Journal of Machine Learning Research 83:1–19, 2018 Submitted 02/2018; Published 04/2018

Robust Inference for Multiclass Classification

Uriel Feige uriel.feige@weizmann.ac.il
Weizmann Institute of Science, Israel

Yishay Mansour mansour.yishay@gmail.com
Tel Aviv University, Israel

Robert E. Schapire schapire@microsoft.com
Microsoft Research, New York

Editor: Mehryar Mohri and Karthik Sridharan

Abstract

We consider the problem of robust inference in which inputs may be maliciously corrupted by a powerful adversary, and the learner's goal is to accurately predict the original, uncorrupted input's true label given only the adversarially corrupted version of the input. We specifically focus on the multiclass version of this problem in which more than two labels are possible. We substantially extend and generalize previous work which had only considered the binary case, thus uncovering stark differences between the two cases. We show how robust inference can be modeled as a zero-sum game between a learner who maximizes the expected accuracy, and an adversary. The value of this game is the best-attainable accuracy rate of any algorithm. We then show how the optimal policy for both the learner and adversary can be exactly characterized in terms of a particular hypergraph, specifically, as the hypergraph's maximum fractional independent set and minimum fractional set cover, respectively. This characterization yields efficient algorithms in the size of the domain (number of possible inputs). For the typical setting that the domain is huge, we also design efficient local computation algorithms for approximating maximum fractional independent set in hypergraphs. This leads to a near optimal algorithm for the learner whose complexity is independent of the domain size, instead depending only on the rank and maximum degree of the underlying hypergraph, and on the desired approximation ratio.

1. Introduction

Consider a detection system whose goal is to determine the access level of a given user. The system observes a large variety of legitimate users and the access level they should be granted. Its task is to guard against malicious users trying to disguise themselves to gain illegitimate access by modifying some of their behavioral characteristics. The challenge is to accurately determine access level in the face of such deliberate maliciousness.

Such a task can be viewed as a classification problem with adversarial corruption of data. The learner (in this case, the detection system) attempts to classify an input (a description of a user) according to its label (access level), but is thwarted by an adversary (malicious user) who modifies the original, uncorrupted input to produce a corrupted input (modified behavioral characteristics). The goal is to determine the correct label, given only the corrupted input.[1]

In this paper, we study the general problem of designing inference algorithms for multiclass classification that are robust against adversarial corruption of inputs.[2] We focus on the case in which the corruption is of a worst-case adversarial nature. This case is interesting for two reasons. First, positive results in the adversarial model extend to any other setting. Second, adversarial corruption can model situations in which corruption is due to maliciousness (such as a malicious user trying to gain a higher access level) that is not merely the result of random noise.

We model such situations as a zero-sum game between a learner and an adversary who may choose how inputs are corrupted. Given an input corrupted by the adversary, the learner's goal is to accurately predict the label of the original, uncorrupted input. Since our focus is on this inference task, we suppose that key aspects of the problem at hand are given, or have already been determined through learning or other means. Specifically, we assume that uncorrupted inputs are randomly generated by a distribution that is known, as is common in other inference works. We also assume that the set of corrupted inputs to which any particular uncorrupted input can be mapped is bounded and known to the learner. Finally, we suppose that the learner is able to query the target function (i.e., obtain the correct label of uncorrupted inputs of its choosing).

We show that this inference problem is fundamentally related to properties of a particular, natural hypergraph. The vertices of this hypergraph are the uncorrupted inputs, and each hyperedge is a set of uncorrupted inputs that are "confusable" in the sense that they all can be mapped to the same corrupted input, but all have different labels. In terms of this hypergraph, we prove that the learner's optimal policy (strategy for selecting labels on a given uncorrupted input) can be characterized exactly as a maximum fractional independent set. Furthermore, the adversary's optimal policy can be similarly characterized as a minimum fractional set cover on this same hypergraph.

As discussed shortly, this characterization substantially generalizes earlier work of Feige et al. (2015) on binary (two-class) classification to the multiclass case, and furthermore, points to intrinsic differences in the nature of the two cases. For instance, in the binary case, an optimal learner can always make deterministic label predictions, whereas in the multiclass case, randomized predictions are sometimes required for optimality. In terms of computation, our characterization immediately yields an efficient algorithm for finding an optimal learner policy that runs in time polynomial in the input domain size.

In many settings, however, we require algorithms that run in time polynomial in the logarithm of the input domain size, i.e., polynomial in the dimension. For this, we develop an efficient local algorithm for approximating the maximum fractional independent set in a hypergraph. The end result is an efficient algorithm for robust inference whose accuracy is near optimal.

In a nutshell, local algorithms receive as input a particular query (e.g., is a given vertex in the independent set), and can access the input through probes (e.g., for a graph, probing the existence of some edge). They then must produce an output for the query. The goal

1. See Mansour et al. (2015) for other motivating examples from spam detection and hardware failures.
2. Corruption can manifest in multiple ways, for example: missing attributes, incorrect measurement units (Kilogram vs. pounds, dollars vs. cents, etc.) and in general almost any noise models.

is that the output should be feasible, that is, consistent across queries (e.g., combining the replies to all queries should produce an independent set), and should also be near optimal.

Our work reinforces the connection between classical machine learning notions, such as inference, and the design of sub-linear algorithms, specifically, local algorithms. The connections already appear in (Mansour et al., 2015; Feige et al., 2015) where local algorithms were designed for solving a specific linear program (Mansour et al., 2015) and minimal vertex cover (Feige et al., 2015). We extend the connection to include maximum fractional independent set and minimum fractional set cover. An intriguing aspect is that we require only the fractional solution, which can be computed in polynomial time.

Related work

The most related work is that of Feige et al. (2015) which discusses the same model of robust inference, but restricted to binary classification. This assumption results in modeling the problem as a bipartite graph where the optimal adversary policy uses a maximum matching and the optimal learner policy uses a minimum vertex cover. In this work, we extend the setting to an arbitrary number of labels, revealing a distinct and rich structure not seen in the binary case. Here, the modeling moves from a bipartite graph to a hypergraph, and the optimal policies of the learner and the adversary change to be a maximum independent set and minimum set cover. Unlike matching and vertex cover, both polynomially solvable on bipartite graphs, maximum independent set and minimum set cover are hard to approximate. Fortunately, we need to find only a fractional solution, thus admitting efficient algorithms.

The work of Mansour et al. (2015) discusses a *static* robust inference problem, where the adversary selects one of m modification rules to corrupt the input *before* the uncorrupted input is sampled. Their main result is an efficient algorithm to derive a near optimal policy for the learner. The main difference is that in their model, the modification rule is selected *before* the uncorrupted input is realized, while in our model the adversary selects the corruption (modification rule) *after* the uncorrupted input is realized.

The work of Globerson and Roweis (2006) and its extensions (Teo et al., 2007; Dekel and Shamir, 2008) discuss a robust learning model where an uncorrupted sample is drawn from an unknown distribution, and the goal is to learn a linear classifier that would be able to overcome missing attributes in future test examples. They discuss both the static model (where the set of missing attributes is selected independently from the uncorrupted input) and the dynamic model (where the set of missing attributes may depend on the uncorrupted input). The work of Feige et al. (2015) extends the robust learning model to handle corrupted inputs (and not only missing attributes) and an arbitrary hypothesis class (rather than only linear classifiers).

Local computation approximation algorithms for combinatorial problems such as maximal independent set and hypergraph 2-coloring were derived in (Rubinfeld et al., 2011; Alon et al., 2012; Mansour et al., 2012). The works of (Mansour and Vardi, 2013; Even et al., 2014) gave local algorithms which find a near optimal maximum matching in bipartite graphs. The work of Feige et al. (2015) gives a near optimal minimum vertex cover for bipartite graphs, and based on it derives an efficient near optimal robust inference for binary classification.

