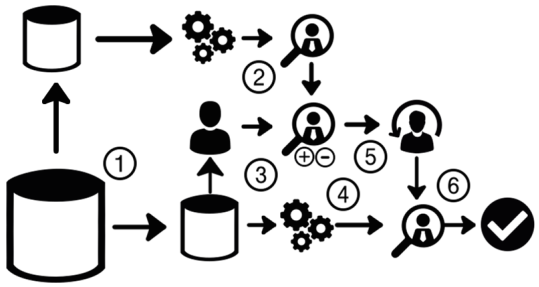


Fig. 1: The experiment process. (1) Data partitioning, (2) Parameter and model learning, (3) Inverse classification process on test instances (4) Learn a validation model, (5) Validate inv. class. test instances, (6) Assess results.[4]

from ARIC, the Atherosclerosis Risk in Communities study [29]. We emphasize that both datasets are publicly available. The latter requires explicit NIH permission[3]. Our experiments on the ARIC dataset, which mitigates patients' long-term risk of developing cardiovascular disease (CVD) is guided by one of the co-authors of this paper who is a CVD specialist.

A. Experimental Setup

In this section we outline a general process of validating inverse classification methods, the learning algorithm used to conduct the inverse classification, two methods for estimating indirectly changeable features, and the method we will use as benchmark to which our method can be compared.

1) Process: Our process of making and evaluating recommendations is based on that proposed by [4]. Essentially, we are attempting to 'go back in time' and make recommendations such that the probability of a known outcome is reduced. In other words, we are using data from the past in which known outcomes are observed. We then make recommendations that reduce the probability of a negative outcome occurring. But, in the absence of a time machine, we need a way to validate whether we have actually reduced the probability of a such an event occurring. Such a method requires careful segmentation of the data such that none of the information used to make recommendations is used in validating the probability of an outcome occurring. The process is related as follows:

Step 1 involves defining two different sets of data: a training set and a testing/validation set. These two sets were generated by splitting the initial full dataset into two equal parts. Data cleansing and preparation were also performed. This involves imputing missing values (mean) and normalizing the data values to be within $[0, 1]$.

Step 2 uses the training set to learn a model f. During this step cross-validation can be used to find the optimal parameters of f, if necessary. We also perform cross-validation to obtain optimal parameters in the model $x_I = H(x_D, x_U)$ for indirectly changeable features. These models are then used with (2) to perform the inverse classification that generates recommendations.

Step 3 involves partitioning the testing set such that a group of test instances are set aside. (2) is applied to these individuals about whom recommendations are obtained. The remaining test instances are used in Step 4.

Step 4: In this step the remaining instances from the second set of data are used to learn a separate validation model. As in Step 2, we learn a model f and $\mathbf{x}_I = H(\mathbf{x}_D, \mathbf{x}_U)$.

Steps 3 and 4 both employ 10-fold cross-validation, in which one fold is used as a set of test instances, and the remaining nine folds are used to train the validation model. In this manner, all of the second set of data can be used as test instances for which we obtain recommendations.

Step 5: Here we apply the recommendations made to each of the test instances in Step 3. For each test instance $\mathbf{x}$ and its modified counterpart $\mathbf{x}'$, we then evaluate $f(\mathbf{x}_U, H(\mathbf{x}'_D, \mathbf{x}_U), \mathbf{x}'_D)$ with f and H being the validation models from Step 4 and obtain a probability for $\mathbf{x}'$.

Step 6: Here we examine the probabilities obtained for each of the test instances $\mathbf{x}$ and $\mathbf{x}'$ to assess the validity of our method.

2) Inverse classification based on SVM: Among classification models, kernel SVM is one of the most widely used. Compared to the classical linear SVM, kernel SVM is more appropriate for data in which two classes of instances have a nonlinear boundary. A kernel SVM model can be trained using its dual formulation which is related by the optimization problem

$$\begin{aligned} \max_{\alpha \in \mathbb{R}^n} \quad & \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^i y^j k(\mathbf{x}^i, \mathbf{x}^j) \qquad (7) \\ \text{s.t.} \quad & \sum_{i=1}^{n} \alpha_i y^i = 0 \text{ and } 0 \leq \alpha_i \leq C \text{ for } i = 1, 2 \ldots, n, \end{aligned}$$

where $k(\mathbf{x}, \mathbf{x}') : \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is a kernel function that measures the similarity between any pair of instances x and x' in $\mathbb{R}^p$. The commonly used kernel functions include linear kernels $k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$, polynomial kernels $k(\mathbf{x}, \mathbf{x}') = (1 + x^T x')^d$ for any positive integer d, and Gaussian kernels $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$ for $\sigma > 0$ where $\|\cdot\|$ represents the Euclidean norm in $\mathbb{R}^p$.

Suppose the optimal solution of (7) is $\alpha^* \in \mathbb{R}^n$. An SVM classifier can be derived based on the function[5]

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y^i k(\mathbf{x}^i, \mathbf{x}), \qquad (8)$$

where the instance $\mathbf{x}_i$ with $\alpha_i^* > 0$ is called a support vector. Given a new instance $\mathbf{x}$, the value of $f(\mathbf{x})$ represents how similar $\mathbf{x}$ is to the positive class. A larger value of $f(\mathbf{x})$ means that $\mathbf{x}$ is more likely to be positive.

[3]Obtained via BioLINCC.

[4]Graphics courtesy of: flaticon.com

[5]In fact, the exact kernel SVM classifier is $f_b(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y^i k(\mathbf{x}^i, \mathbf{x}) + b$ where b is an offset value such that the new instance $\mathbf{x}$ is classified to be positive if $f_b(\mathbf{x}) > 0$ and to be negative otherwise.

