Simple and near-optimal algorithms for hidden stratification and multi-group learning

Christopher Tosh

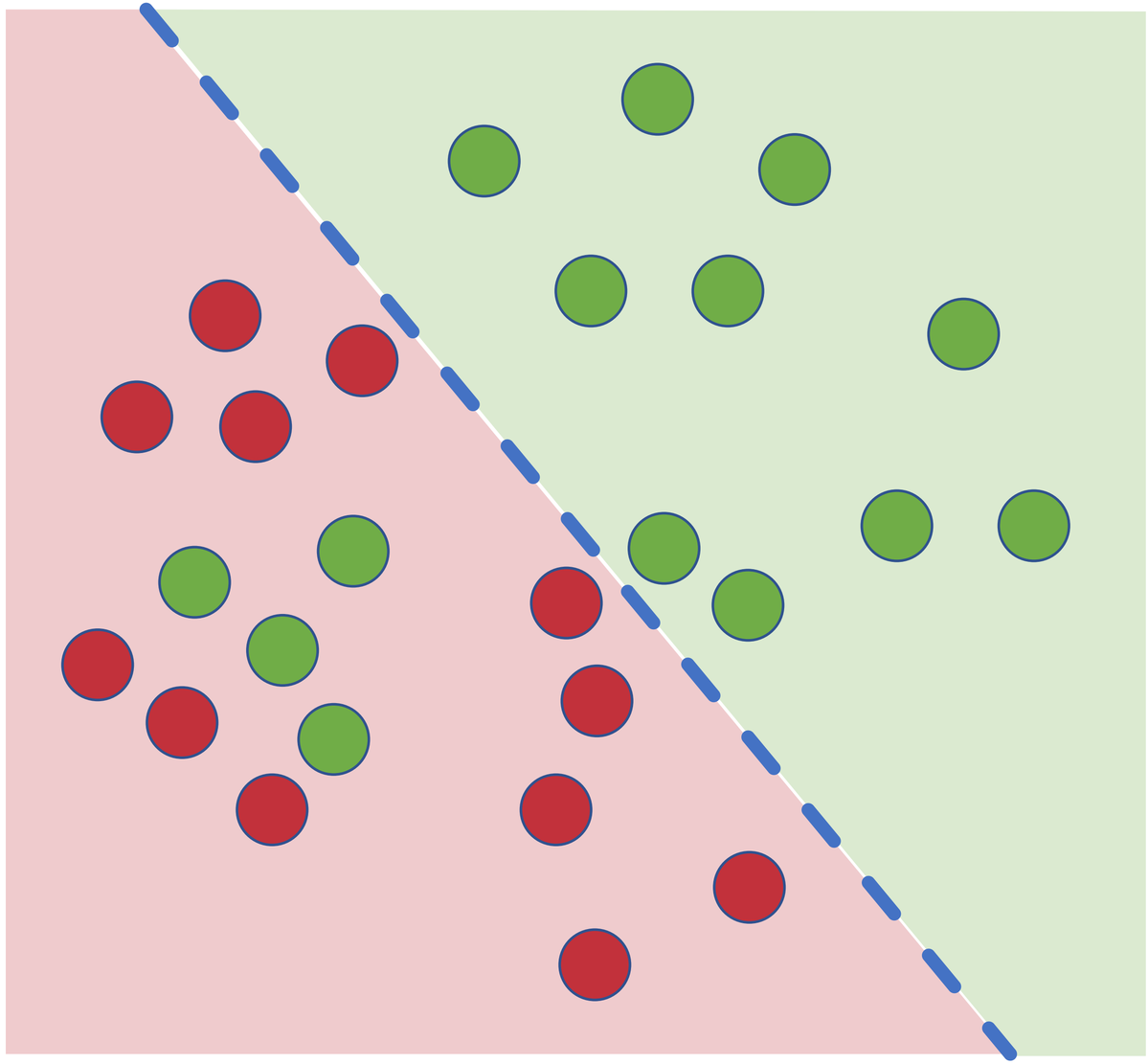
Memorial Sloan Kettering Cancer Center

Daniel Hsu

Columbia University

Motivation

Typical objective: accuracy

