Optimal counterfactual explanations for k-Nearest Neighbors using Mathematical Optimization and Constraint Programming

Claudio Contardo[1], Ricardo Fukasawa[2*], Louis-Martin Rousseau[3], and Thibaut Vidal[3]

[1] Department of Mechanical, Industrial and Aerospace Engineering, Concordia University
[2] Department of Combinatorics and Optimization, University of Waterloo
[3] Mathematics and Industrial Engineering, Polytechnique Montreal
* Corresponding author

Abstract. Within the topic of explainable AI, counterfactual explanations to classifiers have received significant recent attention. We study counterfactual explanations that try to explain why a data point received an undesirable classification by providing the closest data point that would have received a desirable one. Within the context of one the simplest and most popular classification models —k-nearest neighbors (k-NN)— the solution to such optimal counterfactual explanation is still very challenging computationally. In this work, we present techniques that significantly improve the computational times to find such optimal counterfactual explanations for k-NN.

1 Introduction

k-Nearest Neighbors (k-NN) stands as one of the most popular and simplest machine learning (ML) classification models. In k-NN, we are given a set of n *data points* or *observations* (given as points in $\mathbb{R}^d$). Each such observation has an associated label, taken from a set $\mathcal{L}$. Finally, we are also given an integer $k \geq 1$. A new unseen data point $x \in \mathbb{R}^d$ is classified by looking at which label appears more frequently among the labels of its k closest observations.

In this paper, we study the problem of providing counterfactual explanations for k-NN. Counterfactual explanations in ML play a key role in the interpretability of the models. They provide answers to the following fundamental question: “What is the smallest change that should be applied to a sample point x to shift its label from an undesirable classification to a desirable one?”. Counterfactual explanations have been proposed for a variety of ML models, such as tree ensembles [7] and linear classifiers [11], among others. Recent surveys on counterfactual explanations can be found in [2,5].

Despite its algorithmic simplicity, the only counterfactual explanatory model for k-NN has been proposed in [4] via a mixed-integer program (MIP), but as mentioned in that paper, “Explanations of nearest-neighbor predictors can be obtained in short times for small sample sizes, but do not scale as well to large sample sizes". We are interested in applying techniques from mathematical optimization and constraint programming to develop more efficient counterfactual models and algorithms for k-NN. The contributions of our article can therefore be summarized as follows:

1. We introduce a filtering mechanism aimed at reducing the dimension of these models without compromising their correctness.
2. We introduce two relaxation bounds and an incremental solution approach, which exploits them, to accelerate the resolution of the models.
3. We assess the relative performances of the proposed models and methods.

The remainder of this article is organized as follows. Section 2 presents the basic problem definition and formulation. Section 3 focuses on the first dimensionality reduction technique called *filtering*. Section 4 presents the other dimensionality reduction techniques (*partition* and *sampling*). Computational results are presented in Section 5 and final discussions in Section 6.

2 Problem definition

We start by formally defining the k-nearest neighbors (k-NN) problem. In k-NN, we are given a set $\{(x^i, y(x^i)) : i = 1, \dots, n\}$ of labeled data, where $\mathcal{O} = \{x^i : i = 1, \dots, n\} \subseteq \mathbb{R}^d$ is the set of *data points* or *observations*, and $y(x^i) \in \mathcal{L}$, where $\mathcal{L} = \{0, 1\}$ is the set of possible labels. We assume, WLOG, that 0 is an undesirable label and 1 is a desirable one. We are also given a dissimilarity measure $dist(x, w) \geq 0$ that establishes the dissimilarity between two points $x, w \in \mathbb{R}^d$. Throughout this work, we assume $dist(x, y)$ to be $||x - y||_1$. Two observations with a dissimilarity close to zero are therefore interpreted as being similar. Finally, we are also given an integer $k \geq 1$.

The way the k-NN classifier works for a new, unseen data point $x \notin \mathcal{O}$ is as follows. Let $N(k, x)$ be the set of k closest points in $\mathcal{O}$ to x according to the dissimilarity measure $dist$, i.e., $|N(k, x)| = k$ and $dist(u, x) \leq dist(v, x), \forall u \in N(k, x), v \in \mathcal{O} \backslash N(k, x)$. Let $N_t(k, x) = \{\xi \in N(k, x) : y(\xi) = t\}$ for $t \in \mathcal{L}$. The k-NN classifier returns the majority class in $N(k, x)$, i.e., $c(x) = \arg\max\{|N_t(k, x)| : t \in \mathcal{L}\}$. We will assume that ties are not possible, for instance by assuming that k is odd. The k-NN may be ill-defined if two or more points have the same dissimilarity, in which case there may be multiple sets $N(k, x)$. We will assume an *optimistic* definition: In case of ties, we will assume that $N(k, x)$ corresponds to the subset with the minimum number of observations labeled as 0.

