Quantifying Program Bias

Verifying quantitative properties/ post conditions of probabilistic programs

Dennis Roellke

Quick Motivation

Bias in ML/ Fairness

“

[M]any **ML** applications, especially decision makers are **introduced**, **because** they are (or let’s say they were) initially thought to be **less biased** than humans.

“

Verifying Fairness Properties via Concentration, Bastani et al. https://arxiv.org/pdf/1812.02573.pdf

What kind of ML?

(Simple) Decision Makers f(x)

- Linear Regression
- SVM
- NN

Population Model p(i)

- Generative Model that generates records of individuals
  By defining a probability distribution over inputs of f

Post condition in form of a quantitative property

- Here Fairness

What kind of Fairness?

$$\frac{P(Y{=}1|Z{=}0)}{P(Y{=}1|Z{=}1)} > 1 - e$$

Common, legal interpretation of Fairness
“Demographic Parity”, or 80:20 Rule

Read: The ratio between 2 sensitive groups Z=0 and Z=1 shall not exceed **e**

To verify this inequality, tight bounds are enough, the exact values are not required

Probabilistic Condition

$$\frac{P(Y{=}1,Z{=}0)\cdot P(Z{=}1)}{P(Y{=}1,Z{=}1)\cdot P(Z{=}0)} > 1 - e$$

// focus on one term for simplicity of presentation

e.g. P(Y=1,Z=1)

We want to compute the probability that the population model generates a non-minority **and** the (target) decision making program hires this applicant.

The function **φ = popModel o decision** can be composed and translated in to a boolean formula, i.e. "an (arbitrary) SMT formula over an arithmetic theory"

Probabilistic Condition cont'd

φ = popModel o decision

popModel = Z > 0 -> x1 += 5 && Z <= 0 -> x1 = x1 **// rejection sampling**

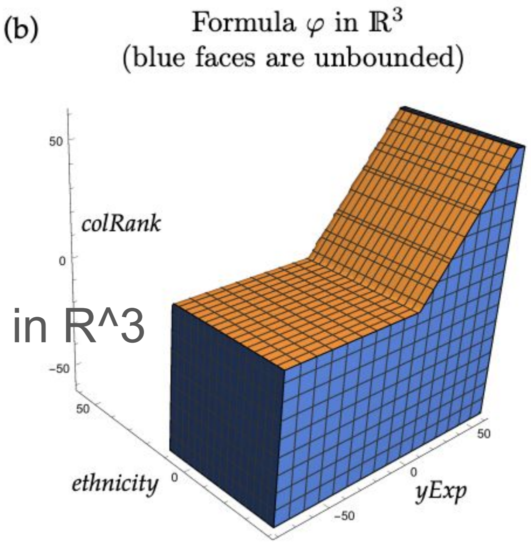
decision = (x1-x2) > -5 || x2 <= 5

φ = Z > 0 -> x1 += 5 && Z <= 0 -> x1 = x1 o (x1-x2) > -5 || x2 <= 5

Recall: We want to compute the probability that the population model generates a non-minority **and** the (target) decision making program hires this applicant.

E.g. P(Y=1,Z=1) ⇔ **φ = (x1-x2) > -5 || x2 <= 5 && Z <= 0**

Weighted Volume Computation



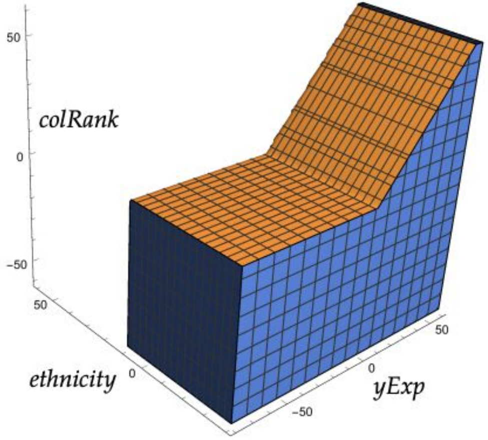
Note that **φ** is a function in R^n
E.g., when looking at 3 (independent) variables **φ** is a region in R^3

Now, the probability of satisfying **φ** can be understood
as the probability of drawing a value that fall in the region
Bounded by **φ**

Claim:
The probability of satisfying **φ** is the volume of **φ** in R^n, weighted by the
probability density of each of the n variables

$$\mathbf{Pr}[\mathtt{hire} \land \neg\mathtt{min}] = \int_{\varphi} p_e p_y p_c \, dV_p$$

(b) Formula φ in $\mathbb{R}^3$
(blue faces are unbounded)



Weighted Volume Computation Problem

How do we compute a numerical value for this integral?

