On Making Stochastic Classifiers Deterministic

Andrew Cotter, Harikrishna Narasimhan, Maya Gupta
{acotter,hnarasimhan,mayagupta}@google.com
Google Research

Stochastic Classifiers

Why do I care about stochastic classifiers? Mostly, constrained optimization

$$\min_{\theta \in \Theta} g_0(\theta)$$

$$\text{s.t.} \quad \forall i \in [m] \; g_i(\theta) \leq 0$$

Examples: statistical fairness problems, churn, FPR/FNR, precision/recall, ...

- All of these are *non-convex*

Stochastic Classifiers are Great

If we optimize the Lagrangian:

$$\mathcal{L}(\theta, \lambda) = g_0(\theta) + \sum_{i=1}^{m} \lambda_i g_i(\theta)$$

Then a pure Nash equilibrium might not *exist*

- So, if we use SGD, we might expect oscillation instead of convergence
  - Running SGD for a while, and taking the last iterate, seems ill-advised
- A mixed equilibrium *will* exist, and will give us a stochastic classifier
  - We can't just take the expected solution, since non-convexity ➔ no Jensen's inequality

Stochastic Classifiers are Terrible

Using a stochastic classifier in a real system is not always possible

- They're can be much larger (e.g. one model vs. a distribution over models)
- More difficult to debug and visualize
- Might be inherently unsuited for certain applications

For example, suppose we want a "fair" classifier

- We might want similar examples to receive similar predictions
- A stochastic classifier won't even be consistent on the *same* example

Dwork, Hardy, Pitassi, Reingold, Zemel. *Fairness through awareness*. Innovations in Theoretical Computer Science, 2012

Stochastic Classifiers are Terrible

This isn't a purely theoretical problem! In experiments, if we:

1. Have the λ and θ players run SGD for a while
2. Find the single best iterate
3. Find the best distribution supported on the sequence of iterates

We sometimes observe a significant gap between the performance of (2) and (3)

Cotter, Jiang, Wang, Narayan, You, Sridharan, Gupta. *Optimization with Non-Differentiable Constraints with Applications to Fairness, Recall, Churn, and Other Goals*. JMLR (*to appear*)

Stochastic to Deterministic Classifiers

- We'd prefer a stochastic classifier $f : \mathcal{X} \to [0, 1]$ for performance reasons
- We'd prefer a deterministic classifier $\hat{f} : \mathcal{X} \to \{0, 1\}$ for practical reasons

Proposal: approximate a stochastic classifier with a deterministic one

1. What does this mean? What constitutes a good approximation?
2. Is there a lower bound on how well a deterministic classifier can perform?
3. How well does the usual approach (thresholding) perform?
4. Can we get closer to the lower bound?

Lower Bound

What constitutes a good approximation?

- A *metric* consists of a loss, a labeling, and a subset of the data distribution
- Given *m* metrics, the original stochastic classifier, and a deterministic classifier, we want every metric to have roughly the same value, for both classifiers

Lower bound: there exists a metric ℓ and labeling s.t. for every deterministic $\hat{f}$:

$$\left|E_\ell(f) - E_\ell(\hat{f})\right| \geq \max_{x \in \mathcal{X}_\ell} \left\{ \Pr_{x' \sim \mathcal{D}_x|_{\mathcal{X}_\ell}} \{x' = x\} \cdot \min\{f(x), 1 - f(x)\} \right\}$$

In words: if there exists a large point mass on which the original stochastic classifier is very stochastic, then we cannot approximate well

Lower Bound

$$\left|E_\ell(f) - E_\ell(\hat{f})\right| \geq \max_{x \in \mathcal{X}_\ell} \left\{ \Pr_{x' \sim \mathcal{D}_x|_{\mathcal{X}_\ell}} \{x' = x\} \cdot \min\{f(x), 1 - f(x)\} \right\}$$

What constitutes a good approximation?

- A *metric* consists of a loss, a labeling, and a subset of the data distribution
- Given m metrics, the original stochastic classifier, and a deterministic classifier, we want every metric to have roughly the same value, for both classifiers

Lower bound: there exists a metric ℓ and labeling s.t. for every deterministic $\hat{f}$:

$$\left|E_\ell(f) - E_\ell(\hat{f})\right| \geq \max_{x \in \mathcal{X}_\ell} \left\{ \Pr_{x' \sim \mathcal{D}_x|_{\mathcal{X}_\ell}} \{x' = x\} \cdot \min\{f(x), 1 - f(x)\} \right\}$$

In words: if there exists a large point mass on which the original stochastic classifier is very stochastic, then we cannot approximate well

Thresholding

$$\left|E_\ell(f) - E_\ell(\hat{f})\right| \geq \max_{x \in \mathcal{X}_\ell} \left\{ \Pr_{x' \sim \mathcal{D}_x|_{\mathcal{X}_\ell}} \{x' = x\} \cdot \min\{f(x), 1 - f(x)\} \right\}$$

Thresholding is probably the most common approach: define $\hat{f}(x) = \mathbf{1}\{f(x) > 1/2\}$

Then, for any metric ℓ and labeling:

$$\left|E_\ell(f) - E_\ell(\hat{f})\right| \leq \mathbb{E}_{x \sim \mathcal{D}_x|_{\mathcal{X}_\ell}} [\min\{f(x), 1 - f(x)\}]$$

- Bound improves as the stochastic classifier becomes less stochastic
- But it does *not* improve as point masses shrink

Example: continuous data distribution (no point masses), and *f(x) = 0.51* for all *x*.

