Non-adaptive Stochastic Score Classification and Explainable Halfspace Evaluation

Rohan Ghuge[1(✉)], Anupam Gupta[2], and Viswanath Nagarajan[1]

[1] University of Michigan, Ann Arbor, MI, USA
rghuge@umich.edu
[2] Carnegie Mellon University, Pittsburgh, PA, USA

Abstract. We consider the stochastic score classification problem. There are several binary tests, where each test i is associated with a probability p_i of being positive, a cost c_i, and a weight a_i. The score of an outcome is a weighted sum of all positive tests, and the range of possible scores is partitioned into intervals corresponding to different classes. The goal is to perform tests sequentially (and possibly adaptively) so as to identify the class at the minimum expected cost. We provide the first constant-factor approximation algorithm for this problem, which improves over the previously-known logarithmic approximation ratio. Moreover, our algorithm is *non adaptive*: it just involves performing tests in a *fixed* order until the class is identified. Our approach also extends to the d-dimensional score classification problem and the "explainable" stochastic halfspace evaluation problem (where we want to evaluate some function on d halfspaces). We obtain an $O(d^2 \log d)$-approximation algorithm for both these extensions. Finally, we perform computational experiments that demonstrate the practical performance of our algorithm for score classification. We observe that, for most instances, the cost of our algorithm is within 50% of an information-theoretic lower bound on the optimal value.

Keywords: Stochastic optimization · Approximation algorithms · Stochastic probing · Adaptivity

1 Introduction

The problem of diagnosing complex systems often involves running a large number of tests for each component of such a system. One option to diagnose such systems is to perform tests on *all* components, which can be prohibitively expensive and slow. Therefore, we are interested in a policy that tests components one by

The full version of the paper is available at [13]. R. Ghuge and V. Nagarajan were supported in part by NSF grants CMMI-1940766 and CCF-2006778. A. Gupta was supported in part by NSF awards CCF-1907820, CCF-1955785, and CCF-2006953.

K. Aardal and L. Sanitá (Eds.): IPCO 2022, LNCS 13265, pp. 277–290, 2022.
https://doi.org/10.1007/978-3-031-06901-7_21

one, and minimizes the average cost of testing. (See [23] for a survey.) Concretely, we consider a setting where the goal is to test various components, in order to assign a risk class to the system (e.g., whether the system has low/medium/high risk).

The *stochastic score classification* (`SSClass`) problem introduced by [14] models such situations. There are n components in a system, where each component i is "working" with independent probability p_i. While the probabilities p_i are known *a priori*, the random outcomes $X_i \in \{0, 1\}$ are initially unknown. The outcome X_i of each component i can be determined by performing a test of cost c_i: $X_i = 1$ if i is working and $X_i = 0$ otherwise. The overall status of the system is determined by a linear score $r(X) := \sum_{i=1}^{n} a_i X_i$, where the coefficients $a_i \in \mathbb{Z}$ are input parameters. We are also given a collection of intervals $I_1, I_2, \ldots, I_k$ that partition the real line (i.e., all possible scores). The goal is to determine the interval I_j (also called the *class*) that contains $r(X)$, while incurring minimum expected cost. A well-studied special case is when there are just two classes, which corresponds to evaluating a halfspace or linear-threshold-function [11].

Example: Consider a system which must be assigned a risk class of *low*, *medium*, or *high*. Suppose there are five components in the system, each of which is working with probability $\frac{1}{2}$. The score for the entire system is the number of working components. A score of 5 corresponds to the "Low" risk class, scores between 2 and 4 correspond to "Medium" risk, and a score of at most 1 signifies "High" risk. Suppose that after testing components $\{1, 2, 3\}$, the system has score 2 (which occurs with probability $\frac{3}{8}$) then it will be classified as medium risk irrespective of the remaining two components: so testing can be stopped. Instead, if the system has score 3 after testing components $\{1, 2, 3\}$ (which occurs with probability $\frac{1}{8}$) then the class of the system cannot be determined with certainty (it may be either medium or low), and so further testing is needed.

A related problem is the *d-dimensional stochastic score classification problem* (d-`SSClass`), which models the situation when a system has d different functions, each with an associated linear score (as above). We must now perform tests on the underlying components to *simultaneously* assign a class to each of the d functions.

In another related problem, a system again has d different functions. Here, the status (working or failed) of each function is determined by some halfspace, and the overall system is considered operational if all d functions are working. The goal of a diagnosing policy is to decide whether the system is operational, and if not, to return at least one function that has failed (and therefore needs maintenance). This is a special case of a problem we call *explainable stochastic halfspace evaluation* (`EX-SFE`).

