```
git clone https://github.com/zachfox/xai-causal-tutorial.git
```

Explainability and Causal AI

Zachary Fox and Ayana Ghosh

Objectives of this tutorial

- Build an understanding of explainability in machine learning
- Build an understanding of explainability in causal modeling
- Gain some hands-on experience with commonly used explainability tools
- Gain some hands-on experience with popular libraries for causal analysis

Part I: Explainability

What is Explainability?

Explainable AI (XAI) refers to a set of processes and methods that enable human users to understand, trust, and manage the outputs of artificial intelligence (AI) systems.

It aims to make the decision-making process of AI models **transparent** by providing clear and interpretable explanations of how models arrive at their predictions or decisions.

This involves generating insights into the internal workings of complex models, such as deep learning algorithms, and presenting these insights in a way that is accessible and meaningful to users, ranging from data scientists and engineers to end-users and regulators.

Explainable AI is essential for ensuring **accountability, fairness, and trustworthiness** in AI applications.

Common techniques for explainability

Model-agnostic methods

- LIME
- SHAP**

Model-dependent methods

- Decision trees
- Attention mechanisms

“Why Should I Trust You?” Explaining the Predictions of Any Classifier

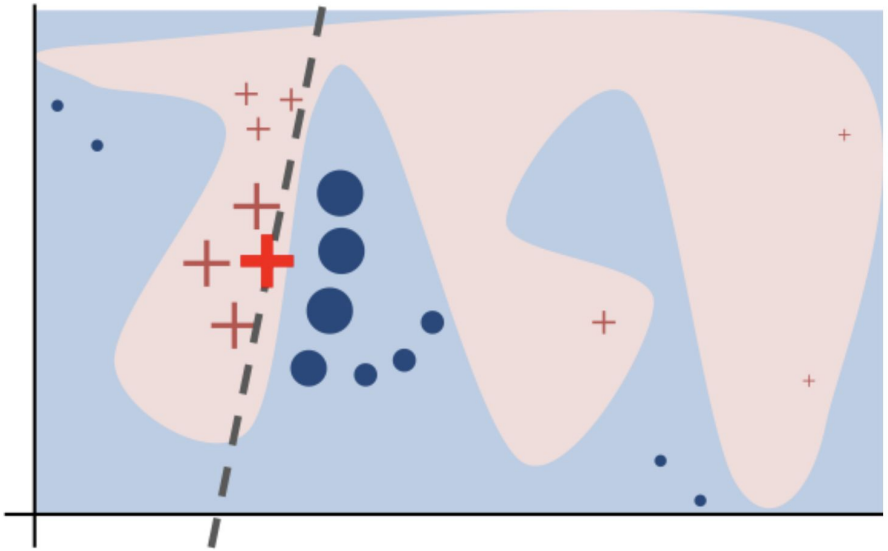
Marco Tulio Ribeiro
University of Washington
Seattle, WA 98105, USA
marcotcr@cs.uw.edu

Sameer Singh
University of Washington
Seattle, WA 98105, USA
sameer@cs.uw.edu

Carlos Guestrin
University of Washington
Seattle, WA 98105, USA
guestrin@cs.uw.edu

https://arxiv.org/abs/1602.04938

What does LIME (Local Interpretable Model-agnostic Explanation) do?



Given a classifier, *f,* and the instance that we are trying to explain +, how can we find a local, approximate model that is explainable?

https://arxiv.org/abs/1602.04938

Notebook 1: Understanding LIME

Summary of LIME

Pros

- Model-agnostic - easily applicable to any model
- Computationally efficient;

Cons

- Need to define the locality, which is a function of the data and model
- Sampling is challenging in high dimensional spaces