We now proceed to define the counterfactual k-NN problem (cnt-k-NN). In addition to the input of k-NN, the cnt-k-NN problem receives as input a point x^0 such that $c(x^0) = 0$. In the cnt-k-NN we wish to find the point $x \in \mathbb{R}^d$ such that $c(x) = 1$ and minimizing $dist(x, x^0)$. We define some notation (that will be used later) to make the dependence on $x^1, \dots, x^n$ explicit. Let $N_v := \{i \in \{1, \dots, n\} : y(x^i) = v\}$ for $v = 0, 1$. The following formulation is given in [4]:

$$\text{cnt}(N_0, N_1) = \begin{cases} \min & \delta_0 & \text{(1a)} \\ \text{s.t.} & \delta_0 = dist(x, x^0) & \text{(1b)} \\ & \delta_i = dist(x, x^i), \forall i \in N_0 \cup N_1 & \text{(1c)} \\ & \lambda_i = 1 \Rightarrow \delta_i \leq \varDelta, \forall i \in N_0 \cup N_1 & \text{(1d)} \\ & \lambda_i = 0 \Rightarrow \delta_i \geq \varDelta, \forall i \in N_0 \cup N_1 & \text{(1e)} \\ & \sum_{i \in N_0 \cup N_1} \lambda_i = k & \text{(1f)} \\ & \sum_{i \in N_0} \lambda_i \leq \lfloor k/2 \rfloor & \text{(1g)} \\ & \lambda \in \{0,1\}^{N_0 \cup N_1}; x \in \mathbb{R}^d; \delta \in \mathbb{R}_+^{\{0\} \cup N_0 \cup N_1}; \varDelta \in \mathbb{R}_+ & \text{(1h)} \end{cases}$$

We denote by $X(N_0, N_1)$ the set of x for which there exist $\delta, \varDelta, \lambda$ for which $(x, \delta, \varDelta, \lambda)$ is feasible for (1).

Note that while formulation (1) is not a MIP directly, it can be input into a MIP solver by replacing constraint (1d) with $\delta_i \leq \varDelta + M(1 - \lambda_i)$ and constraint (1e) with $\delta_i \geq \varDelta - M\lambda_i$. Moreover, since Gurobi [6] accepts constraints like $y = |x|$ and reformulates them automatically in a MIP, the norm constraints can also be passed to it.

Note that, in [4], constraint (1e) was written as $\delta_i \geq \varDelta - M\lambda_i + \epsilon$ for some $\epsilon > 0$. However, such formulation will lead to an infeasible model for instance when $n > k$ and $x^1 = \ldots = x^n$, with $y(x^1) = \ldots = y(x^n) = 1$, even though any x is feasible in that case. We also note that x is feasible for (1) if and only if $c(x) = 1$ according to an optimistic k-NN classifier.

In addition, (1) can be solved using a constraint programming (CP) solver directly. Note, however, that CP solvers assume that all variables are required to be integer, so we scale the variables x to assume that it only assumes integer variables. Such scaling and integrality may cause some possible solutions to (1) to be lost.

The main challenge in solving problem (1) is that n may be very large, becoming too big to solve using either MIP or CP. In addition, for MIP, constraints (1d) and (1e) are "Big-M" type constraints, which are not a very good structure to have in MIP.

In what follows we present techniques that were applied to try and reduce the size of the problem.

3 Filtering

The idea for reducing the size of (1) relies on a basic observation that, if one of the points x^i is "too far" from x^0, then it will not be one of the k nearest neighbors of a point that is "close" to x^0.

Formally, we assume that we have already found a feasible solution $\bar{x} \in X(N_0, N_1)$. Any solution we are interested must be closer to x^0 than $\bar{x}$ and so, the following lemma allows us to eliminate points x^i from (1) which are too far.

Lemma 1. *Suppose that $\bar{x} \in X(N_0, N_1)$ and let $\bar{d} := ||\bar{x} - x^0||$. Assume that we can compute $\bar{\mu} \geq 0$ such that, for every x such that $||x - x^0|| \leq \bar{d}$, there exist k points $x^{i_1}, \ldots, x^{i_k}$ such that $||x - x^{i_l}|| \leq \bar{\mu}$, for all $l = 1, \ldots, k$.*

Then if x^i is such that $||x^i - x^0|| > \bar{d} + \bar{\mu}$, x^i cannot be one of the k nearest neighbors of x, for any such that $||x - x^0|| \leq \bar{d}$.

Proof. The Lemma follows from a simple application of triangle inequality, since $||x^0 - x^i|| \leq ||x - x^0|| + ||x^i - x|| \Rightarrow ||x^i - x|| \geq ||x^i - x^0|| - ||x - x^0|| > \bar{d} + \bar{\mu} - \bar{d} = \bar{\mu}$
Therefore, $x^{i_1}, \ldots, x^{i_k}$ are closer to x than x^i □

We call this operation of removing points x^i based on Lemma 1 *filtering*.

