Sampling from the Complement of a Polyhedron: An MCMC Algorithm for Data Augmentation

Timothy C. Y. Chan, Adam Diamant, Rafid Mahmood*

March 27, 2020

Abstract

We present an MCMC algorithm for sampling from the complement of a polyhedron. Our approach is based on the Shake-and-bake algorithm for sampling from the boundary of a set and provably covers the complement. We use this algorithm for data augmentation in a machine learning task of classifying a hidden feasible set in a data-driven optimization pipeline. Numerical results on simulated and MIPLIB instances demonstrate that our algorithm along with a supervised learning technique outperforms conventional unsupervised baselines.

1 Introduction

High-dimensional sampling is a fundamental tool that is used in various domains such as machine learning (Andrieu et al., 2003), optimization (Bertsimas and Vempala, 2004), and stochastic modeling (Ripley, 2009). Sampling from a high-dimensional set is a key component of approximation algorithms for problems that cannot be tractably solved with conventional methods.

The literature on high-dimensional sampling primarily addresses the problem of efficiently sampling points that lie within a convex set, with the family of Markov Chain Monte Carlo (MCMC) sampling methods being the most commonly used approach in this setting (Brooks et al., 2011). Recent applications in ranking have also generated interest in the related problem of sampling from the boundary of convex sets (Dieker and Vempala, 2015). However, to the best of our knowledge, there has not been prior work on sampling from the complement of a convex set.

In this paper, we consider the task of efficiently sampling from sets defined by the complement of a polyhedron for which there exist many potential applications. For example, the complement operator can be used to represent disjunctions, which when combined with conjunctions, can describe arbitrary sets. We note that both disjunctive sets and MCMC sampling are common tools in mixed-integer programming (Balas, 1979; Huang and Mehrotra, 2013). Another application is in data-driven optimization, where sampling from the complement of the feasible set of a partially described optimization problem can help train a machine learning model to predict aspects of the decision-making problem (Babier et al., 2018).

*tcychan@mie.utoronto.ca, adiamant@schulich.yorku.ca, rmahmood@mie.utoronto.ca

1.1 Motivating task

In practical decision-making problems, an optimization model is designed with structural assumptions and parameter estimates. The growing paradigm of data-driven optimization uses historical decisions to inform the construction of these parameters. For example in data-driven robust optimization, an uncertainty set around the parameters of a constraint is created by analyzing prior instances (Bertsimas et al., 2015). In another example, the hidden constraints of an optimization problem are learned by training a binary classifier using a data set of historical decisions (Babier et al., 2018).

The use of machine learning in most data-driven optimization settings is often limited by the fact that in most applications, the available data consists only of past *implemented* decisions. Decision-makers rarely collect data on unimplemented decisions, meaning that the available information belongs to a single category. Consequently, supervised learning techniques that learn by differentiating between instances of different classes are often untenable and methods are typically limited to statistical frameworks or unsupervised models (Bertsimas et al., 2015; Babier et al., 2018).

Our motivation is to solve an optimization problem that has hidden constraints. To this end, we wish to construct a barrier function for use in an interior point method when we do not know the true feasible set, but are instead given a relaxation and a data set of feasible decisions (Babier et al., 2018). Our approach is to construct a binary classifier that predicts whether a given decision is feasible or not. By sampling from a known subset of the infeasible region, i.e., the complement of the relaxation, we can augment our initial data set of feasible decisions with unimplemented decisions to train the classifier.

1.2 Contributions

The complement of a polyhedron is a non-convex set, making conventional sampling techniques inappropriate for the task. Our key methodological contribution is to propose an efficient MCMC algorithm for generating a sequence of points from the complement by extending current techniques that sample from the boundary of the set. We prove that this algorithm is guaranteed to cover the entire complement region and show that this is a sufficient condition to create a binary classifier that learns to distinguish between feasible and infeasible points for high-dimensional problems.

To demonstrate the effectiveness of our approach, we perform several numerical experiments on a variety of optimization problems. For each problem, a set of feasible decisions from an unknown feasible set are provided and we generate an artificial data set of infeasible decisions that lie in the complement of a known polyhedral relaxation using our MCMC algorithm. We then train a classifier to learn a separating boundary between the feasible and the infeasible data set. We compare our approach with several *unsupervised* density estimation baselines that are not augmented with data sampled from the complement. Using a simulated fractional knapsack problem, we show that our approach is essential for creating classifiers that (i) perform well when a tight separating boundary between feasible and infeasible regions is required; and (ii) when the data set of feasible decisions is small. Further, we consider linearized relaxations of all MIPLIB (miplib2017) instances with less than 80 variables and demonstrate that our sampling-based classifier significantly outperforms all baseline models. Code for our experiments are available at `https://github.com/rafidrm/mcmc-complement`.

2 Preliminaries

Consider a polyhedron $\mathcal{X} := \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{a}_m^\mathsf{T}\mathbf{x} \leq b_m, 1 \leq m \leq M\}$. There exist several algorithms for sampling from the interior $\text{int}(\mathcal{X})$, with the most well-known being the Hit-and-Run (HR) algorithm (Smith, 1984). Similarly, the Shake-and-Bake (SB) algorithm is the most well-known approach to sampling from the boundary $\text{bd}(\mathcal{X})$ (Boender et al., 1991). These algorithms fall under the family of MCMC techniques which operate by constructing a sequence of points governed by a proposal function. The sequence describes a Markov chain whose stationary distribution reflects the desired properties encoded by the proposal function (e.g., HR and SB converge to uniform distributions supported over $\text{int}(\mathcal{X})$ and $\text{bd}(\mathcal{X})$, respectively). Our MCMC approach for sampling from $\mathbb{R}^n \setminus \mathcal{X}$ is based on the SB algorithm.

SB operates on principles of stochastic billiards; intuitively, a ball bounces from each facet of the polyhedron to other facets with the points of contact being the generated points. The algorithm is as follows: assume an initial point $\mathbf{w}_0 \in \text{bd}(\mathcal{X})$ that lies on a single facet. That is, there is a unique m for which $\mathbf{a}_m^\mathsf{T}\mathbf{w}_0 = b_m$ and $\mathbf{a}_{m'}^\mathsf{T}\mathbf{w}_0 < b_{m'}$ for all $m' \neq m$. At every iteration of SB, given a boundary point $\mathbf{x}$ lying on the m-th facet, sample a *feasible direction* vector $\mathbf{r} \in \mathcal{R}_m := \{\mathbf{r} \mid \mathbf{a}_m^\mathsf{T}\mathbf{r} \leq 0, \|\mathbf{r}\| = 1\}$ according to some direction probability distribution $p_\mathbf{r}(\mathbf{r}|\mathbf{w})$. Then, calculate the nearest boundary point $\mathbf{w}' \in \text{bd}(\mathcal{X})$ from $\mathbf{w}$ in the direction of $\mathbf{r}$. This new point is selected as the next point in the Markov chain according to move probability $p_{\mathbf{w}'}(\mathbf{w}'|\mathbf{w})$. If $\mathbf{w}'$ is not selected, then the Markov chain does not update and the iteration repeats. Let N be the total number of iterations. In their seminal work on SB, Boender et al. (1991) proved that (i) the algorithm ensures (almost surely) every point on the Markov chain $\{\mathbf{w}_j\}_{j=1}^N$ lies on a unique facet of $\mathcal{X}$, and (ii) the Markov chain has a stationary uniform distribution over $\text{bd}(\mathcal{X})$.

