A Kolmogorov-distance based approximation of discrete random variables

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present an algorithm that takes a discrete random variable X and a number m and computes a random variable whose support (set of possible outcomes) is of size at most m and whose Kolmogorov distance from X is minimal.

1 Introduction

Many different approaches to approximation of probability distributions are studied in the literature [12, 15, 16]. The approaches vary in the types random variables considered, how they are represented, and in the criteria used for evaluation of the quality of the approximations. This paper is on approximating discrete distributions represented as explicit probability mass functions with ones that are simpler to store and to manipulate. This is needed, for example, when a discrete distribution is given as a large data-set, obtained, e.g., by sampling, and we want to represent it approximately with a small table.

The main contribution of this paper is an efficient algorithm for computing the best possible approximation of a given random variable with a random variable whose complexity is not above a prescribed threshold, where the measures of the quality of the approximation and the complexity of the random variable are as specified in the following two paragraphs.

We measure the quality of an approximation by the distance between the original variable and the approximate one. Specifically, we use the Kolmogorov distance which is commonly used for comparing random variables in statistical practice and literature. Given two random variables X and X' whose cumulative distribution functions (cdfs) are F_X and $F_{X'}$, respectively, the Kolmogorov distance between X and X' is $d_K(X, X') = \sup_t |F_X(t) - F_{X'}(t)|$ (see, e.g., [9]). We say taht X' is a good approximation of X if $d_K(X, X')$ is small.

The complexity of a random variable is measured by the size of its support, the number of values that it can take, $|\operatorname{support}(X)| = |\{x : Pr(X = x) \neq 0\}|$. When distributions are maintained as explicit tables, as done in many implementations of statistical software, the size of the support of a variable is proportional to the amount of memory needed to store it and to the complexity of the computations around it. In summary, the exact notion of optimality of the approximation targeted in this paper is:

Submitted to 32nd Conference on Neural Information Processing Systems (NIPS 2018). Do not distribute.

Definition 1. *A random variable X' is an optimal m-approximation of a random variable X if $|\mathrm{support}(X')| \leq m$ and there is no random variable X'' such that $|\mathrm{support}(X'')| \leq m$ and $d_K(X, X'') < d_K(X, X')$.*

The main contribution of the paper is an efficient algorithm that takes X and m as parameters and constructs an optimal m-approximation of X.

The rest of the paper is organized as follows. In Section 2 we describe how our work relates to other algorithms and problems studied in the literature. In Section 3 we detail the proposed algorithm, analyze its properties, and prove the main theorem. In Section 4 we demonstrate how the proposed approach performs on the problem of estimating the probability of hitting deadlines is plans and compare it to alternatives approximation approaches from the literature. We also demonstrate the performance of our approximation algorithm on some randomly generated random variables. The paper is concluded with a discussion in Section 5.

2 Related work

The most relevant work related to this paper is the papers by Cohen at. al. [5, 4]. These papers study approximations of random variables in the context of estimating deadlines. In this context, X' is defined to be a good approximation of X if $F_{X'}(t) > F_X(t)$ for any t and $\sup_t F_{X'}(t) - F_X(t)$ is small. This is not a distance because it is not symmetric. The motivation given by Cohen at. al. for using this type of approximation is for cases where overestimation of the probability of missing a deadline is acceptable but underestimation is not. In Section 4, we consider the same examples examined by Cohen at. al. and show how the algorithm proposed in this paper performs relative to the algorithms proposed there when both over- and under- estimations are allowed. As expected, the Kolmogorov distance between the approximation and the original random variable is smaller by a factor of one half, on average, when using the algorithm proposed here.

Another relevant prior work is the theory of Sparse Approximation (aka Sparse Representation) that deals with sparse solutions for systems of linear equations, as follows.

Given a matrix $D \in \mathbb{R}^{n \times p}$ and a vector $x \in \mathbb{R}^n$, the most studied sparse representation problem is finding the sparsest possible representation $\alpha \in \mathbb{R}^p$ satisfying $x = D\alpha$:

$$\min_{\alpha \in \mathbb{R}^p} \|\alpha\|_0 \text{ subject to } x = D\alpha.$$

where $\|\alpha\|_0 = |\{i : \alpha_i \neq 0,\, i = 1, \ldots, p\}|$ is the ℓ_0 pseudo-norm, counting the number of non-zero coordinates of α. This problem is known to be NP-Hard with a reduction to NP-complete subset selection problems.