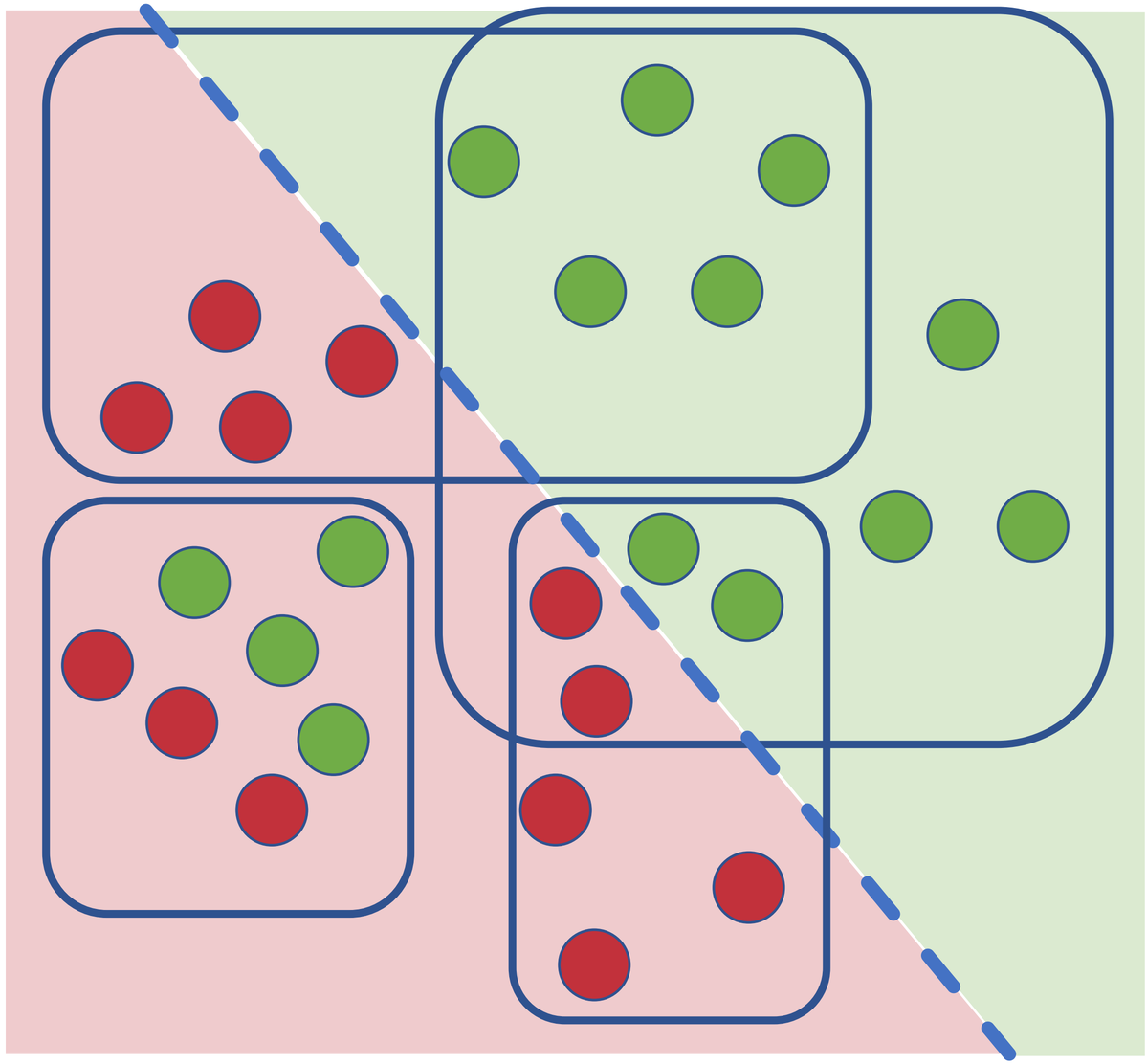


Motivation

Typical objective: accuracy

Many setting have **subgroup structure**

- Demographic attributes
- Cancer subtypes



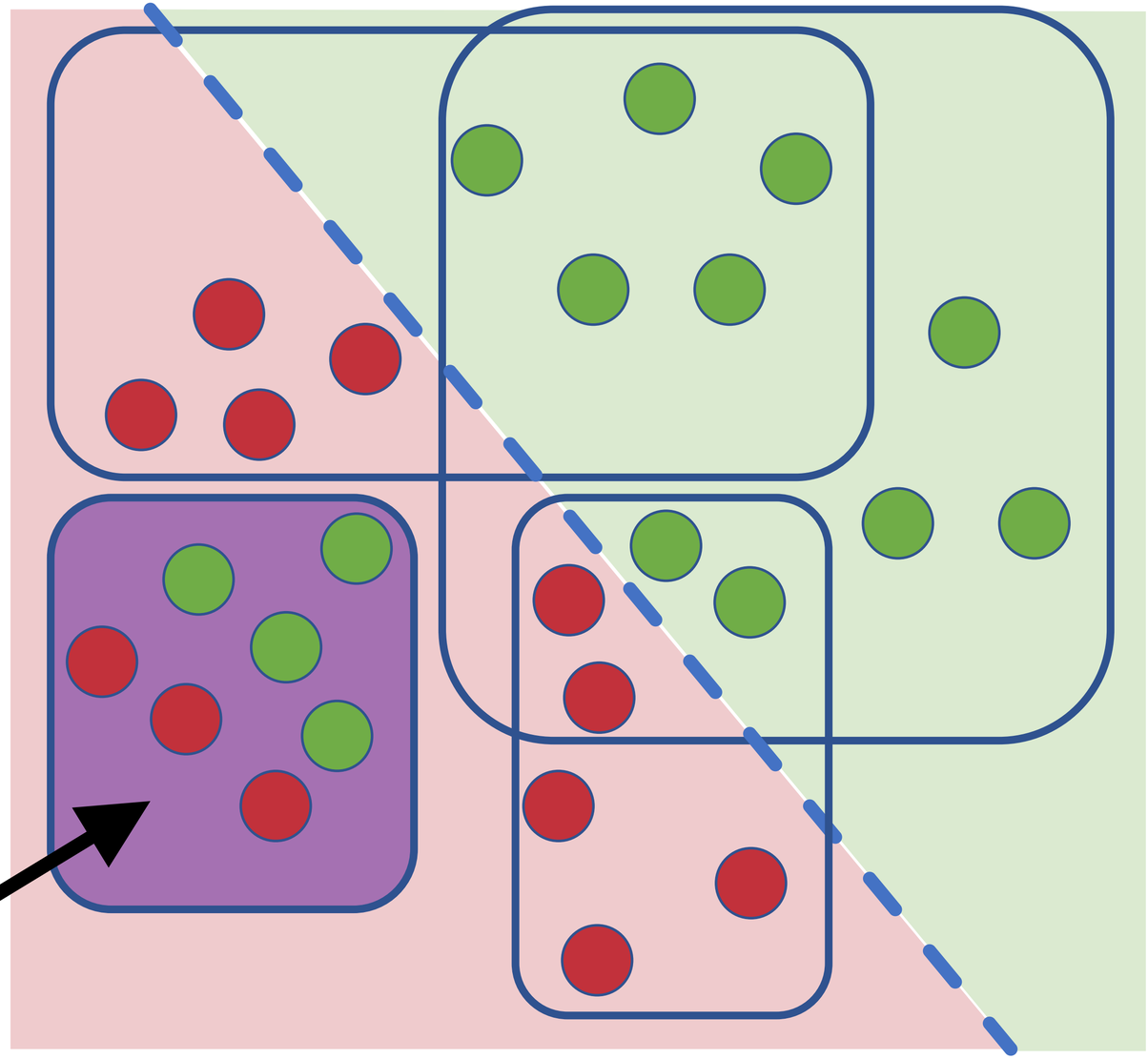
Motivation

Typical objective: accuracy

Many setting have **subgroup structure**

- Demographic attributes
- Cancer subtypes

Focusing solely on accuracy can lead to undesirable subgroup outcomes



Disadvantaged subgroup

Multi-group learning

Introduced by Rothblum and Yona (2021).