There exist several variants of the SB algorithm that differ in their choices for the direction and move probabilities (Boender et al., 1991). The two most common variants are the Original SB and the Running SB. In the Original SB, the direction probability is uniform over the half-space defined by the facet $\mathcal{R}_m$. This leads to a move probability proportional to the angles of incidence. On the other hand, for the Running SB, the direction probability is chosen such that the algorithm moves in every iteration, i.e., $p_{\mathbf{w}'}(\mathbf{w}'|\mathbf{w}) = 1$ for all $\mathbf{w}', \mathbf{w} \in \text{bd}(\mathcal{X})$. In this work, we consider the Original SB algorithm due to its simplicity in calculating the direction probabilities.

3 Sampling from the complement of a polyhedron

Assume that $\mathcal{X}$ is full-dimensional and non-empty. Given a polyhedron $\mathcal{X}$, we generate a sequence of N points $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ such that $\mathcal{D} \subset \mathbb{R}^n \setminus \mathcal{X}$. In each iteration of SB, a direction vector is sampled and the next point on the boundary is found by moving in the given direction from the current point. Notice, however, that moving in the negative direction from the current point yields points that lie in the complement of the polyhedron, i.e., $\mathbf{x} \in \mathbb{R}^n \setminus \mathcal{X}$.

In our algorithm, we treat SB as a Hidden Markov chain. In each iteration, the previous boundary point is the hidden state and the observed state in the complement, $\mathbf{x}$, is generated according to the random direction vector that is sampled. Assume that the direction and move probabilities $p_\mathbf{r}(\mathbf{r}|\mathbf{w})$ and $p_\mathbf{w}(\mathbf{w}'|\mathbf{w})$ are such that $\{\mathbf{w}_i\}_{i=1}^N$ is a Markov chain of points on $\text{bd}(\mathcal{X})$. If we sample a random scale variable $\xi \sim p_\xi(\xi|\mathbf{r}, \mathbf{w})$ according to some positive distribution function, then $\mathbf{x} = \mathbf{w} - \xi\mathbf{r} \in \mathbb{R}^n \setminus \mathcal{X}$. Figure 1 shows a sample path

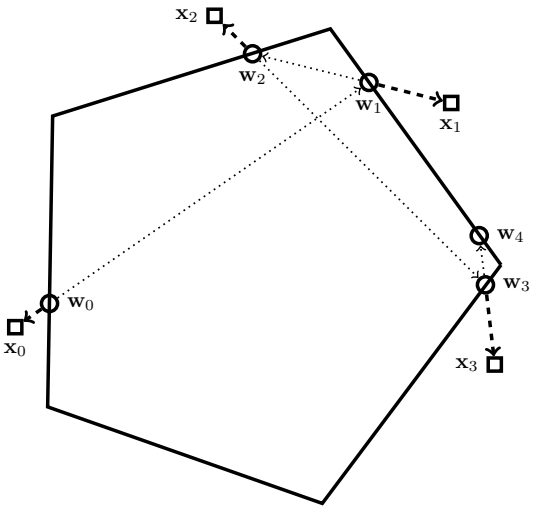


Figure 1: A sample sequence of points generated from the Complement SB algorithm. ◯ and □ are points on the boundary and infeasible points respectively.

of the chain $\{(\mathbf{w}_i, \mathbf{x}_i)\}_{i=1}^N$ which includes the hidden states. A detailed description of the MCMC algorithm is presented in Algorithm 1.

Algorithm 1 Complement Shake-and-Bake

Require: Polyhedron $\mathcal{X} = \{\mathbf{x} \mid \mathbf{a}_m^\mathsf{T}\mathbf{x} \leq b_m, m = 1, \ldots, M\}$; Sampling distributions $p_{\mathbf{w}}(\mathbf{w}'|\mathbf{w})$, $p_{\mathbf{r}}(\mathbf{r}|\mathbf{w})$, $p_\xi(\xi|\mathbf{r}, \mathbf{w})$, Number of points N; Initialization $\hat{\mathbf{w}}_i \in \mathrm{bd}(\mathcal{X}), i = 1$, $\mathcal{D} = \emptyset$
while $i < N$ **do**
  Randomly sample $\mathbf{r}_i \sim p_{\mathbf{r}}(\mathbf{r}|\mathbf{w}_i)$ and $\xi_i \sim p_\xi(\xi|\mathbf{r}, \mathbf{w})$.
  Update data set $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathbf{w}_i - \xi_i \mathbf{r}_i\}$.
  Let $\theta \in \min_m \left\{ \frac{b_m - \mathbf{a}_m^\mathsf{T}\mathbf{w}_i}{\mathbf{a}_m^\mathsf{T}\mathbf{r}_i} > 0 \right\}$.
  With probability $p_{\mathbf{w}}(\mathbf{w}_i + \theta\mathbf{r}|\mathbf{w}_i)$, update $\mathbf{w}_{i+1} \leftarrow \mathbf{w}_i + \theta\mathbf{r}_i$ and increase $i \leftarrow i + 1$, else $\mathbf{w}_{i+1} = \mathbf{w}_i$.
end while

Our algorithm enjoys all of the computational benefits of the original SB algorithm since we recycle the direction vectors $\mathbf{r}$ and only reverse the signs to ensure the generation of infeasible points. Generating the scale variable ξ is the only additional computation. We refer to Boender et al. (1991) for details on $p_{\mathbf{r}}(\mathbf{r}|\mathbf{w})$ and $p_{\mathbf{w}}(\mathbf{w}'|\mathbf{w})$. Any absolutely continuous distribution $p_\xi(\xi)$ supported over $(0, \infty)$ will suffice. Given an appropriate choice of $p_\xi(\xi|\mathbf{r}, \mathbf{w})$, we show that Algorithm 1 covers all of $\mathbb{R}^n \setminus \mathcal{X}$. That is, any measurable region of $\mathbb{R}^n \setminus \mathcal{X}$ has positive stationary probability.