In these terms, using also the ℓ_∞ norm that represents the maximal coordinate and the ℓ_1 norm that represents the sum of the coordinates, our problem can be phrased as:

$$\min_{\alpha \in [0,\infty)^p} \|x - D\alpha\|_\infty \text{ subject to } \|\alpha\|_0 = m \text{ and } \|\alpha\|_1 = 1.$$

where D is the all-ones triangular matrix (the entry at row i and column j is one if $i \leq j$ and zero otherwise), x is related to X such that the ith coordinate of x is $F_X(x_i)$ where $\mathrm{support}(X) = \{x_1 < x_2 < \cdots < x_n\}$ and α is related to X' such that the ith coordinate of α is $f_{X'}(x_i)$. The functions F_X and $f_{X'}$ represent, respectively, the cumulative distribution function of X and the mass distribution function of X'. This, of course, means that the coordinates of x are assumed to be positive and monotonically increasing and that the last coordinate of x is assumed to be one. We demonstrate an application for this specific sparse representation problem and show that it can be solve in $O(n^2m)$ time and $O(m^2)$ memory.

The presented work is also related to the research on binning in statistical inference. Consider, for example, the problem of credit scoring [21] that deals with separating good applicants from bad applicants where the Kolmogorov–Smirnov statistic KS is a standard measure. The KS comparison is often preceded by a procedure called binning where a large table is translated to a smaller one by collecting nearby values together. There are many methods for binning [11, 17, 2, 19]. In this context, our algorithm can be consider as a new binning strategy that provides optimality guarantees with respect to the Kolmogorov distance that none of the existing binning technique that we are aware of provides.

The present study is also related to the work of Pavlikov and Uryasev [15], where a procedure for producing a random variable X' that optimally approximates a random variable X is presented. Their approximation scheme, achieved using linear programming, is designed for a different notion of distance (called CVaR). The new contribution of the present work in this context is that our method is direct, not using linear programming, thus allowing tighter analysis of time and memory complexity. Also, our method is designed for optimizing the Kolmogorov distance that is more prevalent in applications. For comparison, in Section 4 we briefly discuss the performance of linear programming approach similar to the one proposed in [15] for the Kolmogorov distance and compare it to the algorithm proposed in this paper.

3 An algorithm for optimal approximation

In the scope of this section, let X be a given random variable with a finite support of size n, and let $0 < m \leq n$ be a given complexity bound. The section evolves by developing notations and by collecting facts towards an algorithm for finding an optimal m-approximation of X.

The first useful fact is that it is enough to limit our search to approximations X's such that $\mathrm{support}(X') \subseteq \mathrm{support}(X)$:

Lemma 2. *For every random variable X'' there is a random variable X' such that $\mathrm{support}(X') \subseteq \mathrm{support}(X)$ and $d_K(X, X') \leq d_K(X, X'')$.*

Proof. Let $\{x_1, \ldots, x_n\} = \mathrm{support}(X)$, and let $x_0 = -\infty, x_{n+1} = \infty$. Consider the random variable X' whose probability mass function is $f_{X'}(x_i) = P(x_{i-1} < X'' \leq x_i)$ for $i = 1, \ldots, n-1$, $f_{X'}(x_n) = P(x_n - 1 < X'' < x_{n+1})$, and $F_{X'}(x) = 0$ if $x \notin \mathrm{support}(X)$. Since X' only "pushes" the probability mass of X'' to the support of X, we have that $f_{X'}$ is a probability mass function and therefore X' is well defined. By construction, $|F_X(x_i) - F_{X'}(x_i)| = |F_X(x_i) - F_{X''}(x_i)|$ for every $1 \leq i \leq n-1$. For $i = n$ we have $|F_X(x_n) - F_{X'}(x_n)| = |1 - 1| = 0$. Since $|F_X(x) - F_{X'}(x)| = |F_X(x_i) - F_{X'}(x_i)|$ for every $0 \leq i < n+1$ and $x_i < x < x_{i+1}$, we have that $d_K(X, X') = max_i |F_X(x_i) - F_{X'}(x_i)| \leq max_i |F_X(x_i) - F_{X''}(x_i)| \leq d_K(X, X'')$. □

For a set $S \subseteq \mathrm{support}(X)$, let $\mathbb{X}_S$ denote the set of random variables whose supports are contained in S. In Step 1 below, we find a random variable in $\mathbb{X}_S$ that minimizes the Kolmogorov distance from X. We denote the Kolmogorov distance between this variable and X by $\varepsilon(X, S)$. Then, in Step 2, we show how to efficiently find a set $S \subseteq \mathrm{support}(X)$ whose size is smaller or equal to m that minimizes $\varepsilon(X, S)$. Then, in Step 3, an optimal m-approximation is constructed by taking a minimal approximation in $\mathbb{X}_{\mathbb{S}}$ where S is the set that that minimizes $\varepsilon(X, S)$.