Solutions for all these problems (`SSClass`, d-`SSClass`, and `EX-SFE`) are *sequential decision processes*. At each step, a component is tested and its outcome (working or failed) is observed. The information from all previously tested components can then be used to decide on the next component to test; this makes the process *adaptive*. This process continues until the risk class can be determined with certainty from the tested components. One simple class of solutions are *non-adaptive* solutions, which are simply described by a priority list:

we then test components in this fixed order until the class can be uniquely determined. Such solutions are simpler and faster to implement, compared to their adaptive counterparts: the non-adaptive testing sequence needs to be constructed just once, after which it can be used for all input realizations. However, non-adaptive solutions are weaker than adaptive ones, and our goal is to bound the *adaptivity gap*, the multiplicative ratio between the performance of the non-adaptive solution to that of the optimal adaptive one. Our main result shows that `SSClass` has a constant adaptivity gap, thereby answering an open question posed by [14]. Additionally, we show an adaptivity gap of $O(d^2 \log d)$ for both the d-`SSClass` and `EX-SFE` problems.

Before we present the results and techniques, we formally define the problem. For any integer m, we use $[m] := \{1, 2, \ldots, m\}$. An instance of `SSClass` consists of n independent $\{0,1\}$ random variables $X = X_1, \ldots, X_n$, where variable X_i has $\Pr[X_i = 1] = \mathbb{E}[X_i] = p_i$. The cost to probe/query X_i is $c_i \in \mathbb{R}_+$; both p_i, c_i are known to us. We are also given non-negative weights $a_i \in \mathbb{Z}_+$, and the *score* of the outcome $X = (X_1, \ldots, X_n)$ is $r(X) = \sum_{i=1}^{n} a_i X_i$. In addition, we are given $B + 1$ integers $\alpha_1, \ldots, \alpha_{B+1}$ such that *class* j corresponds to the interval $I_j := \{\alpha_j, \ldots, \alpha_{j+1} - 1\}$. The *score classification function* $h : \{0,1\}^n \to \{1, \ldots, B\}$ assigns $h(X) = j$ precisely when $r(X) \in I_j$. The goal is to determine $h(X)$ at minimum expected cost. We assume non-negative weights only for simplicity: any instance with positive and negative weights can be reduced to an equivalent instance with all positive weights (see full version for details). Let $W := \sum_{i=1}^{n} a_i$ denote the total weight. In our algorithm, we associate two numbers $(\beta_j^0, \beta_j^1) \in \mathbb{Z}_+^2$ with each class j, where $\beta_j^0 = W - \alpha_{j+1} + 1$ and $\beta_j^1 = \alpha_j$.

1.1 Results and Techniques

Our main result is the following algorithm (and adaptivity gap).

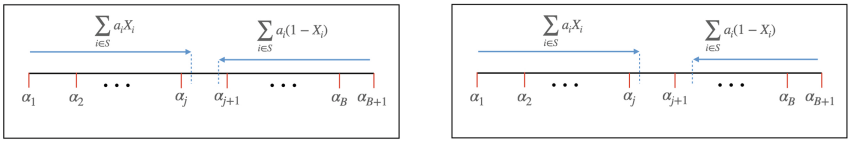
Theorem 1. *There is a polynomial-time non-adaptive algorithm (called* NaCl*) for stochastic score classification with expected cost at most a constant factor times that of the optimal adaptive policy.*

This result improves on the prior work from [11,14] in several ways. Firstly, we get a constant-factor approximation, improving upon the previous $O(\log W)$ and $O(B)$ ratios, where W is the sum of weights, and B the number of classes. Secondly, our algorithm is non-adaptive in contrast to the previous adaptive ones. Finally, our algorithm has nearly-linear runtime, which is faster than the previous algorithms.

An added benefit of our approach is that we obtain a "universal" solution that is simultaneously $O(1)$-approximate for all class-partitions. Indeed, the non-adaptive list produced by NaCl only depends on the probabilities, costs, and weights, and not on the class boundaries $\{\alpha_j\}$; these α_j values are only needed in the stopping condition for probing.

Algorithm Overview. To motivate our algorithm, suppose that we have probed a subset $S \subseteq [n]$ of variables, and there is a class j such that $\sum_{i \in S} a_i X_i \geq \alpha_j$ and $\sum_{i \in S} a_i (1 - X_i) \geq W - \alpha_{j+1} + 1$. The latter condition can be rewritten

as $\sum_{i\in S} a_i X_i + \sum_{i\notin S} a_i \le \alpha_{j+1} - 1$. So, we can conclude that the final score lies in $\{\alpha_j, \ldots, \alpha_{j+1} - 1\}$ irrespective of the outcomes of variables in $[n] \setminus S$. This means that $h(X) = j$. On the other hand, if the above condition is not satisfied for *any* class j, we must continue probing. Towards this end, we define two types of rewards for each variable $i \in [n]$: $R_0(i) = a_i \cdot (1 - X_i)$ and $R_1(i) = a_i \cdot X_i$. (See Fig. 1.) The total R_0-reward and R_1-reward from the probed variables correspond to upper and lower bounds on the score, respectively. Our non-adaptive algorithm NaCl probes the variables in a predetermined order until $\sum_{i\in S} R_0(i) \ge \beta_j^0 = W - \alpha_{j+1} + 1$ and $\sum_{i\in S} R_1(i) \ge \beta_j^1 = \alpha_j$ for *some* class j (at which point it determines $h(X) = j$).



(a) The R_0 reward (upper bound) and R_1 reward (lower bound) lie in class j and hence probing can be stopped.

