Boosting for Fairness-Aware Classification

Team 35

Julia Kudryavtseva
Xavier Aramayo Carrasco
Alisa Kalacheva
Vo Ngoc Bich Uyen
Alexander Lepinskikh

Plan

1. Motivation
2. Problem statement
3. Related work
4. Conducted experiments
5. Obtained results

Motivation

Potential discrimination and bias in machine learning-based decision-making systems based on sensitive attributes such as:

- ➢ gender
- ➢ race
- ➢ nationality
- ➢ age

we **need to improve fairness** for both protected and non-protected groups while maintaining overall classification accuracy.

Problem

The Fairness-Aware Classification problem

Poor prediction of a minor class by standard classifiers (SVM, kNN, log. reg) while the average quality can be good:

classes are equally important

⇒ results are biased towards the major class.

Related work

Overfitting + Information loss ⇒ new techniques

<u>In-processing approaches mitigate discrimination through objective function:</u>

- Faisal Kamiran, Toon Calders, and Mykola Pechenizkiy. 2010. **Discrimination aware decision tree learning.** In Data Mining (ICDM), 2010 IEEE 10th International Conference on. IEEE, 869–874.
- Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. 2012. **Fairness-aware classifer with prejudice remover regularizer**. In ECML PKDD. Springer, 35–50.

<u>Post-processing approaches change the decision boundary of a model or the prediction labels</u> :

- **(white-box approaches)** Benjamin Fish, Jeremy Kun, and Ad´ am D Lelkes. 2016. **A confidence-based ´ approach for balancing fairness and accuracy.** In Proceedings of the 2016 SIAM International Conference on Data Mining. SIAM, 144–152.
- **(black-box approaches)** Moritz Hardt, Eric Price, Nati Srebro, et al. 2016. **Equality of opportunity in supervised learning.** In NIPS. 3315–3323.

<u>Oversampling using knn or clustering</u>:

- N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, **“SMOTE: Synthetic minority over-sampling technique,”** J. Artificial Intell. Res., vol. 16, no. 1, pp. 321–357, Jan. 2002.
- T. Jo and N. Japkowicz, “**Class imbalances vs. small disjuncts,**” ACM SIGKDD Explorations Newslett., vol. 6, no. 1, pp. 40–49, Jun. 2004.

Algorithm implementation

AdaFair

Cumulative Fairness + AdaBoost

Language: > Python 3.7.0

Requirements:

- > sklearn 1.0.2
- > numpy 1.21.5

CUSBoost

Clustering + AdaBoost

Language: > Python 3.7.0

Requirements:

- > sklearn 1.0.2
- > imblearn
- > numpy 1.21.5

SMOTEBoost

SMOTE + AdaBoost

Tips: numpy vectorization

Language: > Python 3.7.0

Requirements:

- > sklearn 1.0.2
- > numpy 1.21.5

RAMOBoost

RAMO + AdaBoost

Tips: class inheritance from AdaBoostClassifier

Language: > Python 3.7.0

Requirements:

- > sklearn 1.0.2
- > numpy 1.21.5

AdaFair: theory

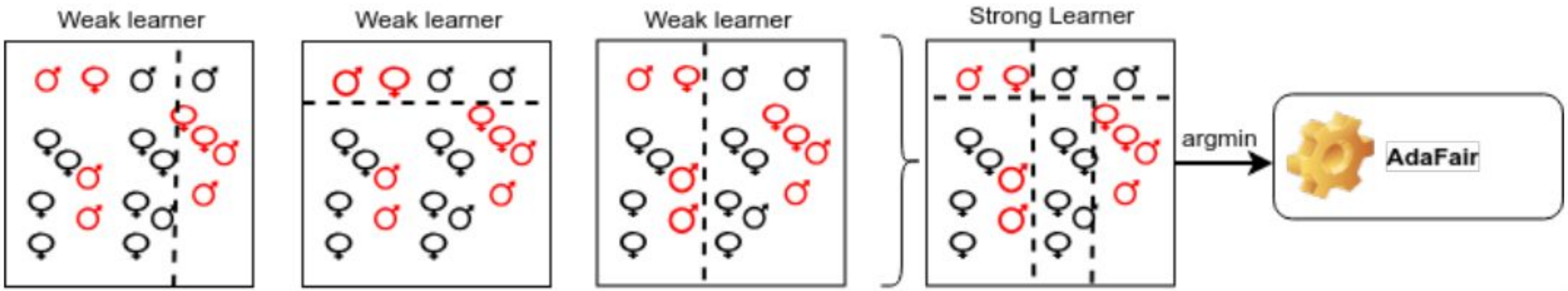
Fairness-aware classifier based on AdaBoost that addresses discrimination by optimizing ensemble members for balanced classification and incorporating a cumulative notion of fairness

Optimizing the number of **weak learners** in the final ensemble

Incorporating **fairness** in the **instance weighting process**

Using **cumulative fairness** to assess the fairness of the model up to the current boosting round

AdaBoost - a sequential ensemble method that in each round, re-weights the training data to focus on misclassified instances.