However, the scores obtained from $f(\mathbf{x})$ do not correspond to likelihood directly. Therefore, we apply Platt's Method [30]. Platt's Method transforms the scores obtained from applying $f(\mathbf{x})$ to probabilities; specifically, the probability of being positive. By applying this method we learn a probability space that is more easily interpretable.

We elect to use the Gaussian kernel SVM in both sets of experiments. This is for three reasons. The first is that such a function is highly nonlinear and complex, giving us the opportunity to explore a more extreme classifier by which we can assess the effectiveness of our method. Secondly, the Gaussian kernel can be used to assess point similarity. This is beneficial in our experiments as one of our assumptions is that similar points will have similar probabilities associated with them. Finally, using the σ parameter, we can control the size of the neighborhood used to assess point similarity. That is, the larger the σ the more similar distant support vectors are considered to be to test point $\mathbf{x}$.

Therefore, our objective function, outlined in (1) and (2), becomes (8) with features segmented into appropriate groups and the indirect feature estimator, outlined in the next subsection, incorporated. We explicitly note that the minimization task is not to minimize the learned margin, but to minimize the SVM score. More appropriately, by applying Platt's method, we will be minimizing probability directly.

3) Estimating Indirectly Changeable Features: In our experiments we propose two different models of $\mathbf{x}_I = H(\mathbf{x}'_D, \mathbf{x}_U)$ for estimating indirectly changeable features. The first one is the linear LASSO [31] model, learned using the training dataset. In particular, for each indirectly changeable feature $j \in I$, we solve

$$\min_{\beta_j \in \mathbb{R}^{|U|+|D|+1}} \quad \frac{1}{2} \left\| \mathbf{x}_j - [\mathbf{1}, X_{U,D}]\beta_j \right\|^2 + \lambda \left\| \beta_j \right\|_1 \quad (9)$$

where $\mathbf{x}_j \in \mathbb{R}^n$ represents the column of feature j in the training dataset and $[\mathbf{1}, X_{U,D}]$ is the matrix of the unchangeable and directly changeable features of the training instances with an all-one column inserted to account for the offset value. The regularization parameter λ is selected based on cross-validation. The ℓ_1-norm regularization in (9) will enforce sparsity in the solution β_j. After solving (9), we build a linear model $x_j = [1, \mathbf{x}_U^T, \mathbf{x}_D^T]\beta_j$ by only retraining the non-zero components of β_j from (9).

Applying this procedure to each indirectly changeable feature $j \in I$ results in a coefficient matrix $B \in \mathbb{R}^{(|U|+|D|+1)\times|I|}$ with $B = [\beta_j]\ j \in I$. Hence, the model $\mathbf{x}_I = H(\mathbf{x}_D, \mathbf{x}_U)$ in (2) becomes $\mathbf{x}_I = B^T[1, \mathbf{x}_U^T, \mathbf{x}_D^T]^T$.

The second model we propose to use is based on Kernel Regression [32, 33]. In particular, the model $\mathbf{x}_I = H(\mathbf{x}_D, \mathbf{x}_U)$ used in (2) is

$$\mathbf{x}_I \quad = \quad \frac{\sum_{i=1}^{n} k([\mathbf{x}_D^i, \mathbf{x}_U^i], [\mathbf{x}_D, \mathbf{x}_U])\mathbf{x}_I^i}{\sum_{i=1}^{n} k([\mathbf{x}_D^i, \mathbf{x}_U^i], [\mathbf{x}_D, \mathbf{x}_U])}, \qquad (10)$$

where the kernel $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$ (Gaussian) and the value $\sigma > 0$ is selected based on cross-validation. By using the model in (10) with the Gaussian kernel we are provided with the added benefit of a point similarity assessment in making estimations. The model works by considering the known $\mathbf{x}_I^i$, that are closer to $\mathbf{x}$, more favorably than those that are further away. In so doing, (10) obtains an estimate for $\mathbf{x}_I$ based on points that are most similar to it.

4) Methodological Benchmark: We compare our method of solving the inverse classification problem with that of a sensitivity analysis-derived variable perturbation method (which we refer to as 'benchmark method' for the sake of brevity) [2], which we incorporate into our framework. This method operates by iteratively perturbing each feature $\mathbf{x}_{D_i}$ $i \in D$ to the bounds of feasibility. The objective function is then evaluated. If this value is found to be better than any of the previous single-feature perturbations, the perturbation is accepted. After making single-feature perturbations, if some amount of budget B remains, then subsequent rounds of perturbations occur (double-feature perturbation, triple-feature perturbation, etc.).

The reason we use such a method for comparison is because all past methodological works (outlined in Section II) are unable to accommodate one, if not many, aspects of our specified framework. As an example, in [6], feasibility equivocates to a definitive "desirable class" classification, which will sometimes lead to infeasible solutions in our framework.