(b) Here $f(X)$ could be j or $j+1$, so probing must continue.

Fig. 1. Illustration of non-adaptive approach

To get this ordering we first build two separate lists: list L_b for the R_b-rewards (for $b = 0, 1$) minimizes the cost required to cover some target amount of R_b-reward. Finally, interleaving lists L_0 and L_1 gives the final list. The idea behind list L_0 is as follows: if we only care about a *single* class j, we can set a target of β_j^0 and use the non-adaptive algorithm for stochastic knapsack cover [21]. Since the class j is unknown, so is the target β_j^0 on the R_0-reward. Interestingly, we show how to construct a "universal" non-adaptive list L_0 that works for *all* targets simultaneously. The construction proceeds in phases: in each phase $\ell \ge 0$, the algorithm adds a subset of variables with cost $O(2^\ell)$ that (roughly) maximizes the *expected* R_0 reward. Naïvely using the expected rewards can lead to poor performance, so a natural idea is to use rewards *truncated* at logarithmically-many scales (corresponding to the residual target); see for example, [12]. Moreover, to get a constant-factor approximation, we use the *critical scaling* idea from [21]. Roughly speaking, this identifies a *single* scale κ such that with constant probability (1) the algorithm obtains large reward (truncated at scale κ), and (2) any subset of cost 2^ℓ has small reward.

Analysis Overview. The analysis of Theorem 1 relates the "non-completion" probabilities of our algorithm after cost $\gamma \cdot 2^\ell$ to that of the optimal adaptive algorithm after cost 2^ℓ, for each phase $\ell \ge 0$. The factor γ corresponds to the approximation ratio of the algorithm. In order to relate these non-completion probabilities, we consider the R_0 and R_1-rewards obtained by an optimal adaptive algorithm, and argue that the non-adaptive algorithm obtains a higher R_0 *as*

well as R_1 reward (with constant probability). Thus, if the optimal adaptive algorithm decides $h(X)$, so does the non-adaptive algorithm. Our algorithm/analysis also use various properties of the fractional knapsack problem.

Extensions. Next, we consider the d-dimensional score classification problem (d-`SSClass`) and obtain the following result:

Theorem 2. *There is a non-adaptive $O(d^2 \log d)$-approximation algorithm for d-dimensional stochastic score classification.*

We achieve this by extending the above approach (for $d = 1$). We now define two rewards (corresponding to R_0 and R_1) for *each* dimension d. Then, we apply the list building algorithm for each of these rewards, resulting in $2d$ separate lists. Finally, we interleave these lists to obtain the non-adaptive probing sequence. The analysis is also an extension of the $d = 1$ case. The main differences are as follows. Just accounting for the $2d$ lists results in an extra $O(d)$ factor in the approximation. Furthermore, we need to ensure (for each phase ℓ) that with constant probability, our non-adaptive algorithm achieves more reward than the optimum for *all* the $2d$ rewards. We incur another $O(d \log d)$ factor in the approximation in order to achieve this stronger property.

In a similar vein, our main result for `EX-SFE` is the following:

Theorem 3. *There is a non-adaptive $O(d^2 \log d)$-approximation algorithm for explainable stochastic halfspace evaluation.*

The non-adaptive list for `EX-SFE` is constructed in the same manner as for d-`SSClass`, but the stopping rule relies on the oracle for verifying witnesses of $f \circ h$. As a special case, we obtain:

Corollary 1. *There is a non-adaptive $O(d^2 \log d)$-approximation algorithm for the explainable stochastic intersection of half-spaces problem.*

The stochastic intersection of halfspaces problem (in a slightly different model) was studied previously by [6], where an $O(\sqrt{n \log d})$-approximation algorithm was obtained assuming all probabilities $p_i = \frac{1}{2}$. The main difference from our model is that [6] do not require a witness at the end. So their policy can stop if it concludes that there exists a violated halfspace (even without knowing which one), whereas our policy can only stop after it identifies a violated halfspace (or determines that all halfspaces are satisfied). We note that our approximation ratio is independent of the number of variables n and holds for arbitrary probabilities. The proofs of Theorems 2 and 3 are omitted in this extended abstract and appear in the full version of the paper.

Computational Results. Finally, we evaluate the empirical performance of our algorithm for score classification. In these experiments, our non-adaptive algorithm performs nearly as well as the previous-best *adaptive* algorithms, while being an order of magnitude faster. In fact, on many instances, our algorithm provides an *improvement* in both the cost as well as the running time. On most instances, the cost of our algorithm is within 50% of an information-theoretic lower bound on the optimal value.

1.2 Related Work

The special case of SSClass with $B = 2$ classes is the well-studied stochastic Boolean function evaluation for *linear threshold functions* (SBFT). Here, the goal is to identify whether a single halfspace is satisfied (i.e., the score is above or below a threshold). [11] gave an elegant 3-approximation algorithm for SBFT using an adaptive dual greedy approach. Prior to their work, only an $O(\log W)$-approximation was known, based on the more general stochastic submodular cover problem [15,20].