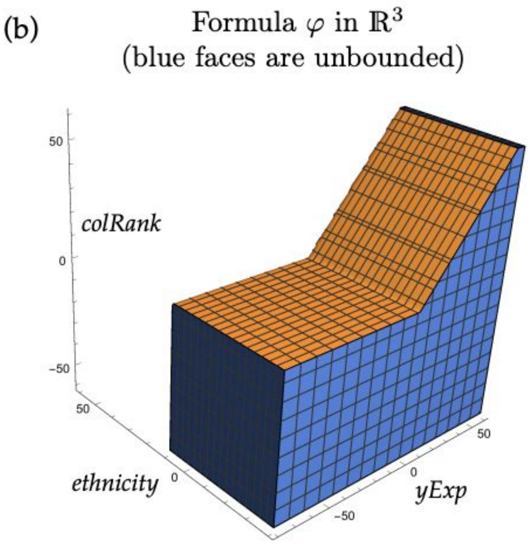
Recall:

φ = (x1-x2) > -5 || x2 <= 5 && Z <= 0

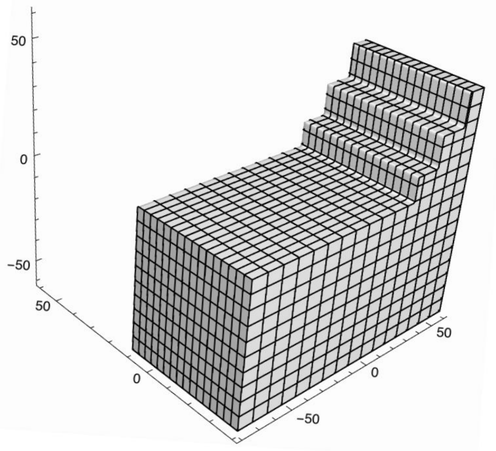
$$\Pr[\mathtt{hire} \wedge \neg\mathtt{min}] = \int_{\varphi} p_e p_y p_c \, dV_p$$

Weighted Volume Computation Observations

1. If the region specified by the formula is a (hyper) rectangular region it has constant upper and lower bounds in all dimensions and integration is simple
2. An SMT formula can be *symbolically* decomposed in to infinite (hyper) rectangles.



Underapproximation of φ as a union of hyperrectangles



Weighted Volume Computation Intuition

Use m = Re(**φ**)
To find disjoint (hyper) rectangles that each under-approximate **φ**

Computing their weighted volume serves a lower bound

Upper bound?

Well, Volume(**φ**) + Volume(!**φ**) = 1, so
Upper_bound = 1 - lower_bound of Volume(!**φ**)

Limitations

The to-be-verified probabilistic program is Loop Free (i.e. no RNN)

The probability distributions are univariate gaussian/ laplacian of constant parameters, i.e. programs are in static single assignment form (SSA)

SMT solver may be black box

Contributions

1. *Symbolic* Volume Computations over SMT formulas

   $$\int_{\varphi} \prod_{x_i \in \mathcal{X}_{\varphi}} p_i(x_i)\, d\mathcal{X}_{\varphi}$$

   Known to be #P-hard

   Existing techniques

   a. restrict φ to linear inequalities
   b. restrict integrands to polynomial or simple distributions
   c. compute approximate solutions w/ only probabilistic guarantees
   d. restrict φ to bounded regions
2. Based on black box SMT solvers
3. A novel technique for approximately encoding PDFs as formulas, and using them to guide the SMT solver towards exact solutions

Algorithm

Weighted Volume Computation Algorithm

1. Hyperrectangular Decomposition
   a. Represent φ as rectangles (Re(φ))
2. Hyperrectangular Sampling
   a. Use and SMT solver to find models of Re(φ)

For each hyperrectangle we sample, we compute its weighted volume and add it to our current solution. Therefore, the current solution maintained by the algorithm is the weighted volume of an underapproximation of φ—that is, a lower bound on the exact weighted volume of φ

Hyperrectangular Decomposition

The potential number hyperrectangulars is infinite

Thus they are characterized symbolically using **universal quantifiers:**

$$\cube_{\varphi} \equiv \left(\bigwedge_{x \in \mathcal{X}_{\varphi}} l_x \leqslant u_x \right) \wedge \forall \mathcal{X}_{\varphi} . \left(\bigwedge_{x \in \mathcal{X}_{\varphi}} l_x \leqslant x \leqslant u_x \right) \Rightarrow \varphi$$

Note how the formula defines fixed bounds in every dimension

Hyperrectangular Decomposition cont'd

A rectangle in R^2 can be specified as SMT formula:

$$\varphi \equiv 0 \leqslant x_1 \leqslant 100 \wedge 4 \leqslant x_2 \leqslant 10$$

