An Inverse Classification Framework with Limited Budget and Maximum Number of Perturbed Samples

Jaehoon Koo[1] (jaehoonkoo@hanyang.ac.kr), Diego Klabjan[2] (d-klabjan@northwestern.edu), Jean Utke[3] (jutke@allstate.com)

[1] School of Business Administration, College of Business and Economics, Hanyang University, Ansan, Gyeonggi, South Korea

[2] Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA

[3] Data, Discovery and Decision Science, Allstate Insurance Company, Northbrook, IL, USA

Corresponding Author:

Diego Klabjan

Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL, USA

Tel: +1 (847) 491-0663

Email: d-klabjan@northwestern.edu

Abstract

Most recent machine learning research focuses on developing new classifiers for the sake of improving classification accuracy. With many well-performing state-of-the-art classifiers available, there is a growing need for understanding interpretability of a classifier necessitated by practical purposes such as to find the best diet recommendation for a diabetes patient. Inverse classification is a post modeling process to find changes in input features of samples to alter the initially predicted class. It is useful in many business applications to determine how to adjust a sample input data such that the classifier predicts it to be in a desired class. In real world applications, a budget on perturbations of samples corresponding to customers or patients is usually considered, and in this setting, the number of successfully perturbed samples is key to increase benefits. In this study, we propose a new framework to solve inverse classification that maximizes the number of perturbed samples subject to a per-feature-budget limits and favorable classification classes of the perturbed samples. We design algorithms to solve this optimization problem based on gradient methods, stochastic processes, Lagrangian relaxations, and the Gumbel trick. In experiments, we find that our algorithms based on stochastic processes exhibit an excellent performance in different budget settings and they scale well. The relative improvement of the proposed stochastic algorithms over an existing method with a traditional formulation is 15% in the real-world dataset and 21% in two public datasets on average.

Keywords: inverse classification, adversarial learning, counterfactual explanation, machine learning, neural networks

1. Introduction

Classification is a building block for solving various machine learning tasks such as customer segmentation, sentimental analysis, and image recognition. To

achieve high classification accuracy various perspectives should be considered in the model development phase; normalization, imputation, and treating outliers of given data, feature selection and extraction, advanced architectures of classifiers, hyperparameter tuning, and training. Especially, designing a classifier including feature engineering is one of the keys to high performance. Through numerous state-of-the-art feature engineering and advanced model architectures deep neural networks made significant progress on classification tasks (Goodfellow et al., 2016; Polap & Wlodarczyk-Sielicka, 2020). On the other hand, there is increasing demand for analyses and use of well-fitted models as a post process (Aggarwal et al., 2010; Lash et al., 2017a). A common post modeling step is to consider changes in features that alter the predicted class, e.g. from the prediction of becoming sick to remaining healthy. Given a trained classifier, inverse classification models identify minimal changes of input features of a sample so that the sample is predicted as a desired class that is different from its originally predicted class (Laugel et al., 2018). It is first introduced as a topic of sensitivity analysis (Mannino & Koushik, 2000) and then augmented as an interpretability approach (Barbella et al., 2009). Viewing inverse classification as a utility-based data mining problem, Lash et al. (2017a) argue that it is a subtopic of strategic learning (Boylu et al., 2010). Inverse classification is also related to counterfactual explanation in interpretable machine learning. A counterfactual explanation reveals how a sample should be perturbed to significantly change its original prediction. By crafting counterfactual samples we can interpret how a classifier computes individual predictions (Wachter et al., 2018; Molnar, 2019; Verma et al., 2020, 2021). Lowd & Meek (2005) and Laugel et al. (2018) point out that inverse classification is related to adversarial learning (Tygar, 2011) that aims to attack a classifier by applying small perturbations to samples to modify their initial predictions. Inverse classification and counterfactual explanation studies focus on interpretability of classification models. Meanwhile, adversarial learning mainly focuses on robustness of associated models. In addition to generating adversarial attacks, developing a defensive system against the attacks is a study of interest in adversarial learning (Machado et al., 2021).

Perturbing samples so that they are predicted as a desired label is a common goal in these areas. In this paper, we focus on generating adversarial attacks without any defensive system; thus, designing a defensive system is out of scope.

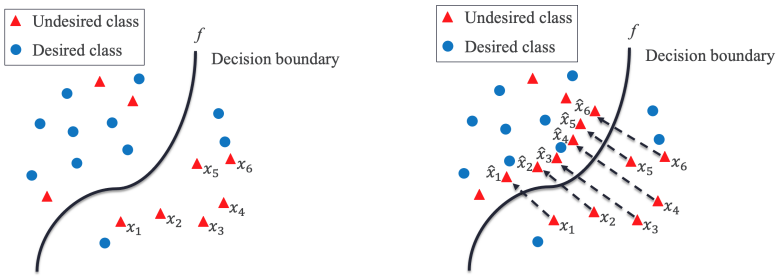
In many business applications samples correspond to treatment of customers or patients and the goal is to perturb their treatment in order to be predicted in a more favorable class. We often have limits on feature perturbation amounts, e.g., a limited number of interactions with all of the customers or we only have limited availability of a drug or procedure. Within per-feature-budget limits we want to treat as many customers or patients as possible. This is equivalent to stating that we want to maximize the number of perturbed samples because we turn each one of them into a more favorable class. Past works deal with objectives such as minimizing budget or other continuous loss functions but to capture the number of samples to perturb requires a discrete function which poses unique challenges addressed herein. In this work we are focusing on the problem of maximizing the number of selected samples subject to per-feature-budget limits and desired reclassification of the perturbed samples. We seek to obtain the maximal number of successfully perturbed samples while the budget is bounded by each input feature. This setting is a direct result of a real-world use case.

A typical setting considered in inverse classification, counterfactual explanations, and adversarial learning is as follows. Let us assume that we have a classifier $f : x \in X \rightarrow y \in Y$ with input x and output y. The goal is to generate an adversarial sample $\hat{x}$ that is of the same form as a given sample x, and is to be predicted as a desired class that is different from the originally predicted label of x. Especially in adversarial learning, there are two types of adversarial examples. A non-targeted adversarial example $\hat{x}$ is generated by adding small perturbation to x so that $\hat{x}$ is classified as any class that is not the original ground truth. A targeted adversarial sample fools a classifier so that it produces a desired label $f(\hat{x}) = \bar{y}$ where $\bar{y}$ is the desired class determined by the adversary. In this paper, we assume that we have a desired class for each sample. Therefore, our problem is aligned with a targeted adversarial attack.

Figure 1 illustrates process of inverse classification. A classifier is fitted to find decision boundaries, which leads to accurate classification. Through the trained classifier, adversarial samples are generated by perturbing input features of the samples such that they are labeled as desired. The L_p norm of the perturbation between adversarial samples and given samples is usually used as the loss function where p can be $0, 1, 2, \infty$ (Dong et al., 2018). In some cases, a set of budget constraints for the perturbation is introduced, and subsequently, not all of candidate samples can be successfully perturbed as desired. Herein we focus on spending the budget so that as many samples as possible can be successfully perturbed within the budget. Our goal is to perturb all of the candidate samples as desired within the budgets as shown in Figure 1 (b). Existing inverse classification and adversarial attack frameworks do not capture this perspective since they minimize the cost of the perturbation.

In this study, we develop a new framework that can be applied to inverse classification and counterfactual explanations as well as adversarial learning. We assume to have a budget constraint on perturbations of continuous input features. In order to obtain the maximum number of successfully perturbed samples within the budget, we define an objective function that maximizes the number of samples to be perturbed, which is different from the existing formulations of minimizing perturbations. To this end, we introduce a binary variable indicating which sample is to be perturbed. In addition, we include a set of constraints that the probability of the desired class produced by the classifier is higher than all other classes by a tunable margin. This is done to avoid a purely adversarial change in the prediction but instead induce perturbations yielding the actual desired change in the real-world process. We propose Langrangian-based models relying on binary variables. An alternative view considering the selection of samples as a stochastic process with unknown probabilities yields even better algorithms. The resulting model uses chance constraints and the Gumbel trick to make algorithms more efficient. We rely on gradient-based optimization in conjunction with Lagrangian relaxations.

For evaluation, we use a real-world proprietary dataset and two public



(a) Trained classifier f (b) Generating adversarial samples $\hat{x}$

Figure 1: Illustration of inverse classification

datasets from a health clinic (Johnson et al., 2016; Goldberger et al., 2000) and hospitals (Dua & Graff, 2017). We compare the performance of our algorithms with respect to the number of selected samples and the budget consumed per selected sample. Algorithms based on stochastic processes significantly outperform those relying on binary variables. We conduct budget and scalability experiments on the real-world data; the relative improvement of the proposed stochastic algorithms over an existing method with a traditional objective function is 7% and 23% on average for a variety of budget and possible sample size settings, respectively. In the budget experiments on the two public datasets, the stochastic algorithms outperform the existing model by 21% on average.

The contributions of this work are as follows.

1. We introduce a new framework to solve inverse classification to achieve the maximal number of successfully perturbed samples within a per-feature budget. As far as we know, this framework has not yet been applied to inverse classification in the existing literature.