Our local algorithms use the multiplicative weights approach, which has many applications in designing approximation algorithms (see, Arora et al. (2012) for a survey). Many such algorithms (e.g., Allen-Zhu and Orecchia (2015)) have a logarithmic number of iterations; however, converting such algorithms to a local algorithm would naively imply that their running time might be linear. The main difference is that our algorithm does not depend on the size of the hypergraph, but only on its rank and degree.

There is a vast literature in statistics, operation research and machine learning regarding various noise models. Typically, most noise models assume a random process that generates the noise. In computational learning theory, popular noise models include random classification noise (Angluin and Laird, 1988) and malicious noise (Valiant, 1985; Kearns and Li, 1993). In the malicious noise model, the adversary gets to arbitrarily corrupt some small fraction of the examples; in contrast, in our model the adversary can always corrupt every example, but only in a limited way.

In statistics, there is a notion of *robust statistics* (see Huber (1981)) whose main goal is to ensure that the small fraction of corrupted inputs (also referred to as outliers) will not significantly change the outcome of the estimation. The main issue is the ability of the learning algorithm to overcome a (small) fraction of outliers. Over the years, for many important machine learning tasks robust statistical estimators have been developed. The main difference is that in our setting *every* input is affected.

Another vast literature in operations research which is remotely related is that of robust optimization and stability, where the main task is to understand how minor changes in the input can affect the outcome of a (specific) algorithm (see, Ben-Tal et al. (2009)). We differ in two important ways from that literature. First, we do not assume any metric for our corruptions, while in large part the robustness there is measured as a function of the norm of the modification. Second, that literature, to a large extent, discusses the stability and robustness of specific algorithms, such as linear programs, while our starting point is trying to derive algorithms immune to such corruption.

2. The inference model

We begin with a formal description of our inference problem. There is a finite domain $\mathcal{X}$ from which uncorrupted inputs are generated randomly according to some distribution D, and a finite domain $\mathcal{Z}$ of corrupted inputs (possibly the same as $\mathcal{X}$). There is a set of k labels $\mathcal{Y} = \{1, \ldots, k\}$, with $\Delta(\mathcal{Y})$ denoting the set of distributions over $\mathcal{Y}$. An arbitrary target function $f : \mathcal{X} \to \mathcal{Y}$ maps each uncorrupted input $x \in \mathcal{X}$ to its "true" label $f(x) \in \mathcal{Y}$.

As discussed earlier, the adversary maliciously replaces each uncorrupted input $x \in \mathcal{X}$ with a corrupted version $z \in \mathcal{Z}$; we suppose that such a replacement z is restricted to come from a nonempty set $\rho(x) \subseteq \mathcal{Z}$. We assume that $1 \leq |\rho(x)| \leq \alpha$ and $|\rho^{-1}(z)| \leq \beta$ where $\rho^{-1}(z) = \{x : z \in \rho(x)\}$.

Thus, the basic scenario is the following. First, an uncorrupted input x is randomly selected according to D. Next, given x, the adversary selects some corrupted input $z \in \rho(x)$. The learner observes z and predicts a distribution over labels $h(z) \in \Delta(\mathcal{Y})$ for the label of $f(x)$. Finally, the learner attains a gain $g(h(z), f(x))$. In this work, we use the accuracy gain $g(p, y) = p(y)$, for $p \in \Delta(\mathcal{Y})$ and $y \in \mathcal{Y}$, which measures the probability of a randomized prediction based on p actually matching the correct label y. (We use gain rather than loss for mathematical convenience.) Since our focus is on the inference problem of predicting $f(x)$ given z, we assume that the learner has direct knowledge or query access to f, ρ, ρ^{-1} and D.

The goal of the learner is to maximize the expected gain, while the adversary would like to minimize it. This defines a zero-sum game which has a value *accuracy**. Specifically, the learner's optimal gain is

$$\textit{accuracy}^* = \max_{h:\mathcal{Z}\to\Delta(\mathcal{Y})} E_{x\sim D}[\min_{z\in\rho(x)} g(h(z), f(x))].$$

Intuitively, the max operator is the learner's selection of a hypothesis h to maximize the accuracy, and the min operator is the adversary's selection of a corrupted input z based on the realized uncorrupted input x to minimize the accuracy. The goal of the learner is to (implicitly) define a hypothesis $h : \mathcal{Z} \to \Delta(\mathcal{Y})$ whose expected accuracy is close to *accuracy**.

We say that the learner's hypothesis is ϵ-optimal if it guarantees accuracy at least $\textit{accuracy}^* - \epsilon$, and the adversary policy is ϵ-optimal if it guarantees an accuracy which is at most $\textit{accuracy}^* + \epsilon$. We refer to a 0-*optimal* policy (for either player) as simply an *optimal* policy.

Note that there is no hypothesis class, and so any hypothesis $h : \mathcal{Z} \to \Delta(\mathcal{Y})$ is permitted. Furthermore, the learner can access the target function f through a query oracle, which can be thought of as solving the inference problem for the uncorrupted inputs, i.e., given an uncorrupted input deduce its label. Note that in our robust inference setting there is no explicit sample; however, the learner can generate uncorrupted inputs using the distribution D and label them using its access to the target function f.

3. Characterizing the optimal policies

In this section we characterize exactly the optimal policies of both the learner and the adversary in terms of a particular hypergraph $G(\mathcal{X}, \mathcal{E})$. The vertices of G are the uncorrupted inputs, namely $\mathcal{X}$, and every vertex $x \in \mathcal{X}$ is weighted by its probability $D(x)$. The hyperedges of G consist of all sets e of uncorrupted inputs that are "maximally confusable" in the sense that: (1) they can all be mapped by the adversary to the same corrupted input z; (2) they all have different labels; and (3) the set e is maximal, meaning it cannot be made larger while maintaining the first two properties. More precisely, for each corrupted input $z \in \mathcal{Z}$ let $\kappa(z) \le k$ be the number of different labels in $\rho^{-1}(z)$, i.e., $\kappa(z) = |\{f(x) : z \in \rho(x)\}|$. We include a hyperedge $e \in \mathcal{E}$ for every subset $e \subseteq \rho^{-1}(z)$ consisting of exactly $\kappa(z)$ vertices, all with different labels (so $|e| = \kappa(z)$, and $f(x_1) \neq f(x_2)$ for all $x_1, x_2 \in e$). We associate the corrupted input z with e, and define $cor(e) = z$.

3.1. Adversary policy

In terms of the hypergraph G, we claim that the optimal adversary strategy can be computed using the following linear program, called LP-cover:

$$\min_{w_e} \sum_{e\in\mathcal{E}} w_e \quad \text{such that} \quad \forall x \in \mathcal{X} \quad \sum_{e:x\in e} w_e \ge D(x)$$
$$\forall e \in \mathcal{E} \quad w_e \ge 0.$$

The LP defines a nonnegative weight w_e for each edge $e \in \mathcal{E}$, along with fractional covering constraints that ensure the weights of edges incident to each vertex x are not less than its weight $D(x)$. This defines a fractional set-cover problem, with weights on the elements. Indeed, standard set cover (which is NP-complete) is obtained by setting $D(x) = 1$ for all x, and by requiring that the weights w_e be integers.

