CENG7880
Trustworthy and Responsible AI

Instructor: Sinan Kalkan

(https://ceng.metu.edu.tr/~skalkan)

For course logistics and materials:
https://metu-trai.github.io

MIDDLE EAST TECHNICAL UNIVERSITY
DEPARTMENT OF COMPUTER ENGINEERING

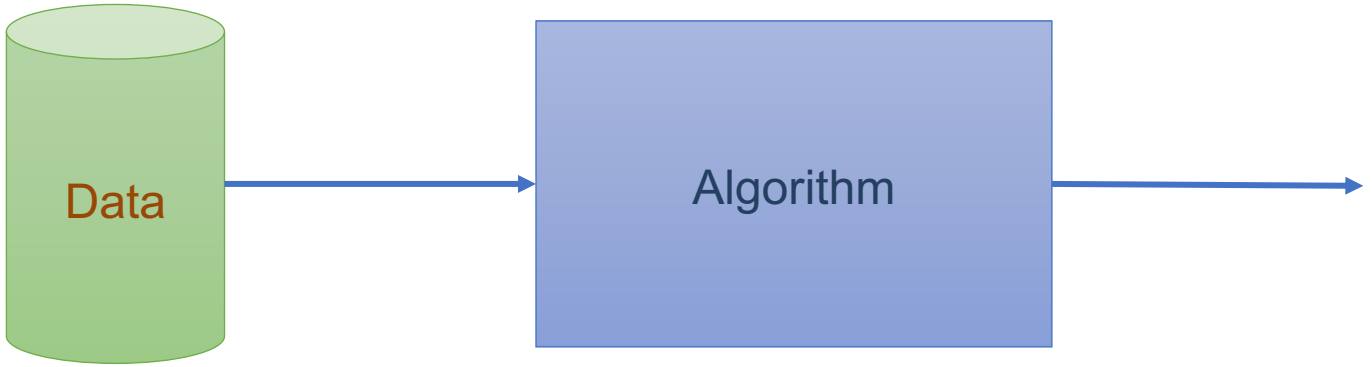
Previously on CENG7880

Fairness Algorithms

Pre-processing Methods

In-processing Methods

Post-processing Methods



Output
(e.g., emotion prediction)

Previously on CENG7880

Fairness Algorithms
Pre-processing Methods

Dataset-level Methods

1. Prepare a new dataset
2. Use sampling methods to oversample or undersample the dataset
3. Augment the datasets with small perturbations on the existing samples
4. Generate synthetic data
5. Fairness through unawareness

Table 4: Credit score distribution by ethnicity

Race or ethnicity	Samples with both score and outcome
White	133,165
Black	18,274
Hispanic	14,702
Asian	7,906
Total	174,047

https://fairmlbook.org/pdf/fairmlbook.pdf

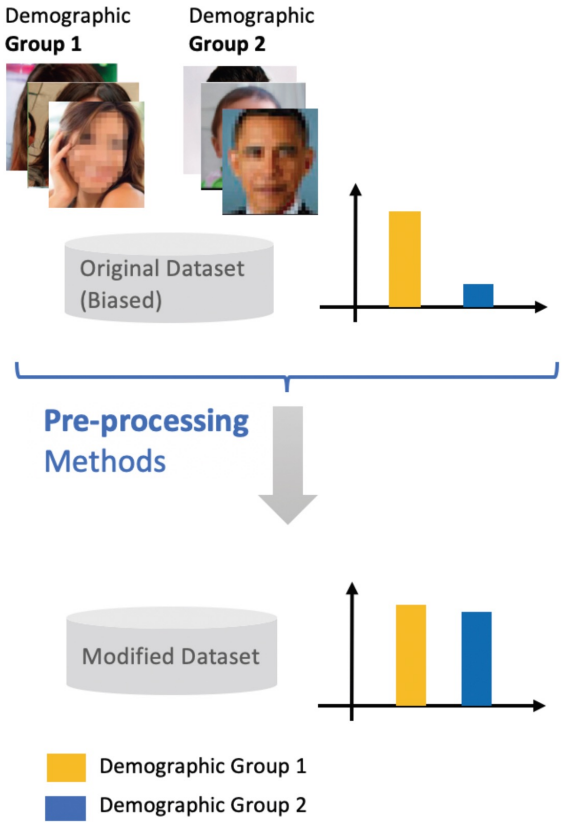


Fig: J. Cheong, S. Kalkan, H. Gunes, "The Hitchhiker's Guide to Bias and Fairness in Facial Affective Signal Processing", IEEE Signal Processing Magazine, 2021.

Fairness Algorithms
Preprocessing Methods

Previously on CENG7880

Advantages:

- Algorithm independent
- Easy to apply

Disadvantages/Challenges

- Sub-sampling can remove critical data
- Over-sampling can cause overfitting
- Balancing samples across demographic groups may not guarantee fairness (Wang et al., 2019; Cheong et al., 2023)

Recommendations

- The use of synthetic samples from generative models are promising (but generative models can also have bias)

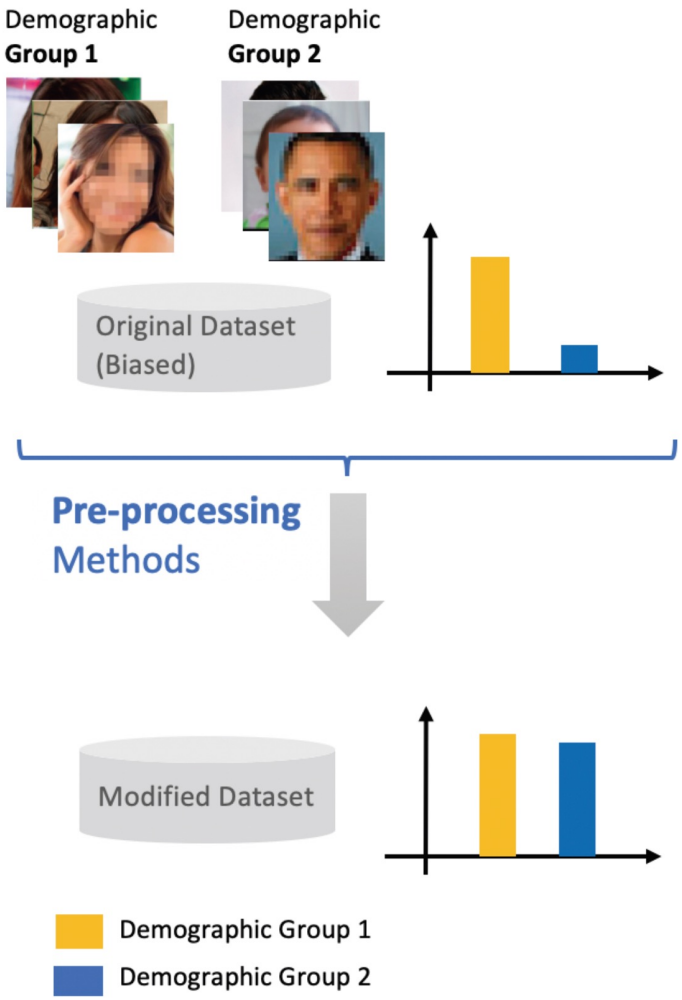


Fig: J. Cheong, S. Kalkan, H. Gunes, "The Hitchhiker's Guide to Bias and Fairness in Facial Affective Signal Processing", IEEE Signal Processing Magazine, 2021.

Wang, T., Zhao, J., Yatskar, M., Chang, K., Ordonez, V.: Balanced datasets are not enough: Estimating and mitigating gender bias in deep image representations. In: IEEE/CVF International Conference on Computer Vision (ICCV). pp. 5309-5318 (2019)
Cheong, J., Kuzucu, S., Kalkan, S., & Gunes, H. (2023). Towards gender fairness for mental health prediction. International Joint Conferences on Artificial Intelligence Organization.

Previously on CENG7880

Fairness Algorithms
Inprocessing Methods

1. Cost-sensitive learning
2. Domain adaptation
3. Disentanglement

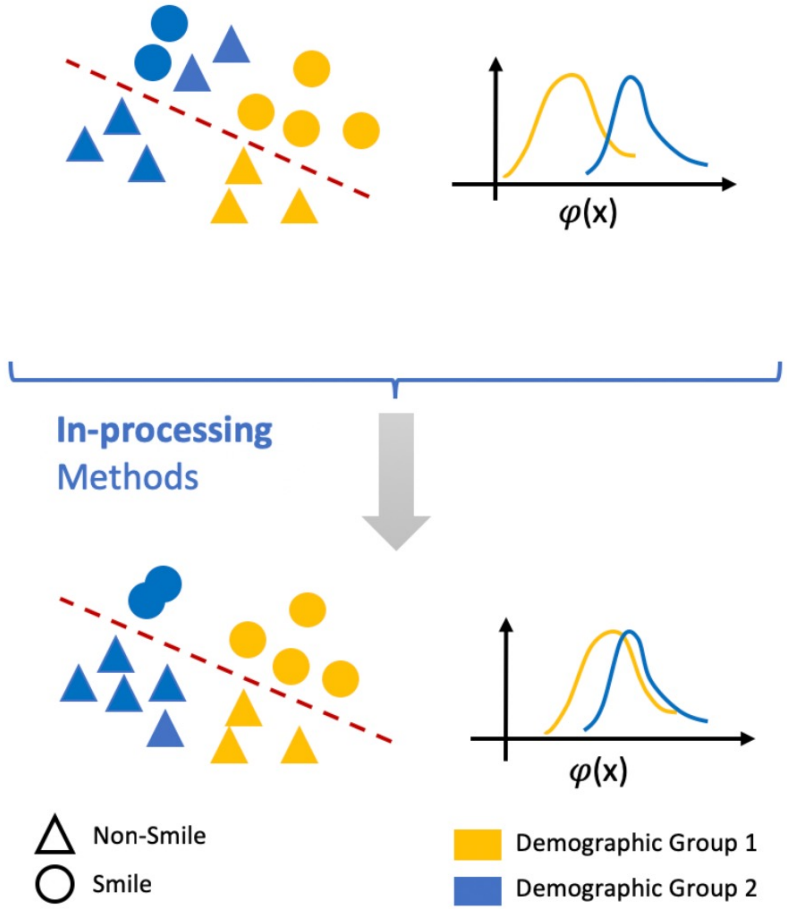


Fig: J. Cheong, S. Kalkan, H. Gunes, "The Hitchhiker's Guide to Bias and Fairness in Facial Affective Signal Processing", IEEE Signal Processing Magazine, 2021.

