Explainable AI through Declarative Probabilistic Programming

Ouael Ben Amara[1] Sami Hadouaj[1] Niccolò Meneghetti[1]

[1]CIS Department
University of Michigan-Dearborn
benamara@umich.edu, shadouaj@umich.edu, niccolom@umich.edu

Introduction

Modern machine learning models, while powerful, often function as black boxes, making their decisions difficult to understand and validate. Current Explainable AI (XAI) approaches attempt to solve this by using surrogate models, but this creates a recursive problem: **how can we trust explanations from equally opaque systems?**

We present a novel framework that leverages declarative probabilistic programming to create transparent, mathematically rigorous explanations of machine learning decisions. Our approach combines De Finetti Logic with Datalog constraints [1], enabling the construction of interpretable surrogate models whose behavior can be formally verified. By encoding both domain expertise and classifier behavior through declarative constraints, our method produces explanations that are not only interpretable but also maintain mathematical rigor through principled uncertainty quantification. This bridges the gap between symbolic reasoning and probabilistic inference in XAI, offering a path toward more trustworthy and interpretable AI systems.

Background: Pólya die

To define a Pólya die $\mathcal{D}$, we adopt the following notational conventions:

1. We denote by $\boldsymbol{V} = \{v_1, \ldots, v_D\}$ a finite, discrete domain consisting of D distinct values
2. We denote by $\boldsymbol{\alpha}$ a D-dimensional vector of positive real numbers
3. We denote by $\mathcal{D}(\boldsymbol{\alpha})$ a Dirichlet density function, parametrized by vector $\boldsymbol{\alpha}$
4. We denote by $\boldsymbol{\theta}$ a D-dimensional random vector having density $\mathcal{D}(\boldsymbol{\alpha})$
5. We denote by $\mathcal{C}(\boldsymbol{\theta})$ a Categorical probability mass function parametrized by vector $\boldsymbol{\theta}$
6. We denote by $\boldsymbol{X}$ a collection $\{X[r_1], X[r_2], \ldots, X[r_J]\}$ of J discrete random variables, that all share the same domain $\boldsymbol{V}$ and are identically distributed as $\mathcal{C}(\boldsymbol{\theta})$

The joint distribution factorizes as:

$$p(\boldsymbol{\theta}, \boldsymbol{X} \mid \boldsymbol{\alpha}) = p(\boldsymbol{\theta} \mid \boldsymbol{\alpha}) \prod_{j=1}^{J} p(X[r_j] \mid \boldsymbol{\theta})$$

Motivation and Prior Work

Our work builds upon LIME's [4] foundational approach to model explanations while addressing its key limitation. LIME generates explanations by training surrogate models to approximate a black-box model's decision boundary in the neighborhood of specific predictions. While effective, this creates a recursive explainability challenge: the surrogate models themselves become black boxes that require explanation.

We address this fundamental issue by replacing LIME's learned surrogates with declarative probabilistic programs. By expressing the surrogate model through First-Order Logic and Pólya dice, we make both the model structure and the explanation process transparent. This combination of symbolic reasoning with probabilistic inference allows us to maintain LIME's local approximation capabilities while providing mathematical guarantees about the explanation mechanism itself.

Unlike LIME's opaque surrogate models, our declarative framework makes explicit how each component contributes to the final explanation, from the probabilistic sampling process to the constraint satisfaction steps. This transparency extends throughout the entire pipeline, ensuring that the explanation process itself requires no further explanation.

Stage 1: Color Quantization

The first stage simplifies the image representation through deterministic preprocessing (no Pólya dice):

- Convert RGB triplets (r, g, b) into single categorical values
- Map each pixel to an index in $\{0, \ldots, K-1\}$
- Maintain spatial relationships while reducing dimensionality

This basic transformation provides a discrete representation for subsequent probabilistic analysis.

Stage 2: Image LDA

We adapt text-based LDA to image segmentation through a careful mapping of concepts and a structured generative process:

1. Text-to-Image Correspondence
   - Document → Image patch: Each local region acts as a "document"
   - Word → Color index: Colors serve as our visual vocabulary
   - Topic → Segment: Topics become coherent image segments
   - Word position → Pixel location: Preserves spatial structure
2. Probabilistic Components
   - N_d red dice for patches, each with K_t faces representing possible segments
   - K_t blue dice for segments, each with K_c faces representing possible colors
   - Prior $\vec{\alpha}_d$ controls segment distribution within patches
   - Prior $\vec{\beta}$ ensures color consistency within segments
3. Generation Process
   - For each pixel position (d, p) with observed color w:
   - First throw the patch die associated with patch d to select segment t
   - Then throw the color die of selected segment t to generate color
   - Accept configuration when generated color matches observed pixel w

This process induces the conditional distribution:

$$P(z_{d,p} = t \mid z_{-(d,p)}, w) \propto \frac{n_{dt} + \alpha}{\sum_{t'}(n_{dt'} + \alpha)} \cdot \frac{n_{tw} + \beta}{\sum_{w'}(n_{tw'} + \beta)}$$

where n_{dt} counts segment assignments in patch d and n_{tw} tracks color occurrences in segment t, directly mirroring text LDA's document-topic and topic-word counts.