2. We design novel algorithms based on gradient methods, stochastic processes, Lagrangian relaxations and the Gumbel trick.

3. In the computational study, stochastic approaches perform well on different budget scenarios, and they scale.

The rest of this paper is organized as follows. In Section 2, the related work is discussed. Section 3 describes the proposed models, and the algorithms to solve them are presented in Section 5. Section 5 provides the computational study including experimental details and analyses of the experimental results. Conclusions are given in Section 6.

2. Related work

In inverse classification and counterfactual explanation studies the focus is either on unconstrained or constrained problems, or algorithmic mechanisms (Lash et al., 2017a,b). A problem statement is related to either feasibility or implementability of perturbed samples, which yield either an unconstrained (Aggarwal et al., 2010; Yang et al., 2012) or a constrained problem (Barbella et al., 2009; Chi et al., 2012; Lash et al., 2017a,b; Wachter et al., 2018; Lash & Street, 2020; Mannino & Koushik, 2000; Gupta et al., 2021). Since an unconstrained formulation does not consider practical constraints such as a budget, it tends to produce unrealistic perturbations of input features of a sample, e.g. cannot offer a drug if a patient is at home. A constrained formulation provides realistic perturbations, however, it is challenging to solve them. There are three factors to be considered: a) identify changeable features to be perturbed, e.g. an unchangeable feature can be a product purchase history, b) how costly is it to change a feature, and c) limit the amount of perturbations over all samples, i.e., a budget (Lash et al., 2017a,b). In (Barbella et al., 2009), only aspect c is considered. Mannino & Koushik (2000) consider b), but do not consider a) and c). Lash et al. (2017a) propose a general framework that considers a, b, and c, however, a prediction confidence constraint is not included. With respect to algorithms there are greedy (Aggarwal et al., 2010; Chi et al., 2012; Yang et al., 2012; Lash et al., 2017b; Mannino & Koushik, 2000) and non-greedy (Barbella et al., 2009; Lash et al., 2017a; Lash & Street, 2020; Gupta et al., 2021) algorithms. Greedy methods are computationally efficient but typically suffer from low solution quality. Non-greedy methods tend to focus on more moderate ob-

jectives not capturing many aspects so that the obtained adversarial samples are more realistic. In (Aggarwal et al., 2010; Chi et al., 2012; Yang et al., 2012; Lash et al., 2017b; Mannino & Koushik, 2000), heuristic methods that do not use gradients such as local search, hill climbing, and genetic algorithm are used. In (Lash et al., 2017a; Lash & Street, 2020; Gupta et al., 2021), a projected gradient method is adopted and in (Barbella et al., 2009), a non-linear solver package is used to solve a constrained problem. Our work is different from the aforementioned research since none of the existing methods consider maximizing the number of perturbed samples in their formulation. Because of the discrete nature of the counting objective functions these algorithms are not appropriate to our problem.

In adversarial learning, recent research mostly focuses on generating adversarial samples to attack deep learning models to study robustness of state-of-the-art classifiers. Algorithms for generating non-targeted adversarial samples have been proposed in Szegedy et al. (2013); Goodfellow et al. (2015); Kurakin et al. (2017); Papernot et al. (2016); Combey et al. (2020). However, these algorithms do not consider a budget constraint that is critical and practical in inverse classification. In addition, our framework is designed to achieve the maximal number of successfully perturbed samples within a budget. Therefore, these algorithms are not applicable in a meaningful way to tackle our setting.

3. Proposed models

In this section, we present our constrained optimization problem to solve inverse classification. The formulation is designed to generate the maximal number of adversarial examples that are classified as a desired class. In addition, we include a set of budget constraints on perturbations of input features. We first introduce notation and the baseline model. Next, we present variations of the baseline model - chance constraint models that assume decision variables follow an unknown probability distribution.

We denote a given sample by $\mathbf{x} \in \mathbb{R}^p$ and a perturbed sample by $\hat{\mathbf{x}} \in \mathbb{R}^p$. We

assume that all features are continuous. Let $f(\mathbf{x}) : \mathbb{R}^p \mapsto [0,1]^k$ be a function associated with a classification model that computes a score of $\mathbf{x}$ being in a class such as the probability of k classes. We optimize over samples $\hat{\mathbf{x}}_j$ given a new desired label vector $\bar{\mathbf{y}}_j \in [0,1]^k$ with $j = 1, \ldots, |\mathcal{S}|$ and $\mathcal{S} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{|\mathcal{S}|}\}$. We denote a perturbed input feature matrix by $\hat{\mathbf{X}} = (\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_{|\mathcal{S}|})$. Furthermore, we introduce binary variables, $\mathbf{z} \in \{0,1\}^{|\mathcal{S}|}$, to decide which sample is to be perturbed. We maximize the number of these binary variables that have value 1. We introduce general budget constraints on perturbations of input features. In addition, we have a per-sample constraint, called the prediction confidence constraint, to capture a margin for prediction reflecting the uncertainty in the score function. Formally, the max samples model (MS) is formulated as

$$\begin{aligned} \max_{\mathbf{z}, \hat{\mathbf{X}}} \quad & \sum_{j=1}^{|\mathcal{S}|} z_j \\ \text{s.t.} \quad & g_i(\mathbf{z}, \hat{\mathbf{x}}) \leq 0, i = 1, \ldots, p, \\ & h_j(\mathbf{z}, \hat{\mathbf{x}}) \leq 0, j = 1, \ldots, |\mathcal{S}| \end{aligned} \tag{1}$$

where each g_i and h_j is a nonlinear function associated with the budget and prediction confidence constraint, respectively. A typical budget constraint for feature i on perturbation of input features is

$$g_i(\mathbf{z}, \hat{\mathbf{x}}) = \sum_{j=1}^{|S|} z_j \, ||\hat{x}_{ij} - x_{ij}||_2^2 - B_i \tag{2}$$

where $B_i \in \mathbb{R}_+$ is a given budget for feature i. Prediction confidence constraints are explicitly expressed as

$$f(\hat{\mathbf{x}}_j)_u + \delta \leq f(\hat{\mathbf{x}}_j)_{\tilde{y}_j} \text{ with } \; j = 1, \ldots, |\mathcal{S}|, u = 1, \ldots, k, u \neq \tilde{y}_j \tag{3}$$

where $\tilde{y}_j \in \underset{u}{\text{argmax}} \, [\bar{\mathbf{y}}_j]_u$ is a desired class of $\mathbf{x}_j$ and $\delta > 0$ is a given margin. In MS, we rewrite (3) by multiplying it with binary variables so that we consider

the constraints only on selected samples as follows:

$$h_j(\mathbf{z}, \hat{\mathbf{x}}) = z_j \cdot \bar{h}_j(\hat{\mathbf{x}}) = z_j \max\{0, \max_{\substack{u=1,\dots,k \\ u \neq \tilde{y}_j}} f(\hat{\mathbf{x}}_j)_u - f(\hat{\mathbf{x}}_j)_{\tilde{y}_j} + \delta\}, j = 1, \dots, |\mathcal{S}|. \tag{4}$$

MS is challenging to solve due to the presence of constraints and binary variables $\mathbf{z}$. To address the latter, we assume that the binary variables have a probability distribution which is an approximation. We impose that the variables follow either Bernoulli or Categorical distributions considering dependency among samples. First, we present the Bernoulli case where no relationship among perturbed samples is assumed.

Let $z_j \sim \text{Bernoulli}(\pi_j), j = 1, \dots, |\mathcal{S}|$, i.e., $z_j = 1$ with probability π_j. Let $0 \leq \mathit{\Pi} = (\pi_1, \pi_2, \dots, \pi_{|\mathcal{S}|}) \leq 1$. Transforming MS, we propose a Bernoulli chance max samples model (BCMS) as

$$\begin{aligned} \max_{\mathit{\Pi}, \hat{\mathbf{X}}} \quad & \mathbb{E}_{\mathbf{z}}\Big[\sum_{j=1}^{|\mathcal{S}|} z_j\Big] \\ \text{s.t.} \quad & \Pr\big(g_i(\mathbf{z}, \hat{\mathbf{x}}) \leq 0\big) \geq 1 - \epsilon, i = 1, \dots, p, \\ & \Pr\big(h_j(\mathbf{z}, \hat{\mathbf{x}}) \leq 0\big) \geq 1 - \epsilon, j = 1, \dots, |\mathcal{S}|, \end{aligned} \tag{5}$$

where ϵ is a parameter. We can explicitly rewrite BCMS as

$$\begin{aligned} \max_{\mathit{\Pi}, \hat{\mathbf{X}}} \quad & \sum_{j=1}^{|\mathcal{S}|} \pi_j \\ \text{s.t.} \quad & \Pr\big(\sum_{j=1}^{|S|} z_j \, ||\hat{x}_{ij} - x_{ij}||_2^2 - B_i \leq 0\big) \geq 1 - \epsilon, i = 1, \dots, p \\ & \pi_j \, \bar{h}_j(\hat{\mathbf{x}}_j) \leq 0, j = 1, \dots, |\mathcal{S}|. \end{aligned} \tag{6}$$