Theorem 1. *Let μ_n denote the n-dimensional Lebesgue measure on a set. If $p_\xi(\xi|\mathbf{r}, \mathbf{w}) > 0$ for all $\xi \in (0, \infty)$ and $\mathbf{r}, \mathbf{w} \in \mathbb{R}^n$, then for any initial point $\mathbf{w}_0$ and any μ_n-measurable subset $\mathcal{A} \subset \mathbb{R}^n \setminus \mathcal{X}$,*

$$\lim_{N \to \infty} \mathbb{P}\{\mathbf{x}_N \in \mathcal{A} \mid \mathbf{w}_0\} > 0. \tag{1}$$

Proof. Without loss of generality, we assume $p_\xi(\xi|\mathbf{r}, \mathbf{w}) = p_\xi(\xi)$. Let $p_{SB}(\mathbf{w})$ denote the stationary distribution of the underlying SB algorithm. Let $\mathcal{A}' \subset \mathbb{R}^n \setminus \mathcal{X}$ denote an open, convex, and μ_n-measurable set for which there exists m such that $\mathbf{a}_m^\mathsf{T}\mathbf{x} > b_m$ for all $\mathbf{x} \in \mathcal{A}'$. We first prove (1) for all $\mathcal{A}'$ with this specific

structure and show that any $\mathcal{A} \subset \mathbb{R}^n \setminus \mathcal{X}$ contains a subset $\mathcal{A}' \subset \mathcal{A}$. Then, the probability for $\mathcal{A}'$ is a lower bound, i.e., $\mathbb{P}\{\mathbf{x}_N \in \mathcal{A} \mid \mathbf{w}_0 = \hat{\mathbf{w}}_0\} \geq \mathbb{P}\{\mathbf{x}_N \in \mathcal{A}' \mid \mathbf{w}_0 = \hat{\mathbf{w}}_0\}$, completing the proof.

Consider a set $\mathcal{A}'$ with the proposed structure. We will construct three measurable sets $\mathcal{W}$, $\mathcal{R}(\mathbf{w})$, and $\Xi(\mathbf{r}, \mathbf{w})$ such that

$$\{\mathbf{w} - \xi\mathbf{r} \mid \mathbf{w} \in \mathcal{W}, \mathbf{r} \in \mathcal{R}(\mathbf{w}), \xi \in \Xi(\mathbf{r}, \mathbf{w})\} \subseteq \mathcal{A}'.$$

Given their existence, we can bound

$$\begin{aligned}
&\lim_{N\to\infty} \mathbb{P}\{\mathbf{x}_N \in \mathcal{A}' \mid \mathbf{w}_0\} \\
&\quad \geq \lim_{N\to\infty} \int_{\mathcal{W}} \mathbb{P}\{\mathbf{w}_N - \xi\mathbf{r}_N \in \mathcal{A}' | \mathbf{w}_N\} p_{SB}(\mathbf{w}_N) d\mathbf{w}_N \\
&\quad \geq \lim_{N\to\infty} \int_{\mathcal{W}} \int_{\mathcal{R}(\mathbf{w}_N)} \int_{\Xi(\mathbf{r}_N, \mathbf{w}_N)} p_\xi(\xi)\, p_{\mathbf{r}}\left(\mathbf{r}_N | \mathbf{w}_N\right) p_{SB}(\mathbf{w}_N) d\xi d\mathbf{r}_N d\mathbf{w}_N.
\end{aligned}$$

We now construct the three sets. First, for some fixed $\epsilon > 0$, let

$$\mathcal{W} := \{\mathbf{w} \mid \mathbf{a}_m^\mathsf{T}\mathbf{w} = b_m, \mathbf{a}_{m'}^\mathsf{T}\mathbf{w} < b_{m'} - \epsilon, \forall m' \neq m\}.$$

Because $\mathcal{X}$ is closed and bounded with a non-empty interior, this set must exist for some $\epsilon > 0$. Furthermore from Boender et al. (1991, Lemma 2), there exists $\varepsilon > 0$ for which $\mathcal{W}$ has positive $(n-1)$-Lebesgue measure, i.e., $\mu_{n-1}(\mathcal{W}) > 0$, and thus, $p_{SB}(\mathcal{W}) > 0$. We avoid degenerate m by assuming that $\mathcal{X}$ has no redundant constraints.

For any $\mathbf{w} \in \mathcal{W}$, let

$$\mathcal{R}(\mathbf{w}) := \left\{ \frac{\mathbf{w} - \mathbf{x}}{\|\mathbf{w} - \mathbf{x}\|} \;\middle|\; \mathbf{x} \in \mathcal{A}' \right\}.$$

Because $\mu_n(\mathcal{A}') > 0$, we must have $\mu_n(\mathcal{R}(\mathbf{w})) > 0$ as well. Furthermore, for any $\mathbf{w} \in \mathcal{W}$, $\mathbf{a}_m^\mathsf{T}(\mathbf{w} - \mathbf{x}) \leq 0$ for all $\mathbf{x} \in \mathcal{A}'$. Therefore, $\mathcal{R}(\mathbf{w}) \subset \mathcal{R}_m$, meaning it is a set of valid directions.

Finally, we construct $\Xi(\mathbf{r}, \mathbf{w})$. By definition, for any $\mathbf{w} \in \mathcal{W}$ and $\mathbf{r} \in \mathcal{R}(\mathbf{w})$, there exist $\mathbf{x} \in \mathcal{A}'$ such that $\mathbf{w} - \|\mathbf{w} - \mathbf{x}\|\,\mathbf{r} \in \mathcal{A}'$. Then, let $\underline{\xi}(\mathbf{r}, \mathbf{w}) = \inf\{\xi \mid \mathbf{w} \in \xi\mathbf{r} \in \mathcal{A}'\}$ and $\overline{\xi}(\mathbf{r}, \mathbf{w}) = \sup\{\xi \mid \mathbf{w} \in \xi\mathbf{r} \in \mathcal{A}'\}$. Because $\mathcal{A}'$ is non-empty and open, we have $\underline{\xi} < \overline{\xi}$. Therefore, $\Xi(\mathbf{r}, \mathbf{w}) := (\underline{\xi}, \overline{\xi})$ has positive μ_1-measure. We can then integrate over all three sets to yield a positive lower bound on the probability of $\mathbf{x}_N \in \mathcal{A}'$.