Previously on CENG7880

Fairness Algorithms
Inprocessing Methods

Advantages:

- More effective than preprocessing methods
- Many options to intervene at and mitigate bias

Disadvantages/Challenges

- Method-specific
- Requires expertise

Recommendations

- Hybrid solutions combining different approaches can be promising

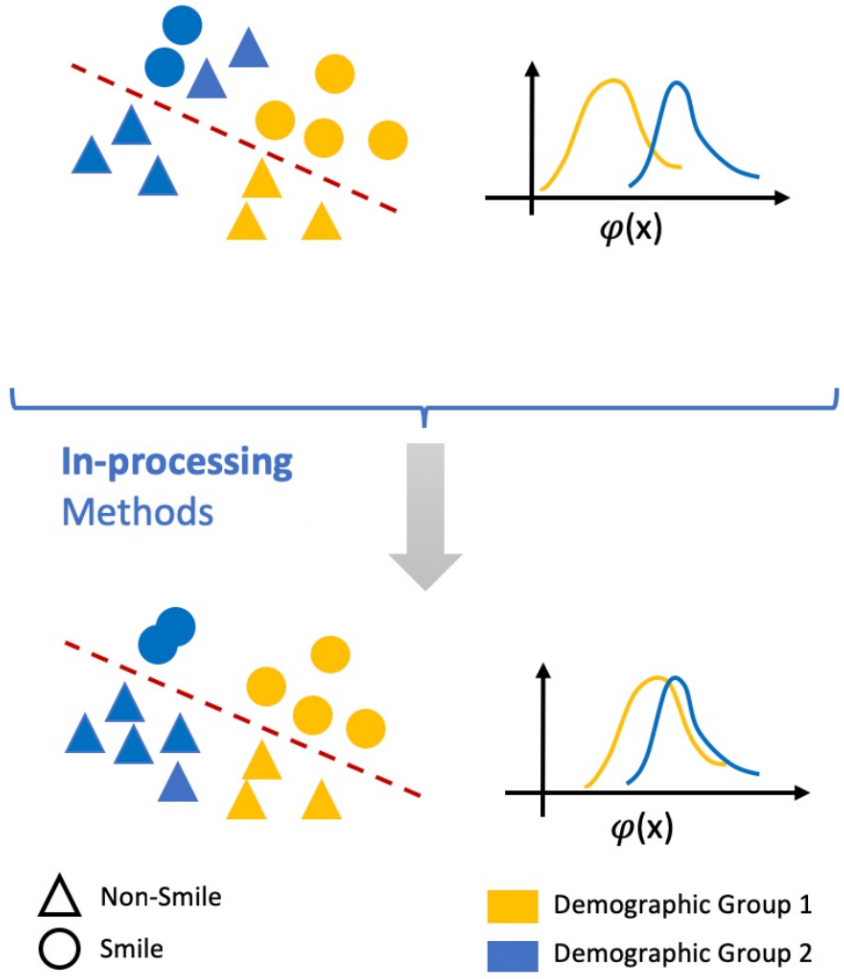


Fig: J. Cheong, S. Kalkan, H. Gunes, "The Hitchhiker's Guide to Bias and Fairness in Facial Affective Signal Processing", IEEE Signal Processing Magazine, 2021.

Previously on CENG7880

Fairness Algorithms
Postprocessing Methods

- Update model predictions (scores)

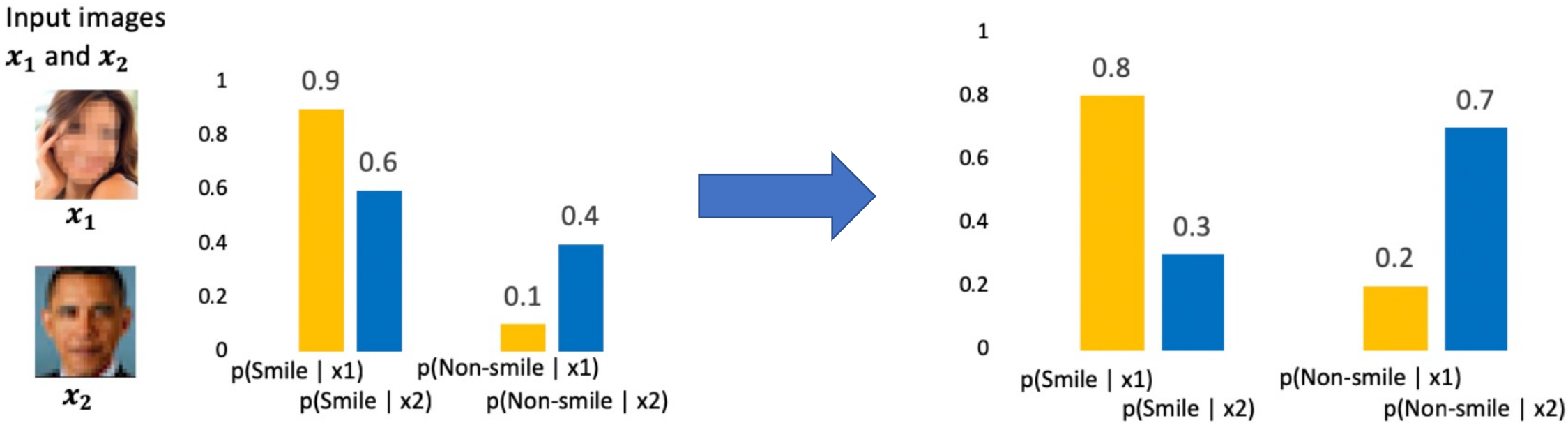


Fig: J. Cheong, S. Kalkan, H. Gunes, "The Hitchhiker's Guide to Bias and Fairness in Facial Affective Signal Processing", IEEE Signal Processing Magazine, 2021.

Fairness Algorithms
Postprocessing Methods

Previously on CENG7880

Advantages:

- Does not depend on the method
- No knowledge required about the method
- Easy to use for non-experts
- Readily available as frameworks

Disadvantages/Challenges

- Correcting outputs may fall insufficient for mitigating bias
- For explainable and interpretable fairness, we may need to intervene at the model
- May require additional training method

Recommendations:

- Before using pre-trained networks in practice, these methods should be tried.

Previously on CENG7880

Biases in LLMs

- Biases in NLP tasks
  - Text generation, machine translation, information retrieval, question-answering, …
- Biases in development and deployment
  - Training data
  - Model
  - Evaluation
  - Deployment

Previously on CENG7880

Metrics of Bias in LLMs

Embedding-based

- **Word embeddings**: Compute distances in embedding space
- **Sentence embeddings**: Adapt to contextualized embeddings

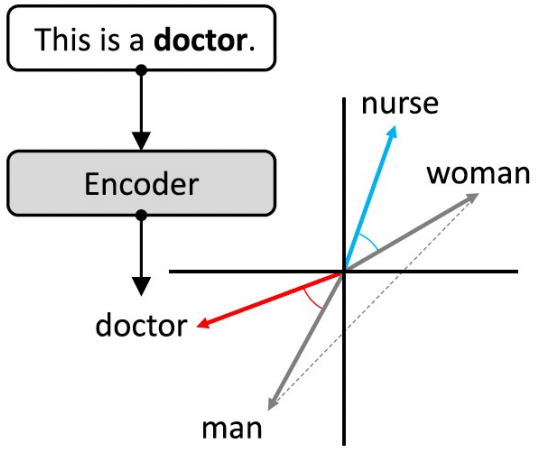


Figure 3
Example embedding-based metrics (§ 3.3). Sentence-level encoders produce sentence embeddings that can be assessed for bias. Embedding-based metrics use cosine similarity to compare words like "doctor" to social group terms like "man." Unbiased embeddings should have similar cosine similarity to opposing social group terms.

Gallegos et al. (2024). Bias and fairness in large language models: A survey. Computational Linguistics, 50(3), 1097-1179.

Previously on CENG7880

Metrics of Bias in LLMs

Probability-based metrics

- **Masked token**: Compare fill-in-the-blank probabilities
- **Pseudo-log-likelihood**: Compare likelihoods between sentences

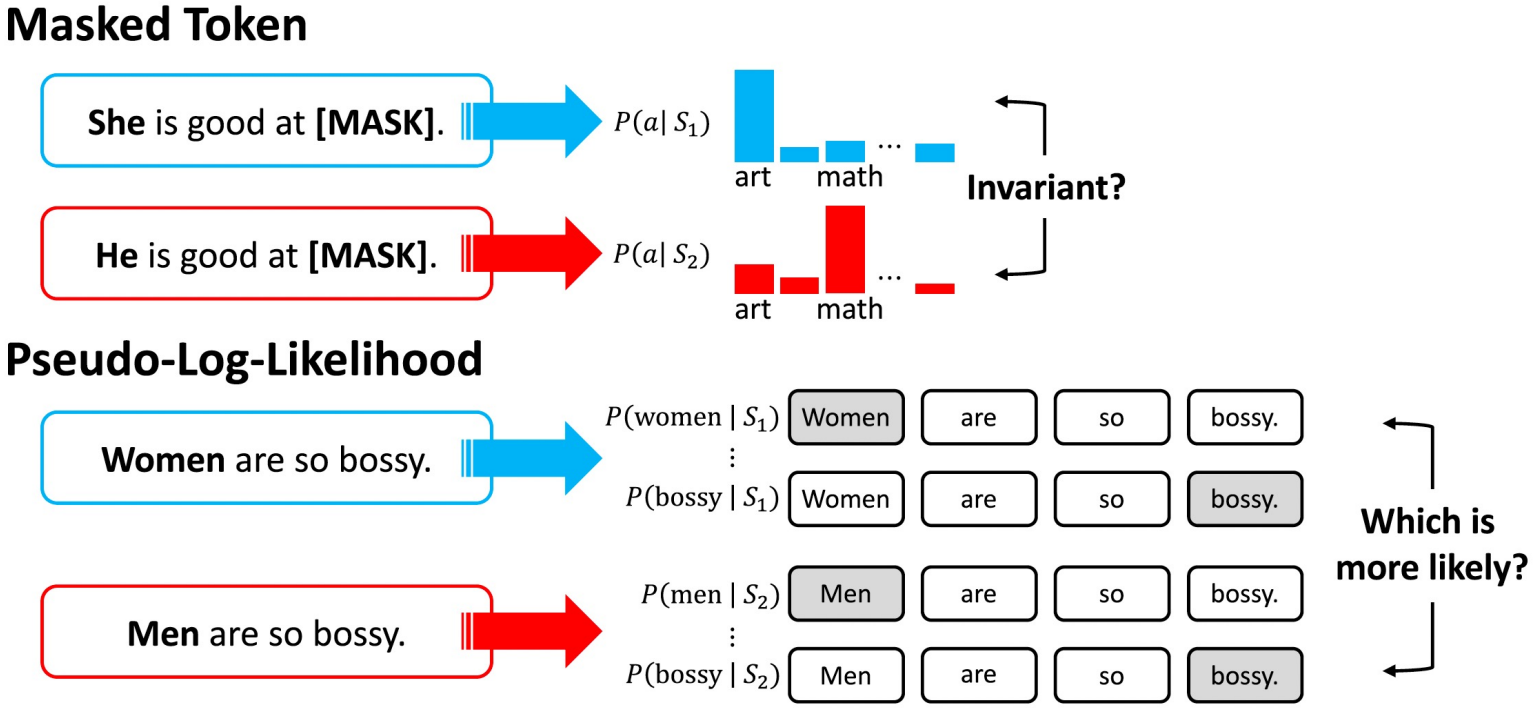


Figure 4
Example probability-based metrics (§ 3.4). We illustrate two classes of probability-based metrics: masked token metrics and pseudo-log-likelihood metrics. Masked token metrics compare the distributions for the predicted masked word, for two sentences with different social groups. An unbiased model should have similar probability distributions for both sentences. Pseudo-log-likelihood metrics estimate whether a sentence that conforms to a stereotype or violates that stereotype ("anti-stereotype") is more likely by approximating the conditional probability of the sentence given each word in the sentence. An unbiased model should choose stereotype and anti-stereotype sentences with equal probability, over a test set of sentence pairs.