We propose two basic filtering methods to compute $\bar{\mu}$. *Filter 1* is based on finding the k nearest neighbors of $\bar{x}$. Suppose that the k nearest neighbors of $\bar{x}$ are all within a distance $\varepsilon \geq 0$ of it. Then, for any point such that $||x - x^0|| \leq \bar{d}$, we get that $||x - x^{i_l}|| \leq ||x - x^0|| + ||x^{i_l} - \bar{x}|| + ||\bar{x} - x^0|| \leq \bar{d} + \bar{d} + \varepsilon$. So we may apply Lemma 1 with $\bar{\mu} = 2\bar{d} + \varepsilon$.

Filter 2 is based on spending some more computational effort to find $\bar{\mu}$, to derive a second bound that can also be used to further reduce the size of (1). What we will do is that, for each i, we solve the following problem

$$\begin{aligned} \mu_i = \max\ & ||x - x^i|| \\ \text{s.t.}\ & ||x - x_0|| \leq \bar{d} \end{aligned} \tag{2}$$

Now, we can pick $\bar{\mu}$ to be the k-th smallest μ_i value, and then can apply Lemma 1 with that value for $\bar{\mu}$. Note that (2) is still a MIP, but it is a relatively simple one that a modern MIP solver like Gurobi can be expected to solve in a not so large computational time.

4 Partition and sampling relaxations

While the filtering procedures in Section 3 may help in reducing the number n of points, more often than not, n will still be relatively large. In this section, we propose two relaxations of $\text{cnt}(N_0, N_1)$ to further reduce the size of problem (1).

4.1 Sampling-based relaxation for dealing with N_0

The first observation is that, if we remove some of the points in N_0, we get a relaxation of $\text{cnt}(N_0, N_1)$. This is formalized in the following lemma.

Theorem 1. *For any $U_0 \subseteq N_0$ such that $|U_0 + N_1| \geq k$, $X(U_0, N_1) \supseteq X(N_0, N_1)$, or in other words, $cnt(U_0, N_1)$ is a relaxation of $cnt(N_0, N_1)$.*

Proof. Let us consider the case when $N_0 \setminus U_0 = \{l\}$.

Suppose that $x \in X(N_0, N_1)$ and let δ, Δ, λ be the values for which $(x, \delta, \Delta, \lambda)$ is feasible for $\text{cnt}(N_0, N_1)$.

If l is not one of the k nearest neighbors of x in $\text{cnt}(N_0, N_1)$ then $\lambda_l = 0$, and so $x \in X(U_0, N_1)$ follows by picking the corresponding components of $(x, \delta, \Delta, \lambda)$ that exist in $\text{cnt}(U_0, N_1)$.

If l is one of the k nearest neighbors of x in $\text{cnt}(N_0, N_1)$, then $\lambda_l = 1$. Let x^t be the $(k+1)$-th nearest neighbor of x in $\text{cnt}(N_0, N_1)$. We can then set $\lambda' = \lambda - e_l + e_t$, and note that λ' satisfies constraints (1f) and (1g), since $\sum\limits_{i \in N_0 \cup N_1} \lambda_i = \sum\limits_{i \in N_0 \cup N_1} \lambda'_i$ and $\sum\limits_{i \in N_0} \lambda'_i \leq \sum\limits_{i \in N_0} \lambda_i$.

We can, therefore, obtain that x is feasible for $cnt{-}\text{k}{-}NN(U_0, N_1)$.

For the more general case, when $|N_0 \setminus U_0| \geq 2$, let $l \in N_0 \setminus U_0$. We have shown that $X(U_0, N_1) \supseteq X(U_0 \cup \{l\}, N_1)$ and, by induction on $|N_0 \setminus U_0|$, $X(U_0 \cup \{l\}, N_1) \supseteq X(N_0, N_1)$, so the result follows. □

Theorem 1 allows us to remove a huge number of the points in N_0 and only be left with a very small number of samples from the set N_0. For this reason, we call this relaxation the *sampling* relaxation. Unfortunately, we cannot do the same for points in N_1, which can still leave us with a problem of relatively large size to solve.

4.2 Partition-based relaxation for dealing with N_1

To cope with a large number of points in N_1, we propose a partitioning-based approach. The main idea is that we will partition N_1 into different sets and then each set will be treated in a unified way.

To formalize the approach, we first start by noting that (1) can be modified so that (1f) becomes $\sum\limits_{i \in N_0 \cup N_1} \lambda_i \geq k$ (we call this new constraint (1f)' and the resulting problem (1)').

Lemma 2. *The optimal value of* (1) *and* (1)' *are equal.*

Proof. It is easy to see that (1)' is a relaxation of (1).

