On a bilevel optimization approach to fair classification

Kangwook Lee @ UW Madison

Joint work with Yuji Roh, Euijong Whang, Changho Suh (KAIST), Yuchen Zeng, Ziqian Lin (UW Madison)

Jan 9, 2023 @ 12th US-Mexico Workshop on Optimization and Its Applications

Research overview

Theory ⟷ Systems

- IT & SP & Queueing & OPT
  - ICLR'22 (SGD)
  - ICML'21 (matrix comp.)
  - NeurIPS'18 (binary matrix comp.)
  - IEEE T-IT'19 (graph clustering)
  - IEEE T-IT'19 (group testing)
  - IEEE JSTSP'18 (graph clustering)
  - IEEE T-IT'17 (phase retrieval)
  - IEEE T-IT'17 (MDS queue)
  - IEEE T-N'17 (task replica)
  - IEEE T-C'16 (task replica)

- Trustworthy ML
  - ISIT'22 (adversarial attack)
  - NeurIPS'21a (fair + robust)
  - NeurIPS'21b (data leakage)
  - ICLR'21 (bilevel opt.)
  - ICML'20 (mutual information)
  - ICML'22 (adv. robustness)
  - NeurIPS'20 (data poisoning)
  - AAAI'19 (domain gen.)
  - ICLR'18 (domain gen.)

- Large ML models
  - NeurIPS'22a (diffusion)
  - NeurIPS'22b (GPT3)
  - NeurIPS'22c (model pruning)
  - EMNLP'22 (translation)

- Distributed ML (coded comp.)
  - ICML'21 (coded deep learning)
  - MLSys'21 (grad. compression)
  - SysML'18 (data shuffling)
  - IEEE T-IT'18 (MDS codes)

Bilevel optimization for machine learning

- Various applications in machine learning
  - Hyper-parameter optimization [KLS, *ICMLW'19*]
  - Multi-task and meta learning (e.g., finding a good initialization) [KJLOO, *NeurIPS'21*]
  - Neural Architecture Search (NAS)
  - Data poisoning [WSRVASLP, *NeurIPS'20*]

This talk

A new ML application of bilevel optimization + a tailored algorithm

- Various applications in machine learning
  - Hyper-parameter optimization [KLS, *ICMLW'19*]
  - Multi-task and meta learning (e.g., finding a good initialization) [KJLOO, *NeurIPS'21*]
  - Neural Architecture Search (NAS)
  - Data poisoning [WSRVASLP, *NeurIPS'20*]
  - ML Fairness [Roh, Lee, Whang, and Suh, *ICLR'21*]

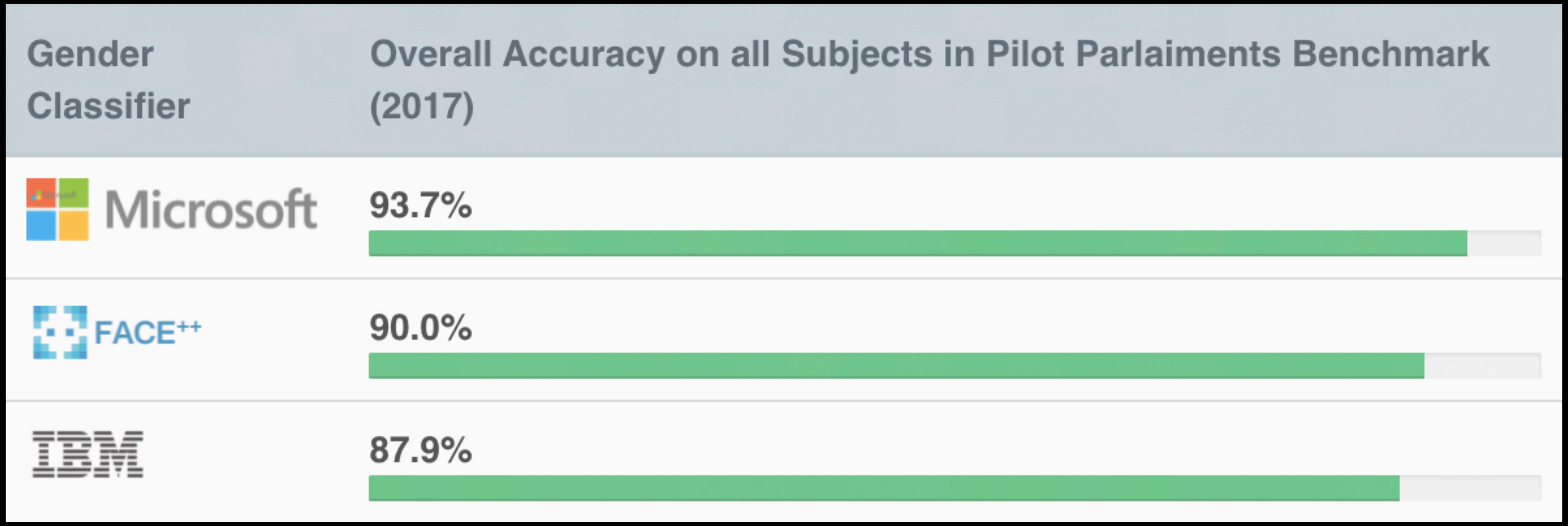
ML Fairness

- Setting
  - Consider classification for simplicity
  - Also consider scenarios where the input data comes from individuals
- Example
  - Facial recognition for security systems
  - Resume screening for recruiting
  - Recidivism prediction for pretrial decision making
- Accuracy alone is not sufficient…
  - Learned classifiers are observed to disproportionally treat different subpopulations