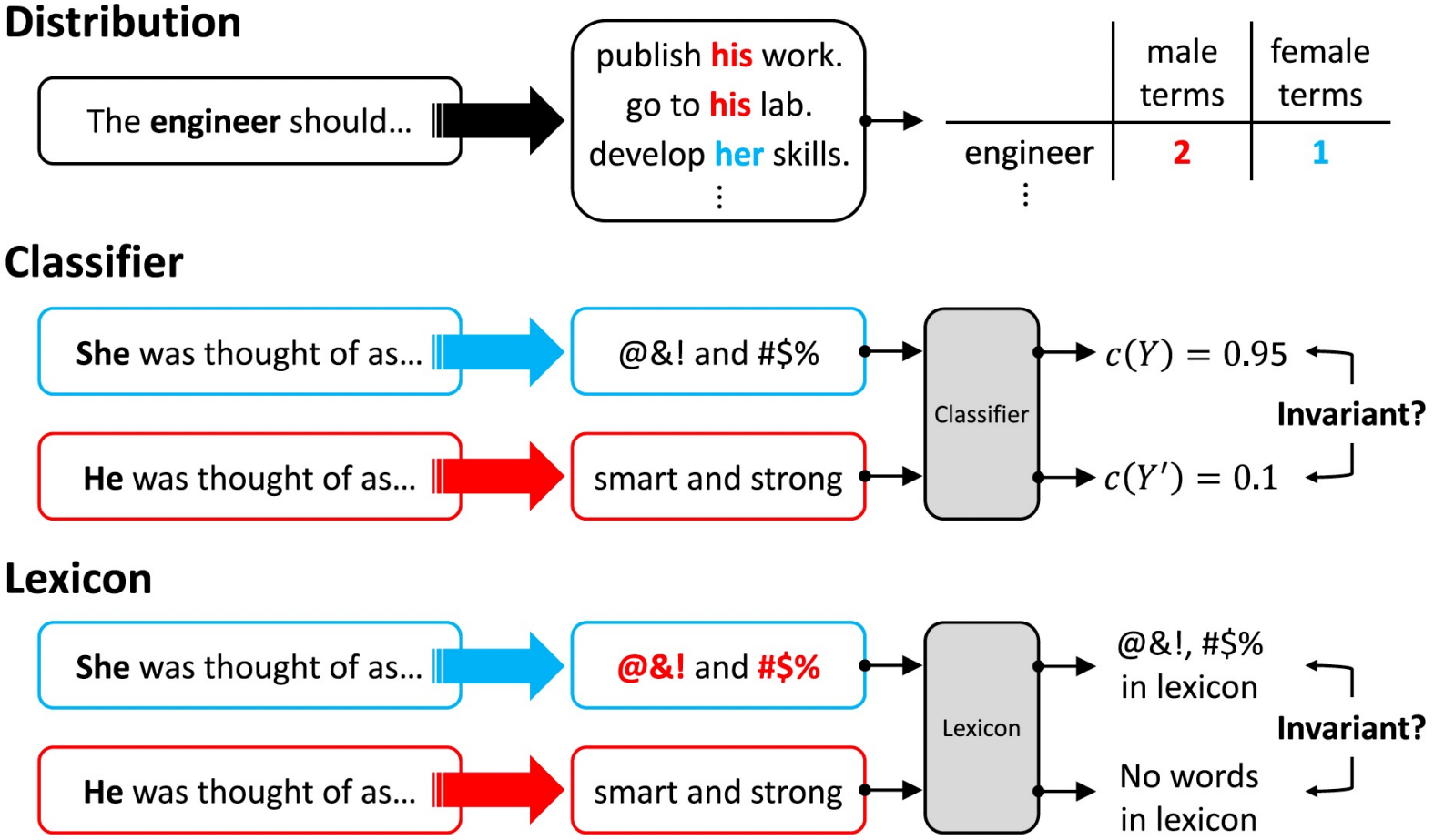
Gallegos et al. (2024). Bias and fairness in large language models: A survey. Computational Linguistics, 50(3), 1097-1179.

Metrics of Bias in LLMs

Previously on CENG7880

Generated text based metrics

- **Distribution**: Compare distribution of co-occurrences
- **Classifier**: Use an auxiliary classifier (e.g., existing sentiment or toxicity classifiers)
- **Lexicon**: Compare each word in the output to a predefined lexicon



Gallegos et al. (2024). Bias and fairness in large language models: A survey. Computational Linguistics, 50(3), 1097-1179.

Previously on CENG7880

Table 5
Taxonomy of techniques for bias mitigation in LLMs. We categorize bias mitigation techniques by the stage at which they intervene. For an illustration of each mitigation stage, as well as inputs and outputs to each stage, see Figure 6.

Mitigation Stage	Mechanism
PRE-PROCESSING (§ 5.1)	Data Augmentation (§ 5.1.1) Data Filtering & Reweighting (§ 5.1.2) Data Generation (§ 5.1.3) Instruction Tuning (§ 5.1.4) Projection-based Mitigation (§ 5.1.5)
IN-TRAINING (§ 5.2)	Architecture Modification (§ 5.2.1) Loss Function Modification (§ 5.2.2) Selective Parameter Updating (§ 5.2.3) Filtering Model Parameters (§ 5.2.4)
INTRA-PROCESSING (§ 5.3)	Decoding Strategy Modification (§ 5.3.1) Weight Redistribution (§ 5.3.2) Modular Debiasing Networks (§ 5.3.3)
POST-PROCESSING (§ 5.4)	Rewriting (§ 5.4.1)

Gallegos et al. (2024). Bias and fairness in large language models: A survey. Computational Linguistics, 50(3), 1097-1179.

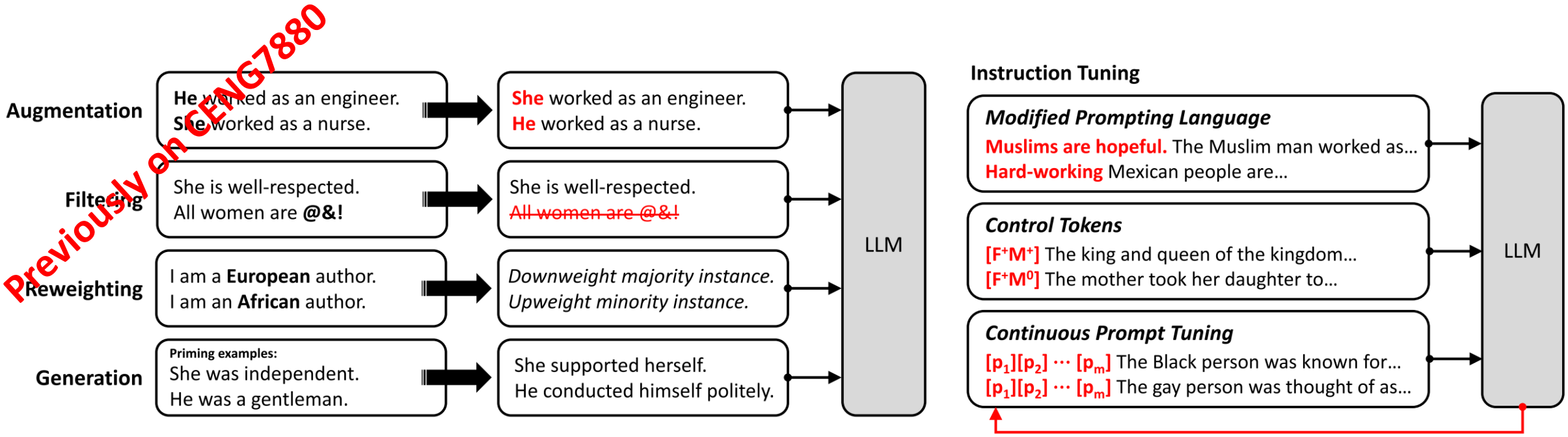


Figure 7
Example Pre-Processing Mitigation Techniques (§ 5.1). We provide examples of data augmentation, filtering, re-weighting, and generation on the left, as well as various types of instruction tuning on the right. The first example illustrates counterfactual data augmentation, flipping binary gender terms to their opposites. Data filtering illustrates the removal of biased instances, such as derogatory language (denoted as "@&!"). Reweighting demonstrates how instances representing underrepresented or minority instances may be upweighted for training. Data generation shows how new examples may be constructed by human or machine writers based on priming examples that illustrate the desired standards for the new data. Instruction

Gallegos et al. (2024). Bias and fairness in large language models: A survey. Computational Linguistics, 50(3), 1097-1179.

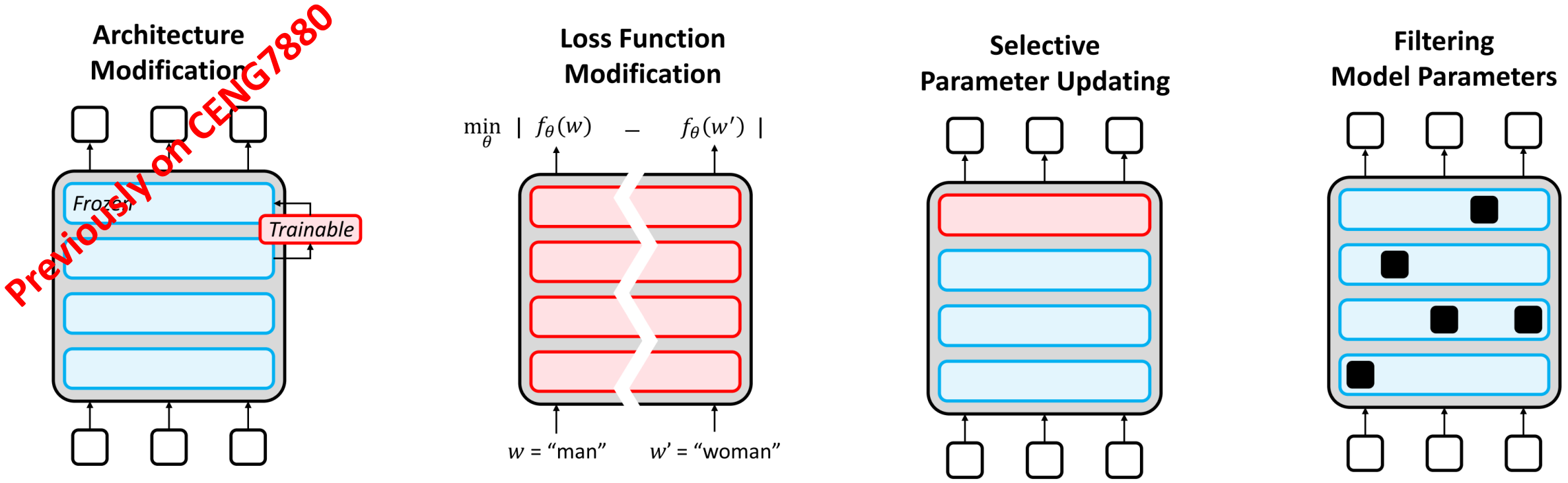


Figure 8

Example in-training mitigation techniques (§ 5.2). We illustrate four classes of methods that modify model parameters during training. Architecture modifications change the configuration of the model, such as adding new trainable parameters with adapter modules as done in this example (Lauscher, Lueken, and Glavaš 2021). Loss function modifications introduce a new optimization objective, such as equalizing the embeddings or predicted probabilities of counterfactual tokens or sentences. Selective parameter updates freeze the majority of the weights and only tune a select few during fine-tuning to minimize forgetting of pre-trained language understanding. Filtering model parameters, in contrast, freezes all pre-trained weights and selectively prunes some based on a debiasing objective.

Gallegos et al. (2024). Bias and fairness in large language models: A survey. Computational Linguistics, 50(3), 1097-1179.

