SURE: Robust, Explainable, and Fair Classification without Sensitive Attributes

Deepayan Chakrabarti
deepay@utexas.edu
University of Texas at Austin
Austin, Texas, USA

ABSTRACT

A classifier that is accurate on average may still underperform for "sensitive" subsets of people. Such subsets could be based on race, gender, age, etc. The goal of a fair classifier is to perform well for all sensitive subsets. But often, the sensitive subsets are not known a priori. So we may want the classifier to perform well on all subsets that are likely to be sensitive. We propose an iterative algorithm called SURE for this problem. In each iteration, SURE identifies high-risk zones in feature space where the risk of unfair classification is statistically significant. By changing the loss function's weights for points from these zones, SURE builds a fair classifier. The emphasis on statistical significance makes SURE robust to noise. The high-risk zones are intuitive and interpretable. Every step of our method is explainable in terms of significance tests. Finally, SURE is fast and parameter-free. Experiments on both simulated and real-world datasets show that SURE is competitive with the state-of-the-art.

CCS CONCEPTS

• **Information systems → Data mining**.

KEYWORDS

fair classification, robustness, statistical significance

ACM Reference Format:
Deepayan Chakrabarti. 2023. SURE: Robust, Explainable, and Fair Classification without Sensitive Attributes. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23), August 6–10, 2023, Long Beach, CA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3580305.3599514

1 INTRODUCTION

We train classifiers by optimizing a chosen loss function over the training data. But, a lower overall loss may hide poor performance over some "sensitive" subsets of the data. For instance, a classifier trained to predict recidivism may underperform for African-Americans as compared to Whites or Asians. In other words, this classifier is unfair for subgroups based on race. More generally, we may have several sensitive attributes such as race, gender, age, etc. A classifier is unfair if it performs poorly for any attribute combination (say, `African-American males with ages between 18 and 30`). By minimizing the overall loss, we may inadvertently create an unfair classifier.

KDD '23, August 6–10, 2023, Long Beach, CA, USA

ACM ISBN 979-8-4007-0103-0/23/08.
https://doi.org/10.1145/3580305.3599514

Recently, there has been a surge of interest in designing fair classifiers. One line of work looks at quantifying fairness via different metrics. We can only achieve fairness for some metrics. For instance, it is impossible to have equal false positive and false negative rates for all sensitive groups, except in special cases [10, 24]. Other works propose algorithms to improve the fairness of classifiers. These impose fairness via constraints or regularizers in the classifier's loss function. By optimizing such a modified loss function, we hope to get a classifier that is both fair and accurate.

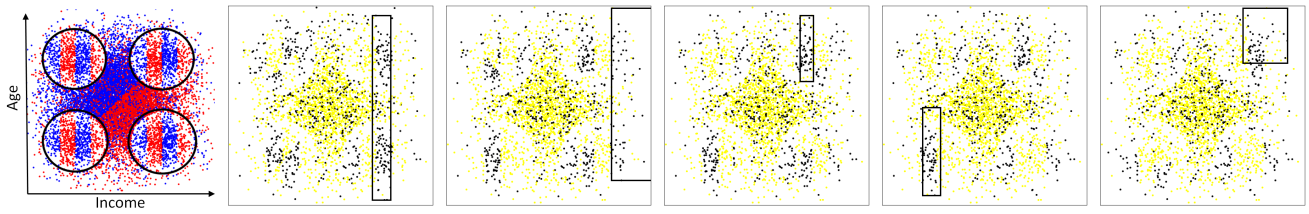
However, to apply the above algorithms, we must know the sensitive features such as race, gender, and age. In practice, this may not be possible for several reasons. People may refuse to provide this information. The sensitive features could be censored for privacy reasons. The feature values may be noisy or untrustworthy even when they are available. An example is self-reported sensitive attributes, such as in social media profiles. Finally, predefined categories may not accurately reflect social reality. For example, the US Census Bureau identifies a *White* person as a "person having origins in any of the original peoples of Europe, the Middle East, or North Africa" [5]. But society may make finer distinctions, which are lost if we assume *Whites* to be a homogeneous group. Hence, we cannot eliminate unfairness by only considering predefined categories. We need fair classifiers where the sensitive attributes are unknown.

1.1 Overview of Related Approaches

Several recent algorithms aim to solve this "fairness without demographics" problem. They optimize a weighted loss function with adaptive weights. In each iteration, they select a subset of the training data where the classification might be unfair. For the next iteration, they increase the weights of these data points in the loss function. In this way, they reduce the chance of unfairness in the resulting classifier.

Different methods consider different subsets and weighting schemes. The subsets can, for example, be the set of currently misclassified points [12, 19, 33]. But many misclassified points may be noisy points. In such cases, increasing the weight of misclassified points may hurt performance. Another approach is to use an adversary to identify the unfair subset of points. For example, the adversary can be another classifier. This classifier separates misclassified from correctly classified points in the current iteration [27]. But the pattern of unfairness may change between datasets and even between iterations. Hence, adversarial classifiers can struggle to identify all unfair subgroups, as we show empirically.

Once we identify the unfair subset, we must update their weights for the next iteration. One approach is to optimize for worst-case performance among a class of subsets [8, 33]. Such weight update schemes guarantee that after enough iterations, no subset in the class has high unfairness. However, such methods assume that we



(a) Dataset (b) Intermediate stages with high-risk zones identified by SURE

Figure 1: *Example of SURE:* (a) The circles represent minority subgroups with special class patterns (red vs. blue), hidden within a majority following a simple pattern. The classifier may underperform for the subgroups. (b) In each iteration, yellow (black) points represent correctly (incorrectly) classified points. SURE finds a box where the misclassification rate is too high.

can approximately optimize any weighted loss. This assumption may not hold for general neural networks [37].

1.2 Main Idea

We propose a new algorithm, called SURE (*Significant Unfairness Risk Elimination*), for fair classification when the sensitive attributes are unknown. Like prior work, SURE minimizes a loss function with adaptive weights. Next, we outline the main ideas of SURE and how it differs from existing methods.

In each iteration, **SURE selects one high-risk box in feature space.** The box identifies a coherent region where the classifier's error rate is too high. By focusing on this box in the next iteration(s), the optimizer can better identify local patterns. So, the resulting classifier is no longer unfair in this region of the feature space. In future iterations, SURE can select other boxes. Thus, over several iterations, SURE can fix complex patterns of unfairness.

SURE only selects boxes when the risk of unfairness is statistically significant. In other words, we are "sure" about unfairness in the box. Thus, SURE avoids the problem of seeing unfairness in randomness [7]. If no statistically significant high-risk zones exist, SURE does not update weights. Hence, in the absence of unfairness, it automatically reduces to a vanilla classifier.

Unlike existing methods, SURE does not try to cover all misclassified points in every iteration. Focusing on one box at a time helps SURE avoid noise among the misclassified points. Also, SURE increases the weights of all points within a box, not just the misclassified ones. The new weights act as a lens on one region of the feature space, helping the optimizer identify local patterns.

EXAMPLE 1. *Figure 1a shows a simulated dataset of points in 2D feature space. Each point represents a person and has a class label (red or blue). Most points follow the baseline pattern, with a diagonal split between the reds and blues. There are also four minority subgroups, marked by circles, where the pattern differs from the baseline.*

Figure 1b shows several intermediate iterations of SURE. Each plot shows the correctly classified points in yellow and the misclassified ones in black. Early on, the classifier learns the baseline pattern. The only misclassified baseline points are random noise. However, this baseline noise outnumbers the misclassified points within the subgroups. So, the optimizer does not learn the patterns within the subgroups (circles). Hence, the subgroups are at risk of unfair classification.

To get a fair classifier, SURE identifies high-risk boxes in each iteration. The misclassification rate within the boxes is significantly higher than the overall rate. Most points within a box come from one (sometimes two) subgroups. The noisy baseline points do not affect the box construction.