Now let $(x', \delta', \Delta', \lambda')$ be optimal for (1)'

Let $I := \{i \in 1, \ldots, n : \lambda'_i = 1\} = \{i_1, \ldots, i_l\}$ for some $l \geq k$. Assume WLOG that $dist(x^{i_j}, x^0) \leq dist(x^{i_{j+1}}, x^0)$ for all $j = 1, \ldots, l-1$.

Note that $dist(x^{i_k}, x^0) \leq dist(x^{i_j}, x^0) \leq \Delta'$ for all $j \geq k$. Then, we can set $\bar{\Delta} = dist(x^{i_k}, x^0)$, $\bar{\lambda}_{i_j} = 0$ for all $j > k$ and $\bar{\lambda}_{i_j} = 1$, for all $j \leq k$.

Then $(x', \delta', \bar{\Delta}, \bar{\lambda})$ is a solution to (1) of the same cost. □

Now let us consider $\mathcal{S} = \{S^1, \ldots, S^p\}$ a partition of $N_0 \cup N_1$, such that, for all $t = 1, \ldots, p$, we have:

(P.1) $S^t \neq \emptyset$;
(P.2) Either $S^t \subseteq N_0$ (in which case $c(S^t) = 0$) or $S^t \subseteq N_1$ (in which case $c(S^t) = 1$);
(P.3) If $c(S^t) = 0$ then $|S^t| = 1$.

We can now formulate the partition-based relaxation of $\text{cnt}(N_0, N_1)$ as:

$$\text{rel-cnt}(\mathcal{S}, N_0, N_1) = \begin{cases} \min & \delta_0 & \text{(3a)} \\ \text{s.t.} & \delta_0 = dist(x, x^0) & \text{(3b)} \\ & \delta_i = dist(x, x^i), \forall i \in N_0 \cup N_1 & \text{(3c)} \\ & \mu_t = \min_{i \in S^t} \delta_i, \forall t = 1, \dots, p & \text{(3d)} \\ & \lambda_t = 1 \Rightarrow \mu_t \le \Delta, \forall t = 1, \dots, p & \text{(3e)} \\ & \lambda_t = 0 \Rightarrow \mu_t \ge \Delta, \forall t = 1, \dots, p & \text{(3f)} \\ & \sum_{t=1}^{p} |S^t| \lambda_t \ge k & \text{(3g)} \\ & \sum_{t=1,\dots,p : c(S^t)=0} \lambda_t \le \lfloor k/2 \rfloor & \text{(3h)} \\ & \lambda \in \{0,1\}^p; x \in \mathbb{R}^d; \delta \in \mathbb{R}_+^{\{0\} \cup N_0 \cup N_1}; \Delta \in \mathbb{R}_+ & \text{(3i)} \end{cases}$$

Let $X'(\mathcal{S}, N_0, N_1)$ be the set of x such that there exist $(\delta, \Delta, \mu, \lambda)$ such that $(x, \delta, \Delta, \mu, \lambda)$ is feasible for rel-cnt$(\mathcal{S}, N_0, N_1)$.

Formulation rel-cnt$(\mathcal{S}, N_0, N_1)$ considers that either we pick all of the set S^t or none of it to be part of our nearest neighbor set. Moreover, we consider the distance to the set to be the smallest distance to any point in the set.

Before proving that this also leads to a relaxation, a few points to note are as follows. Even though the number of δ and x variables has not changed, the number of binary variables can be significantly smaller than the number of binary variables in cnt(N_0, N_1). Therefore, it is reasonable to expect that solving rel-cnt$(\mathcal{S}, N_0, N_1)$ may become significantly cheaper to solve computationally.

In addition, (3) can be input directly into a CP solver. However, while constraint (3d) and (3f) can be easily linearized, constraint (3e) cannot, so writing a MIP to solve (3) is not straightforward and will likely require the addition of extra binary variables, which will defeat the purpose of considering the partition in the first place. The following theorem shows that such partition leads to a relaxation of the problem.

Theorem 2. *Suppose $\mathcal{S} = (S^1, \dots, S^p)$ is a partition of $N_0 \cup N_1$ satisfying* ***(P.1)***, ***(P.2)*** *and* ***(P.3)***. *Also, suppose $p \ge 2$ and $c(S^{p-1}) = c(S^p) = 1$.*

Let $\mathcal{S}' = (S^1, \dots, S^{p-1} \cup S^p)$. Then $\mathcal{S}'$ also satisfies ***(P.1)***, ***(P.2)*** *and* ***(P.3)***. *Moreover $X'(\mathcal{S}, N_0, N_1) \subseteq X'(\mathcal{S}', N_0, N_1)$, that is, rel-cnt$(\mathcal{S}', N_0, N_1)$ is a relaxation of rel-cnt$(\mathcal{S}, N_0, N_1)$.*

Proof. Properties **(P.1)**, **(P.2)** and **(P.3)** for $\mathcal{S}'$ can be easily verified.