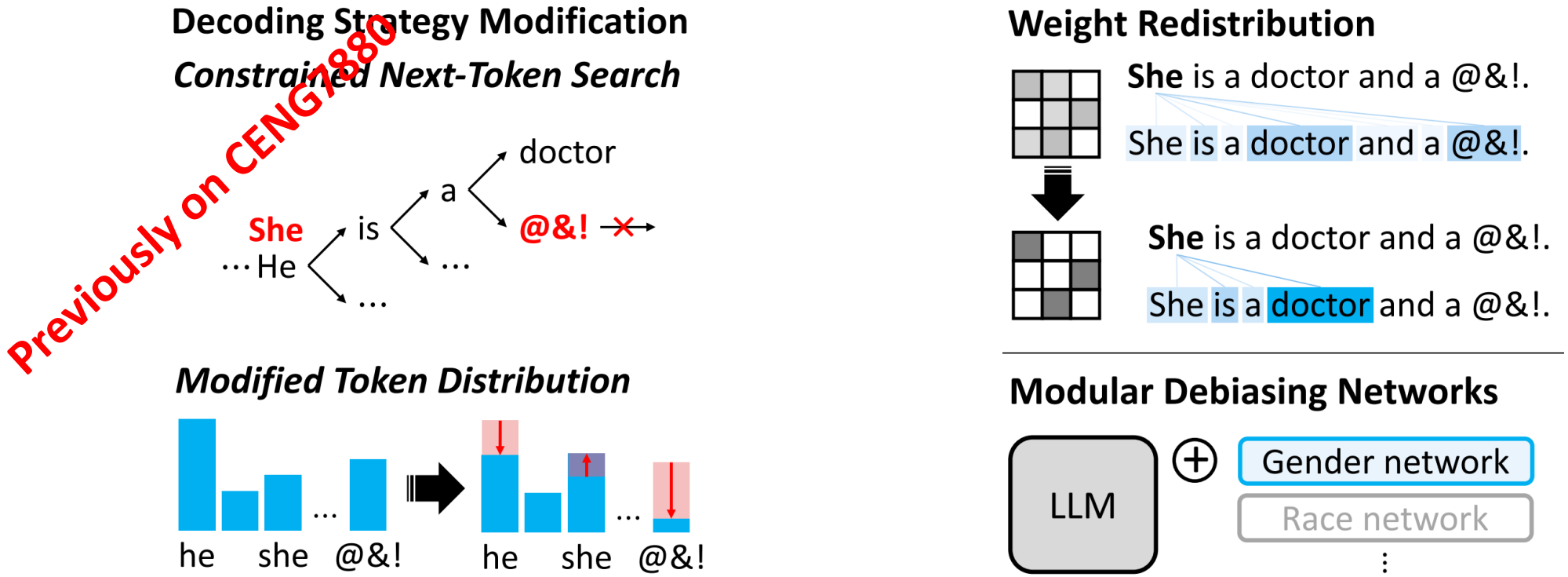


Figure 9
Example intra-processing mitigation techniques (§ 5.3). We show several methods that modify a model's behavior without training or fine-tuning. Constrained next-token search may prohibit certain outputs during beam search (e.g., a derogatory term "@&!," in this example), or generate and rerank alternative outputs (e.g., "he" replaced with "she"). Modified token distribution redistributes next-word probabilities to produce more diverse outputs and avoid biased tokens. Weight distribution, in this example, illustrates how post hoc modifications to attention matrices may narrow focus to less stereotypical tokens (Zayed et al. 2023b). Modular debiasing networks fuse the main LLM with stand-alone networks that can remove specific dimensions of bias, such as gender or racial bias.

Gallegos et al. (2024). Bias and fairness in large language models: A survey. Computational Linguistics, 50(3), 1097-1179.

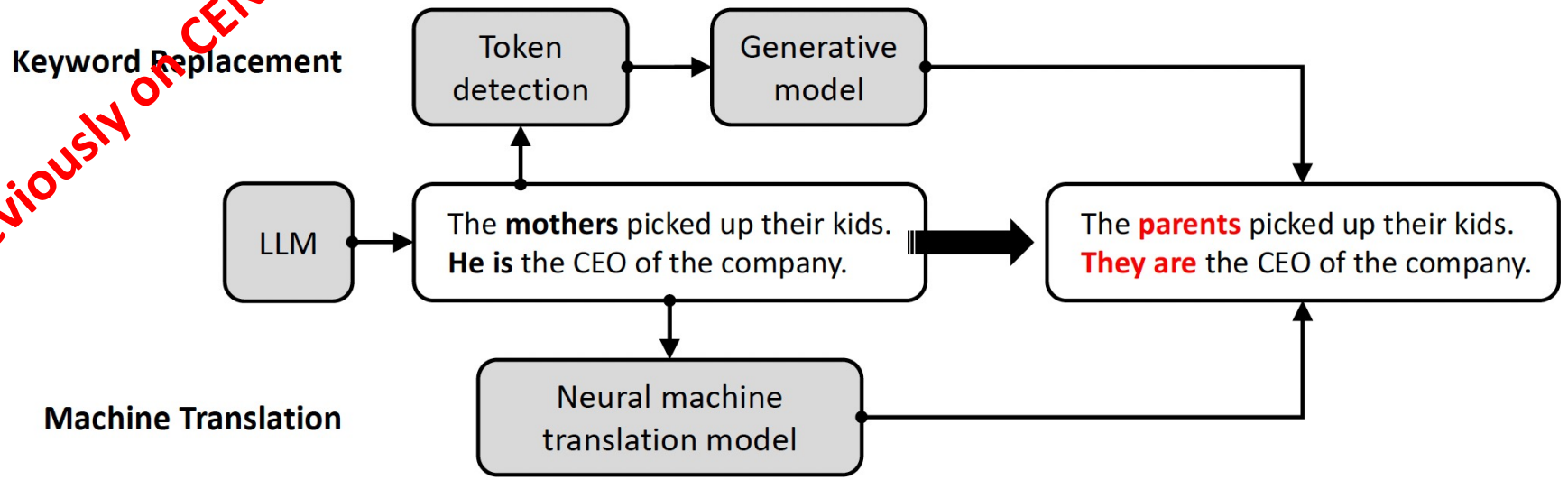


Figure 10
Example post-processing mitigation techniques (§ 5.4). We illustrate how post-processing methods can replace a gendered output with a gender-neutral version. Keyword replacement methods first identify protected attribute terms (i.e., "mothers," "he"), and then generate an alternative output. Machine translation methods train a neural machine translator on a parallel biased-unbiased corpus and feed the original output into the model to produce an unbiased output.

Gallegos et al. (2024). Bias and fairness in large language models: A survey. Computational Linguistics, 50(3), 1097-1179.

Agenda

- Fairness
  - Fairness Algorithms: Sample works from our research
  - Fairness Verification

Administrative Notes

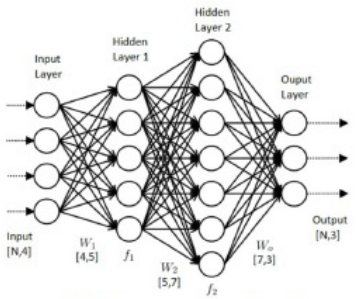
- Final Exam:
  - 13 January 16:30
- Paper selection finalized except for two projects
- Project milestones
  - **1. Milestone (November 23, midnight):**
    - Read & understand the paper
    - Download the datasets
    - Prepare the Readme file excluding the results & conclusion
  - **2. Milestone (December 7, midnight)**
    - The results of the first experiment
  - **3. Milestone (January 4, midnight)**
    - Final report (Readme file)
    - Repo with all code & trained models

Fairness Verification

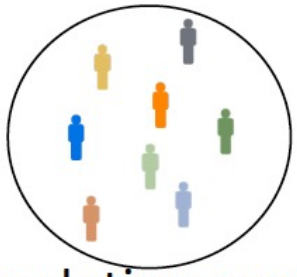
Resources:

- Language Models That Walk the Talk: A Framework for Formal Fairness Certificates, 2025.
- CertiFair: A Framework for Certified Global Fairness of Neural Networks, 2022.

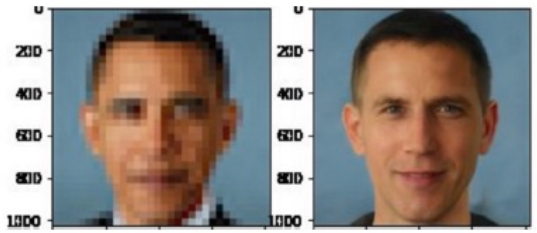
Fairness Verification



ML model



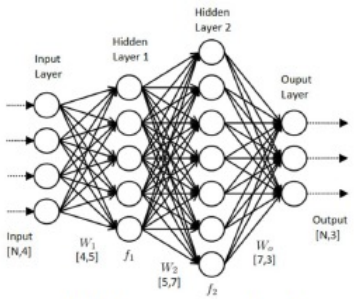
population model



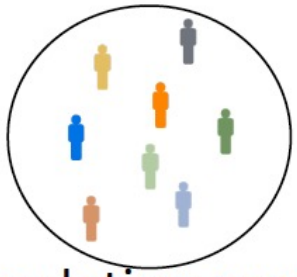
fairness definition

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

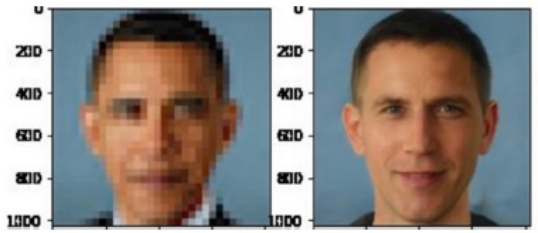
Fairness Verification



ML model



population model



fairness definition

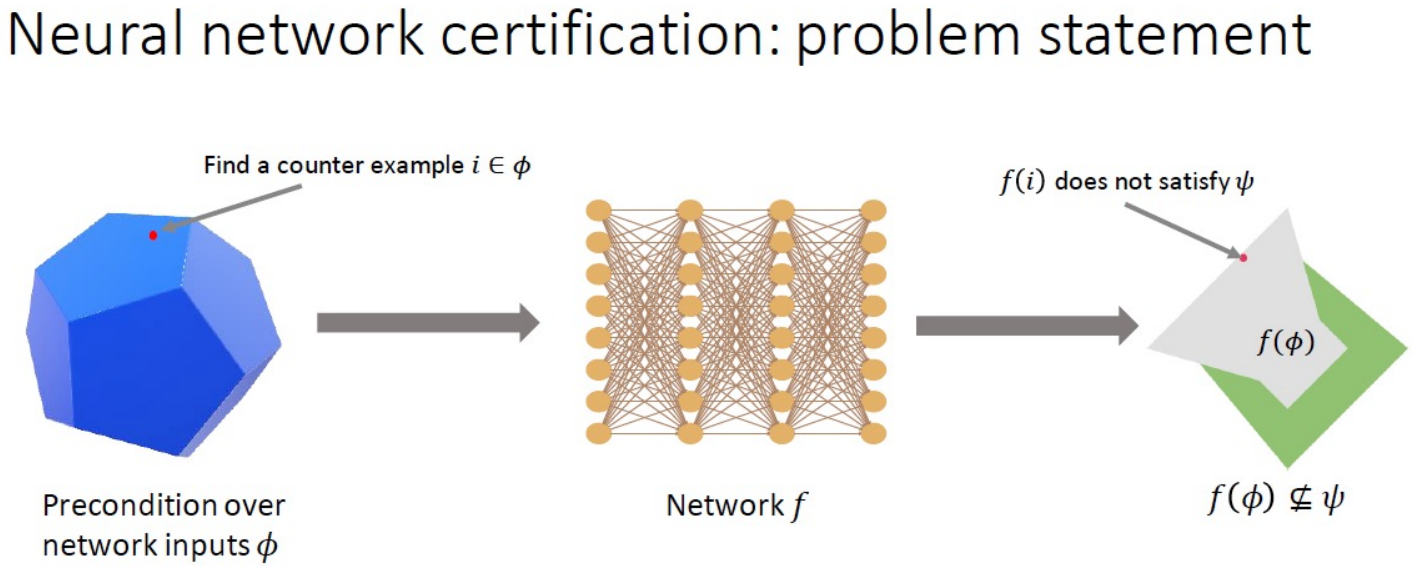
verification algorithm

fair/unfair

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Possible Approaches

- Approach 1: Formal/symbolic verification problem



- **Acknowledgement for slides:**
  - **CIS 6730 (Computer Aided Verification) Lectures**
  - **Formal methods in AI: Gagandeep Singh and Madhu Parthasarathy (UIUC)**