Lemma 1 *The value of LP-cover upper bounds the learner's accuracy. Furthermore, LP-cover's solution yields an explicit optimal policy for the adversary.*

The adversary strategy will be define by computing weights for each edge $e \in \mathcal{E}$ of G to achieve a fractional cover, namely,
Proof [Proof sketch:] The adversary solves LP-cover and uses the weights w_e to define a policy as follows for a given input x. First, from among all edges that include x, the adversary selects edge e randomly with probability $w_e / \sum_{e':x\in e'} w_{e'}$, and maps x to $cor(e)$. The main idea in the proof is to upper bound the accuracy, for any learner hypothesis $h(z) \in \Delta(\mathcal{Y})$, by

$$\Pr_{x,z}[h(z) = f(x)] = \sum_{e\in\mathcal{E}} \sum_{x\in e} \frac{w_e}{\sum_{e':x\in e'} w_{e'}} D(x) \Pr[h(e) = f(x)] \leq \sum_{e\in\mathcal{E}} w_e \ .$$

■

3.2. Learner policy

For the learner we consider the dual program to LP-cover, called LP-MIS, which finds weights b_x on the vertices:

$$\max_{b_x} \sum_{x\in\mathcal{X}} b_x D(x) \quad \text{such that} \quad \forall e \in \mathcal{E} \quad \sum_{x\in e} b_x \leq 1$$
$$\forall x \in \mathcal{X} \quad b_x \geq 0.$$

This LP computes a fractional independent set; standard independent set is obtained, as before, by setting $D(x) = 1$ for all x, and by requiring an integral solution.

Claim 2 *The value of LP-MIS is a lower bound on the learner's accuracy. Furthermore, LP-MIS's solution yields an explicit optimal policy for the learner.*

Proof [Proof sketch:] The learner solves LP-MIS, and uses b_x to define the following hypothesis. For each corrupted input $z \in \mathcal{Z}$ and label $y \in \mathcal{Y}$ let $\max(y|z) = \max_{x\in\rho^{-1}(z), f(x)=y} b_x$, and $\max(y|z) = 0$ if there is no $x \in \rho^{-1}(z)$ with $f(x) = y$. The learner, given a corrupted input z, returns a distribution $h(z) \in \Delta(\mathcal{Y})$ such that $\Pr[h(z) = y] = \frac{\max(y|z)}{\sum_{j\in\mathcal{Y}} \max(j|z)}$. In the full proof we show that for any $x \in \rho^{-1}(z)$ and $f(x) = y$ we have $\Pr[h(z) = y] \geq b_x$. The learner accuracy can be lower bounded as follows:

$$E_{x,z}[\Pr[h(z) = f(x)]] \geq \sum_{x\in\mathcal{X}} D(x) \min_{z\in\rho(x)} \Pr[h(z) = f(x)] \geq \sum_{x\in\mathcal{X}} D(x) \min_{z\in\rho(x)} b_x = \sum_{x\in\mathcal{X}} D(x) b_x,$$

which implies that the objective function of LP-MIS lower bounds the accuracy. ■

If the weights b_x happen to be integral, then LP-MIS yields an ordinary independent set, resulting in a prediction strategy that is perhaps more intuitive: Let $S \subset X$ be an independent set, i.e., for any $e \in \mathcal{E}$ we have $|e \cap S| \leq 1$. Given an observable z, we claim that the inputs in $\rho^{-1}(z) \cap S$ must all have the same label.[3] Therefore, the learner policy is the following: Given z, if $\rho^{-1}(z) \cap S \neq \emptyset$ we return $f(x)$ for some $x \in \rho^{-1}(z) \cap S$, otherwise return an arbitrary label.

In the binary case, it was shown in Feige et al. (2015) that the learner's optimal policy is deterministic (without loss of generality). However, with $k \geq 3$ classes, the use of randomized predictions, as in the policy defined by LP-MIS, becomes necessary for optimality. As a concrete example, suppose there are $k = 3$ classes, and that D is uniform over a domain $\mathcal{X}$ consisting of three inputs x_1, x_2, x_3, each labeled by its index (so $f(x_i) = i$ for $i = 1, 2, 3$). Furthermore, suppose $\mathcal{Z}$ consists of three inputs $z_{1,2}, z_{1,3}, z_{2,3}$, with $\rho^{-1}(z_{i,j}) = \{x_i, x_j\}$. Then the optimal (randomized) policy, given $z_{i,j}$, chooses uniformly at random between classes i and j, thus obtaining accuracy $1/2$. However, it can be shown that no deterministic policy achieves accuracy exceeding $1/3$.

3.3. Optimality

We showed that the value of LP-cover upper bounds the learner's accuracy and that LP-MIS lower bounds the learner's accuracy. By the strong duality theorem, this implies that the value of the zero-sum game, induced by the robust inference game, is exactly this value.

Theorem 3 *Let accuracy* be the value of the LP-cover (and LP-MIS). The learner can guarantee an accuracy of at least accuracy* and the adversary can guarantee that the accuracy is at most accuracy*. Furthermore, the optimal policies can be computed in time polynomial in $|\mathcal{X}|$, and $|\mathcal{E}|$.*

In the above theorem, given a corrupted input z, to generate a prediction $h(z)$ the algorithm queries f on all inputs $\mathcal{X}$. In Section 4, we design a local algorithm that generates predictions according to a near optimal learner policy, and the number of queries depends on α and β but not on $|\mathcal{X}|$. An important observation is that given access to values b_x that give rise to a near optimal value of LP-MIS we can generate a near optimal prediction. By observing that in the proof of Lemma 2, for computing $h(z)$ the learner needs to access b_x only for $x \in \rho^{-1}(z)$, we derive the following corollary.

Corollary 4 *Given oracle access to values b_x which have a value* `accuracy` *for LP-MIS, we can compute a learner prediction $h(z)$ while accessing only β values of b_x, and guarantee a learner accuracy of* `accuracy`*.*

3. Otherwise, there is a corrupted input z and $x_1, x_2 \in S$, $x_1, x_2 \in \rho^{-1}(z)$ and $f(x_1) \neq f(x_2)$. But there is an edge e such that $\{x_1, x_2\} \subset e \in \mathcal{E}$, and we have $|e \cap S| \geq 2$, contradicting the fact that S is an independent set.

4. Efficient Local Algorithm

In Section 3, we showed that the optimal learner policy is characterized by a maximum weight fractional independent set on a specific hypergraph. In this section, we derive a local computation algorithm that computes a near optimal fractional weighted independent set. We begin with an informal overview of local algorithms.

4.1. Informal overview

The goal of local algorithms is to handle the case that the input is so huge that we do not even want to read all of it. The main idea is that rather than computing a complete solution, we produce the solution as needed in the form of responses to queries.

For example, for computing a maximum independent set, the input is a graph $G(V, E)$ and a query asks when a given vertex $v \in V$ is in the independent set. We desire a few properties from such an algorithm. First, we would like it to maintain feasibility, meaning that if we consider all the possible queries $v \in V$ and the replies given by the algorithm, the set of vertices that it declares to be in the independent set actually do form an independent set. Achieving only feasibility is in fact fairly straightforward since we can simply claim that all the vertices are not in the independent set. This means that we also need to assure some level of quality for the resulting solution. Ideally, we would like to guarantee that the resulting independent set is of maximum size, but often we must settle for it being only approximately optimal, namely $1 - \epsilon$ times the optimal size.

In our setting we are given a hypergraph and would like to compute a near maximum weight fractional independent set. That is, given a vertex x, we would like to compute the weight b_x so that the weights satisfy the feasibility of LP-MIS, and the sum of the weights is near maximum. Our main goal is to develop an algorithm that depends only on the degree and the rank of the hypergraph, not on its size.

One popular method for developing local algorithms is to derive them from synchronous distributed algorithms Parnas and Ron (2007). A synchronous distributed algorithm works in rounds, where in each round each vertex sends information to its neighbors. (This is very similar to synchronous message passing algorithms.) The main insight is that the number of rounds of the distributed algorithm bounds the information flow from a vertex in the graph. Thus, if the distributed algorithm runs in T rounds, then the output of a vertex x cannot depend on any vertex y which is of distance $T + 1$ or more. This is because the information has to propagate in the distributed algorithm.