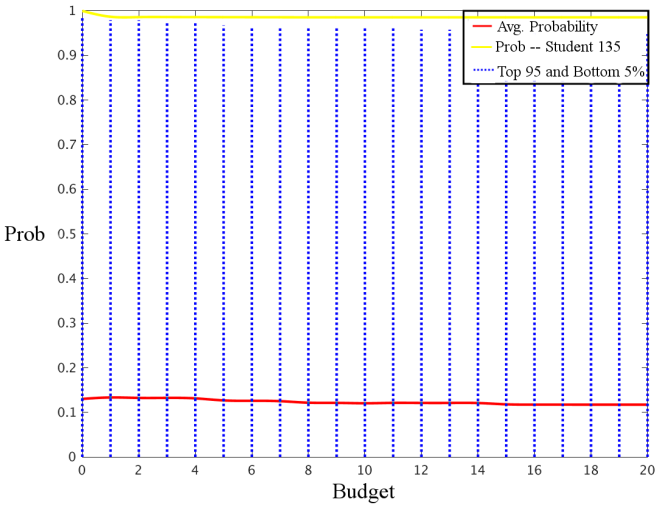
We also attempted to compare our method against the global solver BARON [34], using the simpler objective function in (1), on the CVD dataset. This solver, however, was unable to find a single solution after a runtime > 48 hours. Therefore, such a method was not further explored.

B. Grade-improving Recommendations

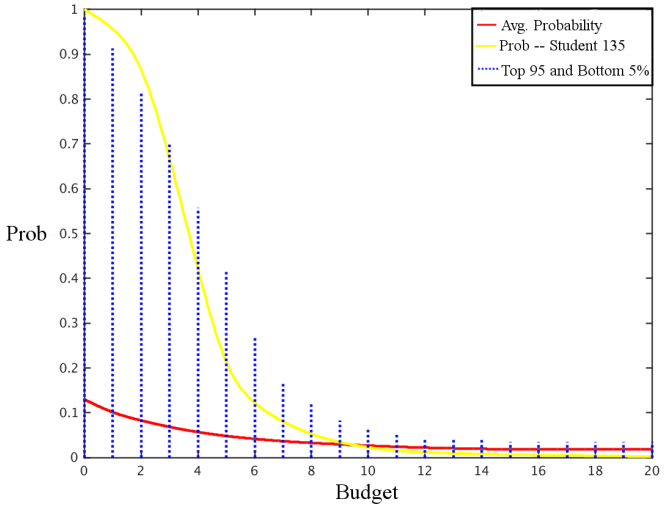
To validate the effectiveness of our method we selected the Student Performance dataset from the UCI Machine Learning repository. We will first describe this dataset and then discuss the results obtained from applying our experimental methodology outlined by Figure 1. On this dataset we use both methods of indirect feature estimation, and both the Hard-line and Elastic bound-setting methods. We compare our method of inverse classification against that outlined in Section IV.A.5.

1) Data Description: These data consist of individual Portuguese students enrolled in two different classes: a math class and a Portuguese language class. These are represented as two different, but overlapping, datasets. The one used in this experiment was the Portuguese language class, as it contained the greater number of instances ($n = 649$).

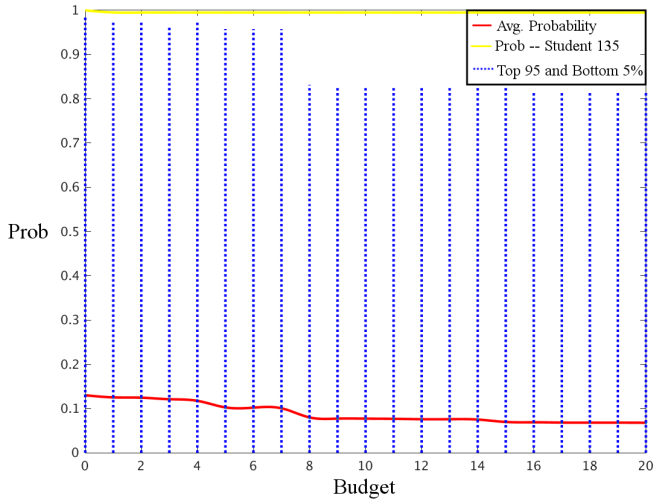
Each student-instance has 45 associated features, including a unique identifier and class variable. Therefore, $p = 43$. These 43 features are either ordinally- or nominally-valued. The nominally-valued features were binarized for this experiment. The variables pertain to several categories of each student's life. There are those that capture parental information (e.g., mother and father education), home-life information (e.g., have internet), personal social information (e.g., amount of time student goes out during the week), and school information (e.g., absences from school).



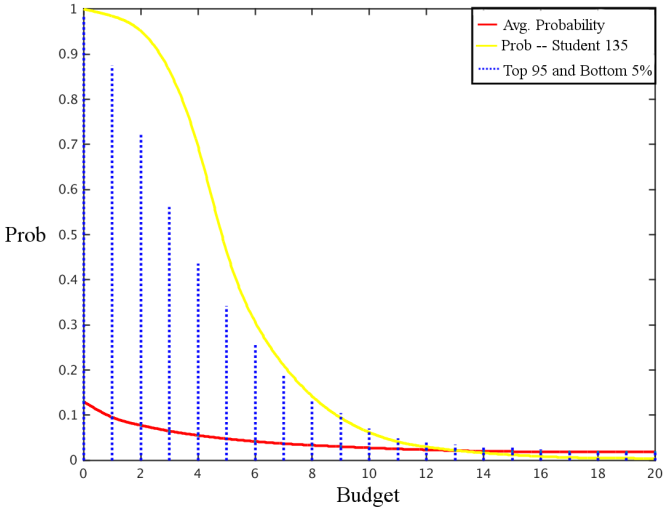
(a) Benchmark Method – Linear Indirect Estimator



(b) Our Method – Linear Indirect Estimator



(c) Benchmark Method – Kernel Indirect Estimator



(d) Our Method – Kernel Indirect Estimator

Fig. 2: The probability of earning a poor grade vs. budget for both Hard-line and Elastic bound-setting methods.