The following holds: $\int_{\varphi} p_1(x_1)p_2(x_2)\, dx_1 dx_2$

$$=(\int_0^{100} p_1(x_1)\, dx_1)(\int_4^{10} p_2(x_2)\, dx_2)$$
$$=(F_1(10) - F_1(4))(F_2(100) - F_2(0))$$
$$= (Pr[x_2 \leq 10] - Pr[x_2 \leq 4])(Pr[x_1 \leq 100] - Pr[x_1 \leq 0])$$

In General = $\prod_{x_i \in \mathcal{X}_\varphi} \int_{H_l^m(x_i)}^{H_u^m(x_i)} p_i(x_i)\, dx_i$

"Independently compute the integral along each dimension and take the product"

Hyperrectangular Sampling

STATE <- (vol, remainder)

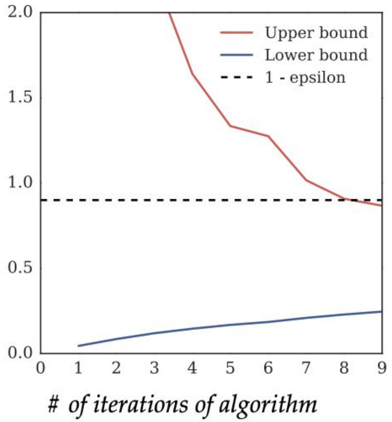
- Vol := current (aggregated) volume
- remainder :=remaining rectangles in the decomposition of **φ** (symbolic)

- Lower Bound

$$vol \leqslant \text{VOL}(\varphi, \mathcal{D}).$$

- Upper Bound

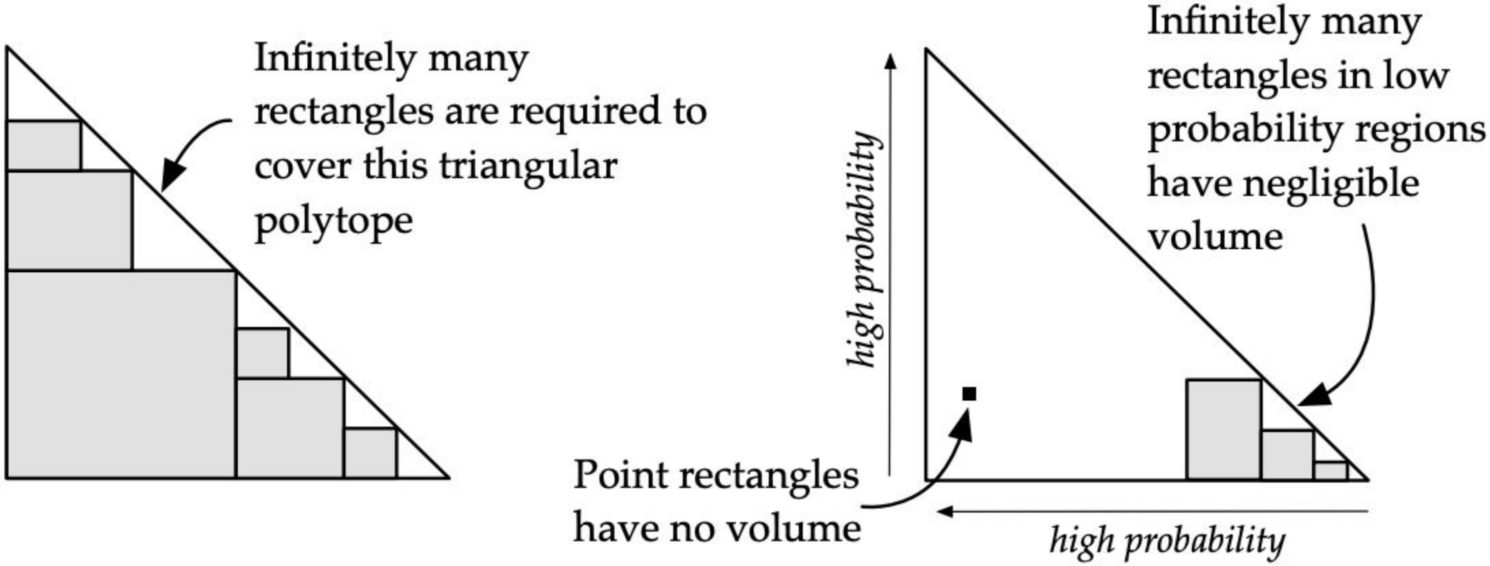
$$1 - vol \geqslant \text{VOL}(\varphi, \mathcal{D})$$




YES
ALL DONE!
memegenerator.net

Optimization

Convergence Guarantees



(a) (b)

Density Directed Sampling

So far, the algorithm provides no progress guarantees, when trying to find a model m of hyperrectangulars that yields the largest possible volume