After each iteration, SURE increases the weights of the points in that iteration's box. This pushes the optimizer to learn the subgroup patterns. So, misclassification within the subgroups is much reduced in the later iterations. In this way, SURE achieves fair classification.

Our contributions: To summarize, SURE builds a fair classifier without knowing the sensitive attributes. To do this, it iteratively identifies boxes in feature space where the classifier's loss is higher than average, and the difference is statistically significant. Our algorithm has several useful properties:

- **Interpretability:** SURE's boxes can be easily communicated, e.g., *"African-American males between 15-25 years old"*. In fact, this is the usual way we identify subgroups facing unfairness.
- **Explainability of each step:** SURE finds boxes via repeated tests of statistical significance. Thus, all of the choices made by SURE in every step can be explained and audited. Note that this is a stronger condition than the explainability of the final classifier.
- **Robustness to noise:** SURE only selects boxes where the misclassification rate is significantly higher than the overall rate. Furthermore, it restricts the search space to boxes rather than more complex patterns. Both of these steps provide robustness against noise.
- **Fast:** The time complexity of each iteration of SURE is linear in the number of data points.
- **Parameter-free:** SURE has only one interpretable parameter. We fix it to a default value for all experiments. There is no need for cross-validation and hyperparameter tuning.

The rest of the paper is organized as follows. We discuss the details of SURE in Section 2. We empirically validate our approach in Section 3. We survey related work in Section 4 and conclude in Section 5. *The Pytorch code for SURE is available at https://github.com/deepayan12/sure.*

2 PROPOSED WORK

Suppose we have a dataset where each data point represents one person. Each person is associated with a race, gender, income level, and so on (the "sensitive" attributes). If we know the sensitive attributes, we can create the set of all sensitive subgroups

$S_{sen} \subset 2^S$. In other words, S_{sen} contains all subsets of the form "`race=African-American and gender=female`". Given a classifier, we can measure its performance on every subset. A fair classifier is one where all subsets in S_{sen} have similar performance. We want a fair classifier where the performance on the worst subset in S_{sen} is as high as possible.

Now, in our problem setting, the sensitive attributes are hidden. So, we do not know S_{sen}. Hence, to ensure fairness, we consider a broad set of subsets S_{brd} that plausibly covers S_{sen}, i.e., $S_{sen} \subseteq S_{brd}$. We train the classifier to maximize the worst-case performance over S_{brd}. This ensures that all sensitive subgroups also achieve this performance threshold.

For this approach, we must answer two questions: (a) what is a reasonable choice for S_{brd}, and (b) how can we optimize worst-case performance over S_{brd}?

2.1 Choice of S_{brd}

The set S_{brd} should be large enough to plausibly contain S_{sen}. But if it is too large, then optimizing for the worst-case over S_{brd} may yield over-conservative classifiers. For example, suppose we chose S_{brd} to be the set of all subsets $S_{brd} = 2^S$. Clearly, we have $S_{sen} \subset S_{brd}$. But for any classifier, the worst-case subset from S_{brd} consists of all misclassified points. So we can get a positive worst-case accuracy over S_{brd} only if all points are correctly classified, which is unlikely.

Hence, our chosen S_{brd} should have certain characteristics:

- **Breadth/conservativeness tradeoff:** Although we optimize for the worst subset in S_{brd}, we also care about the average performance over the entire dataset. Hence, S_{brd} should be broad enough without leading to too-conservative classifiers.
- **Tractable:** S_{brd} should allow for the tractable optimization of worst-case performance.
- **Interpretable:** The subsets in S_{brd} should be easily interpretable to a disinterested observer. In other words, these subsets should match our intuition about groups of people at risk of unfair classification results.

These conditions ensure that the fair classifier is easy to compute and also easy to explain.

We choose S_{brd} to be the cross-product of all values/intervals for all features. We first convert all categorical features into numeric features via one-hot encoding. Now, for a numeric feature i, let $\mathcal{G}_i$ be the set of all intervals over the support of i. Then, we have $S_{brd} = \{(s_1, s_2, \ldots, s_d) \mid s_i \in \mathcal{G}_i\}$, where d is the number of features.

Visually, each element of S_{brd} represents a **box in feature space**, and includes all data points within that box. Using boxes, we can select people of a particular race, or (race, gender), or (gender, age range), if we know these attributes. But since the sensitive features are hidden, we use all features in S_{brd}. For instance, some features in the dataset could be correlated with sensitive attributes. By building boxes from all features, we cover the correlated features, and hence S_{brd} plausibly covers S_{sen}. Figure 1 shows examples of boxes picked by SURE.

2.2 Optimizing the worst-case performance

SURE builds a fair classifier iteratively. In each iteration, it runs an optimizer to minimize a weighted loss for several epochs. SURE then inspects the results, resets the weights, and repeats this process. The new weights should be such that in the next iteration, the optimizer focuses on the high-risk zone. This is the set of points where the risk of unfair classification is the greatest. For example, the box "`race=African-American`" could be a high-risk zone if the classifier's accuracy for African-Americans is much worse than for Whites or Asians. More generally, the current classifier could be unfair for a combination of sensitive features (e.g., `race=African-America` *and* `gender=male` *and* `age in` $[15, 25]$). Since we do not know the sensitive features, we must consider all subsets in S_{brd}. Hence, in each iteration, the key question is:

Which subset of S_{brd} is the high-risk zone?

Since S_{brd} represents the set of boxes in feature space, we seek a high-risk box. We have two desiderata for the algorithm:

- **Explainability for every step:** We seek a box where the classifier's performance is *significantly* worse than average. Otherwise, the algorithm would be "fooled by randomness" and see unfairness in random variation [7]. The algorithm should be able to justify each step of its box choice via significance tests.
- **Speed:** Since we search for high-risk zones every iteration, the algorithm needs to be fast. Hence, an approximate but quick method is preferable to an exact computation of the worst-case box in S_{brd}. Any approximation errors can be repaired in future iterations.

To ease the exposition of SURE, we first consider the case where we have only one feature. Then we generalize to the case with multiple features. Also, we focus on the worst-case misclassification rate over S_{brd} as our performance measure. Our framework can be generalized to other measures via the appropriate modifications.

Special case (only one feature): A box in one dimension is just an interval. Algorithm 1 identifies a high-risk interval for one feature. It takes as input the set of points currently classified correctly ($X_\checkmark$) and incorrectly ($X_\times$). It splits all the points $X_\checkmark \cup X_\times$ into bins. We choose the number of bins so that each bin has $\geq n_{min}$ points from both sets. The bin boundaries are such that all bins have an equal number of points from a balanced subset of $X_\checkmark \cup X_\times$. These conditions ensure that all bins are comparable and contain enough points to detect statistically significant deviations.

Then, we find the fractions of points from $X_\checkmark$ and $X_\times$ in each bin v; call these $p_\checkmark(v)$ and $p_\times(v)$. The classifier's performance on a bin is measured by $\Delta(v) := p_\times(v) - p_\checkmark(v)$. If the classifier is entirely fair across all feature values, then $\Delta(v) = 0$.‘ A value of $\Delta(v) > 0$ may indicate underperformance in bin v, with larger values implying greater risk. However, $\Delta(v) > 0$ can also occur due to random chance. Hence, for robustness, we only consider bins where $\Delta(v)$ is statistically significantly greater than 0:

$$\Delta(v) \geq \alpha \cdot \sqrt{\frac{p_\times(v)(1-p_\times(v))}{|X_\times|} + \frac{p_\checkmark(v)(1-p_\checkmark(v))}{|X_\checkmark|}}, \qquad (1)$$

where α reflects the desired significance level for this standard hypothesis test. We set $\alpha = 2$ in all our experiments.