We now extend the proof to an arbitrary set $\mathcal{A} \subset \mathbb{R}^n \setminus \mathcal{X}$ such that $\mu_n(\mathcal{A}) > 0$. Take any $\tilde{\mathbf{x}} \in \text{int}(\mathcal{A})$. Then, $\tilde{\mathbf{x}} \in \mathbb{R}^n \setminus \mathcal{X}$ implies that there must exist m such that $\mathbf{a}_m^\mathsf{T}\tilde{\mathbf{x}} > b_m$. We construct $\mathcal{A}'$ as an open ball centered on $\tilde{\mathbf{x}}$ that is sufficiently small such that $\mathbf{a}_m^\mathsf{T}\mathbf{x} > b_m$ for all $\mathbf{x} \in \mathcal{A}'$. Finally,

$$0 < \mathbb{P}\{\mathbf{x}_N \in \mathcal{A}' \mid \mathbf{w}_0\} \leq \mathbb{P}\{\mathbf{x}_N \in \mathcal{A} \mid \mathbf{w}_0\}.$$

We already proved the first inequality. The second comes from $\mathcal{A}' \subset \mathcal{A}$. □

Theorem 1 provides a useful property for learning to classify a hidden feasible set. Consider an optimization problem over a feasible set $\hat{\mathcal{X}}$ that is not known a priori. Instead, we have a data set of feasible

decisions $\hat{\mathcal{D}} \sim \hat{\mathbb{P}}$ drawn i.i.d. from a distribution supported over $\hat{\mathcal{X}}$. We also have a relaxation of the feasible set $\mathcal{X}$. Using Algorithm 1, we can generate a data set of infeasible decisions $\mathcal{D} \subset \mathbb{R}^n \setminus \mathcal{X}$ with steady state distribution $\mathbb{P}$. With this augmented data set of feasible and infeasible points, we then train a binary classifier $D(\mathbf{x}) : \mathbb{R}^n \to \{0, 1\}$ such that $D(\mathbf{x}) = 1$ for all $\mathbf{x} \in \hat{\mathcal{X}}$ and $D(\mathbf{x}) = 0$ for $\mathbf{x} \in \mathbb{R}^n \setminus \hat{\mathcal{X}}$. Without loss of generality, suppose we minimize the Binary Cross Entropy loss (Goodfellow et al., 2016):

$$\min_{D} \; -\mathbb{E}_{\hat{\mathbf{x}} \sim \hat{\mathbb{P}}}\Big[\log D(\hat{\mathbf{x}})\Big] - \mathbb{E}_{\mathbf{x} \sim \mathbb{P}}\Big[\log\big(1 - D(\mathbf{x})\big)\Big]. \tag{2}$$

The results below shows that data generated from Algorithm 1 is sufficient to train this SB-based classifier to accurately differentiate between $\hat{\mathcal{X}}$ and $\mathcal{X}$.

Lemma 1 (Arjovsky and Bottou (2017))**.** *Consider two distributions $\mathbb{P}$ and $\hat{\mathbb{P}}$ supported over closed and disjoint sets. Then, the optimal $D^*(\mathbf{x})$ of (2) satisfies $D^*(\mathbf{x}) = 1$ for all $\mathbf{x} \in \text{supp}(\mathbb{P})$ and $D^*(\mathbf{x}) = 0$ for all $\mathbf{x} \in \text{supp}(\hat{\mathbb{P}})$.*

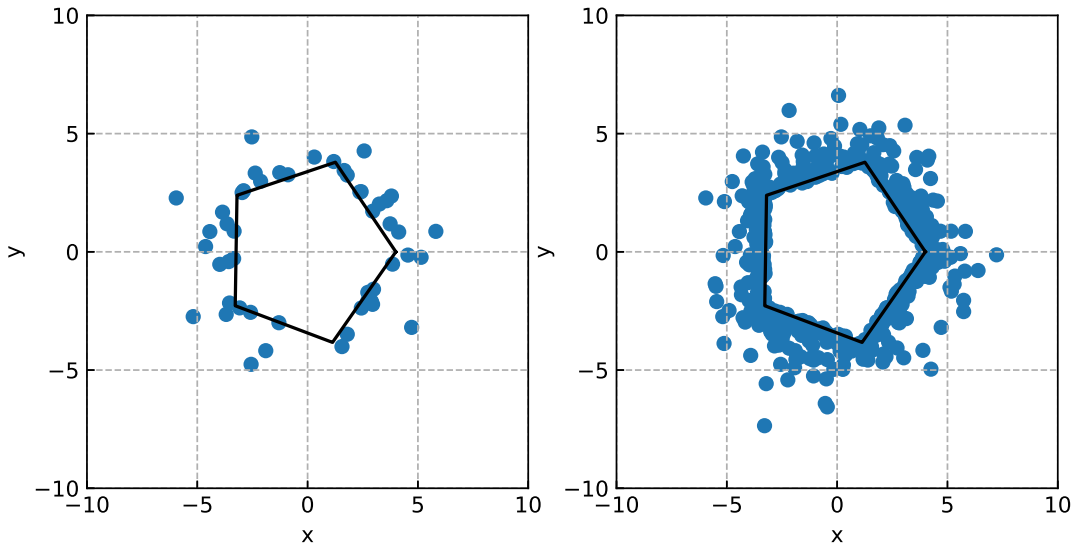
Lemma 1 states that for a binary classifier to provably predict points with perfect accuracy, the data distributions of those points must have closed and disjoint supports. We use this result to demonstrate that it is possible to construct a classifier that approximates the feasible set.

Proposition 1. *Consider $\hat{\mathbb{P}}$ supported over a closed polyhedron $\hat{\mathcal{X}}$. For $\epsilon > 0$, let $\mathcal{X}_\epsilon := \{\hat{\mathbf{x}} + \mathbf{y} \mid \hat{\mathbf{x}} \in \hat{\mathcal{X}}, \|\mathbf{y}\| < \epsilon\}$. Let $\mathbb{P}$ be the steady state distribution of the Markov chain generated by Algorithm 1. Then the optimal classifier $D^*(\mathbf{x})$ satisfies $D^*(\mathbf{x}) = 1$ for $\mathbf{x} \in \hat{\mathcal{X}}$ and $D^*(\mathbf{x}) = 0$ for $\mathbf{x} \in \mathbb{R}^n \setminus \mathcal{X}_\epsilon$.*