AdaFair: theory

Goal of fairness-aware classification:

Find a mapping from f(F,S) → y

\+ achieves good **predictive performance**.

\+ eliminates **discrimination**.

=> Minimize fairness measure and balanced error rate

$$\boxed{\arg\min_{\theta} \left(c \cdot BER_\theta + (1-c) \cdot ER_\theta + Eq.Odds_\theta\right)}$$

$$ER = \frac{FN + FP}{TP + TN + FN + FP} \qquad BER = 1 - \frac{1}{2} \cdot (TPR + TNR)$$

$$Eq.Odds = |\delta FPR| + |\delta FNR|$$

$$\delta FPR = P(y \neq \hat{y}|\bar{s}_-) - P(y \neq \hat{y}|s_-)$$

$$\delta FNR = P(y \neq \hat{y}|\bar{s}_+) - P(y \neq \hat{y}|s_+)$$

AdaFair pseudocode:

Input: $D = (x_i, y_i)_1^N$, T, ϵ
Output: Ensemble H

(1) Initialize $w_i = 1/N$ and $u_i = 0$, for $i = 1, 2, \ldots, N$
(2) For $j = 1$ to T:
- (a) Train a classifier h_j to the training data using weights w_i.
- (b) Compute the error rate $\mathrm{err}_j = \frac{\sum_{i=1}^N w_i I(y_i \neq h_j(x_i))}{\sum_{i=1}^N w_i}$
- (c) Compute the weight $\alpha_j = \frac{1}{2} \cdot \ln\left(\frac{1-\mathrm{err}_j}{\mathrm{err}_j}\right)$
- (d) Compute fairness-related $\delta FNR^{1:j}$
- (e) Compute fairness-related $\delta FPR^{1:j}$
- (f) Compute fairness-related costs u_i
- (g) Update the distribution as

$$w_i \leftarrow \frac{1}{Z_j} w_i \cdot e^{\alpha_j \cdot \hat{h}_j(x) \cdot \mathbb{I}(y_i \neq h_j(x_i))} \cdot (1 + u_i)$$

// Z_j is normalization factor; $\hat{h}_j$ is the confidence score

(3) Output $H(x) = \sum_{j=1}^T \alpha_i h_j(x)$

AdaFair: experimental results

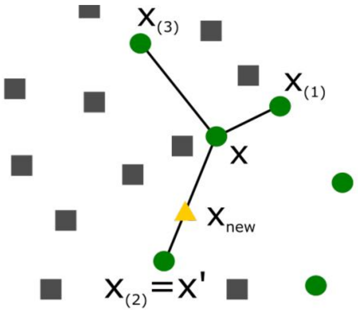
Dataset	Accuracy	Balance Accuracy	Equalized Odds
Adult Census	0.82	**0.70**	0.14
Bank	0.89	**0.70**	**0.02**
COMPASS	0.64	0.65	0.20
KDD Census	**0.94**	0.56	0.03

	Adult Census	Bank	Compass	KDD Census
#Instances	45,175	40,004	5,278	299,285
#Attributes	14	16	9	41
Sen.Attr.	Gender	Marit. Status	Gender	Gender
Class ratio (+:–)	1:3.03	1:7.57	1:1.12	1:15.11
Positive class	*>50K*	*subscription*	*recidivism*	*>50K*

Table 1: An overview of the datasets.

- AdaFair performs with good balance accuracy and low discrimination, dealing with class ratio imbalances greater than 1:3.
- Results are stable.

SMOTEBoost: theory



SMOTE + AdaBoosting = Balanced Samples

Idea: SMOTEBoost updates weights distribution: the examples from the minority class are oversampled by creating synthetic minority class examples at each iteration of Boosting

Given: Set S $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ $x_i \in X$, with labels $y_i \in Y = \{1, \ldots, C\}$, where C_p, ($C_p < C$) corresponds to a minority (positive) class.

Let $B = \{(i, y): i = 1,\ldots,m,\ y \neq y_i\}$

Initialize the distribution D_1 over the examples, such that $D_1(i) = 1/m$.

For $t = 1, 2, 3, 4, \ldots T$

1. Modify distribution D_t by creating N synthetic examples from minority class C_p using the SMOTE algorithm
2. Train a weak learner using distribution D_t
3. Compute weak hypothesis h_t: $X \times Y \rightarrow [0, 1]$
4. Compute the pseudo-loss of hypothesis h_t:
   $$\varepsilon_t = \sum_{(i,y) \in B} D_t(i,y)(1 - h_t(x_i, y_i) + h_t(x_i, y))$$
5. Set $\beta_t = \varepsilon_t / (1 - \varepsilon_t)$ and $w_t = (1/2)\cdot(1 - h_t(x_i, y) + h_t(x_i, y_i))$
6. Update D_t : $D_{t+1}(i, y) = (D_t(i, y) / Z_t) \cdot \beta_t^{w_t}$
   where Z_t is a normalization constant chosen such that D_{t+1} is a distribution.