Simulating a distributed algorithm by a local algorithm is rather straightforward. If the distributed algorithm runs in T rounds, and we are interested in the output of vertex x, we do the following: We run all the vertices up to distance T from x for their first round, all the vertices up to distance $T - 1$ for two rounds, and in general, run vertices at distance $T - t$ for t rounds. With this approach, any vertex has all the information it needs when we run it. This means that the most crucial parameter is the number of rounds, which both bounds the number of simulation rounds and, more importantly, the radius of the neighborhood we need to consider.

Our basic distributed algorithm for the maximum weight fractional independent set uses an algorithm based on the multiplicative weights approach (see, Arora et al. (2012) for a survey). The algorithm works in rounds, building weights on vertices. Intuitively, we would like to increase the fractional weights on vertices which have few "conflicts." We achieve this effect by taking the set of least expensive vertices (up to a multiplicative factor of $1+\epsilon$) and increase their weights. Now, when a vertex increases its weight, it increases the cost of all the edges it belongs to, which in turn, increases the cost of the vertices. This intuitively implies that if we increase the weight of a vertex with a large connectivity, it will influence many edges. Also, a vertex with large connectivity is influenced by the many edges it shares, and its cost can increase significantly even without increasing its weight. One of the ingredients in our algorithm is the initialization of the costs.

In order to analyze our algorithm, we present a class of algorithms, to which our algorithm belongs. We show that any algorithm in this class of algorithms has the desired approximation ratio for the maximum weight fractional independent set on the given hypergraph.

4.2. The set-up

We next present our algorithm for an arbitrary hypergraph $G(\mathcal{X}, \mathcal{E})$, where $\mathcal{X}$ are the vertices and $\mathcal{E} \subset 2^{\mathcal{X}}$ are the hyperedges. We suppose that edges have rank at most r, i.e., $|e| \leq r$, and vertices have degree at most d, i.e., $|\{e : x \in e\}| \leq d$. (Note that in our application, $r \leq k$ and $d \leq \alpha\beta^k$.) We have a nonnegative weight function D over the vertices. We wish to find a fractional independent set of nearly maximal weight. For simplicity in stating our forthcoming algorithms, we assume that every vertex in G is contained in some hyperedge (this holds in our application since $\rho(x) \neq \emptyset$, and also in general since vertices which do not appear in any hyperedge can always be added to the independent set).

Given a vertex x, our local algorithm can be applied to determine the weight b_x, with running time that depends only on the degree d and rank r of G. By Corollary 4 this gives an efficient way to compute a near optimal learner policy.

Essentially, we will show that in order to determine b_x we need to consider only a certain radius T from x in G, which depends only on the degree and the rank and does not depend on the number of nodes $|\mathcal{X}|$. To bound this radius we use a distributed algorithm, which works in rounds, so the number of rounds would bound the radius. To prove the performance of our distributed algorithm, we present a class of algorithms that includes our distributed algorithm, and show that any algorithm in that class is near optimal.

4.3. A distributed algorithm for approximating weighted fractional MIS

We present a synchronous distributed algorithm (based on the multiplicative weights approach) parameterized by an integer T (for total number of rounds) and by a small error parameter $\epsilon > 0$. The algorithm assigns costs $c(e)$ to edges and values $\lambda(x)$ to vertices. The parameters (c or λ) at the end of round t are denoted by a subscript t. Initially, the cost of edge e is $c_0(e) = \max_{x \in e} D(x)$. Edge costs are updated throughout the execution of the algorithm. The edge costs induce costs $c(x)$ on vertices, where $c(x) = \sum_{e | x \in e} c(e)$. In addition, the algorithm assigns and updates integer values $\lambda(x)$ to vertices. Initially all these values are $\lambda_0(x) = 0$. The final fractional solution is obtained by appropriately scaling the final $\lambda_T(x)$ values.

Algorithm WFMIS:

1. Parameters: $T \geq 1$ and $\epsilon > 0$.

2. Initialize $\lambda_0(x) = 0$ for every vertex $x \in \mathcal{X}$, $c_0(e) = \max_{x\in e} D(x)$ for every edge $e \in \mathcal{E}$, and $c_0(x) = \sum_{e|x\in e} c_0(e)$.

3. Repeat for $1 \le t \le T$:

   (a) Every vertex x for which $\frac{c_{t-1}(x)}{D(x)} \le (1+\epsilon)^t$ is considered t-active.

   (b) For every t-active vertex x, update $\lambda_t(x) = \lambda_{t-1}(x) + 1$. For vertices that are not t-active, keep $\lambda_t(x) = \lambda_{t-1}(x)$.

   (c) For an edge e, let $A_t(e)$ denote the number of t-active vertices in e. For every edge e, update $c_t(e) = (1+\epsilon)^{A_t(e)} c_{t-1}(e) = (1+\epsilon)^{\sum_{x\in e}\lambda_t(x)} c_0(e)$.

   (d) For every vertex x, $c_t(x) = \sum_{e|x\in e} c_t(e)$.

4. For every edge e, let s_e be such that $c_T(e) = (1+\epsilon)^{s_e} c_0(e)$, i.e., $s_e = \sum_{x\in e} \lambda_T(x)$. For every vertex x, set $\bar\lambda(x) = \min_{e|x\in e}[\frac{\lambda_T(x)}{s_e}]$.

Proposition 5 *The values of $\bar\lambda(x)$ in WFMIS form a fractional independent set. Moreover, for $s = \max_e s_e$ it holds that $T - \frac{\log d}{\epsilon} \le s \le T + r$.*

Proof [Proof sketch:] For every edge $e \in \mathcal{E}$, for each $x \in e$ we have $\bar\lambda(x) \le \frac{\lambda_T(x)}{s_e}$, which implies that $\sum_{x\in e} \bar\lambda(x) \le \sum_{x\in e} \frac{\lambda_T(x)}{s_e} = 1$, implying that the $\bar\lambda(x)$ values are a fractional independent set. The upper bound that $s \le T + r$ is proved since we can "overshoot" by at most r. The lower bound follows from bounding the cost of the vertex with the highest weight. ■

4.4. Class of fraction MIS algorithms (NFMIS)

It remains to analyze the approximation ratio of algorithm WFMIS. For this we present a class of algorithms NFMIS. NFMIS will represent a class of algorithms, where members of this class differ by the certain choices that they make. We will show that every algorithm in this class has a good approximation ratio, and that WFMIS belongs to the class NFMIS. As in algorithm WFMIS, vertices and edges have costs, and vertices also have weights. Unlike WFMIS, in every iteration, only one vertex updates its value in each iteration.

Algorithm NFMIS (a class of algorithms):

1. Parameter: $T \ge 1$ and $\epsilon > 0$.

2. Initialize $\lambda_0(x) = 0$ for every vertex $x \in \mathcal{X}$, $c_0(e) = \max_{x\in e} D(x)$ for every edge $e \in \mathcal{E}$, and $c_0(x) = \sum_{e|x\in e} c_0(e)$.

3. Repeat for $t = 1, 2, \ldots$ and stop at an arbitrary value of $t = \tau$, provided that at $t = \tau$ the inequalities $(1+\epsilon)^T \le M_{\tau+1} \le (1+\epsilon)^{T+r}$ hold, where $M_t = \min_x[\frac{c_{t-1}(x)}{D(x)}]$:

   (a) Let x_t be an arbitrary vertex satisfying $M_t \le \frac{c_{t-1}(x)}{D(x)} \le (1+\epsilon)^r M_t$.