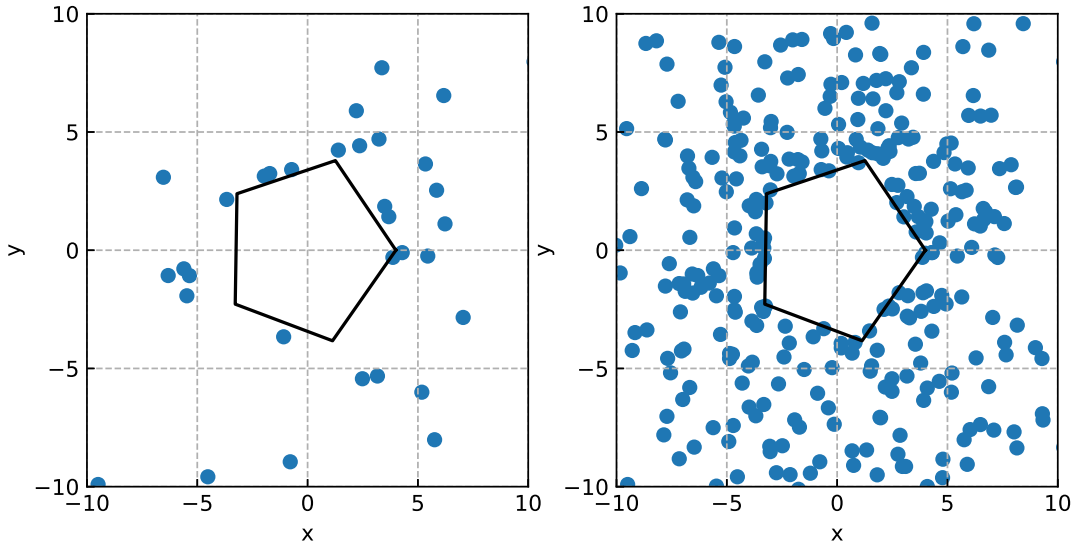
Proof. From Theorem 1, the steady state distribution satisfies $\mathbb{P}\{\mathcal{A}\} > 0$ for any measurable $\mathcal{A} \in \mathbb{R}^n \setminus \mathcal{X}_\epsilon$. Thus, the steady state distribution is supported over the entire $\mathbb{R}^n \setminus \mathcal{X}_\epsilon$. Note that $\hat{\mathcal{X}}$ and $\mathbb{R}^n \setminus \mathcal{X}_\epsilon$ are disjoint. From Lemma 1, the optimal classifier perfectly separates the two supports. □

Proposition 1 states that given sufficient data, a classifier can learn to perfectly distinguish from an unknown set $\hat{\mathcal{X}}$ and its polyhedral relaxation $\mathcal{X}$. We remark that training via the Binary Cross Entropy loss function is not a necessary condition; nearly any loss function will suffice. However, we seek to distinguish between $\hat{\mathcal{X}}$ and $\mathbb{R}^n \setminus \hat{\mathcal{X}}$. Therefore, the proposition is useful insofar as the relaxation $\mathcal{X}$ is relatively tight. In our numerical experiments, we demonstrate that the classifier does indeed accurately learn to identify points in $\hat{\mathcal{X}}$ as feasible and points in the unknown band $\mathcal{X} \setminus \hat{\mathcal{X}}$ as infeasible.

When implementing Algorithm 1, the choice of distribution $p_\xi(\xi)$ will depend on the application. In this work, we assume an Exponential distribution $p_\xi(\xi) = \text{Exp}(\lambda) = \lambda e^{-\lambda\xi}$. In practice when training a classifier $D(\mathbf{x})$ to learn a hidden feasible set, we do not have access to $\mathbb{P}$ and $\hat{\mathbb{P}}$ but rather data sets $\mathcal{D}$ and $\hat{\mathcal{D}}$. Given a limited data set, it is important for the classifier to accurately learn the regions near the boundary $\text{bd}(\mathcal{X})$ because the band near the boundary $\mathcal{X} \setminus \hat{\mathcal{X}}$ is the most challenging region to classify. Note that it is still important to generate some points far from the boundary in order to satisfy Proposition 1. Using an exponential distribution ensures that we generate points with high density near the boundary and low density further away. Figure 2 shows several stages of Algorithm 1 for different values of λ.



(a) $\lambda = 1$



(b) $\lambda = 10$

Figure 2: Sample of points generated using an Exponential distribution $p_\xi(\xi|\mathbf{r}, \mathbf{w}) = \text{Exp}(\lambda)$. The left and right plots show $N = 50$ and $N = 500$ samples, respectively.

4 Numerical analysis

We implement Algorithm 1 and the corresponding SB-based classifier to learn the hidden feasible set $\hat{\mathcal{X}}$. Given a data set of feasible decisions $\hat{\mathcal{D}} = \{\hat{\mathbf{x}}_i\}_{i=1}^N \subseteq \hat{\mathcal{X}}$ and a relaxation $\mathcal{X}$, we augment or data set by sampling infeasible decisions before training an SB-based classifier to predict whether a decision is feasible or not with respect to $\hat{\mathcal{X}}$.

Classical approaches towards constructing a classifier $D(\mathbf{x})$ would not have a set of infeasible points $\mathcal{D}$ and would thus be forced to use some form of unsupervised or generative modeling. We implement two baseline models: a Gaussian Mixture Model (GMM) and Kernel Density Estimation (KDE). Both are generative modeling techniques that use $\hat{\mathcal{D}}$ to estimate a probability distribution over $\hat{\mathcal{X}}$.

In our first set of experiments, we consider a simulated fractional knapsack problem. We use this example to investigate the relative tightness of the relaxation and show that when the relaxation is a reasonable approximation of the hidden feasible set, our approach dominates the baseline models. We then investigate the effect that the size of the data set (i.e., $|\hat{\mathcal{D}}|$) has on the ability to learn the feasible set and show that by sampling from the infeasible region, our classifier achieves competitive performance with the unsupervised baseline models while requiring an order-of-magnitude less feasible data. Finally, we show that as the dimension of the problem increases, our approach still learns the hidden feasible set while the baseline models collapse.

Finally, we conduct experiments on linearizations over a set of MIPLIB problems that have less than 80 variables (miplib2017). Our SB-based classifier dominates the baseline models in terms of accuracy and F_1 score on nearly all instances, often by margins of 20%. Furthermore, we show that for challenging instances with a large number of variables, the baseline models once again completely collapse and either indiscriminately predict all test points as infeasible or all points as feasible. In contrast, the SB-based classifier still demonstrates learning even for these challenging problems.

4.1 Data and methods

Consider a hidden polyhedron $\hat{\mathcal{X}} = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}\}$. We first construct a relaxation $\mathcal{X} = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b} - \mathbf{d}\}$, where $d_m \sim p_d(d)$ is a random perturbation variable. In order to ensure $\hat{\mathcal{X}} \subset \mathcal{X}$ and that $\mathcal{X}$ is a relatively close approximation to $\hat{\mathcal{X}}$, we use an Exponential distribution $p_d(d) = \text{Exp}(\gamma)$ with a scale parameter γ proportional to the polyhedral constraints, i.e.,

$$\gamma = \gamma_0 \max\{\|\mathbf{b}\|_\infty, \|\mathbf{a}_1\|_\infty, \|\mathbf{a}_2\|_\infty, \ldots, \|\mathbf{a}_M\|_\infty\}, \tag{3}$$

for a constant $\gamma_0 > 0$. This ensures that $\mathcal{X}$ is neither too tight nor too loose of a relaxation. We refer to γ as the degree of the relaxation.