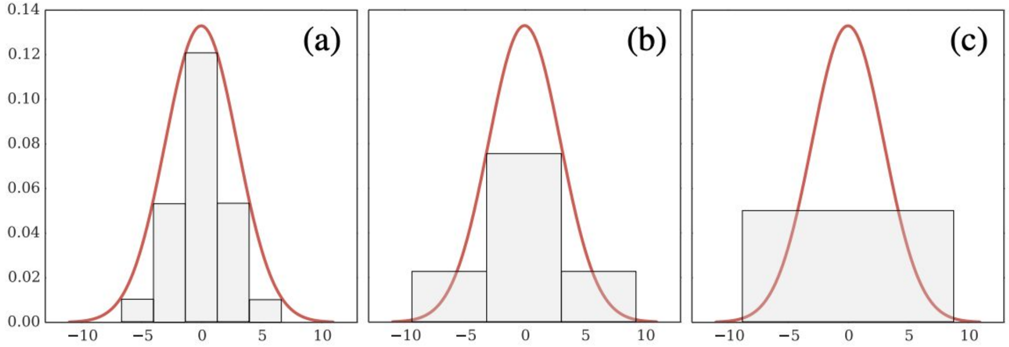
$$\underset{m \models \Psi}{\arg\max} \prod_{x_i \in \mathcal{X}_\varphi} \int_{H_l^m(x_i)}^{H_u^m(x_i)} p_i(x_i)\, dx_i$$

This is hard because … p is an arbitrary density function that cannot be integrated

But we can rethink p as a stepwise function!

ADF directed Volume Computation

Approximate each probability density function with a step function
Guide the sampling towards maximizing these (stepwise) hyperrectangulars



$$step^{\phi}(x) \equiv \delta_x = \sum_{i=1}^{n} c_i \cdot \left| [a_i, b_i) \cap [l_x, u_x] \right|$$

Find a hyperrectangle such that for each dimension x, δx is greater than some lower bound lb, shrink lb using a fixed decay rate

Sample Maximization

Further maximize the sampled hyperrectangular:

- ADF guides toward hyperrectangular with largest possible volume
- That target hyperrectangular gets further maximized
    - Greedily extend dimensions one at a time

Implementation

- Based on Z3 and RedLog (both exchangeable black boxes)
- Decompose conditional probabilities in to 4 (joint) probabilities
- Compute the weighted volume for each probability and its negation (8 total)
- Sample:
    - Obtain sample
    - Compute weighted volume
    - Update bounds
    - Check if bounds are precise enough for conclusion
    - Repeat

Results

Algorithm Evaluation

Decision program	**Acc**	Independent				**Population Model** Bayes Net 1				Bayes Net 2			
		Res	*#*	*Vol*	*QE*	*Res*	*#*	*Vol*	*QE*	*Res*	*#*	*Vol*	*QE*
DT_4	0.79	✓	10	1.3	0.5	✗	12	2.2	0.9	✗	18	6.6	2.2
DT_{14}	0.71	✓	20	4.2	1.4	✓	38	52.3	11.4	✓	73	130.9	33.6
DT_{16}	0.79	✓	21	7.7	2.0	✗	22	15.3	6.3	✗	22	38.2	14.3
$\mathrm{DT}_{16}^{\alpha}$	0.76	✓	18	5.1	3.0	✓	34	32.0	8.2	✓	40	91.0	19.4
DT_{44}	0.82	✓	55	63.5	9.8	✗	113	178.9	94.3	✗	406	484.0	222.4
SVM_3	0.79	✓	10	2.6	0.6	✗	10	3.7	1.7	✗	10	10.8	6.2
SVM_4	0.79	✓	10	2.7	0.8	✗	18	13.3	3.1	✗	14	33.7	20.1
SVM_4^{α}	0.78	✓	10	3.0	0.8	✓	22	15.7	3.2	✓	14	33.4	63.2
SVM_5	0.79	✓	10	8.5	1.3	✗	10	12.2	6.3	TO_q	-	-	TO
SVM_6	0.79	0.02 35.3	634	TO	2.4	0.09 3.03	434	TO	12.8	TO_q	-	-	TO
$\mathrm{NN}_{2,1}$	0.65	✓	78	21.6	0.8	✓	466	456.1	3.4	✓	154	132.9	7.2
$\mathrm{NN}_{2,2}$	0.67	✓	62	27.8	2.0	✓	238	236.5	7.2	✓	174	233.5	18.2
$\mathrm{NN}_{3,2}$	0.74	0.03 674.7	442	TO	10.0	0.00 5.24	34	TO	55.9	TO_q	-	-	TO

Thanks for listening

THOUGHTS

?