Step 1: Finding an X' in $\mathbb{X}_S$ that minimizes $d_K(X, X')$

We first fix a set $S \subseteq \text{support}(X)$ of size at most m, and among all the random variables in $\mathbb{X}_S$ find one with a minimal distance from X. Denote the elements of S in increasing order by $S = \{x_1 < \cdots < x_m\}$ and let $x_0 = -\infty$ and $x_{m+1} = \infty$. For every $1 < i \leq m$ let $\hat{x}_i$ be the maximal element of $\text{support}(X)$ that is smaller than x_i. Consider the following weight function

Definition 3. *For $0 \leq i \leq m$ let*

$$w(x_i, x_{i+1}) = \begin{cases} P(x_i < X < x_{i+1}) & \textit{if } i = 0 \textit{ or } i = m; \\ P(x_i < X < x_{i+1})/2 & \textit{otherwise.} \end{cases}$$

Note that $P(x_i < X < x_{i+1}) = F_X(\hat{x}_{i+1}) - F_X(x_i)$, a fact that we will use throughout this section.

Definition 4. *Let $\varepsilon(X, S) = \max\limits_{i=0,\ldots,m} w(x_i, x_{i+1})$.*

We first show that $\varepsilon(X, S)$ is a lower bound for the distance between random variable in $\mathbb{X}_S$ and X. Then, we present a random variable $X' \in \mathbb{X}_S$ such that $d_K(X, X') = \varepsilon(X, S)$. It then follows that X' is an optimal m-approximation random variable among all random variables in $\mathbb{X}_S$.

Proposition 5. *If $X' \in \mathbb{X}_S$ then $d_K(X, X') \geq \varepsilon(X, S)$.*

Proof. By definition, for every $0 \leq i \leq m$, $d_K(X, X') \geq \max\{|F_X(\hat{x}_{i+1}) - F_{X'}(\hat{x}_{i+1})|, |F_X(x_i) - F_{X'}(x_i)|\}$. Note that $F_{X'}(\hat{x}_{i+1}) = F_{X'}(x_i)$ since the probability values for all the elements not in S are set to 0.

If $i = 0$, that is $x_i = -\infty$, we have that $F_X(x_i) = F_{X'}(x_i) = F_{X'}(\hat{x}_{i+1}) = 0$ and therefore $d_K(X, X') \geq |F_X(\hat{x}_{i+1})| = |F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_i < X < x_{i+1}) = w(x_i, x_{i+1})$.

If $i = m$, that is $x_{i+1} = \infty$, we have that $F_X(\hat{x}_{i+1}) = F_{X'}(\hat{x}_{i+1}) = F_{X'}(x_i) = 1$. and therefore $d_K(X, X') \geq |1 - F_X(\hat{x}_i)| = |F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_i < X < x_{i+1}) = w(x_i, x_{i+1})$.

Otherwise for every $1 \leq i < m$, we use the fact that $max\{|a|, |b|\} \geq |a - b|/2$ for every $a, b \in \mathbb{R}$, to deduce that $d_K(X, X') \geq 1/2|F_X(\hat{x}_{i+1}) - F_X(x_i) + F_{X'}(x_i) - F_{X'}(\hat{x}_{i+1})|$. So $d_K(X, X') \geq 1/2|F_X(\hat{x}_{i+1}) - F_X(x_i)| = P(x_1 < X < x_2)/2 = w(x_i, x_{i+1})$.