For each instance, we use a HR sampler to generate feasible points $\hat{\mathcal{D}} = \{\hat{\mathbf{x}}_i\}_{i=1}^N \subset \hat{\mathcal{X}}$. Thus, $\mathcal{X}$ and $\hat{\mathcal{D}}$ constitute the available information used to learn $\hat{\mathcal{X}}$. Using Algorithm 1, we generate an "infeasible" data set $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N \subset \mathbb{R}^n \setminus \mathcal{X}$ and then train an off-the-shelf Gradient Boosted Tree (GBT) to classify between $\mathcal{D}$ and $\hat{\mathcal{D}}$. We do not tune hyper-parameters for our classifier finding that it outperforms the baseline models in most cases.

We consider two generative baseline models that estimate a probability distribution $\hat{p}(\mathbf{x})$ over $\hat{\mathcal{D}}$. That is, we define a threshold parameter $t = \min_{\hat{\mathbf{x}} \in \hat{\mathcal{D}}} \hat{p}(\hat{\mathbf{x}})$ as the smallest probability such that the data set consists entirely of feasible decisions. Then, the baseline classifier applies a threshold rule over the generative model, i.e., $D(\mathbf{x}) = \mathbf{1}[\mathbf{x} \in \mathcal{X}]\mathbf{1}[\hat{p}(\mathbf{x}) \geq t]$. The first term in the classifier simply checks if the decision lies within the given relaxation $\mathcal{X}$. Thus, it is an intuitive approach to use the knowledge of the polyhedral relaxation. We implement two baseline models: a KDE and a GMM and cross-validate over their respective hyper-parameters using $\hat{\mathcal{D}}$. As these models typically do not scale efficiently to higher dimensions (Theis et al.,

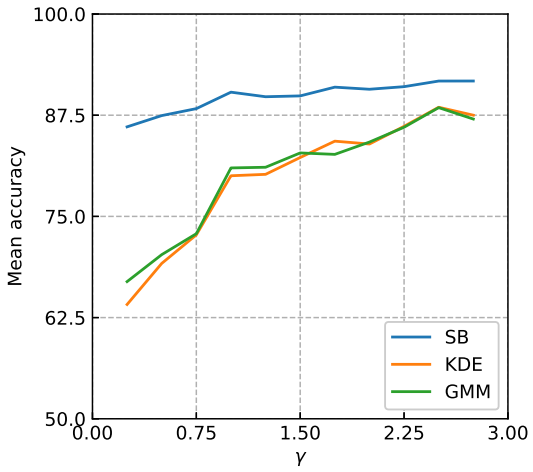


Figure 3: Mean accuracy of the models from increasing the degree of the relaxation γ.

2016), we consider the use of Principal Component Analysis (PCA) to pre-process the training data and reduce the dimensionality of the problem. As a result, we implement all models with and without PCA for ablation.

In order to evaluate our approach, we generate an out-of-sample test set of feasible points $\hat{\mathcal{D}}' \subset \hat{\mathcal{X}}$ and infeasible decisions $\mathcal{D}' \subset \mathcal{X} \setminus \hat{\mathcal{X}}$. Both data sets are generated with an HR sampler. When generating $\mathcal{D}'$, we simply reject points in $\hat{\mathcal{X}}$ for the Markov chain. Note that we do not sample points in $\mathbb{R}^n \setminus \mathcal{X}$ for testing as they would be trivially identified as infeasible given our knowledge of $\mathcal{X}$. Our final out-of-sample test set is $\hat{\mathcal{D}}' \cup \mathcal{D}'$.

4.2 A fractional knapsack problem

Consider a fractional knapsack problem with the following feasible set $\hat{\mathcal{X}}$ and polyhedral relaxation $\mathcal{X}$:

$$\hat{\mathcal{X}} = \left\{ \mathbf{x} \;\middle|\; \sum_{i=1}^{n} x_i \leq 5, x_i \geq 0 \right\}$$
$$\mathcal{X} = \left\{ \mathbf{x} \;\middle|\; \sum_{i=1}^{n} x_i \leq 5 + d_0, x_i \geq -d_i \right\}.$$

We compare the SB-based supervised learning approach with the two generative baselines in three different scenarios. We first analyze the degree of relaxation γ to assess the algorithms' ability to learn under different relaxations. We then investigate how the size of the data set (N) affects the performance of the different algorithms. Finally, we assess the ability of the SB-based classifier to learn in n-dimensional spaces. Each experiment varies a single parameter while holding the others constant; we set $\gamma_0 = 0.1$ (i.e., $\gamma = 0.5$), $N = 200$, $n = 2$, as the default settings. We fix $\lambda = 0.5$ when generating $\mathcal{D}$ using Algorithm 1. All results are averaged from 50 trials.

Unsupervised learning techniques that rely only on $\hat{\mathcal{D}}$ to learn an approximation of the feasible set can mis-classify regions where there is no information. For example, regions outside $\hat{\mathcal{X}}$ but close to $\hat{\mathcal{D}}$ may be mis-

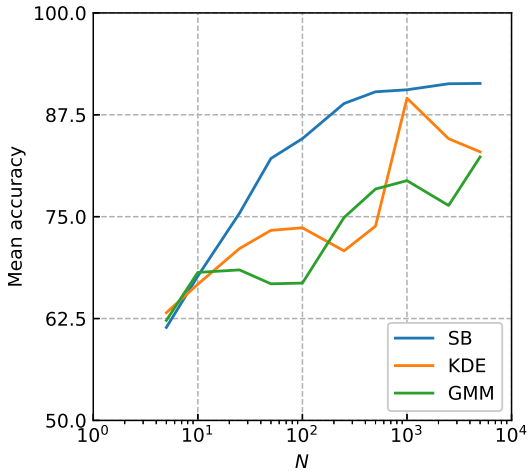
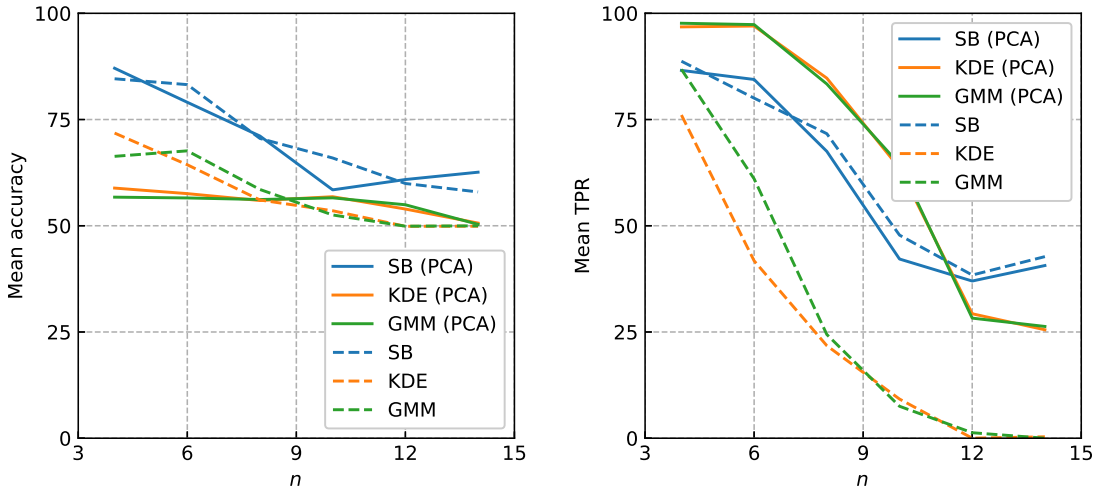


