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 can use any differentiable classification function. We demonstrate the method by using logistic regression and 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 (indirectly changeable) features whose values change as a consequence of actions taken. 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.

Index Terms—Inverse classification, SVM, Logistic regression, Utility-based learning

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 a 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.

Problems such as this are prevalent in personalized medicine settings. Consider, for example, lifestyle choices that minimize Patient 15's long-term risk of cardiovascular disease (CVD) – a randomly selected patient from our experiments in Section IV. An initial risk prediction, estimated to be 32%, is obtained using a trained, nonlinear classifier, based on Patient 15's EHR data. With Patient 15's initial risk now known, we wish to work "backwards" through the classifier to obtain recommendations that minimize the probability of CVD. We approach the recommendation step by defining an optimization problem: what is the smallest (or easiest) set of feasible changes that this person can make in order to minimize the predicted probability of developing CVD?

Our first contribution in this work is to define an inverse classification framework that produces realistic recommendations. We do so by first partitioning features into two categories: unchangeable and changeable. It would be impossible for Patient 15 to reduce her age – this is an unchangeable feature. Changeable features are further partitioned into directly and indirectly changeable categories. Directly changeable features are immediately actionable – we can recommend that Patient 15 adjust her diet, for example. Indirectly changeable features change as a consequence of manipulations to the directly changeable features, but are themselves not actionable. Blood glucose changes as Patient 15's diet is altered, but cannot be directly altered itself.

In our framework, directly changeable features incur individual, attribute-wise cost. Cumulative costs across such features are constrained to be within a budgetary level. These costs and budget can be specified by either a domain expert, the individual (e.g., Patient 15), or some combination of the two.

The second contribution of this work is a method that solves the inverse classification problem within the specified framework. Our method uses the gradient information of classifiers to provide recommendations that minimize the probability of an undesirable class. Using such a method within the specified framework we are able to provide recommendations that reduce Patient 15's probability of CVD from 32% to 3%.

The third contribution we identify is to specify two bound-setting methods, Elastic and Hard-line, that operate within the outlined framework allowing inverse classification to occur more freely or more rigidly, depending upon the problem. Lastly, we incorporate an indirect feature estimator, that adjusts features that change as a consequence of the directly alterable set of features.

In the remainder of the paper we discuss past work (Section II), our proposed framework and new method of inverse classification (Section III), our 16 experiments, conducted on two freely available datasets using our method and a sensitivity analysis-based benchmark method (Section IV), and

the conclusions we make following these experiments (Section V).

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. While there are many forms of sensitivity analysis [1], [2], inverse classification is most similar to local sensitivity analysis and variable perturbation method. Later on (Section III), we propose a benchmark method that is based on these.

Past works on inverse classification can be looked at from three perspectives: the manner in which the algorithm operates, the type of data the algorithm operates on, and the framework that guides the process of obtaining recommendations. Algorithm operation, which represents the optimization method employed, can be broken down into two groups: **greedy** [3]–[6] and **nongreedy** [7], [8]. Greedy methods tend to focus on extreme objectives, which may not be realistic in the real world, while nongreedy methods tend to focus on more moderate objectives. This work uses the latter.

Algorithmic data types, which refers to the type of data a particular optimization algorithm has the capability of operating on, also fall into two categories: **discrete** [3]–[5] and **continuous** [6]–[8]. Discrete data types lead to coarse-grained recommendations, while continuous data types provide those that are more fine-grained. In this work, we focus on the latter, as precision recommendations are the goal.

Framework refers to the constraints that govern recommendation feasibility. These are manifested in the literature as either **unconstrained** [3]–[5] or **constrained** [6]–[8]. Unconstrained problems lead to unrealistic recommendations that may also be very extreme (e.g., 'reduce your age by 30 years'). Constrained frameworks lead to more moderate and realistic recommendations. However, while [7], [8] focus on moderate objectives, they do not consider (1) what can/cannot be changed, (2) how hard it might be to change and (3), cumulatively, how willing someone may be to make changes. In [6] the authors consider (2), but do not consider (1) and (3). Additionally, in [7], the formulation of *border classification* relies on data points which lie exactly on the separating hyperplane; there is not guarantee that such points exist in practice. In this work we propose a framework that considers (1), (2) and (3).

Inverse classification is a utility-based data mining topic and is thereby related to the subtopics of strategic [9] and adversarial [10] learning. In these topics it is assumed that a strategic agent may attempt to *game* a learned classifier in order to conform to a desired class. Classifiers are then constructed taking such behavior into account. Such considerations do not need to be made in an inverse classification setting, however, as the goal is to provide explicit instructions to an intelligent agent (e.g., person) on how they can conform to a desired class, thereby making such accounts both unnecessary and undesirable.