Since $d_K(X, X') \geq w(x_i, x_{i+1})$ for every $0 \leq i \leq m$, the proof follows by the definition of $\varepsilon(X, S)$. □

Next we describe a random variable $X' \in \mathbb{X}_S$ with a distance of $\varepsilon(X, S)$ from X. Thus X' is an optimal m-approximation among the set $\mathbb{X}_S$. The variable X' is described by its probability mass function:

Definition 6. *Let $f_{X'}(x_i) = w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)$ for $i = 1, \ldots, m$ and $f_{X'}(x) = 0$ for $x \notin S$.*

We first show that X' is a properly defined random variable:

Lemma 7. *$f_{X'}$ is a probability mass function.*

Proof. From definition $f_{X'}(x_i) \geq 0$ for every i. To see that $\sum_i f_{X'}(x_i) = 1$, we have $\sum_i f_{X'}(x_i) = \sum_i (w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)) = \sum_{x_i \in S} f_X(x_i)) + w(x_0, x_1) + \sum_{0<i<m} 2w(x_i, x_{i+1}) + w(x_m, x_{m+1}) = \sum_{x_i \in S} P(X{=}x_i) + P(x_0{<}X{<}x_1) + \sum_{0<i<m} P(x_i < X < x_{i+1}) + P(x_m < X < x_{m+1}) = 1$ since this is the entire support of X. □

Note that X' can be constructed in time linear in the size of the support of X. Its main property, of course, the distance between the cumulative distribution functions of X and X' are bounded by $w(x_i, x_{i+1})$, as follows:

Lemma 8. *Let $x \in \mathrm{support}(X)$ and $0 \leq i \leq m$ be such that $x_i \leq x \leq x_{i+1}$ then $-w(x_i, x_{i+1}) \leq F_X(x) - F_{X'}(x) \leq w(x_i, x_{i+1})$.*

Proof. We prove by induction on $0 \leq i < m$.

First see that $F_{X'}(j) = 0$ for every $x_0 < j < x_1$ and therefore $F_X(j) - F_{X'}(j) = F_X(j) - 0 \leq F_X(\hat{x}_1) = F_X(\hat{x}_1) - F_X(x_0) = w(x_0, x_1)$. For $j = x_1$ we have $F_X(x_1) - F_{X'}(x_1) = F_X(\hat{x}_1) + f_X(x_1) - (w(x_0, x_1) + w(x_1, x_2) + f_X(x_1) = w(x_0, x_1) + f_X(x_1) - (w(x_0, x_1) + w(x_1, x_2) + f_X(x_1)) = -w(x_1, x_2)$.

Next assume that $F_X(\hat{x}_i) - F_{X'}(\hat{x}_i) = w(x_{i-1}, x_i)$. Then $F_X(x_i) - F_{X'}(x_i) = F_X(\hat{x}_i) + f_X(x_i) - (w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)) = w(x_{i-1}, x_i) + f_X(x_i) - (w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)) = -w(x_i, x_{i+1})$.

As before we have that for all $x_i < j < x_{i+1}$, we have $F_X(j) - F_{X'}(j) = F_X(j) - F_{X'}(\hat{x}_{i+1}) \leq F_X(\hat{x}_{i+1}) - F_{X'}(\hat{x}_{i+1})$. Then $F_X(\hat{x}_{i+1}) - F_{X'}(\hat{x}_{i+1}) = (F_X(x_i) + P(x_i < x < x_{i+1})) - F_{X'}(x_i) = -w(x_i, x_{i+1}) + 2w(x_i, x_{i+1}) = w(x_i, x_{i+1})$.

Finally for $x_m \leq j \leq x_{m+1}$ we have that $F_{X'}(x_m) = 1$ therefore $F_X(x_m) - F_{X'}(x_m) = (1 - P(x_m < X < x_{m+1})) - 1 = P(x_m < X < x_{m+1}) = w(x_m, x_{m+1})$, and for every $x_m < j < x_{m+1}$ we have $F_X(j) - F_{X'}(j) < (1 - P(x_m < X < x_{m+1})) - 1 < -P(x_m < X < x_{m+1})) = -w(x_m, x_{m+1})$ as required. □

From Lemma 8, by the definition of $\varepsilon(X, S)$, we then have:

Corollary 9. $d_K(X, X') = \varepsilon(X, S)$.

From Proposition 5 we also have:

Corollary 10. *$\varepsilon(X, S)$ is the distance between X and the variable closest to it in $\mathbb{X}_S$.*

Step 2: Finding a set S that minimizes $\varepsilon(X, S)$

We proceed to finding an S that minimizes $\varepsilon(X, S)$. To obtain that we use a graph search approach motivated by a method described in [3]. We construct a directed graph with a source and a target in which each source-to-target path of length smaller or equal to m corresponds to a possible support set of the same size, and the weights along that path correspond to the weight as defined in Definition 3. Thus the problem of finding an S that minimizes $\varepsilon(X, S)$ is reduced to the problem of finding a source-to-target path $\vec{p}$ of length smaller or equal to m in that graph such that the maximal weight of an edge in $\vec{p}$ is minimal among all other such maximal edges in all other such paths.