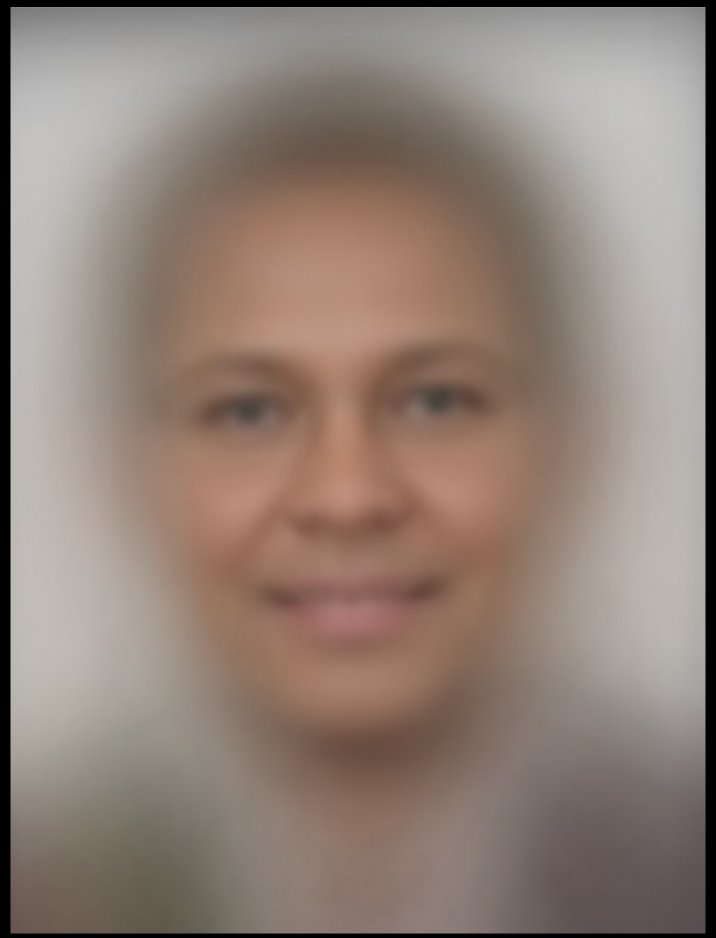
ML Fairness

Examples: Face-to-gender classification

http://gendershades.org/



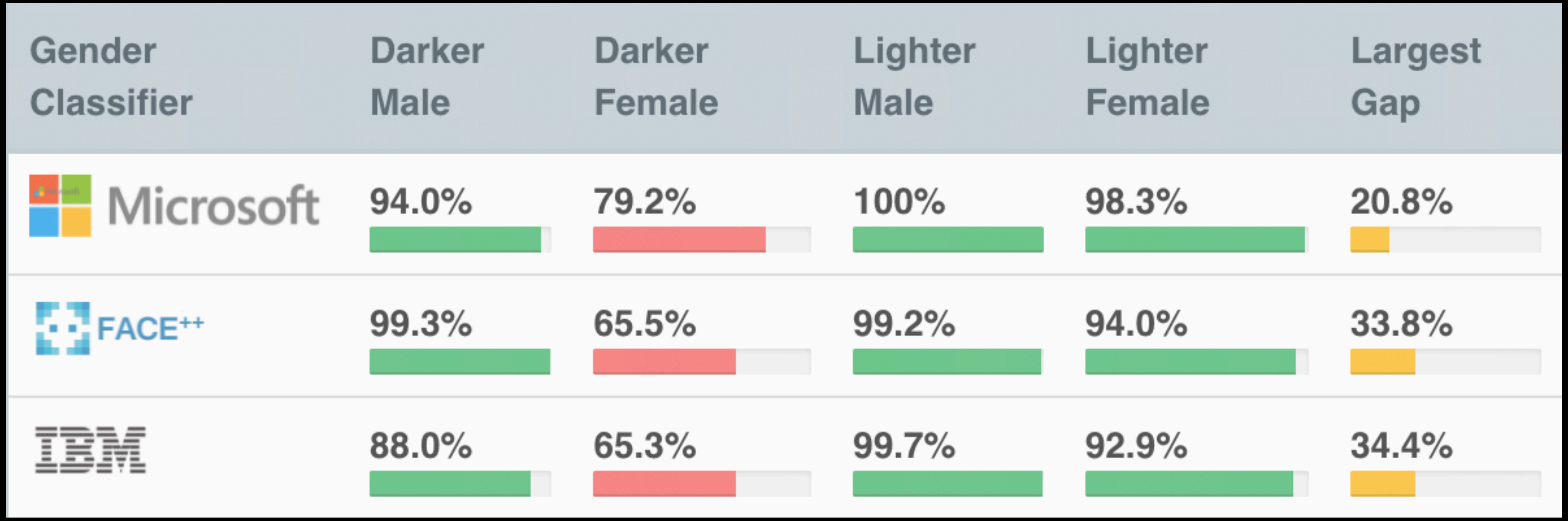
Gender Classifier	Overall Accuracy on all Subjects in Pilot Parlaiments Benchmark (2017)
Microsoft	93.7%
FACE++	90.0%
IBM	87.9%



ML Fairness

Examples: Face-to-gender classification

http://gendershades.org/



Gender Classifier	Darker Male	Darker Female	Lighter Male	Lighter Female	Largest Gap
Microsoft	94.0%	79.2%	100%	98.3%	20.8%
FACE++	99.3%	65.5%	99.2%	94.0%	33.8%
IBM	88.0%	65.3%	99.7%	92.9%	34.4%

Why? Data & algorithmic bias

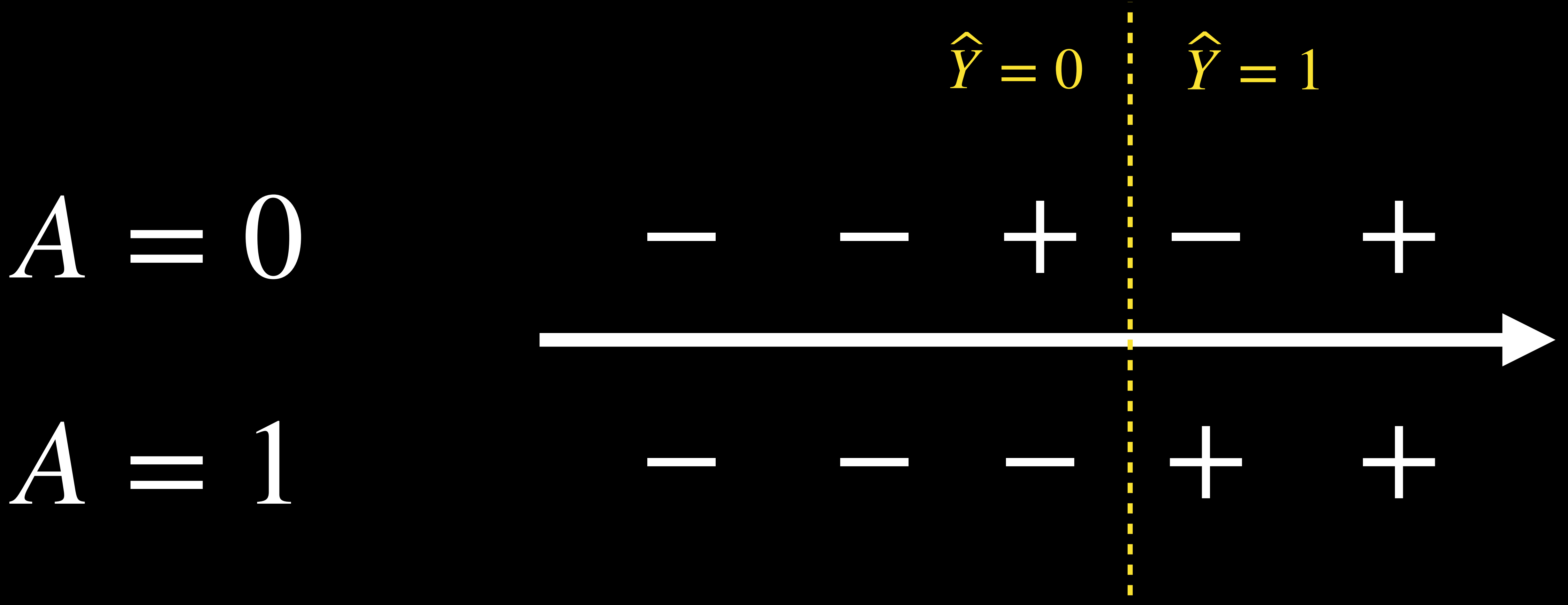
Group Fairness

- Notation
  - $Y \in \{0,1\}$: True labels
  - $\widehat{Y} \in \{0,1\}$: Predicted labels
  - $A \in \{0,1\}$: Group labels (e.g., male/female)
- Group fairness of a (binary) classifier can be defined in various ways:
  - Accuracy parity: $P(\widehat{Y} = Y \mid A = 0) = P(\widehat{Y} = Y \mid A = 1)$
  - Demographic parity: $P(\widehat{Y} = 1 \mid A = 0) = P(\widehat{Y} = 1 \mid A = 1)$
  - Equal opportunity: $P(\widehat{Y} = 1 \mid A = 0, Y = 1) = P(\widehat{Y} = 1 \mid A = 1, Y = 1)$
- Unfairness is usually measured as the absolute difference between the two terms