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, assumed to have been drawn i.i.d. from some population distribution $\mathcal{P}$, 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 $X = [\mathbf{x}^1, \ldots, \mathbf{x}^n]^T \in \mathbb{R}^{n \times p}$ denote the matrix of training instances with $(\mathbf{x}^i)^T$'s being its rows. Any number of classification models can be trained with this dataset and used to predict the class of new instances. Unlike typical classification settings, however, 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 the probability of $\mathbf{x}$ 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, for 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:

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

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 change as a consequence of manipulations made to 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 $H : \mathbb{R}^{|D|+|U|} \to \mathbb{R}^{|I|}$ is assumed to be smooth and differentiable. Note that the mapping H can be trained using the same training instances for $f(\mathbf{x})$. Furthermore, while the estimates elicited from H may be noisy, using H is better than allowing the I values to remain static by definition of what I represents. 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

$$\min_{\mathbf{x}'_D \in \mathbb{R}^{|D|}} f(\mathbf{x}_U, H(\mathbf{x}'_D, \mathbf{x}_U), \mathbf{x}'_D) \qquad (2)$$

$$\text{s.t.} \sum_{i \in D} c_i^+ (x'_i - x_i)_+ + c_i^- (x'_i - x_i)_- \leq B$$

$$l_i \leq x'_i \leq u_i \text{ for } i \in D.$$

We relate a specific method for solving $H(\mathbf{x}'_D, \mathbf{x}_U)$ in Section IV.A.3. We note that, in practice, D is likely to be small and that, while U may be large (e.g., pictorial or text-based features), the efficiency of the optimization won't be affected.

1) Time Complexity of H: We acknowledge that the size of the indirectly changeable feature set I may be large and, as a result, wish to examine the time complexity associated with the indirect feature estimator H, which may prove to be a computational bottleneck.

Let H_a denote the indirect feature estimator for feature $a \in I$ and let r_a denote the corresponding time complexity associated with using H_a; that is, H_a is $\mathcal{O}(r_a)$. We can then write the time complexity of H as

$$R = \sum_{a \in I} r_a \qquad (3)$$

where R is the time complexity of H. As we can see, R increases linearly with the size of I (this is by virtue of the fact that we can estimate each feature in I independently). However, depending on the choice of H_a, and the size of I, this may still prove to be a bottleneck. If this is the case, the user may need to tailor their selection of H_a, or forgo estimating certain I features during the inverse classification process. We empirically show that the time complexity scales linearly using the H defined in the experiments section (kernel

regression), and include the result in the supplementary material that can be found at the publicly accessible repository github.com/michael-lash/BCIC.

2) 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 and is most appropriate when domain experts can interject their own knowledge in designating which directions of change are most beneficial. We refer to this as the Hardline 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 individual costs are incurred differently, depending on whether a specific feature is increased or decreased.

B. Optimization Method

To solve the inverse classification problem, according to (1) and (2), we assume that 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 [11] and extensive references therein) include: (a) deterministic approaches such as branch and bound [11], function approximation [12], cutting plane methods [13], difference of convex functions methods [14]; and (b) stochastic approaches such as genetic algorithms [15]. 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 [16], [17] and the zero-order method [17]. If $f(x)$ is second-order differentiable, the

---

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

list of potential methods can be extended to include regularized Newton's method, sequential quadratic programming and BFGS. 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 remaining 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 presentation of 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

$$\min_{\mathbf{z} \in \Delta_D} g(\mathbf{z}) \tag{4}$$

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

$$\Delta_D \equiv \left\{ \mathbf{z} \in \mathbb{R}^{|D|} \middle| \begin{array}{l} \sum_{i \in D} c_i^+(z_i)_+ + c_i^-(z_i)_- \leq B, \\ l_i' \leq z_i \leq u_i' \text{ for } i \in D. \end{array} \right\}, \tag{5}$$

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

$$\mathbf{Proj}_{\Delta_D}(\mathbf{w}) \equiv \arg\min_{\mathbf{z} \in \Delta_D} \frac{1}{2} \|\mathbf{z} - \mathbf{w}\|^2. \tag{6}$$

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 (5) is then given as Algorithm 1.

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 [16], when $\eta \leq \frac{1}{L}$, Algorithm 1 guarantees that $\mathbf{z}^{(t)}$ converges to a stationary point (or so-called KKT point) of (4) at a rate of $O(\frac{1}{t})$, which is the best convergence for non-convex smooth optimization.