The general SSClass problem was introduced by [14], who showed that it can be formulated as an instance of stochastic submodular cover. Then, using general results such as [15,20], they obtained an adaptive $O(\log W)$-approximation algorithm. Furthermore, [14] obtained an adaptive $3(B-1)$-approximation algorithm for SSClass by extending the approach of [11] for SBFT; recall that B is the number of classes. A main open question from this work was the possibility of a constant approximation for the general SSClass problem. We answer this in the affirmative. Moreover, our algorithm is non-adaptive: so we also bound the adaptivity gap.

The stochastic knapsack cover problem (SKC) is closely related to SBFT. Given a set of items with random rewards and a target k, the goal is to (adaptively) select a subset of items having total reward at least k. The objective is to minimize the expected cost of selected items. [11] gave an adaptive 3-approximation algorithm for SKC. Later, [21] gave a *non-adaptive* $O(1)$-approximation algorithm for SKC. In fact, the result in [21] applied to the more general stochastic k-TSP problem [12]. Our algorithm and analysis use some ideas from [12,21]. We use the notion of a "critical scale" from [21] to identify the correct reward truncation threshold. The approach of using non-completion probabilities in the analysis is similar to [12]. There are also a number of differences: we exploit additional structure in the (fractional) knapsack problem and obtain a simpler and nearly-linear time algorithm.

More generally, non-adaptive solutions (and adaptivity gaps) have been used in solving various other stochastic optimization problems such as max-knapsack [5,10], matching [2,4], probing [18,19] and orienteering [3,16,17]. Our result shows that this approach is also useful for SSClass.

SSClass and EX-SFE also fall under the umbrella of designing query strategies for "priced information", where one wants to evaluate a function by sequentially querying variables (that have costs). There are two lines of work here: comparing to an optimal strategy (as in our model) [1,6,14,22], and comparing to the min-cost solution in hindsight (i.e., competitive analysis) [7–9]. We note that the "explainable" requirement in the EX-SFE problem (that we solve) is similar to the requirement in [22].

2 Preliminaries

We first state some basic results for the deterministic knapsack problem. In an instance of the knapsack problem, we are given a set T of items with non-negative

costs $\{c_i : i \in T\}$ and rewards $\{r_i : i \in T\}$, and a budget D on the total cost. The goal is to select a subset of items of total cost at most D that maximizes the total reward. The LP relaxation is the following:

$$g(D) = \max \left\{ \sum_{i \in T} r_i \cdot x_i \;\middle|\; \sum_{i \in T} c_i \cdot x_i \le D,\, x \in [0,1]^T \right\}, \qquad \forall D \ge 0.$$

The following algorithm $\mathcal{A}_{\mathtt{KS}}$ solves the *fractional* knapsack problem and also obtains an approximate integral solution. Assume that the items are ordered so that $\frac{r_1}{c_1} \ge \frac{r_2}{c_2} \ge \cdots$. Let t index the first item (if any) so that $\sum_{i=1}^{t} c_i \ge D$. Let $\psi := \frac{1}{c_t}(D - \sum_{i=1}^{t-1} c_i)$ which lies in $(0, 1]$. Define

$$x_i = \begin{cases} 1 & \text{if } i \le t-1 \\ \psi & \text{if } i = t \\ 0 & \text{if } i \ge t+1 \end{cases}.$$

Return x as the optimal fractional solution and $Q = \{1, \cdots, t\}$ as an integer solution. We use the following well-known result (proved in the full version, for completeness).

Theorem 4. *Consider algorithm $\mathcal{A}_{\mathtt{KS}}$ on any instance of the knapsack problem with budget D.*

1. *$\langle r, x \rangle = \sum_{i=1}^{t-1} r_i + \psi \cdot r_t = g(D)$ and so x is an optimal LP solution.*
2. *The derivative $g'(D) = \frac{r_t}{c_t}$.*
3. *Solution Q has cost $c(Q) \le D + c_{max}$ and reward $r(Q) \ge g(D)$.*
4. *$g(D)$ is a concave function of D.*

3 The Stochastic Score Classification Algorithm

Our non-adaptive algorithm creates two lists L_0 and L_1 separately. These lists are based on the R_0 and R_1 rewards of the variables, where $R_0(i) = a_i(1 - X_i)$ and $R_1(i) = a_i X_i$. It interleaves lists L_0 and L_1 together (by power-of-2 costs) and then probes the variables in this non-adaptive order until the class is identified.