Group Fairness

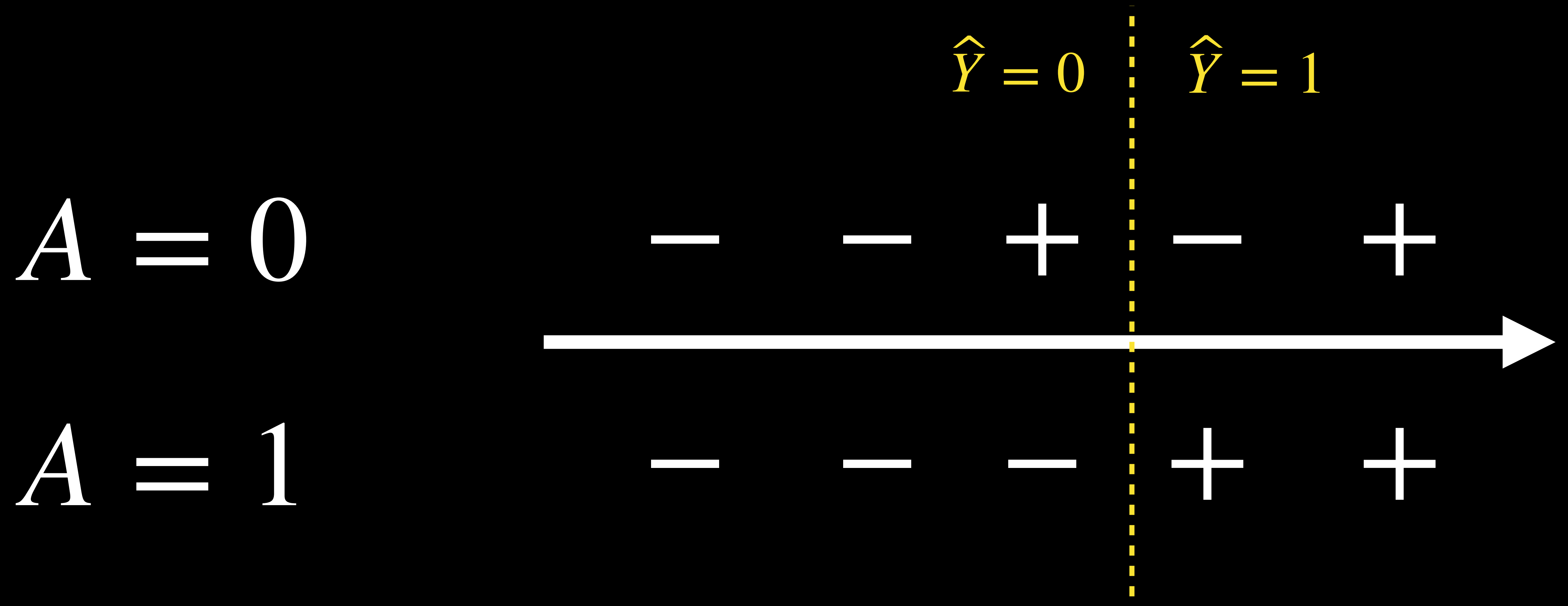
- Notation
  - $Y \in \{0,1\}$: True labels
  - $\hat{Y} \in \{0,1\}$: Predicted labels
  - $A \in \{0,1\}$: Group labels (e.g., male/female)
- Group fairness of a (binary) classifier can be defined in various ways:
  - Accuracy parity: $P(\hat{Y} = Y | A = 0) = P(\hat{Y} = Y | A = 1)$
  - Demographic parity: $P(\hat{Y} = 1 \,|\, A = 0) = P(\hat{Y} = 1 \,|\, A = 1)$
  - Equal opportunity: $P(\hat{Y} = 1 \,|\, A = 0, Y = 1) = P(\hat{Y} = 1 \,|\, A = 1, Y = 1)$
- Unfairness is usually measured as the absolute difference between the two terms
  - Many other definitions exist: variance, CVaR, …

Example: 1d-threshold classifier



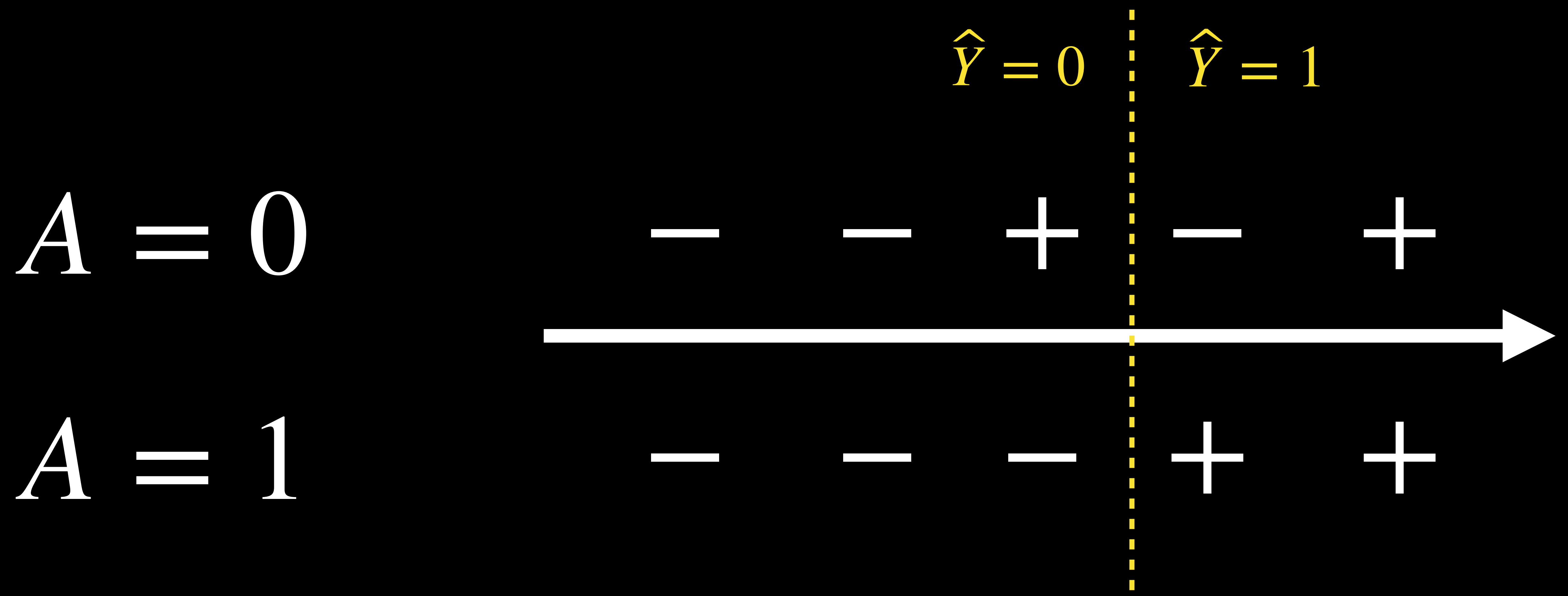
(error rate, unfairness)?

Example: 1d-threshold classifier



$$\text{(error rate, unfairness)} = \left(\frac{2}{10}, \left| \frac{2}{5} - \frac{0}{5} \right| \right)$$

