Spring 2022 CIS 523

Ethical Algorithm Design
Francesca Marini

Bias Bounties Assignment Write-Up

Note: This is the write-up for the Bias Bounty Assignment. I am turning this in with two late days used (two of my three free late days). I did this assignment alone. I conducted all of the experimentation in Google Colab. I have submitted all of the original files, but to run my code, one only needs to refer to the Colab notebook. I took the necessary code from the separate .py files and placed it in the Colab notebook in an appropriate order. To recreate my experimentation, one only needs to run the code in the Colab notebook from start to finish. I tried my best to include documentation and comments throughout the code to make it clear what I was doing. Though I put the source code in my notebook, I did not inspect it in detail. If you have any questions about how anything runs, do not hesitate to reach out to me via my Penn email. My submission includes all of the original files, the Colab notebook (which is really the only other thing that will need to be referenced), and the model I created using the groups detailed in this document.

Part 1.

1. Consider the "simple" update model. Say that you only accepts updates (g, h) where g represents at least 5 percent of the dataset, and h improves on g_t compared to the current f by at least 5 percent. After 5 rounds of updates are accepted, what is the minimum amount of improvement in error that f_5 can have compared to the initial model f_0?

   In the case of the "simple" update model, if we only accept updates where the group represents at least 5% of the dataset and the model must improve on the group by at least 5%, then we can determine how the model performance must change, at minimum. We know that the model accuracy is $1 - error$. Each time we update the model, 95% of the data has the same accuracy as before, and 5% of the data has improved by 5% (multiplying this fraction of the data by 1.05). That is, $acc_{i+1} = (0.95 * acc_i) + (1.05 * acc_i * 0.05)$. Thus, each round, we multiply the old accuracy by $(0.95 + 1.05 * 0.05)$. So the constant factor of change for the accuracy is 1.0025. This means that the accuracy must improve by at least 0.25%.

2. Consider the same update model from question 1.1. After how many rounds are you guaranteed to be unable to find updates that meet the improvement criteria?

Considering the same model, if we assume that the size of the dataset it sufficiently large that we will continually be able to make subgroups of size 5% or greater, then after $T = 2,000$ iterations, we will not be able to find any more subgroups that could possibly improve the model performance. If we take our ϵ to be 0.0025 (the minimum improvement of five rounds of updating), then T must be upper bounded by $\frac{1}{\frac{\epsilon}{5}}$. Naturally, if the size of the dataset were small enough, it could be possible that we would run out of possible subgroups to consider before iterating this amount of times, but for any reasonable, realistically large dataset size, that would not occur. This assumes that each round of updates we improve only by the worst case amount. Obviously, if our improvements fared better, there is a chance that we would exhaust all possibly improvements earlier, but it would not be until T iterations that this is guaranteed.

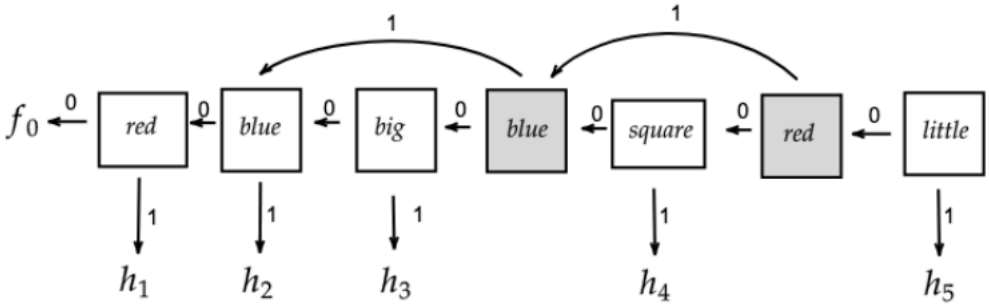

Figure 4: A PDL.

Part 2.

1. Consider the PDL in figure 4, used to classify x which have the following categorical attributes: color (blue, red, green), shape (circle, triangle, or square), and size (big or little). Which of the models $(f0, h1, ...)$ would be used to classify the following x's?

   (a) A little blue triangle
   This would be classified by h_5. The model starts at the rightmost node. If x is little, then it uses h_5 to classify x. In this case, x is little, so it is classified at that node (h_5).

   (b) A big red square
   This would be classified by h_3. The model starts at the rightmost node. We see that x is not little, so we move onto the next check. We see that x is red, so

we need to move along the repair edge to the appropriate node. That next node checks if x is blue, which it is not, so it is passed to the next node. We now check if x is big. In this case, x is big, so it is classified at that node (h_3).

(c) A big blue circle
This would be classified by h_2. The model starts at the rightmost node. We see that x is not little, so we move onto the next check. We see that x is not red, so we so we move onto the next check. We see that x is not a square, so we so we move onto the next check. That next node checks if x is blue, which it is, so we need to move along the repair edge to the appropriate node. We now check if x is blue again, which it is. In this case, x is blue, so it is classified at that node (h_2).