The chance max samples model with the Categorical distribution (CCMS) considers dependency among samples to be perturbed. To this end, let $0 \leq \mathit{\Pi} = (\pi_1, \pi_2, \dots, \pi_{|\mathcal{S}|})$ with $\sum_{j=1}^{|\mathcal{S}|} \pi_j = 1$ and K an integer parameter. CCMS has the same formulation as BCMS, but we use the following binary variables to

determine which sample to perturb:

$$\bar{z}_\xi \in \mathbb{R}^{|\mathcal{S}|} \sim \text{Cat}(\mathit{\Pi}), \xi = 1, \ldots, K \text{ and } \mathbf{z} \in \mathbb{R}^{|\mathcal{S}|}, \mathbf{z} = \min(\mathbf{1}, \sum_{\xi=1}^{K} \bar{z}_\xi). \quad (7)$$

We finish with a benchmark model based on the existing framework (Szegedy et al., 2013; Molnar, 2019) of generating adversarial samples that is designed to solve inverse classification with a minimal cost of perturbation. This model optimizes over samples in $\hat{\mathbf{x}}_j \in \mathcal{S}$ to minimize the loss function $l_j(\hat{\mathbf{x}}) = KL(\bar{\mathbf{y}}_j || f(\hat{\mathbf{x}}_j)) + a\ ||\hat{\mathbf{x}}_j - \mathbf{x}_j||_2^2,\ a \in [0, \infty)$ where KL denotes the Kullback-Leibler divergence. The model (KL) reads

$$\begin{aligned} \min_{\hat{\mathbf{x}}} \quad & \sum_{j=1}^{|\mathcal{S}|} l_j\ (\hat{\mathbf{x}}) \\ \text{s.t.} \quad & g_i(\hat{\mathbf{x}}) \leq 0, i = 1, \ldots, p, \\ & \bar{h}_j(\hat{\mathbf{x}}) \leq 0, j = 1, \ldots, |\mathcal{S}| \end{aligned} \quad (8)$$

where each g and $\bar{h}$ is a nonlinear function associated with budget (2) and prediction confidence constraints (3) without the binary variable $\mathbf{z}$. This model has a smaller number of decision variables than our models since it does not include binary variables or any possible related variables; however, it does not explicitly count the number of perturbed samples and thus it has a different objective. KL does not necessarily achieve the maximal number of successfully perturbed samples.

4. Algorithms

In this section, we describe algorithms based on Lagrangian and subgradient methods. We first reformulate the problem as an unconstrained problem by using Lagrangian multipliers. We develop algorithms to solve Lagrangian functions based on the projected subgradient method. Projection is used to keep Lagrangian multipliers positive during updates or to maintain probability requirements. For chance max samples models, we apply the Gumbel trick

(Maddison et al., 2014) to use approximate gradients when updating the binary variables.

Algorithm for max samples model. We first define the Lagrangian function for MS (1) as

$$\begin{aligned} L(\mathbf{z}, \hat{\mathbf{X}}, \lambda, \mu) &= \sum_{j=1}^{|\mathcal{S}|} z_j - \sum_{i=1}^{p} \lambda_i g_i(\mathbf{z}, \hat{\mathbf{x}}) - \sum_{j=1}^{|\mathcal{S}|} \mu_j h_j(\mathbf{z}, \hat{\mathbf{x}}) \\ &= \sum_{j=1}^{|\mathcal{S}|} z_j - \sum_{i=1}^{p} \lambda_i \big(\sum_{j=1}^{|\mathcal{S}|} z_j \, ||\hat{x}_{ij} - x_{ij}||_2^2 - B_i \big) - \sum_{j=1}^{|\mathcal{S}|} \mu_j z_j \bar{h}_j(\hat{\mathbf{x}}_j) \\ &= \sum_{j=1}^{|\mathcal{S}|} c_j z_j + \sum_{i=1}^{p} \lambda_i B_i \end{aligned}$$

where

$$c_j = 1 - \sum_{i=1}^{p} \lambda_i \, ||\hat{x}_{ij} - x_{ij}||_2^2 - \mu_j \bar{h}_j(\hat{\mathbf{x}}_j) \tag{9}$$

and λ and μ are Lagrangian multipliers. We propose Algorithm 1 to solve $\min_{\lambda,\mu} \max_{\mathbf{z},\hat{\mathbf{X}}} L(\mathbf{z}, \hat{\mathbf{X}}, \lambda, \mu)$. The algorithm consists of two main loops to solve the min max problem; the inner loop updates input features and binary variables $\mathbf{z}$ to maximize L, and the outer loop updates Lagrangian multipliers to minimize L. In the algorithm, we initialize all z with one as we aim to achieve as many successfully perturbed samples as possible. Meanwhile, we add a line to break the inner loop when all entries of $\mathbf{z}$ are zero, which is the case of no updates on variables. Note that $\nabla_{\lambda_i} L = -\sum_{j=1}^{|S|} z_j \, ||\hat{x}_{ij} - x_{ij}||_2^2 + B_i$, and $\nabla_{\mu_j} L = -z_j \bar{h}_j(\hat{\mathbf{x}}_j)$. In addition, lines 7-13 in Algorithm 1 are derived by solving

$$\max_{\mathbf{z}} \quad \sum_{j=1}^{|\mathcal{S}|} c_j z_j$$

where c_j is assumed to be constant in this part of the algorithm.

Algorithm 1 MS

Initialize $\lambda, \mu, \gamma, \eta$.
while until convergence **do**
  $\mathbf{z} \leftarrow 1$
  while until convergence **do**
    Break **if** $\mathbf{z} = 0$
    while until convergence **do**
      $\hat{\mathbf{X}}^* \leftarrow \underset{\hat{\mathbf{X}}}{\text{argmax}}\, L(\mathbf{z}, \hat{\mathbf{X}}, \lambda, \mu)$
    end while
    for $j = 1, \ldots, |\mathcal{S}|$ **do**
      if $c_j \geq 0$ **then**
        $z_j \leftarrow 1$
      else
        $z_j \leftarrow 0$
      end if
    end for
  end while
  $\lambda_i \leftarrow \big(\lambda_i - \gamma\, \nabla_{\lambda_i} L\big)^+, i = 1, \ldots, p$
  $\mu_j \leftarrow \big(\mu_j - \eta\, \nabla_{\mu_j} L\big)^+, j = 1, \ldots, |\mathcal{S}|$
end while

Algorithm for Bernoulli and Categorical chance max samples model. We define the Lagrangian function for BCMS (6) as

$$
\begin{aligned}
L(\mathit{\Pi}, \hat{\mathbf{X}}, \lambda, \mu) = &\sum_{j=1}^{|\mathcal{S}|} \pi_j \big(1 - \mu_j \bar{h}_j(\hat{\mathbf{x}}_j)\big) \\
&+ \sum_{i=1}^{p} \lambda_i \Big[\Pr\big(\sum_{j=1}^{|S|} z_j \, ||\hat{x}_{ij} - x_{ij}||_2^2 - B_i \leq 0 \big) - (1 - \epsilon) \Big].
\end{aligned}
\tag{10}
$$

We need to solve $\min\limits_{\lambda,\mu} \max\limits_{\mathit{\Pi},\hat{\mathbf{X}}} L$ where we have to compute gradients of $\mathbb{E}_{\mathbf{z}\sim \text{Ber}(\mathit{\Pi})}$ with respect to $\mathit{\Pi}$. To relieve the burden of computing exact gradients with respect to discrete distributions, we apply the Gumbel trick (Maddison et al., 2014) to use their approximation. Especially, in our paper we derived our distribution based on a Gumbel Softmax distribution introduced by Jang et al. (2017) where the proposed continuous distribution can approximate any discrete variables, and through the reparameterization trick their gradients can be computed easily. Jang et al. (2017) show that this approach is superior for

approximating the sampling process for categorical variables to conventional single-sample gradient approaches. In addition, the Gumbel Softmax distribution allows applying standard backpropagation. Let the exact probability be $\Pr_i = \mathbb{E}_{\mathbf{z}}\mathcal{X}(\sum_{j=1}^{|S|} z_j ||\hat{x}_{ij} - x_{ij}||_2^2 - B_i \leq 0)$ where $\mathcal{X}$ is the indicator function. We first approximate $\mathcal{X}(\mathbf{x}) \approx (1 + \exp(-\kappa\frac{\mathbf{x}-\tau}{1-\tau}))^{-1}$ where κ and τ are hyperparameters. We have $\Pr_i \approx \mathbb{E}_{\mathbf{z}}(1 + \exp(-\kappa\frac{\mathbf{x}-\tau}{1-\tau}))^{-1}$ with $\mathbf{x} = \sum_{j=1}^{|S|} z_j ||\hat{x}_{ij} - x_{ij}||_2^2 - B_i$. Let us consider first the Bernoulli case. Applying the Gumbel trick, the expectation term is approximately computed as

$$\Pr_i \approx \frac{1}{N}\sum_{n=1}^{N} \mathrm{P}_n^i = \frac{1}{N}\sum_{n=1}^{N}(1 + \exp(-\kappa\frac{\mathbf{x}_n^i - \tau}{1-\tau}))^{-1} \tag{11}$$

where

$$\mathbf{x}_n^i = \sum_{j=1}^{|S|} v_{nj}\, ||\hat{x}_{ij} - x_{ij}||_2^2 - B_i \text{ and}$$

$$v_{nj} = \bar{z}_j(\pi_j, g_1^n, g_2^n) = \frac{\exp((\log\pi_j + g_1^n)/\omega)}{\exp((\log\pi_j + g_1^n)/\omega) + \exp((\log(1-\pi_j) + g_2^n)/\omega)}$$

and $g_1^n \sim \mathcal{G}, g_2^n \sim \mathcal{G}$. Here the Gumbel distribution is denoted by $\mathcal{G}$ and ω and N are hyperparameters. Values $\bar{z}_j$ approximate z_j. We can rewrite (10) as

$$L \approx \sum_{j=1}^{|\mathcal{S}|} \pi_j\big(1 - \mu_j\bar{h}_j(\hat{\mathbf{x}}_j)\big) + \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{p} \lambda_i \mathrm{P}_n^i - (1-\epsilon)\sum_{i=1}^{p}\lambda_i. \tag{12}$$