More specifically, the vertexes of the graph are $V = \mathrm{support}(X) \cup \{-\infty, \infty\}$ and the edges, E, are all the pairs $(x_1, x_2) \in V^2$ such that $x_1 < x_2$. The weight of each edge is as specified in Definition 3. Note that there is a one-to-one correspondence between a set $S \subseteq \mathrm{support}(X)$ of size m, and an $-\infty$-to-∞ path $\vec{p}_S$ in G obtained by removing the $-\infty$ and ∞ from the path in one way and by adding these elements and the sorting on the other way. With this correspondence the maximal weight of an edge on $\vec{p}_S$ is $\varepsilon(X, S)$. We denote this maximal weight of an edge by $w(\vec{p}_S)$, and denote the set of all acyclic $-\infty$-to-∞ paths in G with at most m edges by $paths_m(G, -\infty, \infty)$. Thus, the problem of finding the set S with the minimal $\varepsilon(X, S)$ is now reduced to the problem of finding a path $\vec{p} \in paths_m(G, -\infty, \infty)$ such that $w(\vec{p})$ is minimal among all $\{w(\vec{p'}) \colon \vec{p'} \in$

$paths_m(G, -\infty, \infty)\}$. This problem can be solved by a variant of the Bellman-Ford algorithm as described in [10, 18].

Step 3: Constructing the overall algorithm

We combine Step 1 and Step 2 in the following algorithm called $\mathrm{KolmogorovApprox}$ (Algorithm 1) that follows naturally from the two steps. Given X and $\mathrm{support}(X)$ we add x_0, x_{n+1} and construct the graph (line 2) as in Step 2. Then we execute a variant of the Bellman-Ford algorithm on G for m iterations to obtain a path $\vec{p} = (v_0, \ldots, v_{m+1})$ (line 2) as described in Corollary **??**. Finally we use Definition 6 to construct X' from the weights of $\vec{p}$ (lines 4-5).

Algorithm 1: $\mathrm{KolmogorovApprox}(X, m)$

1 Construct a weighted graph $G = (V, E)$ where $V = \mathrm{support}(X) \cup \{-\infty, \infty\}$,
$E = \{(x_1, x_2) \in V^2 \colon x_1 < x_2\}$, and the weights are as in Definition 3.

2 Find a path $\vec{p} = (v_0, \ldots, v_{m+1})$ such that $w(\vec{p})$ is minimal among all
$\{w(\vec{p'}) \colon \vec{p'} \in paths_m(G, -\infty, \infty)\}$.

3 Return a random variable whose probability mass function is
$f_{X'}(x_i) = w(x_{i-1}, x_i) + w(x_i, x_{i+1}) + f_X(x_i)$ for all $i = 1, \ldots, m$ and zero otherwise.

Theorem 11. $\mathrm{KolmogorovApprox}$ *returns an m-optimal-approximation of X.*

Proof. By the construction of G we get that the path $\vec{p}$ obtained in line 4 of $\mathrm{KolmogorovApprox}$ describes a set S of support of size at most m for which $\varepsilon(S, X)$ is minimal. Then from Definition 6 and Corollary 9 we construct X' in lines 4-5 of $\mathrm{KolmogorovApprox}$ such that $d_K(X, X') = \varepsilon(X, S)$. Therefore X' is an m-approximation among all random variables with support contained in $\mathrm{support}(X)$. Finally from Lemma 2 we have that X' is m-approximation among all random variables os support of size at most m, thus X' is an m-optimal-approximation of X. $\square$

Finally we analyze the complexity of $\mathrm{KolmogorovApprox}$ as follows.

Theorem 12. *The* $\mathrm{KolmogorovApprox}(X, m)$ *algorithm runs in time $O(mn^2)$, using $O(n^2)$ memory where $n = |\,\mathrm{support}(X)|$.*

Proof. Constructing the graph G as described in Step 2 takes $O(n^2)$ time and memory. The number of edges in G is $O(|E|) = O(n^2)$, for every edge the weight is at most the sum of all probabilities between the source node s and the target node t, which can be calculated efficiently by aggregating the weights of already calculated edges. The construction is also the only stage that requires memory allocation, specifically $O(|E| + |V|) = O(n^2)$. Next using the Bellman-Ford algorithm on G for m iterations takes $O(m(|E|+|V|)) \approx O(mn^2)$. [[DF: cite Corman or some algorithms book]]. Finally deriving the new random variable X' from the computed path $\vec{p}$ takes $O(m)$ time: For every node x_i in $\vec{p}$ (at most m nodes), use the already calculated weights to find the probability mass function $f_{X'}(x_i)$. To conclude, the time complexity of $\mathrm{KolmogorovApprox}(X, m)$ is $O(n^2 + mn^2 + m) = O(mn^2)$ and memory complexity is $O(E + V) = O(n^2)$. $\square$