Let $(\bar{x}, \bar{\delta}, \bar{\Delta}, \bar{\mu}, \bar{\lambda})$ be feasible for rel-cnt$(\mathcal{S}, N_0, N_1)$.

Set $\hat{\lambda}_t = \bar{\lambda}_t$, $\hat{\mu}_t = \bar{\mu}_t$, for all $t = 1, \dots, p-2$. And set $\hat{\lambda}_{p-1} = \max\{\bar{\lambda}_{p-1}, \bar{\lambda}_p\}$, $\hat{\mu}_p = \min\{\bar{\mu}_{p-1}, \bar{\mu}_p\}$.

It is clear that (3h) is satisfied, since $\hat{\lambda}_t = \bar{\lambda}_t$ for all $t : c(S^t) = 0$.

Now let's look at (3g). Notice that

$$\sum_{t=1}^{p-2} |S^t| \hat{\lambda}_t + (|S^{p-1}| + |S^p|) \hat{\lambda}_{p-1} \ge \sum_{t=1}^{p} |S^t| \bar{\lambda}_t \ge k$$

Now constraints (3f) and (3e) are immediately satisfied if $t < p-1$, since those sets did not change.

If $\hat{\lambda}_{p-1} = 1$, then assume WLOG that $\bar{\lambda}_p = 1$. Therefore, $\bar{\Delta} \geq \bar{\mu}_p \geq \hat{\mu}_p$, so (3e) is satisfied.

If $\hat{\lambda}_{p-1} = 0$, then $\bar{\lambda}_{p-1} = \bar{\lambda}_p = 0$. Therefore, $\bar{\Delta} \leq \bar{\mu}_{p-1}$ and $\bar{\Delta} \leq \bar{\mu}_p$, so $\bar{\Delta} \leq \hat{\mu}_{p-1}$. So constraint (3f) is also satisfied.

Thus, $(\bar{x}, \bar{\delta}, \bar{\Delta}, \hat{\mu}, \hat{\lambda})$ is feasible for rel-cnt$(\mathcal{S}', N_0, N_1)$, i.e. $\bar{x} \in X'(\mathcal{S}', N_0, N_1)$ □

Convex hull relaxation for MIPs As was mentioned in Section 4, formulating problem (3) as a MIP will require the addition of extra binary variables, which is not desirable. Therefore, we propose a further relaxation that can be modeled as a MIP, based on relaxing constraint (3e) to consider the distance to the convex hull of points in S^t, instead of the minimum distance to the points in S^t.

This can be achieved by deleting constraint (3e) and replacing it with the following constraints (and adding the corresponding decision variables:

$$\sum_{j=1}^{d} |y_j^t - x_j| \leq \Delta + M(1-\lambda_t), \forall t = 1, \ldots, p, i \in S^t \tag{4}$$

$$y^t = \sum_{i \in S^t} \theta_i^t x^i, \forall t = 1, \ldots, p \tag{5}$$

$$\sum_{i \in S^t} \theta_i^t = 1, \forall t = 1, \ldots, p \tag{6}$$

$$y^t \in \mathbb{R}^d, \theta_i^t \in [0,1], \forall t = 1, \ldots, p, i \in S^t \tag{7}$$

Since the distance to the convex hull of S^t is always at most the minimum distance to the points in S^t (as $S^t \subseteq conv(S^t)$), the fact that we remain with a relaxation follows trivially. The ensuing relaxation can now be modeled as a MIP, with the expense of adding extra continuous variables and contraints only, which is preferrable to adding extra binary variables.

4.3 An iterative algorithm

We can put theorems 1 and 2 together and pick some $B \subseteq N_0$ and a partition $\mathcal{S} = (S^1, \ldots, S^p)$ of $B \cup N_1$, and it follows that rel-cnt$(\mathcal{S}, B, N_1)$ is a relaxation of cnt(N_0, N_1). If the solution $\bar{x}$ to rel-cnt$(\mathcal{S}, B, N_1)$ is in $X(N_0, N_1)$, then $\bar{x}$ is optimal for cnt(N_0, N_1). We now proceed to try to find how to modify $\mathcal{S}$ and/or B if that is not the case. We will use the notation $S(i)$ to denote the element of the partition $\mathcal{S}$ that contains i.

We start by showing the following property on $\bar{x}$.

Lemma 3. *Let $B \subseteq N_0$ and $\mathcal{S} = (S^1, \ldots, S^p)$ be a partition of $B \cup N_1$ satisfying **(P.1)**, **(P.2)** and **(P.3)**. Let $(\bar{x}, \bar{\delta}, \bar{\Delta}, \bar{\mu}, \bar{\lambda})$ be optimal for rel-cnt$(\mathcal{S}, B, N_1)$. If $\bar{x} \notin X(N_0, N_1)$ then there exists $t \in \{1, \ldots, p\}$ such that $\bar{\lambda}_t = 1$ and $|S^t| > 1$.*

Proof. Suppose that for every $t \in \{1, \ldots, p\}$ for which $\bar{\lambda}_t = 1$ we have $|S^t| = 1$.