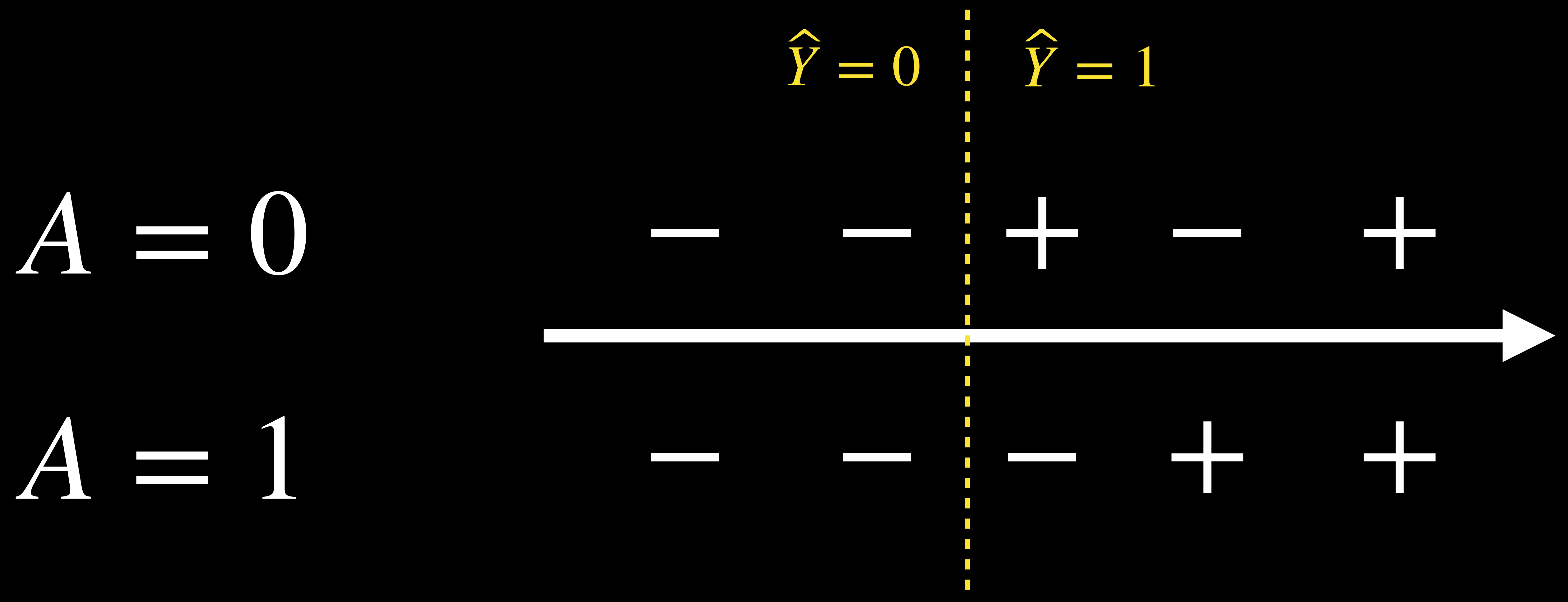
Example: 1d-threshold classifier



$\hat{Y} = 0$
$\hat{Y} = 1$
$A = 0$
$- \quad - \quad + \quad - \quad +$
$A = 1$
$- \quad - \quad - \quad + \quad +$

$$(\text{error rate, unfairness}) = (0.2,\ 0.4)$$

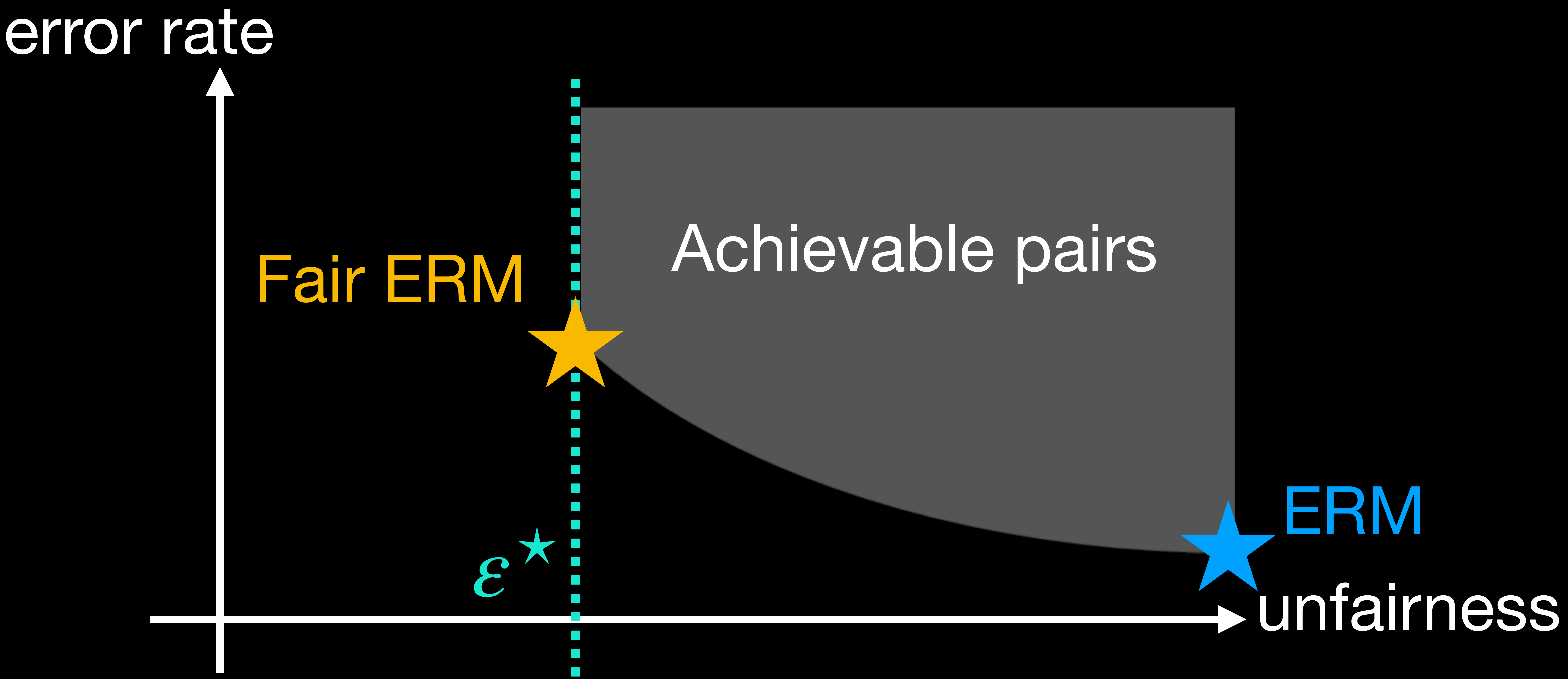
Example: 1d-threshold classifier



$$(\text{error rate, unfairness}) = (0.2,\ 0)$$

Goal

- Find the most fair classifiers and then find the most accurate one among them



Problem setting

- Notation
  - $A \in \{0,1\}$: Group labels (e.g., male/female)
  - L_A: Loss measured on the subgroup A's data
  - $L = L_0 + L_1$
- In this talk, for simplicity, we will consider the loss parity (= accuracy parity for 0/1 loss)
  - $L_0 = L_1$

Existing algorithms

- Pre-processing & post-processing
- Min-max formulation — exponentiated gradient [Agarwal et al., ’18]
- Adversarial training — auxiliary classifier for predicting group label [Zhang et al., 18]
- Distributionally robust opt. [Hashimoto et al., 18]
- Mutual information surrogate [RLWS, *ICML’20*]

A very simple baseline beats everything (?)

Hey, I just tried out the following algorithm last week, and it beat all SOTA algorithms both in performance & time

Yuji Roh (PhD student from the *data-centric AI* lab @ KAIST)

?

1. Train a model with vanilla SGD for many iterations
2. Measure L_0 and L_1
3. If $L_0 > L_1$: Continue training with minibatches more group 0 data
   (Why? The model is performing not well on group 0 so let's feed more group 0 data)

   Otherwise: Continue training with minibatches with more group 1 data

I even gave it a name, ***FairBatch***. Could you explain why FairBatch works?