Setup: Reference predictors $\mathcal{H} \subset \{h : \mathcal{X} \mapsto \mathcal{Y}\}$, subgroups $\mathcal{G} \subset \{\, g \subseteq \mathcal{X} \,\}$,

Observe: Data $(x_1, y_1), \ldots, (x_n, y_n)$

Goal: Find a predictor f satisfying

$$\mathbb{E}[\ell(f(x), y) \mid x \in g] \;\leq\; \min_{h \in \mathcal{H}} \mathbb{E}[\,\ell(h(x), y) \mid x \in g] + \varepsilon_n(g) \qquad \text{for all } g \in \mathcal{G}$$

Best reference predictor error on g

Excess error

Simultaneously across subgroups

ERM: If we fix a single group g, can achieve $\varepsilon_n(g) = O\left(\sqrt{\frac{\log|\mathcal{H}|}{\#_n(g)}}\right)$

Number of observations in g

Main results

Simple algorithm

Iterative boosting algorithm

Outputs a decision list f satisfying

$$\varepsilon_n(g) = O\left(\sqrt[3]{\frac{\log|\mathcal{H}||\mathcal{G}|}{\gamma\,\#_n(g)}}\right)$$

for all $g \in \mathcal{G}$.

Minimum probability mass among $g \in \mathcal{G}$

(Nearly) optimal algorithm

Reduction to sleeping experts

Outputs a probabilistic predictor f satisfying

$$\varepsilon_n(g) \leq O\left(\sqrt{\frac{\log|\mathcal{H}||\mathcal{G}|}{\#_n(g)}}\right)$$

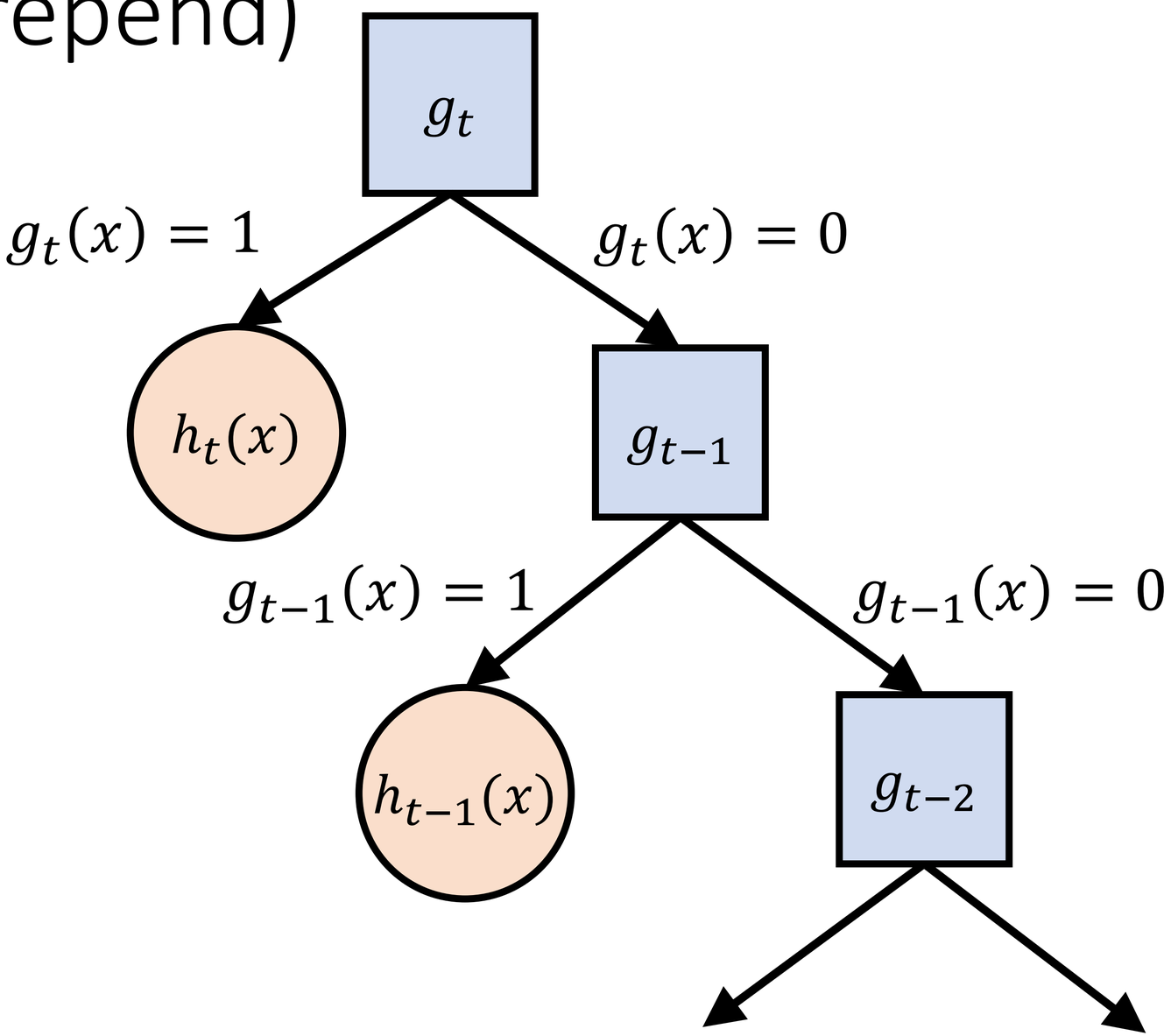
for all $g \in \mathcal{G}$.