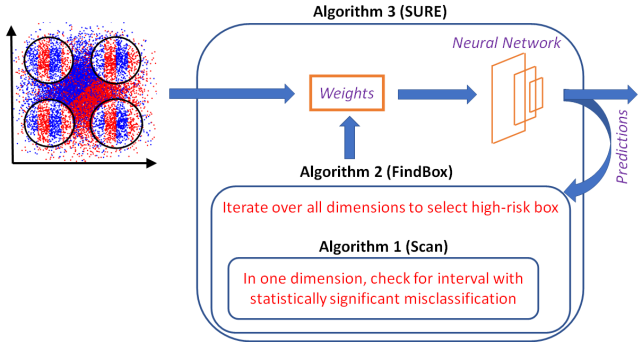


Figure 2: Overview of SURE

Using Eq. 1, we can identify the set T of bins where the classifier's performance is too poor to be explained by randomness. We find the bin $v^\star \in T$ with the largest $\Delta(v)$. This bin represents an interval where the classifier's performance is most unfair. But the bins surrounding $v^\star$ could also belong to T. In such cases, we merge $v^\star$ with other contiguous bins in T to form a larger interval V, and use V as SURE's high-risk zone.

Note that the interval V represents a tradeoff between the size and unfairness. The bin $v^\star$ is the most unfair, but it represents a small interval. At the other extreme, we could use the union of all bins in T. But this might give us disconnected intervals, which are not allowable subsets under S_{brd}. The interval V is both large enough and faces significant risk of unfairness. Hence, we choose V as the high-risk zone.

General case (multiple features): Here, we run Algorithm 1 over each feature and pick the feature $f^\star$ with the greatest total unfairness $\sum_{v\in V} \Delta(v)$. Now, we restrict our training points to those within the interval V of feature $f^\star$. Then, we iterate the process. This results in a sequence of intervals over several features. The cross-product of those intervals is the high-risk box. Algorithm 2 shows the details.

SURE (Algorithm 3) iterates over Algorithm 2 to identify high-risk boxes. Each time, it increases the weights of points within the box and then normalizes all weights. The new weights force the weighted-loss minimizer to focus on potentially unfair zones. Note that misclassifications due to noise are unlikely to form high-risk boxes. So, the weight normalization step effectively down-weights noisy points. Thus, the classifier focuses on statistically significant errors while avoiding noisy misclassifications.

Computational complexity: Algorithm 1 only scans the n data points along one feature dimension and requires $O(n)$ time. Algorithm 2 iteratively calls Algorithm 1 for each of the d available features. There can be at most d iterations, so it needs at most $O(nd^2)$ time. However, we usually need ≤ 3 iterations in practice.

REMARK 1. *SURE uses a simple reweighting scheme (step 8 of Algorithm 3). Alternative reweighting schemes can also yield optimality guarantees [8]. But they are more complex and make strong assumptions. For instance, they assume that the optimizer can approximately minimize any weighted loss. But large enough weights may lead a neural network to memorize the data, which may be undesirable [37]. Hence, we chose a more straightforward and interpretable reweighting method for SURE.*

REMARK 2. *We can interpret the significance tests of SURE as repeated hypothesis tests. The appendix presents an analysis of a simplified version of SURE. We note that one can increase α as the number*

Algorithm 1 Find best interval in one dimension

function SCAN($X_\checkmark, X_\times, n_{min}$) ▷ $X_\checkmark$ and $X_\times$ are 1D arrays
  $b \leftarrow \min(|X_\checkmark|, |X_\times|)/n_{min}$ ▷ Number of bins
  $X_{bal} \leftarrow$ balanced subset of $X_\checkmark \cup X_\times$
  $B \leftarrow$ b equal-height bins from X_{bal}
  $p_\checkmark(v), p_\times(v) \leftarrow$ normalized histogram of $X_\checkmark$ and $X_\times$ for $v \in B$
  $\Delta(v) \leftarrow p_\times(v) - p_\checkmark(v)$
  $v^\star \leftarrow \arg\max_{v\in B} \Delta(v)$ ▷ Most significant bin
  $T \leftarrow \{v \in B \mid v$ is high-risk $\}$ ▷ using Eq. 1
  if $v^\star \notin T$ **then**
    return $(\phi, 0, 0)$
  else
    $V \leftarrow$ merge bins in T surrounding bin $v^\star$
      ▷ V represents the high-risk interval
    return $(V, \sum_{v\in V} \Delta(v), \sum_{v\in T} \Delta(v))$
  end if
end function

Algorithm 2 Find high-risk box in multiple dimensions

function FINDBOX($S_\checkmark, S_\times, n_{min}$)
    ▷ $S_\checkmark$ and $S_\times$ are correctly/incorrectly classified points
  $Z \leftarrow \phi$
  repeat
    for all features $f \notin Z$ **do**
      $X_\checkmark, X_\times \leftarrow$ projection of $S_\checkmark, S_\times$ on f
      $V(f), \Delta_{grp}(f), \Delta_{tot}(f) \leftarrow$ SCAN($X_\checkmark, X_\times, n_{min}$)
    end for
    $f^\star \leftarrow \arg\max_f \Delta_{grp}(f)$
    $Z \leftarrow Z \cup \{f^\star\}$
    Remove points in $S_\checkmark, S_\times$ where the feature value for $f^\star$ falls outside the interval $V(f^\star)$
  until $f^\star = \phi$ or $\Delta_{tot}(f^\star)$ decreases from previous iteration
  return $\bigotimes_{f^\star\in Z} V(f^\star)$ ▷ box as cross-product of intervals
end function

Algorithm 3 Overall algorithm of SURE

function SURE(S, n_{min}) ▷ S=training data
  $w_i \leftarrow 1/|S|$ for all $i \in S$
  Define loss($\boldsymbol{w}$) $= \sum_{i\in S} w_i \cdot$ loss(S_i)
  repeat
    Minimize loss($\boldsymbol{w}$) via gradient steps for a few epochs
    $S_\checkmark, S_\times \leftarrow$ correctly/incorrectly classified points in S
    $Z \leftarrow$ FINDBOX($S_\checkmark, S_\times, n_{min}$)
    $w_i \leftarrow (w_i + \mathbb{1}_{i\in Z})/2$ for all $i \in S$ ▷ Re-weight points
    $w_i \leftarrow w_i / \sum_j w_j$ ▷ Normalize weights
  until convergence
end function

of features grows to compensate for repeated tests. But we keep $\alpha = 2$ for its simplicity.

3 EXPERIMENTS

We compared the performance of SURE against competing methods on simulated and real-world datasets. We also analyzed the sensitivity of SURE to its single parameter n_{min}.

3.1 Simulated Datasets

We constructed several datasets, shown in the top panel of Figure 3. In each dataset, the main pattern (the "baseline") consists of 2,500 points drawn from a Gaussian with a large variance (the large circle). These points belong to two classes (red and blue), with a diagonal separator between them. We added noise by flipping 20% of the class labels. We also added two or more subgroups of 200 points each (the small circles) following a different class pattern. These represent individuals from minority groups who are at risk of unfair classification. Finally, we included more features (not shown in the figure) where both the baseline and the subgroups follow the same distribution.

Performance measure: The baseline contains most of the points. So a classifier minimizing an overall loss will focus on the baseline pattern. Hence, it may fail to model the subgroup class patterns, leading to higher misclassification rates. So, we used the worst accuracy among all subgroups to measure the classifier's fairness.

Competing methods: We compared SURE against ARL [27] and BPF [33]. DRO [19] gave worse results than the other methods, possibly due to noise in our datasets; [27] make a similar observation. We also note that BPF is similar to [12], so we did not compare against the latter. For SURE, we set its sole parameter $n_{min} = 30$ and did not try to optimize it. For the other methods, we report results with the best parameter settings. Finally, we note that SURE always outperformed the baseline neural network. We do not show those results for clarity of exposition.