Initial formulation gave me a constrained opt.

$$\min_{\theta} L(\theta) \qquad \text{s.t. } |L_0(\theta) - L_1(\theta)| = \varepsilon^\star$$

We don't know $\varepsilon^\star$

Key observations

$$\min_{\theta} L(\theta) \qquad \text{s.t. } |L_0(\theta) - L_1(\theta)| = \varepsilon$$

$$g = \min_{\theta} L(\theta) + \lambda'(L_0(\theta) - L_1(\theta) - \varepsilon) + \lambda''(L_0(\theta) - L_1(\theta) + \varepsilon)$$

$$= \min_{\theta} L_0(\theta) + L_1(\theta) + (\lambda' + \lambda'')(L_0(\theta) - L_1(\theta)) - (\lambda' - \lambda'')\varepsilon$$

$$\lambda := \lambda' + \lambda''$$

$$= \min_{\theta} (1 + \lambda)L_0(\theta) + (1 - \lambda)L_1(\theta) - (\lambda' - \lambda'')\varepsilon$$

Key observations

$$\min_{\theta} L(\theta) \qquad \text{s.t. } |L_0(\theta) - L_1(\theta)| = \varepsilon$$

$$g = \min_{\theta} (1+\lambda) L_0(\theta) + (1-\lambda) L_1(\theta) - (\lambda' - \lambda'')\varepsilon$$

1. The optimal model parameter θ can be found by simply minimizing a weighted objective function: $(1+\lambda)L_0(\theta) + (1-\lambda)L_1(\theta)$ for properly chosen λ

Key observations

$$\min_{\theta} L(\theta) \qquad \text{s.t. } |L_0(\theta) - L_1(\theta)| = \varepsilon$$

$$g = \min_{\theta}(1 + \lambda)L_0(\theta) + (1 - \lambda)L_1(\theta) - (\lambda' - \lambda'')\varepsilon$$

2. Instead of $\varepsilon \to (\lambda_\varepsilon, \theta_\varepsilon)$,
we can use $\lambda \to \theta_\lambda \to \varepsilon_\lambda$

Tada! A bilevel formulation

$$\min_{\theta} L(\theta) \quad \text{s.t. } |L_0(\theta) - L_1(\theta)| = \varepsilon^\star$$

$$\Rightarrow \quad \min_{\lambda} \varepsilon(\lambda) = \Big| L_0(\theta^\star) - L_1(\theta^\star) \Big|$$

$$\theta^\star = \arg\min_{\theta} (1 + \lambda) L_0(\theta) + (1 - \lambda) L_1(\theta)$$

Bilevel optimization for machine learning

$$\min_{\lambda} F(\lambda, \theta^\star)$$

$$\theta^\star = \arg\min_{\theta} L(\lambda, \theta)$$

Bilevel optimization for machine learning

- Various applications in machine learning
  - Hyper-parameter optimization
  - Multi-task and meta learning (e.g., finding a good initialization)
  - Neural Architecture Search (NAS)
  - Data poisoning

Bilevel optimization for machine learning

Example: Hyper-parameter optimization [Franceschi et al., *'18*]

$$\min_{\theta} L_{\text{train}}(\theta) + \lambda R(\theta)$$

How can I choose λ?

Bilevel optimization for machine learning

Example: Hyper-parameter optimization [Franceschi et al.,*’18*]

$$\min_{\lambda} L_{\text{val}}(\theta^\star)$$

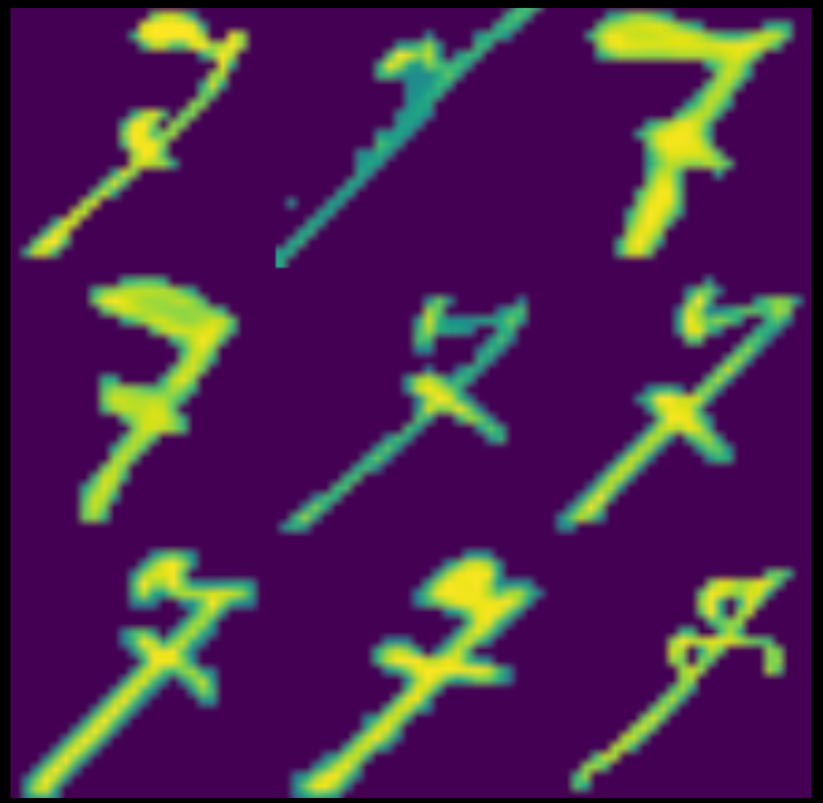
$$\theta^\star = \arg\min_{\theta} L_{\text{train}}(\theta) + \lambda R(\theta)$$

Bilevel optimization for machine learning

Example: Data poisoning [Biggio et al., *'12*] [Steinhard et al.,*'17*]

True label: airplane

Predicted label: truck



True label: 7

Predicted label: 1

[WSRVASLP, *NeurIPS'20*]

Bilevel optimization for machine learning

Example: Data poisoning [Biggio et al., *'12*] [Steinhard et al.,*'17*]

- Learner: minimize the loss computed on the dataset
- Attacker: manipulate the dataset so that the learned model behaves as desired

$$\min_{D_p} d(\theta_{\text{target}}, \theta^\star)$$

$$\theta^\star = \arg\min_{\theta} L(D \cup D_p; \theta)$$

Bilevel Optimization Algorithms

$$\min_{\lambda} F(\lambda, \theta^\star)$$

$$\theta^\star = \arg\min_{\theta} L(\lambda, \theta)$$

Bilevel Optimization Algorithms

$$\min_{\lambda} F(\lambda, \theta^\star)$$

$$\theta^\star = \arg\min_{\theta} L(\lambda, \theta)$$

$$\min_{\lambda} F(\lambda, \theta^\star)$$

s.t. $G(\lambda, \theta^\star) = 0$

- Constraint-based approaches [Hansen et al. (1992); Shi et al. (2005); Moore (2010)]

Bilevel Optimization Algorithms

$$\min_{\lambda} F(\lambda, \theta^\star)$$

$$\nabla F(\lambda, \theta^\star) = \nabla_\lambda F(\lambda, \theta^\star) + \nabla_\theta F(\lambda, \theta^\star)^T \nabla_\lambda \theta^\star$$

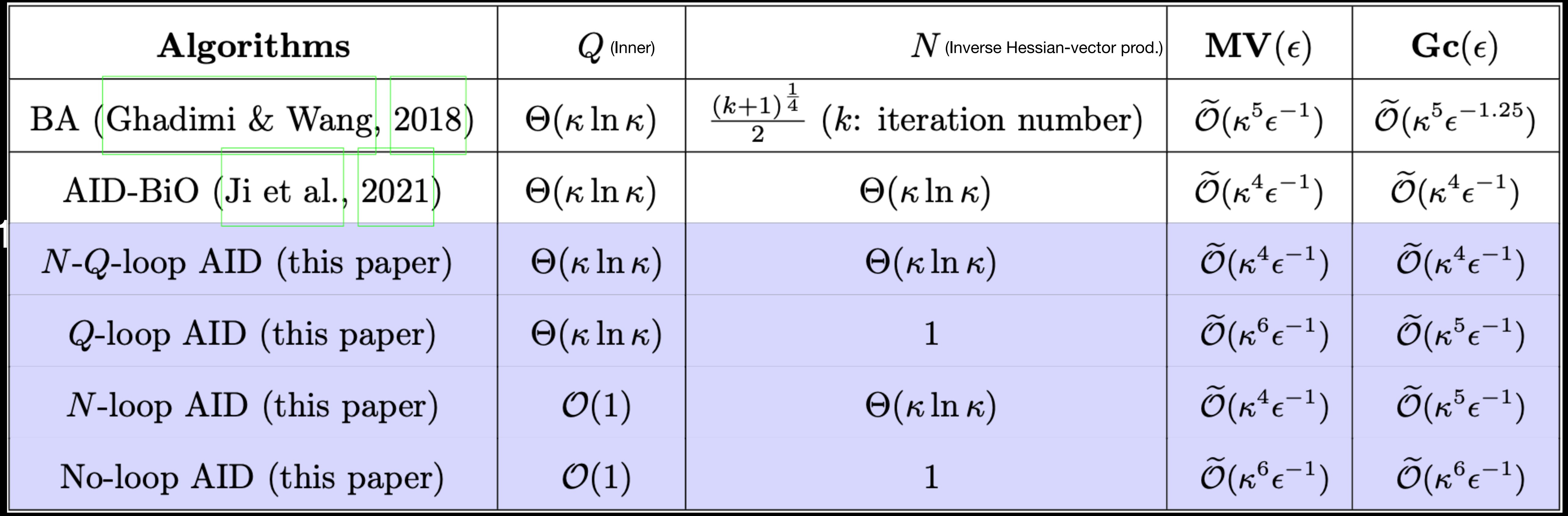
$$\theta^\star = \arg\min_{\theta} L(\lambda, \theta)$$