Simple algorithm (Prepend)

Iterative builds a decision list.

Decision nodes indexed by $\mathcal{G}$.

Prediction nodes indexed by $\mathcal{H}$.



Simple algorithm (Prepend)

Iterative builds a decision list.

Decision nodes indexed by $\mathcal{G}$.

Prediction nodes indexed by $\mathcal{H}$.

At round *t:*

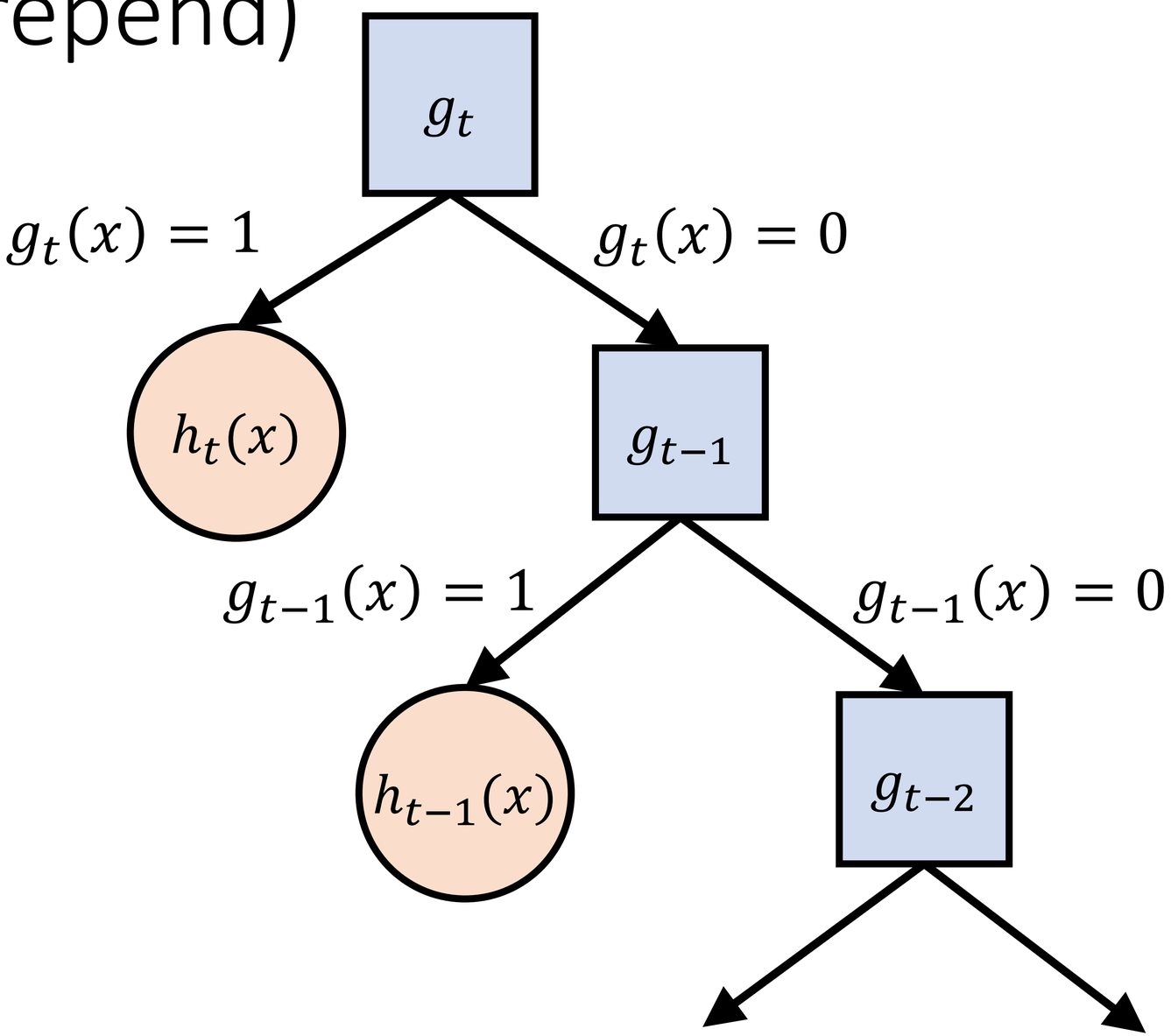
Current decision list f_t.