Output the final hypothesis: $h_{fn} = \arg\max_{y \in Y} \sum_{t=1}^{T} (\log \frac{1}{\beta_t}) \cdot h_t(x, y)$

SMOTEBoost: experimental results

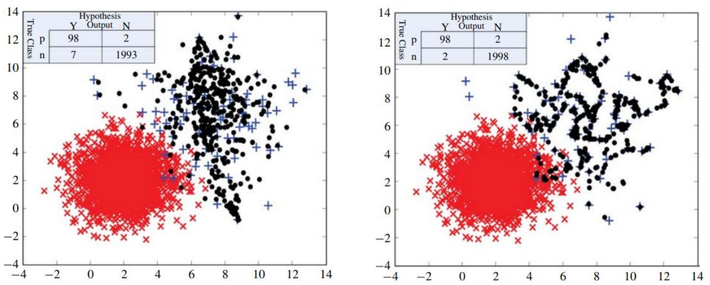
Dataset	Accuracy	Balance Accuracy	Equalized Odds
Adult Census	0.84	0.73	0.17
Bank	0.9	**0.78**	0.024
KDD Census	**0.95**	0.66	0.29
COMPASS	0.66	0.66	**0.41**

	Adult Census	Bank	Compass	KDD Census
#Instances	45,175	40,004	5,278	299,285
#Attributes	14	16	9	41
Sen.Attr.	Gender	Marit. Status	Gender	Gender
Class ratio (+:−)	1:3.03	1:7.57	1:1.12	1:15.11
Positive class	*>50K*	*subscription*	*recidivism*	*>50K*

Table 1: An overview of the datasets.

- SMOTEBoost performs with high predictive accuracy and preserves fairness even on data with class ratio imbalance 1:7.5 and 1:15 respectively.
- But these results are unstable.

RAMOBoost: theory



RAMO + AdaBoosting = Balanced Samples

Idea: RAMOBoost adaptively <u>ranks </u>minority class instances according to a sampling probability distribution that is and can adaptively shift the decision boundary toward minority class

RAMO resampling technique:

1: Sample the mislabeled training data with D_t and get back the sampled dataset Se of identical size. Slice Se into the majority subset e_1 and the minority subset e2 of size m_{lt} and m_{st}, respectively.
2: For each $x_i \in S_e$ find k_1 neighbors and calculate:

$$r_i = \frac{1}{1 + exp(-\alpha * \delta_i)}, \hat{r}_i = \frac{r_i}{\sum_{i=1}^{m_{st}} r_i} \quad d_t = \hat{r}_i. \quad i = 1, ..., m_{st}$$

3: Sample e_2 with d_t and get back a sampling minority dataset g_t, of size m_{st}.
4: For each $x_i \in g_t$ find k_2 neighbors e2 according to and use linear interpolation to generate N synthetic samples.
5: Provide the base classifier with sampling dataset Se and the N synthetic data samples.

+

Implement RAMO instead of SMOTE in SMOTEBoost

RAMOBoost: experimental results

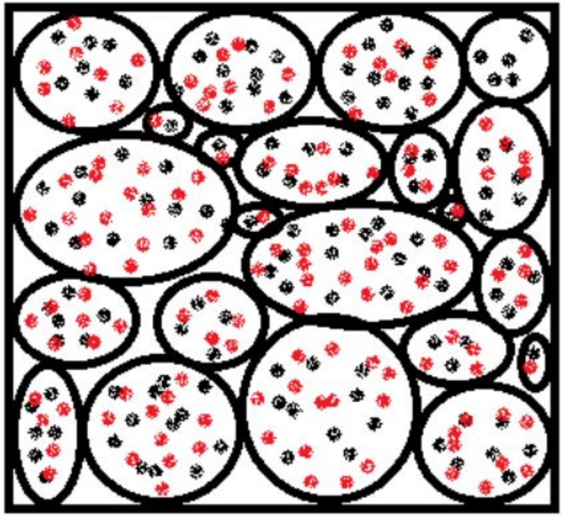
Dataset	Accuracy	Balance Accuracy	Equalized Odds
Adult Census	0.8	**0.75**	0.19
Bank	0.88	0.73	0.04
KDD Census	**0.94**	0.7	0.25
COMPASS	0.51	0.53	**0.42**

	Adult Census	Bank	Compass	KDD Census
#Instances	45,175	40,004	5,278	299,285
#Attributes	14	16	9	41
Sen.Attr.	Gender	Marit. Status	Gender	Gender
Class ratio (+:−)	1:3.03	1:7.57	1:1.12	1:15.11
Positive class	*>50K*	*subscription*	*recidivism*	*>50K*

Table 1: An overview of the datasets.

- RAMOBoost performs with high predictive accuracy dealing with imbalance more than 1:7.5.
- Results are stable.
- Shows poor fairness of a minor class representation.