$$\nabla_\theta L(\lambda, \theta^\star) = 0 \Rightarrow \nabla^2_{\lambda,\theta} L(\lambda, \theta^\star) + \nabla^2_{\theta\theta} L(\lambda, \theta^\star)\, \nabla_\lambda \theta^\star$$

$$\boxed{\Rightarrow \nabla_\lambda \theta^\star = -\,(\nabla^2_{\theta\theta} L(\lambda, \theta^\star))^{-1}\, \nabla^2_{\lambda,\theta} L(\lambda, \theta^\star)}$$

$$\Rightarrow \nabla F(\lambda, \theta^\star) = \nabla_\lambda F(\lambda, \theta^\star) - \nabla_\theta F(\lambda, \theta^\star)^T (\nabla^2_{\theta\theta} L(\lambda, \theta^\star))^{-1}\, \nabla^2_{\lambda,\theta} L(\lambda, \theta^\star)$$

- Constraint-based approaches [Hansen et al. (1992); Shi et al. (2005); Moore (2010)]
- Gradient-based approaches
  - Implicit differentiation [Ghadmi and Wang, 2018; Domke, 201



Algorithms	Q (Inner)	N (Inverse Hessian-vector prod.)	$\mathbf{MV}(\epsilon)$	$\mathbf{Gc}(\epsilon)$
BA (Ghadimi & Wang, 2018)	$\Theta(\kappa \ln \kappa)$	$\frac{(k+1)^{\frac{1}{4}}}{2}$ (k: iteration number)	$\widetilde{\mathcal{O}}(\kappa^5 \epsilon^{-1})$	$\widetilde{\mathcal{O}}(\kappa^5 \epsilon^{-1.25})$
AID-BiO (Ji et al., 2021)	$\Theta(\kappa \ln \kappa)$	$\Theta(\kappa \ln \kappa)$	$\widetilde{\mathcal{O}}(\kappa^4 \epsilon^{-1})$	$\widetilde{\mathcal{O}}(\kappa^4 \epsilon^{-1})$
N-Q-loop AID (this paper)	$\Theta(\kappa \ln \kappa)$	$\Theta(\kappa \ln \kappa)$	$\widetilde{\mathcal{O}}(\kappa^4 \epsilon^{-1})$	$\widetilde{\mathcal{O}}(\kappa^4 \epsilon^{-1})$
Q-loop AID (this paper)	$\Theta(\kappa \ln \kappa)$	1	$\widetilde{\mathcal{O}}(\kappa^6 \epsilon^{-1})$	$\widetilde{\mathcal{O}}(\kappa^5 \epsilon^{-1})$
N-loop AID (this paper)	$\mathcal{O}(1)$	$\Theta(\kappa \ln \kappa)$	$\widetilde{\mathcal{O}}(\kappa^4 \epsilon^{-1})$	$\widetilde{\mathcal{O}}(\kappa^5 \epsilon^{-1})$
No-loop AID (this paper)	$\mathcal{O}(1)$	1	$\widetilde{\mathcal{O}}(\kappa^6 \epsilon^{-1})$	$\widetilde{\mathcal{O}}(\kappa^6 \epsilon^{-1})$

[Ji et al., 2022]

Bilevel Optimization Algorithms

$$\min_{\lambda} F(\lambda, \theta^\star)$$

$$\theta^\star = \arg\min_{\theta} L(\lambda, \theta)$$

$$\min_{\lambda} F(\lambda, \theta^\star)$$

$$\theta^\star = GD(\theta_0, L(\lambda, \theta), k)$$

- Constraint-based approaches [Hansen et al. (1992); Shi et al. (2005); Moore (2010)]
- Gradient-based approaches
  - Implicit differentiation [Ghadmi and Wang, 2018; Domke, 2012; Pedregosa, 2016; Grazzi et al., 2020; Ji et al., 2021]
  - Iterative differentiation [Maclaurin et al., 2015; Franceschi et al., 2017; Shaban et al., 2019]

Hypergradient descent

$$\frac{\mathrm{d}\varepsilon}{\mathrm{d}\lambda}?$$

$$\min_{\lambda} \varepsilon(\lambda) = \left| L_0(\theta^\star) - L_1(\theta^\star) \right|$$

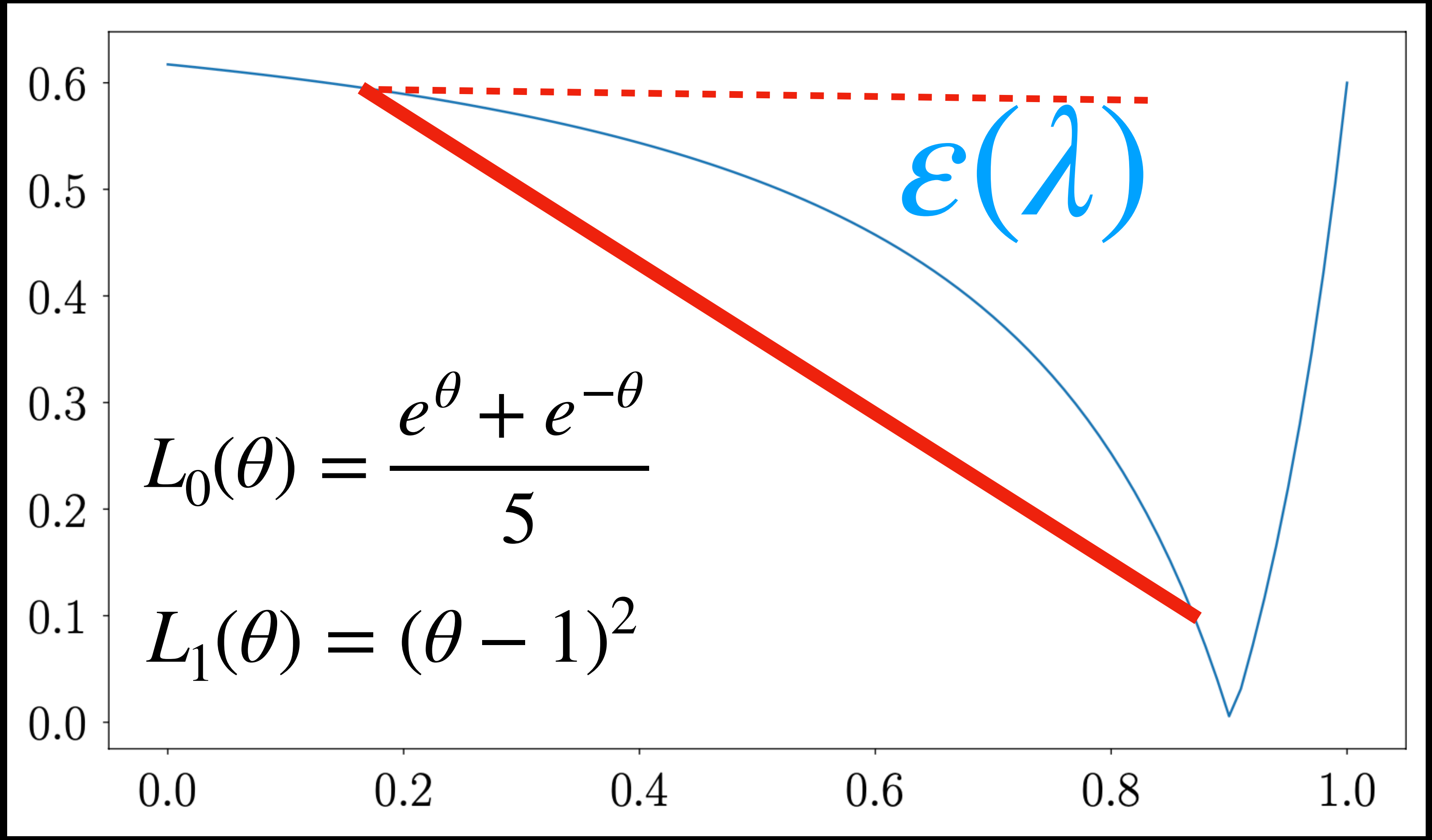
$$\theta^\star = \arg\min_{\theta} (1+\lambda) L_0(\theta) + (1-\lambda) L_1(\theta)$$

