Project 4 Machine Learning Fairness Algorithms Evaluation

LFR vs DM and DM-sen

Information systems are becoming increasingly reliant on statistical inference and learning to render all sorts of decisions such as the targeting of advertising and the issuing of bank loans. With growing use of automated decision-making, problems rise. Unfairness arises around certain attributes, such as race and gender. Such attributes, some call "sensitive attributes", some call "protected group", often lead to unfair results based on attribute group.

Today our group, group 2, is going to compare machine learning fairness algorithm performance that could handle sensitive attributes.

Our target is LFR and DM and DM-sen, which represent Learning Fair Representation Algorithm and Fairness Beyond Disparate Treatment & Disparate Impact: Learning Classification without Disparate Mistreatment Algorithm respectively.

First let me introduce you with what each algorithm does.

Learning fair representations (LFR)

LFR aims at achieving both group fairness (the proportion of members in a protected group receiving positive classification is identical to the proportion in the population as a whole), and individual fairness (similar individuals should be treated similarly).

To achieve this, LFR formulate fairness as an optimization problem of finding a good representation of the data with two competing goals:

1.encode data as well as possible

2.obfuscate information about membership in the protected group.

The method LFR uses is to create an intermediate representation of the original inputs that satisfy the above two goals. Then do further machine learning tasks based on this intermediate representation.

LFR maps each individual, represented as a data point in a given input space, to a probability distribution in a new representation space based on the following LOSS function from the paper.

$$Total\ Loss = A_z * L_z + A_x * L_x + A_y * L_y$$

where

$$L_z = \sum_{k=1}^{K} |M_k^+ - M_k^-|$$

$$L_x = \sum_{n=1}^{N}(x_n - \hat{x}_n)^2$$

$$L_y = \sum_{n=1}^{N} -y_n log\hat{y}_n - (1-y_n)log(1-\hat{y}_n)$$

A_z, A_x, A_y are hyperparameters decide trade-offs between system desired data.

While the first two terms encourage the system to encode all information in the input attributes except for those that can lead to biased decisions, the third term requires that the prediction of y is as accurate as possible.

Data is first splitted based on protected and nonprotected. After doing individual train test split based on protected and nonprotected data, concate them together to make the final training, validation and test data.

Fairness Beyond Disparate Treatment & Disparate Impact: Learning Classification without Disparate Mistreatment (DM and DM-sen)

DM and DM-sen introduced a new notion here, Disparate Mistreatment. As we all know, in order to maximize the utility of information systems (or, classifiers), their training involves minimizing the errors (or, misclassifications) over the given historical data. But it is quite possible that the optimally trained classifier makes decisions for people belonging to different social groups with different misclassification rates. This difference in misclassification rate is Disparate Mistreatment.

In order to measure disparate mistreatment, in the extent of decision boundary-based classifiers (e.g.logistic regression, SVMs), disparate mistreatment is measure using the covariance between the users' sensitive attributes and the signed distance between the feature vectors of misclassified users and the classifier decision boundary.

Then disparate mistreatment can be used in the constraints for minimizing the convex loss theta.

The final optimization problem is as following:

$$min\ \ L\left(\theta\right)$$

$$s.t.\ \ \frac{-N_1}{N}\sum_{(x,y)\in D_0} g_\theta\left(y,x\right) + \frac{N_0}{N}\sum\ _{(x,y)\in\ D_1} g_\theta\ \left(y,x\right) \leq c$$

$$\frac{-N_1}{N}\sum_{(x,y)\in D_0} g_\theta\left(y,x\right) + \frac{N_0}{N}\sum\ _{(x,y)\in\ D_1} g_\theta\ \left(y,x\right) \geq -c$$

Where D_0 and D_1 are the subsets of training dataset D taking values $z=0$ and $z=1$, respectively, $N_0\ =\ |D_0|$ and $N_1\ =\ |D_1|$