3.1 The Algorithm

We first explain how to build the lists L_0 and L_1. We only consider list L_0 below (the algorithm/analysis for L_1 are identical). The list building algorithm operates in phases. For each phase $\ell \ge 0$ it gets a budget of $O(2^\ell)$, and it solves several instances of the deterministic knapsack problem, where rewards are truncated expectations of R_0. We will use the following truncation values, also called *scales*.

$$\mathcal{G} := \left\{ \theta^\ell \;:\; 0 \le \ell \le 1 + \log_\theta W \right\}, \text{ where } \theta > 1 \text{ is a constant.}$$

For each scale $\tau \in \mathcal{G}$, we find a deterministic knapsack solution with reward $\mathbb{E}[\min\{R_0/\tau, 1\}]$ (see Equation 1 for the formal definition) and budget $\approx C2^\ell$

(where $C > 1$ is a constant). Including solutions for each scale would lead to an $O(\log W)$ loss in the approximation factor. Instead, as in [21], we identify a "critical scale" and only include solutions based on the critical scale. To this end, each scale τ is classified as either *rich* or *poor*. Roughly, in a rich scale, the knapsack solution after budget $C2^\ell$ still has large "incremental" reward (formalized by the derivative of g being at least some constant δ). The *critical scale* is the smallest scale κ that is poor, and so represents a transition from rich to poor. For our analysis, we will choose constant parameters C, δ and θ so that $\frac{C\delta}{\theta} > 1$. We note however, that our algorithm achieves a constant approximation ratio for any constant values $C > 1$, $\delta \in (0, 1)$ and $\theta > 0$.

Algorithm 1. PickReps$(\ell, \tau, \mathbf{r})$

1: let $T \subseteq [n]$ denote the variables with non-zero reward and cost at most 2^ℓ
2: run algorithm $\mathcal{A}_{\text{KS}}$ (Theorem 4) on the knapsack instance with items T and budget $D = C2^\ell$
3: let $f = g'(D)$ be the derivative of the LP value and $Q \subseteq T$ the integral solution from $\mathcal{A}_{\text{KS}}$
4: **if** $f > \delta 2^{-\ell}$ **then**
5: scale τ is **rich**
6: **else**
7: scale τ is **poor**
8: **return** Q

Subroutine PickReps (Algorithm 1) computes the knapsack solution for each scale τ, and classifies the scale as rich/poor. The subroutine BuildList(R) (Algorithm 2) builds the list for any set of random rewards $\{R(i) : i \in [n]\}$. List L_b (for $b = 0, 1$) is obtained by running BuildList(R_b). Finally, the non-adaptive algorithm NaCl involves interleaving the variables in lists L_0 and L_1; this is described in Algorithm 3. The resulting policy probes variables in the order given by NaCl until the observed upper and lower bounds on the score lie within the same class. Note that there are $O(\log(nc_{max}))$ phases and $O(\log W)$ scales: so the total number of deterministic knapsack instances solved is poly-logarithmic. Moreover, the knapsack algorithm $\mathcal{A}_{\text{KS}}$ runs in $O(n \log n)$ time. So the overall runtime of our algorithm is nearly linear.

3.2 The Analysis

Lemma 1. *The critical scale κ in Step 6 of Algorithm 2 is always well defined.*

Proof. To prove that there is a smallest poor scale, it suffices to show that not all scales can be rich. We claim that the last scale $\tau \geq W$ cannot be rich. Suppose (for a contradiction) that scale τ is rich. Then, by concavity of g (see property 4 in Theorem 4), we have $g(D) \geq D \cdot g'(D) > D \cdot \delta 2^{-\ell} = C\delta \geq 1$. On the other hand, the total deterministic reward at this scale, $\sum_{i=1}^{n} r_i^\tau \leq \frac{W}{\tau} \leq 1$. Thus, $g(D) \leq 1$, a contradiction.

Algorithm 2. BUILDLIST($\{R(i) : i \in [n]\}$)

1: list $\Pi \leftarrow \emptyset$
2: **for** phase $\ell = 0, 1, \ldots$ **do**
3: **for** each scale $\tau \in \mathcal{G}$ **do**
4: define truncated rewards

$$r_i^\tau = \begin{cases} \mathbb{E}\left[\min\left\{\frac{R(i)}{\tau}, 1\right\}\right], & \text{if } i \notin \Pi \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

5: $S_{\ell,\tau} \leftarrow$ PICKREPS$(\ell, \tau, \mathbf{r}^\tau)$
6: let κ be the smallest **poor** scale in $\mathcal{G}$ (this is called the critical scale)
7: $\Pi_\ell \leftarrow S_{\ell,\kappa}$ and $\Pi \leftarrow \Pi \circ \Pi_\ell$
8: **return** list Π

Algorithm 3. NACL (NON-ADAPTIVE CLASSIFIER)

1: list $L_b \leftarrow$ BUILDLIST$(\{R_b(i) : i \in [n]\})$ for $b = 0, 1$
2: let L_b^ℓ denote the variables in phase ℓ for list L_b
3: for each phase ℓ, set $S_\ell \leftarrow L_0^\ell \cup L_1^\ell$
4: **return** list $S_0, S_1, \cdots, S_\ell \cdots$

Lemma 2. *The cost $c(S_{\ell,\tau}) \le (C+1)2^\ell$ for any phase ℓ. Hence, the cost incurred in phase ℓ of* NACL *is at most $(C+1)2^{\ell+1}$.*

Proof. Consider any call to PICKREPS in phase ℓ. We have $S_{\ell,\tau} = Q$ where Q is the integer solution from Theorem 4. It follows that $c(S_{\ell,\tau}) = c(Q) \le C2^\ell + \max_{i \in T} c_i \le (C+1)2^\ell$; note that we only consider variables of cost at most 2^ℓ (see Step 1 of Algorithm 1). Finally, the variables S_ℓ in phase ℓ of NACL consist of the phase-ℓ variables of both L_0 and L_1. So the total cost of these variables is at most $(C+1)2^{\ell+1}$. □

We now analyze the cost incurred by our non-adaptive strategy NACL. We denote by `OPT` an optimal adaptive solution for `SSClass`. To analyze the algorithm, we use the following notation.