Algorithm 1 requires solving the projection $\mathbf{Proj}_{\Delta_D}(\mathbf{w})$ at each iteration, which is itself an optimization problem. An efficient solution scheme for this subproblem is critical for making Algorithm 1 expeditious. Fortunately, the domain $\Delta_D \neq \emptyset$ has a specific structure which allow us to solve

---

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

$\mathbf{Proj}_{\Delta_D}(\mathbf{w})$ for any $\mathbf{w}$ with an efficient subroutine. To see this, we define

$$h_i(w, \lambda) = \begin{cases} w - \lambda c_i^+ & \text{if } \lambda \leq \frac{w}{c_i^+} \text{ and } w > 0 \\ w + \lambda c_i^- & \text{if } \lambda \leq -\frac{w}{c_i^-} \text{ and } w < 0 \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

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: $\mathcal{A}_- \leftarrow \{i | u_i' \leq \min(0, w_i)\}$
2: $\mathcal{A}_+ \leftarrow \{i | \max(0, w_i) \leq l_i'\}$
3: $z_i \leftarrow u_i'$ for $i \in \mathcal{A}_-$ and $z_i \leftarrow l_i'$ for $i \in \mathcal{A}_+$
4: **if** $\sum_{i \in D \backslash (\mathcal{A}_+ \cup \mathcal{A}_-)} \max\{\min\{h_i(w_i, 0), u_i'\}, l_i'\} \leq B - \sum_{i \in \mathcal{A}_-} u_i' c_i^- - \sum_{i \in \mathcal{A}_+} l_i' c_i^+$ **then**
5: $\quad \lambda \leftarrow 0$
6: **else**
7: $\quad$ Apply bisection search to find $\lambda \in (0, +\infty)$ such that

$$\sum_{i \in D \backslash (\mathcal{A}_+ \cup \mathcal{A}_-)} \max\{\min\{h_i(w_i, \lambda), u_i'\}, l_i'\}$$
$$= B - \sum_{i \in \mathcal{A}_-} u_i' c_i^- - \sum_{i \in \mathcal{A}_+} l_i' c_i^+$$

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

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

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

C. Representativeness and Support

With our methodology defined, we wish to comment on, and subsequently quantify, both the representativeness of the training set from which our f will generalize and the support underlying the inverse classification of an instance. Therefore, we first propose $\boldsymbol{\delta}$**-dissimilarity**, related by Definition 1, which quantifies the dissimilarity between the training set distribution S and population distribution $\mathcal{P}$ using a linear discrepancy distance measure defined in Johansson et al. [18].

Definition 1. *The distribution S of the training set, drawn from the population distribution $\mathcal{P}$, is said to be $\boldsymbol{\delta}$-dissimilar to that of $\mathcal{P}$ if*

$$disc_{\mathcal{H}}(S, \mathcal{P}) \leq \delta. \tag{8}$$

where $disc_{\mathcal{H}}(S, \mathcal{P}) \triangleq \|\mu(S) - \mu(\mathcal{P})\|$ is the discrepancy distance between two samples [18], or in this case the training sample and population, we define $\mu(\cdot)$ to denote the mean of a particular distribution, and $\|\cdot\|$ is the Euclidean norm.

Using Definition 1, we relate the following proposition.

Proposition 2. *As the size of the training set n increases to infinity, the training set distribution S is asymptotically $\delta = 0$-dissimilar to that of population distribution $\mathcal{P}$.*

The proof of Proposition 2 is in the appendix. We wish to point out, however, that the variance and shape of $\mathcal{P}$ and S may be quite different despite S being $\delta = 0$-dissimilar to that of $\mathcal{P}$. Additionally, in practice, the i.i.d. assumption may not hold (in this work we assume it does). We leave methods, taking into account such factors, as tangential future work.

We are also concerned with ensuring that optimized instances be near training data. These underlying training data provide support as to the "trustworthyness" of the recommendations and corresponding probabilities elicited from the inverse classification process. Therefore, we define (ϵ, γ)-**support**, related by Definition 2, which empirically quantifies the degree to which an inversely classified instance can be trusted.

Definition 2. *Define the (ϵ, γ)-support for a particular test instance $\mathbf{x}$, to be the following:*

- *ϵ is the variance in the predicted probabilities of $\mathbf{x}$'s k nearest neighbors (from the training data). This measure provides an assessment as to the stability of the local probability space surrounding $\mathbf{x}$.*
- *γ is the number of neighbors that fall within $\max\bar{D}kNN = \frac{1}{n}\sum_{i=1}^{n} maxDist(kNN(\mathbf{x}^i))$ of $\mathbf{x}$, where the function $maxDist(\cdot)$ returns the maximum distance of training instance $\mathbf{x}^i$'s k nearest neighbors; $\max\bar{D}kNN$ represents the average of these maximum distances. By comparing the γ of $\mathbf{x}$ to the average γ of the training set we can observe whether a particular test instance has more (larger γ) or less (smaller γ) "support" (relative to the average from the training data) underlying the predicted probability.*

We explore (ϵ, γ)-support in the Experiments section.