CUSBoost: theory



CUSBoost is based on the combination of **cluster-based undersampling** and Adaboost algorithm.

It is similar to SMOTEBoost with the critical difference occurring in the sampling technique:

- ➢ SMOTEBoost uses SMOTE method to oversample the minority class instances
- ➢ CUSBoost uses cluster-based undersampling from the majority class using k-means (parameter optimization)

Here the red dots are selected instances from the majority class, where black and red dots are representing all the majority class instances. Better when clustering is easy.

Related work: “CUSBoost: Cluster-based Under-sampling with Boosting for Imbalanced Classification”

CUSBoost: experimental results

Dataset	Accuracy	Balance Accuracy	Equalized Odds
Adult Census	0.84	0.79	0.14
Bank	0.89	**0.80**	**0.05**
KDD Census	**0.95**	0.68	0.27
COMPASS	0.64	0.65	0.36

Algorithm 1 CUSBoost Algorithm

Input: Imbalanced data, D, number of iterations, k, and C4.5 decision tree induction algorithm.
Output: An ensemble model.
Method:

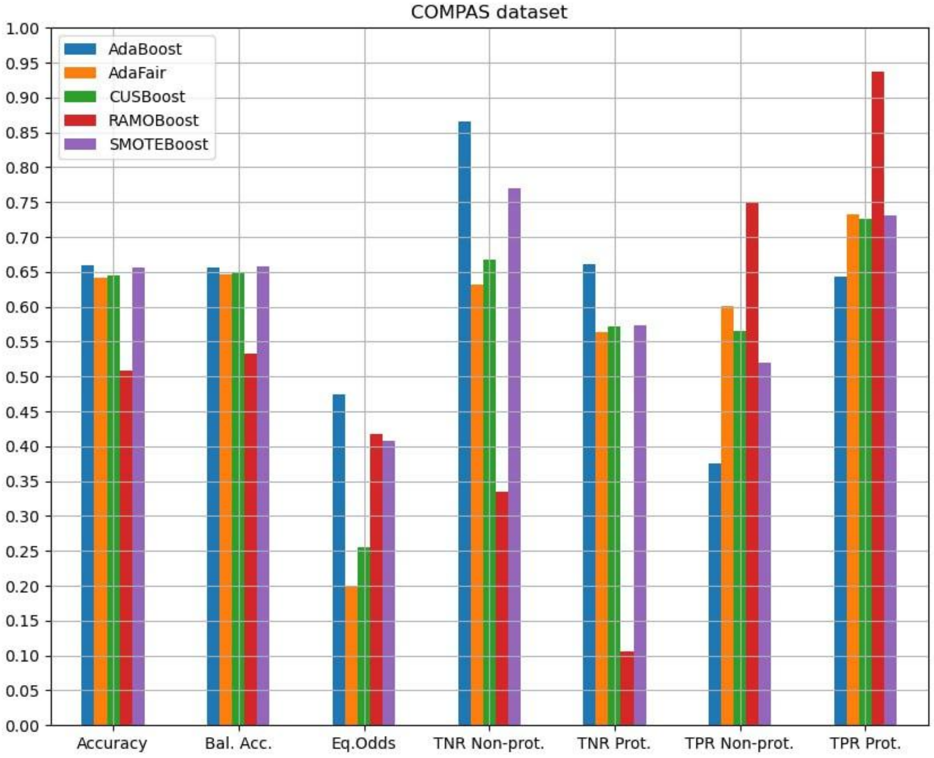
1: initialize weight, $x_i \in D$ to $\frac{1}{d}$;
2: **for** i = 1 to k **do**
3: create balanced dataset D_i with distribution D using cluster-based under-sampling;
4: derive a tree, M_i from D_i employing C4.5 algorithm;
5: compute the error rate of M_i, $error(M_i)$;
6: **if** $error(M_i) \geq 0.5$ **then**
7: go back to step 3 and try again;
8: **end if**
9: **for** each $x_i \in D_i$ that correctly classified **do**
10: multiply weight of x_i by $(\frac{error(M_i)}{1-error(M_i)})$; // update weights
11: **end for**
12: normalise the weight of each instances, x_i;
13: **end for**