- Then $\hat{f}(x) = 1$ for all *x*
- The positive prediction rates (and many other metrics) will be very different

Hashing

$$\left|E_\ell(f) - E_\ell(\hat{f})\right| \geq \max_{x \in \mathcal{X}_\ell} \left\{ \Pr_{x' \sim \mathcal{D}_x|_{\mathcal{X}_\ell}} \{x' = x\} \cdot \min\{f(x), 1 - f(x)\} \right\}$$

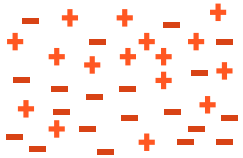
Hashing is based on the idea of “faking” stochasticity

- Assume we have a pairwise independent family of hash functions
- Sample a hash function, and use it to define the deterministic classifier
  - For each example *x*, hash its features to create a random-seeming threshold

We prove a bound that, with high probability over the hash function sample

- Is too complicated to state here
- But *does* go to zero as the stochasticity on large point masses goes to zero
  - It doesn’t exactly “match” the lower bound, but does have the same essential properties

Refinements



The fact that a hashing classifier “looks” stochastic may be a disadvantage

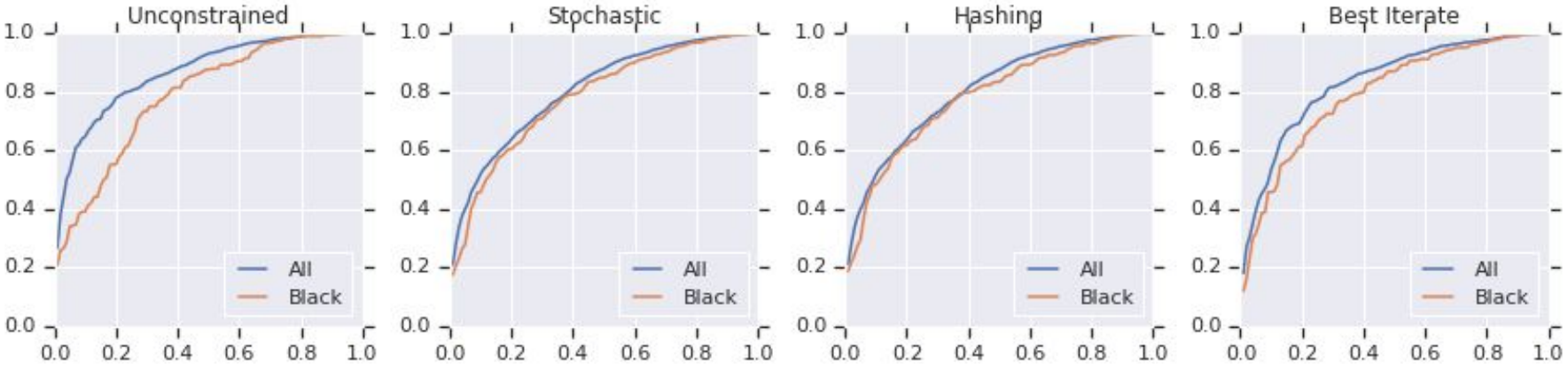
- It can be highly non-smooth
- It makes arbitrary distinctions between otherwise-similar examples
- Visualization and debugging problems comparable to stochastic classifier

One possible solution: *interpolate* between thresholding and hashing

- A thresholding classifier uses the same threshold for each x
- A hashing classifier uses a different threshold for each x
- Create a similarity-preserving clustering, and separately threshold each *cluster*

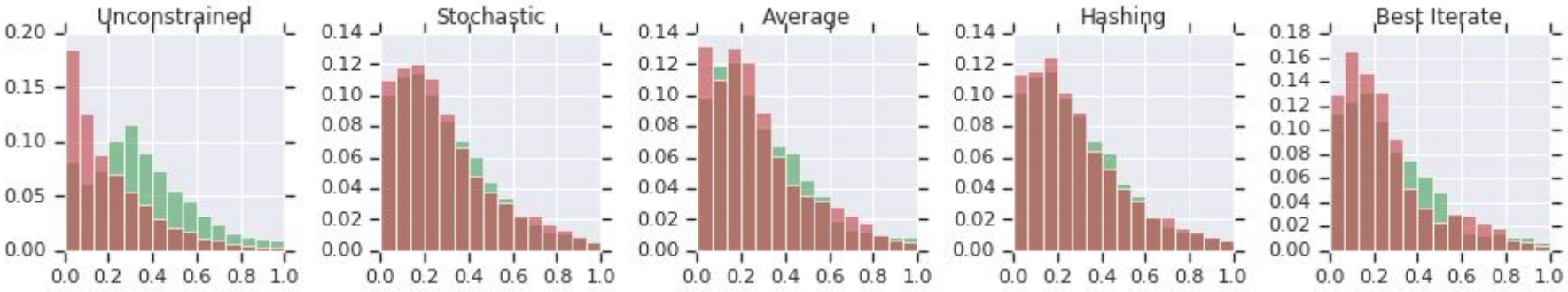
Same bound as hashing, but point masses are measured on *clusters*

Experiments - Law school dataset



Constraints align ROC curves for the protected class and the overall population

Experiments - Adult dataset



Constraints match histograms of model outputs between the protected class and the overall population

Thank You!

{acotter,hnarasimhan,mayagupta}@google.com