IV. EXPERIMENTS

In this section we outline our experimental methods and then apply such methods to two datasets. The first is a benchmark dataset from th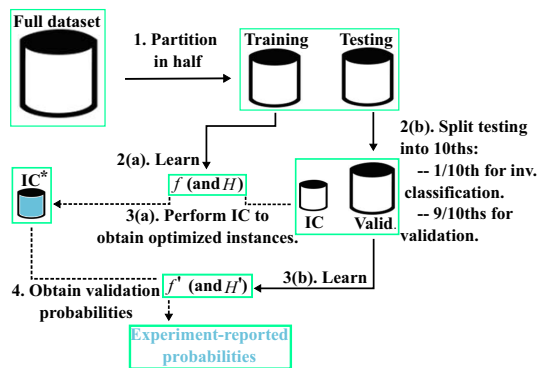e UCI Machine Learning Repository [19] called Student Performance [20]. The second is derived from ARIC, the Atherosclerosis Risk in Communities study [21]. We emphasize that both datasets are publicly available. The latter requires explicit NIH permission[3]. We provide the code used in all experiments, and processed Student Performance data for public use at github.com/michael-lash/BCIC. The list of unchangeable, indirectly changeable, and directly changeable features (and corresponding parameters) for both datasets is also provided at the above mentioned URL.

We emphasize that parameterization of the inverse classification framework, including the costs-to-change and assignment of features to the categories of unchangeable, indirectly changeable and directly changeable, should be guided by domain experts. As such, our experiments on the ARIC dataset are guided by a CVD specialist who is a co-author of this work.

A. Experiment Parameters and Setup

In this section we outline a general process of validating inverse classification methods, the two learning algorithms used to conduct the inverse classification, a method for estimating indirectly changeable features, and a benchmark optimization method which we will compare against our gradient-based method.

1) Process: Our process of making and evaluating recommendations is based on that proposed by [4]. In our experiments, 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 of validating whether we would have actually reduced the probability of such an event occurring. A method that accomplishes this 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, shown in Figure 1, is related as follows:

Step 1: Partition the full dataset into two equal parts: a training set and a testing set. Data cleansing and preparation are also performed, including missing value imputation (mean) and the normalization of data values to be within $[0, 1]$.

Step 2(a): 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.

Step 2(b): Further split the testing set into 10ths. 1/10th is for performing inverse classification on and the other 9/10ths are used for validation.

Step 3(a): Perform inverse classification on the heldout 10th of data using f.

Step 3(b): Learn a validation f' (and H') using the 9/10ths of heldout testing data.

Step 4: Estimate probabilities for the optimized inverse classification instances using f'. These are the probabilities

Fig. 1: Experiment process.

---

[3]Obtained via BioLINCC.

we report in our experiments. Note that we obtain probabilities for each 1/10th of held out testing data.

By setting up the experiment in this manner we are also able to be more confident that the recommendations obtained are not the result of overfitting. Note also that by switching the roles of training and validation/test sets, the full amount of data can be used to obtain results.

2) Classification Functions: Our experiments employ the use of two different learning methods: the linear *logistic regression* model and the nonlinear *kernel SVM*.

Logistic regression is a popular predictive model that works particularly well when the linear feature independence assumption holds. The model is trained via maximum likelihood estimation, given by the optimization problem

$$\max_{\boldsymbol{\beta},\beta_0} \sum_{i=1}^{n} -\log(1 + \exp(\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}^i)) + \sum_{i=1}^{n} y^i(\beta_0 + \boldsymbol{\beta}^\top \mathbf{x}^i)$$
(9)

Employment of the logistic model in our described inverse classification framework can be viewed as a basic method having roots in sensitivity analysis. This is illustrated by observing the link between coefficient examination as a means of sensitivity analysis and the employment of our described gradient-based methodology. Examining the sign and magnitude of a coefficient uncovers a particular feature's bearing – how positive or how negative – on the problem being modeled. Taking the gradient of a linear model has the same effect, thus informing the inverse classification framework which feature perturbations decrease the objective function value, with larger coefficients having a larger effect. Integration of this optimization methodology into the framework allows cost, budget, etc. to be taken into account as well.

Among classification models, the 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

$$\max_{\alpha \in \mathbb{R}^n} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^i y^j k(\mathbf{x}^i, \mathbf{x}^j)$$
(10)

$$\text{s.t.} \sum_{i=1}^{n} \alpha_i y^i = 0 \text{ and } 0 \le \alpha_i \le C \text{ for } i = 1, 2 \ldots, n,$$

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 (10) is $\alpha^* \in \mathbb{R}^n$. An SVM classifier can be derived based on the function[4]

$$f(\mathbf{x}) = \sum_{i=1}^{n} \alpha_i^* y^i k(\mathbf{x}^i, \mathbf{x}),$$
(11)

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.