Experimental setup: All methods were run within a neural network with two hidden layers with 64 and 32 nodes each. We set the dropout rate to 0.5 in all experiments. We did not optimize the structure or parameters of the networks since our goal is an algorithm that reduces unfairness for any classifier. We repeated every experiment 30 times. All reported metrics are averaged over these repetitions.

Results comparing all methods: Figure 3 shows the worst-case accuracy over all subgroups for 2000 epochs. **SURE outperformed other methods in most settings over all epochs.** SURE did particularly well in the initial epochs (epochs$\leq$ 500), where it was always the best or close to the best.

Comparison against ARL: Among all the settings, ARL does best when we have two subgroups separated by 90 degrees (top-left panel of Figure 3). The reason is as follows. Recall that neural networks can quickly learn the simple baseline pattern. But the subgroup patterns are more difficult to learn. So the subgroups have higher misclassification rates, which makes them high-risk zones. Now, ARL finds high-risk zones using linear separators. For two subgroups 90 degrees apart, we can draw a line with both subgroups on one side and most baseline points on the other. Hence, ARL does well in this setting. But in other settings, such as with four subgroups, a linear separator between the subgroups and the baseline is not possible (see Figure 7 in Appendix A). So ARL often does worse in such cases.

We note that it is often easy to isolate any *single subgroup* with a linear separator. But ARL tries to separate *all misclassified points* from the correctly classified points. This is more challenging than isolating one subgroup. In contrast, SURE only selects one high-risk box in each iteration. In other words, it can pick one subgroup in one iteration, another subgroup in the next iteration, and so on. **By focusing only on one high-risk box at a time and iterating over such boxes, SURE can ensure fairness over complex subgroup layouts.**

Comparison against BPF: We observe that the performance of BPF sometimes dips before increasing again. This also occurs in real-world datasets, as discussed in Section 3.2. One possible reason is that BPF increases the weight for all misclassified points. Many of those misclassified points are noisy points in the baseline. Hence, the neural network has to work harder to learn the subgroup patterns. In contrast, **SURE's emphasis on statistical significance makes it robust to noise.**

Analysis of SURE's high-risk boxes: The above results showed that SURE achieves good performance over all subgroups. In Figure 4, we analyze the high-risk boxes found by SURE. For ease of exposition, we focus on the four-subgroup setting (Figure 4a). All results are over 10 repetitions of the experiment.

First, we checked if SURE's high-risk boxes consistently selected the subgroups rather than the baseline. We tagged every high-risk box found by SURE as top/bottom/left/right/baseline, depending on the dominant group among the misclassified points within that box. The baseline has the most misclassified points (due to noise). So if SURE's boxes were random, most high-risk boxes would be tagged as baseline. But if SURE works well, boxes from the baseline should be rare.

Figure 4b shows that this is indeed the case. Very few high-risk boxes are tagged as baseline. However, among the four smaller subgroups, there are significant differences in frequencies. This is due to the neural network learning some subgroup patterns more quickly than others.

Next, we checked if SURE's high-risk boxes were "pure." Ideally, a high-risk box would only contain points from a single subgroup. Figure 4c shows that in SURE's boxes, around 76% of the points belong to a single subgroup. If we only consider misclassified points in the box, 91% belong to this subgroup. Thus, SURE's boxes primarily select a single subgroup. Also, the ratio of misclassified to correctly classified points is greater than $2:1$. Hence, the chosen subgroups are at significant risk of unfair classification. By upweighting these points in the next iteration, SURE focuses the neural network on a single cohesive subgroup. The optimizer is not distracted by noisy misclassified points. This is what leads to SURE's strong performance.

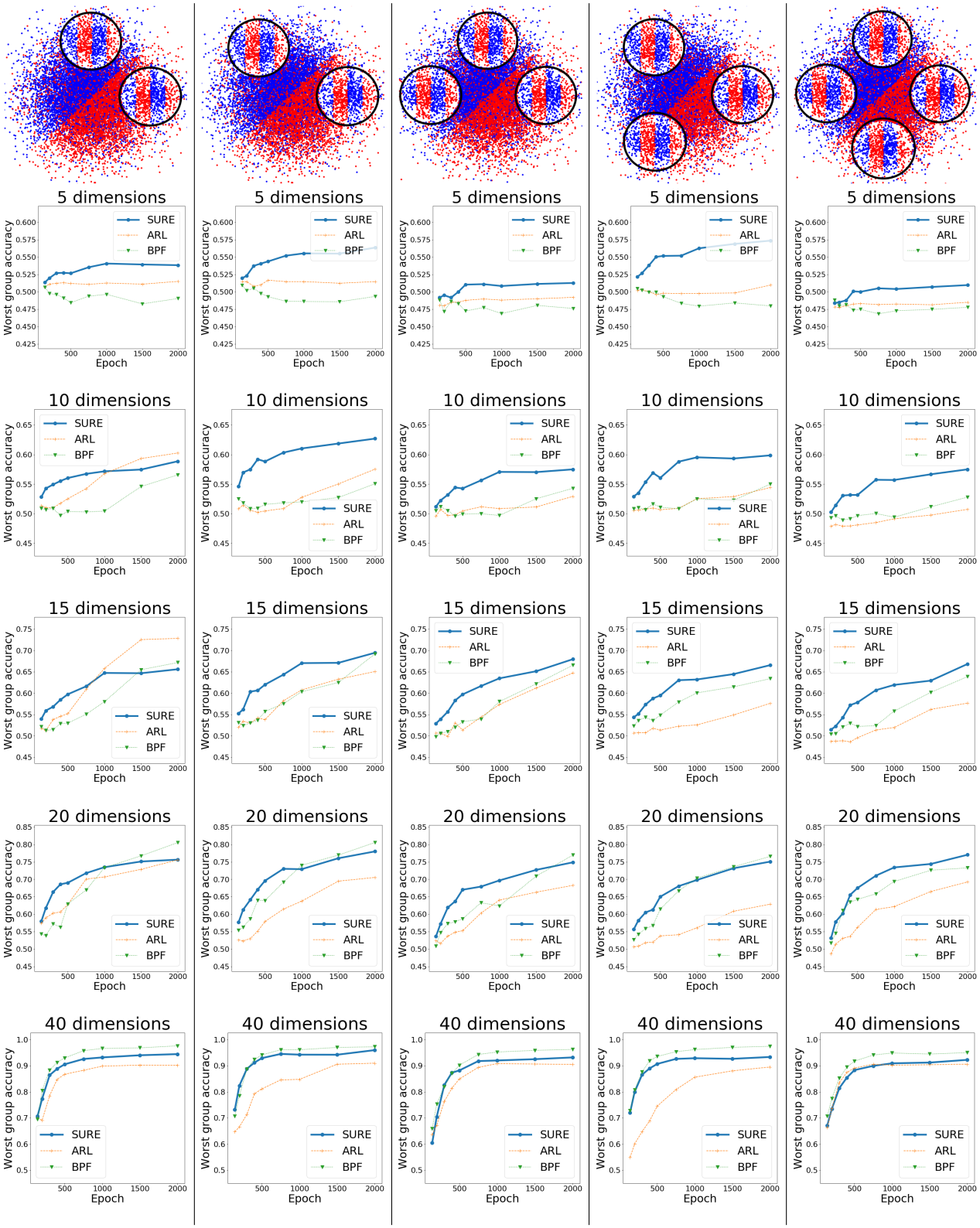
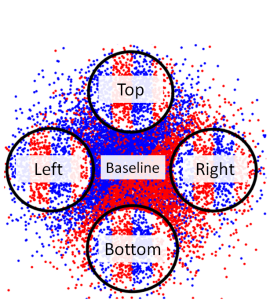
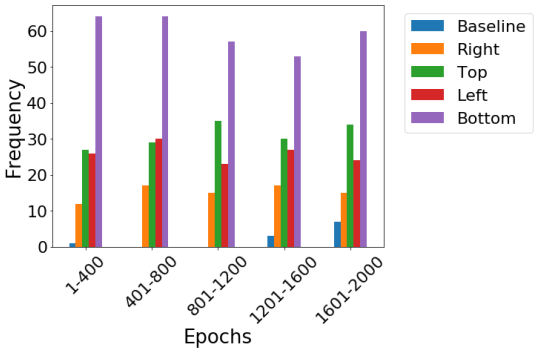


Figure 3: Performance on simulated datasets.