The dependent variable is the student's grade in the class, represented as a continuous numerical value. Cortez and Silva originally provided three different grade updates, representing the student's grade one and two thirds of the way through the class, as well as the final grade [28]. We use the final grade as the dependent variable, and discard the other two. We then turn this continuous variable into a two-class variable. We define "1" to be a grade of C or below and "-1" to be a grade of B or above, using appropriate numerical cutoffs [35]. Because "1" is a lower grade, our task will be to make recommendations that minimize this probability.

We then categorize each variable as being either unchangeable, indirectly changeable, or changeable. The latter class of variable, as discussed, are those for which we can make recommendations. We also specify individual costs to change each of these, and include them as non-zero values in either c^+ or c^-, which dictates whether increasing or decreasing the variable incurs a cost. These include "weekday alcohol consumption" ($c^-_{\text{WkAlco}} = 3$) and "weekend alcohol consumption" ($c^-_{\text{WendAlco}} = 6$). The full set of these can be viewed in Table I in the Appendix.

We also specify several indirectly changeable features. These include a desire to pursue higher education and the amount of free time the student has.

2) Results: Our first set of results relate how, by linearly increasing the budget level B, we can continue to decrease the probability of a low grade.

Figure 2 shows the results of this budgetary increase. The red line shows the average probability among the 324 test instances. The dotted blue lines show the top 95th and bottom 5th percentile of probabilities. The yellow line shows the probability of "Student 135", a randomly selected student who originally earned a grade of C or less (known positive test instance). We will show the recommendations made for this

individual shortly.

We only report four figures for our eight experiments – linear-based indirect feature estimator with the Hard-line bound-setting method for both our method and the benchmark method, linear-based indirect feature estimator with Elastic bound-setting method for both our method and the benchmark method, kernel-based indirect feature estimator with Hard-line bound-setting method for both our method and the benchmark method, and kernel-based indirect feature estimator with Elastic bound-setting method for both our method and the benchmark method – because the results of both bound-setting methods were the same. This occurred because our algorithm found no benefit to making recommendations that went in the opposite direction of those that incurred costs, nor did the benchmark method.

Figures 2a and 2b show the results of the benchmark method and our method,respectively, using the linear indirect feature estimator. As is easily observed, the benchmark method (Figure 2a) was entirely unsuccessful in reducing the probability of a poor grade for both the test set as a whole and Student 135. Our method rapidly rapidly decreased both probabilities.

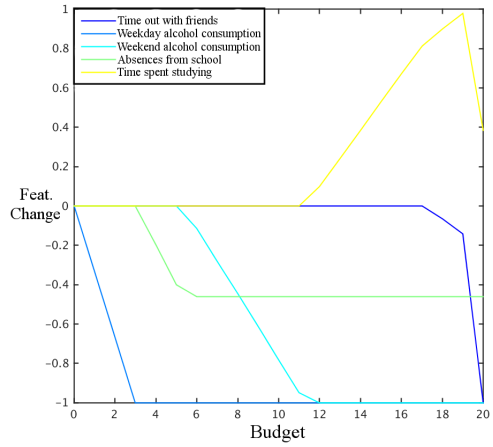
Figures 2c and 2d show the results of the benchmark method and our method, respectively, using the kernel-based indirect feature estimator. Similar to that of the linear indirect feature estimator, the benchmark method (Figure 2c) was unsuccessful in reducing the probability of a poor grade. Here, our method was again rapidly able to reduce both probabilities. It can be noted, however, that the benchmark method was able to produce a slight decrease in average probability as the budget constraint was incremented. The top 95% of the data at a budget level of eight also saw a decrease in probability, if only slightly. The probability of Student 135 was unmoved, however.

In both cases, our method performed substantially better than that of the benchmark. Therefore, the remainder of the discussion and results will focus on our method.

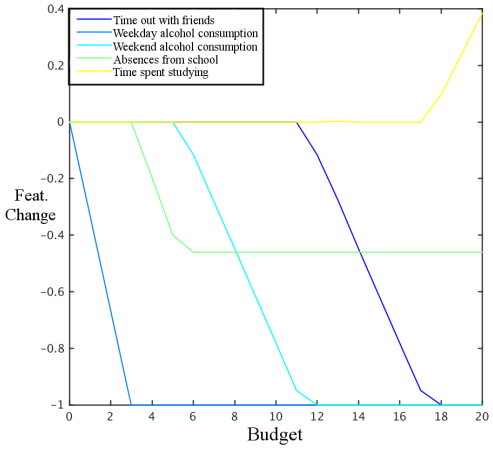
There are differences between the two indirect feature-estimators when employing our method. Overall, the kernel method more quickly reduced the probability of individuals falling within the top 95th percentile (Figure 2d) vs. that of the linear method (Figure 2b). In the average case, the methods performed very similarly. However, the linear method more quickly decreased the probability for "Student 135". This suggests that there are instances in which it is more beneficial to use the linear method, and instances in which it may be more beneficial to use the kernel-based method. For this problem and dataset, it seems that the more effective method is the linear-based estimator.

We now examine the recommendations made for "Student 135" over these 20 budgetary constraints using our method. Figure 3a shows the recommendations made using the linear-based indirect feature estimator and Figure 3b shows the recommendations made using the kernel-based indirect feature estimator.

The two indirect feature estimation methods have resulted in slightly different recommendations being made. From budget 1 to 12 the recommendations made are virtually the same: first daily alcohol consumption is reduced as much as possible, then the student is recommended to attend class more often, before being recommended to curb their weekend drinking. From here the algorithms traded off between recommending more studying and less time out with friends.



(a) The linear-based indirectly changeable feature estimator.



(b) The kernel-based indirectly changeable feature estimator.