However, the scores obtained from $f(\mathbf{x})$ do not correspond to likelihood directly. Therefore, we apply Platt's Method [22]. 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 for three reasons. The first is that such a function is highly nonlinear and complex, giving us the opportunity to explore a more flexible 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, which isn't enforced by linear predictors. Finally, using the σ parameter, we can control the size of the neighborhood used to assess point similarity. That is, larger σ values make more distant support vectors appear more similar to a test point $\mathbf{x}$, which subsequently has the effect of smoother probability transitions during optimization.

Therefore, our objective function, outlined in (1) and (2), becomes (**??**) and (11), logistic and SVM, respectively, with features segmented into appropriate groups and the indirect feature estimator, outlined in the next subsection, incorporated. We explicitly note that, in the case of (11), the minimization task is to minimize the SVM score. More appropriately, by applying Platt's method, we will be minimizing probability directly, as we are when using (**??**).

3) Estimating Indirectly Changeable Features: We employ the use of Kernel Regression [23], [24] as a means of estimating the indirectly changeable features. In particular, the model $\mathbf{x}_I = H(\mathbf{x}_D, \mathbf{x}_U)$ used in (2) is

$$\mathbf{x}_I = \frac{\sum_{i=1}^{n} k([\mathbf{x}_D^i, \mathbf{x}_U^i], [\mathbf{x}_D, \mathbf{x}_U]) \mathbf{x}_I^i}{\sum_{i=1}^{n} k([\mathbf{x}_D^i, \mathbf{x}_U^i], [\mathbf{x}_D, \mathbf{x}_U])},$$
(12)

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 (12) 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 training set $\mathbf{x}_I^i$, that are closer to $\mathbf{x}$, more favorably than those that are further away. In so doing, (12) obtains an estimate for $\mathbf{x}_I$ based on points that are most similar to it.

---

[4]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.

4) Methodological Benchmark: In our experiments we wish to compare our method to that of another. However, to the best of our knowledge, there exists no past methods, including those found in Section II, that can be incorporated into our framework. Therefore we develop a method, based on sensitivity analysis, that we believe represents a reasonable initial attempt at solving the problem from such a standpoint. Our proposed benchmark method operates by iteratively perturbing each feature $\mathbf{x}_{D_i}$ $i \in D$ to the bounds of feasibility (and is therefore akin to the variable perturbation method of sensitivity analysis [2]). 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 perturbation occur (double-feature perturbation, triple-feature perturbation, etc.).

Here we assert that, because we have chosen two different indirectly changeable feature estimators, we will effectively be using two different benchmark methods.

Cumulatively, our experiments will involve two datasets (ARIC, Student Performance), two classification functions (logistic, SVM), two optimization methods (PGD, sensitvity analysis-based), and two bound-setting methods (Hardline and Elastic) which constitute a total of 16 experiments.

B. Data Description

We validate the effectiveness of our inverse classification framework on two datasets: Student Performance and ARIC. Student Performance data consists of individual Portuguese students enrolled in two different classes. 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 43 associated features ($p = 43$). The dependent variable is whether a student earned a final grade of C or below ($y = 1$) or not ($y = -1$). We discard the two intermediary grade reports to reflect the long-term goal of earning a better grade. Therefore, the task is to minimize the probability of earning a C or below.

The ARIC dataset contains $n = 12907$ patients for which we define 110 features (please refer to github.com/michael-lash/BCIC). As the problem domain is medicine-based, we consulted an epidemiologist, a coauthor of this paper. We define $y = 1$ to be a positive CVD diagnosis, which includes 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 $y = -1$. Additionally, patients having one of these diagnoses prior to the study period were excluded from our dataset (giving us the final $n = 12907$ patients).

C. Results: Probability Reduction

The results of our 16 experiments are reported in terms of average probability relative to budget, which can be viewed in Figure 2, where the subfigures stratify results by dataset and bound-setting method.

Comprehensively we can see that, in the general case, all methods except the logistic classifier using PGD on the Student Performance dataset were successful in reducing the average probability of a negative outcome. Depending on the dataset and bound-setting method used, different methods coupled with different classifiers experienced different degrees of success. This seems to suggest that, as in typical classification settings, methodological success varies on a dataset-to-dataset basis.

Interestingly, at a high level, there is no difference between the results obtained using the Hardline and Elastic bound-setting methods on Student Performance and only one distinct difference between the results obtained on ARIC. Here, logistic regressing using the PGD method is observed to have distinctly greater average performance using the Elastic bound-setting method (shown in Figure 2d). Such a result should be viewed cautiously, however, as the recommendations obtained may differ, and perhaps even contradict, those our cardiovascular disease specialist would view as being truly beneficial. Differences of this nature may be attributable to possible noise in the ARIC data.