(a) Four subgroups

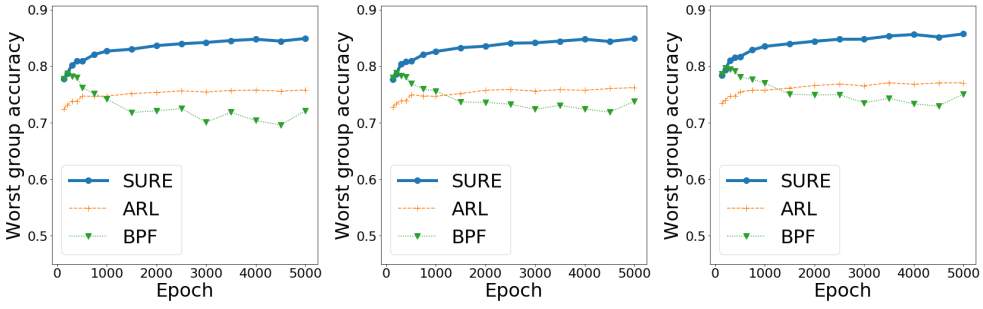


(b) Subgroup frequencies in high-risk boxes

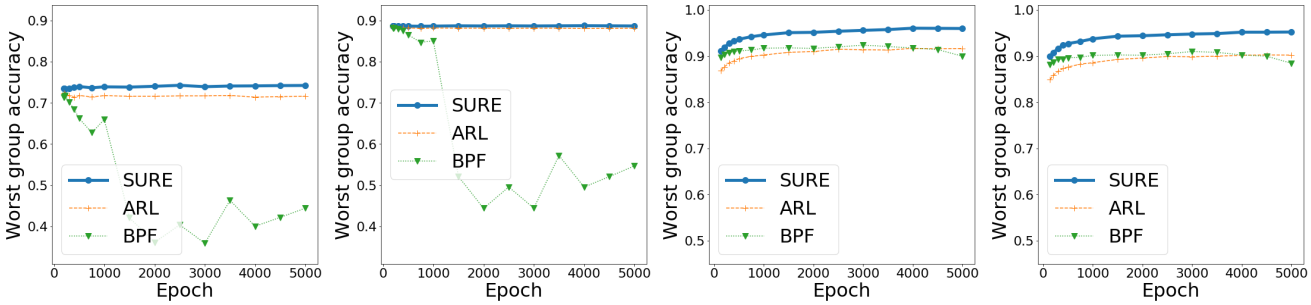
	Main subgroup	Other subgroups & baseline
Incorrectly classified	52%	5%
Correctly classified	24%	19%

(c) Confusion matrix of points in high-risk box

Figure 4: Analysis of the high-risk boxes found by SURE.



(a) COMPAS Age category (b) COMPAS Race (c) COMPAS Gender



(d) LSAC Race (e) LSAC Gender (f) Adult Race (g) Adult Gender

Figure 5: *Performance on real-world datasets.*

3.2 Real-world Datasets

We also compared all competing methods on three real-world datasets frequently used in fairness research. The goal for the *COMPAS* dataset is to predict recidivism [2]. The sensitive features are the person's age (binned into categories), race, and gender. For the *LSAC* dataset, the goal is to predict law school admissions [39]. The sensitive features are race and gender. For the *Adult* dataset, the goal is to distinguish between low-income and high-income earners [25]. The sensitive features are race and gender. All datasets were balanced by subsampling the majority class.

Experimental setup: In our experiments, we provided the sensitive features alongside other features to all algorithms. However, the sensitive features were not identified. As with the simulation experiments, we used a neural network with two hidden layers with 64 and 32 nodes as the classifier. The dropout rate was set to 0, the learning rate to 0.01, and the batch size to 200 points. We repeated each experiment 30 times with different subsets of the datasets. All reported metrics are averaged over these repetitions.

Competing methods: We again compared SURE against ARL and BPF. Note that our goal is to reduce unfairness despite not knowing the subsets of people belonging to each race, gender, or age category. Hence, we cannot expect to use cross-validation to pick the parameters that give the best performance on these subsets. So, for ARL and BPF, we used the best settings found with the

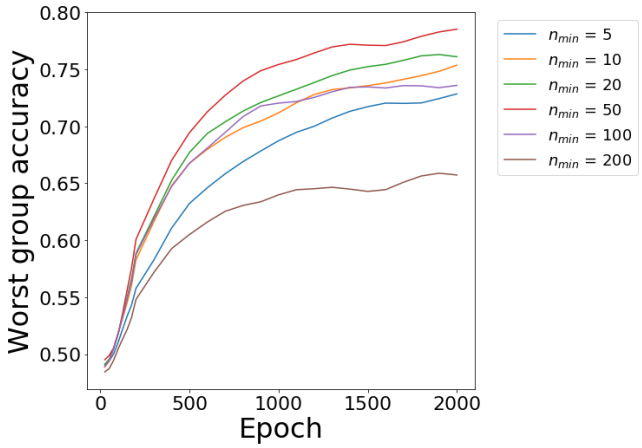


Figure 6: Sensitivity of SURE to the minimum number of points per bin (n_{min}).

simulated data. For SURE, we used the default setting of $n_{min} = 30$ everywhere, as in the simulation experiments.

Results: We find that **SURE outperforms competing methods across all settings.** The standard deviation of worst-group accuracy, averaged over all epochs, is around 0.02 for Compas, and 0.01 for LSAC and Adult datasets. We typically outperform the other methods by a wider margin. We also observe that BPF's performance dips for LSAC before starting to increase again. Both observations mirror our results on the simulated datasets. BPF's dip may be because it upweights all misclassified points, many of which may be just noisy points. ARL does not show a similar pattern because it finds a hyperplane with high-risk points on one side. However, it still does not check for statistical significance. SURE finds high-risk boxes, and achieves robustness against noise by checking for statistical significance. This enables SURE to perform well even without knowing the sensitive attributes.

The features picked by SURE's boxes are easily explainable. For example, on the Compas dataset, SURE picked several sensitive age- and gender-related features. Examples include "age<25," "age between 25-45," "sex is Female," "charge degree for Males," and "charge degree for Females." However, other features are also sometimes used, such as "arrest but no charge."

3.3 Sensitivity Analysis

SURE has a single parameter n_{min}, which is the minimum number of correctly/incorrectly classified points in each bin. We use n_{min} to determine the number of bins and their boundaries. These bins, in turn, are the building blocks for the high-risk boxes. Thus, the choice of n_{min} affects how SURE searches for the high-risk zones.

Higher values of n_{min} mean more data in each bin. So, we can reliably estimate the difference between the proportions of correctly and incorrectly classified points. Hence, identifying risky boxes becomes easier. But higher values of n_{min} also lead to larger bins. So SURE may miss narrow boxes with high misclassification rates. In other words, SURE becomes too inflexible. Hence, the choice of n_{min} must be neither too small nor too large.

Figure 6 shows the variation in SURE's performance when we vary n_{min} for the simulated dataset of Figure 4a. The results confirm the tradeoff between robustness and flexibility. When n_{min} increases from 5 to 50, the performance steadily increases. But increasing n_{min} further hurts performance. Our default choice of $n_{min} = 30$ thus performs well while still allowing us to find narrow high-risk boxes.