Fig. 3: Recommended changes for Student 135 vs. budget for both Hard-line and Elastic bound-setting methods.

C. Cardiovascular Disease-mitigating Lifestyle Recommendations

We now apply our method to a real-world patient dataset called ARIC, initially comparing the result to those obtained using the benchmark method. We follow the same experimental methodology outlined previously, but with a more specific focus. Specifically, we refine our experiments to use only the kernel-based method of indirect feature estimation, along with the Hard-line bound-setting method. We do this for the same reasons that we elected to use the Gaussian kernel in the optimization. This set of experiments is also guided by a cardiovascular disease (CVD) specialist. It is because of such expertise that we have elected to use only the Hard-line bound-setting method, over that of the more permissive Elastic

bound-setting method. The goal of these experiments is to make recommendations that optimally reduce the probability of patients developing CVD over the next 10 years. Similar to the previous experiments we will first provide a description of the dataset and then discuss the results.

1) Data Description: The ARIC dataset contains patient information. This patient data was collected as part of a large prospective epidemiological study. The study began in 1987, recruiting individuals from four different communities. A sample of approximately 4000 patients, ranging in age from 45 to 64, was taken from each. The study began with an initial exam. During this exam patients were asked to fill out a lifestyle survey, which asked how much and how frequently certain foods were consumed, and how often patients exercised, for instance. Measurements were also taken. These included blood samples to record molecular-level information, such as blood glucose, as well as height and weight measurements. Demographic information was also collected. Proceeding the initial exam, follow-ups examinations occurred on a semi-annual basis.

Our dataset contains the patient information collected at the initial exam in 1987. Our class variable is defined based on certain cardiovascular disease events that occurred or did not occur over the following 10 years. Specifically, if a patient had one or more of the following diagnoses, then his or her binary 'CVD' class variable was encoded as 1: probable myocardial infarction (MI), definite MI, suspect MI, definite fatal coronary heart disease (CHD), possible fatal CHD, and stroke. Patients not having any of these diagnoses have their CVD class variable encoded as -1.

Our dataset contains 12907 patients, and excludes those individuals in the ARIC study who had one of the aforementioned diagnoses at the beginning of the study. Each individual has 110 features recorded. These include demographics, such as age and race; lifestyle, such as number of exercise hours and cigarettes smoked per day; measurements that are easy to take, such as height and weight; and molecular-level measurements that are taken from a blood sample, such as sodium and potassium levels.

We define each of the 110 features to be either unchangeable, directly changeable, or indirectly changeable. Demographic features fall entirely in the unchangeable category, for example. Measurement features fall almost exclusively in the indirectly changeable feature category, except features such as height. There are 61 such indirectly changeable features. The features that we include in the directly changeable category are those that pertain to lifestyle, such as diet and exercise. For each of the changeable features our CVD specialist parameterized an associated cost to change, also specifying whether each feature cost belonged to the c^+ or c^- vector. The full list of these can be viewed in Table II located in the Appendix.

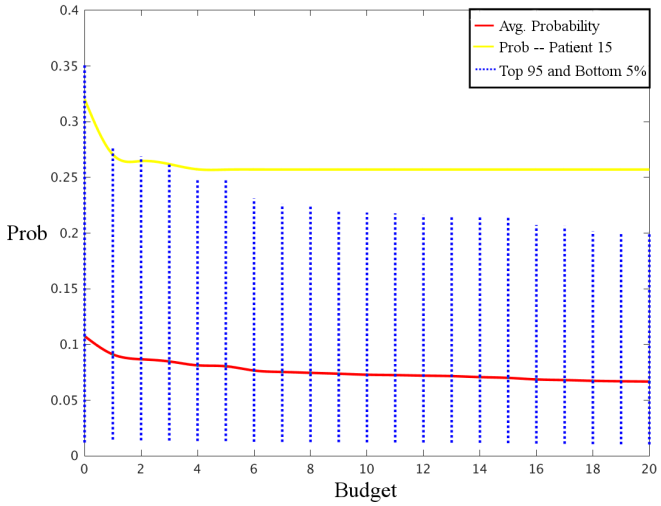
Our CVD specialist also recommended we impose special caveats on two of the directly changeable features: alcohol intake and exercise hours per week. According to our specialist, both alcohol and exercise are best in moderation. We take a somewhat naive approach in determining what these moderate values are. On the training data, for each of the variables, we created 10 equally-spaced bins. Each training point was a member of one bin. We then compute the probability of CVD for each bin based on the number of people who had a CVD event occur over the population in that bin. We then selected the bin with the lowest probability of CVD, defining its center to be the ideal value. Therefore, if a test patient was found to be below this value, then the upper bound u_i was set equal to it. If, on the other hand, the patient was found to have a value above this, then the lower bound l_i was set equal to it. These values were found to be 9.55 exercise hours per week and 66.29 grams of alcohol per week.

2) Results: Figure 4 shows the decrease in probability as budget increases. The red line shows the average probability of CVD among the 6454 test patients. The blue line shows the CVD probability of the top 95 and bottom 5% of test patients. The yellow line shows the CVD probability of a randomly selected test patient who ended up having a CVD event (known positive test instance). We refer to this patient as "Patient 15".

