Metric-Fair Classifier Derandomization

Jimmy Wu **Yatong Chen** Yang Liu

Computer Science and Engineering

UC Santa Cruz

What is a Stochastic Classifier?

Stochastic (binary) classifier: maps each input to the probability of a positive prediction

$$f : X \to [0, 1]$$

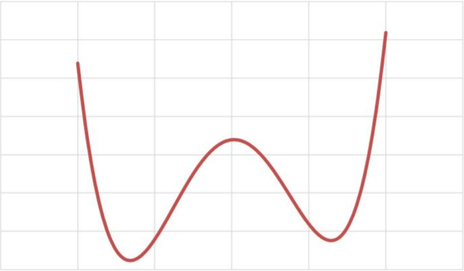
Input

Probability
(of being classified as 1)

Why Derandomize Stochastic Classifiers?

Stochastic classifiers: useful for **performance reasons**

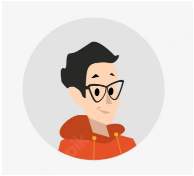
e.g. solving non-convex optimization problems



$$\min_{\theta \in \Theta} g_0(\theta)$$
$$\text{s.t. } g_i(\theta) \leq 0$$

Deterministic classifiers: better for **practical reasons**

e.g. consistent, easy to debug

$f($  $) = 0.5$

Even the ***same*** person may get completely ***different*** prediction every time!

Classifier Derandomization

Problem statement

- Input: a **stochastic** classifier $f : X \to [0, 1]$
- Sample: a **deterministic classifier** $\hat{f} : X \to \{0, 1\}$ that *preserves various properties of* f in expectation.

Our Contribution

A sample-efficient procedure to derandomize f to $\hat{f}$, while preserving:

1) expected output of f on any *x*:

$$\mathbb{E}_{\hat{f}}\left[\hat{f}(x)\right] \approx f(x), \quad \forall x \in X$$

2) individual (metric) fairness of f:

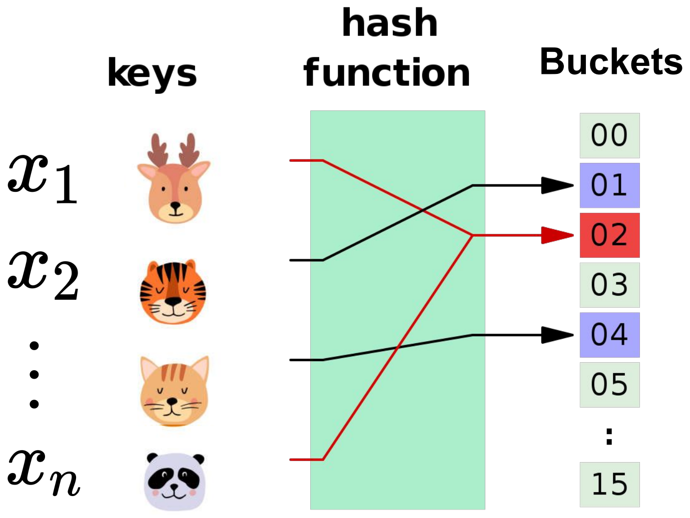
$$|f(x) - f(x')| \leq \alpha \cdot d(x, x')$$

$$\Rightarrow \mathbb{E}_{\hat{f}}\left[\left|\hat{f}(x) - \hat{f}(x')\right|\right] \leq O(\alpha) \cdot d(x, x')$$

distance metric

Previous Approach: Hashing [CNG 2019]

Main idea: simulate randomness with pairwise-independent hashing



|Buckets| = ***k***

Previous Approach: Hashing [CNG 2019]

CNG's Derandomization Procedure:

1) Sample a pairwise-independent hash function $h_{\mathrm{PI}} \sim \mathcal{H}_{\mathrm{PI}}$ with *k* buckets
2) Define $\hat{f}$ based on h_{PI}:

$$\hat{f}(x) := 1\left\{f(x) \geq \frac{h_{\mathrm{PI}}(x)}{k}\right\}$$

Previous Approach: Hashing [CNG 2019]

CNG's Derandomization Procedure:

1) Sample a pairwise-independent hash function $h_{\mathrm{PI}} \sim \mathcal{H}_{\mathrm{PI}}$ with *k* buckets
2) Define $\hat{f}$ based on h_{PI}:

Why does it work?

$$\hat{f}(x) := 1\left\{f(x) \geq \boxed{\frac{h_{\mathrm{PI}}(x)}{k}}\right\}$$

pseudo-random threshold in [0, 1]

Previous Approach: Hashing [CNG 2019]