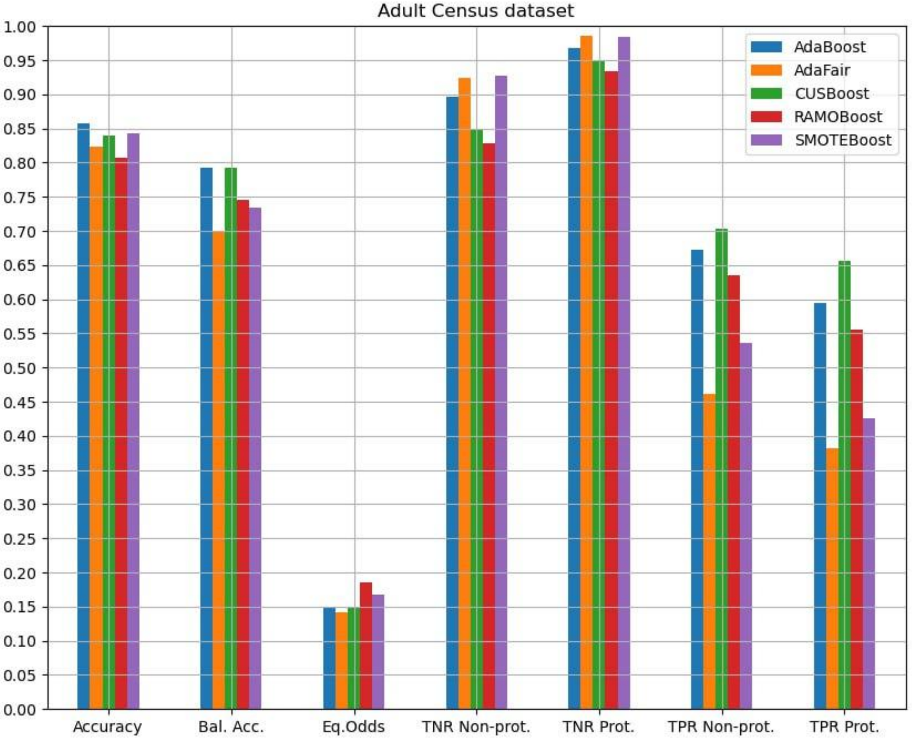
To use the ensemble to classify instance, x_{New}:

1: initialise weight of each class to 0;
2: **for** i = 1 to k **do**
3: $w_i = log\frac{1-error(M_i)}{error(M_i)}$; // weight of the classifier's vote
4: $c = M_i(x_{New})$; // class prediction by M_i
5: add w_i to weight for class c;
6: **end for**
7: return the class with largest weight;