4 A case study and experimental results

The case study examined in our experiments is the problem of task trees with deadlines [5, 4]. Hierarchical planning is a well-established field in AI [6, 7, 8], and is still relevant nowadays [1, 20]. A hierarchical plan is a method for representing problems of automated planning in which the dependency among tasks can be given in the form of networks, here we focus on hierarchical plans represented by task trees. The leaves in a task tree are *primitive* actions (or tasks), and the internal nodes are either *sequence* or *parallel* actions. The plans we deal with are of stochastic nature, and the task duration is described as probability distributions in the leaf nodes. We assume that the distributions are independent but *not* necessarily identically distributed and that the random variables are discrete and have a finite support.

A sequence node denotes a series of tasks that should be performed consecutively, whereas a parallel node denotes a set of tasks that begin at the same time. A *valid* plan is one that is fulfilled before some given *deadline*, i.e., its *makespan* is less than or equal to the deadline. The objective in this context is to compute the probability that a given plan is valid, or more formally computing $P(X < T)$, where X is a random variable representing the makespan of the plan and T is the deadline. The problem of finding the probability that a task tree satisfies a deadline is known to be NP-hard. In fact, even the problem of summing a set of random variables is NP-hard [13]. This is an example of an explicitly given random variable that we need to estimate deadline meeting probabilities for.

The first experiment we focus on is the problem of task trees with deadlines, and consider three types of task trees. The first type includes logistic problems of transporting packages by trucks and airplanes (from IPC2 http://ipc.icaps-conference.org/). Hierarchical plans of those logistic problems were generated by the JSHOP2 planner [14], one parallel node with all descendant task nodes being in sequence. The second type consists of task trees used as execution plans for the ROBIL team entry in the DARPA robotics challenge (DRC simulation phase), and the third type is of linear plans (sequential task trees). The primitive tasks in all the trees are modeled as discrete random variables with support of size M obtained by discretization of uniform distributions over various intervals. The number of tasks in a tree is denoted by N.

We implemented the approximation algorithm for solving the deadline problem with four different methods of approximation. The first two are for achieving a one-sided Kolmogorov approximation – the $\mathrm{OptTrim}$ [4] and the Trim [5] operators, and the third is a simple sampling scheme. We used those methods as a comparison to the Kolmogorov approximation with the suggested $\mathrm{KolmogorovApprox}$ algorithm. The parameter m of $\mathrm{OptTrim}$ and $\mathrm{KolmogorovApprox}$ corresponds to the inverse of ε given to the Trim operator. Note that in order to obtain some error ε, one must take into consideration the size of the task tree N, therefore, $m/N = 1/(\varepsilon \cdot N)$. We ran also an exact computation as a reference to the approximated one in order to calculate the error. The experiments conducted with the following operators and their parameters: $\mathrm{KolmogorovApprox}$ operator with $m = 10 \cdot N$, the $\mathrm{OptTrim}$ operator with $m = 10 \cdot N$, the Trim as operator with $\varepsilon = 0.1/N$, and two simple simulations, with a different samples number $s = 10^4$ and $s = 10^6$.

Task Tree	M	$\mathrm{KolmogorovApprox}$	$\mathrm{OptTrim}$	Trim	Sampling	
		m/N=10	m/N=10	$\varepsilon \cdot N$=0.1	s=10^4	s=10^6
Logistics ($N = 34$)	2	0	0	0.0019	0.007	0.0009
	4	0.0024	0.0046	0.0068	0.0057	0.0005
Logistics (N=45)	2	0.0002	0.0005	0.002	0.015	0.001
	4	0	0.003	0.004	0.008	0.0006
DRC-Drive (N=47)	2	0.0014	0.004	0.009	0.0072	0.0009
	4	0.001	0.008	0.019	0.0075	0.0011
Sequential (N=10)	2	0.0093	0.015	0.024	0.0063	0.0008
	4	0.008	0.024	0.04	0.008	0.0016

Table 1: Comparison of estimated errors with respect to the reference exact computation on various task trees.