Define $\hat{\lambda} \in \{0,1\}^n$ as follows: $\hat{\lambda}_i = 1$, if $S(i) = S^t$ and $\bar{\lambda}_t = 1$, and 0 otherwise. Note that $\hat{\lambda}$ will immediately satisfy (1f) and (1g).

Now from (3e), it follows that $\bar{\Delta} \geq \bar{\delta}_i$ for all $i : \hat{\lambda}_i = 1$.

Moreover, for all t such that $\bar{\lambda}_t = 0$, we get that $\bar{\Delta} \leq \bar{\mu}_t \leq \bar{\delta}_i$ for all $i \in S^t$.

Therefore constraints (1d) and (1e) are also satisfied.

Therefore $(\bar{x}, \bar{\delta}, \bar{\Delta}, \hat{\lambda})$ is feasible for $\text{cnt}(N_0, N_1)$ and thus $\bar{x} \in X(N_0, N_1)$ □

With this, now we can determine how to proceed if $\bar{x} \notin X(N_0, N_1)$.

Theorem 3. *Let $B \subseteq N_0$ and $\mathcal{S} = (S^1, \ldots, S^p)$ be a partition of $B \cup N_1$ satisfying **(P.1)**, **(P.2)** and **(P.3)**. Let $(\bar{x}, \bar{\delta}, \bar{\Delta}, \bar{\mu}, \bar{\lambda})$ be optimal for rel-cnt$(\mathcal{S}, B, N_1)$, such that $\bar{x} \notin X(N_0, N_1)$. Let $x^{i_1}, \ldots, x^{i_k}$ be the k nearest neighbors of $\bar{x}$ in $\mathcal{O}$. Then either there exists $l \in \{1, \ldots, k\}$ such that $i_l \in N_0 \setminus B$ or there exists $l \in \{1, \ldots, k\}$ such that $|S(i_l)| > 1$.*

Proof. We prove the contrapositive. Suppose that for all $l \in \{1, \ldots, k\}$ either $i_l \in B$ or $|S(i_l)| = 1$. But note that, property **(P.3)** implies that $i_l \in B \Rightarrow |S(i_l)| = 1$, so in fact, we may write that for all $l \in \{1, \ldots, l\}$, $|S(i_l)| = 1$.

We may also assume (by proceeding as in the proof of Lemma 2) that there are exactly k values for which $\bar{\lambda}_t = 1$.

But these values of t for which $\bar{\lambda}_t = 1$ represent the sets S^t that are closest to $\bar{x}$, where the distance from $\bar{x}$ to S^t is represented by $\bar{\mu}_t$.

But since each one of the sets $S(i_l)$ is a distinct set in $\mathcal{S}$, the sets $S(i_1), \ldots, S(i_p)$ are the k closest to $\bar{x}$, so $\bar{\lambda}_{i_1} = \ldots = \bar{\lambda}_{i_p} = 1$. But this contradicts Lemma 3 □

These results lead to algorithm 1 which is an iterative algorithm for solving $\text{cnt}(N_0, N_1)$. We note that the following corollary is also immediate from Theorems 1 and 2:

Corollary 1. *Let $\bar{z}_q$ be the bound obtained when line 6 of Algorithm 1 is executed for the q-th time. Then $\bar{z}_{q+1} \geq \bar{z}_q$.*

We end this section by commenting that the partition-based relaxation and its refinement in Algorithm 1 shares similarities with dynamic discretization [13] approaches, where discretization points are clustered and clusters are subsequently refined. In addition, partition-based approaches to stochastic programs [10,12] also share a similar philosophy.

5 Computational experiments

To test the several different approaches, we performed computational experiments in several datasets from the literature. We present the results in this section. All implementations were made in Python 3.8, using Gurobi 8.1.1 [6] as a MIP solver and CP-SAT from OR-Tools [9] as a constraint programming solver. All experiments were run in single-thread mode in a machine with Intel Xeon Gold 6142 CPU 2.60GHz processors, and 264GB of RAM.

Algorithm 1: Algorithm to solve (3)

```
1  Set $\mathcal{S} = \{N_1\}$
2  $B \leftarrow \emptyset$
3  $LB \leftarrow 0$
4  $UB \leftarrow \infty$
5  while $UB - LB > \epsilon$ do
6      Let $\bar{x}, \bar{\lambda}$ be optimal for rel-cnt$(\mathcal{S}, B, N_1)$, with value $\bar{z}$
7      $LB \leftarrow \max\{LB, \bar{z}\}$
8      if $\bar{x}$ is feasible for cnt$(N_0, N_1)$ then
9          $UB \leftarrow \bar{z}$
10     else
11         Let $x^{i_1}, \dots, x^{i_k}$ be the $k$ nearest neighbors of $\bar{x}$ in $\mathcal{O}$.
12         for $l = 1, \dots, k$ do
13             if $y(x^{i_l}) = 1$ and $|S(i_l)| > 1$ then
14                 $\mathcal{S} \leftarrow \mathcal{S} \setminus \{S(i_l)\} \cup \{\{i_l\}, \{S(i_l) \setminus \{i_l\}\}$
15             if $y(x^{i_l}) = 0$ and $i_l \notin B$ then
16                 $B \leftarrow B \cup \{i_l\}$
17                 $\mathcal{S} \leftarrow \mathcal{S} \cup \{\{i_l\}\}$
```

