Assignment 2: Fairness in Classification and Representation Learning

1 Preliminaries

Assignment Objectives This assignment is more open-ended than the last one. We're giving you less helper code and fewer explicit directions on how to implement things. This is closer to the research process. In particular, getting comfortable with training classifiers from scratch and making modifications is important for machine learning research. We recommend you spend some time planning the code you'll write for this assignment before you start.

In this question, we'll train some binary classifiers on a dataset. Let Y be the class label in the dataset, A be a sensitive attribute which we are interested in fairness with respect to, and $\hat{Y}$ the classifier prediction. All of Y, A, and $\hat{Y}$ are binary (either 0 or 1). Sometimes we will be interested in predicting A rather than Y. In this case, our classifier prediction will be $\hat{A}$.

We'll need the following three metrics to analyze our classifier performance. Let n be the number of examples we evaluate our classifier on, and $n_{A=0}, n_{A=1}$ are the number of examples with $A = 0$ or 1 respectively. First, define the *accuracy* $\mathcal{A}$ as:

$$\mathcal{A} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}[\hat{Y} = Y], \tag{1}$$

where $\mathbb{1}$ is the indicator function (equal to 1 if the statement inside is true, 0 otherwise). If we are interested in predicting A rather than Y, we can define this metric analogously (with $\hat{A}$ and A).

Next, define the *reweighted accuracy* ($\mathcal{R}$) of a classifier as the mean accuracy normalized by the size of the two groups:

$$\mathcal{R} = \frac{1}{2}\Big(\frac{1}{n_{A=0}}\sum_{i=1}^{n} \mathbb{1}[\hat{Y} = Y, A = 0] + \frac{1}{n_{A=1}}\sum_{i=1}^{n} \mathbb{1}[\hat{Y} = Y, A = 1]\Big) \tag{2}$$

If we are interested in predicting A rather than Y, we can define this metric analogously as

$$\mathcal{R} = \frac{1}{2}\Big(\frac{1}{n_{A=0}}\sum_{i=1}^{n} \mathbb{1}[\hat{A} = A, A = 0] + \frac{1}{n_{A=1}}\sum_{i=1}^{n} \mathbb{1}[\hat{A} = A, A = 1]\Big) \tag{3}$$

Finally, a fairness metric Δ_{DP}, which measures the demographic parity (DP) of the classifier:

$$\Delta_{DP} = \Big|\frac{1}{n_{A=0}}\sum_{i=1}^{n} \hat{Y}\cdot(1 - A) - \frac{1}{n_{A=1}}\sum_{i=1}^{n} \hat{Y}\cdot A\Big| \tag{4}$$

This is a lot of notation, but the concepts are fairly simple: accuracy measures how often the classifier is correct; reweighted accuracy measures how often the classifier is correct if we weight each group equally, and Δ_{DP} measures the absolute difference in predictions between the two groups.

Note that Δ_{DP} and $\mathcal{R}$ need to both be calculated as functions of the entire dataset, rather than averages over minibatches.

2 Classification

1. Let's start by looking at our data. In this assignment, we'll use the Adult dataset[1], which is a classic machine learning dataset, using data from a US census. The label (Y) we are trying to predict is income, which is binarized to two categories. The sensitive attribute (A) we are concerned about fairness with respect to is gender (male or female). You can find more info about this data in a README in the assignment folder.

   Feel free to sample half or a quarter of the dataset for this assignment if memory issues arise.

   Let's look just at the training set for now. Name the 10 features which are most correlated with Y, and the 10 which are most correlated with A, as measured by (absolute) Pearson correlation (ignore any NaN correlations you see.

2. Now let's train a binary classifier to predict Y. There should be a training set and test set specified in the assignment folder. You may create a validation set as well if you wish, by splitting it out from the training set. Feel free to use any type of classifier you'd like (logistic regression, random forest, neural network, etc.). Note you'll need a neural network for the final question, so it may be faster to use one here as well, but you don't have to. We'll use $\hat{Y}$ to denote the binary prediction of the classifier (0 or 1).

   Report classification accuracy and Δ_{DP} on the test set for your trained classifier. Remove the 10 attributes that you identified in the first part as being most highly correlated with A, and retrain. Report accuracy and Δ_{DP} on this retrained classifier. Which sensitive group has higher values of $\hat{Y}$, on average?

3. Let's take a look at how the features in our data correlate with the learned predictor $\hat{Y}$. Which three features in the data are most correlated with $\hat{Y}$? Which three features are most correlated with $\hat{Y}$, only looking at examples where $A = 0$? Which three features are most correlated with $\hat{Y}$, only looking at examples where $A = 1$?

4. Now, make a version of this data, but with the attributes called "sex_Female" and "sex_Male" removed. Train a classifier on this data, exactly as in the previous section, but with the goal of predicting A instead of Y. Report accuracy and reweighted accuracy on the test set of your trained classifier. Remove the attributes that you identified in the first part as being most highly correlated with A, and retrain. Report accuracy and reweighted accuracy on this retrained classifier.

5. So, we can predict both Y and A from the same data with reasonable accuracy. Now, we're going to theoretically connect the two classifiers we

[1]https://archive.ics.uci.edu/ml/datasets/adult

just trained. Suppose you've trained some classifier g to predict Y from X. Show that there exists a classifier h which predicts A from X with reweighted accuracy greater than or equal to $\frac{1}{2}\Delta_{DP} + \frac{1}{2}$, where Δ_{DP} is measured with respect to g.

3 Representation Learning

We saw at the end of the first question that by pre-processing the data, we can change the predictions of the learned classifier. There are many types of pre-processing. Here, we'll look at two simple ways to pre-process the groups so that they look similar to each other — we hope this will decrease unfairness.

1. Let's start by just looking at the marginal distribution of each feature in each group ($A = 0, A = 1$). For each feature, fit a Gaussian to that feature for each group – this should give us two Gaussians (parameters μ_0, σ_0 or μ_1, σ_1) for each feature. Then, we'll use these simple distributions to pre-process the features of group $A = 0$ and 1 so they more closely match each other. For each feature x for a point in group $A = a$, let the pre-processed feature $x' = \frac{x-\mu_a}{\sigma_a}$. This pre-processing step should match the first two moments of the features of each group.

   As in the previous question, learn a classifier g to predict Y and a classifier h to predict A from this pre-processed dataset. Report the accuracy and Δ_{DP} for g and accuracy and reweighted accuracy for h. What happened?

2. Now, let's try a neural network solution. Train a neural network with at least one hidden layer as your binary classifier. Use a cross-entropy loss, and include an MMD regularizer on the final hidden layer of the network. You can find Pytorch code for MMD in mmd.py in the assignment folder.

   The regularizer should measure the MMD between the internal representation of the network for each group ($A = 0$ or 1), multiplied by a coefficient hyperparameter α. Start with $\alpha = 0.1$. Note, this is not really pre-processing, since we're learning the solution end-to-end.

   As in the previous question, learn to predict Y and A with this model, and report the same results. Which method was better?

3. Hyperparameters can take a number of useful values. Try to find the useful range of the hyperparameter α. Report this range. Plot the values of accuracy and Δ_{DP} against various α in this range.

4. We compared two methods of removing sensitive information. What do you think is the best way to compare various methods for this task?

5. Can you think of any other ways you might remove information about A from a representation?