Comparison

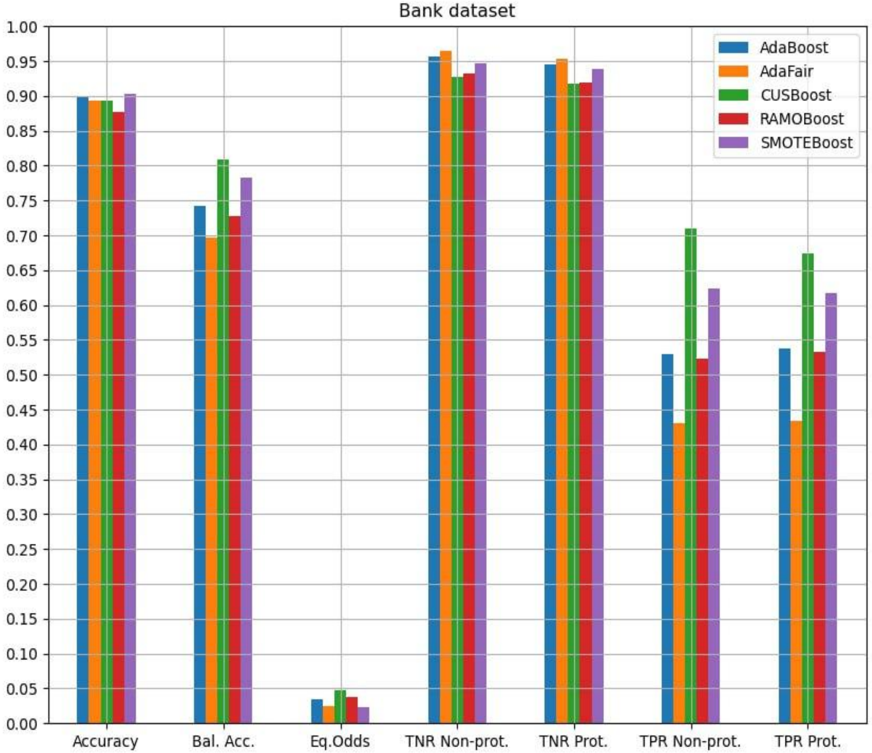
Class Ratio: 1:1.12 1:3.03

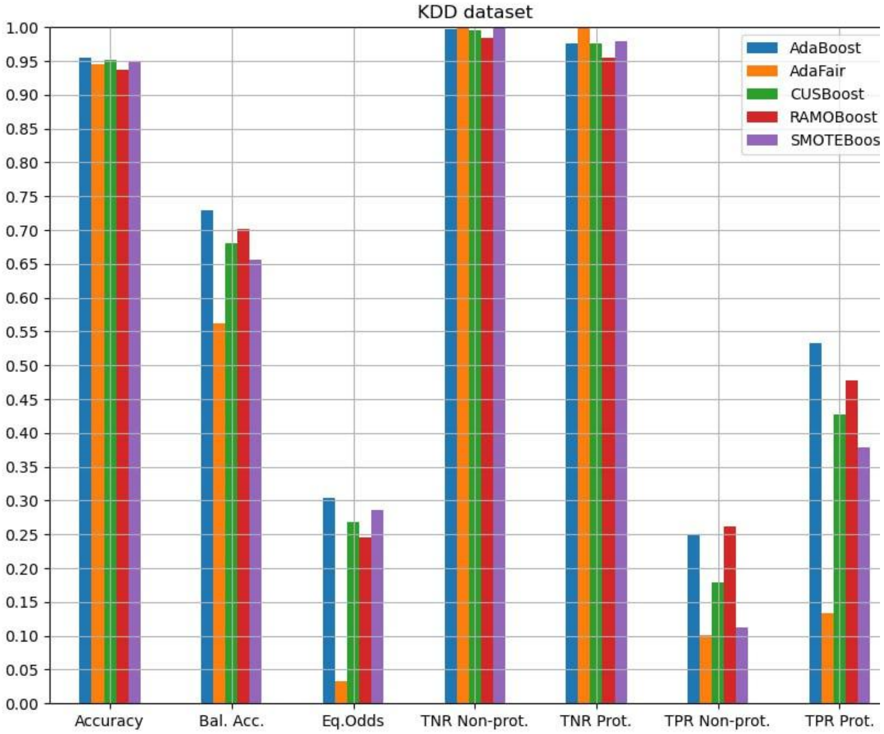




Comparison

Class Ratio: 1:7.57 1:15.11





Comparison

AdaFair:

- It is **dominated in predictive performance** by other algorithms on all datasets , but achieves **good performance in fairness** term for both classes. AdaFair tends to show the high TNRs for both protected and non-protected classes, but low TPRs, e.g. on Adult census dataset rejecting more instances of the positive class in order to minimize Eq.Odds.

CUSBoost:

- It **outperforms** all algorithms on all datasets except KDD dataset **in prediction accuracy**. CUSBoost **doesn't equalizes classes** as good as AdaFair or SMOTEBoost, especially on datasets with high imbalance, e.g. Compass and KDD. Still, CUSBoost shows high TPRs on data with class ratio 1:7 and 1:15, which indicates bias towards a minor class.

SMOTEBoost:

- It shows **a weak predictive ability** surpassing only AdaFair, archiving "the first place" only on data with low imbalance class ratio 1:1.12. SMOTEBoost **perform worse in equalizing classes** as it does not consider fairness. However, it showed high TNRs and TPRs for both groups on the Bank dataset with class ratio 1:7.5 without discrimination of the minority.

RAMOBoost:

- It shows **good and stable predictive performance** for all datasets except COMPAS data (1:1.12). Additionally, RAMOBoost **indicates high fairness** representation i.e. low Eq. Odds on all datasets, as it **biased toward positive class** - rejecting observations of the negative class and paying more attention to the minor class.

Conclusion

1. **Bias** of ML algorithms towards the major class **could be eliminated** by mitigate discrimination through objective function (AdaFair) and oversampling techniques (SMOTE-, RAMO- or CUSBoost).
2. There is an **empirical trade-off between accuracy** of predictions **and fairness**. The higher balanced ratio, the higher Eq. Odds for considered algorithms.
3. The **best algorithm in prediction is CUSBoost** - the 1st in 3 out of 4 datasets in balanced accuracy score.
4. The **best algorithm in fairness is AdaFair** - the 1st in 4 out of 4 datasets in Eq. Odds metrics.

Thank you
for attention!

CUSBoost considers a **series of decision trees** and combines the votes of each individual tree to classify new instances.

To compute the error rate of model M_i, we sum the weights of misclassified instances in D_i:

$$error(M_i) = \sum_{i=1}^{d} w_i * err(x_i)$$

If an instance, x_i is misclassified, then err(x_i) = 1
Otherwise, err(x_i) = 0 (when x_i is correctly classified).

Algorithm 1 CUSBoost Algorithm

Input: Imbalanced data, D, number of iterations, k, and C4.5 decision tree induction algorithm.
Output: An ensemble model.
Method:

1: initialize weight, $x_i \in D$ to $\frac{1}{d}$;
2: **for** i = 1 to k **do**
3:    create balanced dataset D_i with distribution D using cluster-based under-sampling;
4:    derive a tree, M_i from D_i employing C4.5 algorithm;
5:    compute the error rate of M_i, $error(M_i)$;
6:    **if** $error(M_i) \geq 0.5$ **then**
7:       go back to step 3 and try again;
8:    **end if**
9:    **for** each $x_i \in D_i$ that correctly classified **do**
10:       multiply weight of x_i by $(\frac{error(M_i)}{1-error(M_i)})$; // update weights
11:    **end for**
12:    normalise the weight of each instances, x_i;
13: **end for**

To use the ensemble to classify instance, x_{New}:

1: initialise weight of each class to 0;
2: **for** i = 1 to k **do**
3:    $w_i = log\frac{1-error(M_i)}{error(M_i)}$; // weight of the classifier's vote
4:    $c = M_i(x_{New})$; // class prediction by M_i
5:    add w_i to weight for class c;
6: **end for**
7: return the class with largest weight;