2. Consider the PDL in figure 4, and say that you have found a (g, h) that improves on the existing PDL, where g identifies red little squares. Do you have enough information to say whether or not there will be repairs?

   There will not be repairs in this case. There are only three metrics that we look at in this dataset: size, color, and shape. This model considers a group where all three of these values are defined. An input will either be accepted at this node if it is a little, red square, or passed along to the other nodes earlier in the model. There is no other metric where this node could be hurting model performance. If, for example there was a fourth dimension to consider: say, weight, then it could be possible for there to be repairs. Maybe this new node improves performance on heavy, little, red squares but hurts performance on light, little red squares. Then there would be need for a repair to check if an input to that node is light, in which case it would be diverted to an earlier model. However, this is not the case with our data, and there would be no need for a repair once this node is added.

Part 3.

1. Describe each of the (g, h)s that you found and the order in which they were accepted.

   My model underwent 28 successful updates from (g, h) pairs that I was able to find. Here they are listed in the order in which they were accepted:

   - Occupation: EDU
   - Occupation: CMS
   - Occupation: CMS
   - Occupation: PRT
   - Occupation: LGL

- Occupation: RPR
- Class of Worker: Local Government
- Education Level: Associate's Degree
- Education Level: Bachelor's Degree
- Occupation: SCI
- Class of Worker: Minorities (those that make up less than 5% of the dataset on their own, combined)
- Class of Worker: Non-Profit
- Class of Worker: Self-Employed
- Occupation: BUS
- Occupation: SAL
- Occupation: CON
- Working Hours: Overtime ($\geq$ 60 hours per week)
- Occupation: FIN
- Place of Birth: Oceania
- Race: Black
- Place of Birth: US and Territories
- Occupation: MIL
- Race: White
- Occupation: TRN
- Occupation: PRS
- Occupation: CLN
- Occupation: FFF
- Occupation: EAT

For occupation categories, I looked at the dataset guide to understanding what the numerical codes meant for each occupation. I noticed that occupations were grouped by industry, so instead of trying each occupation code individually, I made those groups on a per-industry basis.

All other groups should be self-explanatory. Some were created by strategically grouping numerical codes together, as with the occupation groups (such as in the case of working hours, where I grouped part-time workers < 30 hours per week, full-time workers $30 - 59$ hours, and over-time workers 60+ hours, or in the case of place of birth, where I grouped individuals by the continental region where they were born). Sometimes, if a particular group was too small in the dataset, I also combined it with

other minorities to create a larger subgroup to try (as in the cases of Native American populations or the class of worker minorities).

All models h were basic depth-10 decision trees created using the simple update method given in the notebook.

2. Provide a description of the strategies you used to identify (g, h) pairs.

   There were two main strategies that I used to identify update groups. I figured it might make sense to test all subgroups classified by a single field in the dataset. So I initially created group functions for each individual race, marital status, and sex. In the case of race, I combined the various categorizations for Native Americans, since the groups were so small. For class of worker, I similarly made groups for each type, but I combined any classes that made up less than 5% of the data sample. For age, I made groups based on the decade that the person was in. For occupation, I grouped people based on the industry that they were in, as denoted by inspecting the meanings of the occupation codes in the dataset documentation. For place of birth, I grouped American-born individuals together, and then I made other groups for people born in different continents. For schooling, I grouped those with no schooling, those with some schooling but no high school diploma, those with a high school degree but no college degree, associates, bachelors, and advanced degrees. For workplace hours, I grouped people based on what I thought was a fair metric for people who work part-time, full-time, and over-time. I thought that these intelligently formed groups might be good heuristics for assessing income. For groups with many unique values, I tried to intelligently group them together and then consider those groups in accordance with the ones that could offer the most improvement soonest, by the addition of a new model to the PDL.

   I then got the initial decision stump model's performance on each of these groups, and I ordered them based on the ones that had the poorest performance (highest error) first. I considered them in this order when I passed them to the simple update function.

   In addition, I thought that it would make sense to try and consider some more intersectional groupings of individuals. Due to time constraints, it was not possible to test all possible combinations of intersectional groups, so I just made some educated guesses about which groups of people tend to be discriminated against most and who might be most likely to receive biased results. I manually made groupd that considered men and women of different marital statuses, people of different race / sex combinations, people of different race / marital status combinations, people who worked overtime across different races (I thought that those who worked a ton of hours were probably either very poor or very high earners), and people with different race / age grouping combinations. For more granular detail of the groups, please see the notebook. Each group is given a highly informative name, so it should be clear to what demographics they are referring.

I did not consider these multifaceted groups in any particular order, but just passed them to the simple update function. I also did not use any models that were more complicated than a depth-10 decision tree in the interest of time, though I think it is likely that using some small neural nets or regression models for this problem could have further improved things.

3. Give a general description of what strategies you attempted that failed and what aspects of the challenge were most difficult.