   (b) Update $\lambda_t(x_t) = \lambda_{t-1}(x_t) + 1$. For vertices $x \ne x_t$ keep $\lambda_t(x) = \lambda_{t-1}(x)$.

   (c) For every edge e containing x_t, update $c_t(e) = (1+\epsilon)c_{t-1}(e)$. For every edge e not containing x_t keep $c_t(e) = c_{t-1}(e)$.

   (d) For every vertex x, $c_t(x) = \sum_{e|x\in e} c_t(e)$.

4. Let τ denote the last iteration of the algorithm. For every edge e, let s_e be such that $c_\tau(e) = (1+\epsilon)^{s_e} c_0(e)$. For every vertex x, set $\bar{\lambda}(x) = \min_{e|x\in e}[\frac{\lambda_\tau(x)}{s_e}]$.

The different algorithms in NFMIS may differ in the decision when to stop (Step 3) and the selection of the vertex x_t (Step 3a).

Proposition 6 *The values $\bar{\lambda}(x)$ in NFMIS form a fractional independent set.*

The proof of Proposition 6 is similar to that of Proposition 5, and hence omitted.

Proposition 7 *The WFMIS algorithm belongs to the class of algorithms NFMIS.*

Proof [Proof sketch:] The proof is by induction on the number of iterations, showing that at the beginning of iteration t, for every vertex x we have $\frac{c_{t-1}(x)}{D(x)} \geq (1+\epsilon)^{t-1}$. This implies that all the active vertices are at most a factor $1+\epsilon$ from the minimum ratio. By making the process sequential, we might have a vertex wait for $r-1$ other vertices, which implies that it might be $(1+\epsilon)^r$ from the minimum ratio, and in NFMIS. The stopping condition is also related by the same ratio. ■

4.5. Near optimality

In both algorithms we scaled $\lambda(x)$ to $\bar{\lambda}$ by using the individual s_e. The feasibility is maintained if we used the coarser bound of $\bar{\lambda}(x) = \lambda(x)/s$, where $s = \max_e s_e$. In the following, to derive the approximation bound, we assume the coarser bound $\bar{\lambda}(x) = \lambda(x)/s$. Note that the more refined bound of $\bar{\lambda}(x) = \lambda(x)/s_e \geq \lambda(x)/s$ can only increase the weight of the fractional independent set.

Lemma 8 *Every algorithm within the class NFMIS has an approximation ratio no worse than*

$$\frac{W_\tau}{opt^*(\frac{(1+\epsilon)\ln rd}{\epsilon} + r + 1) + (1+2\epsilon)(1+2r\epsilon)W_\tau},$$

where $W_\tau = \sum_{t=1}^{\tau} D(x_t)$ and opt^* is the weight of a maximum fractional independent set.

Proof [Proof sketch:] Let $W = \sum_x D(x)$, $\lambda^*(x)$ be vertex values in a maximum weight fractional independent set in G, and let opt^* be its weight, namely $opt^* = \sum_x \lambda^*(x)D(x)$.

Let C_t be the total cost of edges at the end of round t, i.e., $C_t = \sum_e c_t(e)$. Consider an assignment c' of costs to vertices such that $c'(x) = \sum_{e:x\in e} c_t(e)\lambda^*(x)$. Since for every edge e we have $\sum_{x\in e}\lambda^*(x) \leq 1$, we have that,

$$C_{t-1} \geq \sum_{e\in\mathcal{E}}\sum_{x\in e} c_{t-1}(e)\lambda^*(x) \geq M_t \sum_{x\in\mathcal{X}} \lambda^*(x)D(x) = M_t opt^*,$$

since, by definition, $M_t = \min_x \frac{c_{t-1}(x)}{D(x)}$ and $opt^* = \sum_x \lambda^*(x)D(x)$. This implies that $M_t \leq \frac{C_{t-1}}{opt^*}$.

Recall that we denote by x_t the vertex chosen at iteration t (hence it could be that x_t and $x_{t'}$ for $t' \neq t$ actually refer to the same vertex in G). Then x_t has ratio at most $(1+\epsilon)^r M_t$. It follows that

$$C_t \leq C_{t-1} + \epsilon(1+\epsilon)^r M_t D(x_t) \leq C_{t-1}\left(1 + \frac{\epsilon(1+\epsilon)^r D(x_t)}{opt^*}\right).$$

For $\epsilon \leq \frac{1}{4r}$, we bound $C_\tau \leq C_0(1+\epsilon)^{(1+2\epsilon)(1+2r\epsilon)\frac{\sum_{t=1}^{\tau} D(x_t)}{opt^*}}$. We define $W_\tau = \sum_{t=1}^{\tau} D(x_t)$, and note that W_τ is the total weight accumulated by WFMIS before the scaling of Step 4, i.e., $W_\tau = \sum_x \lambda_\tau(x)D(x)$. Then we can write $C_\tau \leq dW(1+\epsilon)^{\frac{(1+2\epsilon)(1+2r\epsilon)W_\tau}{opt^*}}$, since $C_0 \leq dW$.

Given C_τ, the sum of costs of all edges, we upper bound $C_{\max} = \max_e[\frac{c_\tau(e)}{c_0(e)}]$, the largest multiplicative increase of cost of any edge e, and show that $C_{\max} \leq \frac{(1+\epsilon)^{r+1} r C_\tau}{W} \leq rd(1+\epsilon)^{\frac{(1+2\epsilon)(1+2r\epsilon)W_\tau}{opt^*}+r+1}$.

We derive that $C_{\max} \leq (1+\epsilon)^{\sigma + \frac{(1+2\epsilon)(1+2r\epsilon)W_\tau}{opt^*}+r+1}$, where $\sigma = \frac{(1+\epsilon)\ln rd}{\epsilon}$. Consequently, in Step 4 of NFMIS we have $s_e \leq \sigma + \frac{(1+2\epsilon)(1+2r\epsilon)W_\tau}{opt^*} + r + 1$ for every edge e. Since $s = \max_e s_e$ we have that $s \leq \sigma + \frac{(1+2\epsilon)(1+2r\epsilon)W_\tau}{opt^*} + r + 1$. We also recall that $\sum_x \lambda_\tau(x)D(x) = W_\tau$. Hence after scaling $\lambda_\tau(x)$ by at most s we have that the value of the fractional solution is at least $opt^* \frac{W_\tau}{opt^*(\sigma+r+1)+(1+2\epsilon)(1+2r\epsilon)W_\tau}$. ■

From Lemma 8 we can deduce the following theorem.

Theorem 9 *For a given $0 < \delta < \frac{1}{8}$ and for $d \geq 2$, algorithm WFMIS has an approximation ratio of at least $\frac{1}{1+\delta}$ whenever T is at least $\frac{11(r+3)\ln rd}{\delta^2}$.*

4.6. From distributed to local algorithm

The following theorem summarizes our main result regarding our efficient robust inference algorithm.

Theorem 10 *For any $0 < \delta < 1/8$, a $(1-\delta)$-optimal randomized strategy for the learner in the robust inference game can be computed by a local algorithm that, given a corrupted input z, explores a neighborhood of radius $T = O\left(\frac{k(\ln k\alpha + \max_{i<k}[i \ln \frac{\beta}{i}])}{\delta^2}\right) = O\left(\frac{k^2 \ln(\alpha\beta)}{\delta^2}\right)$ from the vertices $\rho^{-1}(z)$ and runs in time $O(\beta d^T) = exp(O(\frac{k^2 \log^2(\alpha\beta)}{\delta^2}))$.*
For a constant $k = O(1)$ this gives a radius of $O\left(\frac{\ln(\alpha\beta)}{\delta^2}\right)$ and a running time of $exp(\frac{\log^2(\alpha\beta)}{\delta^2})$.