We propose Algorithm 2 to solve $\min_{\lambda,\mu}\max_{\mathit{\Pi},\hat{\mathbf{X}}} L$. This algorithm has two main loops to maximize L with respect to $\mathit{\Pi}$ and $\hat{\mathbf{X}}$, and to minimize L with respect to the Lagrangian multipliers. A part of generating Gumbel's samples is added to the inner loop so that approximated gradients are used to update variables. In the algorithm we use

$$\ddot{L} = \sum_{j=1}^{|\mathcal{S}|} \pi_j\big(1 - \mu_j\bar{h}_j(\hat{\mathbf{x}}_j)\big) \tag{13}$$

For CCMS, we define the Lagrangian function based on (5), which is written

Algorithm 2 BCMS

Initialize $\mathit{\Pi}, \lambda, \mu, \alpha, \beta, \gamma, \eta$.
while until convergence **do**
  while until convergence **do**
    for $n = 1, \ldots, N$ **do**
      for $j = 1, \ldots, |\mathcal{S}|$ **do**
        $g_1^n \sim \mathcal{G}_j, g_2^n \sim \mathcal{G}_j$
        $v_{nj} \leftarrow \bar{z}_j(\pi_j, g_1^n, g_2^n)$
      end for
      $\dot{L}_n \leftarrow \sum_{i=1}^{p} \lambda_i \mathrm{P}_n^i$ with P_n^i as in (11)
    end for
    $\nabla_{\mathit{\Pi}} L \leftarrow \frac{1}{N} \sum_{n=1}^{N} \nabla_{\mathit{\Pi}} \dot{L}_n + \nabla_{\mathit{\Pi}} \ddot{L}$ with $\ddot{L}$ as in (13)
    $\mathit{\Pi} \leftarrow \min\{1, \big(\mathit{\Pi} + \alpha \nabla_{\mathit{\Pi}} L\big)^+\}$
    $\hat{\mathbf{X}} \leftarrow \hat{\mathbf{X}} + \beta \nabla_{\hat{\mathbf{X}}} L$
  end while
  $\lambda_i \leftarrow \big(\lambda_i - \gamma \nabla_{\lambda_i} L\big)^+, i = 1, \ldots, p$
  $\mu_j \leftarrow \big(\mu_j - \eta \nabla_{\mu_j} L\big)^+, j = 1, \ldots, |\mathcal{S}|$
end while

as

$$L(\mathit{\Pi}, \hat{\mathbf{X}}, \mathit{\Lambda}, M) = \mathbb{E}_{\mathbf{z}}\Bigg[\sum_{j=1}^{|\mathcal{S}|} z_j\big(1 - \mu_j \bar{h}_j(\hat{\mathbf{x}}_j)\big) + \sum_{i=1}^{p} \lambda_i \Bigg[\Pr\Big(\sum_{j=1}^{|S|} z_j \, ||\hat{x}_{ij} - x_{ij}||_2^2 - B_i \leq 0\Big) - (1 - \epsilon)\Bigg]\Bigg]$$

where z_j is j^{th} element of $\mathbf{z}$ as defined in (7). Based on the Gumbel's approach we approximate $\mathbf{z}$ by

$$[\bar{z}_\xi]_j = \frac{\exp((\log\pi_j + g_{j\xi})/\omega)}{\sum_{k=1}^{|\mathcal{S}|} \exp((\log\pi_k + g_{k\xi})/\omega)} \text{ where } g_{1\xi}, \ldots, g_{|\mathcal{S}|\xi} \sim \mathcal{G}, \xi = 1, \ldots, K.$$

The approximate Lagrangian function for CCMS reads

$$
\begin{aligned}
L \approx \mathbb{E}_{\mathbf{G}\sim\mathcal{G}}\Bigg[& \sum_{j=1}^{|\mathcal{S}|} \bar{z}_j(\mathit{\Pi}, \mathbf{G})\big(1-\mu_j \bar{h}_j(\hat{\mathbf{x}}_j)\big) \\
& + \sum_{i=1}^{p} \lambda_i \Bigg[\Pr\big(\sum_{j=1}^{|\mathcal{S}|} \bar{z}_j(\mathit{\Pi}, \mathbf{G})\, ||\hat{x}_{ij} - x_{ij}||_2^2 - B_i \le 0\big) - (1-\epsilon) \Bigg] \Bigg] \\
\approx & \frac{1}{N}\sum_{n=1}^{N}\sum_{j=1}^{|\mathcal{S}|} v_{nj}\big(1-\mu_j\bar{h}_j\big) + \frac{1}{N}\sum_{n=1}^{N}\sum_{i}^{p}\lambda_i \mathrm{P}_n^i - (1-\epsilon)\sum_{i=1}^{p}\lambda_i \\
= & \frac{1}{N}\sum_{n=1}^{N}\tilde{L}_n + \frac{1}{N}\sum_{n=1}^{N}\dot{L}_n - (1-\epsilon)\sum_{i=1}^{p}\lambda_i
\end{aligned}
\tag{14}
$$

where $\mathbf{G} \in \mathbb{R}^{|\mathcal{S}|\times K}$ is a Gumbel matrix, and P_n^i is the same as in (11) except that

$$
v_{nj} = \bar{z}_j(\mathit{\Pi}, \mathbf{G}^n) = \min(1, \sum_{\xi=1}^{K} \frac{\exp((\log\pi_j + g_{j\xi}^n)/\omega)}{\sum_{k=1}^{|\mathcal{S}|}\exp((\log\pi_k + g_{k\xi}^n)/\omega)}).
$$

We propose Algorithm 3 to solve $\min\limits_{\lambda,\mu}\max\limits_{\mathit{\Pi},\hat{\mathbf{x}}} L$, which has the same structure as Algorithm 2 for BCMS. The part of simulating Gumbel's samples is edited for the Categorical distribution (lines 4-13).

Algorithm for KL. The Lagrangian function for KL (8) reads

$$
L(\hat{\mathbf{X}}, \lambda, \mu) = \sum_{j=1}^{|\mathcal{S}|} l_j\,(\hat{\mathbf{x}}_j, \bar{\mathbf{y}}_j) + \sum_{i=1}^{p} \lambda_i\, g_i(\hat{\mathbf{x}}) + \sum_{j=1}^{|\mathcal{S}|} \mu_j\, \bar{h}_j(\hat{\mathbf{x}}) \tag{15}
$$

where λ and μ are Lagrangian multipliers. Algorithm 4 is designed to solve $\max\limits_{\lambda,\mu}\min\limits_{\hat{\mathbf{X}}} L$. Similar to Algorithm 1, it consists of two loops; the inner loop updates input features to minimize L, and the outer loop updates Lagrangian multipliers to maximize L.

Obtaining the final solution. Since Lagrangian methods do not necessarily guarantee feasibility of solutions $\hat{\mathbf{X}}$ with respect to budget and prediction confidence, we conduct the following steps to obtain the final solution set.