3.4 Running Time

The running time of SURE depends on the number of data points and features. It varies from 0.01 seconds per 1,000 points for LSAC (14 features) to 0.8 seconds per 1,000 points for Compas (452 features). This is a small fraction of the time spent training the neural network in all cases.

4 RELATED WORK

There is significant work on both the definitions of fairness that should be used and the algorithms to achieve them. We discuss these questions below, and point the reader to surveys for more details [4, 35].

4.1 Definitions of Fairness

Suppose we have a dataset where each data point represents a person. For each person, we have some features representing special attributes of that person, such as race, gender, ethnicity, and so on. These attributes are often called sensitive or protected attributes. Our goal is a machine learning system whose output is fair with respect to these attributes.

There are several extant definitions of fairness. The simplest is *fairness through unawareness.* This posits that to achieve fairness, algorithms should simply not use any of the sensitive attributes as features. While intuitive, it has severe deficiencies. For example, the algorithm could end up relying on feature combinations that are highly correlated with the sensitive attributes. Hence, this definition is rarely used.

One common choice is *group fairness*, where minority groups receive equal treatment as other groups [4, 18]. Thus, for instance, group fairness requires that the performance of a classifier on a minority group (say, Hispanics) be comparable to the performance over other groups. However, "equal treatment" may be defined in a variety of ways. Let Y and $\hat{Y}$ denote the actual and predicted binary class labels, and let $S \in \{0, 1\}$ be the sensitive feature. *Demographic parity* requires that the probability of belonging to the positive class be same for sensitive group members and non-members: $P(\hat{Y} \mid S = 1) = P(\hat{Y} \mid S = 0)$. *Equalized odds* requires that the fraction of true positives and false positive be the same for members and non-members: $P(\hat{Y} = 1 \mid S = 0, Y = y) = P(\hat{Y} = 1 \mid S = 1, Y = y)$ for $y \in \{0, 1\}$. *Equality of opportunity* only requires equality of the fraction of true positives:$P(\hat{Y} = 1 \mid S = 0, Y = 1) = P(\hat{Y} = 1 \mid S = 1, Y = 1)$. *Separation* requires the predicted score to be independent of the sensitive attribute given the class label [29]. *Multicalibration* requires that for each predicted score and every subgroup from a given class, the expectation of the true class labels over that subgroup is close to the score [20].

However, there are tradeoffs between different group-based fairness measures. For instance, we cannot have equality of both the

false positive rate and the false negative rate for all sensitive groups, except in special cases [10, 24]. The need for fairness may also compete against the algorithm's utility [17] or robustness [28]. Furthermore, when applied to defendants awaiting trial, fairness constraints may worsen public safety [11].

There are several definitions of fairness apart from group fairness. In *individual fairness*, any two individuals with similar attributes should have similar outcomes/predictions [14]. Formally, the goal is to maximize some utility (say, the overall performance of the classifier) subject to a Lipschitz constraint on the maximum allowable difference in performance for similar individuals. However, the similarity measure is exogenous, and must be chosen for each application (however, see [3]). Simple similarity measures such as Euclidean distance may be problematic in high-dimensional datasets where each individual has many attributes.

In *counterfactual fairness*, the outcome for any individual remains the same if her sensitive attributes are changed [26]. This notion of fairness closely follow intuitive notions of causality. However, strong assumptions may be needed regarding how the sensitive attributes are connected to other features and to the class labels.

Finally, the above definitions can be weakened to notions of approximate fairness. These have been shown to be related to financial risk measures [40].

4.2 Algorithms for Fair Machine Learning

There have been many algorithms developed to do fair machine learning. Examples include fair algorithms for clustering [9], influence maximization [16], graph neural networks [13], graph mining [23], and community detection [34], among others.

Many approaches consider fairness as constraints placed on utility-maximizing algorithms. For example, for classifiers, the utility is the negative of the loss. Fair solutions can be encouraged via regularization [22] or by using randomized classifiers [1]. Other work uses standard classifiers but achieves fairness by post-processing their results [36]. Madras et al. [32] infer causal relationships between variables, and use these to encourage fairness. Another popular approach is to use an adversarial system, where the adversary exploits failures in fairness. Adversarial ideas have been used for algorithms for fair representation learning [15, 31] and autoencoders [30].

The aforementioned algorithms either assume that the groups are known or apply individual fairness considerations. In our work, the groups are unknown. In such cases, existing methods typically try to identify unfair regions of the feature space and re-optimize the classifier to improve its fairness. Lahoti et al. [27] use a linear classifier to identify such regions. Another approach is to increase the weights for all points where the fairness-related loss is too high [12, 33]. This is also the intuition behind distributionally robust optimization methods [19]. We show in our experiments that SURE outperforms [27] and [33], which in turn have been shown to be better than distributionally robust optimization.

There is also related work on problem settings where the sensitive attributes are noisy or uncertain. Jalal et al. [21] consider fair image reconstruction, and show that reconstruction via sampling from the posterior distribution is the only method that achieves a notion of fairness called Conditional Proportional Representation. Celis et al. [6] consider fairness under noisy sensitive attributes, but the form of the noise must be known a priori or estimated from a validation set containing both the clean and noisy labels. There is also work on robust optimization for noisy sensitive attributes [38]. However, our problem setting is different in that the sensitive attributes are not provided at all.

5 CONCLUSIONS

Fair machine learning requires all sensitive subgroups to be treated equally. But the sensitive subgroups are often unknown. This may happen because the sensitive attributes are missing, are noisy, or are present but not identified as being the sensitive attributes. Even when the sensitive attributes are reported correctly, they may not accurately reflect how society treats people. SURE is aimed at fair classification without knowing the sensitive attributes.

The heart of SURE lies in identifying high-risk boxes in feature space. These are regions where the risk of unfairness is significantly higher than the rest of the data. By finding such boxes, SURE achieves several objectives. The boxes are intuitive and interpretable. The algorithm is robust to noise. Each step of the algorithm can be explained and audited. SURE is fast; its time complexity is linear in the number of data points. Finally, SURE is practically parameter-free, with the sole parameter being set to an intuitive default value.

We showed that SURE is comparable to or better than existing recent methods via experiments on simulated and real-world datasets. For our simulations, we varied the number and the locations of the subgroups, which are "planted" within a large, noisy baseline group. SURE can identify and correct errors within the subgroups in all settings. We observed similar results for the real-world datasets too.

ACKNOWLEDGMENTS

We acknowledge support from Dell, the McCombs Research Excellence Grant, and the NSF AI Institute for Foundations of Machine Learning (IFML).

REFERENCES

[1] Alekh Agarwal, Alina Beygelzimer, Miroslav Dudik, John Langford, and Hanna Wallach. 2018. A Reductions Approach to Fair Classification. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 60–69. https://proceedings.mlr.press/v80/agarwal18a.html ISSN: 2640-3498.

[2] J. Angwin, J. Larson, S. Mattu, and L. Kirchner. 2016. Machine bias: There's software used across the country to predict future criminals and it's biased against blacks. ProPublica.

[3] Yahav Bechavod, Christopher Jung, and Zhiwei Steven Wu. 2020. Metric-Free Individual Fairness in Online Learning. 12.

[4] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. 2018. Fairness in Criminal Justice Risk Assessments: The State of the Art. *Sociological Methods & Research* (2018).

[5] US Census Bureau. 2021. 2020 Census Frequently Asked Questions About Race and Ethnicity. https://www.census.gov/programs-surveys/decennial-census/decade/2020/planning-management/release/faqs-race-ethnicity.html.