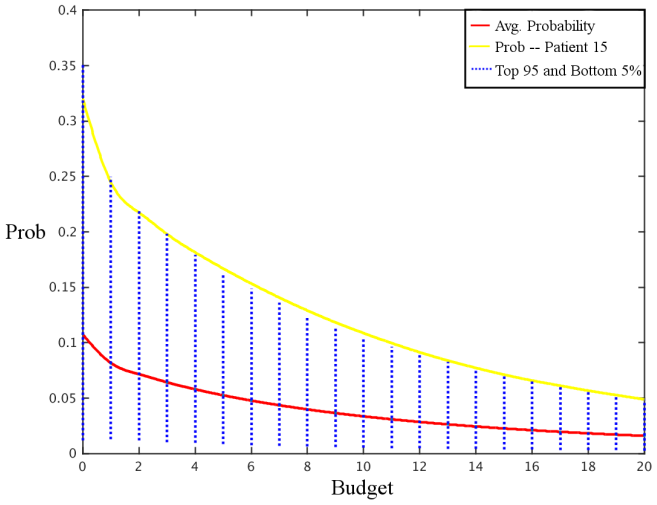
Figures 4a and 4b show the results of the benchmark method and our method, respectively. As we can observe, the benchmark method (Figure 4a) was somewhat successful in initially reducing the average probability, and that of Patient 15. The top 95% of the data also decreases as budget is increased. The probability of Patient 15%, however, plateaus shortly after a budget of one. By comparison, our method (Figure 4b) was far more successful in reducing the probability of CVD at each budget constraint in both the average case and for "Patient 15". The reduction in probability of those patients who fall in the top 95%, including "Patient 15" is reduced drastically as budget increases. The average case decreases at a slower rate. This makes sense, as we would expect to see diminishing returns for healthier individuals. Of additional note is the large initial decrease in probability when $B = 1$. This effect is observed in the average case and in the case of "Patient 15".

We will now examine the changes recommended to "Patient 15" over the 20 budgetary constraints using our method. These can be observed in Figure 5. At a budgetary constraint of one, our algorithm has decided that the most beneficial and cost-effective thing to do is to eat more dark or grain breads. From here, at $B = 2$, our method trades consuming more bread for decreasing the number of cigarettes smoked. At a budget of three, our algorithm recommends increasing the intake of fruits and vegetables. Fruits and vegetables are increased until a budgetary constraint of 8. Interestingly, at a budget of eight, and again at a budget 11, our method trades on and off decreasing sodium intake, while continuing to increase the consumption of fruits and vegetables. At a budget of 13, however, sodium begins to become a more dominant recommendation. At a budget of 16, dietary fiber is recommended to increase as the budget continues to go up. The last recommendation made is to slightly increase the consumption of nuts.

These changes across different budgetary constraints



(a) Benchmark method



(b) Our method

Fig. 4: CVD probability vs budget using the kernel-based indirectly changeable feature estimator and the Hard-line bound-setting method.

demonstrate the power of our method. Recommendations depend highly on the amount of effort an individual is willing to exert. Things like smoking and the consumption of fruits and vegetables are consistently recommended, given certain budget levels. Others, like sodium, depend highly on the level of effort.

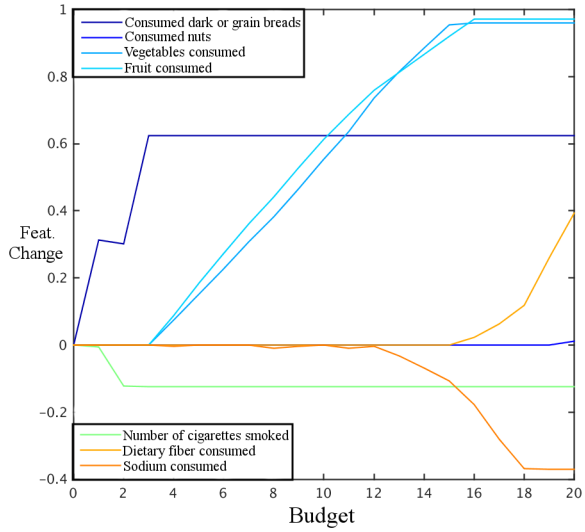


Fig. 5: The recommendations made to Patient 15 using the kernel-based indirectly changeable feature estimator and the Hard-line bound-setting method.

V. Conclusions and Future Work

In this work we propose and validate a new framework and method for inverse classification. These overcome the shortcomings of past works in a number of ways. First, our framework accounts for the fact that there are features that cannot be acted upon, features that can be directly acted upon, and features that are in-actionable, but are indirectly affected by the direct actions that can be taken. We incorporate the various costs needed to take different actions, as well as a threshold amount that individuals are willing to cumulatively change. Imposing such costs and budgetary constraints avoids overly recommendations while still moving the individual in an optimal direction. Our framework makes use of features of all types (discrete and continuous), and is highly modular.

This modularity is expressed in four different ways. The first is the number of model choices available. While we use SVMs, which themselves express modularity through the different kernel options available (Gaussian, polynomial, linear), in practice any differentiable classifier can be used. These include, but aren't limited to: artificial neural networks and logistic regression. The second is feature segmentation: the user can decide how to segment features and whether or not to incorporate indirectly changeable feature estimation into their optimization. Third, the user can select virtually any learning algorithm to estimate indirectly changeable features. Finally, the user can decide which bound-setting method is appropriate for their problem.

We demonstrated the efficacy of our method, as compared to a sensitivity analysis-based method, on two datasets covering two different domains: ten year cardiovascular disease-mitigating lifestyle recommendations and grade-improving recommendations on a dataset from the UCI Machine Learning Repository. On both datasets we were able to reduce the probability of negative outcomes using an innovative validation method. The sensitivity analysis-based benchmark method was unable to achieve even comparable performance. The recommendations that achieved these results were reported over differing levels of cumulative willingness to change

(budget). Such results demonstrate that our algorithm can be tailored to the individual, and makes recommendations that reflect what is most optimal and cost effective.