1. Run one of the Algorithms 1-4 to obtain a 'good' but possibly infeasible

Algorithm 3 CCMS

Initialize $\mathit{\Pi}, \lambda, \mu, \alpha, \beta, \gamma, \eta$.
while until convergence **do**
  while until convergence **do**
    for $n = 1, \ldots, N$ **do**
      for $j = 1, \ldots, |\mathcal{S}|$ **do**
        for $\xi = 1, \ldots, K$ **do**
          $[\mathbf{G}^n]_{j\xi} \leftarrow g^n_{j\xi} \sim \mathcal{G}$
        end for
      end for
      $v_{nj} \leftarrow \bar{z}_j(\mathit{\Pi}, \mathbf{G}^n), j = 1, \ldots, |\mathcal{S}|$
      $\tilde{L}_n \leftarrow \sum_{j=1}^{|\mathcal{S}|} v_{nj}\big(1 - \mu_j \bar{h}_j\big)$
      $\dot{L}_n \leftarrow \sum_{i=1}^{p} \lambda_i \mathrm{P}^i_n$
    end for
    $\nabla_{\mathit{\Pi}} L \leftarrow \frac{1}{N} \sum_{n=1}^{N} (\nabla_{\mathit{\Pi}} \tilde{L}_n + \nabla_{\mathit{\Pi}} \dot{L}_n)$
    $\mathit{\Pi} \leftarrow \big(\mathit{\Pi} + \alpha \, \nabla_{\mathit{\Pi}} L\big)^+$
    $\hat{\mathbf{X}} \leftarrow \hat{\mathbf{X}} + \beta \, \nabla_{\hat{\mathbf{X}}} L$
  end while
  $\lambda_i \leftarrow \big(\lambda_i - \gamma \, \nabla_{\lambda_i} L\big)^+, i = 1, \ldots, p$
  $\mu_j \leftarrow \big(\mu_j - \eta \, \nabla_{\mu_j} L\big)^+, j = 1, \ldots, |\mathcal{S}|$
end while

Algorithm 4 KL

Initialize $\lambda, \mu, \gamma, \eta$
while until convergence **do**
  while until convergence **do**
    $\hat{\mathbf{X}} \leftarrow \underset{\hat{\mathbf{X}}}{\operatorname{argmin}} \, L(\hat{\mathbf{X}}, \lambda, \mu)$
  end while
  $\lambda_i \leftarrow \big(\lambda_i + \gamma \, \nabla_{\lambda_i} L\big)^+, i = 1, \ldots, p$
  $\mu_j \leftarrow \big(\mu_j + \eta \, \nabla_{\mu_j} L\big)^+, j = 1, \ldots, |\mathcal{S}|$
end while

solution $\hat{\mathbf{X}}$.

2. Find a subset $\hat{\mathcal{S}} \subseteq \mathcal{S}$ of samples satisfying prediction confidence constraints, i.e., all of the samples in $\hat{\mathcal{S}}$ are classified as desired.

3. Solve the following problem over $\hat{\mathcal{S}}$ to find the final set $\tilde{\mathcal{S}}$ of feasible samples:

$$\begin{aligned} \max_{z_1,\ldots,z_{|\hat{\mathcal{S}}|}} \quad & \sum_{s=1}^{|\hat{\mathcal{S}}|} z_s \\ \text{s.t.} \quad & \sum_{s=1}^{|\hat{\mathcal{S}}|} a_{is}\, z_s \leq B_i, i = 1, \ldots, p \\ & z_s \in \{0,1\}, s = 1, \ldots, |\hat{\mathcal{S}}|, \end{aligned} \tag{16}$$

where $a_{is} = ||\hat{x}_{is} - x_{is}||_2^2$.

Sequences. Let us consider the case when samples are sequences of varying length of feature vectors of the same length; e.g., f corresponds to an LSTM or transformer. In this case, $\hat{\mathbf{x}}$ is not well defined, i.e. it is not a matrix and thus (1) is ill-posed. If we consider only sequences of the same length, then (1) is well defined. If we have R different lengths, then we can form R different disjoint subsets $\mathcal{S}^r$ of samples and define (1) for each one subset. The link between all subsets becomes a joint per-feature budget. This budget needs to be allocated to the R problems. Herein we use a simple strategy of allocating $\frac{|\mathcal{S}^r|}{|\mathcal{S}|}\, B_i$ for each feature i (note that $|\mathcal{S}| = \sum_{r=1}^{R} |\mathcal{S}^r|$).

5. Computational study

In this section, we conduct a computational study on two datasets: a proprietary dataset and a public dataset. We experiment with different budget scenarios and we assess scalability with respect to the number of samples. Model implementations for all the experiments are done in Python using Tesla V100 GPU and Intel Xeon CPU E5-2697 v4 @ 2.30Hz for the real-world dataset, and Titan XP 1080 GPU and Intel Xeon Silver 4112 CPU @ 2.60GHz for the public dataset.

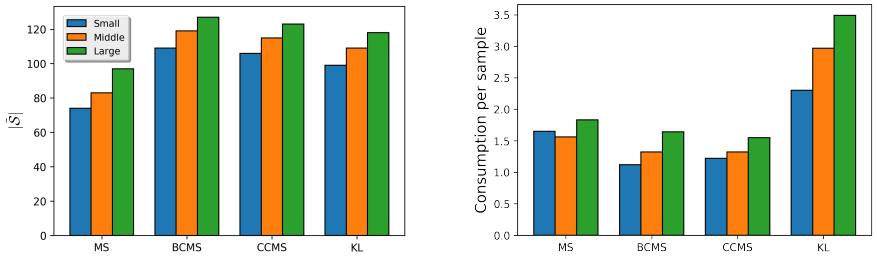
We use the following hyperparameter values: $\delta = 0.1$, $\kappa = 2$, $\tau \in \{30, 50\}$, $\omega = 1$, and $N = 100$. The learning rates α and β affecting $\hat{\mathbf{x}}$ and $\mathbf{z}$ are selected as one of $\{0.01, 0.05, 0.1\}$. We use a decaying learning rates γ and η, affecting the Lagrangian multipliers λ and μ, initially set as one in $\{0.5, 1\}$. The stopping criterion is set to be the maximum number of iterations (variable updates). For the outer loop it is set to be 10 for MS, BCMS, and KL and 20 for CCMS, and for the inner loop it is set to be 10, 100, 100, and 5,000 for MS, BCMS, CCMS and KL, respectively. The initial Lagrangian multipliers are selected as one from $\{1, 10\}$ but adding white Gaussian noise.

5.1. Real-world data

We conduct experiments on a real-world proprietary dataset that contains sequential input features for 5 classes, which is introduced in (Stec et al., 2018). The data has 169 features and sequences are of size from 1 to 150 and thus the joint per-feature budget needs to be employed. The classification model Sparse Time LSTM from (Stec et al., 2018) is selected as our classifier f. The accuracy of the model on approximately 700,000 training samples is around 70%.

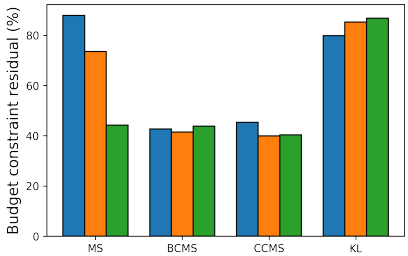
5.1.1. Budget experiments

We perturb 300 samples selected from the test set which are grouped into 5 different groups by input sequence length. Thus, we have $|\mathcal{S}| = 300$, and $R = 5$. The 300 samples are correctly predicted by the trained classifier into a "negative" class - four of the five classes - (e.g. have disease) and thus the perturbed samples should fall into the positive class - the remaining class - (e.g. does not have the disease). We perturb 19 features since the remaining features cannot be altered in practice. To decide the sizes of the budgets, we first run Algorithm 4 with unlimited budgets to measure how much budget is needed for successful perturbation. Then, as well as based on practical considerations from subject matter experts and data stockholders, we determine small, middle, and large sizes of budgets by the amounts that are proportional to the total budget consumption with the unlimited budgets.

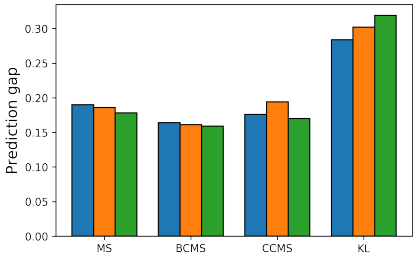


(a) Size of $\tilde{\mathcal{S}}$

(b) Consumption per sample average



(c) Budget constraint residual



(d) Prediction gap average

Figure 2: Real-world data: Budget experiment

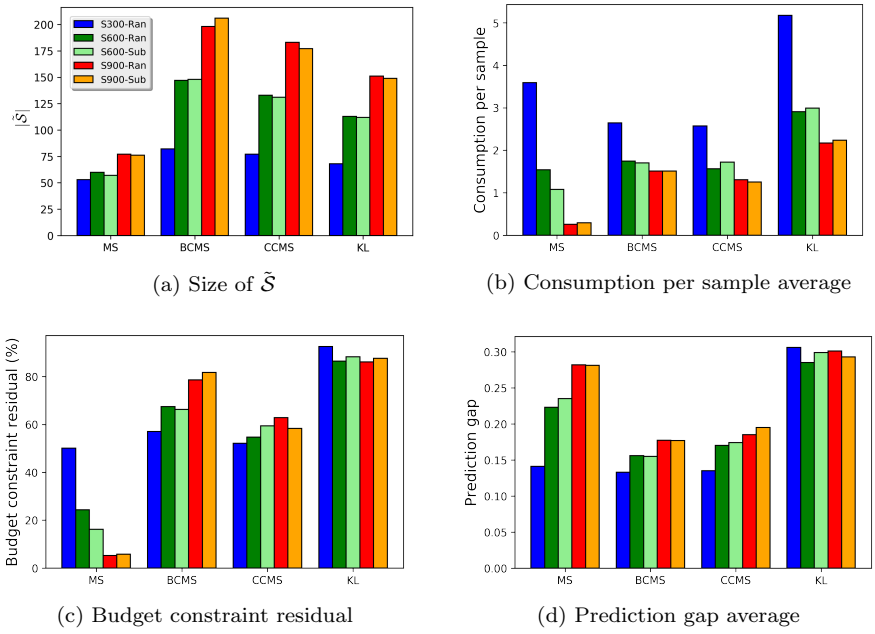
Figure 2 shows the results of the budget experiment. We find that algorithms with the Gumbel's method BCMS and CCMS perform better than other algorithms. They achieve a larger size of successfully perturbed samples than other algorithms, and they also achieve lower consumption per sample defined as the budget used by all of the samples divided by $|\tilde{S}|$. The relative improvement of BCMS and CCMS over KL is 10% and 7%, 9% and 6%, and 8% and 4% for small, middle, and large budget, respectively. This is because the objective of the max samples models is to maximize the number of successfully perturbed samples. In addition, in plot (a) we observe that a larger budget achieves a larger size of successfully perturbed samples for all algorithms, which is expected. We also analyze budget and prediction confidence constraints. For budget constraint residuals defined by $-g_i$ in (2) divided by the total available budget, we compute how much of the budget is spent for each budget constraint, and calculate the mean of them, see plot (b). In addition, a prediction gap is computed by measuring the gap between the top and the second best predic-