5.1 Instances

We took the 9 base instances that were used in the paper [7] consisting on a variety of data sets from different sources and applications. Each of these base instances may have several thousands of data points, so we generated our instances based on picking (at random) a subset of n points for n being the minimum between the number of data points and a parameter $TCAP \in \{500, 1000, \dots, 4000\}$. The selection of the points was made via a direct call to the `train_test_split` function in the `sk-learn` package [8]. In addition, we tested each instance by picking different values of $k \in \{5, 15, 25, 35\}$.

We also note that each instance contains specification of possible actionability constraints that were already implemented in [4]. These restrict some features to being fixed (e.g. gender at birth cannot be changed), or restricted to being increasing only (e.g. age), and other types of restrictrions. These are implemented as explicit bound constraints on the x variables and are taken as part of the input as well.

5.2 Evaluation

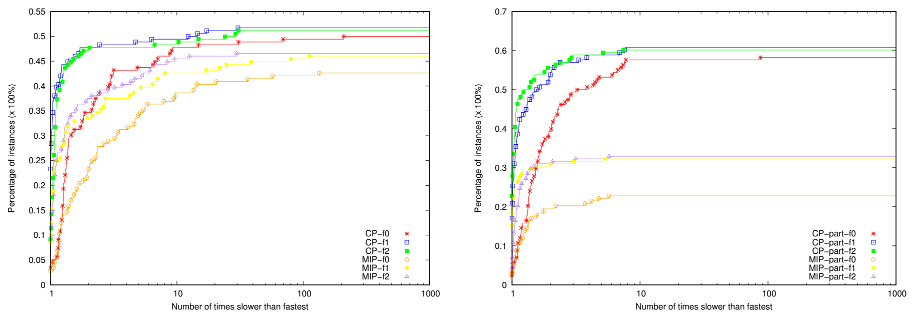
To evaluate each possible approach, we use performance profiles [3], which are distribution functions of a performance metric (time). A performance profile is computed as follows. Given a set of methods M, and a set of instances I, we compute, for each $m \in M$, $i \in I$, the desired metric t_{im}. We then compute the best performance for each instance $\tau_i := \min_{m \in M} t_{im}$, and then for each $m \in M$, $i \in I$, we compute the ratio $r_{im} = \frac{t_{im}}{\tau i}$. Such ratio represents how many times worse than the best a method performed in an instance. With this, the performance profile consists of a set of lines (one for each $m \in M$) where a point (x, y) in a line represent that in y percent of the instances, the ratio r_{im} was at most x. It can be seen as a cumulative graph of the r_{im} values.

These types of graphs help identify benchmark trends by normalizing "easy" and "hard" instances in similar ways. They give immediate ways to identify better approaches that are more informative than averages. An easy way to read such graphs is to see which lines are on top. *Upper and to the left* means better. We considered a time limit of 3600s for all approaches and, for the performance profiles, we considered $t_{im} = \infty$ if $t_{im} \geq 3600$, with $\frac{\infty}{\infty} := \infty$.

5.3 Effect of filtering

We test our filters by taking as an initial feasible solution $\bar{x} = x^i$ for each data point in the input for which $c(x^i) = 1$. Three different filtering settings were tested: No filter (`f0`), Filter type 1 (`f1`) and Filter type 2 (`f2`).

The methods tested are: `CP` and `MIP` which just solve problem (1) directly using CP/MIP solvers; `CP-part` and `MIP-part` which solve problem (3) using Algorithm 1. Note that `MIP-part` uses also the convex hull relaxation from Section 4.2. Approaches `MIP-part` and `CP-part` were used by initializing B to have the data points x^i with $y(x^i) = 0$ among the $2k$ nearest neighbors of x^0, and initializing $\mathcal{S}$ to having one set for each of $2k$ nearest neighbors of x^0, plus one set containing all remaining elements in N_1.



(a) Baseline approaches (b) Partition and sampling relaxations

Fig. 1: Results of different filters

Figure 1 shows the results for both the baseline approaches (`CP` and `MIP`), as well as for the approaches based on Algorithm 1. It is clear from these figures that the idea of filtering helps significantly, with `F1` being the best setting for `CP`, and `F2` the best for `MIP`, `MIP-part`, `CP-part`.