There are numerous directions for future work. Recommendations that lead to a test point being re-classified are not made instantaneously, but rather over some period of time. We would like to incorporate these temporal changes into our model. Treatment plans that suggest changes in temporal increments leading to optimal values is an additional direction of future work. We would also like to incorporate sparsity constraints that lead to more sparse recommendations, as well as formulations that can be used with and overcome noisy data.

There are also areas of future work pertaining to the application of such a method. In addition to cardiovascular disease-mitigating lifestyle recommendations, we would also like to make recommendations about medications, such as statins. Such a problem is inherently difficult to address as medication-takers are often associated with negative outcomes. Future work will also apply this algorithm to recommending optimal cancer treatment plans.

REFERENCES

[1] S. S. Isukapalli, *Uncertainty Analysis of Transport-transformation Models*. PhD thesis, Citeseer, 1999.

[2] J. Yao, "Sensitivity analysis for data mining," in *Fuzzy Information Processing Society, 2003. NAFIPS 2003. 22nd International Conference of the North American*, pp. 272–277, July 2003.

[3] C. C. Aggarwal, C. Chen, and J. Han, "The inverse classification problem," *Journal of Computer Science and Technology*, vol. 25, no. May, pp. 458–468, 2010.

[4] C. L. Chi, W. N. Street, J. G. Robinson, and M. A. Crawford, "Individualized patient-centered lifestyle recommendations: An expert system for communicating patient specific cardiovascular risk information and prioritizing lifestyle options," *Journal of Biomedical Informatics*, vol. 45, no. 6, pp. 1164–1174, 2012.

[5] C. Yang, W. N. Street, and J. G. Robinson, "10-year CVD risk prediction and minimization via inverse classification," in *Proceedings of the 2nd ACM SIGHIT symposium on International health informatics - IHI '12*, pp. 603–610, 2012.

[6] M. V. Mannino and M. V. Koushik, "The cost minimizing inverse classification problem : A genetic algorithm approach," *Decision Support Systems*, vol. 29, no. 3, pp. 283–300, 2000.

[7] D. Barbella, S. Benzaid, J. Christensen, B. Jackson, X. V. Qin, and D. Musicant, "Understanding support vector machine classifications via a recommender system-like approach," in *Proceedings of the International Conference on Data Mining*, pp. 305–11, 2009.

[8] P. C. Pendharkar, "A potential use of data envelopment analysis for the inverse classification problem," *Omega*, vol. 30, no. 3, pp. 243–248, 2002.

[9] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on*, vol. 13, no. 1, pp. 21–27, 1967.

[10] R. Horst and P. M. Pardalos, eds., *Handbook of Global Optimization (Volumne 1)*. Springer, 1995.

[11] P. M. Pardalos and H. E. Romeijn, eds., *Handbook of Global Optimization (Volume 2)*. Springer, 2002.

[12] A. Neumaier, "Complete search in continuous global optimization and constraint satisfaction," *Acta Numerica,*, vol. 13, pp. 271–369, 2004.

[13] I. P. Androulakis, C. D. Maranas, and C. A. Floudas, "A global optimization method for general constrained nonconvex problems," *Journal of Global Optimization*, vol. 7(4), pp. 337–363, 1995.

[14] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, vol. 21, pp. 345–383, Dec. 2001.

[15] H. Tuy, T. V. Thieu, and N. Q. Thai, "A conical algorithm for globally minimizing a concave function over a closed convex set," *Mathematics of Operations Research*, vol. 10, pp. 498–514, 1985.

[16] H. Tuy, "Global minimization of a difference of two convex functions," *Mathematical Programming Studies*, vol. 30, pp. 150–182, 2009.

[17] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, pp. 671–680, 1983.

[18] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1989.

[19] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Boston: Kluwer, 2004.

[20] Y. Nesterov, "Gradient methods for minimizing composite objective function," *Mathematical Programming, Series B*, vol. 140, pp. 125–161, 2013.

[21] S. Ghadimi and G. Lan, "Stochastic first- and zeroth-order methods for nonconvex stochastic programming," *SIAM Journal on Optimization*, vol. 23, pp. 2341–2368, 2013.

[22] S. Ghadimi, G. Lan, and H. Zhang, "Mini-batch stochastic approximation methods for nonconvex stochastic composite optimization," *Mathematical Programming*, vol. 155, pp. 267–305, 2014.

[23] C. Cartis, N. I. M. Gould, and P. L. Toint, "On the complexity of steepest descent, Newton's and regularized Newton's methods for nonconvex unconstrained optimization problems," *SIAM Journal on Optimization*, vol. 20, pp. 2833–2852, 2010.

[24] N. I. M. Gould and P. L. Toint, "An adaptive cubic regularization algorithm for nonconvex optimization with convex constraints and its function-evaluation complexity," *IMA Journal of Numerical Analysis*, 2010.