tion probabilities, see plot (d) for average prediction gaps. We find that budget constraint residuals of BCMS and CCMS are lower than those from other algorithms, and their prediction gaps are smaller than the others. We reason this as BCMS and CCMS spend budgets large enough to guarantee a certain level of prediction confidence; it is larger than δ in their predictions, but not more than necessary. On the other hand, KL has a large prediction gap that shows high confidence in its prediction which is more than necessary. This is a reason their budget residual per sample is relatively large.

5.1.2. Scalability experiments

We also conduct a scalability analysis of our algorithms. We use three different sizes of samples $|\mathcal{S}| = 300, 600$, and 900, with samples in each set being grouped into $R = 15$ based on their input sequence length. In addition, they have inclusive relationships such that $\mathcal{S}_{300} \subset \mathcal{S}_{600} \subset \mathcal{S}_{900}$. In this context, we have two strategies of initializing samples to be perturbed. First, we initialize input features of samples in the larger set with previously obtained values from the subset, and the rest of samples that are not in the subset are initialized randomly. For example, we run an algorithm on $\mathcal{S}_{300}$, and then run the algorithm on $\mathcal{S}_{600}$. When we run it on $\mathcal{S}_{600}$, we initialize samples from $\mathcal{S}_{300}$ in $\mathcal{S}_{600}$ with obtained values from the run on $\mathcal{S}_{300}$, and samples from $\mathcal{S}_{600} \setminus \mathcal{S}_{300}$ randomly. The other strategy is to initialize all samples randomly. Similar to the budget experiments, all samples are originally labeled as one of negative classes and correctly predicted by the trained classifier. We use the middle size budget and the other hyperparameters are the same as those used in the budget experiments.

Figure 3 shows the results of the scalability experiment. Note that a run on $\mathcal{S}_{300}$ with subset initialization is denoted by 300-Sub, and one with random initialization is denoted by 300-Ran in the figure. Similar to the budget experiments, the algorithms with Gumbel's method BCMS and CCMS perform better than other algorithms. The count of samples successfully perturbed by them is larger than the counts from other algorithms, and also the average consumption



(a) Size of $\tilde{\mathcal{S}}$

(b) Consumption per sample average

(c) Budget constraint residual

(d) Prediction gap average

Figure 3: Real-world data: Scalability experiment

per sample is smaller. The relative improvement of the stochastic models BCMS and CCMS over KL is 21% and 13% for $|\mathcal{S}| = 300$, 31% and 17% for $|\mathcal{S}| = 600$, and 35% and 20% for $|\mathcal{S}| = 900$ on average over different initialization strategies. The observations from Section 5.1.1 apply to each individual sample size. We find that the relative improvement of BCMS and CCMS over KL increases linearly as sample size increases in Figure 4. In terms of the two initialization strategies, both cases show similar results and thus the benefits of warm-start are negligible. Regarding the budget and prediction confidence constraints, we find similar results to the budget experiments. Budget constraint residuals for BCMS and CCMS are lower than KL, and their prediction gaps are smaller than for the other algorithm. The aforementioned conclusions apply to all of different sizes of samples and thus we conclude that our algorithms scale efficiently.

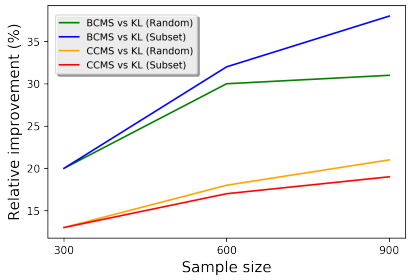


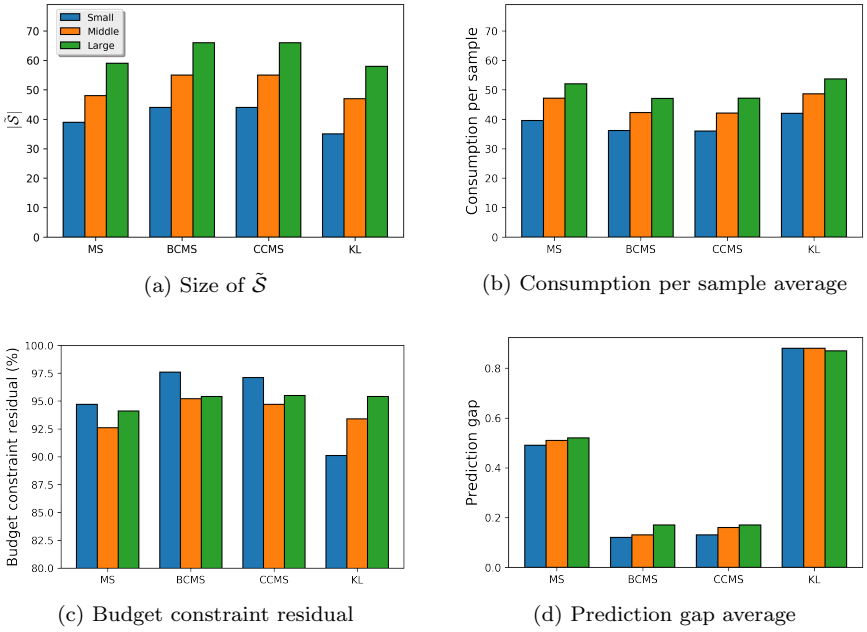
Figure 4: Real-world data: Relative improvement as the sample size increases

5.2. The MIMIC healthcare dataset

MIMIC is a public dataset that describes clinical information of patients admitted to an Intensive Care Unit at the Beth Israel Deaconess Medical Center in Boston, Massachusetts from 2001 to 2012. It contains 58,576 samples for patient admissions. Descriptive statistics can be found in (Johnson et al., 2016; Goldberger et al., 2000). In this study, we use 13 input features corresponding to the health state for 30-day mortality predictions used in (Luo et al., 2018). Since MIMIC is a time series dataset and has missing values, we use a recurrent network based on Gated Recurrent Unit, called GRU-D, that is widely used to deal with multivariate time series with missing values for imputation and predictions (Che et al., 2018). Its AUC is around 0.78 which is comparable to state-of-the-art. We conduct only a budget experiment for this dataset due to its limited size.

We perturb 75 samples selected from the test set, i.e., $|\mathcal{S}| = 75$. The 75 samples are originally labeled as "dead" and correctly predicted by the trained classifier. Our purpose is to perturb the samples so that they are predicted as "alive." To decide the size of budgets which represent the per drug amount, we first run Algorithm 4 with unlimited budget constraints to measure how much perturbation is needed. Then, we determine small, middle, and large sizes of budgets by imposing 40%, 60%, and 80% of the total budget achieved by unlimited budgets.

Figure 5 shows the results of the budget experiment. Similar to the results



(a) Size of $\tilde{\mathcal{S}}$

(b) Consumption per sample average

(c) Budget constraint residual

(d) Prediction gap average

Figure 5: MIMIC: Budget experiment

on the real-world data, stochastic algorithms for max samples models, BCMS and CCMS perform better than KL and MS. They obtain a larger size of successfully perturbed samples than the other algorithms, and also achieve smaller consumption per sample. The relative improvement of our max samples models MS, BCMS and CCMS over KL is 5%, 19% and 19% on average for all different budget scenarios. It is interesting to observe that KL uses a much bigger portion of the budget than the other algorithms.