Possible Approaches

- Approach 1: Formal/symbolic verification problem

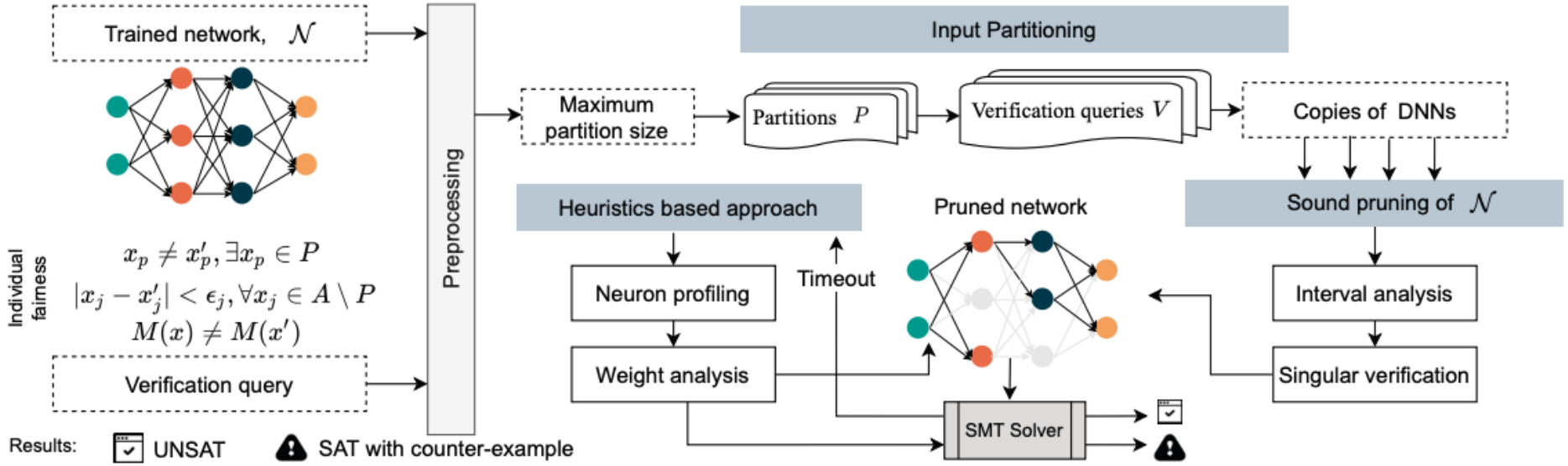


Fig. 1: The overview of our approach for the fairness verification of neural networks

https://dl.acm.org/doi/pdf/10.1109/ICSE48619.2023.00134

Possible Approaches

- Approach 2: Statistical verification
  - Use random sampling to check correctness, and use statistical tools to bound probability of false negatives

Probably Approximately Correct (PAC)

$$\Pr_{X^{(1)},\ldots,X^{(n)}\sim P_X}\left[f \text{ correct} \mid \mathcal{A}\left(f; X^{(1)},\ldots,X^{(n)}\right) = \text{correct}\right] \geq 1-\delta$$

(is it really fair?)

(Algorithm says it is fair)

Fairness Verification

- **Goal:** Check if a given model satisfies a given fairness definition

- Ideally, the verification strategy should be flexible, and work on a broad family of fairness definitions
  - Focus on group fairness

- **Note:** Fairness is a **statistical property**!
  - Depends on data distribution $p(x, y)$
  - Therefore, we also need to specify $p(x, y)$, which we call the **population model**

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Fairness

- **Problem Setup**
  - Distribution $P_{\mathcal{V}}$ over individuals $v = (\tilde{v}, a) \in \mathcal{V}$ (called the **population model**)
  - Sensitive attribute $a \in \{\text{majority}, \text{minority}\}$
  - Binary classifier $f: \mathcal{V} \to \{0,1\}$, where 1 indicates a positive outcome

- **Fairness Properties:** Demographic parity, equality of opportunity, etc.

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Demographic Parity

- Majority and minority members get positive outcomes at the same rate
- Let the **acceptance probability** for a be

$$\mu_a^* = \Pr_{v \sim \mathcal{V}}[\, f(v) = 1 \mid A = a \,]$$

- Then, f satisfies demographic parity if $Y^*_{\text{parity}} = 1$, where

$$Y^*_{\text{parity}} = 1\left[\frac{\mu^*_{\text{minority}}}{\mu^*_{\text{majority}}} \geq c\right]$$

  - The constant $c \in [0,1]$ is domain specific
  - **Question:** Does $Y^*_{\text{parity}} = 1$?

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

col_rank: college rank
years_exp: years of experience

Fairness Verification Problem

Assume that we have access to the decision program & population model.

```
def population_model():
    is_male ~ bernoulli(0.5)
    col_rank ~ normal(25, 10)
    if is_male:
        years_exp ~ normal(15, 5)
    else:
        years_exp ~ normal(10, 5)
    return col_rank, years_exp
```

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

col_rank: college rank
years_exp: years of experience

Fairness Verification Problem

```
def population_model():
    is_male ~ bernoulli(0.5)
    col_rank ~ normal(25, 10)
    if is_male:
        years_exp ~ normal(15, 5)
    else:
        years_exp ~ normal(10, 5)
    return col_rank, years_exp
```

```
def offer_job(col_rank, years_exp)
    if col_rank <= 5:
        return true
    elif years_exp > 5:
        return true
    else:
        return false
```

Question: Does OfferJob satisfy demographic parity?

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Fairness Verification

- **Goal:** Check if $Y^*_{\text{parity}} = 1$, where

$$Y^*_{\text{parity}} = 1\left[\frac{\mu^*_{\text{minority}}}{\mu^*_{\text{majority}}} \geq c\right]$$

$$\mu^*_a = \Pr_{v \sim \mathcal{V}}[\, f(v) = 1 \mid A = a \,]$$

- **Step 1:** Compute approximation $\hat{\mu}_a \approx \mu^*_a$
- **Step 2:** Compute approximation $\hat{Y}_{\text{parity}} \approx Y^*_{\text{parity}}$

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Fairness Verification Strategy

```
def population_model():
    is_male ~ bernoulli(0.5)
    col_rank ~ normal(25, 10)
    if is_male:
        years_exp ~ normal(15, 5)
    else:
        years_exp ~ normal(10, 5)
    return col_rank, years_exp
```

```
def offer_job(col_rank, years_exp)
    if col_rank <= 5:
        return true
    elif years_exp > 5:
        return true
    else:
        return false
```

Question: Does OfferJob satisfy demographic parity?

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Fairness Verification Strategy

```
def population_model():
    is_male ~ bernoulli(0.5)
    col_rank ~ normal(25, 10)
    if is_male:
        years_exp ~ normal(15, 5)
    else:
        years_exp ~ normal(10, 5)
    return col_rank, years_exp
```

```
def offer_job(col_rank, years_exp)
    if col_rank <= 5:
        return true
    elif years_exp > 5:
        return true
    else:
        return false
```

Question: What is $\Pr\left[\text{OfferJob}\left(\text{PopulationModel}(\,)\right) \mid \text{IsMale} = \text{True}\right]$?

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Fairness Verification Strategy

```
def population_model():
    is_male ~ bernoulli(0.5)
    col_rank ~ normal(25, 10)
    if is_male:
        years_exp ~ normal(15, 5)
    else:
        years_exp ~ normal(10, 5)
    return col_rank, years_exp
```

```
def offer_job(col_rank, years_exp)
    if col_rank <= 5:
        return true
    elif years_exp > 5:
        return true
    else:
        return false
```

Question: What is Pr[OfferJob]?

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

$$\Pr[\text{OfferJob}]$$
$$= \int \text{OfferJob}(a, r, e) \cdot p_{\text{IsMale}}(a) \cdot p_{\text{ColRank}}(r) \cdot p_{\text{YearsExp}}(e) \cdot da \cdot dr \cdot de$$

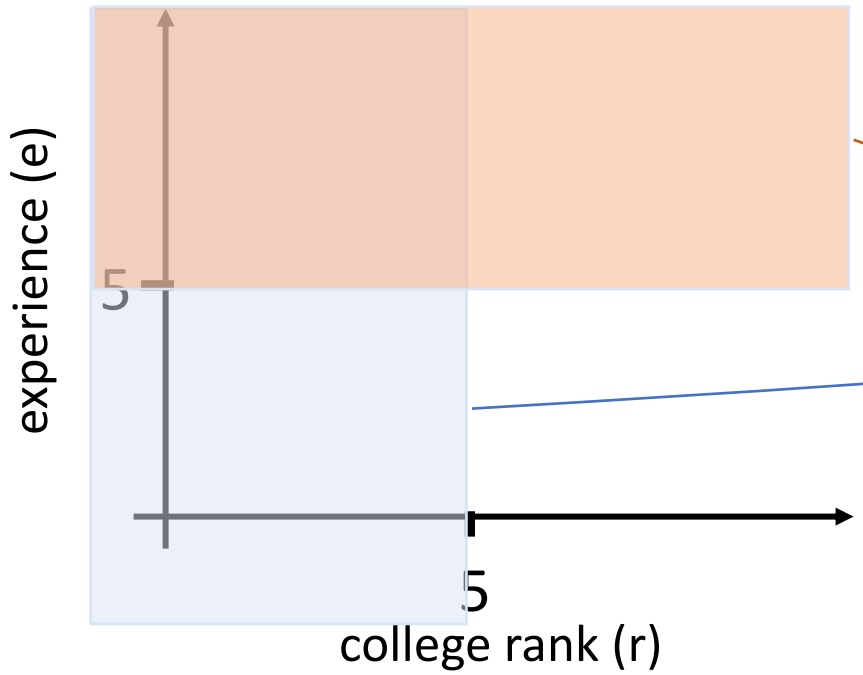
```
def population_model():
    is_male ~ bernoulli(0.5)
    col_rank ~ normal(25, 10)
    if is_male:
        years_exp ~ normal(15, 5)
    else:
        years_exp ~ normal(10, 5)
    return col_rank, years_exp


def offer_job(col_rank, years_exp)
    if col_rank <= 5:
        return true
    elif years_exp > 5:
        return true
    else:
        return false
```

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

$$\Pr[\text{OfferJob}]$$
$$= \int \text{OfferJob}(a, r, e) \cdot p_{\text{IsMale}}(a) \cdot p_{\text{ColRank}}(r) \cdot p_{\text{YearsExp}}(e) \cdot da \cdot dr \cdot de$$

$$\text{OfferJob} = (\text{ColRank} \leq 5 \vee \text{YearsExp} > 5)$$

We can calculate this probability by noticing that we have well-defined rectangular regions (rectangles) covering different ranges of variables.




```
def population_model():
    is_male ~ bernoulli(0.5)
    col_rank ~ normal(25, 10)
    if is_male:
        years_exp ~ normal(15, 5)
    else:
        years_exp ~ normal(10, 5)
    return col_rank, years_exp
```

```
def offer_job(col_rank, years_exp)
    if col_rank <= 5:
        return true
    elif years_exp > 5:
        return true
    else:
        return false
```

$$\Pr[\text{OfferJob}]$$

$$= \int \text{OfferJob}(a, r, e) \cdot p_{\text{IsMale}}(a) \cdot p_{\text{ColRank}}(r) \cdot p_{\text{YearsExp}}(e) \cdot da \cdot dr \cdot de$$