References

Zeyuan Allen-Zhu and Lorenzo Orecchia. Using optimization to break the epsilon barrier: A faster and simpler width-independent algorithm for solving positive linear programs in parallel. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '15, pages 1439–1456, 2015.

Noga Alon, Ronitt Rubinfeld, Shai Vardi, and Ning Xie. Space-efficient local computation algorithms. In *SODA*, pages 1132–1139, 2012.

Dana Angluin and Philip Laird. Learning from noisy examples. *Mach. Learn.*, 2(4):343–370, April 1988.

Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(6):121–164, 2012.

A. Ben-Tal, L. El Ghaoui, and A.S. Nemirovski. *Robust Optimization.* Princeton Series in Applied Mathematics. Princeton University Press, October 2009.

Ofer Dekel and Ohad Shamir. Learning to classify with missing and corrupted features. In *Machine Learning, Proceedings of the Twenty-Fifth International Conference (ICML 2008)*, pages 216–223, 2008.

Guy Even, Moti Medina, and Dana Ron. Best of two local models: Local centralized and local distributed algorithms. *CoRR*, abs/1402.3796, 2014. URL http://arxiv.org/abs/1402.3796.

Uriel Feige, Yishay Mansour, and Robert E. Schapire. Learning and inference in the presence of corrupted inputs. In *Proceedings of The 28th Conference on Learning Theory, COLT 2015*, pages 637–657, 2015.

Amir Globerson and Sam T. Roweis. Nightmare at test time: robust learning by feature deletion. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006)*, pages 353–360, 2006.

Peter J. Huber. *Robust Statistics.* Wiley, 1981.

Michael J. Kearns and Ming Li. Learning in the presence of malicious errors. *SIAM J. Comput.*, 22(4):807–837, 1993.

Yishay Mansour and Shai Vardi. A local computation approximation scheme to maximum matching. In *APPROX-RANDOM*, pages 260–273, 2013.

Yishay Mansour, Aviad Rubinstein, Shai Vardi, and Ning Xie. Converting online algorithms to local computation algorithms. In *ICALP (1)*, pages 653–664, 2012.

Yishay Mansour, Aviad Rubinstein, and Moshe Tennenholtz. Robust probabilistic inference. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 449–460, 2015.

M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theoretical Computer Science*, 381(1–3), 2007.

Ronitt Rubinfeld, Gil Tamir, Shai Vardi, and Ning Xie. Fast local computation algorithms. In *ICS*, pages 223–238, 2011.

Choon Hui Teo, Amir Globerson, Sam T. Roweis, and Alexander J. Smola. Convex learning with invariances. In *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*, pages 1489–1496, 2007.

L. G. Valiant. Learning disjunction of conjunctions. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'85, pages 560–566, 1985.

Appendix A. Missing proofs

Proof [Proof of Lemma 1] Given an input x, the adversary solves the LP-cover and uses the weights w_e to define a policy as follows. First, the adversary selects an edge e such that $x \in e$ with probability $w_e / \sum_{e':x\in e'} w_{e'}$. Recall that each edge e is associated with a unique corrupted input $cor(e)$. Given that the adversary selected edge e it maps x to $cor(e)$. (By construction we have $cor(e) \in \rho(x)$.) The learner hypothesis predicts some $h(z) \in \Delta(\mathcal{Y})$, where $h(z)(j)$ is the probability for label j. We allow the learner to have a prediction h_e for each edge e, thus allow the prediction to depend on the selected edge (and not only the selected corrupted input z, essentially disclosing more information to the learner). The upper bound on the expected accuracy is computed as follows,

$$\begin{aligned}
\Pr_{x,z}[h(z) = f(x)] = E_{x,z}[h(z)(f(x))] &= \sum_{e\in\mathcal{E}} \sum_{x\in\mathcal{X}} \Pr[e|x]\Pr[x] h_e(f(x)) \\
&= \sum_{e\in\mathcal{E}} \sum_{x\in\mathcal{X}} \frac{w_e}{\sum_{e':x\in e'} w_{e'}} D(x) I[x\in e] h_e(f(x)) \\
&\le \sum_{e\in\mathcal{E}} w_e \sum_{x\in\mathcal{X}} I[x\in e] h_e(f(x)) = \sum_{e\in\mathcal{E}} w_e \sum_{x\in e} h_e(f(x)) \\
&\le \sum_{e\in\mathcal{E}} w_e \sum_{y\in\mathcal{Y}} h_e(y) = \sum_{e\in\mathcal{E}} w_e \ ,
\end{aligned}$$

where the first inequality follows from the constraint $D(x) \le \sum_{e':x\in e'} w_{e'}$ in LP-cover, the second inequality follows since there is at most one vertex in e for each label in $\mathcal{Y}$, and the last equality follows since $h_e \in \Delta(\mathcal{Y})$. ■

Proof [Proof of Lemma 2]We show how the learner can use the values of b_x to build a prediction. Given the weight b_x the learner defines the following hypothesis. For each corrupted input $z \in \mathcal{Z}$ and label $y \in \mathcal{Y}$ we define $\max(y|z) = \max_{x\in\rho^{-1}(z), f(x)=y} b_x$, and $\max(y|z) = 0$ if $\rho^{-1}(z) = \emptyset$. Namely, $\max(y|z)$ is the maximum value of b_x over all uncorrupted inputs x which can be mapped to z and have label y. The learner hypothesis, given a corrupted input z, returns a distribution $h(z) \in \Delta(\mathcal{Y})$ such that $h(z)(y) = \frac{\max(y|z)}{\sum_{j\in\mathcal{Y}} \max(j|z)}$. Namely, $h(z)$ normalizes the values $\max(y|z)$ to a distribution.

Note that $\sum_{j\in\mathcal{Y}} \max(j|z) \le 1$ since there is an edge $e \in \mathcal{E}$ that includes all $x_j = \arg\max_{x\in\rho^{-1}(z), f(x)=j} b_x$. We can now relate the accuracy to the b_x values as follows. For any $x \in \rho^{-1}(z)$ and $f(x) = y$ we have,

$$h(z)(y) = \frac{\max(y|z)}{\sum_{j\in\mathcal{Y}} \max(j|z)} \ge \max(y|z) \ge b_x$$

The learner accuracy can be lower bounded as follows:

$$\Pr_{x,z}[h(z) = f(x)] \ge \sum_{x\in\mathcal{X}} D(x) \min_{z\in\rho(x)} h(z)(f(x)) \ge \sum_{x\in\mathcal{X}} D(x) \min_{z\in\rho(x)} b_x = \sum_{x\in\mathcal{X}} D(x) b_x,$$

which implies that the objective function of LP-MIS lower bounds the accuracy. ■

Proof [Proof of Corollary 4] We can observe, as in the proof of Lemma 2, that we need to compute,

$$h_z(y) = \frac{\max(y|z)}{\sum_{j\in\mathcal{Y}} \max(j|z)} \geq \max(y|z) \geq b_x \ ,$$

and for that we need only the values b_x of $x \in \rho^{-1}(z)$, and there are at most β such values. Lemma 2 guarantees that the accuracy will be at least `accuracy`. ■

Proof [Proof of Proposition 5]For every edge $e \in \mathcal{E}$, for each $x \in e$ we have $\bar{\lambda}(x) \leq \frac{\lambda_T(x)}{s_e}$, which implies that $\sum_{x\in e} \bar{\lambda}(x) \leq \sum_{x\in e} \frac{\lambda_T(x)}{s_e} = 1$, implying that the $\bar{\lambda}(x)$ values are a fractional independent set.