Tools

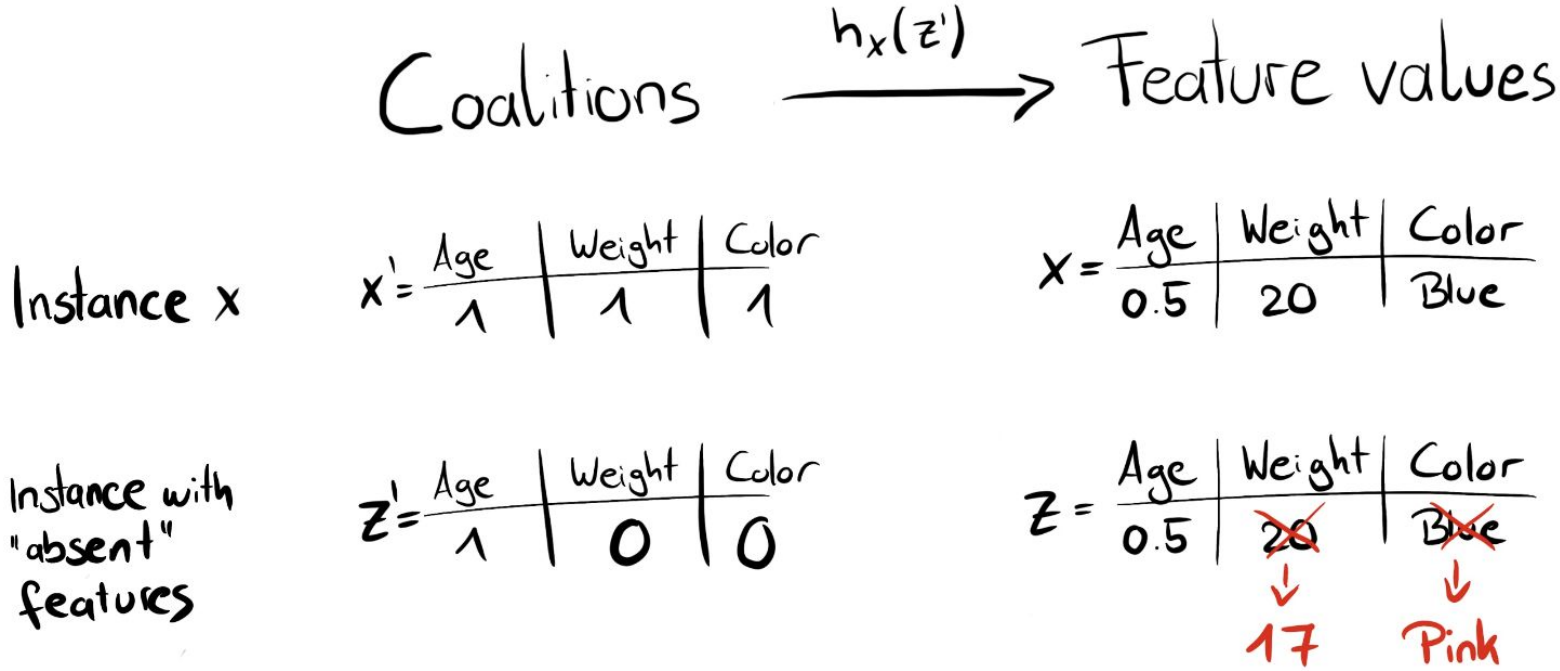
- Python lime package; also available in R

Background on SHAP and Shapley values

In the early 1950's Lloyd Shapley figured out a way to distribute the surplus, or profits, or winnings, amongst players of a cooperative game.

It turns out that this theory is very useful for assigning contributions to different features for complex machine learning algorithms.

An attempt at an intuitive explanation



Source: interpretable machine learning, chapter 9.6.

Shapley values are defined as

$$\phi_j(val) = \sum_{S \subseteq \{1,\ldots,p\} \setminus \{j\}} \frac{|S|!\,(p - |S| - 1)!}{p!} \left(val\left(S \cup \{j\}\right) - val(S)\right)$$

$$val_x(S) = \int \hat{f}\left(x_1, \ldots, x_p\right) d\mathbb{P}_{x \notin S} - E_X(\hat{f}\left(X\right))$$

Example:

$$val_x(S) = val_x(\{1,3\}) = \int_{\mathbb{R}} \int_{\mathbb{R}} \hat{f}\left(x_1, X_2, x_3, X_4\right) d\mathbb{P}_{X_2 X_4} - E_X(\hat{f}\left(X\right))$$