Table 1 shows the results of the case study experiment. The quality of the solutions provided by using the $\mathrm{KolmogorovApprox}$ operator are better than those provided by the Trim and $\mathrm{OptTrim}$ operators, following the optimality guarantees, but is interesting to see that the quality gaps happen in practice in each of the examined task trees. However, in some of the task trees the sampling method produced better results than the approximation algorithm with $\mathrm{KolmogorovApprox}$. Nevertheless, the approximation algorithm comes with an inherent advantage of providing an exact quality guarantees, as opposed to the probabilistic guarantees provided by sampling.

In order to better understand the quality gaps in practice between $\mathrm{KolmogorovApprox}$, $\mathrm{OptTrim}$, and Trim, we investigate their relative errors when applied on single random variables with support size $n = 100$, and different support sizes of the resulting random variable approximation (m). In each instance of this experiment, a random variable is randomly generated by choosing the probabilities of each element in the support uniformly and then normalizing these probabilities so that they sum to 1.

Figure 1 present the error produced by the above methods. The depicted results are averages over several instances (50 instances) of random variables. The curves in the figure show the average error of $\mathrm{OptTrim}$ and Trim operators with comparison to the average error of the optimal approximation provided by $\mathrm{KolmogorovApprox}$ as a function of m. According to the depicted results it is evident that increasing the support size of the approximation m reduces the error, as expected, in all three methods. However, errors produced by the $\mathrm{KolmogorovApprox}$ are significantly smaller, a half of the error produced by $\mathrm{OptTrim}$ and Trim.

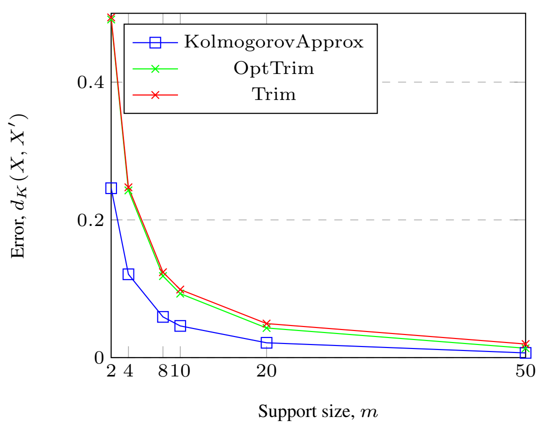

Figure 1: Error comparison between $\mathrm{KolmogorovApprox}$, $\mathrm{OptTrim}$, and Trim, on randomly generated random variables as function of m.

We also examined how our algorithm compares to linear programing as described and discussed, for example, in [15]. We ran an experiment to compare the run-time between the $\mathrm{KolmogorovApprox}$ algorithm with the run-time of a state-of-art implementation of linear programing. We used the "Minimize" function of Wolfram Mathematica and fed it with the equations $\min_{\alpha\in\mathbb{R}^n} \|x - \alpha\|_\infty$ subject to $\|\alpha\|_0 \leq m$ and $\|\alpha\|_1 = 1$. The run-time comparison results were clear and persuasive, for a random variable with support size $n = 10$ and $m = 5$, the LP algorithm run-time was 850 seconds, where the $\mathrm{KolmogorovApprox}$ algorithm run-time was less than a tenth of a second. For $n = 100$ and $m = 5$, the $\mathrm{KolmogorovApprox}$ algorithm run-time was 0.14 seconds and the LP algorithm took more than a day. Due to these timing results of the LP algorithm we did not proceed to examine it any further. Since it is not trivial to formally analyze the run-time of the LP algorithm, we conclude by the reported experiment that in this case the LP algorithm might not be as efficient as $\mathrm{KolmogorovApprox}$ algorithm whose complexity is proven to be polynomial in Theorem 12.

5 Discussion

Compact representations of distributions is mentioned in the literature in various contexts for various applications. In this paper, we are interested in finding optimal approximation of a random variables under the Kolmogorov metric which we define as optimal m-approximation. In order to achieve this optimal approximation two steps were taken, find the support of the optimal random variable and then calculate the pmf of each and every value in that support to minimize the error. Proofs of existences, optimality and run-time were detailed in Section 3 and the main algorithm was presented, the $\mathrm{KolmogorovApprox}$ algorithm. Establishing the main contribution of this paper which is to present an optimal approximation scheme and to show it can be achieved in polynomial run-time. Furthermore, empirical evaluation was conducted on different domains and application to examine the algorithm performance in practice. We were interested in two aspects of performance - accuracy and run-time. Regarding to accuracy, as expected, the suggested $\mathrm{KolmogorovApprox}$ algorithm results much smaller error compared to the other methods, sometimes, in more then factor 0f 2. Regarding to run-time, $\mathrm{KolmogorovApprox}$ algorithm run-time is significantly much faster then LP approach. However, compared to other approximation methods accuracy vs. run-time is a trade off yet to be examined. Another interesting experiment that can be conducted in future work is to add the presented approach as one of the methods examined in [21] and compare it to the binning approaches.