Figure 4: Mean accuracy of the models as we increase the training set size N.

classified as feasible since a KDE and GMM would show a gradual drop in density from the data. Generating points in $\mathbb{R}^n \setminus \mathcal{X}$ offers a counter-balance to unsupervised techniques that incorrectly mis-classify infeasible regions as feasible. This effect is particularly prominent when the relaxation is tighter as demonstrated in Figure 3 which plots the out-of-sample accuracy as a function of γ. The SB-based classifier yields out-of-sample accuracy of approximately 91% regardless of the value of γ. The unsupervised baselines, in contrast, show poor out-of-sample accuracy when γ is small and slowly increase in performance as γ increases. Even at the largest value, $\gamma = 2.75$, the SB-based classifier still outperforms the baseline models. Thus, unsupervised learning only becomes competitive once the given relaxed bound is at least 50% larger than the true hidden bound.

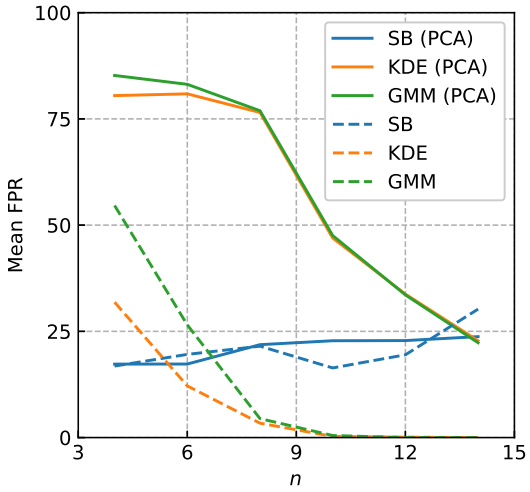
The unsupervised learning baselines require significantly more data than our SB-based classifier. Figure 4 plots out-of-sample accuracy as a function of N. When $N = 5$, all of the methods are equally poor and achieve approximately 63% out-of-sample accuracy. However, by using generated infeasible data, the SB-based classifier converges to 93% out-of-sample accuracy with $N = 100$. Note that the baseline models are non-monotone due to the grid-search algorithm used to find the best hyper-parameters of KDE and GMM, respectively. The optimal selection of these hyper-parameters change as we increase the amount of data and thus, the baseline models require extensive tuning. Nonetheless, even if we take the envelope of the KDE and GMM curves, we can still conclude that our sampling approach is on average an order-of-magnitude more data-efficient than the unsupervised baselines at learning the feasible set.

As previously shown, the unsupervised learning baselines require large data sets. As a result, in higher dimensions, these models assign small probabilities to regions where there may not be sufficient data. Thus, the differences in probabilities between regions where there are a small number of points and where there are no points can become negligible and it may appear as if these unsupervised learning models are applying nearly uniform (small) density over large areas. To address this issue, we pre-process the data that is used to train the baselines using PCA in order to reduce the dimension of the problem. While for $n \leq 8$, the baselines have better accuracy without PCA, reducing the dimensionality proves effective for $n \geq 9$. When n is low, using PCA leads to a loss of information for the unsupervised baselines.

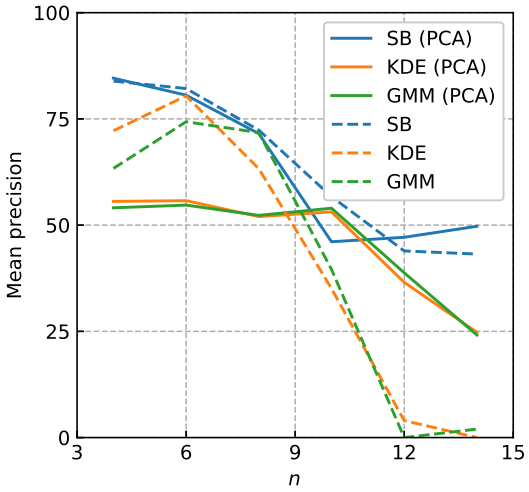


(a) Mean accuracy vs n

(b) Mean TPR (recall) vs n



(c) Mean FPR vs n (lower is better)



(d) Mean precision vs n

Figure 5: Evaluating accuracy, TPR (recall), FPR, and precision of the different models as we increase the number of variables in the knapsack n. All models are are pre-processed using PCA to reduce the dimension by 25%.

More specifically, we consider increasing number of variables in the knapsack n, while holding $N = 200$. Because KDE and GMMs are known to perform poorly in high-dimensions, we use PCA in conjunction with the baseline models and our SB-based classifier to reduce the dimension by 25%. Figure 5 plots the accuracy, True Positive Rate (TPR) or recall, False Positive Rate (FPR), and precision over increasing n. Overall, the SB-based classifier (without PCA) strictly dominates all baselines on accuracy and precision. Furthermore, our classifier maintains a relatively flat FPR (around 25%) that scales slowly with the number of variables. All of the baselines converge to exactly 50% accuracy on the out-of-sample data demonstrating that they are not capable of learning the feasible set with the given amount of data. Note that for the baselines,

Table 1: Out-of-sample accuracy over instances of MIPLIB problems. We implement all models with and without PCA (reducing dimension by 50%). The best performing models per MIPLIB instance are highlighted.

Instance	Without PCA			With PCA		
	SB	KDE	GMM	SB	KDE	GMM
ej	90.6	97.3	94.2	85.5	84.5	82.3
gen-ip002	90.0	58.3	58.9	64.1	61.1	61.7
gen-ip016	53.0	50.0	50.0	47.4	61.0	61.0
gen-ip021	93.6	54.9	59.0	78.2	54.1	52.3
gen-ip036	95.2	56.7	63.6	85.1	61.3	60.3
gen-ip054	89.0	60.6	65.5	67.4	53.6	51.1
gr4x6	79.9	53.0	55.2	67.1	61.3	56.8
markshare_4_0	94.9	59.8	55.0	76.9	61.1	54.5
markshare_5_0	86.7	61.0	64.1	65.1	63.2	61.3
neos5	85.6	50.0	50.0	84.1	51.2	51.7

Table 2: Out of sample TPR, precision, and F_1-score over instances of MIPLIB problems. We draw the best-performing version of each model with respect to PCA. The best performing models in terms of F_1-score are highlighted.