In examining the results obtain on Student Performance, shown in Figures 2a and 2b, some interesting findings emerge [5]. We can see that the best result obtained using the logistic classifier was through the sensitivity analysis-based method and the best obtained using the SVM classifier was through PGD. This may suggest that simpler, linear classifiers may experience better inverse classification results using simpler means of optimization and that more complicated, non-linear classifiers may see better results using those that are more complicated.

This latter point is somewhat supported by the results obtained on the ARIC dataset, shown in Figures 2c and 2d. In examining Figure 2c we can see that PGD outperformed the sensitivity analysis-based method when using the nonlinear SVM classifier and that the sensitivity analysis-based method outperformed PGD when using the linear logistic classifier. However, in Figure 2d, which represents results obtained using the Elastic bound-setting method PGD has dominated in the case of both classifiers. This result seems to suggest that, regardless of classifier complexity, if there exist optimizations that benefit from an Elastic setting (recall that no benefits were found from such a setting on Student Performance), PGD may dominate (on average).

Unexpectedly, looking at the results obtained for a randomly selected individual from either dataset, we can see that there is no difference in probabilistic improvement between the two bound-setting methods based when using SVM with PGD. The specific recommendations made to these individuals are discussed in the next subsection along with recommendations most commonly made to individuals in each dataset at a budget of four.
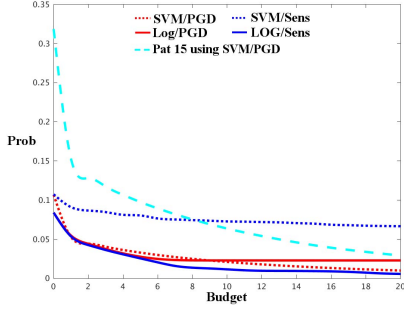
---

[5]We wish to point out that the probabilistic estimates obtained from the two classifiers are disparate, which we believe stems from small amounts of training data
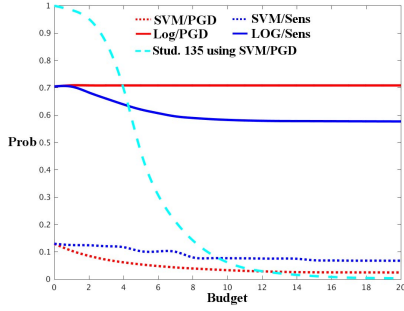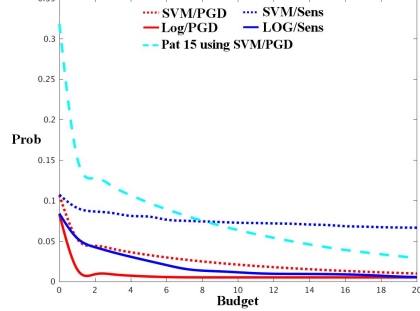
(a) SP dataset using Hardline Bound-setting.
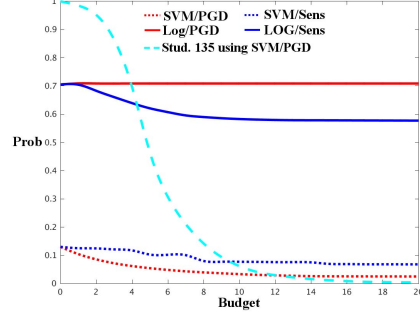
(b) SP dataset using Elastic Bound-setting.

(c) ARIC dataset using Hardline Bound-setting.

(d) ARIC dataset using Elastic Bound-setting.

Fig. 2: Average probability vs. budget by dataset (Student Performance or ARIC) and by bound-setting method. Solid lines represent a result obtained using the logistic model, while dotted lines represent a result obtained using the SVM model. PGD denotes use of the gradient method, while Sens denotes use of the sensitivity analysis-based method. The cyan dashed line is a randomly selected individual whose recommendations will be shown and discussed in the next subsection.
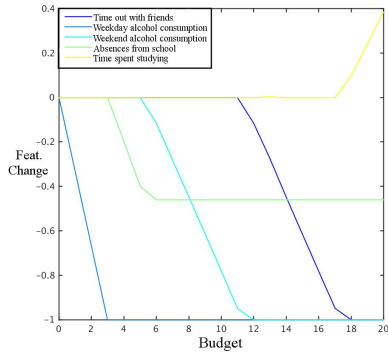
(a) Student 135.
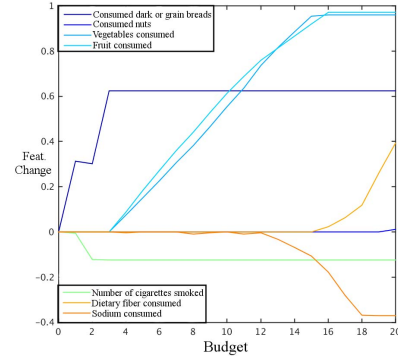
(b) Patient 15.