[25] F. E. Curtis and M. L. Overton, "A sequantial quadratic programming algorithm for nonconvex, nonsmooth con-

strained optimization," *SIAM Journal on Optimization*, vol. 22, pp. 474–500, 2012.
[26] F. E. Curtis, T. Mitchell, and M. L. Overton, "A BFGS-SQP Method for Nonsmooth, Nonconvex, Constrained Optimization and its Evaluation using Relative Minimization Profiles," tech. rep., New York University, 2015.
[27] A. Asuncion and D. Newman, "UCI Machine Learning Repository," 2007.
[28] P. Cortez and A. M. G. Silva, "Using data mining to predict secondary school student performance," in *Proceedings of 5th Annual Future Business Technology Conference*, EUROSIS, 2008.
[29] ARIC Investigators and others, "The Atherosclerosis Risk in Communitities (ARIC) study: Design and objectives," 1989.
[30] J. Platt *et al.*, "Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods," *Advances in Large Margin Classifiers*, vol. 10, no. 3, pp. 61–74, 1999.
[31] R. Tibshirani, "Regression Selection and Shrinkage via the Lasso," 1994.
[32] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.
[33] G. S. Watson, "Smooth regression analysis," *The Indian Journal of Statistics, Series A*, vol. 26, no. 4, pp. 359–372, 1964.
[34] N. V. Sahinidis, *BARON 15: Global Optimization of Mixed-Integer Nonlinear Programs,* User's Manual, 2015.
[35] StudyInEurope.eu, "Study in Portugal." http://www.studyineurope.eu/study-in-portugal/grades, 2016.

APPENDIX

Lists of Features

$\mathbf{c}^+/\mathbf{c}^-$	**Feature:Cost**
c^+	Study time: 7, Paid tutoring: 8
c^-	Time out with friends: 6, Weekday alcohol: 3, Weekend alcohol: 6, Absences from class: 5

TABLE I: Directly changeable variables for the Student Performance dataset.

$\mathbf{c}^+/\mathbf{c}^-$	**Feature:Cost**
c^+	Dark or grain breads: 3, Peanut butter: 4, Nuts: 5, Other(prunes,avocado): 5, Vegetables: 6, Fruit: 6, Fiber: 7, Vegetable fat: 5, Polyunsaturated fat: 5
c^-	Liver: 8, White carbs: 6, Fish: 9, Cereal: 4, Cigarettes: 9, Caffeine: 7, Carbs: 7, Cholesterol: 6, Sodium: 7, Animal fat: 7, Saturated fat: 6
c^+/c^-	Exercise hours: 10, Alcohol: 9

TABLE II: Directly changeable variables for the ARIC CVD dataset.

Proof of Proposition 1

Consider the index $i \in \mathcal{A}_-$. Due to the relationship $l_i' \leq z_i \leq u_i' \leq \min(0, w_i)$, any feasible value of z_i can be at most u_i' while deviating z_i from u_i' increases the objective value of (5) and generates cost at a rate of c_i^-. Hence, the optimal value for z_i must be u_i' for each index $i \in \mathcal{A}_-$. Similarly, the optimal value for z_i must be l_i' for this index $i \in \mathcal{A}_+$.

With the optimal value of z_i for $i \in \mathcal{A}_+ \cup \mathcal{A}_-$ determined, the optimization problem (5) is reduced to

$$\min_{\tilde{\mathbf{z}} \in \Delta_{\tilde{D}}} \quad \frac{1}{2}\|\tilde{\mathbf{z}} - \tilde{\mathbf{w}}\|^2 \tag{11}$$

where $\tilde{D} = D \backslash (\mathcal{A}_+ \cup \mathcal{A}_-)$, $\tilde{\mathbf{w}} = \mathbf{w}_{\tilde{D}}$, i.e., the sub-vector of $\mathbf{w}$ containing the features in $\tilde{D}$, and

$$\Delta_{\tilde{D}} \equiv \left\{ \tilde{\mathbf{z}} \in \mathbb{R}^{|\tilde{D}|} \;\middle|\; \begin{array}{l} \sum_{i\in\tilde{D}} c_i^+(\tilde{z}_i)_+ + c_i^-(\tilde{z}_i)_- \\ B - \sum_{i\in\mathcal{A}_-} u_i' c_i^- - \sum_{i\in\mathcal{A}_+} l_i' c_i^+, \\ l_i' \leq \tilde{z}_i \leq u_i' \text{ for } i \in \tilde{D}. \end{array} \right\}.$$

For any $\lambda \geq 0$, let $z_i = \max\{\min\{h_i(w_i, \lambda), u_i'\}, l_i'\}$ for $i \in \tilde{D}$. Using the definition of h_i in (6), we can show that the elements in the set

$$z_i - w_i + \lambda c_i^+ \partial(z_i)_+ + \lambda c_i^- \partial(z_i)_-$$

are all positive only if $z_i = l_i'$ and the elements in the set

$$z_i - w_i + \lambda c_i^+ \partial(z_i)_+ + \lambda c_i^- \partial(z_i)_-$$

are all negative only if $z_i = u_i'$ for any $i \in \tilde{D}$, where $\partial(z)_+$ and $\partial(z)_-$ represent the subdifferentials of the functions $(z)_+$ and $(z)_-$[6]. This indicates that $(z_i)_{i\in\tilde{D}}$ is the optimal solution of the Lagrangian relaxation problem

$$\min_{l_i' \leq \tilde{z}_i \leq u_i', i\in\tilde{D}} \quad \frac{1}{2}\|\tilde{\mathbf{z}} - \tilde{\mathbf{w}}\|^2 + \lambda \left(\sum_{i\in\tilde{D}} c_i^+(\tilde{z}_i)_+ + c_i^-(\tilde{z}_i)_- \right)$$

with λ being the Lagrangian multiplier. Step 4 and Step 8 in Algorithm (2) ensure $(z_i)_{i\in\tilde{D}}$ is a feasible solution of (11) and satisfies the complementary slackness conditions with λ. This implies that $(z_i)_{i\in\tilde{D}}$ is the optimal solution of (11) so that $(z_i)_{i\in D}$ is the optimal solution of (5).

[6]Note that the subdifferential of a non-smooth function at some point can be a set.