Instance	TPR			Precision			F_1-score		
	SB	KDE	GMM	SB	KDE	GMM	SB	KDE	GMM
ej	99.9	99.6	99.9	84.7	95.2	90.0	91.7	97.4	94.7
gen-ip002	93.5	98.0	96.5	90.5	57.0	57.5	92.0	72.1	72.1
gen-ip016	12.2	27.1	27.1	23.7	35.7	35.6	16.1	30.8	30.8
gen-ip021	98.2	9.74	18.0	90.7	70.0	99.9	94.3	17.1	42.6
gen-ip036	99.8	99.6	34.2	91.9	56.6	86.3	95.7	72.2	49.0
gen-ip054	87.4	21.4	32.0	90.3	99.3	97.8	88.8	35.2	48.2
gr4x6	88.6	99.8	94.4	79.6	57.4	55.7	83.9	72.9	70.1
markshare_4_0	97.7	99.3	10.2	92.9	56.7	100	95.2	72.2	18.5
markshare_5_0	92.4	99.8	28.3	85.7	58.2	90.0	88.9	73.5	43.1
neos5	94.9	96.9	100	81.8	51.6	51.0	87.9	67.3	67.5

TPR, FPR, and precision all decrease as n increases. When TPR and FPR are both 0, as in the case of the baselines with no PCA for $n \geq 12$, there are zero true and false positives and the models predict all points as infeasible.

4.3 Learning hidden feasible sets on MIPLIB instances

We next consider learning the feasible set of realistic benchmark problems, by drawing all instances of optimization problems with less than 80 variables from the MIPLIB database (miplib2017). We ignore problems marked "infeasible," those with more than 5000 constraints (e.g., `supportcase21i`), and those with large optimal values (e.g., `flugpl`), noting that these instances typically have pathological feasible sets.

For each instance, we use the LP relaxation of the feasible set and convert it to inequality form $\{\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b}\}$. However, these problems may yet have pathological low-dimensional shapes. Consequently, we relax the right-hand-side terms by γ to obtain $\hat{\mathcal{X}} = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b} - \gamma\mathbf{1}\}$. We set γ as in (3), with $\gamma_0 = 1$. We then construct hidden feasible sets $\mathcal{X} = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} \geq \mathbf{b} - \gamma\mathbf{1} - \mathbf{d}\}$ where $d_m \sim p_d(d) = \text{Exp}(\gamma)$ with the same γ

as before. In each experiment, we generate training sets of $N = 4000$ feasible points $\hat{\mathcal{D}} \subset \hat{\mathcal{X}}$ and infeasible points $\mathcal{D} \subset \mathbb{R}^n \setminus \mathcal{X}$. When generating $\mathcal{D}$, we use Algorithm 1 and fix $p_\xi(\xi) = \text{Exp}(1)$ for all instances. All results are averaged over 40 trials.

Table 1 shows the accuracy of each model on out-of-sample test sets with and without the use of PCA (reducing the dimension by 50%). The best performing model for each MIPLIB instance is highlighted. The SB-based classifier outperforms all other models in nearly every instance and is often over 10% better than the best baseline model. Furthermore, the SB-based classifier performs better without the use of PCA. This is because $\hat{\mathcal{X}}$ possesses structure (i.e., linear constraints) that is useful for learning. Reducing the dimensionality of the problem through the application of PCA may result in a loss of important information for training. However, PCA is useful for the baselines and improves the performance of the KDE baseline on 10 instances. The GMM baseline sees a similar improvement for 7 instances.

For many instances, the baseline models achieve an accuracy level that is near 50%. Thus, we explore secondary metrics in order to understand the nature of these errors. Table 2 shows the out-of-sample TPR, precision, and F_1-score of the best performing SB-based classifier, KDE, and GMM, respectively. For the majority of the instances, the KDE baseline observes a TPR greater than 95% and precision less than 60% suggesting that the number of false negatives is relatively small but the number of false positives is approximately equal to the number of true positives. That is, the KDE baseline predicts nearly every test point to be feasible. We observe the opposite behavior for the GMM baseline in that TPR is small but precision is greater than 90%. That is, the GMM baseline predicts nearly every test point to be infeasible. Note that the SB-based classifier does not display these biases as can be observed by the F_1-score (which combines TPR and precision). Here, our classifier consistently dominates both of the baseline models.

5 Conclusion

We propose a MCMC method for sampling points in the complement of a polyhedron. We prove that our algorithm will eventually sample all points in the complement and demonstrate an application of our approach in a machine learning problem, i.e., augmenting data when learning a hidden feasible set using data from past implemented decisions. In a series of numerical experiments, we show that our method is more data-efficient and effectively scales to high dimensions as compared to the baseline models. We also show that it is more adept at learning to classify feasibility when the separating boundary is tight, as is a requirement in many optimization problems.

References

Andrieu, C., De Freitas, N., Doucet, A., Jordan, M.I., 2003. An introduction to mcmc for machine learning. Machine Learning 50, 5–43.

Arjovsky, M., Bottou, L., 2017. Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862 .

Babier, A., Chan, T.C.Y., Diamant, A., Mahmood, R., 2018. Learning to optimize with hidden constraints. arXiv preprint arXiv:1805.09293 .

Balas, E., 1979. Disjunctive programming, in: Annals of discrete mathematics. Elsevier. volume 5, pp. 3–51.

Bertsimas, D., Gupta, V., Paschalidis, I.C., 2015. Data-driven estimation in equilibrium using inverse optimization. Mathematical Programming 153, 595–633.

Bertsimas, D., Vempala, S., 2004. Solving convex programs by random walks. Journal of the ACM (JACM) 51, 540–556.

Boender, C.G.E., Caron, R.J., McDonald, J.F., Kan, A.H.G.R., Romeijn, H.E., Smith, R.L., Telgen, J., Vorst, A.C.F., 1991. Shake-and-bake algorithms for generating uniform points on the boundary of bounded polyhedra. Operations research 39, 945–954.

Brooks, S., Gelman, A., Jones, G., Meng, X.L., 2011. Handbook of markov chain monte carlo. CRC press.

Dieker, A.B., Vempala, S.S., 2015. Stochastic billiards for sampling from the boundary of a convex set. Mathematics of Operations Research 40, 888–901.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. volume 1. MIT press Cambridge.

Huang, K.L., Mehrotra, S., 2013. An empirical evaluation of walk-and-round heuristics for mixed integer linear programs. Computational optimization and applications 55, 545–570.

miplib2017, 2018. MIPLIB 2017. Http://miplib.zib.de.

Ripley, B.D., 2009. Stochastic simulation. volume 316. John Wiley & Sons.

Smith, R.L., 1984. Efficient monte carlo procedures for generating points uniformly distributed over bounded regions. Operations Research 32, 1296–1308.

Theis, L., van den Oord, A., Bethge, M., 2016. A note on the evaluation of generative models, in: International Conference on Learning Representations (ICLR 2016), pp. 1–10.