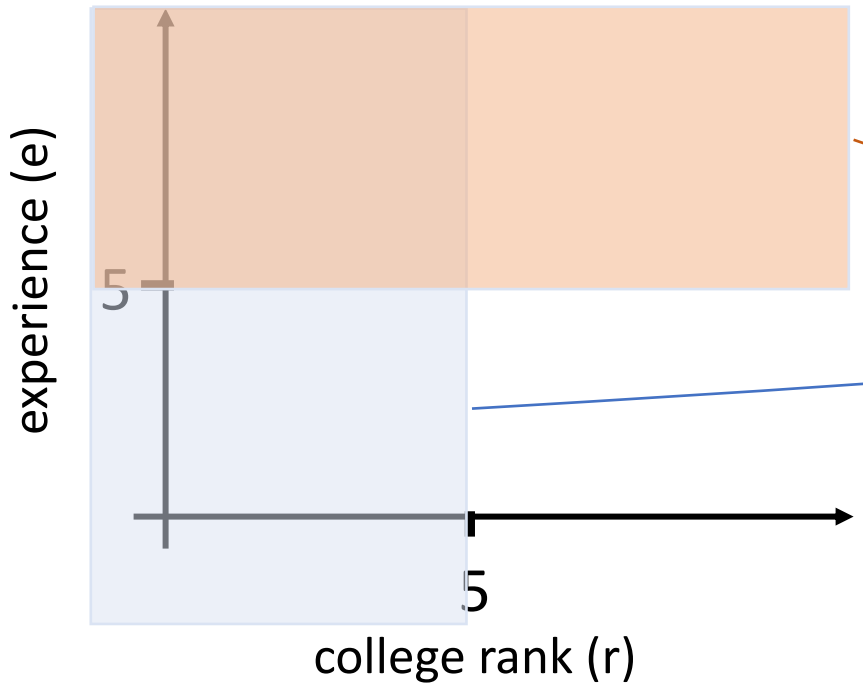
$$\text{OfferJob} = (\text{ColRank} \leq 5 \vee \text{YearsExp} > 5)$$

$$= (\text{ColRank} \leq 5) \vee (\text{ColRank} > 5 \wedge \text{IsMale} \wedge \text{YearsExpLarge} > 5)$$

$$\vee (\text{ColRank} > 5 \wedge \neg\text{IsMale} \wedge \text{YearsExpSmall} > 5)$$

```
def population_model():
    is_male ~ bernoulli(0.5)
    col_rank ~ normal(25, 10)
    if is_male:
        years_exp ~ normal(15, 5)
    else:
        years_exp ~ normal(10, 5)
    return col_rank, years_exp
```

```
def offer_job(col_rank, years_exp)
    if col_rank <= 5:
        return true
    elif years_exp > 5:
        return true
    else:
        return false
```



$$\Pr[\text{OfferJob}]$$

$$= \int \text{OfferJob}(a, r, e) \cdot p_{\text{IsMale}}(a) \cdot p_{\text{ColRank}}(r) \cdot p_{\text{YearsExp}}(e) \cdot da \cdot dr \cdot de$$

$$\begin{aligned}\text{OfferJob} &= (\text{ColRank} \leq 5 \vee \text{YearsExp} > 5) \\ &= (\text{ColRank} \leq 5) \vee (\text{ColRank} > 5 \wedge \text{IsMale} \wedge \text{YearsExpLarge} > 5) \\ &\quad \vee (\text{ColRank} > 5 \wedge \neg\text{IsMale} \wedge \text{YearsExpSmall} > 5)\end{aligned}$$

$$\begin{aligned}&\Pr[\text{OfferJob}] \\ &= \Pr[\text{ColRank} \leq 5] + \Pr[\text{ColRank} > 5 \wedge \text{IsMale} \wedge \text{YearsExpLarge} > 5] \\ &\quad + \Pr[\text{ColRank} > 5 \wedge \neg\text{IsMale} \wedge \text{YearsExpSmall} > 5]\end{aligned}$$

```
def population_model():
    is_male ~ bernoulli(0.5)
    col_rank ~ normal(25, 10)
    if is_male:
        years_exp ~ normal(15, 5)
    else:
        years_exp ~ normal(10, 5)
    return col_rank, years_exp


def offer_job(col_rank, years_exp)
    if col_rank <= 5:
        return true
    elif years_exp > 5:
        return true
    else:
        return false
```

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

$$\Pr[\text{OfferJob}]$$

$$= \int \text{OfferJob}(a, r, e) \cdot p_{\text{IsMale}}(a) \cdot p_{\text{ColRank}}(r) \cdot p_{\text{YearsExp}}(e) \cdot da \cdot dr \cdot de$$

$$\text{OfferJob} = (\text{ColRank} \leq 5 \vee \text{YearsExp} > 5)$$

$$= (\text{ColRank} \leq 5) \vee (\text{ColRank} > 5 \wedge \text{IsMale} \wedge \text{YearsExpLarge} > 5)$$

$$\vee (\text{ColRank} > 5 \wedge \neg\text{IsMale} \wedge \text{YearsExpSmall} > 5)$$

$$\Pr[\text{OfferJob}]$$

$$= \Pr[\text{ColRank} \leq 5] + \Pr[\text{ColRank} > 5 \wedge \text{IsMale} \wedge \text{YearsExpLarge} > 5]$$

$$+ \Pr[\text{ColRank} > 5 \wedge \neg\text{IsMale} \wedge \text{YearsExpSmall} > 5]$$

$$= \Pr[\text{ColRank} \leq 5] + \Pr[\text{ColRank} > 5 \mid] \cdot \Pr[\text{IsMale}] \cdot \Pr[\text{YearsExpLarge} > 5]$$

$$+ \Pr[\text{ColRank} > 5\ ] \cdot \Pr[\neg\text{IsMale}] \cdot \Pr[\text{YearsExpSmall} > 5]$$

```
def population_model():
    is_male ~ bernoulli(0.5)
    col_rank ~ normal(25, 10)
    if is_male:
        years_exp ~ normal(15, 5)
    else:
        years_exp ~ normal(10, 5)
    return col_rank, years_exp


def offer_job(col_rank, years_exp)
    if col_rank <= 5:
        return true
    elif years_exp > 5:
        return true
    else:
        return false
```

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

$$\Pr[\text{OfferJob}]$$

$$= \int \text{OfferJob}(a, r, e) \cdot p_{\text{IsMale}}(a) \cdot p_{\text{ColRank}}(r) \cdot p_{\text{YearsExp}}(e) \cdot da \cdot dr \cdot de$$

$$\text{OfferJob} = (\text{ColRank} \leq 5 \vee \text{YearsExp} > 5)$$

$$= (\text{ColRank} \leq 5) \vee (\text{ColRank} > 5 \wedge \text{IsMale} \wedge \text{YearsExpLarge} > 5)$$

$$\vee (\text{ColRank} > 5 \wedge \neg\text{IsMale} \wedge \text{YearsExpSmall} > 5)$$

$$\Pr[\text{OfferJob}]$$

$$= \Pr[\text{ColRank} \leq 5] + \Pr[\text{ColRank} > 5 \wedge \text{IsMale} \wedge \text{YearsExpLarge} > 5]$$

$$+ \Pr[\text{ColRank} > 5 \wedge \neg\text{IsMale} \wedge \text{YearsExpSmall} > 5]$$

$$= \Pr[\text{ColRank} \leq 5] + \Pr[\text{ColRank} > 5 \mid] \cdot \Pr[\text{IsMale}] \cdot \Pr[\text{YearsExpLarge} > 5]$$

$$+ \Pr[\text{ColRank} > 5\ ] \cdot \Pr[\neg\text{IsMale}] \cdot \Pr[\text{YearsExpSmall} > 5]$$

$$= N(5; 25{,}10) + \big(1 - N(5; 25{,}10)\big) \cdot 0.5 \cdot \big(1 - N(5; 15{,}5)\big)$$

$$+ \big(1 - N(5; 25{,}10)\big) \cdot 0.5 \cdot \big(1 - N(5; 10{,}5)\big)$$

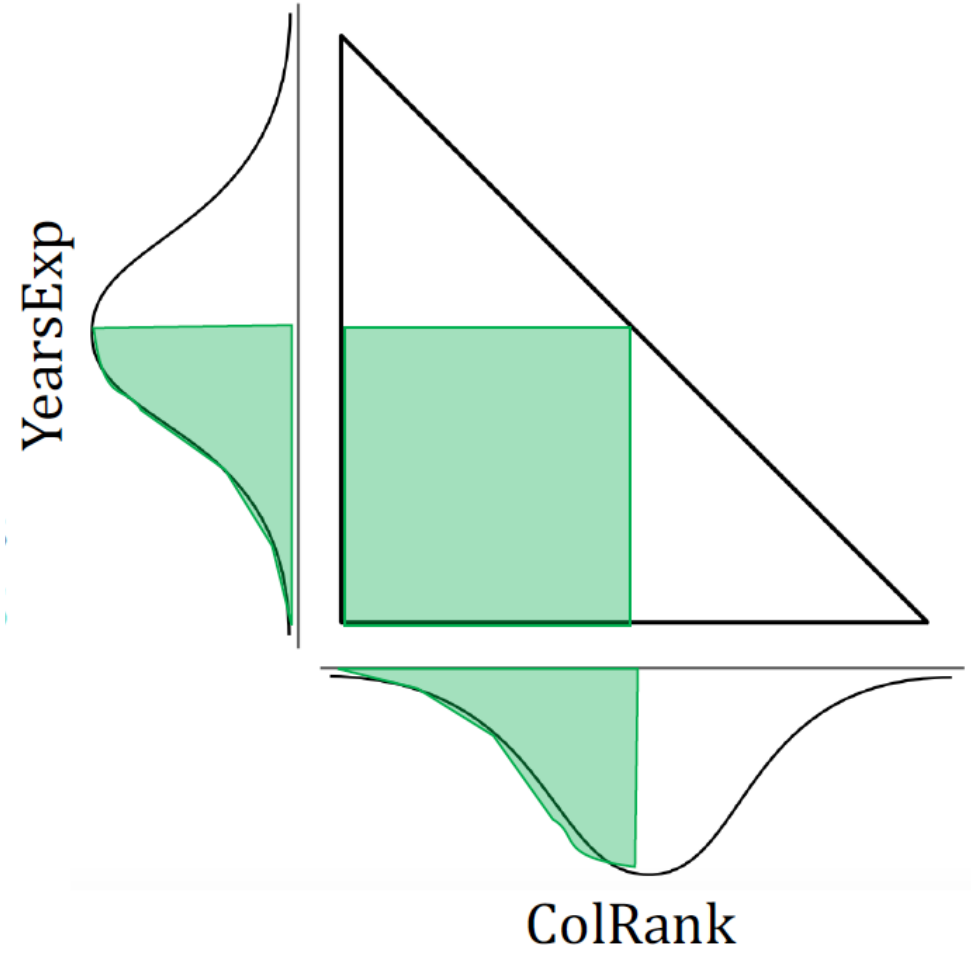
```
def population_model():
    is_male ~ bernoulli(0.5)
    col_rank ~ normal(25, 10)
    if is_male:
        years_exp ~ normal(15, 5)
    else:
        years_exp ~ normal(10, 5)
    return col_rank, years_exp


def offer_job(col_rank, years_exp)
    if col_rank <= 5:
        return true
    elif years_exp > 5:
        return true
    else:
        return false
```

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Fairness Verification Strategy

- **Alternative example**
  - $\text{OfferJob} = (\text{ColRank} + \text{YearsExp}) \leq 10$
  - Assume $\text{ColRank} \sim N(25,10)$ and $\text{YearsExp} \sim N(15,5)$
  - **Goal:** Compute $\Pr[\text{OfferJob}]$

- In high-dimensional spaces, probability integration is hard.
- Computers are very fast at calculating probabilities inside simple rectangles compared to complex, curvy arbitrary shapes.
- Solution: Split complex regions into simple geometric shapes.