   As seen by the groups that were actually incorporated into the PDL model by the update function, none of the more "intersectional" groups I tried ended up providing improvements to the current iteration of the model. One reason for this was that possibly the improvements conveyed by earlier models incorporated had already taken care of any improvements possible from including those groups. That is likely in this case, since the final model did improve performance on all of the groups I considered, even those which were not incorporated into the PDL. In addition, while I thought I was considering some intuitively correct groups for who might be incorrectly predicted in this dataset, it might be that I was simply not guessing at the right intersectional groupings. I tended to stick with personal demographics, but maybe the model is already very accurate and fair along those metrics, and the groups where the model performs the least fairly is actually across different industries. That might be likely, since such a large majority of improvements came by testing groups based on industry (and we are predicting on an income-related task). If I had more time, I would like to consider some intersectional groups that also incorporate the industry that the person works in. In all, my initial, more adaptive strategy of identifying single-metric groups intelligently and then considering them in decreasing order of decision stump error tended to pay off the most.

Part 4.

1. One issue with this notion of fairness is that if a group is not represented in the dataset because the dataset is not representative of that population, then we won't accept updates that improve on those groups. Why not?

   If the dataset is not representative of the population and excludes a particular group, then we will not accept any updates that improve on those groups. The updating method we are using checks to confirm that the group we wish to improve on is represented in the dataset, and if it is not, then the update fails. There is no way to actually know based on the training set whether the group and function proposed improve the model's performance on that group because the group is not present in the training set. To say it another way, if a particular group is not in the training data, then we have no

instances in the training data on which to compare the previous model's performance with the hypothesized model's performance; so even if on the general population, the new model would improve, there is no way for the updating function to ascertain this.

In addition, if the group is present in the training data but not in the validation data, then it may also be rejected by the updating method since the updating method confirms that the group also improves on the validation set in order to avoid overfitting to the training data. This is why it is important to have both a training and validation set that are as representative as possible of the actual population.

2. Do the other notions of fairness that we discussed in class have this same problem? Why or why not?

   The other notions of fairness discussed in class tend to also have this issue. If the training / validation data does not contain an instance, then it will be impossible to improve performance on that instance. If we were to attempt to consider individual fairness, rather than the group fairness metric we are currently using, we see that if a particular individual does not have similar representative individuals in the training / validation dataset, it is unlikely that a model would behave much better on that individual, and there would be no notion of performance improvement, since that could not be measured.

   In addition, when we consider different ways of ascertaining group fairness, we see that we still need to have a group represented in the data, since the model learns from the training data (with help from the validation data to avoid overfitting). If some type of person is not in the data, there is no way for the model to know whether or not it is performing poorly, regardless of the definition of fairness we are using. The only way that this could possibly be avoided (and this would more than anything be a result of luck) would be if a demographic was not represented but it behaved similarly across all other metrics to the individuals in the training data. The model might treat similar individuals similarly, across all other metrics, and this could work out (but then there would be no need to update or improve the current model because it already does well on those individuals); otherwise, we would arrive at the same problem.

3. Say that in addition to letting bounty hunters submit possible (g, h) pairs for consideration, we let them submit gs that are not well represented in our dataset, and a new dataset D where that g is well represented. Could we incorporate that new data into our model? How? What guarantees would we have in this setting? What makes this a difficult problem?

   It would be difficult to incorporate that data into our model because including it could change the distributions of other demographics within the population. For example, say our current training dataset had an accurate distribution of the races in

our population, but it was missing representation of people with disabilities. The new dataset might provide an accurate representation of the disabled population, but that dataset could heavily skew to one particular race, where including it would no longer make the racial distribution in the training data representative.

If we were positive that the new dataset was an accurate sample of the population, then another thing that we could do is train the model from scratch on that new dataset. If the new dataset is on the whole more representative of the general population than the initial dataset, then it may make sense to re-train the entire model on D'. If both the old and new datasets are fair samples and their combination is a fair sample, than it may be okay to combine the datasets and start from scratch, or add in the new dataset and continue updating the model. However, in general, it is very difficult to know this about D and D'.

Another thing we might be able to try is train one model on D and another on D' and compare where the two differ. If there are not strong differences in performance, then it may make sense to combine the models in some fashion. If there are large differences, then it could be possible to take the parts of the models that perform best on groups, where a particular group is more representative of the population. Again, this would require a lot of in-depth analysis of the data and the models generated from it, and it would require us to have a very good sense of the population distribution as a whole. In many cases this is not possible and feasible, so incorporating D' may cause more harm than good.

Returning to our race / disability example. If we simply added D' to our current dataset and kept updating the model, the updates that occurred before the addition of D' might improve the model's performance on a particular racial group in a way that will be useful for the general population. We know that the model's performance on the group represented in D' will improve as well. However, if, following the inclusion of D', we encounter another group that involves race, we may fail to actually improve model performance on the group in a way that will benefit the population because the racial distribution is no longer representative of the population.