It is also clear that CP-based approaches significantly outperform MIP-based ones. This may be due to the fact that LP-relaxations for the MIP-based formulations are bad (lower bounds at the root are often 0.0, which is the trivial lower bound for minimizing distance), typically giving little information about what the integer solution should look like.

5.4 Overall results

In this section, we compare the best filter settings of each approach against each other. In addition, we leave the setting `MIP-f0` in the experiments, since this is the baseline MIP formulation that was proposed in [4], so it represents the best in the literature so far.

An additional approach was tested, with the rationale that using a single partition S^t for all elements in N_1 may not be too good, since it may treat data points that are very far from each other in a uniform manner. With that in mind, we also tested the `-km` variation, which initializes $\mathcal{S}$ with clusters computed using the k-means algorithm implemented in [8].

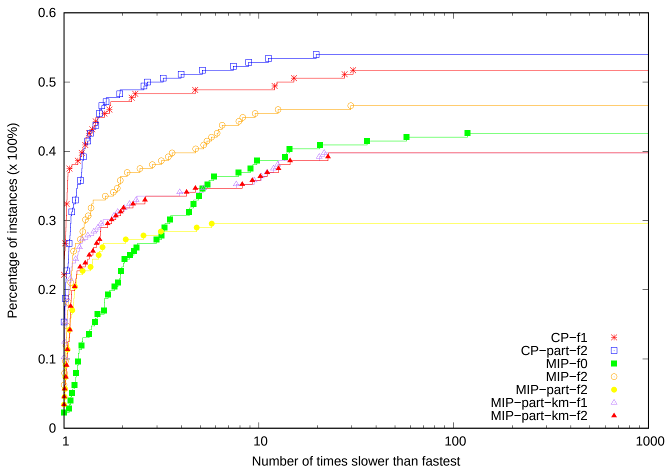


Fig. 2: Overall comparison of approaches

Figure 2 shows that both Algorithm 1 and filtering made a significant difference in improving solution times for the problem. In addition, it is clear that for MIP-based approaches, the best option is to just use filtering and give the resulting problem to a MIP solver directly. Once more, we suspect that while the initial LP relaxation is bad, the further relaxation that needed to be made in Section 4.2 weakens it even more, while making the solution to each subproblem not fast enough.

The average (shifted geometric average with a shift of 1s) solution time decreased from 780s in the baseline `MIP-f0` to 451s in `CP-part-f2` (for the purposes of average computation, a solution time of 3600s were considered for all runs which did not terminate within the time limit). We use shifted geometric mean since it reduces influence of outliers that are "too big" or "too small". See [1] for more discussions on the topic.

6 Conclusion

In this work, we proposed several techniques to speed up the computation of counterfactual explanations for k-NN. The overall combined effect of these techniques was an average reduction of 42% in running time.

While performing more refined techniques to reduce the dimension may lead to additional time savings, other possible research directions are designing better dual bounds for MIP formulations. In addition, a more integrated approach, where solution to previous MIPs/CPs can be utilized to speed up computation of similar, but more refined MIPs/CPs is a promising area of improvement.

References

1. Achterberg, T.: Constraint Integer Programming. Ph.D. thesis, TU Berlin (2007)
2. Artelt, A., Hammer, B.: On the computation of counterfactual explanations–a survey. arXiv preprint arXiv:1911.07749 (2019)
3. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Mathematical Programming **91**, 201–213 (2002)
4. Forel, A., Parmentier, A., Vidal, T.: Explainable data-driven optimization: From context to decision and back again (2023)
5. Guidotti, R.: Counterfactual explanations and how to find them: literature review and benchmarking. Data Mining and Knowledge Discovery pp. 1–55 (2022)
6. Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual (2023), https://www.gurobi.com
7. Parmentier, A., Vidal, T.: Optimal counterfactual explanations in tree ensembles. In: International Conference on Machine Learning. pp. 8422–8431. PMLR (2021)
8. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al.: Scikit-learn: Machine learning in python. Journal of machine learning research **12**(Oct), 2825–2830 (2011)
9. Perron, L., Didier, F.: CP-SAT, https://developers.google.com/optimization/cp/cp_solver/
10. Roland, M., Forel, A., Vidal, T.: Adaptive partitioning for chance-constrained problems with finite support (2023)
11. Russell, C.: Efficient search for diverse coherent explanations. In: Proceedings of the Conference on Fairness, Accountability, and Transparency. pp. 20–28 (2019)
12. Song, Y., Luedtke, J.: An adaptive partition-based approach for solving two-stage stochastic programs with fixed recourse. SIAM Journal on Optimization **25**(3), 1344–1367 (2015)
13. Vu, D.M., Hewitt, M., Boland, N., Savelsbergh, M.: Dynamic discretization discovery for solving the time-dependent traveling salesman problem with time windows. Transportation science **54**(3), 703–720 (2020)