Adaptive Weights: theory

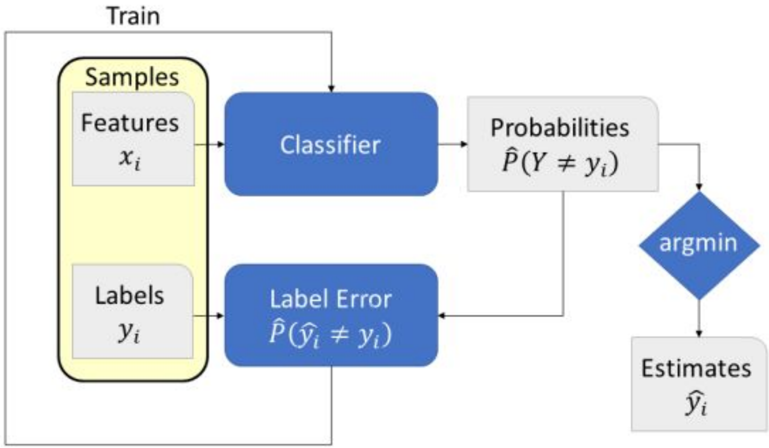


Figure 1: Probabilistic classifier training.

Probability estimates:

$$\hat{P}(Y = y_i) = 1 - \hat{P}(Y \neq y_i)$$

Classifier estimates class labels as:

$$\hat{y}_i = \underset{Y \in \{0,1\}}{\operatorname{argmax}} \hat{P}(Y = y_i) = \underset{Y \in \{0,1\}}{\operatorname{argmin}} \hat{P}(Y \neq y_i)$$

Related work: “Adaptive Sensitive Reweighting to Mitigate Bias in Fairness-aware Classification”

Adaptive Weights: theory

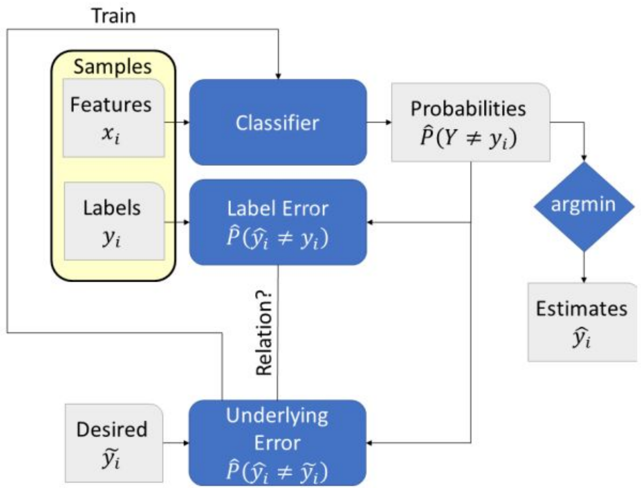


Figure 2: Directly training on observable desired labels. This can be ethically or legally questionable.

We try to minimize both weighted error on observed labels as well as the distance between weighted observed labels and unweighted underlying labels:

$$\min \sum_i w_i \hat{P}(\hat{y}_i \neq y_i)$$

$$\min \sum_i \left(w_i \hat{P}(\hat{y}_i \neq y_i) - \hat{P}(\hat{y}_i \neq \tilde{y}_i) \right)^2$$

Related work: “Adaptive Sensitive Reweighting to Mitigate Bias in Fairness-aware Classification”

Adaptive Weights: theory

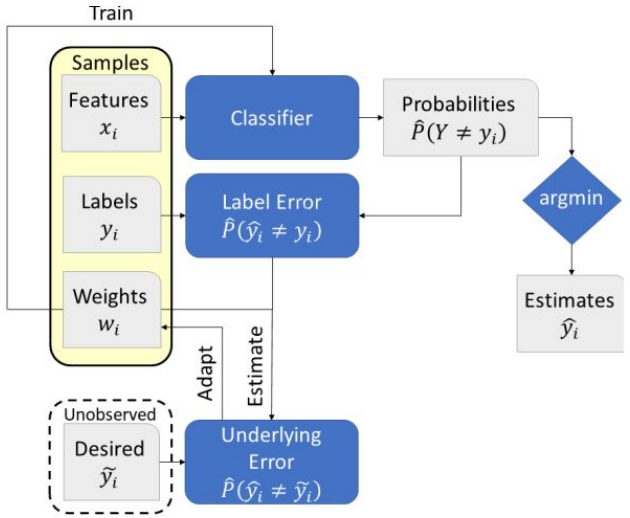


Figure 3: Training on unobservable desired labels.

Contrary to previous works on treating dataset bias, we assume that $\hat{P}(\hat{y}_i \neq \tilde{y}_i)$ cannot be estimated through group-specific dependencies.