Search for $h \in \mathcal{H}, g \in \mathcal{G}$ such that

$$L_n(f_t \mid g) \geq L_n(h \mid g) + \varepsilon_n(g)$$

If such a pair exists, prepend it to the beginning of f_t to get f_{t+1}.

Else, terminate.



Simple algorithm (Prepend)

Iterative builds a decision list.

Decision nodes indexed by $\mathcal{G}$.

Prediction nodes indexed by $\mathcal{H}$.

At round *t:*

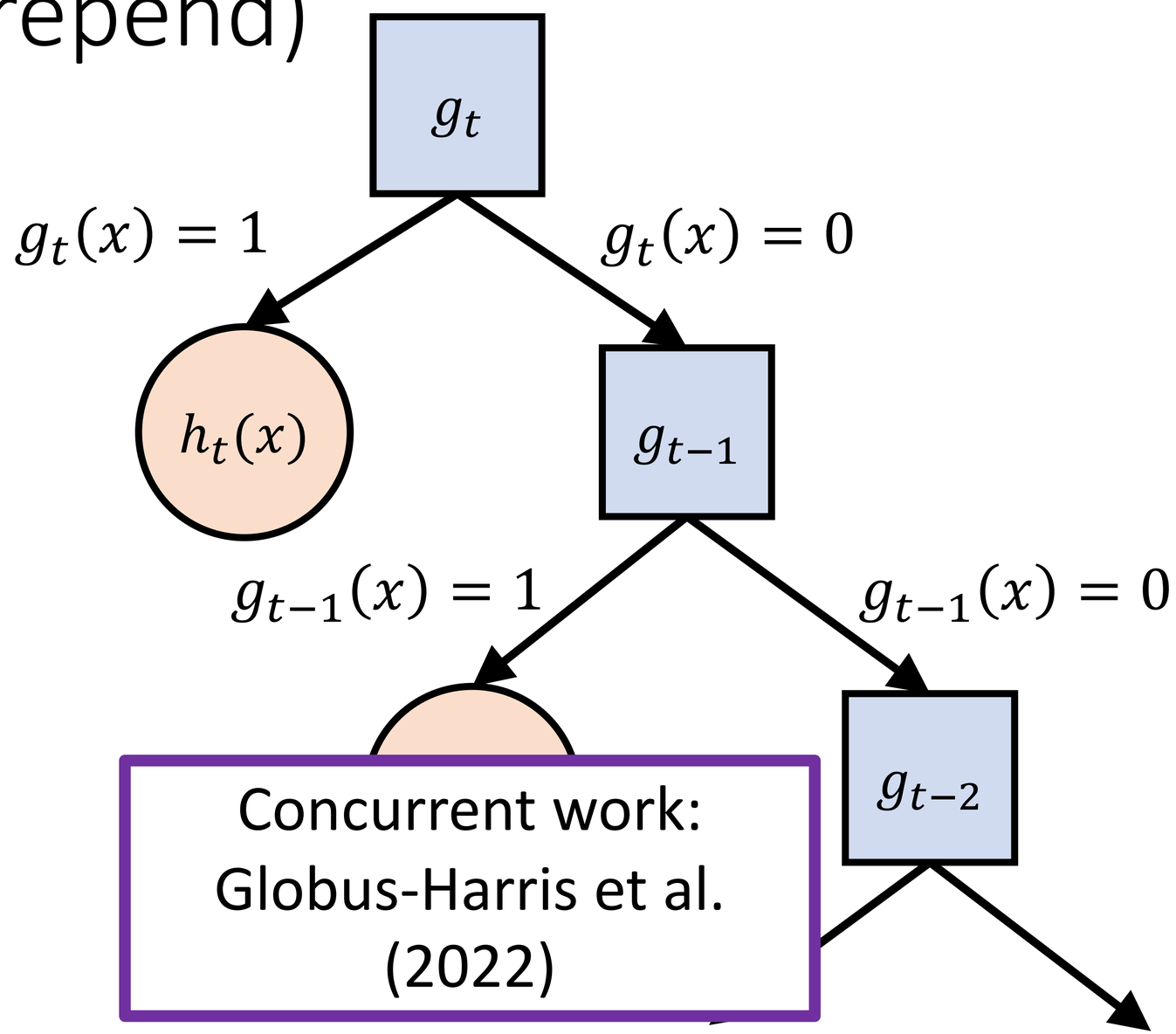
Current decision list f_t.

Search for $h \in \mathcal{H}, g \in \mathcal{G}$ such that

$$L_n(f_t \mid g) \geq L_n(h \mid g) + \varepsilon_n(g)$$

If such a pair exists, prepend it to the beginning of f_t to get f_{t+1}.

Else, terminate.



(Nearly) Optimal algorithm

Reduction to sleeping experts

Create an "expert" for every $(h, g) \in \mathcal{H} \times \mathcal{G}$.

For data points $i = 1, \ldots, n$:

Expert (h, g) is awake if $g(x_i) = 1$.

Create distribution p_i over awake experts.

Suffer loss $\ell_i = \sum_{(h,g)} g(x_i)\ell(h(x_i), y_i)$.

Final predictor: Uniform distribution over internal hypotheses p_i.

(Nearly) Optimal algorithm

Reduction to sleeping experts

Create an "expert" for every $(h, g) \in \mathcal{H} \times \mathcal{G}$.

For data points $i = 1, \ldots, n$:

Expert (h, g) is awake if $g(x_i) = 1$.

Create distribution p_i over awake experts.

Suffer loss $\ell_i = \sum_{(h,g)} g(x_i)\ell(h(x_i), y_i)$.

Online subgroup fairness:
Blum and Lykouris (2020)

Online-to-batch is tricky:
$|\mathcal{G}|$ constraints to satisfy!

Final predictor: Uniform distribution over internal hypotheses p_i.

Conclusion and future directions

Algorithms for multi-group learning setting.

Simple but suboptimal approach.

(Nearly) optimal but complicated approach.

Open problems:

A simple **and** optimal algorithm?

Computationally efficient algorithms?

Conclusion and future directions

Algorithms for multi-group learning setting.

Simple but suboptimal approach.

(Nearly) optimal but complicated approach.

Open problems:

A simple **and** optimal algorithm?

Computationally efficient algorithms?

Thank you!