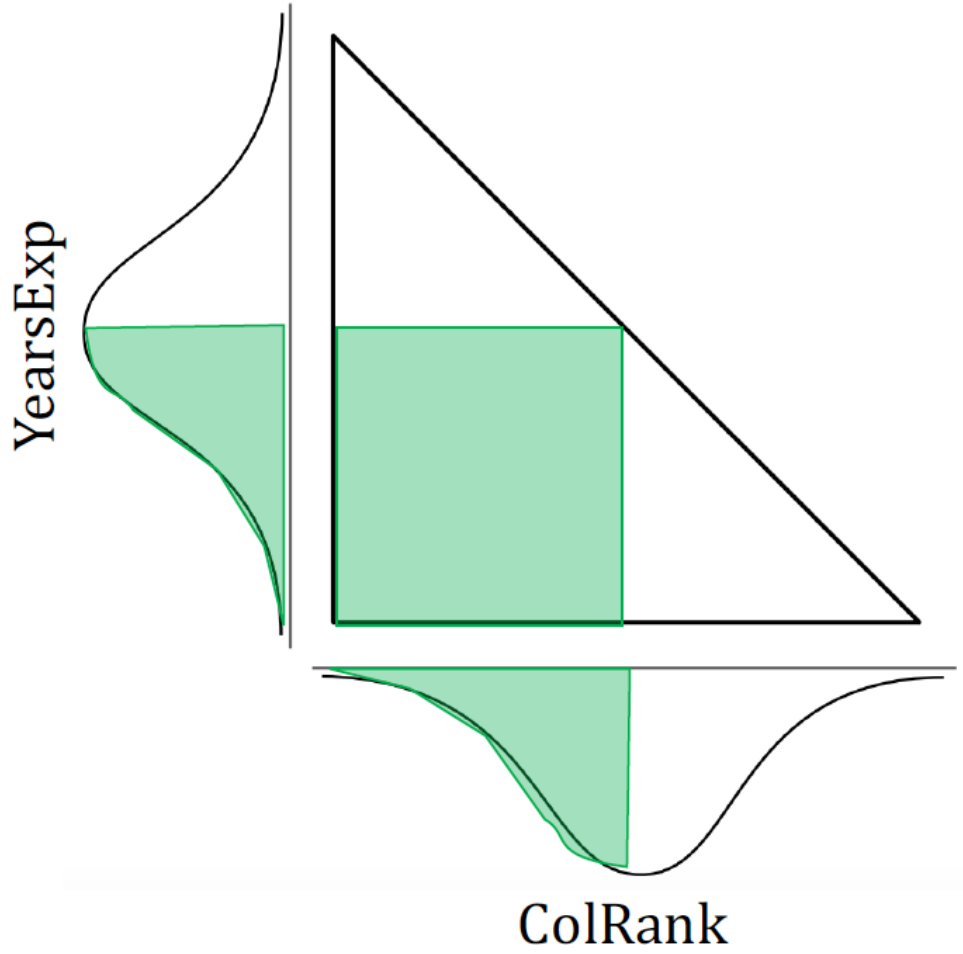


Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Fairness Verification Strategy

- **Alternative example**
  - $\text{OfferJob} = (\text{ColRank} + \text{YearsExp}) \leq 10$
  - Assume $\text{ColRank} \sim N(25,10)$ and $\text{YearsExp} \sim N(15,5)$
  - **Goal:** Compute $\Pr[\text{OfferJob}]$

- **Idea:** Break OfferJob into hyperrectangles
  - $R_1 = 0 \leq \text{ColRank} \leq 5 \wedge 0 \leq \text{YearsExp} \leq 5$
  - $\Pr[R_1] = \left(N(5; 25,10) - N(0; 25,10)\right) \cdot \left(N(5; 15,5) - N(0; 15,5)\right)$

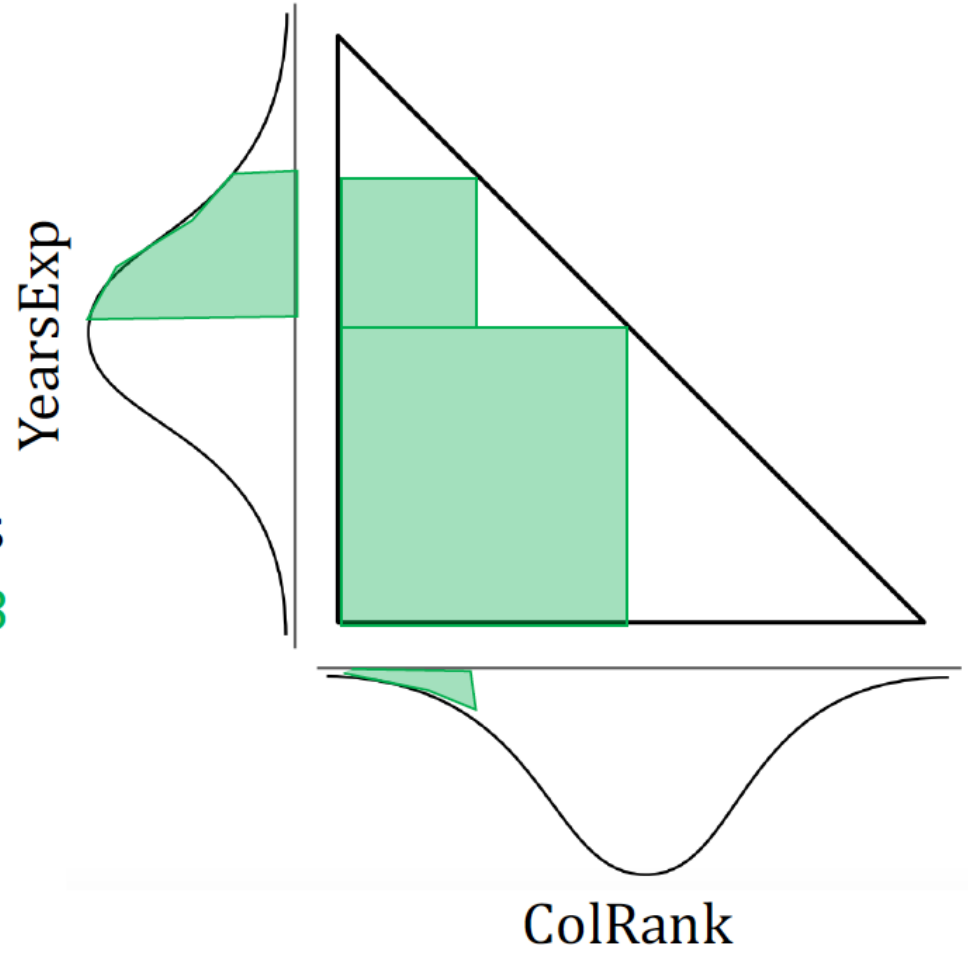


Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Fairness Verification Strategy

- **Alternative example**
  - $\text{OfferJob} = (\text{ColRank} + \text{YearsExp}) \leq 10$
  - Assume $\text{ColRank} \sim N(25,10)$ and $\text{YearsExp} \sim N(15,5)$
  - **Goal:** Compute $\Pr[\text{OfferJob}]$

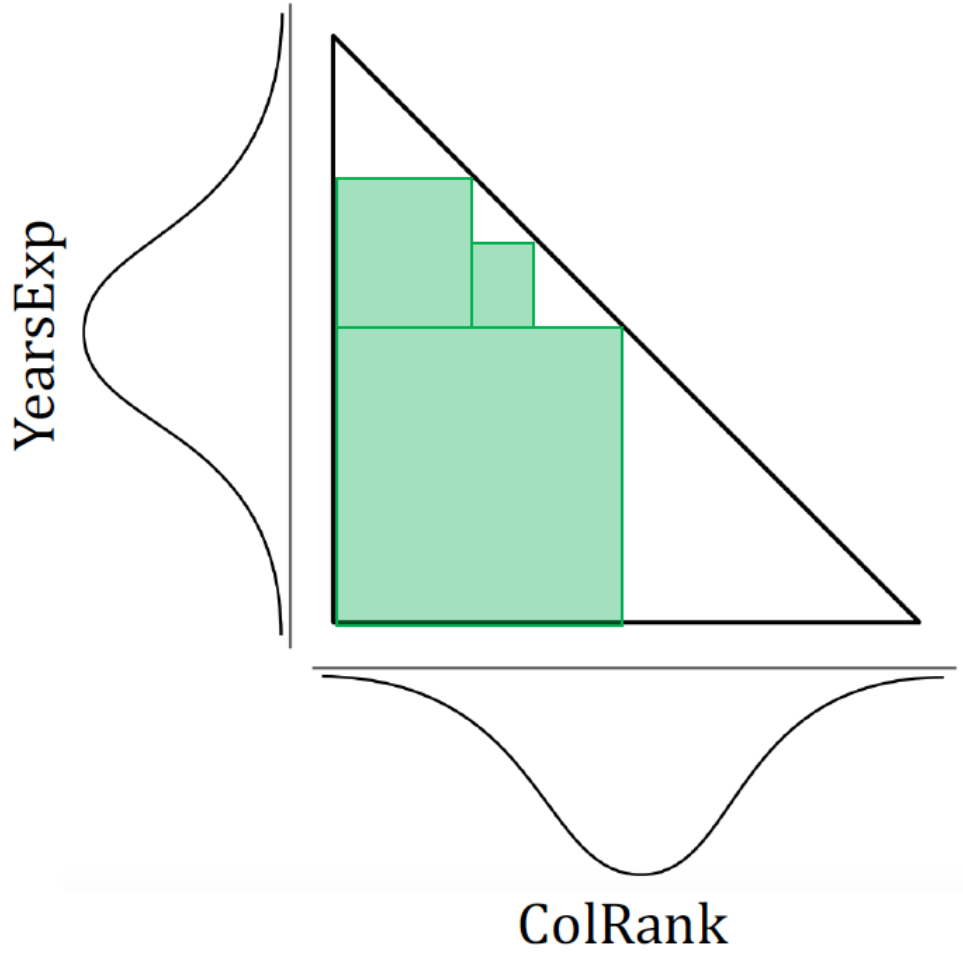
- **Idea:** Break OfferJob into hyperrectangles
  - $R_2 = 0 \leq \text{ColRank} \leq 2 \wedge 5 \leq \text{YearsExp} \leq 8$
  - $\Pr[R_2] = (N(2; 25,10) - N(0; 25,10) \cdot \big(N(8; 15,5) - N(5; 15,5)\big)$



Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Fairness Verification Strategy

- **Alternative example**
  - $\text{OfferJob} = (\text{ColRank} + \text{YearsExp}) \leq 10$
  - Assume $\text{ColRank} \sim N(25,10)$ and $\text{YearsExp} \sim N(15,5)$
  - **Goal:** Compute $\Pr[\text{OfferJob}]$
- **Idea:** Break OfferJob into hyperrectangles
  - $\Pr[\text{OfferJob}] = \Pr[R_1] + \Pr[R_2] + \cdots$



Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Fairness Verification Algorithm

for $t \in \{1,2, \ldots\}$: # Iterate over timesteps

  for $i \in \{1, \ldots, k\}$ # Iterate over regions/partitions

    compute rectangle $R_{i,t}$ for ϕ_i

    compute estimate $\hat{\mu}_a \approx \mu_a^*$ using $R_{i,t}$

    compute estimate $\hat{Y} \approx Y^*_{\text{parity}}$ using $\hat{\mu}_a$

    if converged: **return** $\hat{Y}$

See the paper for more details: Albarghouthi, A., D'Antoni, L., Drews, S., & Nori, A. V. (2017). Fairsquare: probabilistic verification of program fairness. *Proceedings of the ACM on Programming Languages*, *1*(OOPSLA), 1-30.