Model Performance and Comparison

In [1]:

```
import warnings
warnings.filterwarnings('ignore')
import os
```

In [2]:
```
%%capture
%run ../lib/DM_DM_sen_Model.ipynb
```

DM Logistic Regression

In [3]:
```
loss_function = "logreg"
EPS = 1e-6
cons_type = 0 # No constraint at very beginning
tau = 5.0
mu = 1.2
sensitive_attrs_to_cov_thresh = {"race": {0:{0:0, 1:0}, 1:{0:0, 1:0}, 2:{0:0, 1:0}}} #

cons_params = None
cons_params = {"cons_type": cons_type, "tau": tau, "mu": mu, "sensitive_attrs_to_cov_th
start_dm = time.time()
return_accuracy_noConstraint()
return_accuracy_FPR()
return_accuracy_FNR()
return_accuracy_allConstraints()

end_dm = time.time()
runtime_dm = (end_dm-start_dm)
print(f'runtime of the complete DM model is {np.round(runtime_dm, 2)} seconds')
```

```
== Unconstrained (original) classifier ==
{'cons_type': 0, 'tau': 5.0, 'mu': 1.2, 'sensitive_attrs_to_cov_thresh': {'race': {0:
{0: 0, 1: 0}, 1: {0: 0, 1: 0}, 2: {0: 0, 1: 0}}}}


Accuracy: 0.660
||  s  || FPR. || FNR. ||
||  0  || 0.34 || 0.32 ||
||  1  || 0.18 || 0.62 ||



== Constraints on FPR ==
{'cons_type': 1, 'tau': 5.0, 'mu': 1.2, 'sensitive_attrs_to_cov_thresh': {'race': {0:
{0: 0, 1: 0}, 1: {0: 0, 1: 0}, 2: {0: 0, 1: 0}}}}


Accuracy: 0.649
||  s  || FPR. || FNR. ||
||  0  || 0.27 || 0.41 ||
||  1  || 0.25 || 0.53 ||



== Constraints on FNR ==
{'cons_type': 2, 'tau': 5.0, 'mu': 1.2, 'sensitive_attrs_to_cov_thresh': {'race': {0:
{0: 0, 1: 0}, 1: {0: 0, 1: 0}, 2: {0: 0, 1: 0}}}}


Accuracy: 0.651
||  s  || FPR. || FNR. ||
```

```
||  0  || 0.28 || 0.39 ||
||  1  || 0.29 || 0.47 ||
```

```
== Constraints on FNR and FPR ==
{'cons_type': 4, 'tau': 5.0, 'mu': 1.2, 'sensitive_attrs_to_cov_thresh': {'race': {0:
{0: 0, 1: 0}, 1: {0: 0, 1: 0}, 2: {0: 0, 1: 0}}}}
```

```
Accuracy: 0.651
||  s  || FPR. || FNR. ||
||  0  || 0.27 || 0.41 ||
||  1  || 0.24 || 0.53 ||
```

```
runtime of the complete DM model is 5.8 seconds
```

DM Support vector machine (SVM)

In [4]:

```
loss_function = "svm"
EPS = 1e-6
cons_type = 0 # No constraint at very beginning
tau = 5.0
mu = 1.2
sensitive_attrs_to_cov_thresh = {"race": {0:{0:0, 1:0}, 1:{0:0, 1:0}, 2:{0:0, 1:0}}} #

cons_params = None
cons_params = {"cons_type": cons_type, "tau": tau, "mu": mu, "sensitive_attrs_to_cov_th
start_dm = time.time()
return_accuracy_noConstraint()
return_accuracy_FPR()
return_accuracy_FNR()
return_accuracy_allConstraints()

end_dm = time.time()
runtime_dm = (end_dm-start_dm)
print(f'runtime of the complete DM model is {np.round(runtime_dm, 2)} seconds')
```

```
== Unconstrained (original) classifier ==
{'cons_type': 0, 'tau': 5.0, 'mu': 1.2, 'sensitive_attrs_to_cov_thresh': {'race': {0:
{0: 0, 1: 0}, 1: {0: 0, 1: 0}, 2: {0: 0, 1: 0}}}}
```

```
Accuracy: 0.649
||  s  || FPR. || FNR. ||
||  0  || 0.38 || 0.30 ||
||  1  || 0.20 || 0.62 ||
```

```
== Constraints on FPR ==
{'cons_type': 1, 'tau': 5.0, 'mu': 1.2, 'sensitive_attrs_to_cov_thresh': {'race': {0:
{0: 0, 1: 0}, 1: {0: 0, 1: 0}, 2: {0: 0, 1: 0}}}}
```

```
Accuracy: 0.650
```

```
||  s  || FPR. || FNR. ||
||  0  || 0.31 || 0.37 ||
||  1  || 0.27 || 0.51 ||
```

```
== Constraints on FNR ==
{'cons_type': 2, 'tau': 5.0, 'mu': 1.2, 'sensitive_attrs_to_cov_thresh': {'race': {0:
{0: 0, 1: 0}, 1: {0: 0, 1: 0}, 2: {0: 0, 1: 0}}}}
```

```
Accuracy: 0.652
||  s  || FPR. || FNR. ||
||  0  || 0.33 || 0.35 ||
||  1  || 0.27 || 0.50 ||
```

```
== Constraints on FNR and FPR ==
{'cons_type': 4, 'tau': 5.0, 'mu': 1.2, 'sensitive_attrs_to_cov_thresh': {'race': {0:
{0: 0, 1: 0}, 1: {0: 0, 1: 0}, 2: {0: 0, 1: 0}}}}
```

```
Accuracy: 0.650
||  s  || FPR. || FNR. ||
||  0  || 0.28 || 0.40 ||
||  1  || 0.24 || 0.53 ||
```

```
runtime of the complete DM model is 3.68 seconds
```