Stage 3: Spatial Smoothing

The final stage enforces spatial consistency using the same patch dice from Stage 2:

- For each pair of neighboring patches (d, n):
  - Roll the corresponding patch dice $\mathcal{D}_d$ and $\mathcal{D}_n$ from Stage 2
  - Constrain their segment assignments to be identical: $X[d] = X[n]$
- This enforces segment continuity between adjacent patches
- Process repeats proportionally to desired smoothing strength

This constraint-based approach ensures spatially coherent segmentation while maintaining the unified probabilistic framework established in Stage 2.

Declarative XAI Through DFL

We model explanations as a declarative probabilistic program where:

- Model decisions represented as (I, O) pairs
- Perturbations defined through Pólya dice $(\mathbb{H}, \Theta, P)$
- Domain knowledge encoded in Datalog constraints Φ

The joint distribution factorizes as:

$$p(\Theta, I, P, O, \Phi \mid \mathbb{H}) = p(I) \cdot p(\Theta \mid \mathbb{H}) \cdot p(P \mid \Theta) \cdot p(\Phi \mid I, P) \cdot p(O \mid I, P)$$

Generating Explainable Segments

Instead of operating on raw pixels, we use DFL to create meaningful segments:

1. Declarative Segmentation
   - Define segments through Pólya dice $\mathcal{D}_k$
   - Enforce spatial coherence via Datalog constraints
   - Ensure segments respect image structure

Explanation Generation

2. Probabilistic Analysis
   - Compute posterior over segments:
   - $$p(I, P \mid \mathbb{H}, O \neq d, \Phi = \top)$$
   - Preserve segment-level semantic meaning
3. Adaptive Learning
   - Update hyperparameters via expectation propagation:
   - $$\mathbb{H}^* = \min_{\mathbb{Z}} \mathbb{E}_{\Theta \sim p(\Theta \mid \hat{\Theta})} \left[\log \frac{p(\Theta \mid \hat{\Theta})}{q(\Theta \mid \mathbb{Z})} \right]$$
   - Balance local evidence with global constraints

Computing Segment Importance

We demonstrate our framework's systematic explanation process:

Figure 1. * Original Image

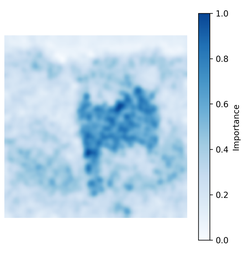

Figure 2. * Segment Importance

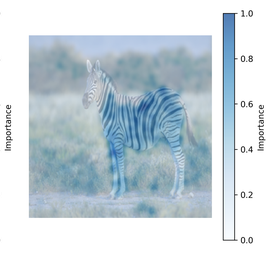

Figure 3. * Influential Regions

For each EP iteration:

1. Perturbation Sampling
   - Draw segment modifications from $p(P|\Theta)$
   - Apply white patch to selected segments
   - Generate 100 modified images per iteration
2. Importance analysis
   - Measure classification changes for each modification
   - Compute misclassification rates per segment
   - Update segment parameters via EP: $\mathbb{H}^* = \min_{\mathbb{Z}} \mathbb{E}[\log \frac{p(\Theta|\hat{\Theta})}{q(\Theta|\mathbb{Z})}]$

After 20 iterations, the heatmap highlights segments that most influence the model's decisions.

References

[1] Ouael Ben Amara, Sami Hadouaj, and Niccolò Meneghetti.
Starfishdb: A query execution engine for relational probabilistic programming.
Proc. ACM Manag. Data, 2(3):185, 2024.

[2] Vince Bárány, Balder ten Cate, Benny Kimelfeld, Dan Olteanu, and Zografoula Vagena.
Declarative probabilistic programming with datalog.
ACM Trans. Database Syst., 42(4):22:1–22:35, 2017.

[3] Niccolò Meneghetti and Ouael Ben Amara.
Gamma probabilistic databases: Learning from exchangeable query-answers.
In Julia Stoyanovich, Jens Teubner, Paolo Guagliardo, Milos Nikolic, Andreas Pieris, Jan Mühlig, Fatma Özcan, Sebastian Schelter, H. V. Jagadish, and Meihui Zhang, editors, *Proceedings of the 25th International Conference on Extending Database Technology, EDBT 2022, Edinburgh, UK, March 29 - April 1, 2022*, pages 2:260–2:273. OpenProceedings.org, 2022.

[4] Marco Túlio Ribeiro, Sameer Singh, and Carlos Guestrin.
"why should I trust you?": Explaining the predictions of any classifier.
In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144. ACM, 2016.