Fig. 3: Recommended changes vs. budget for a randomly selected individual from each dataset.

D. Results: Cumulative and Individual Recommendations

In this subsection we briefly relate the most common changes recommended to individuals in each dataset and then discuss the definitive recommendations made to two randomly selected instances.

Table I shows the most common recommendations by raw count, the highest ranking of which pertain to features relevant to nearly all individuals (time with friends and eating food, for instance).

Not all changes could be made to all individuals, how-

Rank	Student Perf.	ARIC
1	Time w/ friends	Eat dark/grain bread
2	Study time	Eat fruit
3	Absences	Cigs/day
4	Weekday alco. cons.	Eat veggies

TABLE I: Most commonly recommended feature changes by dataset using SVM with the PGD method at a budget of four.
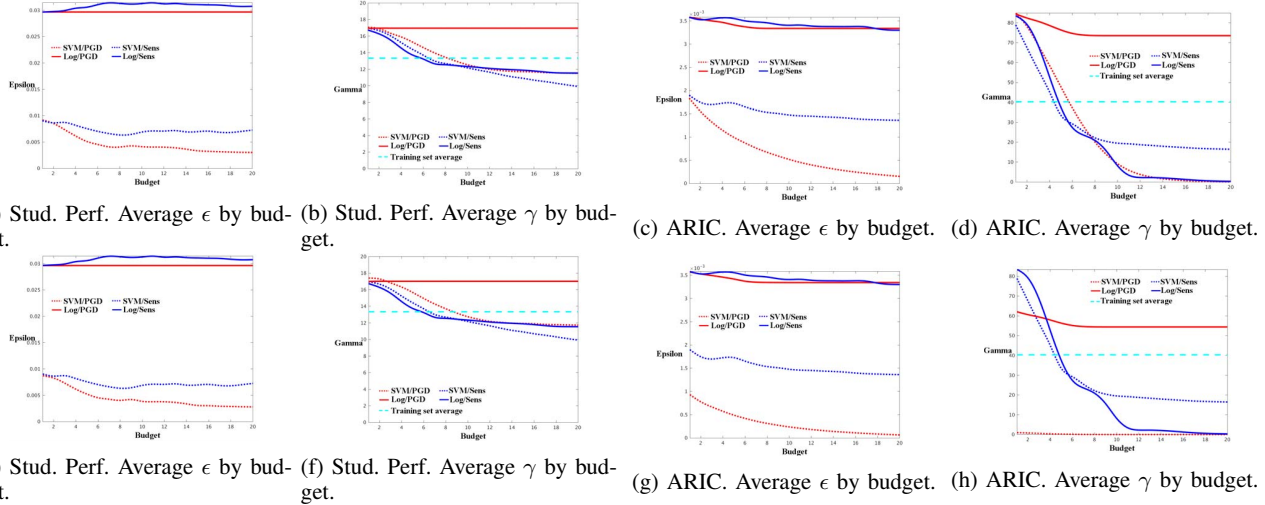
(a) Stud. Perf. Average ϵ by budget.
(b) Stud. Perf. Average γ by budget.
(c) ARIC. Average ϵ by budget.
(d) ARIC. Average γ by budget.

(e) Stud. Perf. Average ϵ by budget.
(f) Stud. Perf. Average γ by budget.
(g) ARIC. Average ϵ by budget.
(h) ARIC. Average γ by budget.

Fig. 4: (ϵ, γ)-support for Student Performance and ARIC using both the **Hardline** (4a-4d) and **Elastic** (4e-4h) bound-setting methods with $k = 10$.

ever. For instance, not all individuals drink during the weekdays (Student Performance) and not all individuals smoke cigarettes(ARIC). Therefore, red shows that when recommendation commonality is normalized by the number of individuals who were engaging in weekday drinking and smoking, 97.97% and 99.98% of the time alterations to such behaviors were respectively recommended. Such a result shows that while such risky behaviors are not necessarily common among all individuals, those who do engage in them are frequently recommended to make alterations.

Figures 3a and 3b show the changes recommended to a randomly selected individual from Student Performance and ARIC, respectively, using SVM with the PGD method.

Contrasting Figure 3a with Figure 3b we can see that, in the case of the former, a single feature was optimized to the extent of feasibility before perturbations were made to another, whereas in the case of the latter, optimization of several features happened in tandem.

In examining the specific recommendations made to Student 135 in Figure 3a, we can see that first weekday drinking was curbed, followed by a reduction in school absences, weekend alcohol consumption, and time out with friends, as the budget was increased. Last, at the second highest budgetary level, time spent studying was increased. In the aggregate, it seems as though risk-related behavioral mitigations were determined to be optimal for this student.