LFR

In [5]:
```
%%capture
%run LFR.ipynb
```

In [6]:
```
final_param = np.array([0.0860446 , 0.48391339, 0.14352093, 0.2260606 , 0.06334933,
       0.62182417, 0.17559923, 0.5987066 , 0.18218406, 0.6286742 ,
       0.04404399, 0.69881137, 0.21485092, 0.46821003, 0.90281652,
       0.02003778, 0.0995888 , 0.46292028, 0.9506006 , 0.32273576,
       0.39858341, 0.62061651, 0.95609486, 0.51230091, 0.11617536,
       0.70181324, 0.21057602, 0.58888266, 0.76104689, 0.18014743,
       0.08054532, 0.00893841, 0.63332043, 0.37238256, 0.10919819,
       0.36424165, 0.44532985, 0.66402941, 0.85402702, 0.95541148,
       0.56231683, 0.18015636, 0.65636991, 0.39717764, 0.64549661,
       0.3397862 , 0.80643798, 0.16736284, 0.47670641, 0.04396053,
       0.81379983, 0.32009883, 0.51548235, 0.16167711, 0.27604056,
       0.64820925, 0.19231329, 0.20385995, 0.38391196, 0.78971249,
       0.58531676, 0.36953208, 0.31819169, 0.47968703, 0.78070219,
       0.18015894, 0.61398203, 0.88346662, 0.69913591, 0.08984385,
       0.6508417 , 0.72624455, 0.78769812, 0.18715203, 0.22588234,
       0.72355318, 0.81678137, 0.16168611, 0.85644431, 0.03415686,
       0.45084892, 0.84425949, 0.28715857, 0.09035437, 0.54641595,
       0.31267792, 0.55704861, 0.64015822, 0.65608821])
```

In [7]:
```
model_test = LFR(
    data_train,
    data_val,
    label_column,
    "race",
    1,
    K,
    0.00000001,
    0.01,
    1000
)
model_test.__name__
```

Out[7]:
```
'5 1e-08 0.01 1000'
```

In [8]:
```
# Predict

y_hat_priv = model_test.predict_with_param(test_X_plus, final_param, priv=True)
y_hat_nonpriv = model_test.predict_with_param(test_X_minus, final_param, priv=False)
print("priv error:", compute_error(y_hat_priv, test_y_plus))
print("non priv error:", compute_error(y_hat_nonpriv, test_y_minus))
print("overall error:", compute_error(np.concatenate((y_hat_nonpriv, y_hat_priv)),
                                      np.concatenate((test_y_minus, test_y_plus))))
```

```
priv error: 0.3803827751196172
non priv error: 0.542319749216301
overall error: 0.47821969696969696
```

Result Interpretation

As we see, the testing accuracy for overall LFR model is 52%, but test accuracy for nonprotected group using LFR drops to 46%. By contrast, the testing accuracy using DM and DM-sen all goes around 65%. There is an apparent decrease in accuracy between DM and DM-sen model and the LFR model. This makes sense because fairness measure "cleans" the data to achieve fairness while DM and DM-sen does not remove information from the data.

But there could also be other explainations. There could be sampling bias. There might also be label bias where some mis-labeled entities fall into a sensitive group. Or maybe the ground truth(target) itself is just unfair, fairness measure might just harm model accuracy. There are more factors to consider.