Now we show that $s_e \leq T + r$. Consider the last iteration t at which edge e contained an active vertex x (clearly, $t \leq T$). At that iteration, $\frac{c_{t-1}(x)}{D(x)} \leq (1+\epsilon)^t$, and consequently $c_{t-1}(e) \leq (1+\epsilon)^t D(x) \leq (1+\epsilon)^t c_0(e)$. In that iteration, at most $|e| \leq r$ vertices in e were active, hence $c_t(e) \leq (1+\epsilon)^r c_{t-1}(e)$. Hence $c_T(e) = c_t(e) \leq (1+\epsilon)^r c_{t-1}(e) \leq (1+\epsilon)^{t+r} c_0(e) \leq (1+\epsilon)^{T+r} c_0(e)$, as desired.

We now show that $s \geq T - \frac{\log d}{\epsilon}$. First, we show by induction on the number of iterations of WFMIS, that at the beginning of iteration t, for every vertex x we have $\frac{c_{t-1}(x)}{D(x)} \geq (1+\epsilon)^{t-1}$. The base case is $t = 1$ and follows from the facts that every vertex of G is contained in some edge and that $c_0(e) \geq D(x)$ for every edge containing x. For the inductive step, observe that if a vertex x is active in round t then its cost increases by a factor of at least $(1+\epsilon)$ in that round, because of the increase that x itself inflicted on its incident edges.

Let x be the vertex with highest $D(x)$. Then after the last iteration T we have that $\frac{c_T(x)}{D(x)} \geq (1+\epsilon)^T$. This implies that x has an incident edge e with $c_T(e) \geq \frac{c_T(x)}{d} \geq \frac{(1+\epsilon)^T D(x)}{d}$. Maximality of $D(x)$ implies that $c_0(e) = D(x)$, and hence $s \geq s_e \geq T - \frac{\log d}{\epsilon}$. ■

Proof [Proof of Proposition 7]We show by induction on the number of iterations of WFMIS, that at the beginning of iteration t, for every vertex x we have $\frac{c_{t-1}(x)}{D(x)} \geq (1+\epsilon)^{t-1}$. The base case is $t = 1$ and follows from the facts that every vertex of G is contained in some edge and that $c_0(e) \geq D(x)$ for every edge containing x. For the inductive step, observe that if a vertex x is active in round t then its cost increases by a factor of at least $(1+\epsilon)$ in that round, because of the increase that x itself inflicted on its incident edges.

The inequality $\frac{c_{t-1}(x)}{D(x)} \geq (1+\epsilon)^{t-1}$ implies that in every round of WFMIS, the active vertices all have ratio within a factor of $(1+\epsilon)$ from the minimum ratio at that round. However, as vertices are processed in parallel in a round, if we would try to serialize the activations, an active vertex x might be considered only after some neighbors already increased the cost of its incident edges, at most by a factor of $(1+\epsilon)^{r-1}$. Hence x's own ratio in the serial activation might be a factor of $(1+\epsilon)^{r-1}(1+\epsilon) = (1+\epsilon)^r$ larger than the minimum vertex ratio at the time that x becomes active. Step 3a of NFMIS accommodates for this.

Observe also that when WFMIS stops, the minimum ratio vertex x has $(1+\epsilon)^T \leq \frac{c_t(x)}{D(x)} \leq (1+\epsilon)^{T+r}$. This is in agreement with the stopping condition of NFMIS. ■

Proof [Proof of Lemma 8] Let $W = \sum_x D(x)$ (in the case that D is a distribution we have $W = 1$). Let $\lambda^*(x)$ be vertex values in a maximum weight fractional independent set in G, and let opt^* be its weight, namely $opt^* = \sum_x \lambda^*(x)D(x)$.

Let C_t be the total cost of edges at the end of round t, i.e., $C_t = \sum_e c_t(e)$. Consider an assignment c' of costs to vertices such that $c'(x) = \sum_{e:x\in e} c_t(e)\lambda^*(x)$. Since for every edge e we have $\sum_{x\in e}\lambda^*(x) \le 1$, then we have that,

$$\begin{aligned}
C_{t-1} =& \sum_{e\in\mathcal{E}} c_{t-1}(e)\\
\ge& \sum_{e\in\mathcal{E}}\sum_{x\in e} c_{t-1}(e)\lambda^*(x)\\
=& \sum_{x\in\mathcal{X}} \lambda^*(x) \sum_{e:x\in e} c_{t-1}(e)\\
=& \sum_{x\in\mathcal{X}} \lambda^*(x)D(x)\frac{c_{t-1}(x)}{D(x)}\\
\ge& M_t \sum_{x\in\mathcal{X}} \lambda^*(x)D(x) = M_t opt^*,
\end{aligned}$$

since, by definition, $M_t = \min_x \frac{c_{t-1}(x)}{D(x)}$ and $opt^* = \sum_x \lambda^*(x)D(x)$. This implies that

$$M_t \le \frac{C_{t-1}}{opt^*}.$$

Recall that we denote by x_t the vertex chosen at iteration t (hence it could be that x_t and $x_{t'}$ for $t' \ne t$ actually refer to the same vertex in G). Then x_t has ratio at most $(1+\epsilon)^r M_t$. It follows that

$$\begin{aligned}
C_t =& C_{t-1} + \epsilon \sum_{e:x_t\in e} c_{t-1}(e)\\
=& C_{t-1} + \epsilon c_{t-1}(x_t)\\
\le& C_{t-1} + \epsilon(1+\epsilon)^r M_t D(x_t)\\
\le& C_{t-1} + \epsilon(1+\epsilon)^r \frac{C_{t-1}}{opt^*} D(x_t)\\
\le& C_{t-1}\left(1 + \frac{\epsilon(1+\epsilon)^r D(x_t)}{opt^*}\right).
\end{aligned}$$

Observe that $1+\epsilon z \le (1+\epsilon)^{z(1+2\epsilon)}$ for $\epsilon \le 1/2$,[4] and that $(1+\epsilon)^r \le (1+2r\epsilon)$ for $\epsilon \le \frac{1}{4r}$. Consequently, $C_t \le C_{t-1}(1+\epsilon)^{(1+2\epsilon)(1+2r\epsilon)\frac{D(x_t)}{opt^*}}$, and $C_\tau \le C_0(1+\epsilon)^{(1+2\epsilon)(1+2r\epsilon)\frac{\sum_{t=1}^{\tau} D(x_t)}{opt^*}}$. Observe that $C_0 \le dW$. We define $W_\tau = \sum_{t=1}^{\tau} D(x_t)$, and note that W_τ is the total weight accumulated by WFMIS before the scaling of Step 4, i.e., $W_\tau = \sum_x \lambda_\tau(x)D(x)$. Then we can write $C_\tau \le dW(1+\epsilon)^{\frac{(1+2\epsilon)(1+2r\epsilon)W_\tau}{opt^*}}$.

Given C_τ, the sum of costs of all edges, let us upper bound $C_{max} = \max_e[\frac{c_\tau(e)}{c_0(e)}]$, the largest multiplicative increase of cost of any edge e. Let x be the vertex whose last increase

4. $\ln(1+\epsilon z) \le \epsilon z$ and $z(1+2\epsilon)\ln(1+\epsilon) \ge z(1+2\epsilon)(\epsilon-\epsilon^2) = z\epsilon(1+\epsilon(1-2\epsilon))$

in value gave e the cost $C_{\max}c_0(e)$. Then prior to being last chosen, the ratio of x was at least $\frac{c_\tau(x)}{D(x)} \geq \frac{C_{\max}c_0(e)}{(1+\epsilon)D(x)} \geq \frac{C_{\max}}{1+\epsilon}$, where the last inequality follows because $c_0(e) \geq D(x)$. Step 3a of NFMIS implies that at that point every vertex had ratio at least $\frac{C_{\max}}{(1+\epsilon)^{r+1}}$, implying that $\frac{c_\tau(x)}{D(x)} \geq \frac{C_{\max}}{(1+\epsilon)^{r+1}}$ for every vertex x. Consequently, $\frac{\sum_x c_\tau(x)}{\sum_x D(x)} \geq \frac{C_{\max}}{(1+\epsilon)^{r+1}}$. As each edge has rank at most r, we also have that $\sum_x c_\tau(x) \leq rC_\tau$. Recall also that $\sum_x D(x) = W$. It follows that $C_{\max} \leq \frac{(1+\epsilon)^{r+1}rC_\tau}{W} \leq rd(1+\epsilon)^{\frac{(1+2\epsilon)(1+2r\epsilon)W_\tau}{opt^*}+r+1}$.