Linear models provide some intuition about Shapley values

$$\hat{f}(x) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

$$\phi_j(\hat{f}) = \beta_j x_j - E(\beta_j X_j)$$

$$\begin{aligned}
\sum_{j=1}^{p} \phi_j(\hat{f}) &= \sum_{j=1}^{p} (\beta_j x_j - E(\beta_j X_j)) \\
&= (\beta_0 + \sum_{j=1}^{p} \beta_j x_j) - (\beta_0 + \sum_{j=1}^{p} E(\beta_j X_j)) \\
&= \hat{f}(x) - E(\hat{f}(X))
\end{aligned}$$

Source: interpretable machine learning, chapter 9.5.

Background on SHAP and Shapley values

In order to connect game theory with machine learning models, it is necessary to both match a model's input features with players in a game, and also match the model function with the rules of the game

The value of the j-th feature contributed to the prediction of this particular instance compared to the average prediction for the dataset.

Source: interpretable machine learning, chapter 9.6.

From one point of view, SHAP is LIME with a different weighting scheme

$$g(z') = \phi_0 + \sum_{j=1}^{M} \phi_j z'_j \qquad \pi_x(z') = \frac{(M-1)}{\binom{M}{|z'|}|z'|(M-|z'|)}$$

$$L(\hat{f}, g, \pi_x) = \sum_{z' \in Z} [\hat{f}(h_x(z')) - g(z')]^2 \pi_x(z')$$

Notebook 2: Working with SHAP

Summary of SHAP

Pros

- Solid theoretical foundation
- Fast to compute for tree based models

Cons

- Slow to compute in general

Tools

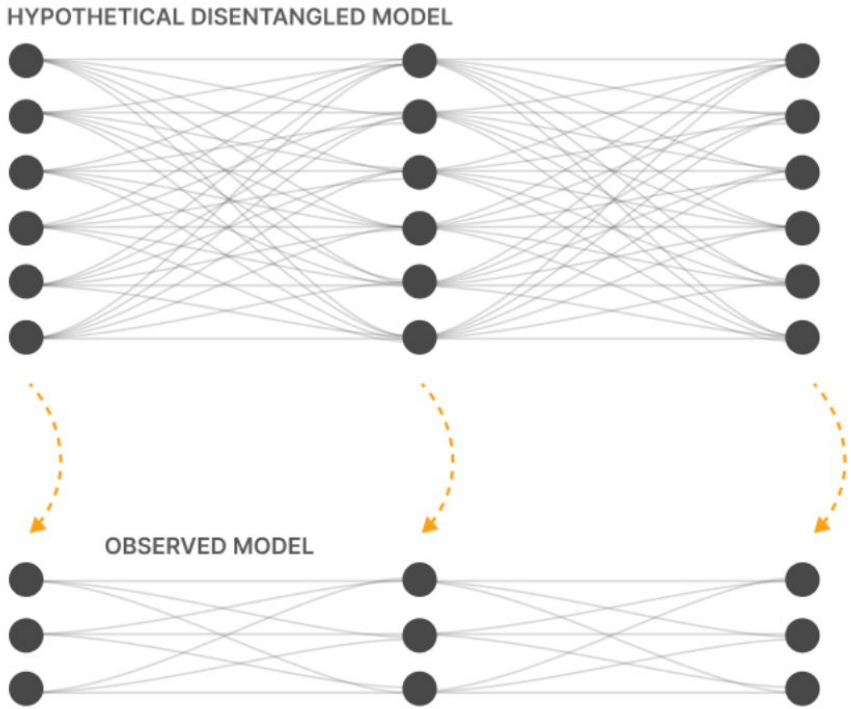
- Python SHAP package;

Anthropic’s monosemanticity

These units, called features, correspond to patterns (linear combinations) of neuron activations.