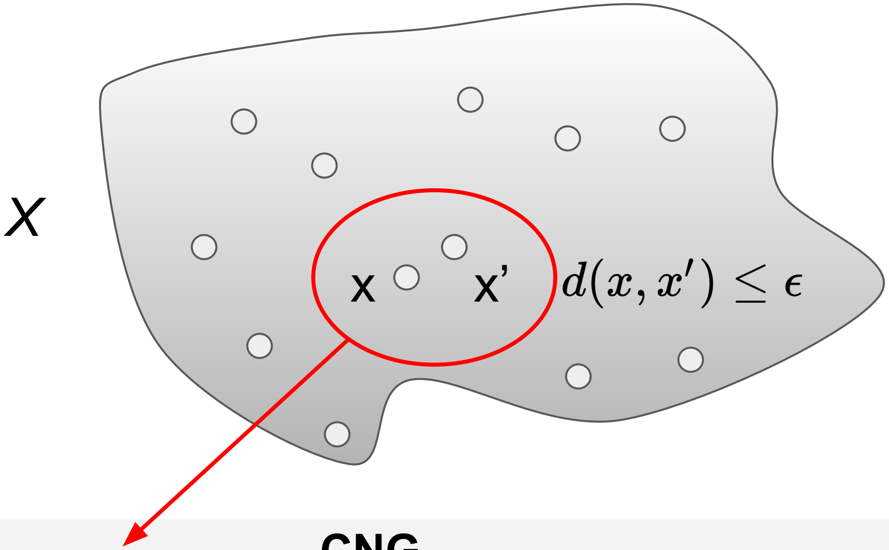
$$\hat{f}(x) := 1\left\{ f(x) \geq \frac{h_{\mathrm{PI}}(x)}{k} \right\}$$

Theorem [CNG 2019, informal] *Given f, this procedure samples* $\hat{f}$ *satisfying:*

(Output Approximation) $\mathbb{E}_{x\sim\mathcal{D}}[\hat{f}(x)] \approx \mathbb{E}_{x\sim\mathcal{D}}[f(x)]$ w.h.p. over $\hat{f}$

[CNG 2019] Does Not Preserve Metric Fairness

Suppose *f* is ***metric-fair***: $|f(x) - f(x')| \leq \alpha \cdot d(x, x'), \ \ \forall x, x'$



$f(x) = 0.32$
$f(x') = 0.35$

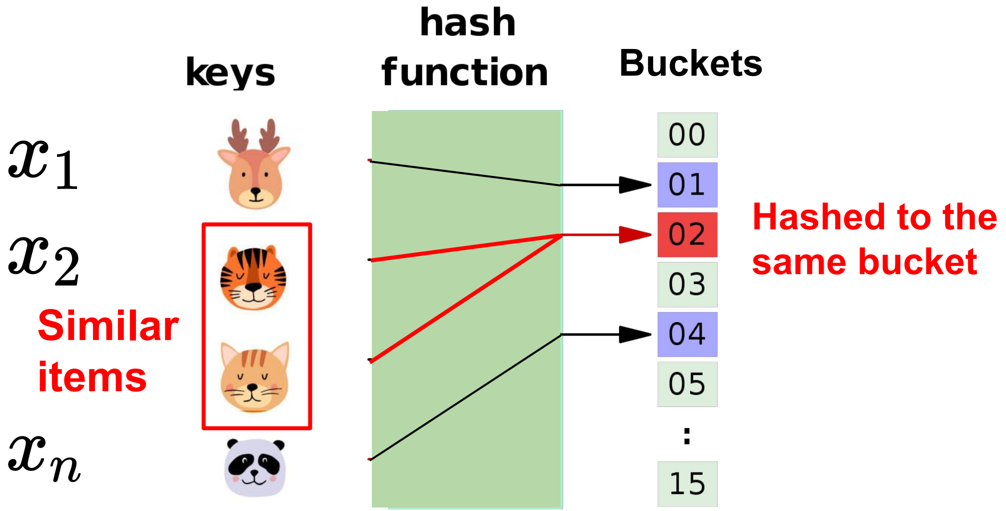
CNG derandomization

$\hat{f}(x) = 0$
$\hat{f}(x') = 1$

Our Approach: Locality Sensitive Hashing

Locality-sensitive hashing (LSH): $h_{\mathrm{LS}} \sim \mathcal{H}_{\mathrm{LS}}$

$$\Pr_{h\sim\mathcal{H}_{\mathrm{LS}}}\left[h(x) \neq h\left(x'\right)\right] = d\left(x, x'\right), \ \forall x \neq x'$$



Our Approach: Locality Sensitive Hashing

Our Derandomization Procedure:

1) **[New] Sample a LSH function** $h_{\mathrm{LS}} \sim \mathcal{H}_{\mathrm{LS}}$
2) Sample a pairwise-independent hash function $h_{\mathrm{PI}} \sim \mathcal{H}_{\mathrm{PI}}$
3) Define $\hat{f}$ based on both h_{PI} and h_{LS} :

$$\hat{f}(x) := 1\left\{f(x) \geq \frac{h_{\mathrm{PI}}(h_{\mathrm{LS}}(x))}{k}\right\}$$

Intuition:

- h_{LS} : ensures similar items get the same prediction
- h_{PI} : ensures dissimilar items are treated randomly

Our Approach: Locality Sensitive Hashing

Our theoretical guarantee:

Theorem [informal] *Given a **metric-fair** f that satisfies*

$$|f(x) - f(x')| \leq \alpha \cdot d(x, x'), \;\; \forall x, x'$$

Our procedure samples $\hat{f}$ satisfying:

(Output approximation) $$\mathbb{E}_{x \sim \mathcal{D}}[\hat{f}(x)] \approx \mathbb{E}_{x \sim \mathcal{D}}[f(x)] \;\; \text{w.h.p. over } \hat{f}$$

(Preserves metric fairness) $$\mathbb{E}_{\hat{f}}[|\hat{f}(x) - \hat{f}(x')|] \lesssim (\alpha + \frac{1}{2}) \cdot d(x, x')$$

Thank you!

- Paper: https://arxiv.org/abs/2206.07826
- Poster Session: Hall E #1221