5.3. A diabetes healthcare dataset

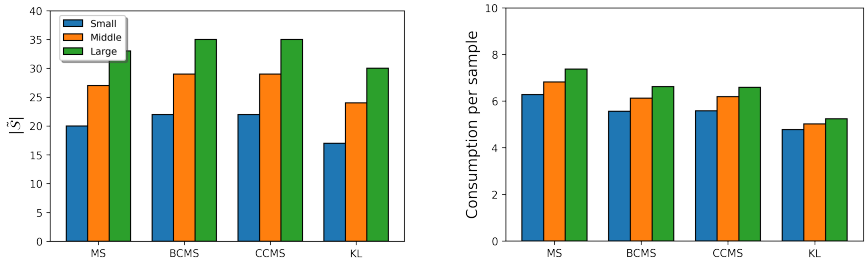
We experiment with another public dataset that describes clinical information of diabetes patients, Diabetes 130-US hospitals for years 1999-2008 Data Set from the UCI Machine Learning Repository (Dua & Graff, 2017). It contains 100,000 samples and 50 features representing patient and hospital outcomes, which were collected over 10 years at 130 US hospitals and integrated

delivery networks. In this study, we use a subset of the total data provided by (Li, 2018). It has 769 samples without missing values and eight input features corresponding to the health state for diabetes patient predictions. We adopt a feed-forward neural network as a classifier, which is the best model reported in (Li, 2018). The neural network has four hidden layers with 100 units and the Rectified Linear Unit is used as the activation function. The accuracy of the model on the test set is around 80%. We conduct only a budget experiment for this dataset due to its limited size.

We perturb 45 samples selected from the test set, i.e., $|\mathcal{S}| = 45$. The 45 samples are originally labeled as "diabetes" and correctly predicted by the fitted classifier. The goal is to perturb the samples so that they are predicted as "no diabetes." We perturb six features since the remaining features such as "Pregnancy" and "Age" cannot be changed in practice. To decide the size of budgets for each feature, we run algorithms with unlimited budget constraints to measure the resulting budgets. Then, we determine small, middle, and large sizes of budgets by imposing 40%, 60%, and 80% of the budget used by unlimited budgets. Figure 6 shows the results of the budget experiment. Similar to the results on the other two datasets, the stochastic algorithms for max samples models, BCMS and CCMS perform better than KL and MS. They obtain a larger size of successfully perturbed samples than the other algorithms with tight bindings of the budget constraints from (a) and (c) in Figure 6. The relative improvement of our max samples models MS, BCMS, and CCMS over KL is 13%, 22%, and 22%, respectively on average for all different budget scenarios.

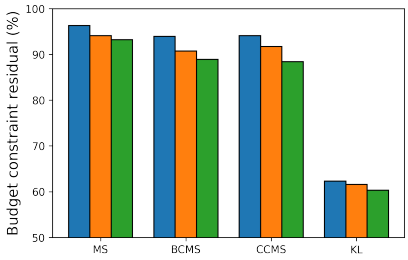
5.4. Fitness of the classifier

Inverse classification requires a well-trained classifier since the trained classifier is used to determine whether candidate samples are successfully perturbed as desired. Especially, in our study we are only interested in correctly classified samples. We use the best classifier as reported in the previous literature for each dataset under consideration; the real-world data in Stec et al. (2018), MIMIC in Che et al. (2018), and the diabetes healthcare dataset in Li (2018). For the

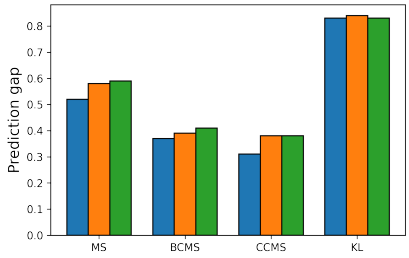


(a) Size of $\tilde{\mathcal{S}}$

(b) Consumption per sample average



(c) Budget constraint residual



(d) Prediction gap average

Figure 6: Diabetes: Budget experiment

real-world data, the Sparse Time LSTM model is selected as our classifier that can deal with both static and sequential features efficiently as proposed in Stec et al. (2018). It obtains 70% test accuracy. For MIMIC, a recurrent network based on Gated Recurrent Unit (GRU-D) that is proposed for coping with multivariate time series with missing values is selected since it is reported as the best model in Che et al. (2018) among Logistic regression (LR), Random Forest (RF), and Support Vector Machine (SVM). The proposed GRU-D as the best performer obtains the AUC of 0.78. For the diabetes healthcare dataset, a feed-forward network (NN) is chosen as it is reported in Li (2018) that NN performs the best among k-Nearest Neighbors, Decision tree, Gradient boosting, LR, RF, and SVM. The best NN with four hidden layers achieves 80% accuracy on the test dataset.

5.5. Optimization aspects

In this part we discuss optimization aspects of the algorithms. Figure 7 shows sizes of $\tilde{\mathcal{S}}$ (successfully perturbed samples) for each algorithm at each outer iteration. We observe that the number of $\tilde{\mathcal{S}}$ increases with iterations and stays at the highest point, which implies that the algorithms converge to a (local) optimal value. We also find that BCMS converges faster than CCMS in all cases. We also note that KL converges faster than CCMS, however, the largest $|\tilde{\mathcal{S}}|$ from KL is not larger than the one from CCMS and BCMS.

Figure 9 shows the values of Lagrangian multipliers at each outer iteration for the MIMIC dataset. The average values of λ with respect to the budget constraint decrease as each algorithm iterates, which implies that the amount of constraint violation decreases. Regarding μ that is associated with the prediction confidence constraint, KL values decrease as the algorithm proceeds while the max sample algorithms drive them higher. In Figure 8 we further observe that the norm of gradients of μ for the max samples algorithms shrink to zero, which implies that the algorithms reduce the amount of constraint violation. MS shows erratic behaviors in some cases and its performance is unstable. We reason this is due to the presence of constraints and binary variables in MS.

5.6. Complexity analysis

Here, we discuss complexity of the algorithms. Given the number of inner and outer loops, I^{in} and I^{out}, the dimension of the input feature vector p, and the number of samples $|\mathcal{S}|$, the number of total iterations of Algorithm 1 (MS) is $I^{\text{out}}(I^{\text{in}}(I^{\hat{X}} + |\mathcal{S}|) + p + |\mathcal{S}|)$, Algorithm 2 (BCMS) requires $I^{\text{out}}(I^{\text{in}}(N|\mathcal{S}|) + p + |\mathcal{S}|)$, Algorithm 3 (CCMS) needs $I^{\text{out}}(I^{\text{in}}(N|\mathcal{S}|K) + p + |\mathcal{S}|)$, and Algorithm 4 (KL) has $I^{\text{out}}(I^{\text{in}} + p + |\mathcal{S}|)$ steps where $I^{\hat{X}}$ is the number of iterations to solve argmax for MS, and N and K are the number of samples and categories used for BCMS and CCMS. Algorithms MS, BCMS, and CCMS require a larger number of iterations than KL if I^{in} and I^{out} are the same. However, we found that runtimes of BCMS and CCMS are shorter. Table 1 shows the actual runtime of the algorithms on the larger, real-world dataset. The maximum number of

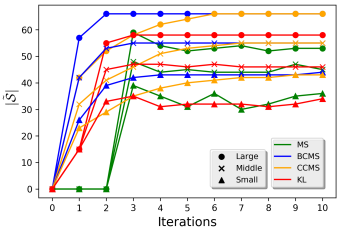
iterations is set large enough to ensure the algorithms converge as discussed in Section 5.5 (Figure 7). Algorithms for stochastic models BCMS and CCMS take longer to update variables per iteration in the inner loop than KL requires since the Gumbel's simulation is implemented. However, the total runtime of BCMS and CCMS does not necessarily take longer than others. Based on the runtime, and the convergence and size of $\tilde{\mathcal{S}}$ we conclude that BCMS is the best performer among all the algorithms considered herein. Algorithms CCMS is a close second.

Algorithm	Runtime per iteration (No. of max iterations)		
	Outer loop	Inner loop	Total runtime
MS	20 sec (10)	700 sec (10)	19 hr
BCMS	25 sec (10)	18 sec (100)	5 hr
CCMS	25 sec (20)	15 sec (100)	8 hr
KL	20 sec (10)	0.9 sec (5,000)	10 hr

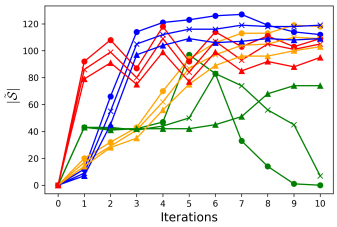
Table 1: Runtime of algorithms

6. Conclusion

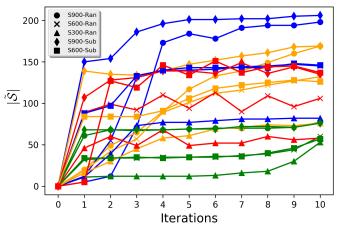
In this paper, a new framework for inverse classification is proposed. We formulate a constrained optimization problem that maximizes the number of successfully perturbed samples with budget and prediction confidence constraints. In addition, we formulate a stochastic problem with chance constraints. To solve the constrained problems, algorithms based on Lagrangian and subgradient methods are developed. Based on the computational study, we find that the algorithms perform well in various budget settings and are scalable.