- u_ℓ: probability that NACL is not complete by end of phase ℓ.
- u_ℓ^*: probability that `OPT` costs at least 2^ℓ.

We can assume by scaling that the minimum cost is 1. So $u_0^* = 1$. For ease of notation, we use `OPT` and `NA` to denote the *random* cost incurred by `OPT` and NACL respectively. We also divide `OPT` into phases: phase ℓ corresponds to variables in `OPT` after which the cumulative cost is between $2^{\ell-1}$ and 2^ℓ. The following lemma forms the crux of the analysis.

Lemma 3. *For any phase $\ell \ge 1$, we have $u_\ell \le q \cdot u_{\ell-1} + u_\ell^*$ where $q \le 0.3$.*

Given Lemma 3, the proof of Theorem 1 is standard (see, for example, [12]).

3.3 Proof of Lemma 3

Recall that NaCl denotes the non-adaptive algorithm, and NA its random cost. Fix any phase $\ell \geq 1$, and let σ denote the realization of the variables probed in the first $\ell - 1$ phases of NaCl. We further define the following *conditioned on* σ:

- $u_\ell(\sigma)$: probability that NaCl is not complete by end of phase ℓ.
- $u^*_\ell(\sigma)$: probability that OPT costs at least 2^ℓ, i.e., OPT is not complete by end of phase ℓ.

If NaCl does not complete before phase ℓ then $u_{\ell-1}(\sigma) = 1$, and we will prove

$$u_\ell(\sigma) \leq u^*_\ell(\sigma) + 0.3. \tag{2}$$

We can complete the proof using this. Note that $u_{\ell-1}(\sigma)$ is either 0 or 1. If $u_{\ell-1}(\sigma) = 0$ then $u_\ell(\sigma) = 0$ as well. So, Eq. (2) implies that $u_\ell(\sigma) \leq u^*_\ell(\sigma) + 0.3u_{\ell-1}(\sigma)$ for all σ. Taking expectation over σ gives Lemma 3. It remains to prove Eq. (2).

We denote by $\mathcal{R}_0$ and $\mathcal{R}^*_0$ the total R_0 reward obtained in the first ℓ phases by NaCl and OPT respectively. We similarly define $\mathcal{R}_1$ and $\mathcal{R}^*_1$. To prove Equation (2), we will show that the probabilities (conditioned on σ) $\mathbf{P}(\mathcal{R}^*_0 > \mathcal{R}_0)$ and $\mathbf{P}(\mathcal{R}^*_1 > \mathcal{R}_1)$ are small. Intuitively, this implies that with high probability, if OPT finishes in phase ℓ, then so does NaCl. Formally, we prove the following key lemma.

Lemma 4 (Key Lemma). *For $b \in \{0, 1\}$, we have $\mathbf{P}(\mathcal{R}_b < \mathcal{R}^*_b \mid \sigma) \leq 0.15$.*

Using these lemmas, we prove Equation (2).

Proof (Proof of Equation (2)). Recall that we associate a pair (β^0_j, β^1_j) with every class j. If OPT finishes in phase ℓ, then there exists some j such that $\mathcal{R}^*_0 \geq \beta^0_j$ and $\mathcal{R}^*_1 \geq \beta^1_j$. Thus,

$$\mathbf{P}(\texttt{OPT finishes in phase } \ell \mid \sigma) = 1 - u^*_\ell(\sigma) = \mathbf{P}(\exists j : \mathcal{R}^*_0 \geq \beta^0_j \text{ and } \mathcal{R}^*_1 \geq \beta^1_j \mid \sigma).$$

From Lemma 4 and union bound, we have $\mathbf{P}(\mathcal{R}_0 < \mathcal{R}^*_0 \text{ or } \mathcal{R}_1 < \mathcal{R}^*_1 \mid \sigma) \leq 0.3$. Then, we have

$$\begin{aligned}
1 - u_\ell(\sigma) &= \mathbf{P}(\texttt{NA finishes in phase } \ell \mid \sigma) \\
&\geq \mathbf{P}\left((\texttt{OPT finishes in phase } \ell) \bigwedge \mathcal{R}_0 \geq \mathcal{R}^*_0 \bigwedge \mathcal{R}_1 \geq \mathcal{R}^*_1 \,\middle|\, \sigma\right) \\
&\geq \mathbf{P}(\texttt{OPT finishes in phase } \ell \mid \sigma) - \mathbf{P}(\mathcal{R}_0 < \mathcal{R}^*_0 \text{ or } \mathcal{R}_1 < \mathcal{R}^*_1 \mid \sigma) \\
&\geq (1 - u^*_\ell(\sigma)) - 0.3
\end{aligned}$$