Algorithm 1 Adaptive Sensitive Reweighting

---

function REWEIGHT(classifier C, data $\mathcal{D}$, sensitive group $\mathcal{S}$)
  $w_i \leftarrow 1 \, \forall i \in \mathcal{D}$
  $w_{i,prev} \leftarrow 1 + \sqrt{\epsilon} \, \forall i \in \mathcal{D}$
  while $\sum_{i \in \mathcal{D}} (w_i - w_{i,prev})^2 \geq \epsilon$ **do**
    train C on samples $i = (x_i, y_i) \in \mathcal{D}$ and weights $\frac{w_i}{\sum_{j \in \mathcal{D}} w_j}$
    use C to obtain $\hat{P}(\hat{y}_i \neq y_i)$.
    estimate $\hat{P}(\hat{y}_i \neq \tilde{y}_i)$ using $\hat{P}(\hat{y}_i \neq y_i) \, \forall i \in \mathcal{D}$
    $w_{i,prev} \leftarrow w_i \, \forall i \in \mathcal{D}$
    $w_i \leftarrow P(\hat{y}_i \neq \tilde{y}_i)/P(\hat{y}_i \neq y_i) \, \forall i \in \mathcal{D}$ (see Section 4)
  return trained classifier C, $\{w_i\}$

Related work: “Adaptive Sensitive Reweighting to Mitigate Bias in Fairness-aware Classification”

RAMOBoost: full algorithm

Algorithm 1 $RAMOBoost(N, T, k_1, k_2, \alpha)$

Input:

1) Training dataset with m class examples $\langle(\boldsymbol{x_1}, y_1), \ldots, (\boldsymbol{x_m}, y_m)\rangle$, where $\boldsymbol{x_i}$ $(i = 1, \ldots, m)$ is an instance of the n dimensional feature space $\boldsymbol{X}$ and $y_i \in Y = \{major, minor\}$ is the class identity label associated with instance $\boldsymbol{x_i}$.
2) N: number of synthetic data samples to be generated at each iteration
3) T: number of iterations; i.e., the number of base classifiers
4) k_1: number of nearest neighbors in adjusting the sampling probability of the minority examples
5) k_2: number of nearest neighbors used to generate the synthetic data instances
6) α: the scaling coefficient

Let $B = \{(i, y) : i \in \{1, \ldots, m\}, y \neq y_i\}$

Initialize: $D_1(i, y) = 1/|B|$ for $(i, y) \in B$ (for two class problems, $|B| = m$)

Do for $t = 1, 2, \ldots, T$.

1) Sample the mislabeled training data with D_t and get back the sampled dataset S_e of identical size. Slice S_e into the majority subset e_1 and the minority subset e_2 of size m_{lt} and m_{st}, respectively.
2) For each example $x_i \in e_2$, find its k_1 nearest neighbors in the dataset S_e according to the Euclidean distance in n-dimensional space and calculate r_i defined as
$$r_i = \frac{1}{1 + \exp(-\alpha \cdot \delta_i)}, \qquad i = 1, 2, \ldots, m_{st} \qquad (2)$$
where δ_i is the number of majority cases in k_1 examples.
3) Normalize r_i according to
$$\hat{r_i} = \frac{r_i}{\sum_{i=1}^{m_{st}} r_i} \qquad (3)$$
such that $\{\hat{r_i}\}$ is a distribution function: $\sum_{i=1}^{m_{st}} r_i = 1$. Define $\boldsymbol{d_t} = \{\hat{r_i}\}$.

4) Sample e_2 with $\boldsymbol{d_t}$ and get back a sampling minority dataset g_t, of size m_{st}.
5) For each example $\boldsymbol{x_i} \in g_t$, find its k_2 nearest neighbors in e_2 according to the Euclidean distance in n dimensional space and use linear interpolation to generate N synthetic data samples.
6) Provide the base classifier with sampling dataset S_e and the N synthetic data samples.
7) Get back a hypothesis h_t: $\boldsymbol{X} \times Y \rightarrow [0,\ 1]$.
8) Calculate the pseudo-loss of h_t

$$\varepsilon_t = \frac{1}{2} \sum_{(i,y)\in B} D_t(i,y)\,(1 - h_t(\boldsymbol{x_i}, y_i) + h_t(\boldsymbol{x_i}, y)) \quad (4)$$

9) Set $\beta_t = \varepsilon_t / 1 - \varepsilon_t$
10) Update D_t

$$D_{t+1}(i,y) = \frac{D_t(i,y)}{Z_t} \beta_t^{(1+h_t(\boldsymbol{x_i},y_i)-h_t(\boldsymbol{x_i},y))} \quad (5)$$

where Z_t is a normalization constant.

End loop

output: The output hypothesis $h_{final}(\boldsymbol{x})$ is calculated as follows:

$$h_{final}(\boldsymbol{x}) = \arg\max_{y\in Y} \sum_{t=1}^{T} \left(\log \frac{1}{\beta_t}\right) h_t(\boldsymbol{x}, y) \quad (6)$$