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Shortcomings of Symbolic Verification

- Scales poorly to large models
  - Neural networks now have billions of parameters!
  - Assumes access to the decision code and the population model!
- Fairness is a statistical property

- Can we use a statistical approach to verify fairness?

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Statistical Verification

- Use random sampling to check correctness, and use statistical tools to bound probability of false negatives

$$\Pr_{X^{(1)},\ldots,X^{(n)}\sim P_X}\left[f \text{ correct} \mid \mathcal{A}\left(f; X^{(1)},\ldots,X^{(n)}\right) = \text{correct}\right] \geq 1-\delta$$

(is it really fair?) (Algorithm says it is fair)

- Guarantee of symbolic verification is equivalent to $\delta = 0$

(since chance of error is zero)

- For statistical verification, some chance of error is inevitable ($\delta > 0$), but we can make δ as small as desired with sufficiently many samples

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Statistical Verification for Fairness

- Given samples $v_a^{(1)}, \ldots, v_a^{(n)} \sim P_V \mid A = a$ (for each a)
  - Obtained via rejection sampling
- The estimated acceptance probability is

$$\hat{\mu}_a = \frac{1}{n} \sum_{i=1}^{n} 1\left[f\left(v_a^{(i)}\right) = 1\right]$$

- We can bound $|\hat{\mu}_a - \mu_a^*|$ using **Hoeffding's inequality**

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Hoeffding's Inequality

- Let $b_1, \ldots, b_n \sim_{\text{i.i.d.}} \text{Bernoulli}(\mu)$ be samples
- Let $\hat{\mu} = n^{-1} \sum_{k=1}^{n} b_k$ be the empirical mean
- Then, with probability at least $1 - \delta$, we have

$$|\hat{\mu} - \mu| \leq \sqrt{\frac{\log(2/\delta)}{2n}}$$

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Hoeffding's Inequality

- Apply Hoeffding's inequality to $\mu_a^* = \Pr[f(V) = 1 \mid A = a]$
- Ensures that $\hat{\mu}_a$ is "good estimate" of μ_a^* with high probability:

$$\Pr\left[|\hat{\mu}_a - \mu_a^*| \leq \sqrt{\frac{\log(2/\delta)}{2n}}\right] \geq 1 - \delta$$

$$\hat{\mu}_a = \frac{1}{n}\sum_{i=1}^{n} 1\left[f\left(v_a^{(i)}\right) = 1\right]$$

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Algorithm

for $i \in \{1,2,\ldots,n\}$:

sample individual $v_a^{(i)} \sim P_V \mid A = a$ (for each a)

obtain high-probability bound $\hat{\mu}_a \leq \mu_a^* \leq \hat{\mu}'_a$ using Hoeffding's inequality

compute estimate $Y^*_{\text{parity}} \in \gamma(\hat{Y})$ using $\hat{\mu}_a, \hat{\mu}'_a$

return $\hat{Y}$

$$\hat{\mu}_a = \frac{1}{n}\sum_{i=1}^{n} 1\left[f\left(v_a^{(i)}\right) = 1\right]$$

$$|\hat{\mu}_a - \mu_a^*| \leq \sqrt{\frac{\log(2/\delta)}{2n}}$$

Denote this by ϵ

Combine the following:

1. $\hat{\mu}_a - \mu_a^* > 0$ => $\hat{\mu}_a - \mu_a^* \leq \epsilon$
   => $\hat{\mu}_a - \epsilon \leq \mu_a^*$

2. $\hat{\mu}_a - \mu_a^* < 0$ => $-(\hat{\mu}_a - \mu_a^*) \leq \epsilon$
   => $\mu_a^* \leq \hat{\mu}_a + \epsilon$

Algorithm

for $i \in \{1,2,\ldots,n\}$:

sample individual $v_a^{(i)} \sim P_V \mid A = a$ (for each a)

obtain high-probability bound $\hat{\mu}_a \leq \mu_a^* \leq \hat{\mu}'_a$ using Hoeffding's inequality

compute estimate $Y^*_{\text{parity}} \in \gamma(\hat{Y})$ using $\hat{\mu}_a, \hat{\mu}'_a$

return $\hat{Y}$

$$Y^*_{\text{parity}} = 1\left[\frac{\mu^*_{\text{minority}}}{\mu^*_{\text{majority}}} \geq c\right]$$

- Calculate the lower bounds (L_a and L_b) and upper bounds (U_a and U_b) for each group
- Given **Parity** $= \frac{\hat{\mu}_{\text{minority}}}{\hat{\mu}_{\text{majority}}}$, we can use the lower ($L_a$ and L_b) & upper bounds (U_a and U_b) to:

$$\frac{L_a}{U_b} \leq \frac{\mu^*_{\text{minority}}}{\mu^*_{\text{majority}}} \leq \frac{U_a}{L_b}$$

uncertain

$$\gamma \in \{0, 1, \{0,1\}\}$$

Modified from: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Algorithm

for $i \in \{1,2,\ldots,n\}$**:**

sample individual $v_a^{(i)} \sim P_v \mid A = a$ (for each a)

obtain high-probability bound $\hat{\mu}_a \leq \mu_a^* \leq \hat{\mu}'_a$ using Hoeffding's inequality

compute estimate $Y^*_{\text{parity}} \in \gamma(\hat{Y})$ using $\hat{\mu}_a, \hat{\mu}'_a$

return $\hat{Y}$

$$Y^*_{\text{parity}} = 1\left[\frac{\mu^*_{\text{minority}}}{\mu^*_{\text{majority}}} \geq c\right]$$

Interval: [0.75, 0.85]
Threshold: 0.80
Problem: The true value could be 0.79 (Unfair) **OR** 0.81 (Fair).
γ**:** {Fair, Unfair}

Interval: [0.82, 0.85]
Threshold: 0.80
Result: Every possible value in the interval is greater than 0.80.
γ: {Fair}

Modified from: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Adaptive Concentration Inequalities

- **Key Shortcoming**
  - In Hoeffding's inequality, number of samples n must be chosen beforehand
  - Algorithm may not converge (i.e., $\hat{Y} = \text{uncertain}$)
  - In practice, we often have a fixed test set!

- **Idea:** Try increasing values of n until one works
  - **Problem:** Need a union bound!

- **Solution:** Use a concentration inequality that allows us to iteratively take more samples

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Statistical Verification

- Use an *adaptive* variant of Hoeffding's, which lets us incrementally increase n and still maintain the guarantee

- **Adaptive Concentration:** With probability $\geq 1 - \delta$, we have

$$\forall n \,.\, \Pr\left[\left|\hat{\mu}_a^{(n)} - \mu_a^*\right| \leq \epsilon(n, \delta)\right]$$

- Here, $\hat{\mu}_a^{(n)} = n^{-1} \sum_{i=1}^{n} 1\left[f\left(v_a^{(i)}\right) = 1\right]$

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Adaptive Concentration Inequalities

THEOREM 4.1. *Given a Bernoulli random variable Z with distribution $P_{\mathcal{Z}}$, let $\{Z_i \sim P_{\mathcal{Z}}\}_{i \in \mathbb{N}}$ be i.i.d. samples of Z, let*

$$\hat{\mu}_Z^{(n)} = \frac{1}{n} \sum_{i=1}^{n} Z_i,$$

let J be a random variable on $\mathbb{N} \cup \{\infty\}$ such that $Pr[J < \infty] = 1$, and let

$$\varepsilon(\delta, n) = \sqrt{\frac{\frac{3}{5} \cdot \log(\log_{11/10} n + 1) + \frac{5}{9} \cdot \log(24/\delta)}{n}}. \tag{10}$$

Then, given any $\delta \in \mathbb{R}_+$, we have

$$Pr[|\hat{\mu}_Z^{(J)} - \mu_Z| \leq \varepsilon(\delta, J)] \geq 1 - \delta.$$

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Algorithm

for $i \in \{1,2, \ldots\}$**:**

sample individual $v_a^{(i)} \sim P_{\mathcal{V}} \mid A = a$ (for each a)

obtain high-probability bound $\hat{\mu}_a \leq \mu_a^* \leq \hat{\mu}'_a$

compute estimate $Y^*_{\text{parity}} \in \gamma\left(\hat{Y}\right)$ using $\hat{\mu}_a, \hat{\mu}'_a$

if $\left|\gamma\left(\hat{Y}\right)\right| = 1$: **return** $\hat{Y}$

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Algorithm

for $i \in \{1,2, \ldots\}$:

sample individual $v_a^{(i)} \sim P_V \mid A = a$ (for each a)

obtain high-probability bound $\hat{\mu}_a \leq \mu_a^* \leq \hat{\mu}'_a$

compute estimate $Y^*_{\text{parity}} \in \gamma(\hat{Y})$ using $\hat{\mu}_a, \hat{\mu}'_a$

if $|\gamma(\hat{Y})| = 1$: **return** $\hat{Y}$

Interval: [0.75, 0.85]
Threshold: 0.80
Problem: The true value could be 0.79 (Unfair) **OR** 0.81 (Fair).
Set of Outcomes: {Fair, Unfair}
Cardinality (Size): 2.
Action: The algorithm must continue

Interval: [0.82, 0.85]
Threshold: 0.80
Result: Every possible value in the interval is greater than 0.80.
Set of Outcomes: {Fair}
Cardinality (Size): 1.
Action: Stop and return the result.

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Theoretical Guarantees

- **Probabilistic correctness**

$$\Pr_{X^{(1)},\ldots,X^{(n)}\sim P_X}\left[f \text{ correct} \mid \mathcal{A}\left(f; X^{(1)},\ldots,X^{(n)}\right) = \text{correct}\right] \geq 1-\delta$$

$$\Pr_{X^{(1)},\ldots,X^{(n)}\sim P_X}\left[f \text{ incorrect} \mid \mathcal{A}\left(f; X^{(1)},\ldots,X^{(n)}\right) = \text{incorrect}\right] \geq 1-\delta$$

- **Probabilistic termination**
  - Assume fairness does not "just barely" hold
  - Then, with probability 1, terminates after finitely many steps

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000

Value of Verification

- Concentration inequalities can give you provable guarantees for statistical properties

- What is the value of verification over directly using the estimate $\hat{Y}_{\text{parity}} = 1\left[\frac{\hat{\mu}_{\text{minority}}}{\hat{\mu}_{\text{majority}}} \geq c\right]$?
  - Verification quantifies uncertainty in our estimate of fairness
  - Do not mis-report fair or unfair due to too few samples

Slide: Uni of Pennsylvania, Trustworthy ML - CIS 7000