Upon rearranging, this gives $u_\ell(\sigma) \leq u^*_\ell(\sigma) + 0.3$ as desired. □

Proof Sketch of the Key Lemma. We now provide an intuition for the proof of Lemma 4 with $b = 0$ (the case $b = 1$ is identical). Henceforth, reward will only refer to R_0. Observe that in phase ℓ of Algorithm 2, the previously probed variables $\Pi \subseteq \sigma$. For ease of notation, let σ also represent the set of variables probed in the first $\ell - 1$ phases. Recall that S_ℓ is the set of variables probed by NaCl in phase ℓ. Let O_ℓ be the variables probed by `OPT` in phase ℓ; so the total cost of O_ℓ is at most 2^ℓ. Note that O_ℓ can be a random subset as `OPT` is adaptive. Also, S_ℓ is a deterministic subset as NaCl is a non-adaptive list. Roughly, we show that (conditioned on σ) the probability that O_ℓ has more reward than S_ℓ is small. The key idea is to use the *critical scale* κ (in phase ℓ) to argue that the following hold with constant probability (1) reward of $O_\ell \setminus (S_\ell \cup \sigma)$ is at most κ, and (2) reward of $S_\ell \setminus O_\ell$ is at least κ. This would imply that with constant probability, NaCl gets at least as much reward as `OPT` by the end of phase ℓ. Formally, we show:

Lemma 5. *If $\mathcal{A}$ is any* adaptive *policy of selecting variables from $[n] \setminus (S_\ell \cup \sigma)$ with total cost $\leq 2^\ell$ then $\mathbf{P}[R_0(\mathcal{A}) < \kappa] \geq 1 - \delta$. Hence,*

$$\mathbf{P}[R_0(O_\ell \setminus (S_\ell \cup \sigma)) \geq \kappa] \leq \delta.$$

Lemma 6. *We have $\mathbf{P}(R_0(S_\ell \setminus O_\ell) < \kappa) \leq e^{-(\mu - \ln \mu - 1)}$. Here, $\mu := (C-1)\delta/\theta$.*

Combining these two lemmas with an appropriate choice of constants C, δ and θ, we obtain Lemma 4. We defer the proofs to the full version.

4 Computational Results

We provide a summary of computational results of our non-adaptive algorithm for the stochastic score classification problem. We conducted all of our computational experiments using Python 3.8 with a 2.3 GHz Intel Core i5 processor and 16 GB 2133MHz LPDDR3 memory. We use synthetic data to generate instances of `SSClass` for our experiments.

Instance Generation. We test our algorithm on synthetic data generated as follows. We first set $n \in \{100, 200, \ldots, 1000\}$. Given n, we generate n Bernoulli variables, each with probability chosen uniformly from $(0, 1)$. We set the costs of each variable to be an integer in $[10, 100]$. To select cutoffs (when $B \neq 2$), we first select $B \in \{5, 10, 15\}$ and then select the cutoffs (based on the value of B) uniformly at random in the score interval. We provide more details and plots in the full version of the paper. For each n we generate 10 instances. For each instance, we sample 50 realizations in order to calculate the average cost and average runtime.

Algorithms. We compare our non-adaptive `SSClass` algorithm (Theorem 1) against a number of prior algorithms. For `SBFT` instances, we compare to the *adaptive* 3-approximation algorithm from [11]. For *unweighted* `SSClass`

instances, we compare to the non-adaptive $2(B-1)$-approximation algorithm from [14]. For general SSClass instances, we compare to the *adaptive* $O(\log W)$-approximation algorithm from [14]. As a benchmark, we also compare to a naive non-adaptive algorithm that probes variables in a random order. We also compare to an information-theoretic lower bound (no adaptive policy can do better than this lower bound). We obtain this lower bound by using an integer linear program to compute the (offline) optimal probing cost for a given realization (see full version for details), and then taking an average over 50 realizations.

Parameters C, δ, and θ. As noted in Sect. 3, our algorithm achieves a constant factor approximation guarantee for any constant $C > 1$, $\delta \in (0,1)$ and $\theta > 1$. For our final computations, we (arbitrarily) choose values $C = 2$, $\delta = 0.01$, and $\theta = 2$.

Reported Quantities. For every instance, we compute the cost and runtime of each algorithm by taking an average over 50 independent realizations. For the non-adaptive algorithms, note that we only need *one* probing sequence for each instance. On the other hand, adaptive algorithms need to find the probing sequence afresh for each realization. Thus, the non-adaptive algorithms are significantly faster (see full version for corresponding runtime plots).

In Table 1, we report the average performance ratio (cost of the algorithm divided by the information-theoretic lower bound) of the various algorithms. For each instance type (SBFT, Unweighted SSClass and SSClass), we report the performance ratio averaged over *all* values of n (10 choices) and all instances (10 each). Note that values closer to 1 demonstrate better performance.

Table 1. Average performance ratios relative to the lower bound.