	Transformer	Sparse Autoencoder
Layers	1 Attention Block 1 MLP Block (ReLU)	1 ReLU (up) 1 Linear (down)
MLP Size	512	512 (1×) – 131,072 (256×)
Dataset	The Pile [19] (100 billion tokens)	Transformer MLP Activations (8 billion samples)
Loss	Autoregressive Log-Likelihood	L2 reconstruction + L1 on hidden layer activation

Superposition



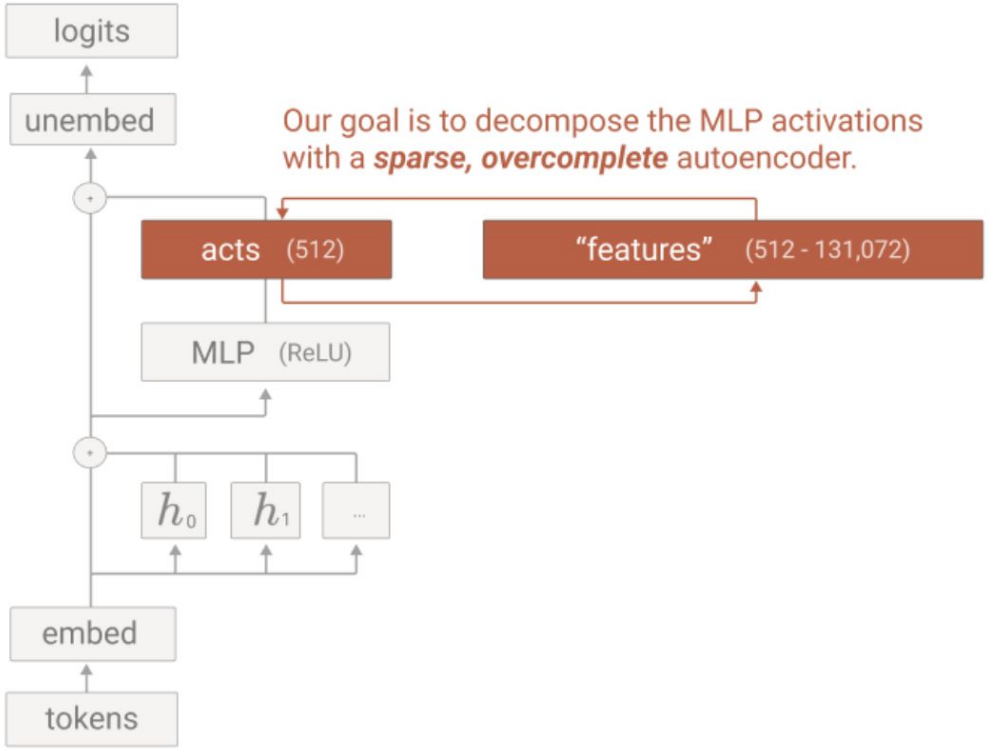
Under the superposition hypothesis, the neural networks we observe are **simulations of larger networks** where every neuron is a disentangled feature.

These idealized neurons are **projected** on to the actual network as “almost orthogonal” vectors over the neurons.

The network we observe is a **low-dimensional projection** of the larger network. From the perspective of individual neurons, this presents as polysemanticity.

From https://transformer-circuits.pub/2023/monosemantic-features/index.html

Training on activations to discover features



From https://transformer-circuits.pub/2023/monosemantic-features/index.html

Resources

Ribeiro, Marco Tulio, Sameer Singh, and Carlos Guestrin. “Why should I trust you?: Explaining the predictions of any classifier.” Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM (2016).

Alvarez-Melis, David, and Tommi S. Jaakkola. “On the robustness of interpretability methods.” arXiv preprint arXiv:1806.08049 (2018)

Slack, Dylan, Sophie Hilgard, Emily Jia, Sameer Singh, and Himabindu Lakkaraju. “Fooling lime and shap: Adversarial attacks on post hoc explanation methods.” In Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society, pp. 180-186 (2020).

Resources

https://transformer-circuits.pub/2023/monosemantic-features/index.html

https://shap.readthedocs.io/en/latest/example_notebooks/overviews/An%20introduction%20to%20explainable%20AI%20with%20Shapley%20values.html