Let $\sigma = \frac{(1+\epsilon)\ln rd}{\epsilon}$. Since $1+z \geq exp(\frac{z}{1+z})$, for $z \geq 0$, we have that $(1+\epsilon)^\sigma \geq rd$. Then $C_{\max} \leq (1+\epsilon)^{\sigma+\frac{(1+2\epsilon)(1+2r\epsilon)W_\tau}{opt^*}+r+1}$. Consequently, in Step 4 of NFMIS we have $s_e \leq \sigma + \frac{(1+2\epsilon)(1+2r\epsilon)W_\tau}{opt^*} + r + 1$ for every edge e. Defining $s = \max_e s_e$ we have that $s \leq \sigma + \frac{(1+2\epsilon)(1+2r\epsilon)W_\tau}{opt^*} + r + 1$. We also recall that $\sum_x \lambda_\tau(x)D(x) = W_\tau$. Hence after scaling $\lambda_\tau(x)$ by at most s we have that the value of the fractional solution is at least $\frac{W_\tau}{\sigma+\frac{(1+2\epsilon)(1+2r\epsilon)W_\tau}{opt^*}+r+1} = opt^*\frac{W_\tau}{opt^*(\sigma+r+1)+(1+2\epsilon)(1+2r\epsilon)W_\tau}$. ■

Proof [Proof of Theorem 9] Consider the denominator of the approximation bound in Lemma 8: $opt^*(\frac{(1+\epsilon)\ln rd}{\epsilon} + r + 1) + (1+2\epsilon)(1+2r\epsilon)W_\tau$. For a given $0 < \delta < 1$, we wish to choose ϵ and T in such a way that we can express it as $(1+\delta)W_\tau$. First we choose ϵ to be small enough so that $(1+2\epsilon)(1+2r\epsilon) \leq 1+\frac{\delta}{2}$. Choosing $\epsilon = \frac{\delta}{4(r+3)}$ suffices for this purpose, and then the denominator becomes at most $opt^*(\frac{4(r+3)\ln rd}{\delta} + \ln rd + r + 1) + (1+\frac{\delta}{2})W_\tau$. Choosing $W_\tau \geq \bar{W} = \frac{2opt^*}{\delta}(\frac{4(r+3)\ln rd}{\delta} + \ln rd + r + 1)$ the denominator indeed becomes at most $(1+\delta)W_\tau$, and the approximation ratio in Lemma 8 is then $\frac{1}{1+\delta}$.

Recall that $s = \max_e s_e$. For the choices of ϵ and $\bar{W}$ as above, since we have a $\frac{1}{1+\delta}$ approximation, we have that

$$\frac{opt^*}{1+\delta} \leq \frac{W_\tau}{s} \leq opt^*.$$

This implies that

$$\frac{W_\tau}{opt^*} \leq s \leq (1+\delta)\frac{W_\tau}{opt^*}.$$

In addition we have

$$s \leq (1+\delta)\frac{W_\tau}{opt^*} = \frac{2+2\delta}{\delta}\left(\frac{4(r+3)\ln rd}{\delta} + \ln rd + r + 1\right) \leq \frac{10(r+3)\ln rd}{\delta^2}\ , \qquad (1)$$

where the last inequality uses the bounds on δ and d in the theorem, and the fact that $r \geq 2$.

Let us consider now algorithm WFMIS, and recall from Proposition 5 that there the number of rounds satisfies $T \leq s + \frac{\log d}{\epsilon}$. Using our upper bound on s and our choice of $\epsilon = \frac{\delta}{4(r+3)}$ we get that:

$$T \leq \frac{10(r+3)\ln rd}{\delta^2} + \frac{4(r+3)\log d}{\delta} \leq \frac{11(r+3)\ln rd}{\delta^2}$$

Observe that we started from a value $\bar{W}$, enforced $W_\tau \geq \bar{W}$, then deduced that $s \leq \frac{10(r+3)\ln rd}{\delta^2}$ and finally deduced that $T = \frac{11(r+3)\ln rd}{\delta^2}$ rounds suffice in WFMIS. To complete the proof we need to show that if we choose $T = \frac{11(r+3)\ln rd}{\delta^2}$ we indeed necessarily have that $W_\tau \geq \bar{W}$, validating the chain of implications. This follows from the fact that both s and W_τ are monotonically nondecreasing between rounds in NFMIS.

Specifically, consider running WFMIS for $\frac{11(r+3)\ln rd}{\delta^2}$ rounds. By Proposition 5, at that time necessarily $s > \frac{10(r+3)\ln rd}{\delta^2}$. Likewise, at that point $W_\tau \geq \bar{W}$. (Otherwise continue running NFMIS until $W_\tau \geq \bar{W}$ for the first time (this will eventually happen). At that time, by (1), we have $s \leq \frac{10(r+3)\ln rd}{\delta^2}$, and since s is monotone, we cannot have at an earlier time $s > \frac{10(r+3)\ln rd}{\delta^2}$.) ■

Proof [Proof of Theorem 10] Theorem 9 shows that our distributed algorithm runs in $T = O(\frac{r\ln rd}{\delta^2})$ rounds and computes a $1-\delta$ approximation to the maximum weighted fractional independent set for any algorithm in the class NFMIS. Proposition 7 shows that WFMIS belongs to the class of NFMIS algorithms. Similar to Parnas and Ron (2007) we can then transform WFMIS to a distributed algorithm to a local algorithm which needs access only to vertices up to distance T, which implies a running time of at most $O(d^T)$.

We now need to translate the parameters of the hypergraph G, rank r and degree d to the parameters of the original robust inference problems: α, β and k. Recall that $k = |Y|$ is the number of labels, α bounds the number of corrupted inputs an uncorrupted input can be mapped to, i.e., for any x we have $1 \leq |\rho(x)| \leq \alpha$, and β bound the number of pre-images of a corrupted input, i.e., for any z we have $|\rho^{-1}(z)| \leq \beta$.

Since an edge in G includes only inputs with different labels, we have that $r \leq k$. The degree α of an edge is upper bounded by $\alpha \max_{i<k} \binom{\beta-1}{i} \leq \alpha\beta^k$, since an uncorrupted input can be mapped to at most α corrupted inputs, and each one has at most β pre-images, finally, an edge includes at most k uncorrupted inputs with different labels.

Using the above bounds in $T = O(\frac{r\ln rd}{\delta^2})$ we get

$$T = O\left(\frac{k(\ln k\alpha + \max_{i<k}[i \ln \frac{\beta}{i}])}{\delta^2}\right) = O\left(\frac{k\ln k\alpha + k^2 \ln \beta}{\delta^2}\right) = O\left(\frac{k^2 \ln(\alpha\beta)}{\delta^2}\right)$$

The running time is $O(d^T)$ which becomes

$$O(d^T) = exp(O\left(\frac{k^2 \log^2(\alpha\beta)}{\delta^2}\right))$$

Since we found a $(1-\delta)$-optimal solution to the weighted fractional set cover problem, by Corollary 4 this implies a near optimal policy for the learner, with an extra factor of β for the maximum number of the pre-images of z. ■