Instance type	Our Alg.	GGHK Alg.	Random list
Unweighted SSClass, $B = 5$	1.50	1.48	1.80
Unweighted SSClass, $B = 10$	1.25	1.24	1.33
Unweighted SSClass, $B = 15$	1.13	1.13	1.19
SSClass, $B = 5$	1.59	1.94	2.43
SSClass, $B = 10$	1.34	1.45	1.73
SSClass, $B = 15$	1.22	1.39	1.47

Instance type	Our Alg.	DHK Alg.	Random list
SBFT	2.18	1.74	5.63

We observe that for unweighted SSClass instances, our algorithm performs nearly as well as the $2(B-1)$-approximation algorithm. For general (weighted) SSClass instances, our algorithm performs considerably better than the adaptive $O(\log W)$-approximation algorithm. For SBFT instances, the performance of our algorithm is about 25% worse than the adaptive 3-approximation algorithm while its runtime is an order of magnitude faster.

References

1. Allen, S., Hellerstein, L., Kletenik, D., Ünlüyurt, T.: Evaluation of monotone DNF formulas. Algorithmica **77** (2015)
2. Bansal, N., Gupta, A., Li, J., Mestre, J., Nagarajan, V., Rudra, A.: When LP is the cure for your matching woes: improved bounds for stochastic matchings. Algorithmica **63**(4), 733–762 (2012). https://doi.org/10.1007/s00453-011-9511-8
3. Bansal, N., Nagarajan, V.: On the adaptivity gap of stochastic orienteering. Math. Program. **154**(1–2), 145–172 (2015). https://doi.org/10.1007/s10107-015-0927-9
4. Behnezhad, S., Derakhshan, M., Hajiaghayi, M.: Stochastic matching with few queries: (1-ϵ) approximation. In: Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, pp. 1111–1124 (2020)
5. Bhalgat, A., Goel, A., Khanna, S.: Improved approximation results for stochastic knapsack problems. In: SODA, pp. 1647–1665 (2011)
6. Blanc, G., Lange, J., Tan, L.Y.: Query strategies for priced information, revisited. In: Proceedings of the Thirty-Second Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2021), pp. 1638–1650 (2021)
7. Charikar, M., Fagin, R., Guruswami, V., Kleinberg, J., Raghavan, P., Sahai, A.: Query strategies for priced information (extended abstract), pp. 582–591 (2000)
8. Cicalese, F., Laber, E.S.: A new strategy for querying priced information. In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, STOC 2005, pp. 674–683 (2005)
9. Cicalese, F., Laber, E.S.: On the competitive ratio of evaluating priced functions. J. ACM, **58**(3) (2011)
10. Dean, B.C., Goemans, M.X., Vondrák, J.: Approximating the stochastic knapsack problem: the benefit of adaptivity. Math. Oper. Res. **33**(4), 945–964 (2008)
11. Deshpande, A., Hellerstein, L., Kletenik, D.: Approximation algorithms for stochastic submodular set cover with applications to Boolean function evaluation and min-knapsack. ACM Trans. Algorithms **12**(3), 1–28 (2016)
12. Ene, A., Nagarajan, V., Saket, R.: Approximation algorithms for stochastic k-TSP. In: 37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, pp. 27:14–27:27 (2017)
13. Ghuge, R., Gupta, A., Nagarajan, V.: Non-adaptive stochastic score classification and explainable halfspace evaluation. CoRR abs/2111.05687 (2021)
14. Gkenosis, D., Grammel, N., Hellerstein, L., Kletenik, D.: The stochastic score classification problem. In: 26th Annual European Symposium on Algorithms (ESA), vol. 112, pp. 36:1–36:14 (2018)
15. Golovin, D., Krause, A.: Adaptive submodularity: a new approach to active learning and stochastic optimization. CoRR abs/1003.3967 (2017)
16. Guha, S., Munagala, K.: Multi-armed bandits with metric switching costs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5556, pp. 496–507. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-02930-1_41
17. Gupta, A., Krishnaswamy, R., Nagarajan, V., Ravi, R.: Running errands in time: approximation algorithms for stochastic orienteering. Math. Oper. Res. **40**(1), 56–79 (2015)
18. Gupta, A., Nagarajan, V.: A stochastic probing problem with applications. In: Goemans, M., Correa, J. (eds.) IPCO 2013. LNCS, vol. 7801, pp. 205–216. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36694-9_18

19. Gupta, A., Nagarajan, V., Singla, S.: Adaptivity gaps for stochastic probing: submodular and XOS functions. In: Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1688–1702 (2017)
20. Im, S., Nagarajan, V., van der Zwaan, R.: Minimum latency submodular cover. ACM Trans. Algorithms **13**(1), 13:1–13:28 (2016)
21. Jiang, H., Li, J., Liu, D., Singla, S.: Algorithms and adaptivity gaps for stochastic k-TSP. In: 11th Innovations in Theoretical Computer Science Conference (ITCS), vol. 151, pp. 45:1–45:25 (2020)
22. Kaplan, H., Kushilevitz, E., Mansour, Y.: Learning with attribute costs. In: Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing, STOC 2005, pp. 356–365 (2005)
23. Ünlüyurt, T.: Sequential testing of complex systems: a review. Discrete Appl. Math. **142**(1), 189–205 (2004)