Looking at the recommendations made to Patient 15 in Figure 3b we can see that, at low budgetary levels, an increase in dark or grain breads and a decrease in the number of cigarettes were recommended. Following these, as the budget was further incremented, consumption of more fruits and vegetables, in tandem, was recommended. At a budget of 13 it was also recommended that the patient decrease sodium intake and then subsequently, at a budget of 18, dietary fiber intake was increased. Finally, at a budget of 20, an increase in

the consumption of nuts was recommended. Comprehensively, the recommendations deemed optimal for this patient were dietary-based, with the exception of a reduction in the number of cigarettes.

E. Results: (ϵ, γ)-support

The results in Figure 4 show that our inverse classifications are well supported in terms of probability space (ϵ) and underlying training data (γ) for both Student Performance and ARIC, up to certain budgetary levels (sans SVM/PGD in 4h). This suggests that, in future work, a constraint on the underlying γ-support may be desirable. The results were obtained by taking the average over the ϵ, γ values of all optimized test instances for each budgetary level explored in past experiments.

V. CONCLUSIONS

In this work we propose and validate a new framework and method for inverse classification. The framework ensures that recommendations are realistic by accounting for what can actually be changed, the cost/effort required to make changes, the cumulative effort (budget) an individual is willing to put forth, and the effects that making changes have on features that are not directly actionable. Additionally, we impose bounds on the changeable features that further ensure recommendations are realistic, as well as two bound-setting methods that govern algorithmic recommendation-generating behavior. Furthermore, our methods are very modular, allowing for the use of any differentiable classification function (logistic regression, neural networks, etc.), as well as virtually any estimator of the indirectly changeable features. We demonstrated the efficacy of these methods on two freely available datasets as compared to a baseline method. Future work will focus on augmenting the framework with additional utility, as well as on

conducting an in-depth analysis exploring situations in which PGD outperforms sensitivity analysis-based methods.

REFERENCES

[1] S. S. Isukapalli, "Uncertainty analysis of transport-transformation models," Ph.D. dissertation, 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*, July 2003, pp. 272–277.

[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. [Online]. Available: http://dx.doi.org/10.1016/j.jbi.2012.07.011

[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*, 2012, pp. 603–610. [Online]. Available: http://dl.acm.org/citation.cfm?id=2110430

[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*, 2009, pp. 305–11.

[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] F. Boylu, H. Aytug, and G. J. Koehler, "Induction over strategic agents," *Information Systems Research*, vol. 21, no. 1, pp. 170–189, 2010.

[10] D. Lowd and C. Meek, "Adversarial learning," in *Proceedings of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2005, pp. 641–647.

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

[12] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *Journal of Global Optimization*, vol. 21, no. 4, pp. 345–383, Dec. 2001. [Online]. Available: http://dx.doi.org/10.1023/A:1012771025575

[13] 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.

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

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

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

[17] 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.

[18] F. D. Johansson, U. Shalit, and D. Sontag, "Learning representations for counterfactual inference," in *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*, ser. ICML'16. JMLR.org, 2016, pp. 3020–3029. [Online]. Available: http://dl.acm.org/citation.cfm?id=3045390.3045708

[19] A. Asuncion and D. Newman, "UCI Machine Learning Repository," 2007.

[20] 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.

[21] ARIC Investigators and others, "The atherosclerosis risk in communities (ARIC) study: design and objectives," *American Journal of Epidemiology*, vol. 129, no. 4, pp. 687–702, 1989.

[22] 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.

[23] E. A. Nadaraya, "On estimating regression," *Theory of Probability & Its Applications*, vol. 9, no. 1, pp. 141–142, 1964.

[24] G. S. Watson, "Smooth regression analysis," *The Indian Journal of Statistics, Series A*, vol. 26, no. 4, pp. 359–372, 1964.

APPENDIX

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 (6) 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 (6) is reduced to

$$\min_{\tilde{\mathbf{z}} \in \Delta_{\tilde{D}}} \quad \frac{1}{2}\|\tilde{\mathbf{z}} - \tilde{\mathbf{w}}\|^2 \tag{13}$$

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

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

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 (7), 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 (13) and satisfies the complementary slackness conditions with λ. This implies that $(z_i)_{i \in \tilde{D}}$ is the optimal solution of (13) so that $(z_i)_{i \in D}$ is the optimal solution of (6). $\square$

Proof of Proposition 2

Assume that the training set is drawn i.i.d. from population distribution $\mathcal{P}$, having distribution S, where each dimension is in the range $[0, 1]$, and that the size of the training set n is large, then by the central limit theorem $\mu(S) \xrightarrow{d} \mu(\mathcal{P}) \implies \text{disc}_{\mathcal{H}}(S, \mathcal{P}) \xrightarrow{d} \|\mathbf{0}\| \leq \delta = 0$, as desired. $\square$

---

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