$$\frac{\mathrm{d}\varepsilon}{\mathrm{d}\lambda} = (L_0(\theta^\star) - L_1(\theta^\star)) \cdot \underbrace{\frac{\mathrm{d}L_0(\theta^\star) - L_1(\theta^\star)}{\mathrm{d}\lambda}}_{\text{requires the Inverse Hessian}}$$

$L_0(\theta^\star) - L_1(\theta^\star)$ is nonincreasing

$$\Downarrow$$

$\varepsilon(\lambda)$ is quasi-convex



Algorithm

$$\frac{\mathrm{d}\varepsilon}{\mathrm{d}\lambda}?$$

$$\min_{\lambda} \varepsilon(\lambda) = \left| L_0(\theta^\star) - L_1(\theta^\star) \right|$$

$$\theta^\star = \arg\min_{\theta} (1+\lambda) L_0(\theta) + (1-\lambda) L_1(\theta)$$

$$\frac{\mathrm{d}\varepsilon}{\mathrm{d}\lambda} = (L_0(\theta^\star) - L_1(\theta^\star) \cdot \underbrace{\frac{\mathrm{d}L_0(\theta^\star) - L_1(\theta^\star)}{\mathrm{d}\lambda}}_{\text{requires the Inverse Hessian}}$$

$$\Downarrow$$

$L_0(\theta^\star) - L_1(\theta^\star)$ is nonincreasing

$$\Downarrow$$

$\varepsilon(\lambda)$ is quasi-convex

$$\Downarrow$$ [Hazan, Levy, Shalev-Shwartz, '15]

$$\frac{\frac{\mathrm{d}\varepsilon}{\mathrm{d}\lambda}}{\left\| \frac{\mathrm{d}\varepsilon}{\mathrm{d}\lambda} \right\|_2} = \mathrm{sign}\left(\frac{\mathrm{d}\varepsilon}{\mathrm{d}\lambda}\right)$$ is all you need

$$\mathrm{sign}\left(\frac{\mathrm{d}\varepsilon}{\mathrm{d}\lambda}\right) = \mathrm{sign}(L_1(\theta^\star) - L_0(\theta^\star))$$

$$\searrow$$

$$\lambda \leftarrow \lambda - \alpha(\mathrm{sign}(L_1(\theta^\star) - L_0(\theta^\star)))$$

Signed hypergradient descent

```
$\lambda = 0$
while not converged:
    $\theta^\star(\lambda) = \arg\min_\theta (1+\lambda)L_0(\theta) + (1-\lambda)L_1(\theta)$
    $\lambda \leftarrow \lambda - \alpha(\text{sign}(L_1(\theta^\star) - L_0(\theta^\star)))$
```

Theorem. This algorithm converges to $\lambda^\star$ in $\left|\frac{1}{\alpha}\right|$ steps

No assumptions at all — DNN or whatever

Signed hypergradient descent + no-loop approx.

```
Initialize $\lambda, \theta$

while not converged:
    $F(\theta) = (1+\lambda)L_0(\theta) + (1-\lambda)L_1(\theta)$
    $\theta \leftarrow \theta - \beta \nabla_\theta F(\theta)$
    $\lambda \leftarrow \lambda - \alpha(\text{sign}(L_1(\theta^\star) - L_0(\theta^\star)))$
```

Convergence is never formally proved, but it is pretty straightforward…

With some assumptions, analysis should be similar to [Ghadimi and Wang, '18] and [Ji et al., '22]

Signed hypergradient descent + no-loop approx. +“adaptive” minibatches

```
Initialize $\lambda, \theta$
while not converged:
    $F(\theta) = (1+\lambda)L_0(\theta) + (1-\lambda)L_1(\theta)$
    $\theta \leftarrow \theta - \beta \nabla_\theta F(\theta)$
    $\lambda \leftarrow \lambda - \alpha(\text{sign}(L_1(\theta^\star) - L_0(\theta^\star)))$
```

Draw samples with $y = 0$ w.p. $\frac{1+\lambda}{2}$

Draw samples with $y = 1$ w.p. $\frac{1-\lambda}{2}$

Signed hypergradient descent + no-loop approx. +“adaptive” minibatches

```
Initialize $\lambda, \theta$
while not converged:
    $F(\theta) = (1+\lambda)L_0(\theta) + (1-\lambda)L_1(\theta)$
    $\theta \leftarrow \theta - \beta \nabla_\theta F(\theta)$
    $\lambda \leftarrow \lambda - \alpha(\text{sign}(L_1(\theta^\star) - L_0(\theta^\star)))$
```