[6] L. Elisa Celis, Lingxiao Huang, Vijay Keswani, and Nisheeth K. Vishnoi. 2021. Fair Classification with Noisy Protected Attributes: A Framework with Provable Guarantees. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 1349–1361. https://proceedings.mlr.press/v139/celis21a.html ISSN: 2640-3498.

[7] Irene Chen, Fredrik D. Johansson, and David Sontag. 2018. Why Is My Classifier Discriminatory? http://arxiv.org/abs/1805.12002 arXiv:1805.12002 [cs, stat].

[8] Robert Chen, Brendan Lucier, Yaron Singer, and Vasilis Syrgkanis. 2017. Robust optimization for non-convex objectives. In *Proceedings of the 31st International*

Conference on Neural Information Processing Systems (NIPS'17). Curran Associates Inc., Red Hook, NY, USA, 4708–4717.
[9] Xingyu Chen, Brandon Fain, Liang Lyu, and Kamesh Munagala. 2019. Proportionally Fair Clustering. 10.
[10] Alexandra Chouldechova. 2016. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *arXiv:1610.07524 [cs, stat]* (Oct. 2016). http://arxiv.org/abs/1610.07524 arXiv: 1610.07524.
[11] Sam Corbett-Davies, Emma Pierson, Avi Feller, Sharad Goel, and Aziz Huq. 2017. Algorithmic Decision Making and the Cost of Fairness. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Halifax NS Canada, 797–806. https://doi.org/10.1145/3097983.3098095
[12] Emily Diana, Wesley Gill, Michael Kearns, Krishnaram Kenthapadi, and Aaron Roth. 2021. Minimax Group Fairness: Algorithms and Experiments. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. Association for Computing Machinery, New York, NY, USA, 66–76. http://doi.org/10.1145/3461702.3462523
[13] Yushun Dong, Jian Kang, Hanghang Tong, and Jundong Li. 2021. Individual Fairness for Graph Neural Networks: A Ranking based Approach. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. ACM, Virtual Event Singapore, 300–310. https://doi.org/10.1145/3447548.3467266
[14] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Rich Zemel. 2011. Fairness Through Awareness. In *ITCS*.
[15] Harrison Edwards and Amos Storkey. 2016. Censoring Representations with an Adversary. In *ICLR*. http://arxiv.org/abs/1511.05897 arXiv: 1511.05897.
[16] Golnoosh Farnad, Behrouz Babaki, and Michel Gendreau. 2020. A Unifying Framework for Fairness-Aware Influence Maximization. In *Companion Proceedings of the Web Conference 2020*. ACM, Taipei Taiwan, 714–722. https://doi.org/10.1145/3366424.3383555
[17] Michael Feldman, Sorelle A. Friedler, John Moeller, Carlos Scheidegger, and Suresh Venkatasubramanian. 2015. Certifying and Removing Disparate Impact. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, Sydney NSW Australia, 259–268. https://doi.org/10.1145/2783258.2783311
[18] Moritz Hardt, Eric Price, and Nathan Srebro. 2016. Equality of Opportunity in Supervised Learning. In *NeurIPS*.
[19] Tatsunori B Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. 2018. Fairness Without Demographics in Repeated Loss Minimization. 10.
[20] Ursula Hebert-Johnson, Michael Kim, Omer Reingold, and Guy Rothblum. 2018. Multicalibration: Calibration for the (Computationally-Identifiable) Masses. In *Proceedings of the 35th International Conference on Machine Learning*. PMLR, 1939–1948. https://proceedings.mlr.press/v80/hebert-johnson18a.html ISSN: 2640-3498.
[21] Ajil Jalal, Sushrut Karmalkar, Jessica Hoffmann, Alexandros G Dimakis, and Eric Price. 2021. Fairness for Image Generation with Uncertain Sensitive Attributes. 12.
[22] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. 2012. Fairness-Aware Classifier with Prejudice Remover Regularizer. In *Machine Learning and Knowledge Discovery in Databases (Lecture Notes in Computer Science)*, Peter A. Flach, Tijl De Bie, and Nello Cristianini (Eds.). Springer, Berlin, Heidelberg, 35–50. https://doi.org/10.1007/978-3-642-33486-3_3
[23] Jian Kang, Jingrui He, Ross Maciejewski, and Hanghang Tong. 2020. InFoRM: Individual Fairness on Graph Mining. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM, Virtual Event CA USA, 379–389. https://doi.org/10.1145/3394486.3403080
[24] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. 2016. Inherent Trade-Offs in the Fair Determination of Risk Scores. *arXiv:1609.05807 [cs, stat]* (Nov. 2016). http://arxiv.org/abs/1609.05807 arXiv: 1609.05807.
[25] Ronny Kohavi and Barry Becker. 1996. Adult data set. http://archive.ics.uci.edu/ml/datasets/Adult.
[26] Matt J. Kusner, Joshua R. Loftus, Chris Russell, and Ricardo Silva. 2018. Counterfactual Fairness. In *NeurIPS*.
[27] Preethi Lahoti, Alex Beutel, Jilin Chen, Kang Lee, Flavien Prost, Nithum Thain, Xuezhi Wang, and Ed H Chi. 2020. Fairness without Demographics through Adversarially Reweighted Learning. 13.
[28] Tian Li, Shengyuan Hu, Ahmad Beirami, and Virginia Smith. 2021. Ditto: Fair and Robust Federated Learning Through Personalization. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 6357–6368. https://proceedings.mlr.press/v139/li21h.html ISSN: 2640-3498.
[29] Lydia T. Liu, Max Simchowitz, and Moritz Hardt. 2019. The Implicit Fairness Criterion of Unconstrained Learning. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 4051–4060. https://proceedings.mlr.press/v97/liu19f.html ISSN: 2640-3498.
[30] Christos Louizos, Kevin Swersky, Yujia Li, Max Welling, and Richard Zemel. 2016. The Variational Fair Autoencoder. http://arxiv.org/abs/1511.00830 arXiv: 1511.00830.
[31] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. 2018. Learning Adversarially Fair and Transferable Representations. In *ICML*. http://arxiv.org/abs/1802.06309 arXiv: 1802.06309.
[32] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. 2019. Fairness through Causal Awareness: Learning Causal Latent-Variable Models for Biased Data. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*. ACM, Atlanta GA USA, 349–358. https://doi.org/10.1145/3287560.3287564
[33] Natalia L. Martinez, Martin A. Bertran, Afroditi Papadaki, Miguel Rodrigues, and Guillermo Sapiro. 2021. Blind Pareto Fairness and Subgroup Robustness. In *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 7492–7501. https://proceedings.mlr.press/v139/martinez21a.html ISSN: 2640-3498.
[34] Ninareh Mehrabi, Fred Morstatter, Nanyun Peng, and Aram Galstyan. 2019. Debiasing community detection: the importance of lowly connected nodes. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM '19)*. Association for Computing Machinery, New York, NY, USA, 509–512. https://doi.org/10.1145/3341161.3342915
[35] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A Survey on Bias and Fairness in Machine Learning. *Comput. Surveys* 54, 6 (July 2021), 115:1–115:35. https://doi.org/10.1145/3457607
[36] Felix Petersen, Debarghya Mukherjee, Yuekai Sun, and Mikhail Yurochkin. 2021. Post-processing for Individual Fairness. https://arxiv.org/abs/2110.13796v1
[37] Shiori Sagawa, Aditi Raghunathan, Pang Wei Koh, and Percy Liang. 2020. An Investigation of Why Overparameterization Exacerbates Spurious Correlations. In *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 8346–8356. https://proceedings.mlr.press/v119/sagawa20a.html ISSN: 2640-3498.
[38] Serena Wang, Harikrishna Narasimhan, Maya Gupta, Wenshuo Guo, Andrew Cotter, and Michael I Jordan. 2020. Robust Optimization for Fairness with Noisy Protected Groups. 14.
[39] L. F. Wightman. 1998. LSAC National Longitudinal Bar Passage Study. LSAC Research Report Series.
[40] Robert Williamson and Aditya Menon. 2019. Fairness risk measures. In *Proceedings of the 36th International Conference on Machine Learning*. PMLR, 6786–6797. https://proceedings.mlr.press/v97/williamson19a.html ISSN: 2640-3498.