(a) MIMIC: Budget experiment

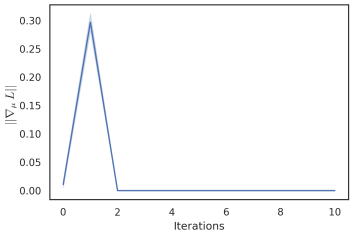


(b) Real-world data: Budget experiment

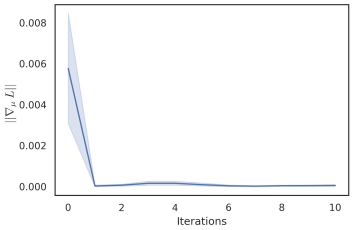


(c) Real-world data: Scalability experiment

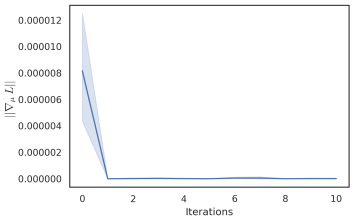
Figure 7: Size of $\tilde{\mathcal{S}}$ at each outer iteration



(a) $||\nabla_\mu L||$ of MS

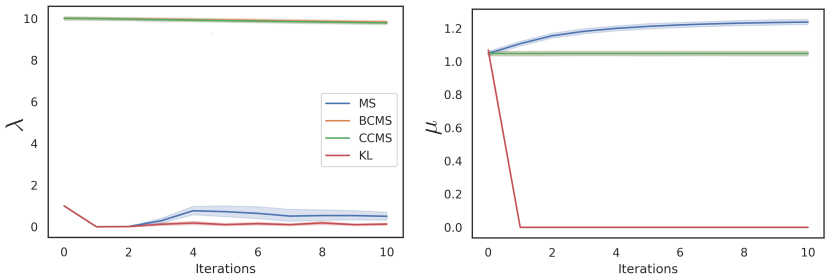


(b) $||\nabla_\mu L||$ of BCMS



(c) $||\nabla_\mu L||$ of CCMS

Figure 8: Norm of Lagrangian multipliers of max samples algorithms at each outer iteration of MIMIC



(a) λ (b) μ

Figure 9: Lagrangian multipliers at each outer iteration of MIMIC

References

Aggarwal, C. C., Chen, C., & Han, J. (2010). The inverse classification problem. *Journal of Computer Science and Technology*, *18*, 458–468.

Barbella, D., Benzaid, S., Christensen, J. M., Jackson, B., Qin, X. V., & Musicant, D. R. (2009). Understanding support vector machine classifications via a recommender system-like approach. In R. Stahlbock, S. F. Crone, & S. Lessmann (Eds.), *Proceedings of the 2009 International Conference on Data Mining, DMIN 2009, July 13-16, 2009, Las Vegas, USA* (pp. 305–311). CSREA Press.

Boylu, F., Aytug, H., & Koehler, G. J. (2010). Induction over strategic agents. *Information Systems Research*, *21*, 170–189.

Che, Z., Purushotham, S., Cho, K., Sontag, D., & Liu, Y. (2018). Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, *8*, 645–653.

Chi, C.-L., Street, W. N., & Ward, M. M. (2012). Individualized patient-centered lifestyle recommendations: An expert system for communicating patient specific cardiovascular risk information and prioritizing lifestyle options. *Journal of Biomedical Informatics*, *45*, 1164–1174.

Combey, T., Loison, A., Faucher, M., & Hajri, H. (2020). Probabilistic jacobian-based saliency maps attacks. *Machine Learning and Knowledge Extraction*, *2*, 558–578. URL: https://www.mdpi.com/2504-4990/2/4/30. doi:10.3390/make2040030.

Dong, Y., Liao, F., Pang, T., Su, H., Zhu, J., Hu, X., & Li, J. (2018). Boosting adversarial attacks with momentum. In *IEEE Conference on Computer Vision and Pattern Recognition* (pp. 1–12). IEEE.

Dua, D., & Graff, C. (2017). UCI machine learning repository. URL: http://archive.ics.uci.edu/ml.

Goldberger, A. L., Amaral, L. A. N., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C.-K., & Stanley, H. E. (2000). Physiobank, physiotoolkit, and physionet: Components of a new research resource for complex physiologic signals. *Circulation*, *101*, 2155–220.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*. URL: http://arxiv.org/abs/1412.6572.

Gupta, A., Lash, M. T., & Nachimuthu, S. K. (2021). Optimal sepsis patient treatment using human-in-the-loop artificial intelligence. *Expert Systems with Applications*, *169*, 114476. URL: https://www.sciencedirect.com/science/article/pii/S0957417420311258. doi:https://doi.org/10.1016/j.eswa.2020.114476.

Jang, E., Gu, S., & Poole, B. (2017). Categorical reparameterization with gumbel-softmax. URL: https://arxiv.org/abs/1611.01144. doi:10.48550/ARXIV.1611.01144.

Johnson, A. E. W., Pollard, T. J., Shen, L., Lehman, L., Feng, M., Ghassemi, M., Moody, B., Szolovits, P., Celi, L. A., & Mark, R. G. (2016). MIMIC-III, a freely accessible critical care database.

Kurakin, A., Goodfellow, I., & Bengio, S. (2017). Adversarial examples in the physical world. *Computing Research Repository (CoRR)*, *abs/1607.02533v4*. URL: `https://arxiv.org/abs/1607.02533`. `arXiv:1607.02533v4`.

Lash, M. T., Lin, Q., Street, W. N., & Robinson, J. G. (2017a). A budget-constrained inverse classification framework for smooth classifiers. In *IEEE International Conference on Data Mining Workshops* (pp. 1184–1193).

Lash, M. T., Lin, Q., Street, W. N., Robinson, J. G., & Ohlmann, J. (2017b). Generalized inverse classification. In *the 2017 SIAM International Conference on Data Mining* (pp. 162–170).

Lash, M. T., & Street, W. N. (2020). Personalized cardiovascular disease risk mitigation via longitudinal inverse classification. In *IEEE International Conference on Bioinformatics and Biomedicine* (pp. 2610–2617). IEEE.

Laugel, T., Lesot, M.-J., Marsala, C., Renard, X., & Detyniecki, M. (2018). Comparison-based inverse classification for interpretability in machine learning. In J. Medina, M. Ojeda-Aciego, J. L. Verdegay, D. A. Pelta, I. P. Cabrera, B. Bouchon-Meunier, & R. R. Yager (Eds.), *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Theory and Foundations* (pp. 100–111). Springer International Publishing.

Li, S. (2018). Machine learning for diabetes with python. URL: `https://datascienceplus.com/machine-learning-for-diabetes-with-python/`.

Lowd, D., & Meek, C. (2005). Adversarial learning. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery in Data Mining* (pp. 641–647).

Luo, Y., Szolovits, P., Dighe, A. S., & Baron, J. M. (2018). 3D-MICE: Integration of cross-sectional and longitudinal imputation for multi-analyte longitudinal clinical data. *Journal of the American Medical Informatics Association*, *25*, 645–653.

Machado, G. R., Silva, E., & Goldschmidt, R. R. (2021). Adversarial machine learning in image classification: A survey toward the defender's perspective. *ACM Comput. Surv.*, *55*. URL: https://doi.org/10.1145/3485133. doi:10.1145/3485133.

Maddison, C. J., Tarlow, D., & Minka, T. (2014). A* sampling. In *Proceedings of the 27th Conference on Neural Information Processing Systems* (pp. 3086–3094).

Mannino, M. V., & Koushik, M. V. (2000). The cost-minimizing inverse classification problem: A genetic algorithm approach. *Decision Support Systems*, *29*, 283–300.

Molnar, C. (2019). *Interpretable machine learning: A guide for making black box models explainable*. Online: https://christophm.github.io/interpretable-ml-book/. URL: https://christophm.github.io/interpretable-ml-book/.

Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, Z. B., & Swami, A. (2016). The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy* (p. 372–387).

Polap, D., & Wlodarczyk-Sielicka, M. (2020). Classification of non-conventional ships using a neural bag-of-words mechanism. *Sensors*, *20*. URL: https://www.mdpi.com/1424-8220/20/6/1608. doi:10.3390/s20061608.

Stec, A., Klabjan, D., & Utke, J. (2018). Unified recurrent neural network for many feature types. *ArXiv e-prints*, *abs/1809.08717*. URL: https://arxiv.org/abs/1809.08717. arXiv:1809.08717.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2013). Intriguing properties of neural networks. *Computing Research Repository (CoRR)*, *abs/1312.6199*. URL: http://arxiv.org/abs/1312.6199. arXiv:1312.6199.

Tygar, J. D. (2011). Adversarial machine learning. *IEEE Internet Computing*, *15*, 4–6.

Verma, S., Dickerson, J. P., & Hines, K. (2020). Counterfactual explanations for machine learning: A review. *CoRR*, *abs/2010.10596*. URL: https://arxiv.org/abs/2010.10596. arXiv:2010.10596.

Verma, S., Dickerson, J. P., & Hines, K. (2021). Counterfactual explanations for machine learning: Challenges revisited. *CoRR*, *abs/2106.07756*. URL: https://arxiv.org/abs/2106.07756. arXiv:2106.07756.

Wachter, S., Mittelstadt, B., & Russell, C. (2018). Counterfactual explanations without opening the black box: Automated decisions and the GDPR. *Harvard Journal of Law and Technology*, *31*, 842–887.

Yang, C., Street, W. N., & Robinson, J. G. (2012). 10-year CVD risk prediction and minimization via inverse classification. In *the 2nd ACM SIGHIT Symposium on International Health Informatics* (pp. 603–610).