As elaborated in the paper, our algorithm improves on the approach of Cohen, Shimony and Weiss [5] and [4] in that it finds an optimal two sided Kolmogorov approximation, and not just one sided. We consider this paper as a step in the examination of algorithms for optimal approximations of random variables. Beyond the Kolmogorov measure studied here we believe that similar approaches may apply also to total variation, the Wasserstein distance, and to other measures of approximations for other purposes.

References

[1] R. Alford, V. Shivashankar, M. Roberts, J. Frank, and D. W. Aha. Hierarchical planning: Relating task and goal decomposition with task sharing. In *IJCAI*, pages 3022–3029, 2016.

[2] C. Bolton et al. *Logistic regression and its application in credit scoring*. PhD thesis, Citeseer, 2010.

[3] A. Chakravarty, J. Orlin, and U. Rothblum. A partitioning problem with additive objective with an application to optimal inventory groupings for joint replenishment. *Operations Research*, 30(5):1018–1022, 1982.

[4] L. Cohen, T. Grinshpoun, and G. Weiss. Optimal approximation of random variables for estimating the probability of meeting a plan deadline. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018*, 2018.

[5] L. Cohen, S. E. Shimony, and G. Weiss. Estimating the probability of meeting a deadline in hierarchical plans. In *IJCAI*, pages 1551–1557, 2015.

[6] T. Dean, R. J. Firby, and D. Miller. Hierarchical planning involving deadlines, travel time, and resources. *Computational Intelligence*, 4(3):381–398, 1988.

[7] K. Erol, J. Hendler, and D. S. Nau. HTN planning: Complexity and expressivity. In *AAAI*, volume 94, pages 1123–1128, 1994.

[8] K. Erol, J. Hendler, and D. S. Nau. Complexity results for HTN planning. *Annals of Mathematics and Artificial Intelligence*, 18(1):69–93, 1996.

[9] J. D. Gibbons and S. Chakraborti. Nonparametric statistical inference. In *International encyclopedia of statistical science*, pages 977–979. Springer, 2011.

[10] R. Guérin and A. Orda. Computing shortest paths for any number of hops. *IEEE/ACM Transactions on Networking (TON)*, 10(5):613–620, 2002.

[11] E. Mays. *Handbook of credit scoring*. Global Professional Publishi, 2001.

[12] A. C. Miller and T. R. Rice. Discrete approximations of probability distributions. *Management Science*, 29(3):352–362, 1983.

[13] R. Möhring. Scheduling under uncertainty: Bounding the makespan distribution. *Computational Discrete Mathematics*, pages 79–97, 2001.

[14] D. S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu, and F. Yaman. SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*, 20:379–404, 2003.

[15] K. Pavlikov and S. Uryasev. CVaR distance between univariate probability distributions and approximation problems. Technical Report 2015-6, University of Florida, 2016.

[16] A. N. Pettitt and M. A. Stephens. The kolmogorov-smirnov goodness-of-fit statistic with discrete and grouped data. *Technometrics*, 19(2):205–210, 1977.

[17] M. Refaat. *Credit Risk Scorecard: Development and Implementation Using SAS*. Lulu. com, 2011.

[18] E. Shufan, H. Ilani, and T. Grinshpoun. A two-campus transport problem. In *MISTA*, pages 173–184, 2011.

[19] N. Siddiqi. *Credit risk scorecards: developing and implementing intelligent credit scoring*, volume 3. John Wiley & Sons, 2012.

[20] Z. Xiao, A. Herzig, L. Perrussel, H. Wan, and X. Su. Hierarchical task network planning with task insertion and state constraints. In *IJCAI*, pages 4463–4469, 2017.

[21] G. Zeng. A comparison study of computational methods of kolmogorov–smirnov statistic in credit scoring. *Communications in Statistics-Simulation and Computation*, 46(10):7744–7760, 2017.