Draw samples with $y = 0$ w.p. $\dfrac{1+\lambda}{2}$

Draw samples with $y = 1$ w.p. $\dfrac{1-\lambda}{2}$

$=$

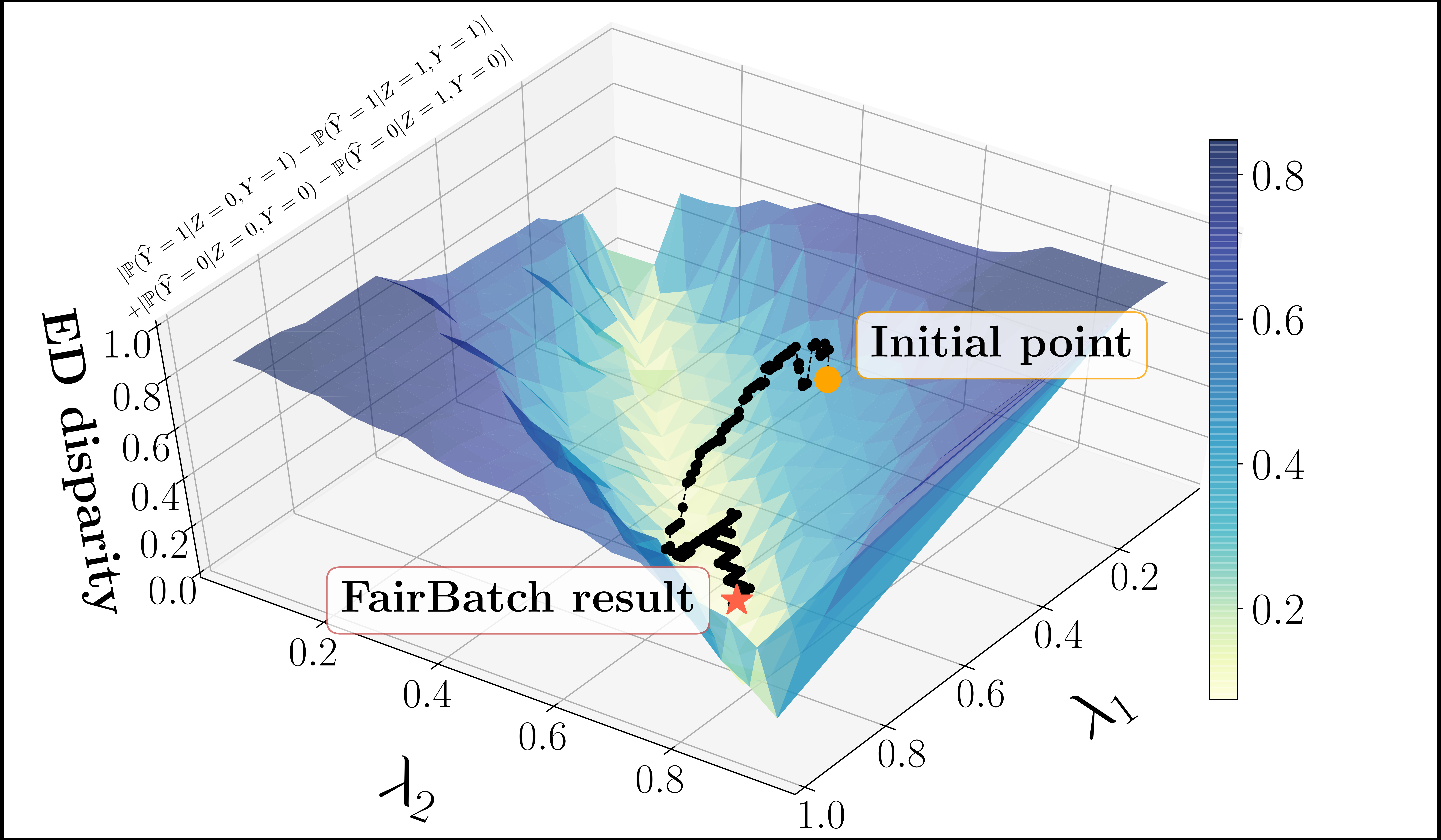
FairBatch

[Roh, Lee, Whang, and Suh, *ICLR’21*]

1. Train a model with vanilla SGD
2. Measure L_0 and L_1
3. If $L_0 > L_1$: Continue training with minibatches with more group 0 data

   OR: Continue training with minibatches with more group 1 data

Experimental results

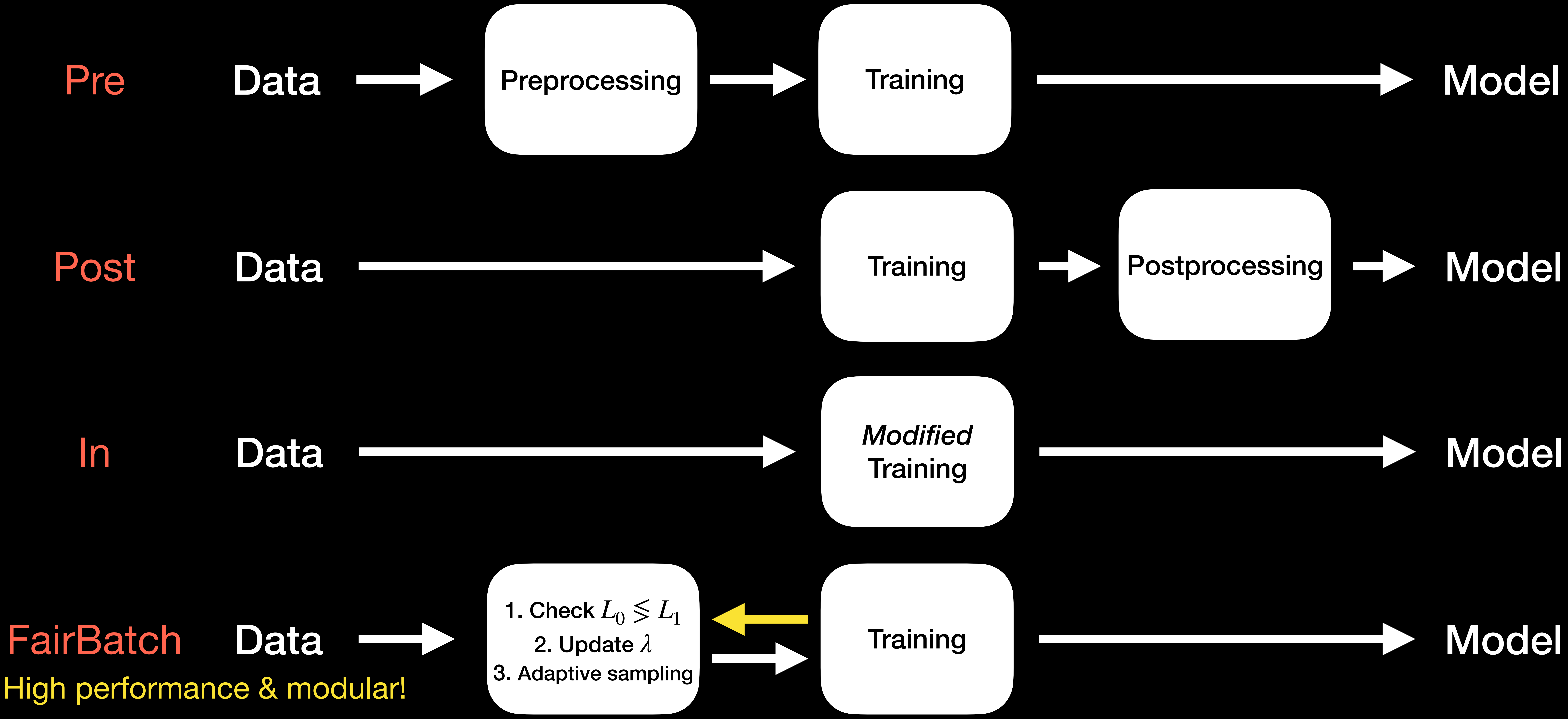


Experimental results

		Test accuracy	EO diff	Runtime (s)
Vanilla	Logistic regression	0.84	0.54	23
Fairness-aware	Logistic regression + Fairness constraints [1]	0.84	0.21	29
	Label bias correction [2]	**0.84**	**0.11**	**558**
	Adversarial debiasing [3]	0.84	0.16	32
	AdaFair [4]	0.84	0.38	792
	FairBatch (ours)	**0.84**	**0.11**	**47 -> 23**

[1] Zafar et al., 2017 [2] Jiang & Nachum, 2019 [3] Zhang et al., 2018 [4] Iosifidis & Ntoutsi, 2019

FairBatch = Adaptive pre-processing



Applications of FairBatch

- Fair and robust training [Roh, Lee, Whang, and Suh, *NeurIPS'21*]
  - Bilevel optimization (for fairness) + Integer optimization (for robustness)
- Federated fair training [Zeng, Chen, and Lee, *AAAIW'22*]
  - FairBatch is inherently "federatable" — Easy to check $\sum L_0 \lessgtr \sum L_1$
- Fair ML with non-differentiable models
  - Blackbox fine-tuning (e.g., GPT3) [Zeng, Lin, Park, Oh, Lee, *in progress*]
  - Decision trees [Lin and Lee, *in progress*]

Conclusion

- FairBatch: A new ML application of bilevel optimization
  - The single-loop version comes with a convergence guarantee
  - The no-loop version works very well in practice
  - They achieve the state-of-the-art performances on most datasets
  - Very easy-to-implement!
- Many applications due to its modularity
  - Fair learning + Robustness
  - Fair learning + Federated learning
  - Fair learning + Black-box training (e.g., GPT3 finetuning & decision trees)

Thanks! Any questions?