A EXAMPLE HIGH-RISK ZONES BY ARL

Figure 7 shows examples of high-risk zones found by ARL. The figures show correctly classified points in yellow and incorrectly classified ones in black. ARL uses a linear separator to isolate high-risk zones. We find that these zones isolate the bottom subgroup but also include many other points outside this subgroup. Many points in the isolated zone are noisy points from the baseline. Hence, the optimizer of the weighted loss receives a weaker signal than Sure.

B ANALYSIS

Consider a problem setting with d features, b intervals per feature, for b^d bins total. We denote each bin by an index from the set $I = [b]^d$. Each bin $i \in I$ has a baseline misclassification rate of $q_\times(i)$. We assume that in one bin, the misclassification rate is increased by δ. Thus, this bin has higher misclassifications than the baseline rate. Without loss of generality, let this special bin have index $\mathbf{1} \in I$ (i.e., the first interval for each feature). Now, we split n data points equally among the b^d bins. For each point in bin $i \in I$, we independently draw a $Bernoulli(q_\times(i))$ (or $q_\times(\mathbf{1}) + \delta$ for bin $\mathbf{1}$) to label the point as "misclassified" or "correctly classified". Let the observed fraction of misclassified points in bin i be $\hat{q}_\times(i)$. We want to find conditions under which bin $\mathbf{1}$ will be identified by Sure.

For any two indices $a \in I$ and $b \in I$, we say that $a \sim_k b$ if $a(1) = b(1), a(2) = b(2), \ldots, a(k) = b(k)$, where $a(i)$ is the i^{th} component of index a (i.e., the interval for the i^{th} feature). Thus, the subset of indices that are pairwise related via $\sim_k$ forms an equivalence class. Intuitively, if I is a hypercube in d dimensions, then this equivalence class of indices represents a slice from that hypercube. For any slice S, we define $q_\times(S) = \sum_{i \in S} q_\times(i) + \delta \cdot \mathbb{1}_{\mathbf{1} \in S}$ and $\hat{q}_\times(S) = \sum_{i \in S} \hat{q}_\times(i)$.

We analyze a simplified version of Sure, where it picks the best single interval for first feature, then the best single interval

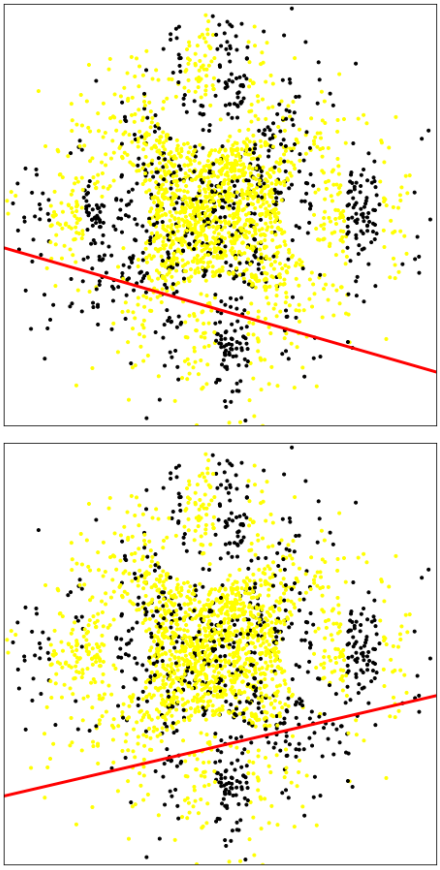


Figure 7: Examples of linear separators found by ARL in intermediate steps for the four-subgroup setting of Figure 4a.

for the second feature within this slice, and so on. Let S_{k-1} be the slice corresponding to features chosen in the first $k-1$ steps ($S_0 = I$). Then, in step k, we consider all slices $T_k = S_{k-1}/\sim_k$ of S_{k-1} based on different values of the k^{th} feature. For each $S \in T_k$, we compute $p_\times(S) = \hat{q}_\times(S)/(\sum_{S' \in T_k} \hat{q}_\times(S'))$ and $p_\checkmark(S) = (1 - \hat{q}_\times(S))/(\sum_{S' \in T_k} 1 - \hat{q}_\times(S'))$. We compute the difference $\Delta(S) := p_\times(S) - p_\checkmark(S)$, and pick the S with the highest value. The algorithm succeeds if, for each $k \in [d]$, the slice S_k equals $I_k^{in} := \{j \in I; j(1) = 1, j(2) = 1, \ldots, j(k) = 1\}$ (corresponding to the special index $\mathbf{1}$).

Theorem B.1. *Suppose d is fixed and $b \to \infty$ such that $M := \max_{k\in[d]} \max_{S,T \in I/\sim_k} (q_\times(S) - q_\times(T))$ is a constant and $\delta >= M + \epsilon$ for some constant $\epsilon > 0$. If $n = \omega(b^d \log(b))$ then for any k and any slice $S \in T_k$ such that $S \neq I_k^{in}$, we have $\Delta(I_k^{in}) > \Delta(S)$ with probability $1 - o(1)$.*

Proof. Since there are b^d bins, each gets n/b^d datapoints. Let $t = c \cdot \sqrt{\frac{\log b}{n/b^d}}$ for some constant $c > 0$. Note that $t = o(1)$ when $n = \omega(b^d \log(b))$. By a Chernoff bound, $\hat{q}_\times(i) = (q_\times(i) + \delta \cdot \mathbb{1}_{i=\mathbf{1}})(1+t)$ for all $i \in I$, with failure probability upper-bounded by $O(b^d \cdot \exp(-\frac{n}{b^d} \cdot t^2)) = O(1/b) = o(1)$ for c large enough. Under this condition, for slice I_k^{in} and any $S \in T_k, S \neq I_k^{in}$, we have

$$
\begin{aligned}
&p_\times(I_k^{in}) - p_\times(S) \\
&\quad = \frac{\hat{q}_\times(I_k^{in}) - \hat{q}_\times(S)}{\sum_{S \in T_k} \hat{q}_\times(S)} \\
&\quad = \frac{(\delta + q_\times(I_k^{in}))(1 + O(t)) - q_\times(S) \cdot (1 + O(t))}{(\delta + q_\times(I_k^{in}))(1 + O(t)) + \sum_{S \in T_k, S \neq I_k^{in}} q_\times(S) \cdot (1 + O(t))} \\
&\quad = \frac{\delta + q_\times(I_k^{in}) - q_\times(S)}{\delta + \sum_{S \in T_k} q_\times(S)} \cdot (1 + O(t)) \\
&\quad \leq \frac{M}{b^d} \cdot (1 + O(t)).
\end{aligned}
$$

where in the last step, we used the condition on δ for the numerator, and the fact that the denominator is at most the sum of misclassification rates (≤ 1) over at most b^d indices. Similarly,

$$
\begin{aligned}
p_\checkmark(I_k^{in}) - p_\checkmark(S) &= -\frac{\hat{q}_\times(I_k^{in}) - \hat{q}_\times(S)}{\sum_{S \in T_k} 1 - \hat{q}_\times(S)} \\
&\geq -\frac{M}{b^d} \cdot (1 + O(t)).
\end{aligned}
$$

Hence, $\Delta(I_k^{in}) - \Delta(S) \geq \frac{2M}{b^d}(1 + O(t))$. So we always choose the first interval among all available intervals when scanning over a feature. □

Thus, under the conditions of Theorem B.1, Sure selects the bin $\mathbf{1}$ whose misclassification rate is higher than all others (due to the addition of δ).